Package 'broom'

September 13, 2025

Type Package

Title Convert Statistical Objects into Tidy Tibbles

Version 1.0.10

Description Summarizes key information about statistical objects in tidy tibbles. This makes it easy to report results, create plots and consistently work with large numbers of models at once. Broom provides three verbs that each provide different types of information about a model. tidy() summarizes information about model components such as coefficients of a regression. glance() reports information about an entire model, such as goodness of fit measures like AIC and BIC. augment() adds information about individual observations to a dataset, such as fitted values or influence measures.

License MIT + file LICENSE

URL https://broom.tidymodels.org/, https://github.com/tidymodels/broom

BugReports https://github.com/tidymodels/broom/issues

Depends R (>= 4.1)

Imports backports, cli, dplyr (>= 1.0.0), generics (>= 0.0.2), glue, lifecycle, purrr, rlang (>= 1.1.0), stringr, tibble (>= 3.0.0), tidyr (>= 1.0.0)

Suggests AER, AUC, bbmle, betareg (>= 3.2-1), biglm, binGroup, boot, btergm (>= 1.10.6), car (>= 3.1-2), carData, caret, cluster, cmprsk, coda, covr, drc, e1071, emmeans, epiR (>= 2.0.85), ergm (>= 3.10.4), fixest (>= 0.9.0), gam (>= 1.15), gee, geepack, ggplot2, glmnet, glmnetUtils, gmm, Hmisc, interp, irlba, joineRML, Kendall, knitr, ks, Lahman, lavaan (>= 0.6.18), leaps, lfe, lm.beta, lme4, lmodel2, lmtest (>= 0.9.38), lsmeans, maps, margins, MASS, mclust, mediation, metafor, mfx, mgcv, mlogit, modeldata, modeltests (>= 0.1.6), muhaz, multcomp, network, nnet, ordinal, plm, poLCA, psych, quantreg, rmarkdown, robust, robustbase, rsample, sandwich, spatialreg, spdep (>= 1.1), speedglm, spelling, stats4, survey, survival (>= 3.6-4), systemfit, testthat (>= 3.0.0), tseries, vars, zoo

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Config/usethis/last-upkeep 2025-04-25

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Collate 'aaa-documentation-helper.R' 'null-and-default.R' 'aer.R'
'auc.R' 'base.R' 'bbmle.R' 'betareg.R' 'biglm.R' 'bingroup.R'
'boot.R' 'broom-package.R' 'broom.R' 'btergm.R' 'car.R'
'caret.R' 'cluster.R' 'cmprsk.R' 'data-frame.R'
'deprecated-0-7-0.R' 'drc.R' 'emmeans.R' 'epiR.R' 'ergm.R'
'fixest.R' 'gam.R' 'geepack.R' 'glmnet-cv-glmnet.R'
'glmnet-glmnet.R' 'gmm.R' 'hmisc.R'
'import-standalone-obj-type.R'
'import-standalone-types-check.R' 'joinerml.R' 'kendall.R'
'ks.R' 'lavaan.R' 'leaps.R' 'lfe.R' 'list-irlba.R'
'list-optim.R' 'list-svd.R' 'list-xyz.R' 'list.R' 'lm-beta.R'
'lmodel2.R' 'lmtest.R' 'maps.R' 'margins.R' 'mass-fitdistr.R'
'mass-negbin.R' 'mass-polr.R' 'mass-ridgelm.R' 'stats-lm.R'
'mass-rlm.R' 'mclust.R' 'mediation.R' 'metafor.R' 'mfx.R'
'mgcv.R' 'mlogit.R' 'muhaz.R' 'multcomp.R' 'nnet.R' 'nobs.R'
'ordinal-clm.R' 'ordinal-clmm.R' 'plm.R' 'polca.R' 'psych.R'
'stats-nls.R' 'quantreg-nlrq.R' 'quantreg-rq.R'
'quantreg-rqs.R' 'robust-glmrob.R' 'robust-lmrob.R'
'robustbase-glmrob.R' 'robustbase-lmrob.R' 'sp.R' 'spdep.R'
'speedglm-speedglm.R' 'speedglm-speedlm.R' 'stats-anova.R'
'stats-arima.R' 'stats-decompose.R' 'stats-factanal.R'
'stats-glm.R' 'stats-htest.R' 'stats-kmeans.R' 'stats-loess.R'
'stats-mlm.R' 'stats-prcomp.R' 'stats-smooth.spline.R'
'stats-summary-lm.R' 'stats-time-series.R' 'survey.R'
'survival-aareg.R' 'survival-cch.R' 'survival-coxph.R'
'survival-pyears.R' 'survival-survdiff.R' 'survival-survexp.R'
'survival-survfit.R' 'survival-survreg.R' 'systemfit.R'
'tseries.R' 'utilities.R' 'vars.R' 'zoo.R' 'zzz.R'

NeedsCompilation no

Author David Robinson [aut],
Alex Hayes [aut] (ORCID: <https://orcid.org/0000-0002-4985-5160>),
Simon Couch [aut, cre] (ORCID: <https://orcid.org/0000-0001-5676-5107>),
Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>),
Indrajeet Patil [ctb] (ORCID: <https://orcid.org/0000-0003-1995-6531>),
Derek Chiu [ctb],
Matthieu Gomez [ctb],
Boris Demeshev [ctb],
Dieter Menne [ctb],
Benjamin Nutter [ctb],
Luke Johnston [ctb],

Ben Bolker [ctb],
Francois Briatte [ctb],
Jeffrey Arnold [ctb],
Jonah Gabry [ctb],
Luciano Selzer [ctb],
Gavin Simpson [ctb],
Jens Preussner [ctb],
Jay Hesselberth [ctb],
Hadley Wickham [ctb],
Matthew Lincoln [ctb],
Alessandro Gasparini [ctb],
Lukasz Komsta [ctb],
Frederick Novometsky [ctb],
Wilson Freitas [ctb],
Michelle Evans [ctb],
Jason Cory Brunson [ctb],
Simon Jackson [ctb],
Ben Whalley [ctb],
Karissa Whiting [ctb],
Yves Rosseel [ctb],
Michael Kuehn [ctb],
Jorge Cimentada [ctb],
Erle Holgersen [ctb],
Karl Dunkle Werner [ctb] (ORCID:
<https://orcid.org/0000-0003-0523-7309>),
Ethan Christensen [ctb],
Steven Pav [ctb],
Paul PJ [ctb],
Ben Schneider [ctb],
Patrick Kennedy [ctb],
Lily Medina [ctb],
Brian Fannin [ctb],
Jason Muhlenkamp [ctb],
Matt Lehman [ctb],
Bill Denney [ctb] (ORCID: <https://orcid.org/0000-0002-5759-428X>),
Nic Crane [ctb],
Andrew Bates [ctb],
Vincent Arel-Bundock [ctb] (ORCID:
<https://orcid.org/0000-0003-2042-7063>),
Hideaki Hayashi [ctb],
Luis Tobalina [ctb],
Annie Wang [ctb],
Wei Yang Tham [ctb],
Clara Wang [ctb],
Abby Smith [ctb] (ORCID: <https://orcid.org/0000-0002-3207-0375>),
Jasper Cooper [ctb] (ORCID: <https://orcid.org/0000-0002-8639-3188>),
E Auden Krauska [ctb] (ORCID: <https://orcid.org/0000-0002-1466-5850>),
Alex Wang [ctb],

Malcolm Barrett [ctb] (ORCID: <https://orcid.org/0000-0003-0299-5825>),
Charles Gray [ctb] (ORCID: <https://orcid.org/0000-0002-9978-011X>),
Jared Wilber [ctb],
Vilmantas Gegzna [ctb] (ORCID: <https://orcid.org/0000-0002-9500-5167>),
Eduard Szoecs [ctb],
Frederik Aust [ctb] (ORCID: <https://orcid.org/0000-0003-4900-788X>),
Angus Moore [ctb],
Nick Williams [ctb],
Marius Barth [ctb] (ORCID: <https://orcid.org/0000-0002-3421-6665>),
Bruna Wundervald [ctb] (ORCID: <https://orcid.org/0000-0001-8163-220X>),
Joyce Cahoon [ctb] (ORCID: <https://orcid.org/0000-0001-7217-4702>),
Grant McDermott [ctb] (ORCID: <https://orcid.org/0000-0001-7883-8573>),
Kevin Zarca [ctb],
Shiro Kuriwaki [ctb] (ORCID: <https://orcid.org/0000-0002-5687-2647>),
Lukas Wallrich [ctb] (ORCID: <https://orcid.org/0000-0003-2121-5177>),
James Martherus [ctb] (ORCID: <https://orcid.org/0000-0002-8285-3300>),
Chuliang Xiao [ctb] (ORCID: <https://orcid.org/0000-0002-8466-9398>),
Joseph Larmarange [ctb],
Max Kuhn [ctb],
Michal Bojanowski [ctb],
Hakon Malmedal [ctb],
Clara Wang [ctb],
Sergio Oller [ctb],
Luke Sonnet [ctb],
Jim Hester [ctb],
Ben Schneider [ctb],
Bernie Gray [ctb] (ORCID: <https://orcid.org/0000-0001-9190-6032>),
Mara Averick [ctb],
Aaron Jacobs [ctb],
Andreas Bender [ctb],
Sven Templer [ctb],
Paul-Christian Buerkner [ctb],
Matthew Kay [ctb],
Erwan Le Pennec [ctb],
Johan Junkka [ctb],
Hao Zhu [ctb],
Benjamin Soltoff [ctb],
Zoe Wilkinson Saldana [ctb],
Tyler Littlefield [ctb],
Charles T. Gray [ctb],
Shabbh E. Banks [ctb],
Serina Robinson [ctb],
Roger Bivand [ctb],
Riinu Ots [ctb],
Nicholas Williams [ctb],
Nina Jakobsen [ctb],
Michael Weylandt [ctb],
Lisa Lendway [ctb],

Karl Hailperin [ctb],
Josue Rodriguez [ctb],
Jenny Bryan [ctb],
Chris Jarvis [ctb],
Greg Macfarlane [ctb],
Brian Mannakee [ctb],
Drew Tyre [ctb],
Shreyas Singh [ctb],
Laurens Geffert [ctb],
Hong Ooi [ctb],
Henrik Bengtsson [ctb],
Eduard Szocs [ctb],
David Hugh-Jones [ctb],
Matthieu Stigler [ctb],
Hugo Tavares [ctb] (ORCID: <https://orcid.org/0000-0001-9373-2726>),
R. Willem Vervoort [ctb],
Brenton M. Wiernik [ctb],
Josh Yamamoto [ctb],
Jasme Lee [ctb],
Taren Sanders [ctb] (ORCID: <https://orcid.org/0000-0002-4504-6008>),
Ilaria Prosdocimi [ctb] (ORCID:
<https://orcid.org/0000-0001-8565-094X>),
Daniel D. Sjoberg [ctb] (ORCID:
<https://orcid.org/0000-0003-0862-2018>),
Alex Reinhart [ctb] (ORCID: <https://orcid.org/0000-0002-6658-514X>)

Maintainer Simon Couch <simon.couch@posit.co>

Repository CRAN

Date/Publication 2025-09-13 05:11:00 UTC

Contents

`augment.betamfx` *Augment data with information from a(n) betamfx object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'betamfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("response", "link", "precision", "variance", "quantile"),
 type.residuals = c("sweighted2", "deviance", "pearson", "response", "weighted",
   "sweighted"),
  ...
)
```

Arguments

`x`	A `betamfx` object.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`type.predict`	Character indicating type of prediction to use. Passed to the `type` argument of `betareg::predict.betareg()`. Defaults to `"response"`.
`type.residuals`	Character indicating type of residuals to use. Passed to the `type` argument of `betareg::residuals.betareg()`. Defaults to `"sweighted2`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.

- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

This augment method wraps `augment.betareg()` for `mfx::betamfx()` objects.

Value

A `tibble::tibble()` with columns:

`.cooksd`	Cooks distance.
`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`augment.betareg()`, `mfx::betamfx()`

Other mfx tidiers: `augment.mfx()`, `glance.betamfx()`, `glance.mfx()`, `tidy.betamfx()`, `tidy.mfx()`

Examples

```
library(mfx)

# Simulate some data
set.seed(12345)
n <- 1000
x <- rnorm(n)

# Beta outcome
y <- rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2 * x)))
# Use Smithson and Verkuilen correction
y <- (y * (n - 1) + 0.5) / n

d <- data.frame(y, x)
mod_betamfx <- betamfx(y ~ x | x, data = d)

tidy(mod_betamfx, conf.int = TRUE)

# Compare with the naive model coefficients of the equivalent betareg call (not run)
# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)
glance(mod_betamfx)
```

`augment.betareg` *Augment data with information from a(n) betareg object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'betareg'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict,
  type.residuals,
  ...
)
```

Arguments

`x`	A `betareg` object produced by a call to `betareg::betareg()`.

data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals	Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

For additional details on Cook's distance, see stats::cooks.distance().

Value

A tibble::tibble() with columns:

.cooksd	Cooks distance.
.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.

See Also

augment(), betareg::betareg()

Examples

```
# load libraries for models and data
library(betareg)

# load dats
```

```
data("GasolineYield", package = "betareg")

# fit model
mod <- betareg(yield ~ batch + temp, data = GasolineYield)

mod

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)

augment(mod)

glance(mod)
```

`augment.clm` *Augment data with information from a(n) clm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'clm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("prob", "class"),
  ...
)
```

Arguments

x	A clm object returned from ordinal::clm().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Which type of prediction to compute, either "prob" or "class", passed to ordinal::predict.clm(). Defaults to "prob".
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

See Also

tidy, ordinal::clm(), ordinal::predict.clm()

Other ordinal tidiers: augment.polr(), glance.clm(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(ordinal)

# fit model
```

```
fit <- clm(rating ~ temp * contact, data = wine)

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

# ...and again with another model specification
fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)

tidy(fit2)
glance(fit2)
```

`augment.coxph` *Augment data with information from a(n) coxph object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'coxph'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = "lp",
  type.residuals = "martingale",
  ...
)
```

Arguments

x	A coxph object returned from survival::coxph().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals	Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

When the modeling was performed with na.action = "na.omit" (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with na.action = "na.exclude", one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to augment() and na.action = "na.exclude", a warning is raised and the incomplete rows are dropped.

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.

See Also

stats::na.action

augment(), survival::coxph()

Other coxph tidiers: glance.coxph(), tidy.coxph()

Other survival tidiers: augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
cfit <- coxph(Surv(time, status) ~ age + sex, lung)

# summarize model fit with tidiers
tidy(cfit)
tidy(cfit, exponentiate = TRUE)

lp <- augment(cfit, lung)
risks <- augment(cfit, lung, type.predict = "risk")
expected <- augment(cfit, lung, type.predict = "expected")

glance(cfit)

# also works on clogit models
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(
  logan[indx, ],
  id = indx,
  tocc = factor(rep(resp, each = n))
)

logan2$case <- (logan2$occupation == logan2$tocc)

cl <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
```

```
tidy(cl)
glance(cl)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(risks, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(expected, aes(time, .fitted, color = sex)) +
  geom_point()
```

`augment.decomposed.ts` *Augment data with information from a(n) decomposed.ts object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'decomposed.ts'
augment(x, ...)
```

Arguments

x	A decomposed.ts object returned from stats::decompose().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble with one row for each observation in the original times series:

.seasonal	The seasonal component of the decomposition.
.trend	The trend component of the decomposition.
.remainder	The remainder, or "random" component of the decomposition.
.weight	The final robust weights (stl only).
.seasadj	The seasonally adjusted (or "deseasonalised") series.

See Also

augment(), stats::decompose()

Other decompose tidiers: augment.stl()

Examples

```
# time series of temperatures in Nottingham, 1920-1939:
nottem

# perform seasonal decomposition on the data with both decompose
# and stl:
d1 <- decompose(nottem)
d2 <- stl(nottem, s.window = "periodic", robust = TRUE)

# compare the original series to its decompositions.

cbind(
  tidy(nottem), augment(d1),
  augment(d2)
)

# visually compare seasonal decompositions in tidy data frames.

library(tibble)
```

```
library(dplyr)
library(tidyr)
library(ggplot2)

decomps <- tibble(
  # turn the ts objects into data frames.
  series = list(as.data.frame(nottem), as.data.frame(nottem)),
  # add the models in, one for each row.
  decomp = c("decompose", "stl"),
  model = list(d1, d2)
) |>
  rowwise() |>
  # pull out the fitted data using broom::augment.
  mutate(augment = list(broom::augment(model))) |>
  ungroup() |>
  # unnest the data frames into a tidy arrangement of
  # the series next to its seasonal decomposition, grouped
  # by the method (stl or decompose).
  group_by(decomp) |>
  unnest(c(series, augment)) |>
  mutate(index = 1:n()) |>
  ungroup() |>
  select(decomp, index, x, adjusted = .seasadj)

ggplot(decomps) +
  geom_line(aes(x = index, y = x), colour = "black") +
  geom_line(aes(
    x = index, y = adjusted, colour = decomp,
    group = decomp
  ))
```

`augment.drc` *Augment data with information from a(n) drc object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'drc'
augment(
  x,
  data = NULL,
  newdata = NULL,
  se_fit = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  ...
)
```

Arguments

`x`	A `drc` object produced by a call to `drc::drm()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`se_fit`	Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.cooksd`	Cooks distance.
`.fitted`	Fitted or predicted value.
`.lower`	Lower bound on interval for fitted values.
`.resid`	The difference between observed and fitted values.
`.se.fit`	Standard errors of fitted values.
`.upper`	Upper bound on interval for fitted values.

See Also

`augment()`, `drc::drm()`

Other drc tidiers: `glance.drc()`, `tidy.drc()`

Examples

```
# load libraries for models and data
library(drc)

# fit model
mod <- drm(dead / total ~ conc, type,
  weights = total, data = selenium, fct = LL.2(), type = "binomial"
)

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)

glance(mod)

augment(mod, selenium)
```

`augment.factanal` *Augment data with information from a(n) factanal object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'factanal'
augment(x, data, ...)
```

Arguments

`x`	A `factanal` object created by `stats::factanal()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

When `data` is not supplied `augment.factanal` returns one row for each observation, with a factor score column added for each factor X, (`.fsX`). This is because `stats::factanal()`, unlike other stats methods like `stats::lm()`, does not retain the original data.

When `data` is supplied, `augment.factanal` returns one row for each observation, with a factor score column added for each factor X, (`.fsX`).

See Also

`augment()`, `stats::factanal()`

Other factanal tidiers: `glance.factanal()`, `tidy.factanal()`

`augment.felm` *Augment data with information from a(n) felm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'felm'
augment(x, data = model.frame(x), ...)
```

Arguments

`x`	A `felm` object returned from `lfe::felm()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`augment()`, `lfe::felm()`

Other felm tidiers: `tidy.felm()`

Examples

```
# load libraries for models and data
library(lfe)

# use built-in `airquality` dataset
head(airquality)

# no FEs; same as lm()
```

```
est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)

# summarize model fit with tidiers
tidy(est0)
augment(est0)

# add month fixed effects
est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)

# summarize model fit with tidiers
tidy(est1)
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# the "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)

# summarize model fit with tidiers
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")
```

`augment.fixest` *Augment data with information from a(n) fixest object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses splines::ns(), stats::poly(), or survival::Surv(), it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'fixest'
augment(
  x,
  data = NULL,
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("response", "deviance", "pearson", "working"),
  ...
)
```

Arguments

x	A fixest object returned from any of the fixest estimators
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Passed to predict.fixest type argument. Defaults to "link" (like predict.glm).
type.residuals	Passed to predict.fixest type argument. Defaults to "response" (like residuals.lm, but unlike residuals.glm).
...	Additional arguments passed to summary and confint. Important arguments are se and cluster. Other arguments are dof, exact_dof, forceCovariance, and keepBounded. See summary.fixest.

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.

Note

Important note: `fixest` models do not include a copy of the input data, so you must provide it manually.

augment.fixest only works for `fixest::feols()`, `fixest::feglm()`, and `fixest::femlm()` models. It does not work with results from `fixest::fenegbin()`, `fixest::feNmlm()`, or `fixest::fepois()`.

See Also

`augment()`, `fixest::feglm()`, `fixest::femlm()`, `fixest::feols()`

Other fixest tidiers: `tidy.fixest()`

Examples

```
# load libraries for models and data
library(fixest)

gravity <-
  feols(
    log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade
  )

tidy(gravity)
glance(gravity)
augment(gravity, trade)

# to get robust or clustered SEs, users can either:

# 1) specify the arguments directly in the `tidy()` call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))

tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) or, feed tidy() a summary.fixest object that has already accepted
# these arguments

gravity_summ <- summary(gravity, cluster = c("Product", "Year"))

tidy(gravity_summ, conf.int = TRUE)

# approach (1) is preferred.
```

`augment.gam` *Augment data with information from a(n) gam object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'gam'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict,
  type.residuals,
  ...
)
```

Arguments

x	A gam object returned from a call to `mgcv::gam()`.
data	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.

newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals	Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

For additional details on Cook's distance, see stats::cooks.distance().

Value

A tibble::tibble() with columns:

.cooksd	Cooks distance.
.fitted	Fitted or predicted value.
.hat	Diagonal of the hat matrix.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.
.sigma	Estimated residual standard deviation when corresponding observation is dropped from model.

See Also

augment(), mgcv::gam()

Examples

```
# load libraries for models and data
library(mgcv)

# fit model
g <- gam(mpg ~ s(hp) + am + qsec, data = mtcars)
```

```
# summarize model fit with tidiers
tidy(g)
tidy(g, parametric = TRUE)
glance(g)
augment(g)
```

`augment.glm` *Augment data with information from a(n) glm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'glm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
```

```
  ...
)
```

Arguments

x	A glm object returned from stats::glm().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Passed to stats::predict.glm() type argument. Defaults to "link".
type.residuals	Passed to stats::residuals.glm() and to stats::rstandard.glm() type arguments. Defaults to "deviance".
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

If the weights for any of the observations in the model are 0, then columns ".infl" and ".hat" in the result will be 0 for those observations.

A .resid column is not calculated when data is specified via the newdata argument.

Value

A tibble::tibble() with columns:

.cooksd	Cooks distance.
.fitted	Fitted or predicted value.
.hat	Diagonal of the hat matrix.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.

.sigma	Estimated residual standard deviation when corresponding observation is dropped from model.
.std.resid	Standardised residuals.

See Also

stats::glm()

Other lm tidiers: augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

---

augment.glmRob *Augment data with information from a(n) glmRob object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses splines::ns(), stats::poly(), or survival::Surv(), it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'glmRob'
augment(x, ...)
```

Arguments

x	Unused.
...	Unused.

augment.glmrob *Augment data with information from a(n) glmrob object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'glmrob'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)
```

Arguments

x	A glmrob object returned from `robustbase::glmrob()`.

data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals	Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see tidy.rlm().

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.

See Also

robustbase::glmrob()

Other robustbase tidiers: augment.lmrob(), glance.lmrob(), tidy.glmrob(), tidy.lmrob()

Examples

```
if (requireNamespace("robustbase", quietly = TRUE)) {
  # load libraries for models and data
  library(robustbase)

  data(coleman)
  set.seed(0)

  m <- lmrob(Y ~ ., data = coleman)
  tidy(m)
  augment(m)
  glance(m)

  data(carrots)

  Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
    family = binomial, data = carrots, method = "Mqle",
    control = glmrobMqle.control(tcc = 1.2)
  )

  tidy(Rfit)
  augment(Rfit)
}
```

`augment.htest` *Augment data with information from a(n) htest object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters

the model as part of a matrix of covariates, such as when the model formula uses splines::ns(), stats::poly(), or survival::Surv(), it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'htest'
augment(x, ...)
```

Arguments

x	An htest objected, such as those created by stats::cor.test(), stats::t.test(), stats::wilcox.test(), stats::chisq.test(), etc.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

See stats::chisq.test() for more details on how residuals are computed.

Value

A tibble::tibble() with exactly one row and columns:

.observed	Observed count.
.prop	Proportion of the total.
.row.prop	Row proportion (2 dimensions table only).
.col.prop	Column proportion (2 dimensions table only).
.expected	Expected count under the null hypothesis.
.resid	Pearson residuals.
.std.resid	Standardized residual.

See Also

augment(), stats::chisq.test()

Other htest tidiers: tidy.htest(), tidy.pairwise.htest(), tidy.power.htest()

Examples

```
tt <- t.test(rnorm(10))

tidy(tt)

# the glance output will be the same for each of the below tests
glance(tt)

tt <- t.test(mpg ~ am, data = mtcars)

tidy(tt)

wt <- wilcox.test(mpg ~ am, data = mtcars, conf.int = TRUE, exact = FALSE)

tidy(wt)

ct <- cor.test(mtcars$wt, mtcars$mpg)

tidy(ct)

chit <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))

tidy(chit)
augment(chit)
```

`augment.ivreg` *Augment data with information from a(n) ivreg object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'ivreg'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

`x`	An `ivreg` object created by a call to `AER::ivreg()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

This tidier currently only supports `ivreg`-classed objects outputted by the `AER` package. The `ivreg` package also outputs objects of class `ivreg`, and will be supported in a later release.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

augment(), AER::ivreg()

Other ivreg tidiers: glance.ivreg(), tidy.ivreg()

Examples

```
# load libraries for models and data
library(AER)

# load data
data("CigarettesSW", package = "AER")

# fit model
ivr <- ivreg(
  log(packs) ~ income | population,
  data = CigarettesSW,
  subset = year == "1995"
)

# summarize model fit with tidiers
tidy(ivr)
tidy(ivr, conf.int = TRUE)
tidy(ivr, conf.int = TRUE, instruments = TRUE)

augment(ivr)
augment(ivr, data = CigarettesSW)
augment(ivr, newdata = CigarettesSW)

glance(ivr)
```

---

augment.kmeans *Augment data with information from a(n) kmeans object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'kmeans'
augment(x, data, ...)
```

Arguments

`x`	A `kmeans` object created by `stats::kmeans()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.cluster`	Cluster assignment.

See Also

`augment()`, `stats::kmeans()`

Other kmeans tidiers: `glance.kmeans()`, `tidy.kmeans()`

Examples

```
library(cluster)
library(modeldata)
library(dplyr)

data(hpc_data)

x <- hpc_data[, 2:5]

fit <- pam(x, k = 4)

tidy(fit)
glance(fit)
augment(fit, x)
```

`augment.lm` *Augment data with information from a(n) lm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'lm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  se_fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  conf.level = 0.95,
  ...
)
```

Arguments

x	An lm object created by stats::lm().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
interval	Character indicating the type of confidence interval columns to be added to the augmented output. Passed on to predict() and defaults to "none".
conf.level	The confidence level to use for the interval created if interval is "confidence" or "prediction". Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence/prediction interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

When the modeling was performed with na.action = "na.omit" (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with na.action = "na.exclude", one should provide the original data as a

second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to augment() and na.action = "na.exclude", a warning is raised and the incomplete rows are dropped.

Some unusual lm objects, such as rlm from MASS, may omit .cooksd and .std.resid. gam from mgcv omits .sigma.

When newdata is supplied, only returns .fitted, .resid and .se.fit columns.

Value

A tibble::tibble() with columns:

.cooksd	Cooks distance.
.fitted	Fitted or predicted value.
.hat	Diagonal of the hat matrix.
.lower	Lower bound on interval for fitted values.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.
.sigma	Estimated residual standard deviation when corresponding observation is dropped from model.
.std.resid	Standardised residuals.
.upper	Upper bound on interval for fitted values.

See Also

stats::na.action

augment(), stats::predict.lm()

Other lm tidiers: augment.glm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

Examples

```
library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)

tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)

ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()
```

```
# aside: There are tidy() and glance() methods for lm.summary objects too.
# this can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars |>
  head(6) |>
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval

# simpler bivariate model since we're plotting in 2D
mod2 <- lm(mpg ~ wt, data = mtcars)

au <- augment(mod2, newdata = newdata, interval = "prediction")

ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata |>
  select(-mpg)

augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)

ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
```

```
b <- a + rnorm(length(a))
result <- lm(b ~ a)

tidy(result)
```

---

augment.lmRob *Augment data with information from a(n) lmRob object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'lmRob'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

`x`	A `lmRob` object returned from `robust::lmRob()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and

cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata — A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

... — Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- tidy() methods will warn when supplied an exponentiate argument if it will be ignored.
- augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see tidy.rlm().

See Also

robust::lmRob()

Other robust tidiers: glance.glmRob(), glance.lmRob(), tidy.glmRob(), tidy.lmRob()

Examples

```
# load modeling library
library(robust)

# fit model
m <- lmRob(mpg ~ wt, data = mtcars)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

---

augment.lmrob *Augment data with information from a(n) lmrob object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'lmrob'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)
```

Arguments

`x`	A `lmrob` object returned from `robustbase::lmrob()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`se_fit`	Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`robustbase::lmrob()`

Other robustbase tidiers: `augment.glmrob()`, `glance.lmrob()`, `tidy.glmrob()`, `tidy.lmrob()`

Examples

```
if (requireNamespace("robustbase", quietly = TRUE)) {
  # load libraries for models and data
  library(robustbase)

  data(coleman)
  set.seed(0)

  m <- lmrob(Y ~ ., data = coleman)
  tidy(m)
  augment(m)
  glance(m)

  data(carrots)

  Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
    family = binomial, data = carrots, method = "Mqle",
    control = glmrobMqle.control(tcc = 1.2)
  )

  tidy(Rfit)
  augment(Rfit)
}
```

augment.loess *Tidy a(n) loess object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'loess'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)
```

Arguments

x	A loess objects returned by stats::loess().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

When the modeling was performed with na.action = "na.omit" (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with na.action = "na.exclude", one should provide the original data as a

second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to augment() and na.action = "na.exclude", a warning is raised and the incomplete rows are dropped.

Note that loess objects by default will not predict on data outside of a bounding hypercube defined by the training data unless the original loess object was fit with control = loess.control(surface = \"direct\")). See stats::predict.loess() for details.

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.

See Also

stats::na.action

augment(), stats::loess(), stats::predict.loess()

Examples

```
lo <- loess(
  mpg ~ hp + wt,
  mtcars,
  control = loess.control(surface = "direct")
)

augment(lo)

# with all columns of original data
augment(lo, mtcars)

# with a new dataset
augment(lo, newdata = head(mtcars))
```

---

augment.Mclust *Augment data with information from a(n) Mclust object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object.

Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'Mclust'
augment(x, data = NULL, ...)
```

Arguments

`x`	An `Mclust` object return from `mclust::Mclust()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.class`	Predicted class.
`.uncertainty`	The uncertainty associated with the classification. Equal to one minus the model class probability.

See Also

augment(), mclust::Mclust()

Other mclust tidiers: tidy.Mclust()

Examples

```
# load library for models and data
library(mclust)

# load data manipulation libraries
library(dplyr)
library(tibble)
library(purrr)
library(tidyr)

set.seed(27)

centers <- tibble(
  cluster = factor(1:3),
  # number points in each cluster
  num_points = c(100, 150, 50),
  # x1 coordinate of cluster center
  x1 = c(5, 0, -3),
  # x2 coordinate of cluster center
  x2 = c(-1, 1, -2)
)

points <- centers |>
  mutate(
    x1 = map2(num_points, x1, rnorm),
    x2 = map2(num_points, x2, rnorm)
  ) |>
  select(-num_points, -cluster) |>
  unnest(c(x1, x2))

# fit model
m <- Mclust(points)

# summarize model fit with tidiers
tidy(m)
augment(m, points)
glance(m)
```

---

augment.mfx *Augment data with information from a(n) mfx object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'mfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)

## S3 method for class 'logitmfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)
```

```
## S3 method for class 'negbinmfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)

## S3 method for class 'poissonmfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)

## S3 method for class 'probitmfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se_fit = FALSE,
  ...
)
```

Arguments

x	A logitmfx, negbinmfx, poissonmfx, or probitmfx object. (Note that betamfx objects receive their own set of tidiers.)
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Passed to stats::predict.glm() type argument. Defaults to "link".

type.residuals	Passed to stats::residuals.glm() and to stats::rstandard.glm() type arguments. Defaults to "deviance".
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

This generic augment method wraps augment.glm() for applicable objects from the mfx package.

Value

A tibble::tibble() with columns:

.cooksd	Cooks distance.
.fitted	Fitted or predicted value.
.hat	Diagonal of the hat matrix.
.resid	The difference between observed and fitted values.
.se.fit	Standard errors of fitted values.
.sigma	Estimated residual standard deviation when corresponding observation is dropped from model.
.std.resid	Standardised residuals.

See Also

augment.glm(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx()

Other mfx tidiers: augment.betamfx(), glance.betamfx(), glance.mfx(), tidy.betamfx(), tidy.mfx()

Examples

```
# load libraries for models and data
library(mfx)

# get the marginal effects from a logit regression
mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
```

```
tidy(mod_logmfx, conf.int = TRUE)

# compare with the naive model coefficients of the same logit call
tidy(
  glm(am ~ cyl + hp + wt, family = binomial, data = mtcars),
  conf.int = TRUE
)

augment(mod_logmfx)
glance(mod_logmfx)

# another example, this time using probit regression
mod_probmfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)

tidy(mod_probmfx, conf.int = TRUE)
augment(mod_probmfx)
glance(mod_probmfx)
```

`augment.mjoint` *Augment data with information from a(n) mjoint object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'mjoint'
augment(x, data = x$data, ...)
```

Arguments

x	An mjoint object returned from joineRML::mjoint().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

See joineRML::fitted.mjoint() and joineRML::residuals.mjoint() for more information on the difference between population-level and individual-level fitted values and residuals.

If fitting a joint model with a single longitudinal process, make sure you are using a named list to define the formula for the fixed and random effects of the longitudinal submodel.

Value

A tibble::tibble() with one row for each original observation with addition columns:

.fitted_j_0	population-level fitted values for the j-th longitudinal process
.fitted_j_1	individuals-level fitted values for the j-th longitudinal process
.resid_j_0	population-level residuals for the j-th longitudinal process
.resid_j_1	individual-level residuals for the j-th longitudinal process

Examples

```
# broom only skips running these examples because the example models take a
# while to generate—they should run just fine, though!
## Not run:
```

```
# load libraries for models and data
library(joineRML)

# fit a joint model with bivariate longitudinal outcomes
data(heart.valve)

hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
  heart.valve$num <= 50, ]

fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
  inits = list("gamma" = c(0.11, 1.51, 0.80)),
  timeVar = "time"
)

# extract the survival fixed effects
tidy(fit)

# extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# extract model statistics
glance(fit)

## End(Not run)
```

`augment.mlogit` *Augment data with information from a(n) mlogit object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'mlogit'
augment(x, data = x$model, ...)
```

Arguments

`x`	an object returned from `mlogit::mlogit()`.
`data`	Not currently used
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

At the moment this only works on the estimation dataset. Need to set it up to predict on another dataset.

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.probability	Class probability of modal class.
.resid	The difference between observed and fitted values.

See Also

augment()

Other mlogit tidiers: glance.mlogit(), tidy.mlogit()

Examples

```
# load libraries for models and data
library(mlogit)

data("Fishing", package = "mlogit")
Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")

# fit model
m <- mlogit(mode ~ price + catch | income, data = Fish)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

---

augment.nlrq *Tidy a(n) nlrq object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'nlrq'
augment(x, data = NULL, newdata = NULL, ...)
```

Arguments

x	A nlrq object returned from quantreg::nlrq().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

See Also

augment(), quantreg::nlrq()

Other quantreg tidiers: augment.rq(), augment.rqs(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rq(), tidy.rqs()

Examples

```
# fit model
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

# summarize model fit with tidiers + visualization
tidy(n)
augment(n)
glance(n)

library(ggplot2)

ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1

augment(n, newdata = newdata)
```

augment.nls *Augment data with information from a(n) nls object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'nls'
augment(x, data = NULL, newdata = NULL, ...)
```

Arguments

`x`	An `nls` object returned from `stats::nls()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

augment.nls does not currently support confidence intervals due to a lack of support in stats::predict.nls().

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

tidy, `stats::nls()`, `stats::predict.nls()`

Other nls tidiers: `glance.nls()`, `tidy.nls()`

Examples

```
# fit model
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

# summarize model fit with tidiers + visualization
tidy(n)
augment(n)
glance(n)

library(ggplot2)

ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1

augment(n, newdata = newdata)
```

augment.pam *Augment data with information from a(n) pam object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'pam'
augment(x, data = NULL, ...)
```

Arguments

`x`	An `pam` object returned from `cluster::pam()`
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.cluster`	Cluster assignment.
`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`augment()`, `cluster::pam()`

Other pam tidiers: `glance.pam()`, `tidy.pam()`

Examples

```
# load libraries for models and data
library(dplyr)
library(ggplot2)
library(cluster)
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)

# summarize model fit with tidiers + visualization
tidy(p)
glance(p)
augment(p, x)

augment(p, x) |>
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)
```

augment.plm *Augment data with information from a(n) plm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'plm'
augment(x, data = model.frame(x), ...)
```

Arguments

`x`	A `plm` objected returned by `plm::plm()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`augment()`, `plm::plm()`

Other plm tidiers: `glance.plm()`, `tidy.plm()`

Examples

```
# load libraries for models and data
library(plm)

# load data
data("Produc", package = "plm")

# fit model
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
  data = Produc, index = c("state", "year")
)

# summarize model fit with tidiers
summary(zz)

tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = 0.9)

augment(zz)
glance(zz)
```

augment.poLCA *Augment data with information from a(n) poLCA object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'poLCA'
augment(x, data = NULL, ...)
```

Arguments

`x`	A `poLCA` object returned from `poLCA::poLCA()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

If the `data` argument is given, those columns are included in the output (only rows for which predictions could be made). Otherwise, the `y` element of the poLCA object, which contains the manifest variables used to fit the model, are used, along with any covariates, if present, in `x`.

Note that while the probability of all the classes (not just the predicted modal class) can be found in the `posterior` element, these are not included in the augmented output.

Value

A `tibble::tibble()` with columns:

`.class`	Predicted class.
`.probability`	Class probability of modal class.

See Also

`augment()`, `poLCA::poLCA()`

Other poLCA tidiers: `glance.poLCA()`, `tidy.poLCA()`

Examples

```
# load libraries for models and data
library(poLCA)
library(dplyr)

# generate data
data(values)

f <- cbind(A, B, C, D) ~ 1

# fit model
M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

M1

# summarize model fit with tidiers + visualization
tidy(M1)
augment(M1)
glance(M1)

library(ggplot2)
```

```
ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
  geom_bar(stat = "identity", width = 1) +
  facet_wrap(~variable)

# three-class model with a single covariate.
data(election)

f2a <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ PARTY

nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)
td

ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
  geom_line() +
  facet_wrap(~variable, nrow = 2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)

au

count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)

au2

dim(au2)
```

`augment.polr` *Augment data with information from a(n) polr object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object.

Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'polr'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("class"),
  ...
)
```

Arguments

`x`	A `polr` object returned from `MASS::polr()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`type.predict`	Which type of prediction to compute, passed to `MASS:::predict.polr()`. Only supports `"class"` at the moment.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- tidy() methods will warn when supplied an exponentiate argument if it will be ignored.
- augment() methods will warn when supplied a newdata argument if it will be ignored.

See Also

tidy(), MASS::polr()

Other ordinal tidiers: augment.clm(), glance.clm(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(MASS)

# fit model
fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

# summarize model fit with tidiers
tidy(fit, exponentiate = TRUE, conf.int = TRUE)

glance(fit)
augment(fit, type.predict = "class")

fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)

tidy(fit, p.values = TRUE)
```

---

augment.prcomp *Augment data with information from a(n) prcomp object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'prcomp'
augment(x, data = NULL, newdata, ...)
```

Arguments

`x`	A `prcomp` object returned by `stats::prcomp()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble containing the original data along with additional columns containing each observation's projection into PCA space.

See Also

stats::prcomp(), svd_tidiers

Other svd tidiers: tidy.prcomp(), tidy_irlba(), tidy_svd()

---

augment.rlm *Augment data with information from a(n) rlm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses splines::ns(), stats::poly(), or survival::Surv(), it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'rlm'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)
```

Arguments

x	An rlm object returned by MASS::rlm().
data	A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata — A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

se_fit — Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.

... — Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- tidy() methods will warn when supplied an exponentiate argument if it will be ignored.
- augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

.fitted — Fitted or predicted value.

.hat — Diagonal of the hat matrix.

.resid — The difference between observed and fitted values.

.se.fit — Standard errors of fitted values.

.sigma — Estimated residual standard deviation when corresponding observation is dropped from model.

See Also

MASS::rlm()

Other rlm tidiers: glance.rlm(), tidy.rlm()

Examples

```
# load libraries for models and data
library(MASS)

# fit model
r <- rlm(stack.loss ~ ., stackloss)

# summarize model fit with tidiers
tidy(r)
augment(r)
glance(r)
```

augment.rma *Augment data with information from a(n) rma object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'rma'
augment(x, interval = c("prediction", "confidence"), ...)
```

Arguments

`x`	An `rma` object such as those created by `metafor::rma()`, `metafor::rma.uni()`, `metafor::rma.glmm()`, `metafor::rma.mh()`, `metafor::rma.mv()`, or `metafor::rma.peto()`.
`interval`	For `rma.mv` models, should prediction intervals (`"prediction"`, default) or confidence intervals (`"confidence"`) intervals be returned? For `rma.uni` models, prediction intervals are always returned. For `rma.mh` and `rma.peto` models, confidence intervals are always returned.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.lower`	Lower bound on interval for fitted values.
`.moderator`	In meta-analysis, the moderators used to calculate the predicted values.
`.moderator.level`	In meta-analysis, the level of the moderators used to calculate the predicted values.
`.resid`	The difference between observed and fitted values.
`.se.fit`	Standard errors of fitted values.
`.upper`	Upper bound on interval for fitted values.
`.observed`	The observed values for the individual studies

Examples

```
# load modeling library
library(metafor)

# generate data and fit
df <-
  escalc(
    measure = "RR",
    ai = tpos,
    bi = tneg,
    ci = cpos,
    di = cneg,
    data = dat.bcg
  )

meta_analysis <- rma(yi, vi, data = df, method = "EB")

# summarize model fit with tidiers
augment(meta_analysis)
```

augment.rq *Augment data with information from a(n) rq object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'rq'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

`x`	An `rq` object returned from `quantreg::rq()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.

`...` Arguments passed on to `quantreg::predict.rq`

`object` object of class rq or rqs or rq.process produced by `rq`

`interval` type of interval desired: default is 'none', when set to 'confidence' the function returns a matrix predictions with point predictions for each of the 'newdata' points as well as lower and upper confidence limits.

`level` converage probability for the 'confidence' intervals.

`type` For `predict.rq`, the method for 'confidence' intervals, if desired. If 'percentile' then one of the bootstrap methods is used to generate percentile intervals for each prediction, if 'direct' then a version of the Portnoy and Zhou (1998) method is used, and otherwise an estimated covariance matrix for the parameter estimates is used. Further arguments to determine the choice of bootstrap method or covariance matrix estimate can be passed via the `...` argument. For `predict.rqs` and `predict.rq.process` when `stepfun = TRUE`, `type` is "Qhat", "Fhat" or "fhat" depending on whether the user would like to have estimates of the conditional quantile, distribution or density functions respectively. As noted below the two former estimates can be monotonized with the function `rearrange`. When the "fhat" option is invoked, a list of conditional density functions is returned based on Silverman's adaptive kernel method as implemented in `akj` and `approxfun`.

`na.action` function determining what should be done with missing values in 'newdata'. The default is to predict 'NA'.

Details

Depending on the arguments passed on to `predict.rq` via `...`, a confidence interval is also calculated on the fitted values resulting in columns `.lower` and `.upper`. Does not provide confidence intervals when data is specified via the `newdata` argument.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.
`.tau`	Quantile.

See Also

augment, `quantreg::rq()`, `quantreg::predict.rq()`

Other quantreg tidiers: `augment.nlrq()`, `augment.rqs()`, `glance.nlrq()`, `glance.rq()`, `tidy.nlrq()`, `tidy.rq()`, `tidy.rqs()`

Examples

```
# load modeling library and data
library(quantreg)
```

```
data(stackloss)

# median (l1) regression fit for the stackloss data.
mod1 <- rq(stack.loss ~ stack.x, .5)

# weighted sample median
mod2 <- rq(rnorm(50) ~ 1, weights = runif(50))

# summarize model fit with tidiers
tidy(mod1)
glance(mod1)
augment(mod1)

tidy(mod2)
glance(mod2)
augment(mod2)

# varying tau to generate an rqs object
mod3 <- rq(stack.loss ~ stack.x, tau = c(.25, .5))

tidy(mod3)
augment(mod3)

# glance cannot handle rqs objects like `mod3`--use a purrr
# `map`-based workflow instead
```

`augment.rqs` *Augment data with information from a(n) rqs object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'rqs'
augment(x, data = model.frame(x), newdata, ...)
```

Arguments

`x`	An `rqs` object returned from `quantreg::rq()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`...`	Arguments passed on to `quantreg::predict.rq` `object` object of class rq or rqs or rq.process produced by `rq` `interval` type of interval desired: default is 'none', when set to 'confidence' the function returns a matrix predictions with point predictions for each of the 'newdata' points as well as lower and upper confidence limits. `level` converage probability for the 'confidence' intervals. `type` For `predict.rq`, the method for 'confidence' intervals, if desired. If 'percentile' then one of the bootstrap methods is used to generate percentile intervals for each prediction, if 'direct' then a version of the Portnoy and Zhou (1998) method is used, and otherwise an estimated covariance matrix for the parameter estimates is used. Further arguments to determine the choice of bootstrap method or covariance matrix estimate can be passed via the `...` argument. For `predict.rqs` and `predict.rq.process` when `stepfun = TRUE`, `type` is "Qhat", "Fhat" or "fhat" depending on whether the user would like to have estimates of the conditional quantile, distribution or density functions respectively. As noted below the two former estimates can be monotonized with the function `rearrange`. When the "fhat" option is invoked, a list of conditional density functions is returned based on Silverman's adaptive kernel method as implemented in `akj` and `approxfun`. `na.action` function determining what should be done with missing values in 'newdata'. The default is to predict 'NA'.

Details

Depending on the arguments passed on to `predict.rq` via `...`, a confidence interval is also calculated on the fitted values resulting in columns `.lower` and `.upper`. Does not provide confidence intervals when data is specified via the `newdata` argument.

See Also

augment, `quantreg::rq()`, `quantreg::predict.rqs()`

Other quantreg tidiers: `augment.nlrq()`, `augment.rq()`, `glance.nlrq()`, `glance.rq()`, `tidy.nlrq()`, `tidy.rq()`, `tidy.rqs()`

Examples

```
# load modeling library and data
library(quantreg)

data(stackloss)

# median (l1) regression fit for the stackloss data.
mod1 <- rq(stack.loss ~ stack.x, .5)

# weighted sample median
mod2 <- rq(rnorm(50) ~ 1, weights = runif(50))

# summarize model fit with tidiers
tidy(mod1)
glance(mod1)
augment(mod1)

tidy(mod2)
glance(mod2)
augment(mod2)

# varying tau to generate an rqs object
mod3 <- rq(stack.loss ~ stack.x, tau = c(.25, .5))

tidy(mod3)
augment(mod3)

# glance cannot handle rqs objects like `mod3`--use a purrr
# `map`-based workflow instead
```

`augment.sarlm` *Augment data with information from a(n) spatialreg object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'sarlm'
augment(x, data = x$X, ...)
```

Arguments

`x`	An object returned from `spatialreg::lagsarlm()` or `spatialreg::errorsarlm()`.
`data`	Ignored, but included for internal consistency. See the details below.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The predict method for sarlm objects assumes that the response is known. See ?predict.sarlm for more discussion. As a result, since the original data can be recovered from the fit object, this method currently does not take in `data` or `newdata` arguments.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`augment()`

Other spatialreg tidiers: `glance.sarlm()`, `tidy.sarlm()`

Examples

```
# load libraries for models and data
library(spatialreg)
library(spdep)

# load data
data(oldcol, package = "spdep")

listw <- nb2listw(COL.nb, style = "W")

# fit model
crime_sar <-
  lagsarlm(CRIME ~ INC + HOVAL,
    data = COL.OLD,
    listw = listw,
    method = "eigen"
  )

# summarize model fit with tidiers
tidy(crime_sar)
tidy(crime_sar, conf.int = TRUE)
glance(crime_sar)
augment(crime_sar)

# fit another model
crime_sem <- errorsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sem)
tidy(crime_sem, conf.int = TRUE)
glance(crime_sem)
```

```
augment(crime_sem)

# fit another model
crime_sac <- sacsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sac)
tidy(crime_sac, conf.int = TRUE)
glance(crime_sac)
augment(crime_sac)
```

`augment.smooth.spline` *Tidy a(n) smooth.spline object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'smooth.spline'
augment(x, data = x$data, ...)
```

Arguments

`x`	A `smooth.spline` object returned from `stats::smooth.spline()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

.fitted	Fitted or predicted value.
.resid	The difference between observed and fitted values.

See Also

augment(), stats::smooth.spline(), stats::predict.smooth.spline()

Other smoothing spline tidiers: glance.smooth.spline()

Examples

```
# fit model
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)

# summarize model fit with tidiers
augment(spl, mtcars)

# calls original columns x and y
augment(spl)

library(ggplot2)
ggplot(augment(spl, mtcars), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```

---

augment.speedlm *Augment data with information from a(n) speedlm object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'speedlm'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

`x`	A `speedlm` object returned from `speedglm::speedlm()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.

See Also

`speedglm::speedlm()`

Other speedlm tidiers: `glance.speedglm()`, `glance.speedlm()`, `tidy.speedglm()`, `tidy.speedlm()`

Examples

```
# load modeling library
library(speedglm)

# fit model
mod <- speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)

# summarize model fit with tidiers
tidy(mod)
glance(mod)
augment(mod)
```

`augment.stl` *Augment data with information from a(n) stl object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'stl'
augment(x, data = NULL, weights = TRUE, ...)
```

Arguments

x	An stl object returned from stats::stl().
data	Ignored, included for consistency with the augment generic signature only.
weights	Logical indicating whether or not to include the robust weights in the output.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble with one row for each observation in the original times series:

.seasonal	The seasonal component of the decomposition.
.trend	The trend component of the decomposition.
.remainder	The remainder, or "random" component of the decomposition.
.weight	The final robust weights, if requested.
.seasadj	The seasonally adjusted (or "deseasonalised") series.

See Also

augment(), stats::stl()

Other decompose tidiers: augment.decomposed.ts()

---

augment.survreg *Augment data with information from a(n) survreg object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. If a predictor enters the model as part of a matrix of covariates, such as when the model formula uses `splines::ns()`, `stats::poly()`, or `survival::Surv()`, it is represented as a matrix column.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```
## S3 method for class 'survreg'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = "response",
  type.residuals = "response",
  ...
)
```

Arguments

`x`	An `survreg` object returned from `survival::survreg()`.
`data`	A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`type.predict`	Character indicating type of prediction to use. Passed to the `type` argument of the `stats::predict()` generic. Allowed arguments vary with model class, so be sure to read the `predict.my_class` documentation.
`type.residuals`	Character indicating type of residuals to use. Passed to the `type` argument of `stats::residuals()` generic. Allowed arguments vary with model class, so be sure to read the `residuals.my_class` documentation.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`.fitted`	Fitted or predicted value.
`.resid`	The difference between observed and fitted values.
`.se.fit`	Standard errors of fitted values.

See Also

`augment()`, `survival::survreg()`

Other survreg tidiers: `glance.survreg()`, `tidy.survreg()`

Other survival tidiers: `augment.coxph()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
# load libraries for models and data
library(survival)

# fit model
sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx,
  ovarian,
  dist = "exponential"
)

# summarize model fit with tidiers + visualization
tidy(sr)
augment(sr, ovarian)
glance(sr)

# coefficient plot
td <- tidy(sr, conf.int = TRUE)

library(ggplot2)

ggplot(td, aes(estimate, term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
```

```
  geom_vline(xintercept = 0)
```

`augment_columns` *Add fitted values, residuals, and other common outputs to an augment call*

Description

`augment_columns` is intended for use in the internals of `augment` methods only and is exported for developers extending the broom package. Please instead use `augment()` to appropriately make use of the functionality in `augment_columns()`.

Usage

```
augment_columns(
  x,
  data,
  newdata = NULL,
  type,
  type.predict = type,
  type.residuals = type,
  se.fit = TRUE,
  ...
)
```

Arguments

`x`	a model
`data`	original data onto which columns should be added
`newdata`	new data to predict on, optional
`type`	Type of prediction and residuals to compute
`type.predict`	Type of prediction to compute; by default same as `type`
`type.residuals`	Type of residuals to compute; by default same as `type`
`se.fit`	Value to pass to predict's `se.fit`, or NULL for no value. Ignored for model types that do not accept an `se.fit` argument
`...`	extra arguments (not used)

Details

Note that, in the case that a `residuals()` or `influence()` generic is not implemented for the supplied model `x`, the function will fail quietly.

bootstrap *Set up bootstrap replicates of a dplyr operation*

Description

The `bootstrap()` function is deprecated and will be removed from an upcoming release of broom. For tidy resampling, please use the rsample package instead. Functionality is no longer supported for this method.

Usage

```
bootstrap(df, m, by_group = FALSE)
```

Arguments

`df`	a data frame
`m`	number of bootstrap replicates to perform
`by_group`	If `TRUE`, then bootstrap within each group if `df` is a grouped tibble.

Details

This code originates from Hadley Wickham (with a few small corrections) here: https://github.com/tidyverse/dplyr/issues/269

See Also

Other deprecated: `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

confint_tidy *(Deprecated) Calculate confidence interval as a tidy data frame*

Description

This function is now deprecated and will be removed from a future release of broom.

Usage

```
confint_tidy(x, conf.level = 0.95, func = stats::confint, ...)
```

Arguments

x	a model object for which confint() can be calculated
conf.level	confidence level
func	A function to compute a confidence interval for x. Calling func(x, level = conf.level, ...) must return an object coercible to a tibble. This dataframe like object should have to columns corresponding the lower and upper bounds on the confidence interval.
...	extra arguments passed on to confint

Details

Return a confidence interval as a tidy data frame. This directly wraps the confint() function, but ensures it follows broom conventions: column names of conf.low and conf.high, and no row names.

confint_tidy

Value

A tibble with two columns: conf.low and conf.high.

See Also

Other deprecated: bootstrap(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable(), tidy.numeric()

---

data.frame_tidiers *Tidiers for data.frame objects*

Description

Data frame tidiers are deprecated and will be removed from an upcoming release of broom.

Usage

```
## S3 method for class 'data.frame'
tidy(x, ..., na.rm = TRUE, trim = 0.1)

## S3 method for class 'data.frame'
augment(x, data, ...)

## S3 method for class 'data.frame'
glance(x, ...)
```

Arguments

x	A data.frame
...	Additional arguments for other methods.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Passed to the trim argument of mean
data	data, not used

Details

These perform tidy summaries of data.frame objects. tidy produces summary statistics about each column, while glance simply reports the number of rows and columns. Note that augment.data.frame will throw an error.

Value

tidy.data.frame produces a data frame with one row per original column, containing summary statistics of each:

column	name of original column
n	Number of valid (non-NA) values
mean	mean
sd	standard deviation
median	median
trimmed	trimmed mean, with trim defaulting to .1
mad	median absolute deviation (from the median)
min	minimum value
max	maximum value
range	range
skew	skew
kurtosis	kurtosis
se	standard error

glance returns a one-row data.frame with

nrow	number of rows
ncol	number of columns
complete.obs	number of rows that have no missing values
na.fraction	fraction of values across all rows and columns that are missing

Author(s)

David Robinson, Benjamin Nutter

Source

Skew and Kurtosis functions are adapted from implementations in the `moments` package:
Lukasz Komsta and Frederick Novomestky (2015). moments: Moments, cumulants, skewness, kurtosis and related tests. R package version 0.14.
https://CRAN.R-project.org/package=moments

See Also

Other deprecated: `bootstrap()`, `confint_tidy()`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

Other deprecated: `bootstrap()`, `confint_tidy()`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

Other deprecated: `bootstrap()`, `confint_tidy()`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

Examples

```
td <- tidy(mtcars)
td

glance(mtcars)

library(ggplot2)
# compare mean and standard deviation
ggplot(td, aes(mean, sd)) + geom_point() +
     geom_text(aes(label = column), hjust = 1, vjust = 1) +
     scale_x_log10() + scale_y_log10() + geom_abline()
```

`durbinWatsonTest_tidiers` *Tidy/glance a(n) durbinWatsonTest object*

Description

For models that have only a single component, the `tidy()` and `glance()` methods are identical. Please see the documentation for both of those methods.

Usage

```
## S3 method for class 'durbinWatsonTest'
tidy(x, ...)

## S3 method for class 'durbinWatsonTest'
glance(x, ...)
```

Arguments

x	An object of class durbinWatsonTest created by a call to car::durbinWatsonTest().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

alternative	Alternative hypothesis (character).
autocorrelation	Autocorrelation.
p.value	The two-sided p-value associated with the observed statistic.
statistic	Test statistic for Durbin-Watson test.
method	Always 'Durbin-Watson Test'.

See Also

tidy(), glance(), car::durbinWatsonTest()

Other car tidiers: leveneTest_tidiers

Examples

```
# load modeling library
library(car)

# fit model
dw <- durbinWatsonTest(lm(mpg ~ wt, data = mtcars))

# summarize model fit with tidiers
tidy(dw)

# same output for all durbinWatsonTests
glance(dw)
```

finish_glance *(Deprecated) Add logLik, AIC, BIC, and other common measurements to a glance of a prediction*

Description

This function is now deprecated in favor of using custom logic and the appropriate `nobs()` method.

Usage

```
finish_glance(ret, x)
```

Arguments

`ret`	a one-row data frame (a partially complete glance)
`x`	the prediction model

Value

a one-row data frame with additional columns added, such as

`logLik`	log likelihoods
`AIC`	Akaike Information Criterion
`BIC`	Bayesian Information Criterion
`deviance`	deviance
`df.residual`	residual degrees of freedom

See Also

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

fix_data_frame *Ensure an object is a data frame, with rownames moved into a column*

Description

This function is deprecated as of broom 0.7.0 and will be removed from a future release. Please see `tibble::as_tibble`.

Usage

```
fix_data_frame(x, newnames = NULL, newcol = "term")
```

Arguments

x	a data.frame or matrix
newnames	new column names, not including the rownames
newcol	the name of the new rownames column

Value

a data.frame, with rownames moved into a column and new column names assigned

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable(), tidy.numeric()

---

glance.aareg *Glance at a(n) aareg object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'aareg'
glance(x, ...)
```

Arguments

x	An aareg object returned from survival::aareg().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`df`	Degrees of freedom used by the model.
`nobs`	Number of observations used.
`p.value`	P-value corresponding to the test statistic.
`statistic`	Test statistic.

See Also

`glance()`, `survival::aareg()`

Other aareg tidiers: `tidy.aareg()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
# load libraries for models and data
library(survival)

# fit model
afit <- aareg(
  Surv(time, status) ~ age + sex + ph.ecog,
  data = lung,
  dfbeta = TRUE
)

# summarize model fit with tidiers
tidy(afit)
```

`glance.anova` *Glance at a(n) anova object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'anova'
glance(x, ...)
```

Arguments

x	An anova object, such as those created by stats::anova(), car::Anova(), car::leveneTest(), or car::linearHypothesis().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

deviance	Deviance of the model.
df.residual	Residual degrees of freedom.

Note

Note that the output of glance.anova() will vary depending on the initializing anova call. In some cases, it will just return an empty data frame. In other cases, glance.anova() may return columns that are also common to tidy.anova(). This is partly to preserve backwards compatibility with early versions of broom, but also because the underlying anova model yields components that could reasonably be interpreted as goodness-of-fit summaries too.

See Also

glance()

Other anova tidiers: glance.aov(), tidy.TukeyHSD(), tidy.anova(), tidy.aov(), tidy.aovlist(), tidy.manova()

Examples

```
# fit models
a <- lm(mpg ~ wt + qsec + disp, mtcars)
b <- lm(mpg ~ wt + qsec, mtcars)

mod <- anova(a, b)

# summarize model fit with tidiers
tidy(mod)
glance(mod)

# car::linearHypothesis() example
library(car)
mod_lht <- linearHypothesis(a, "wt - disp")
tidy(mod_lht)
glance(mod_lht)
```

---

glance.aov *Glance at a(n) lm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'aov'
glance(x, ...)
```

Arguments

x	An aov object, such as those created by stats::aov().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- tidy() methods will warn when supplied an exponentiate argument if it will be ignored.
- augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

Note

Note that tidy.aov() now contains the numerator and denominator degrees of freedom, which were included in the output of glance.aov() in some previous versions of the package.

See Also

glance()

Other anova tidiers: glance.anova(), tidy.TukeyHSD(), tidy.anova(), tidy.aov(), tidy.aovlist(), tidy.manova()

Examples

```
a <- aov(mpg ~ wt + qsec + disp, mtcars)
tidy(a)
```

glance.Arima *Glance at a(n) Arima object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'Arima'
glance(x, ...)
```

Arguments

x	An object of class Arima created by stats::arima().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
sigma	Estimated standard error of the residuals.

See Also

stats::arima()

Other Arima tidiers: tidy.Arima()

Examples

```
# fit model
fit <- arima(lh, order = c(1, 0, 0))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

---

glance.betamfx *Glance at a(n) betamfx object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'betamfx'
glance(x, ...)
```

Arguments

x	A betamfx object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored.

- augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

This glance method wraps glance.betareg() for mfx::betamfx() objects.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df.null	Degrees of freedom used by the null model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
pseudo.r.squared	Like the R squared statistic, but for situations when the R squared statistic isn't defined.

See Also

glance.betareg(), mfx::betamfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.mfx(), tidy.betamfx(), tidy.mfx()

Examples

```
library(mfx)

# Simulate some data
set.seed(12345)
n <- 1000
x <- rnorm(n)

# Beta outcome
y <- rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2 * x)))
# Use Smithson and Verkuilen correction
y <- (y * (n - 1) + 0.5) / n

d <- data.frame(y, x)
mod_betamfx <- betamfx(y ~ x | x, data = d)

tidy(mod_betamfx, conf.int = TRUE)

# Compare with the naive model coefficients of the equivalent betareg call (not run)
# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)
```

```
glance(mod_betamfx)
```

`glance.betareg` *Glance at a(n) betareg object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'betareg'
glance(x, ...)
```

Arguments

`x`	A `betareg` object produced by a call to `betareg::betareg()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`df.null`	Degrees of freedom used by the null model.

df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
pseudo.r.squared	Like the R squared statistic, but for situations when the R squared statistic isn't defined.

See Also

glance(), betareg::betareg()

Examples

```
# load libraries for models and data
library(betareg)

# load dats
data("GasolineYield", package = "betareg")

# fit model
mod <- betareg(yield ~ batch + temp, data = GasolineYield)

mod

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)

augment(mod)

glance(mod)
```

---

glance.biglm *Glance at a(n) biglm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'biglm'
glance(x, ...)
```

Arguments

x	A biglm object created by a call to biglm::biglm() or biglm::bigglm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
deviance	Deviance of the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.

See Also

glance(), biglm::biglm(), biglm::bigglm()

Other biglm tidiers: tidy.biglm()

Examples

```
# load modeling library
library(biglm)

# fit model -- linear regression
bfit <- biglm(mpg ~ wt + disp, mtcars)

# summarize model fit with tidiers
```

```
tidy(bfit)
tidy(bfit, conf.int = TRUE)
tidy(bfit, conf.int = TRUE, conf.level = .9)

glance(bfit)

# fit model -- logistic regression
bgfit <- bigglm(am ~ mpg, mtcars, family = binomial())

# summarize model fit with tidiers
tidy(bgfit)
tidy(bgfit, exponentiate = TRUE)
tidy(bgfit, conf.int = TRUE)
tidy(bgfit, conf.int = TRUE, conf.level = .9)
tidy(bgfit, conf.int = TRUE, conf.level = .9, exponentiate = TRUE)

glance(bgfit)
```

`glance.binDesign` *Glance at a(n) binDesign object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'binDesign'
glance(x, ...)
```

Arguments

`x`	A binGroup::binDesign object.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`power`	Power achieved by the analysis.
`n`	Sample size used to achieve this power.
`power.reached`	Whether the desired power was reached.
`maxit`	Number of iterations performed.

See Also

`glance()`, `binGroup::binDesign()`

Other bingroup tidiers: `tidy.binDesign()`, `tidy.binWidth()`

Examples

```
# load libraries for models and data
library(binGroup)

des <- binDesign(
  nmax = 300, delta = 0.06,
  p.hyp = 0.1, power = .8
)

glance(des)
tidy(des)

library(ggplot2)

ggplot(tidy(des), aes(n, power)) +
  geom_line()
```

`glance.cch` *Glance at a(n) cch object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'cch'
glance(x, ...)
```

Arguments

x	An cch object returned from survival::cch().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

iter	Iterations of algorithm/fitting procedure completed.
p.value	P-value corresponding to the test statistic.
rscore	Robust log-rank statistic
score	Score.
n	number of predictions
nevent	number of events

See Also

glance(), survival::cch()

Other cch tidiers: glance.survfit(), tidy.cch()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# examples come from cch documentation
subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel == 1 | subcoh == 1)
ccoh.data <- nwtco[selccoh, ]
ccoh.data$subcohort <- subcoh[selccoh]

# central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol, labels = c("FH", "UH"))

# tumour stage
ccoh.data$stage <- factor(ccoh.data$stage, labels = c("I", "II", "III", "IV"))
ccoh.data$age <- ccoh.data$age / 12 # age in years

# fit model
fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age,
  data = ccoh.data,
  subcoh = ~subcohort, id = ~seqno, cohort.size = 4028
)

# summarize model fit with tidiers + visualization
tidy(fit.ccP)

# coefficient plot
library(ggplot2)

ggplot(tidy(fit.ccP), aes(x = estimate, y = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
```

---

glance.clm *Glance at a(n) clm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'clm'
glance(x, ...)
```

Arguments

x	A clm object returned from ordinal::clm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df.residual	Residual degrees of freedom.
edf	The effective degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

See Also

tidy, ordinal::clm()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(ordinal)

# fit model
fit <- clm(rating ~ temp * contact, data = wine)

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

# ...and again with another model specification
fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)

tidy(fit2)
glance(fit2)
```

`glance.clmm` *Glance at a(n) clmm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'clmm'
glance(x, ...)
```

Arguments

x	A clmm object returned from ordinal::clmm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
edf	The effective degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

See Also

tidy, ordinal::clmm()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(ordinal)

# fit model
fit <- clmm(rating ~ temp + contact + (1 | judge), data = wine)

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, exponentiate = TRUE)

glance(fit)

# ...and again with another model specification
fit2 <- clmm(rating ~ temp + (1 | judge), nominal = ~contact, data = wine)

tidy(fit2)
```

```
glance(fit2)
```

`glance.coeftest` *Glance at a(n) coeftest object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'coeftest'
glance(x, ...)
```

Arguments

`x`	A `coeftest` object returned from `lmtest::coeftest()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`adj.r.squared`	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
`AIC`	Akaike's Information Criterion for the model.

BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
statistic	Test statistic.

Note

Because of the way that lmtest::coeftest() retains information about the underlying model object, the returned columns for glance.coeftest() will vary depending on the arguments. Specifically, four columns are returned regardless: "Loglik", "AIC", "BIC", and "nobs". Users can obtain additional columns (e.g. "r.squared", "df") by invoking the "save = TRUE" argument as part of lmtest::coeftest(). See examples.

As an aside, goodness-of-fit measures such as R-squared are unaffected by the presence of heteroskedasticity. For further discussion see, e.g. chapter 8.1 of Wooldridge (2016).

References

Wooldridge, Jeffrey M. (2016) *Introductory econometrics: A modern approach.* (6th edition). Nelson Education.

See Also

glance(), lmtest::coeftest()

Examples

```
# load libraries for models and data
library(lmtest)

m <- lm(dist ~ speed, data = cars)

coeftest(m)
tidy(coeftest(m))
tidy(coeftest(m, conf.int = TRUE))

# a very common workflow is to combine lmtest::coeftest with alternate
# variance-covariance matrices via the sandwich package. The lmtest
# tidiers support this workflow too, enabling you to adjust the standard
# errors of your tidied models on the fly.
```

```
library(sandwich)

# "HC3" (default) robust SEs
tidy(coeftest(m, vcov = vcovHC))

# "HC2" robust SEs
tidy(coeftest(m, vcov = vcovHC, type = "HC2"))

# N-W HAC robust SEs
tidy(coeftest(m, vcov = NeweyWest))

# the columns of the returned tibble for glance.coeftest() will vary
# depending on whether the coeftest object retains the underlying model.
# Users can control this with the "save = TRUE" argument of coeftest().
glance(coeftest(m))
glance(coeftest(m, save = TRUE))
```

`glance.coxph` *Glance at a(n) coxph object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'coxph'
glance(x, ...)
```

Arguments

`x`	A coxph object returned from `survival::coxph()`.
`...`	For `tidy()`, additional arguments passed to `summary(x, ...)`. Otherwise ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
n	The total number of observations.
nevent	Number of events.
nobs	Number of observations used.

See survival::coxph.object for additional column descriptions.

See Also

glance(), survival::coxph()

Other coxph tidiers: augment.coxph(), tidy.coxph()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
cfit <- coxph(Surv(time, status) ~ age + sex, lung)

# summarize model fit with tidiers
tidy(cfit)
tidy(cfit, exponentiate = TRUE)

lp <- augment(cfit, lung)
risks <- augment(cfit, lung, type.predict = "risk")
expected <- augment(cfit, lung, type.predict = "expected")

glance(cfit)

# also works on clogit models
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(
  logan[indx, ],
  id = indx,
  tocc = factor(rep(resp, each = n))
)
```

```
logan2$case <- (logan2$occupation == logan2$tocc)

cl <- clogit(case ~ tocc + tocc:education + strata(id), logan2)

tidy(cl)
glance(cl)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(risks, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(expected, aes(time, .fitted, color = sex)) +
  geom_point()
```

`glance.crr` *Glance at a(n) crr object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'crr'
glance(x, ...)
```

Arguments

`x`	A `crr` object returned from `cmprsk::crr()`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`converged`	Logical indicating if the model fitting procedure was succesful and converged.
`df`	Degrees of freedom used by the model.
`logLik`	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
`nobs`	Number of observations used.
`statistic`	Test statistic.

See Also

`glance()`, `cmprsk::crr()`

Other cmprsk tidiers: `tidy.crr()`

Examples

```
library(cmprsk)

# time to loco-regional failure (lrf)
lrf_time <- rexp(100)
lrf_event <- sample(0:2, 100, replace = TRUE)
trt <- sample(0:1, 100, replace = TRUE)
strt <- sample(1:2, 100, replace = TRUE)

# fit model
x <- crr(lrf_time, lrf_event, cbind(trt, strt))

# summarize model fit with tidiers
tidy(x, conf.int = TRUE)
glance(x)
```

`glance.cv.glmnet` *Glance at a(n) cv.glmnet object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'cv.glmnet'
glance(x, ...)
```

Arguments

`x`	A `cv.glmnet` object returned from `glmnet::cv.glmnet()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`lambda.1se`	The value of the penalization parameter lambda that results in the sparsest model while remaining within one standard error of the minimum loss.
`lambda.min`	The value of the penalization parameter lambda that achieved minimum loss as estimated by cross validation.
`nobs`	Number of observations used.

See Also

glance(), glmnet::cv.glmnet()

Other glmnet tidiers: glance.glmnet(), tidy.cv.glmnet(), tidy.glmnet()

Examples

```
# load libraries for models and data
library(glmnet)

set.seed(27)

nobs <- 100
nvar <- 50
real <- 5

x <- matrix(rnorm(nobs * nvar), nobs, nvar)
beta <- c(rnorm(real, 0, 1), rep(0, nvar - real))
y <- c(t(beta) %*% t(x)) + rnorm(nvar, sd = 3)

cvfit1 <- cv.glmnet(x, y)

tidy(cvfit1)
glance(cvfit1)

library(ggplot2)

tidied_cv <- tidy(cvfit1)
glance_cv <- glance(cvfit1)

# plot of MSE as a function of lambda
g <- ggplot(tidied_cv, aes(lambda, estimate)) +
  geom_line() +
  scale_x_log10()
g

# plot of MSE as a function of lambda with confidence ribbon
g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
g

# plot of MSE as a function of lambda with confidence ribbon and choices
# of minimum lambda marked
g <- g +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
g

# plot of number of zeros for each choice of lambda
ggplot(tidied_cv, aes(lambda, nzero)) +
  geom_line() +
  scale_x_log10()
```

```
# coefficient plot with min lambda shown
tidied <- tidy(cvfit1$glmnet.fit)

ggplot(tidied, aes(lambda, estimate, group = term)) +
  scale_x_log10() +
  geom_line() +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
```

`glance.drc` *Glance at a(n) drc object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'drc'
glance(x, ...)
```

Arguments

`x`	A `drc` object produced by a call to `drc::drm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
AICc	AIC corrected for small samples

See Also

glance(), drc::drm()

Other drc tidiers: augment.drc(), tidy.drc()

Examples

```
# load libraries for models and data
library(drc)

# fit model
mod <- drm(dead / total ~ conc, type,
  weights = total, data = selenium, fct = LL.2(), type = "binomial"
)

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)

glance(mod)

augment(mod, selenium)
```

---

glance.ergm *Glance at a(n) ergm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'ergm'
glance(x, deviance = FALSE, mcmc = FALSE, ...)
```

Arguments

x	An ergm object returned from a call to ergm::ergm().
deviance	Logical indicating whether or not to report null and residual deviance for the model, as well as degrees of freedom. Defaults to FALSE.
mcmc	Logical indicating whether or not to report MCMC interval, burn-in and sample size used to estimate the model. Defaults to FALSE.
...	Additional arguments to pass to ergm::summary(). **Cautionary note**: Misspecified arguments may be silently ignored.

Value

glance.ergm returns a one-row tibble with the columns

independence	Whether the model assumed dyadic independence
iterations	The number of MCMLE iterations performed before convergence
logLik	If applicable, the log-likelihood associated with the model
AIC	The Akaike Information Criterion
BIC	The Bayesian Information Criterion

If deviance = TRUE, and if the model supports it, the tibble will also contain the columns

null.deviance	The null deviance of the model
df.null	The degrees of freedom of the null deviance
residual.deviance	The residual deviance of the model
df.residual	The degrees of freedom of the residual deviance

See Also

glance(), ergm::ergm(), ergm::summary.ergm()

Other ergm tidiers: tidy.ergm()

glance.factanal *Glance at a(n) factanal object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'factanal'
glance(x, ...)
```

Arguments

x	A factanal object created by stats::factanal().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

converged	Logical indicating if the model fitting procedure was succesful and converged.
df	Degrees of freedom used by the model.
method	Which method was used.
n	The total number of observations.
n.factors	The number of fitted factors.

nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
statistic	Test statistic.
total.variance	Total cumulative proportion of variance accounted for by all factors.

See Also

glance(), stats::factanal()

Other factanal tidiers: augment.factanal(), tidy.factanal()

Examples

```
set.seed(123)

# generate data
library(dplyr)
library(purrr)

m1 <- tibble(
  v1 = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 4, 5, 6),
  v2 = c(1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 3, 4, 3, 3, 3, 4, 6, 5),
  v3 = c(3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 4, 6),
  v4 = c(3, 3, 4, 3, 3, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 5, 6, 4),
  v5 = c(1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 6, 4, 5),
  v6 = c(1, 1, 1, 2, 1, 3, 3, 3, 4, 3, 1, 1, 1, 2, 1, 6, 5, 4)
)

# new data
m2 <- map_dfr(m1, rev)

# factor analysis objects
fit1 <- factanal(m1, factors = 3, scores = "Bartlett")
fit2 <- factanal(m1, factors = 3, scores = "regression")

# tidying the object
tidy(fit1)
tidy(fit2)

# augmented dataframe
augment(fit1)
augment(fit2)

# augmented dataframe (with new data)
augment(fit1, data = m2)
augment(fit2, data = m2)
```

`glance.felm` *Glance at a(n) felm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'felm'
glance(x, ...)
```

Arguments

`x`	A `felm` object returned from `lfe::felm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`adj.r.squared`	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
`df`	Degrees of freedom used by the model.
`df.residual`	Residual degrees of freedom.
`nobs`	Number of observations used.
`p.value`	P-value corresponding to the test statistic.

r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
statistic	Test statistic.

Examples

```
# load libraries for models and data
library(lfe)

# use built-in `airquality` dataset
head(airquality)

# no FEs; same as lm()
est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)

# summarize model fit with tidiers
tidy(est0)
augment(est0)

# add month fixed effects
est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)

# summarize model fit with tidiers
tidy(est1)
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# the "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)

# summarize model fit with tidiers
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")
```

---

glance.fitdistr *Glance at a(n) fitdistr object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'fitdistr'
glance(x, ...)
```

Arguments

x	A fitdistr object returned by MASS::fitdistr().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

See Also

tidy(), MASS::fitdistr()

Other fitdistr tidiers: tidy.fitdistr()

Examples

```
# load libraries for models and data
library(MASS)

# generate data
set.seed(2015)
x <- rnorm(100, 5, 2)

#  fit models
fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

`glance.fixest` *Glance at a(n) fixest object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'fixest'
glance(x, ...)
```

Arguments

`x`	A `fixest` object returned from any of the `fixest` estimators
`...`	Additional arguments passed to `summary` and `confint`. Important arguments are `se` and `cluster`. Other arguments are `dof`, `exact_dof`, `forceCovariance`, and `keepBounded`. See `summary.fixest`.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
pseudo.r.squared	Like the R squared statistic, but for situations when the R squared statistic isn't defined.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
within.r.squared	R squared within fixed-effect groups.

Note

All columns listed below will be returned, but some will be NA, depending on the type of model estimated. sigma, r.squared, adj.r.squared, and within.r.squared will be NA for any model other than feols. pseudo.r.squared will be NA for feols.

Examples

```
# load libraries for models and data
library(fixest)

gravity <-
  feols(
    log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade
  )

tidy(gravity)
glance(gravity)
augment(gravity, trade)

# to get robust or clustered SEs, users can either:

# 1) specify the arguments directly in the `tidy()` call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))

tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) or, feed tidy() a summary.fixest object that has already accepted
# these arguments
```

```
gravity_summ <- summary(gravity, cluster = c("Product", "Year"))

tidy(gravity_summ, conf.int = TRUE)

# approach (1) is preferred.
```

`glance.Gam` *Glance at a(n) Gam object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'Gam'
glance(x, ...)
```

Arguments

`x`	A `Gam` object returned from a call to `gam::gam()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

Glance at `gam` objects created by calls to `mgcv::gam()` with `glance.gam()`.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

See Also

glance(), gam::gam()

Other gam tidiers: tidy.Gam()

---

glance.gam *Glance at a(n) gam object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'gam'
glance(x, ...)
```

Arguments

x	A gam object returned from a call to mgcv::gam().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
npar	Number of parameters in the model.

See Also

glance(), mgcv::gam()

Other mgcv tidiers: tidy.gam()

Examples

```
# load libraries for models and data
library(mgcv)

# fit model
g <- gam(mpg ~ s(hp) + am + qsec, data = mtcars)

# summarize model fit with tidiers
tidy(g)
tidy(g, parametric = TRUE)
glance(g)
augment(g)
```

`glance.garch` *Tidy a(n) garch object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'garch'
glance(x, test = c("box-ljung-test", "jarque-bera-test"), ...)
```

Arguments

`x`	A `garch` object returned by `tseries::garch()`.
`test`	Character specification of which hypothesis test to use. The `garch` function reports 2 hypothesis tests: Jarque-Bera to residuals and Box-Ljung to squared residuals.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`logLik`	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
`method`	Which method was used.
`nobs`	Number of observations used.
`p.value`	P-value corresponding to the test statistic.
`statistic`	Test statistic.
`parameter`	Parameter field in the htest, typically degrees of freedom.

See Also

glance(), tseries::garch(), []

Other garch tidiers: tidy.garch()

---

glance.geeglm *Glance at a(n) geeglm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'geeglm'
glance(x, ...)
```

Arguments

x	A geeglm object returned from a call to geepack::geeglm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

alpha	Estimated correlation parameter for geepack::geeglm.
df.residual	Residual degrees of freedom.

gamma	Estimated scale parameter for geepack::geeglm.
max.cluster.size	Max number of elements in clusters.
n.clusters	Number of clusters.

See Also

glance(), geepack::geeglm()

Examples

```
# load modeling library
library(geepack)

# load data
data(state)


ds <- data.frame(state.region, state.x77)

# fit model
geefit <- geeglm(Income ~ Frost + Murder,
  id = state.region,
  data = ds,
  corstr = "exchangeable"
)

# summarize model fit with tidiers
tidy(geefit)
tidy(geefit, conf.int = TRUE)
```

---

glance.glm *Glance at a(n) glm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'glm'
glance(x, ...)
```

Arguments

x	A glm object returned from stats::glm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df.null	Degrees of freedom used by the null model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
null.deviance	Deviance of the null model.

See Also

stats::glm()

Other lm tidiers: augment.glm(), augment.lm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

Examples

```
g <- glm(am ~ mpg, mtcars, family = "binomial")
glance(g)
```

`glance.glmnet` *Glance at a(n) glmnet object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'glmnet'
glance(x, ...)
```

Arguments

`x`	A `glmnet` object returned from `glmnet::glmnet()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`nobs`	Number of observations used.
`npasses`	Total passes over the data across all lambda values.
`nulldev`	Null deviance.

See Also

glance(), glmnet::glmnet()

Other glmnet tidiers: glance.cv.glmnet(), tidy.cv.glmnet(), tidy.glmnet()

Examples

```
# load libraries for models and data
library(glmnet)

set.seed(2014)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)
fit1 <- glmnet(x, y)

# summarize model fit with tidiers + visualization
tidy(fit1)
glance(fit1)

library(dplyr)
library(ggplot2)

tidied <- tidy(fit1) |> filter(term != "(Intercept)")

ggplot(tidied, aes(step, estimate, group = term)) +
  geom_line()

ggplot(tidied, aes(lambda, estimate, group = term)) +
  geom_line() +
  scale_x_log10()

ggplot(tidied, aes(lambda, dev.ratio)) +
  geom_line()

# works for other types of regressions as well, such as logistic
g2 <- sample(1:2, 100, replace = TRUE)
fit2 <- glmnet(x, g2, family = "binomial")
tidy(fit2)
```

---

glance.glmRob *Glance at a(n) glmRob object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'glmRob'
glance(x, ...)
```

Arguments

x	A glmRob object returned from robust::glmRob().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

deviance	Deviance of the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
null.deviance	Deviance of the null model.
sigma	Estimated standard error of the residuals.

See Also

robust::glmRob()

Other robust tidiers: augment.lmRob(), glance.lmRob(), tidy.glmRob(), tidy.lmRob()

Examples

```
# load libraries for models and data
library(robust)

# fit model
gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")

# summarize model fit with tidiers
tidy(gm)
glance(gm)
```

`glance.gmm` *Glance at a(n) gmm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'gmm'
glance(x, ...)
```

Arguments

`x`	A `gmm` object returned from `gmm::gmm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
statistic	Test statistic.

See Also

glance(), gmm::gmm()

Other gmm tidiers: tidy.gmm()

Examples

```
# load libraries for models and data
library(gmm)

# examples come from the "gmm" package
# CAPM test with GMM
data(Finance)
r <- Finance[1:300, 1:10]
rm <- Finance[1:300, "rm"]
rf <- Finance[1:300, "rf"]

z <- as.matrix(r - rf)
t <- nrow(z)
zm <- rm - rf
h <- matrix(zm, t, 1)
res <- gmm(z ~ zm, x = h)

# tidy result
tidy(res)
tidy(res, conf.int = TRUE)
tidy(res, conf.int = TRUE, conf.level = .99)

# coefficient plot
library(ggplot2)
library(dplyr)

tidy(res, conf.int = TRUE) |>
  mutate(variable = reorder(term, estimate)) |>
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)

# from a function instead of a matrix
```

```
g <- function(theta, x) {
  e <- x[, 2:11] - theta[1] - (x[, 1] - theta[1]) %*% matrix(theta[2:11], 1, 10)
  gmat <- cbind(e, e * c(x[, 1]))
  return(gmat)
}

x <- as.matrix(cbind(rm, r))
res_black <- gmm(g, x = x, t0 = rep(0, 11))

tidy(res_black)
tidy(res_black, conf.int = TRUE)

# APT test with Fama-French factors and GMM

f1 <- zm
f2 <- Finance[1:300, "hml"] - rf
f3 <- Finance[1:300, "smb"] - rf
h <- cbind(f1, f2, f3)
res2 <- gmm(z ~ f1 + f2 + f3, x = h)

td2 <- tidy(res2, conf.int = TRUE)
td2

# coefficient plot
td2 |>
  mutate(variable = reorder(term, estimate)) |>
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)
```

`glance.ivreg` *Glance at a(n) ivreg object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'ivreg'
glance(x, diagnostics = FALSE, ...)
```

Arguments

x	An ivreg object created by a call to AER::ivreg().
diagnostics	Logical indicating whether or not to return the Wu-Hausman and Sargan diagnostic information.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

This tidier currently only supports ivreg-classed objects outputted by the AER package. The ivreg package also outputs objects of class ivreg, and will be supported in a later release.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
statistic	Wald test statistic.
p.value	P-value for the Wald test.

Note

Beginning 0.7.0, glance.ivreg returns statistics for the Wu-Hausman test for endogeneity and the Sargan test of overidentifying restrictions. Sargan test values are returned as NA if the number of instruments is not greater than the number of endogenous regressors.

See Also

glance(), AER::ivreg()

Other ivreg tidiers: augment.ivreg(), tidy.ivreg()

Examples

```
# load libraries for models and data
library(AER)

# load data
data("CigarettesSW", package = "AER")

# fit model
ivr <- ivreg(
  log(packs) ~ income | population,
  data = CigarettesSW,
  subset = year == "1995"
)

# summarize model fit with tidiers
tidy(ivr)
tidy(ivr, conf.int = TRUE)
tidy(ivr, conf.int = TRUE, instruments = TRUE)

augment(ivr)
augment(ivr, data = CigarettesSW)
augment(ivr, newdata = CigarettesSW)

glance(ivr)
```

---

glance.kmeans *Glance at a(n) kmeans object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'kmeans'
glance(x, ...)
```

Arguments

x	A kmeans object created by stats::kmeans().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

betweenss	The total between-cluster sum of squares.
iter	Iterations of algorithm/fitting procedure completed.
tot.withinss	The total within-cluster sum of squares.
totss	The total sum of squares.

See Also

glance(), stats::kmeans()

Other kmeans tidiers: augment.kmeans(), tidy.kmeans()

Examples

```
library(cluster)
library(modeldata)
library(dplyr)

data(hpc_data)

x <- hpc_data[, 2:5]

fit <- pam(x, k = 4)

tidy(fit)
glance(fit)
augment(fit, x)
```

`glance.lavaan` *Glance at a(n) lavaan object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'lavaan'
glance(x, ...)
```

Arguments

`x`	A `lavaan` object, such as those returned from `lavaan::cfa()`, and `lavaan::sem()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A one-row tibble::tibble with columns:

`chisq`	Model chi squared
`npar`	Number of parameters in the model
`rmsea`	Root mean square error of approximation
`rmsea.conf.high`	95 percent upper bound on RMSEA

srmr	Standardised root mean residual
agfi	Adjusted goodness of fit
cfi	Comparative fit index
tli	Tucker Lewis index
AIC	Akaike information criterion
BIC	Bayesian information criterion
ngroups	Number of groups in model
nobs	Number of observations included
norig	Number of observation in the original dataset
nexcluded	Number of excluded observations
converged	Logical - Did the model converge
estimator	Estimator used
missing_method	Method for eliminating missing data

For further recommendations on reporting SEM and CFA models see Schreiber, J. B. (2017). Update to core reporting practices in structural equation modeling. Research in Social and Administrative Pharmacy, 13(3), 634-643. https://doi.org/10.1016/j.sapharm.2016.06.006

See Also

`glance()`, `lavaan::cfa()`, `lavaan::sem()`, `lavaan::fitmeasures()`

Other lavaan tidiers: `tidy.lavaan()`

Examples

```
library(lavaan)

# fit model
cfa.fit <- cfa(
  "F =~ x1 + x2 + x3 + x4 + x5",
  data = HolzingerSwineford1939, group = "school"
)

# summarize model fit with tidiers
glance(cfa.fit)
```

`glance.lm` *Glance at a(n) lm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'lm'
glance(x, ...)
```

Arguments

`x`	An `lm` object created by `stats::lm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`adj.r.squared`	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.
`df.residual`	Residual degrees of freedom.

logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
statistic	Test statistic.
df	The degrees for freedom from the numerator of the overall F-statistic. This is new in broom 0.7.0. Previously, this reported the rank of the design matrix, which is one more than the numerator degrees of freedom of the overall F-statistic.

See Also

glance(), glance.summary.lm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

Examples

```
library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)

tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)

ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# aside: There are tidy() and glance() methods for lm.summary objects too.
# this can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars |>
```

```
  head(6) |>
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval

# simpler bivariate model since we're plotting in 2D
mod2 <- lm(mpg ~ wt, data = mtcars)

au <- augment(mod2, newdata = newdata, interval = "prediction")

ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata |>
  select(-mpg)

augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)

ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
b <- a + rnorm(length(a))
result <- lm(b ~ a)

tidy(result)
```

`glance.lmodel2` *Glance at a(n) lmodel2 object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'lmodel2'
glance(x, ...)
```

Arguments

x	A lmodel2 object returned by lmodel2::lmodel2().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
theta	Angle between OLS lines 'lm(y ~ x)' and 'lm(x ~ y)'
H	H statistic for computing confidence interval of major axis slope

See Also

glance(), lmodel2::lmodel2()

Other lmodel2 tidiers: tidy.lmodel2()

Examples

```
# load libraries for models and data
library(lmodel2)

data(mod2ex2)
Ex2.res <- lmodel2(Prey ~ Predators, data = mod2ex2, "relative", "relative", 99)
Ex2.res

# summarize model fit with tidiers + visualization
tidy(Ex2.res)
glance(Ex2.res)

# this allows coefficient plots with ggplot2
library(ggplot2)

ggplot(tidy(Ex2.res), aes(estimate, term, color = method)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
```

`glance.lmRob` *Glance at a(n) lmRob object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'lmRob'
glance(x, ...)
```

Arguments

x	A lmRob object returned from robust::lmRob().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

deviance	Deviance of the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.

See Also

robust::lmRob()

Other robust tidiers: augment.lmRob(), glance.glmRob(), tidy.glmRob(), tidy.lmRob()

Examples

```
# load modeling library
library(robust)

# fit model
m <- lmRob(mpg ~ wt, data = mtcars)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

`glance.lmrob` *Glance at a(n) lmrob object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'lmrob'
glance(x, ...)
```

Arguments

`x`	A `lmrob` object returned from `robustbase::lmrob()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

Value

A `tibble::tibble()` with exactly one row and columns:

`df.residual`	Residual degrees of freedom.
`r.squared`	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
`sigma`	Estimated standard error of the residuals.

See Also

robustbase::lmrob()

Other robustbase tidiers: augment.glmrob(), augment.lmrob(), tidy.glmrob(), tidy.lmrob()

Examples

```
if (requireNamespace("robustbase", quietly = TRUE)) {
  # load libraries for models and data
  library(robustbase)

  data(coleman)
  set.seed(0)

  m <- lmrob(Y ~ ., data = coleman)
  tidy(m)
  augment(m)
  glance(m)

  data(carrots)

  Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
    family = binomial, data = carrots, method = "Mqle",
    control = glmrobMqle.control(tcc = 1.2)
  )

  tidy(Rfit)
  augment(Rfit)
}
```

---

glance.margins *Glance at a(n) margins object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'margins'
glance(x, ...)
```

Arguments

x	A margins object returned from margins::margins().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma	Estimated standard error of the residuals.
statistic	Test statistic.

Examples

```
# load libraries for models and data
library(margins)

# example 1: logit model
mod_log <- glm(am ~ cyl + hp + wt, data = mtcars, family = binomial)

# get tidied "naive" model coefficients
tidy(mod_log)

# convert to marginal effects with margins()
marg_log <- margins(mod_log)
```

```
# get tidied marginal effects
tidy(marg_log)
tidy(marg_log, conf.int = TRUE)

# requires running the underlying model again. quick for this example
glance(marg_log)

# augmenting `margins` outputs isn't supported, but
# you can get the same info by running on the underlying model
augment(mod_log)

# example 2: threeway interaction terms
mod_ie <- lm(mpg ~ wt * cyl * disp, data = mtcars)

# get tidied "naive" model coefficients
tidy(mod_ie)

# convert to marginal effects with margins()
marg_ie0 <- margins(mod_ie)
# get tidied marginal effects
tidy(marg_ie0)
glance(marg_ie0)

# marginal effects evaluated at specific values of a variable (here: cyl)
marg_ie1 <- margins(mod_ie, at = list(cyl = c(4,6,8)))

# summarize model fit with tidiers
tidy(marg_ie1)

# marginal effects of one interaction variable (here: wt), modulated at
# specific values of the two other interaction variables (here: cyl and drat)
marg_ie2 <- margins(mod_ie,
                    variables = "wt",
                    at = list(cyl = c(4,6,8), drat = c(3, 3.5, 4)))

# summarize model fit with tidiers
tidy(marg_ie2)
```

`glance.Mclust` *Glance at a(n) Mclust object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'Mclust'
glance(x, ...)
```

Arguments

x	An Mclust object return from mclust::Mclust().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

BIC	Bayesian Information Criterion for the model.
df	Degrees of freedom used by the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
model	A string denoting the model type with optimal BIC
G	Number mixture components in optimal model
hypvol	If the other model contains a noise component, the value of the hypervolume parameter. Otherwise 'NA'.

Examples

```
# load library for models and data
library(mclust)

# load data manipulation libraries
library(dplyr)
library(tibble)
```

```
library(purrr)
library(tidyr)

set.seed(27)

centers <- tibble(
  cluster = factor(1:3),
  # number points in each cluster
  num_points = c(100, 150, 50),
  # x1 coordinate of cluster center
  x1 = c(5, 0, -3),
  # x2 coordinate of cluster center
  x2 = c(-1, 1, -2)
)

points <- centers |>
  mutate(
    x1 = map2(num_points, x1, rnorm),
    x2 = map2(num_points, x2, rnorm)
  ) |>
  select(-num_points, -cluster) |>
  unnest(c(x1, x2))

# fit model
m <- Mclust(points)

# summarize model fit with tidiers
tidy(m)
augment(m, points)
glance(m)
```

`glance.mfx` *Glance at a(n) mfx object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'mfx'
glance(x, ...)

## S3 method for class 'logitmfx'
glance(x, ...)

## S3 method for class 'negbinmfx'
glance(x, ...)

## S3 method for class 'poissonmfx'
glance(x, ...)

## S3 method for class 'probitmfx'
glance(x, ...)
```

Arguments

x	A logitmfx, negbinmfx, poissonmfx, or probitmfx object. (Note that betamfx objects receive their own set of tidiers.)
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

This generic glance method wraps glance.glm() for applicable objects from the mfx package.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df.null	Degrees of freedom used by the null model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
null.deviance	Deviance of the null model.

See Also

glance.glm(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.betamfx(), tidy.betamfx(), tidy.mfx()

Examples

```
# load libraries for models and data
library(mfx)

# get the marginal effects from a logit regression
mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)

tidy(mod_logmfx, conf.int = TRUE)

# compare with the naive model coefficients of the same logit call
tidy(
  glm(am ~ cyl + hp + wt, family = binomial, data = mtcars),
  conf.int = TRUE
)

augment(mod_logmfx)
glance(mod_logmfx)

# another example, this time using probit regression
mod_probmfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)

tidy(mod_probmfx, conf.int = TRUE)
augment(mod_probmfx)
glance(mod_probmfx)
```

---

glance.mjoint *Glance at a(n) mjoint object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'mjoint'
glance(x, ...)
```

Arguments

x	An mjoint object returned from joineRML::mjoint().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
sigma2_j	The square root of the estimated residual variance for the j-th longitudinal process

See Also

glance(), joineRML::mjoint()

Other mjoint tidiers: tidy.mjoint()

Examples

```
# broom only skips running these examples because the example models take a
# while to generate—they should run just fine, though!
## Not run:


# load libraries for models and data
library(joineRML)

# fit a joint model with bivariate longitudinal outcomes
data(heart.valve)

hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
```

```
  heart.valve$num <= 50, ]

fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
  inits = list("gamma" = c(0.11, 1.51, 0.80)),
  timeVar = "time"
)

# extract the survival fixed effects
tidy(fit)

# extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# extract model statistics
glance(fit)

## End(Not run)
```

`glance.mlogit` *Glance at a(n) mlogit object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'mlogit'
glance(x, ...)
```

Arguments

x	an object returned from mlogit::mlogit().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
rho2	McFadden's rho squared with respect to a market shares (constants-only) model.
rho20	McFadden's rho squared with respect to an equal shares (no information) model.

See Also

glance(), mlogit::mlogit()

Other mlogit tidiers: augment.mlogit(), tidy.mlogit()

Examples

```
# load libraries for models and data
library(mlogit)
```

```
data("Fishing", package = "mlogit")
Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")

# fit model
m <- mlogit(mode ~ price + catch | income, data = Fish)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

`glance.muhaz` *Glance at a(n) muhaz object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'muhaz'
glance(x, ...)
```

Arguments

`x`	A `muhaz` object returned by `muhaz::muhaz()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

max.hazard	Maximal estimated hazard.
max.time	The maximum observed event or censoring time.
min.hazard	Minimal estimated hazard.
min.time	The minimum observed event or censoring time.
nobs	Number of observations used.

See Also

glance(), muhaz::muhaz()

Other muhaz tidiers: tidy.muhaz()

Examples

```
# load libraries for models and data
library(muhaz)
library(survival)

# fit model
x <- muhaz(ovarian$futime, ovarian$fustat)

# summarize model fit with tidiers
tidy(x)
glance(x)
```

---

glance.multinom *Glance at a(n) multinom object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'multinom'
glance(x, ...)
```

Arguments

x	A multinom object returned from nnet::multinom().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
deviance	Deviance of the model.
edf	The effective degrees of freedom.
nobs	Number of observations used.

See Also

glance(), nnet::multinom()

Other multinom tidiers: tidy.multinom()

Examples

```
# load libraries for models and data
library(nnet)
library(MASS)

example(birthwt)

bwt.mu <- multinom(low ~ ., bwt)

tidy(bwt.mu)
glance(bwt.mu)

# or, for output from a multinomial logistic regression
fit.gear <- multinom(gear ~ mpg + factor(am), data = mtcars)
tidy(fit.gear)
```

```
glance(fit.gear)
```

`glance.negbin` *Glance at a(n) negbin object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'negbin'
glance(x, ...)
```

Arguments

`x`	A `negbin` object returned by `MASS::glm.nb()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.

df.null	Degrees of freedom used by the null model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
null.deviance	Deviance of the null model.

See Also

glance(), MASS::glm.nb()

Other glm.nb tidiers: tidy.negbin()

Examples

```
# load libraries for models and data
library(MASS)

# fit model
r <- glm.nb(Days ~ Sex / (Age + Eth * Lrn), data = quine)

# summarize model fit with tidiers
tidy(r)
glance(r)
```

---

glance.nlrq *Glance at a(n) nlrq object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'nlrq'
glance(x, ...)
```

Arguments

x	A nlrq object returned from quantreg::nlrq().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
tau	Quantile.

See Also

glance(), quantreg::nlrq()

Other quantreg tidiers: augment.nlrq(), augment.rq(), augment.rqs(), glance.rq(), tidy.nlrq(), tidy.rq(), tidy.rqs()

Examples

```
# load modeling library
library(quantreg)

# build artificial data with multiplicative error
set.seed(1)
dat <- NULL
dat$x <- rep(1:25, 20)
dat$y <- SSlogis(dat$x, 10, 12, 2) * rnorm(500, 1, 0.1)

# fit the median using nlrq
mod <- nlrq(y ~ SSlogis(x, Asym, mid, scal),
  data = dat, tau = 0.5, trace = TRUE
)

# summarize model fit with tidiers
tidy(mod)
glance(mod)
```

```
augment(mod)
```

`glance.nls` *Glance at a(n) nls object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'nls'
glance(x, ...)
```

Arguments

`x`	An `nls` object returned from `stats::nls()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.

df.residual	Residual degrees of freedom.
finTol	The achieved convergence tolerance.
isConv	Whether the fit successfully converged.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
sigma	Estimated standard error of the residuals.

See Also

tidy, stats::nls()

Other nls tidiers: augment.nls(), tidy.nls()

Examples

```
# fit model
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

# summarize model fit with tidiers + visualization
tidy(n)
augment(n)
glance(n)

library(ggplot2)

ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1

augment(n, newdata = newdata)
```

---

glance.pam *Glance at a(n) pam object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'pam'
glance(x, ...)
```

Arguments

x	An pam object returned from cluster::pam()
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

avg.silhouette.width
: The average silhouette width for the dataset.

See Also

glance(), cluster::pam()

Other pam tidiers: augment.pam(), tidy.pam()

Examples

```
# load libraries for models and data
library(dplyr)
library(ggplot2)
library(cluster)
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)
```

```
# summarize model fit with tidiers + visualization
tidy(p)
glance(p)
augment(p, x)

augment(p, x) |>
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)
```

`glance.plm` *Glance at a(n) plm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'plm'
glance(x, ...)
```

Arguments

`x`	A `plm` objected returned by `plm::plm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
deviance	Deviance of the model.
df.residual	Residual degrees of freedom.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
statistic	F-statistic

See Also

glance(), plm::plm()

Other plm tidiers: augment.plm(), tidy.plm()

Examples

```
# load libraries for models and data
library(plm)

# load data
data("Produc", package = "plm")

# fit model
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
  data = Produc, index = c("state", "year")
)

# summarize model fit with tidiers
summary(zz)

tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = 0.9)

augment(zz)
glance(zz)
```

`glance.poLCA` *Glance at a(n) poLCA object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'poLCA'
glance(x, ...)
```

Arguments

`x`	A `poLCA` object returned from `poLCA::poLCA()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`chi.squared`	The Pearson Chi-Square goodness of fit statistic for multiway tables.
`df`	Degrees of freedom used by the model.
`df.residual`	Residual degrees of freedom.

logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
g.squared	The likelihood ratio/deviance statistic

See Also

glance(), poLCA::poLCA()

Other poLCA tidiers: augment.poLCA(), tidy.poLCA()

Examples

```
# load libraries for models and data
library(poLCA)
library(dplyr)

# generate data
data(values)

f <- cbind(A, B, C, D) ~ 1

# fit model
M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

M1

# summarize model fit with tidiers + visualization
tidy(M1)
augment(M1)
glance(M1)

library(ggplot2)

ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
  geom_bar(stat = "identity", width = 1) +
  facet_wrap(~variable)

# three-class model with a single covariate.
data(election)

f2a <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ PARTY

nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)
td

ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
```

```
  geom_line() +
  facet_wrap(~variable, nrow = 2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)

au

count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)

au2

dim(au2)
```

`glance.polr` *Glance at a(n) polr object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'polr'
glance(x, ...)
```

Arguments

`x`	A `polr` object returned from `MASS::polr()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.
`df.residual`	Residual degrees of freedom.
`edf`	The effective degrees of freedom.
`logLik`	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
`nobs`	Number of observations used.

See Also

tidy, `MASS::polr()`

Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clm()`, `glance.clmm()`, `glance.svyolr()`, `tidy.clm()`, `tidy.clmm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```
# load libraries for models and data
library(MASS)

# fit model
fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

# summarize model fit with tidiers
tidy(fit, exponentiate = TRUE, conf.int = TRUE)

glance(fit)
augment(fit, type.predict = "class")

fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)

tidy(fit, p.values = TRUE)
```

`glance.pyears` *Glance at a(n) pyears object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'pyears'
glance(x, ...)
```

Arguments

`x`	A `pyears` object returned from `survival::pyears()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`nobs`	Number of observations used.
`total`	total number of person-years tabulated
`offtable`	total number of person-years off table

See Also

glance(), survival::pyears()

Other pyears tidiers: tidy.pyears()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# generate and format data
temp.yr <- tcut(mgus$dxyr, 55:92, labels = as.character(55:91))
temp.age <- tcut(mgus$age, 34:101, labels = as.character(34:100))
ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
pfit <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus,
  data.frame = TRUE
)

# summarize model fit with tidiers
tidy(pfit)
glance(pfit)

# if data.frame argument is not given, different information is present in
# output
pfit2 <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus)

tidy(pfit2)
glance(pfit2)
```

---

glance.ridgelm *Glance at a(n) ridgelm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'ridgelm'
glance(x, ...)
```

Arguments

x	A ridgelm object returned from MASS::lm.ridge().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

This is similar to the output of select.ridgelm, but it is returned rather than printed.

Value

A tibble::tibble() with exactly one row and columns:

kHKB	modified HKB estimate of the ridge constant
kLW	modified L-W estimate of the ridge constant
lambdaGCV	choice of lambda that minimizes GCV

See Also

glance(), MASS::select.ridgelm(), MASS::lm.ridge()

Other ridgelm tidiers: tidy.ridgelm()

Examples

```
# load libraries for models and data
library(MASS)

names(longley)[1] <- "y"

# fit model and summarizd results
fit1 <- lm.ridge(y ~ ., longley)
```

```
tidy(fit1)

fit2 <- lm.ridge(y ~ ., longley, lambda = seq(0.001, .05, .001))
td2 <- tidy(fit2)
g2 <- glance(fit2)

# coefficient plot
library(ggplot2)
ggplot(td2, aes(lambda, estimate, color = term)) +
  geom_line()

# GCV plot
ggplot(td2, aes(lambda, GCV)) +
  geom_line()

# add line for the GCV minimizing estimate
ggplot(td2, aes(lambda, GCV)) +
  geom_line() +
  geom_vline(xintercept = g2$lambdaGCV, col = "red", lty = 2)
```

`glance.rlm` *Glance at a(n) rlm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'rlm'
glance(x, ...)
```

Arguments

`x`	An `rlm` object returned by `MASS::rlm()`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`converged`	Logical indicating if the model fitting procedure was succesful and converged.
`deviance`	Deviance of the model.
`logLik`	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
`nobs`	Number of observations used.
`sigma`	Estimated standard error of the residuals.

See Also

`glance()`, `MASS::rlm()`

Other rlm tidiers: `augment.rlm()`, `tidy.rlm()`

Examples

```
# load libraries for models and data
library(MASS)

# fit model
r <- rlm(stack.loss ~ ., stackloss)

# summarize model fit with tidiers
tidy(r)
augment(r)
glance(r)
```

`glance.rma` *Glance at a(n) rma object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'rma'
glance(x, ...)
```

Arguments

`x`	An `rma` object such as those created by `metafor::rma()`, `metafor::rma.uni()`, `metafor::rma.glmm()`, `metafor::rma.mh()`, `metafor::rma.mv()`, or `metafor::rma.peto()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`cochran.qe`	In meta-analysis, test statistic for the Cochran's Q_e test of residual heterogeneity.
`cochran.qm`	In meta-analysis, test statistic for the Cochran's Q_m omnibus test of coefficients.
`df.residual`	Residual degrees of freedom.

h.squared	Value of the H-Squared statistic.
i.squared	Value of the I-Squared statistic.
measure	The measure used in the meta-analysis.
method	Which method was used.
nobs	Number of observations used.
p.value.cochran.qe	In meta-analysis, p-value for the Cochran's Q_e test of residual heterogeneity.
p.value.cochran.qm	In meta-analysis, p-value for the Cochran's Q_m omnibus test of coefficients.
tau.squared	In meta-analysis, estimated amount of residual heterogeneity.
tau.squared.se	In meta-analysis, standard error of residual heterogeneity.

Examples

```
library(metafor)

df <-
  escalc(
    measure = "RR",
    ai = tpos,
    bi = tneg,
    ci = cpos,
    di = cneg,
    data = dat.bcg
  )

meta_analysis <- rma(yi, vi, data = df, method = "EB")

glance(meta_analysis)
```

---

glance.rq *Glance at a(n) rq object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'rq'
glance(x, ...)
```

Arguments

x	An rq object returned from quantreg::rq().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

Only models with a single tau value may be passed. For multiple values, please use a purrr::map() workflow instead, e.g.

```
taus |>
  map(function(tau_val) rq(y ~ x, tau = tau_val)) |>
  map_dfr(glance)
```

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
tau	Quantile.

See Also

glance(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rq(), augment.rqs(), glance.nlrq(), tidy.nlrq(), tidy.rq(), tidy.rqs()

Examples

```
# load modeling library and data
library(quantreg)

data(stackloss)

# median (l1) regression fit for the stackloss data.
mod1 <- rq(stack.loss ~ stack.x, .5)

# weighted sample median
mod2 <- rq(rnorm(50) ~ 1, weights = runif(50))

# summarize model fit with tidiers
tidy(mod1)
glance(mod1)
augment(mod1)

tidy(mod2)
glance(mod2)
augment(mod2)

# varying tau to generate an rqs object
mod3 <- rq(stack.loss ~ stack.x, tau = c(.25, .5))

tidy(mod3)
augment(mod3)

# glance cannot handle rqs objects like `mod3`--use a purrr
# `map`-based workflow instead
```

`glance.sarlm` *Glance at a(n) spatialreg object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'sarlm'
glance(x, ...)
```

Arguments

x	An object returned from spatialreg::lagsarlm() or spatialreg::errorsarlm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.

See Also

glance(), spatialreg::lagsarlm(), spatialreg::errorsarlm(), spatialreg::sacsarlm()

Other spatialreg tidiers: augment.sarlm(), tidy.sarlm()

Examples

```
# load libraries for models and data
library(spatialreg)
library(spdep)

# load data
data(oldcol, package = "spdep")

listw <- nb2listw(COL.nb, style = "W")

# fit model
crime_sar <-
  lagsarlm(CRIME ~ INC + HOVAL,
```

```
    data = COL.OLD,
    listw = listw,
    method = "eigen"
  )

# summarize model fit with tidiers
tidy(crime_sar)
tidy(crime_sar, conf.int = TRUE)
glance(crime_sar)
augment(crime_sar)

# fit another model
crime_sem <- errorsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sem)
tidy(crime_sem, conf.int = TRUE)
glance(crime_sem)
augment(crime_sem)

# fit another model
crime_sac <- sacsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sac)
tidy(crime_sac, conf.int = TRUE)
glance(crime_sac)
augment(crime_sac)
```

`glance.smooth.spline` *Tidy a(n) smooth.spine object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'smooth.spline'
glance(x, ...)
```

Arguments

`x`	A `smooth.spline` object returned from `stats::smooth.spline()`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`crit`	Minimized criterion
`cv.crit`	Cross-validation score
`df`	Degrees of freedom used by the model.
`lambda`	Choice of lambda corresponding to ‘spar‘.
`nobs`	Number of observations used.
`pen.crit`	Penalized criterion.
`spar`	Smoothing parameter.

See Also

`augment()`, `stats::smooth.spline()`

Other smoothing spline tidiers: `augment.smooth.spline()`

Examples

```
# fit model
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)

# summarize model fit with tidiers
augment(spl, mtcars)

# calls original columns x and y
augment(spl)

library(ggplot2)
ggplot(augment(spl, mtcars), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```

`glance.speedglm` *Glance at a(n) speedglm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'speedglm'
glance(x, ...)
```

Arguments

`x`	A `speedglm` object returned from `speedglm::speedglm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.
`df.null`	Degrees of freedom used by the null model.
`df.residual`	Residual degrees of freedom.

logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
null.deviance	Deviance of the null model.

See Also

speedglm::speedlm()

Other speedlm tidiers: augment.speedlm(), glance.speedlm(), tidy.speedglm(), tidy.speedlm()

Examples

```
# load libraries for models and data
library(speedglm)

# generate data
clotting <- data.frame(
  u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
  lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18)
)

# fit model
fit <- speedglm(lot1 ~ log(u), data = clotting, family = Gamma(log))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

---

glance.speedlm *Glance at a(n) speedlm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'speedlm'
glance(x, ...)
```

Arguments

x	A speedlm object returned from speedglm::speedlm().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

adj.r.squared	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
r.squared	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
statistic	F-statistic.

See Also

speedglm::speedlm()

Other speedlm tidiers: augment.speedlm(), glance.speedglm(), tidy.speedglm(), tidy.speedlm()

Examples

```
# load modeling library
library(speedglm)

# fit model
mod <- speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)

# summarize model fit with tidiers
tidy(mod)
glance(mod)
augment(mod)
```

`glance.summary.lm` *Glance at a(n) summary.lm object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'summary.lm'
glance(x, ...)
```

Arguments

`x`	An `lm` object created by `stats::lm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The `glance.summary.lm()` method is a potentially useful alternative to `glance.lm()`. For instance, if users have already converted large `lm` objects into their leaner `summary.lm` equivalents to conserve memory. Note, however, that this method does not return all of the columns of the non-summary method (e.g. AIC and BIC will be missing.)

Value

A `tibble::tibble()` with exactly one row and columns:

`adj.r.squared`	Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
`df.residual`	Residual degrees of freedom.
`nobs`	Number of observations used.
`p.value`	P-value corresponding to the test statistic.
`r.squared`	R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
`sigma`	Estimated standard error of the residuals.
`statistic`	Test statistic.
`df`	The degrees for freedom from the numerator of the overall F-statistic. This is new in broom 0.7.0. Previously, this reported the rank of the design matrix, which is one more than the numerator degrees of freedom of the overall F-statistic.

See Also

`glance()`, `glance.summary.lm()`

Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.svyglm()`, `tidy.glm()`, `tidy.lm()`, `tidy.lm.beta()`, `tidy.mlm()`, `tidy.summary.lm()`

Examples

```
library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)

tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)

ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()
```

```
# aside: There are tidy() and glance() methods for lm.summary objects too.
# this can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars |>
  head(6) |>
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval

# simpler bivariate model since we're plotting in 2D
mod2 <- lm(mpg ~ wt, data = mtcars)

au <- augment(mod2, newdata = newdata, interval = "prediction")

ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata |>
  select(-mpg)

augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)

ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
```

```
b <- a + rnorm(length(a))
result <- lm(b ~ a)

tidy(result)
```

`glance.survdiff` *Glance at a(n) survdiff object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'survdiff'
glance(x, ...)
```

Arguments

`x`	An `survdiff` object returned from `survival::survdiff()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`df`	Degrees of freedom used by the model.
`p.value`	P-value corresponding to the test statistic.
`statistic`	Test statistic.

See Also

glance(), survival::survdiff()

Other survdiff tidiers: tidy.survdiff()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

# summarize model fit with tidiers
tidy(s)
glance(s)
```

---

glance.survexp *Glance at a(n) survexp object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'survexp'
glance(x, ...)
```

Arguments

x	An survexp object returned from survival::survexp().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

n.max	Maximum number of subjects at risk.
n.start	Initial number of subjects at risk.
timepoints	Number of timepoints.

See Also

glance(), survival::survexp()

Other survexp tidiers: tidy.survexp()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
sexpfit <- survexp(
  futime ~ 1,
  rmap = list(
    sex = "male",
    year = accept.dt,
    age = (accept.dt - birth.dt)
  ),
  method = "conditional",
  data = jasa
)

# summarize model fit with tidiers
```

```
tidy(sexpfit)
glance(sexpfit)
```

`glance.survfit` *Glance at a(n) survfit object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'survfit'
glance(x, ...)
```

Arguments

`x`	An `survfit` object returned from `survival::survfit()`.
`...`	Additional arguments passed to `survival::summary.survfit()`. Important arguments include `rmean`.

Value

A `tibble::tibble()` with exactly one row and columns:

`events`	Number of events.
`n.max`	Maximum number of subjects at risk.
`n.start`	Initial number of subjects at risk.
`nobs`	Number of observations used.
`records`	Number of observations
`rmean`	Restricted mean (see [survival::print.survfit()]).
`rmean.std.error`	Restricted mean standard error.
`conf.low`	lower end of confidence interval on median
`conf.high`	upper end of confidence interval on median
`median`	median survival

See Also

glance(), survival::survfit()

Other cch tidiers: glance.cch(), tidy.cch()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)

# summarize model fit with tidiers + visualization
tidy(sfit)
glance(sfit)

library(ggplot2)

ggplot(tidy(sfit), aes(time, estimate)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)

# multi-state
fitCI <- survfit(Surv(stop, status * as.numeric(event), type = "mstate") ~ 1,
  data = mgus1, subset = (start == 0)
)

td_multi <- tidy(fitCI)

td_multi

ggplot(td_multi, aes(time, estimate, group = state)) +
  geom_line(aes(color = state)) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
```

---

glance.survreg *Glance at a(n) survreg object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'survreg'
glance(x, ...)
```

Arguments

x	An survreg object returned from survival::survreg().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
df	Degrees of freedom used by the model.
df.residual	Residual degrees of freedom.
iter	Iterations of algorithm/fitting procedure completed.
logLik	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs	Number of observations used.
p.value	P-value corresponding to the test statistic.
statistic	Chi-squared statistic.

See Also

glance(), survival::survreg()

Other survreg tidiers: augment.survreg(), tidy.survreg()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx,
  ovarian,
  dist = "exponential"
)

# summarize model fit with tidiers + visualization
tidy(sr)
augment(sr, ovarian)
glance(sr)

# coefficient plot
td <- tidy(sr, conf.int = TRUE)

library(ggplot2)

ggplot(td, aes(estimate, term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
```

---

glance.svyglm *Glance at a(n) svyglm object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'svyglm'
glance(x, maximal = x, ...)
```

Arguments

x	A svyglm object returned from survey::svyglm().
maximal	A svyglm object corresponding to the maximal model against which to compute the BIC. See Lumley and Scott (2015) for details. Defaults to x, which is equivalent to not using a maximal model.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df.null	Degrees of freedom used by the null model.
df.residual	Residual degrees of freedom.
null.deviance	Deviance of the null model.

References

Lumley T, Scott A (2015). AIC and BIC for modelling with complex survey data. *Journal of Survey Statistics and Methodology*, 3(1).

See Also

survey::svyglm(), stats::glm(), survey::anova.svyglm

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

Examples

```
# load libraries for models and data
library(survey)

set.seed(123)
data(api)

# survey design
dstrat <-
  svydesign(
    id = ~1,
    strata = ~stype,
    weights = ~pw,
    data = apistrat,
    fpc = ~fpc
  )

# model
m <- svyglm(
  formula = sch.wide ~ ell + meals + mobility,
  design = dstrat,
  family = quasibinomial()
)

glance(m)
```

---

glance.svyolr *Glance at a(n) svyolr object*

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```
## S3 method for class 'svyolr'
glance(x, ...)
```

Arguments

x	A svyolr object returned from survey::svyolr().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

df.residual	Residual degrees of freedom.
edf	The effective degrees of freedom.
nobs	Number of observations used.

See Also

tidy, survey::svyolr()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clm(), glance.clmm(), glance.polr(), tidy.clm(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
library(broom)
library(survey)

data(api)
dclus1 <- svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
dclus1 <- update(dclus1, mealcat = cut(meals, c(0, 25, 50, 75, 100)))

m <- svyolr(mealcat ~ avg.ed + mobility + stype, design = dclus1)

m
```

```
tidy(m, conf.int = TRUE)
```

`glance.varest` *Glance at a(n) varest object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```
## S3 method for class 'varest'
glance(x, ...)
```

Arguments

`x`	A `varest` object produced by a call to `vars::VAR()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with exactly one row and columns:

`lag.order`	Lag order.
`logLik`	The log-likelihood of the model. [stats::logLik()] may be a useful reference.
`n`	The total number of observations.
`nobs`	Number of observations used.

See Also

glance(), vars::VAR()

Examples

```
# load libraries for models and data
library(vars)

# load data
data("Canada", package = "vars")

# fit models
mod <- VAR(Canada, p = 1, type = "both")

# summarize model fit with tidiers
tidy(mod)
glance(mod)
```

---

glance_optim *Tidy a(n) optim object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), svd() and interp::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, they throw an error.

Usage

```
glance_optim(x, ...)
```

Arguments

x	A list returned from stats::optim().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.

- augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with exactly one row and columns:

convergence	Convergence code.
function.count	Number of calls to 'fn'.
gradient.count	Number of calls to 'gr'.
value	Minimized or maximized output value.

See Also

glance(), stats::optim()

Other list tidiers: list_tidiers, tidy_irlba(), tidy_optim(), tidy_svd(), tidy_xyz()

Examples

```
f <- function(x) (x[1] - 2)^2 + (x[2] - 3)^2 + (x[3] - 8)^2
o <- optim(c(1, 1, 1), f)
```

---

leveneTest_tidiers *Tidy/glance a(n) leveneTest object*

Description

For models that have only a single component, the tidy() and glance() methods are identical. Please see the documentation for both of those methods.

Usage

```
## S3 method for class 'leveneTest'
tidy(x, ...)
```

Arguments

x	An object of class anova created by a call to car::leveneTest().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

df	Degrees of freedom used by this term in the model.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
df.residual	Residual degrees of freedom.

See Also

tidy(), glance(), car::leveneTest()

Other car tidiers: durbinWatsonTest_tidiers

Examples

```
# load libraries for models and data
library(car)

data(Moore)

lt <- with(Moore, leveneTest(conformity, fcategory))

tidy(lt)
glance(lt)
```

---

list_tidiers *Tidying methods for lists / returned values that are not S3 objects*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), base::svd() and interp::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

Usage

```
## S3 method for class 'list'
tidy(x, ...)

## S3 method for class 'list'
glance(x, ...)
```

Arguments

x	A list, potentially representing an object that can be tidied.
...	Additionally, arguments passed to the tidying function.

Details

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

See Also

Other list tidiers: `glance_optim()`, `tidy_irlba()`, `tidy_optim()`, `tidy_svd()`, `tidy_xyz()`

---

null_tidiers *Tidiers for NULL inputs*

Description

`tidy(NULL)`, `glance(NULL)` and `augment(NULL)` all return an empty tibble::tibble. This empty tibble can be treated a tibble with zero rows, making it convenient to combine with other tibbles using functions like `purrr::map_df()` on lists of potentially `NULL` objects.

Usage

```
## S3 method for class '`NULL`'
tidy(x, ...)

## S3 method for class '`NULL`'
glance(x, ...)

## S3 method for class '`NULL`'
augment(x, ...)
```

Arguments

x	The value `NULL`.
...	Additional arguments (not used).

Value

An empty tibble::tibble.

See Also

tibble::tibble

sp_tidiers *Tidy a(n) SpatialPolygonsDataFrame object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Note that the `sf` package now defines tidy spatial objects and is the recommended approach to spatial data. sp tidiers are now deprecated in favor of `sf::st_as_sf()` and coercion methods found in other packages. See https://r-spatial.org/r/2023/05/15/evolution4.html for more on migration from retiring spatial packages.

Usage

```
## S3 method for class 'SpatialPolygonsDataFrame'
tidy(x, region = NULL, ...)

## S3 method for class 'SpatialPolygons'
tidy(x, ...)

## S3 method for class 'Polygons'
tidy(x, ...)

## S3 method for class 'Polygon'
tidy(x, ...)

## S3 method for class 'SpatialLinesDataFrame'
tidy(x, ...)

## S3 method for class 'Lines'
tidy(x, ...)

## S3 method for class 'Line'
tidy(x, ...)
```

Arguments

x	A `SpatialPolygonsDataFrame`, `SpatialPolygons`, `Polygons`, `Polygon`, `SpatialLinesDataFrame`, `Lines` or `Line` object.
region	name of variable used to split up regions
...	not used by this method

summary_tidiers *(Deprecated) Tidy summaryDefault objects*

Description

Tidiers for summaryDefault objects have been deprecated as of broom 0.7.0 in favor of `skimr::skim()`.

Usage

```
## S3 method for class 'summaryDefault'
tidy(x, ...)

## S3 method for class 'summaryDefault'
glance(x, ...)
```

Arguments

`x`	A `summaryDefault` object, created by calling `summary()` on a vector.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A one-row tibble::tibble with columns:

`minimum`	Minimum value in original vector.
`q1`	First quartile of original vector.
`median`	Median of original vector.
`mean`	Mean of original vector.
`q3`	Third quartile of original vector.
`maximum`	Maximum value in original vector.
`na`	Number of `NA` values in original vector. Column present only when original vector had at least one `NA` entry.

See Also

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

Examples

```
v <- rnorm(1000)
s <- summary(v)
s

tidy(s)
glance(s)

v2 <- c(v,NA)
tidy(summary(v2))
```

`tidy.aareg` *Tidy a(n) aareg object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'aareg'
tidy(x, ...)
```

Arguments

`x`	An `aareg` object returned from `survival::aareg()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

`robust.se` is only present when `x` was created with `dfbeta = TRUE`.

Value

A tibble::tibble() with columns:

estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
robust.se	robust version of standard error estimate.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
z	z score.

See Also

tidy(), survival::aareg()

Other aareg tidiers: glance.aareg()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
afit <- aareg(
  Surv(time, status) ~ age + sex + ph.ecog,
  data = lung,
  dfbeta = TRUE
)

# summarize model fit with tidiers
tidy(afit)
```

---

tidy.acf *Tidy a(n) acf object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'acf'
tidy(x, ...)
```

Arguments

x	An acf object created by stats::acf(), stats::pacf() or stats::ccf().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

acf	Autocorrelation.
lag	Lag values.

See Also

tidy(), stats::acf(), stats::pacf(), stats::ccf()

Other time series tidiers: tidy.spec(), tidy.ts(), tidy.zoo()

Examples

```
tidy(acf(lh, plot = FALSE))
tidy(ccf(mdeaths, fdeaths, plot = FALSE))
tidy(pacf(lh, plot = FALSE))
```

---

tidy.anova *Tidy a(n) anova object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'anova'
tidy(x, ...)
```

Arguments

x	An anova object, such as those created by stats::anova(), car::Anova(), car::leveneTest(), or car::linearHypothesis().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

The term column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for car::leveneTest() output, see tidy.leveneTest()

Value

A tibble::tibble() with columns:

df	Degrees of freedom used by this term in the model.
meansq	Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
sumsq	Sum of squares explained by this term.
term	The name of the regression term.

See Also

tidy(), stats::anova(), car::Anova(), car::leveneTest()

Other anova tidiers: glance.anova(), glance.aov(), tidy.TukeyHSD(), tidy.aov(), tidy.aovlist(), tidy.manova()

Examples

```
# fit models
a <- lm(mpg ~ wt + qsec + disp, mtcars)
b <- lm(mpg ~ wt + qsec, mtcars)

mod <- anova(a, b)

# summarize model fit with tidiers
tidy(mod)
glance(mod)

# car::linearHypothesis() example
library(car)
mod_lht <- linearHypothesis(a, "wt - disp")
tidy(mod_lht)
glance(mod_lht)
```

`tidy.aov` *Tidy a(n) aov object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'aov'
tidy(x, intercept = FALSE, ...)
```

Arguments

`x`	An aov object, such as those created by `stats::aov()`.
`intercept`	A logical indicating whether information on the intercept ought to be included. Passed to `stats::summary.aov()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The `term` column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for `car::leveneTest()` output, see `tidy.leveneTest()`

See Also

`tidy()`, `stats::aov()`

Other anova tidiers: `glance.anova()`, `glance.aov()`, `tidy.TukeyHSD()`, `tidy.anova()`, `tidy.aovlist()`, `tidy.manova()`

Examples

```
a <- aov(mpg ~ wt + qsec + disp, mtcars)
tidy(a)
```

---

tidy.aovlist *Tidy a(n) aovlist object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'aovlist'
tidy(x, ...)
```

Arguments

`x`	An `aovlist` objects, such as those created by `stats::aov()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The `term` column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for `car::leveneTest()` output, see `tidy.leveneTest()`

Value

A `tibble::tibble()` with columns:

`df`	Degrees of freedom used by this term in the model.
`meansq`	Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`stratum`	The error stratum.
`sumsq`	Sum of squares explained by this term.
`term`	The name of the regression term.

See Also

`tidy()`, `stats::aov()`

Other anova tidiers: `glance.anova()`, `glance.aov()`, `tidy.TukeyHSD()`, `tidy.anova()`, `tidy.aov()`, `tidy.manova()`

Examples

```
a <- aov(mpg ~ wt + qsec + Error(disp / am), mtcars)
tidy(a)
```

`tidy.Arima` *Tidy a(n) Arima object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'Arima'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	An object of class Arima created by stats::arima().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

stats::arima()

Other Arima tidiers: glance.Arima()

Examples

```
# fit model
fit <- arima(lh, order = c(1, 0, 0))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

`tidy.betamfx` *Tidy a(n) betamfx object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'betamfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `betamfx` object.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The `mfx` package provides methods for calculating marginal effects for various generalized linear models (GLMs). Unlike standard linear models, estimated model coefficients in a GLM cannot be directly interpreted as marginal effects (i.e., the change in the response variable predicted after a one unit change in one of the regressors). This is because the estimated coefficients are multiplicative, dependent on both the link function that was used for the estimation and any other variables that were included in the model. When calculating marginal effects, users must typically choose whether they want to use i) the average observation in the data, or ii) the average of the sample marginal effects. See `vignette("mfxarticle")` from the `mfx` package for more details.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
atmean	TRUE if the marginal effects were originally calculated as the partial effects for the average observation. If FALSE, then these were instead calculated as average partial effects.

See Also

tidy.betareg(), mfx::betamfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.betamfx(), glance.mfx(), tidy.mfx()

Examples

```
library(mfx)

# Simulate some data
set.seed(12345)
n <- 1000
x <- rnorm(n)

# Beta outcome
y <- rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2 * x)))
# Use Smithson and Verkuilen correction
y <- (y * (n - 1) + 0.5) / n

d <- data.frame(y, x)
mod_betamfx <- betamfx(y ~ x | x, data = d)

tidy(mod_betamfx, conf.int = TRUE)

# Compare with the naive model coefficients of the equivalent betareg call (not run)
# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)
glance(mod_betamfx)
```

`tidy.betareg` *Tidy a(n) betareg object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'betareg'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `betareg` object produced by a call to `betareg::betareg()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The tibble has one row for each term in the regression. The `component` column indicates whether a particular term was used to model either the `"mean"` or `"precision"`. Here the precision is the inverse of the variance, often referred to as `phi`. At least one term will have been used to model the precision `phi`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.

estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
component	Whether a particular term was used to model the mean or the precision in the regression. See details.

See Also

tidy(), betareg::betareg()

Examples

```
# load libraries for models and data
library(betareg)

# load dats
data("GasolineYield", package = "betareg")

# fit model
mod <- betareg(yield ~ batch + temp, data = GasolineYield)

mod

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)

augment(mod)

glance(mod)
```

---

tidy.biglm *Tidy a(n) biglm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'biglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

x	A biglm object created by a call to biglm::biglm() or biglm::bigglm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), biglm::biglm(), biglm::bigglm()

Other biglm tidiers: glance.biglm()

Examples

```
# load modeling library
library(biglm)

# fit model -- linear regression
bfit <- biglm(mpg ~ wt + disp, mtcars)

# summarize model fit with tidiers
tidy(bfit)
tidy(bfit, conf.int = TRUE)
tidy(bfit, conf.int = TRUE, conf.level = .9)

glance(bfit)

# fit model -- logistic regression
bgfit <- bigglm(am ~ mpg, mtcars, family = binomial())

# summarize model fit with tidiers
tidy(bgfit)
tidy(bgfit, exponentiate = TRUE)
tidy(bgfit, conf.int = TRUE)
tidy(bgfit, conf.int = TRUE, conf.level = .9)
tidy(bgfit, conf.int = TRUE, conf.level = .9, exponentiate = TRUE)

glance(bgfit)
```

---

tidy.binDesign *Tidy a(n) binDesign object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'binDesign'
tidy(x, ...)
```

Arguments

x	A binGroup::binDesign() object.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`n`	Number of trials in given iteration.
`power`	Power achieved for given value of n.

See Also

`tidy()`, `binGroup::binDesign()`

Other bingroup tidiers: `glance.binDesign()`, `tidy.binWidth()`

Examples

```
library(binGroup)
des <- binDesign(
  nmax = 300, delta = 0.06,
  p.hyp = 0.1, power = .8
)

glance(des)
tidy(des)

# the ggplot2 equivalent of plot(des)
library(ggplot2)
ggplot(tidy(des), aes(n, power)) +
  geom_line()
```

`tidy.binWidth` *Tidy a(n) binWidth object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'binWidth'
tidy(x, ...)
```

Arguments

x	A binGroup::binWidth() object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

alternative	Alternative hypothesis (character).
ci.width	Expected width of confidence interval.
p	True proportion.
n	Total sample size

See Also

tidy(), binGroup::binWidth()

Other bingroup tidiers: glance.binDesign(), tidy.binDesign()

Examples

```
# load libraries
library(binGroup)

# fit model
bw <- binWidth(100, .1)

bw

# summarize model fit with tidiers
tidy(bw)
```

`tidy.boot` *Tidy a(n) boot object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'boot'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = c("perc", "bca", "basic", "norm"),
  exponentiate = FALSE,
  ...
)
```

Arguments

`x`	A `boot::boot()` object.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`conf.method`	Passed to the `type` argument of `boot::boot.ci()`. Defaults to `"perc"`. The allowed types are `"perc"`, `"basic"`, `"bca"`, and `"norm"`. Does not support `"stud"` or `"all"`.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

If weights were provided to the `boot` function, an `estimate` column is included showing the weighted bootstrap estimate, and the standard error is of that estimate.

If there are no original statistics in the "boot" object, such as with a call to `tsboot` with `orig.t = FALSE`, the `original` and `statistic` columns are omitted, and only `estimate` and `std.error` columns shown.

Value

A `tibble::tibble()` with columns:

`bias`	Bias of the statistic.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.
`statistic`	Original value of the statistic.

See Also

`tidy()`, `boot::boot()`, `boot::tsboot()`, `boot::boot.ci()`, `rsample::bootstraps()`

Examples

```
# load modeling library
library(boot)

clotting <- data.frame(
  u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
  lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18),
  lot2 = c(69, 35, 26, 21, 18, 16, 13, 12, 12)
)

# fit models
g1 <- glm(lot2 ~ log(u), data = clotting, family = Gamma)

bootfun <- function(d, i) {
  coef(update(g1, data = d[i, ]))
}

bootres <- boot(clotting, bootfun, R = 999)

# summarize model fits with tidiers
tidy(g1, conf.int = TRUE)
tidy(bootres, conf.int = TRUE)
```

`tidy.btergm` *Tidy a(n) btergm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

This method tidies the coefficients of a bootstrapped temporal exponential random graph model estimated with the **xergm**. It simply returns the coefficients and their confidence intervals.

Usage

```
## S3 method for class 'btergm'
tidy(x, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `btergm::btergm()` object.
`conf.level`	Confidence level for confidence intervals. Defaults to 0.95.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `btergm::btergm()`

Examples

```
library(btergm)
library(network)

set.seed(5)

# create 10 random networks with 10 actors
networks <- list()
for (i in 1:10) {
  mat <- matrix(rbinom(100, 1, .25), nrow = 10, ncol = 10)
  diag(mat) <- 0
  nw <- network(mat)
  networks[[i]] <- nw
}

# create 10 matrices as covariates
covariates <- list()
for (i in 1:10) {
  mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
  covariates[[i]] <- mat
}

# fit the model
mod <- btergm(networks ~ edges + istar(2) + edgecov(covariates), R = 100)

# summarize model fit with tidiers
tidy(mod)
```

`tidy.cch` *Tidy a(n) cch object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'cch'
tidy(x, conf.level = 0.95, ...)
```

Arguments

`x`	An cch object returned from `survival::cch()`.
`conf.level`	confidence level for CI

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `survival::cch()`

Other cch tidiers: `glance.cch()`, `glance.survfit()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
# load libraries for models and data
library(survival)

# examples come from cch documentation
subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel == 1 | subcoh == 1)
ccoh.data <- nwtco[selccoh, ]
ccoh.data$subcohort <- subcoh[selccoh]

# central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol, labels = c("FH", "UH"))

# tumour stage
```

```
ccoh.data$stage <- factor(ccoh.data$stage, labels = c("I", "II", "III", "IV"))
ccoh.data$age <- ccoh.data$age / 12 # age in years

# fit model
fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age,
  data = ccoh.data,
  subcoh = ~subcohort, id = ~seqno, cohort.size = 4028
)

# summarize model fit with tidiers + visualization
tidy(fit.ccP)

# coefficient plot
library(ggplot2)

ggplot(tidy(fit.ccP), aes(x = estimate, y = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
```

`tidy.cld` *Tidy a(n) cld object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'cld'
tidy(x, ...)
```

Arguments

`x`	A `cld` object created by calling `multcomp::cld()` on a `glht`, `confint.glht()` or `summary.glht()` object.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

contrast	Levels being compared.
letters	Compact letter display denoting all pair-wise comparisons.

See Also

tidy(), multcomp::cld(), multcomp::summary.glht(), multcomp::confint.glht(), multcomp::glht()

Other multcomp tidiers: tidy.confint.glht(), tidy.glht(), tidy.summary.glht()

Examples

```
# load libraries for models and data
library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

tidy(wht)

ggplot(wht, aes(lhs, estimate)) +
  geom_point()

CI <- confint(wht)

tidy(CI)

ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
  geom_pointrange()

tidy(summary(wht))
ggplot(mapping = aes(lhs, estimate)) +
  geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
  geom_point(aes(size = p), data = summary(wht)) +
  scale_size(trans = "reverse")

cld <- cld(wht)
tidy(cld)
```

---

tidy.clm *Tidy a(n) clm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'clm'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.type = c("profile", "Wald"),
  exponentiate = FALSE,
  ...
)
```

Arguments

x	A clm object returned from ordinal::clm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
conf.type	Whether to use "profile" or "Wald" confidence intervals, passed to the type argument of ordinal::confint.clm(). Defaults to "profile".
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

In broom 0.7.0 the coefficient_type column was renamed to coef.type, and the contents were changed as well.

Note that intercept type coefficients correspond to alpha parameters, location type coefficients correspond to beta parameters, and scale type coefficients correspond to zeta parameters.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy, ordinal::clm(), ordinal::confint.clm()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clm(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(ordinal)

# fit model
fit <- clm(rating ~ temp * contact, data = wine)

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

# ...and again with another model specification
fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)

tidy(fit2)
glance(fit2)
```

`tidy.clmm` *Tidy a(n) clmm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'clmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `clmm` object returned from `ordinal::clmm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.

statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

Note

In broom 0.7.0 the coefficient_type column was renamed to coef.type, and the contents were changed as well.

Note that intercept type coefficients correspond to alpha parameters, location type coefficients correspond to beta parameters, and scale type coefficients correspond to zeta parameters.

See Also

tidy, ordinal::clmm(), ordinal::confint.clm()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clm(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.polr(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(ordinal)

# fit model
fit <- clmm(rating ~ temp + contact + (1 | judge), data = wine)

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, exponentiate = TRUE)

glance(fit)

# ...and again with another model specification
fit2 <- clmm(rating ~ temp + (1 | judge), nominal = ~contact, data = wine)

tidy(fit2)
glance(fit2)
```

---

tidy.coeftest *Tidy a(n) coeftest object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'coeftest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A coeftest object returned from lmtest::coeftest().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), lmtest::coeftest()

Examples

```
# load libraries for models and data
library(lmtest)

m <- lm(dist ~ speed, data = cars)
```

```
coeftest(m)
tidy(coeftest(m))
tidy(coeftest(m, conf.int = TRUE))

# a very common workflow is to combine lmtest::coeftest with alternate
# variance-covariance matrices via the sandwich package. The lmtest
# tidiers support this workflow too, enabling you to adjust the standard
# errors of your tidied models on the fly.
library(sandwich)

# "HC3" (default) robust SEs
tidy(coeftest(m, vcov = vcovHC))

# "HC2" robust SEs
tidy(coeftest(m, vcov = vcovHC, type = "HC2"))

# N-W HAC robust SEs
tidy(coeftest(m, vcov = NeweyWest))

# the columns of the returned tibble for glance.coeftest() will vary
# depending on whether the coeftest object retains the underlying model.
# Users can control this with the "save = TRUE" argument of coeftest().
glance(coeftest(m))
glance(coeftest(m, save = TRUE))
```

`tidy.confint.glht` *Tidy a(n) confint.glht object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'confint.glht'
tidy(x, ...)
```

Arguments

`x`	A `confint.glht` object created by calling `multcomp::confint.glht()` on a `glht` object created with `multcomp::glht()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`contrast`	Levels being compared.
`estimate`	The estimated value of the regression term.

See Also

`tidy()`, `multcomp::confint.glht()`, `multcomp::glht()`

Other multcomp tidiers: `tidy.cld()`, `tidy.glht()`, `tidy.summary.glht()`

Examples

```
# load libraries for models and data
library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

tidy(wht)

ggplot(wht, aes(lhs, estimate)) +
  geom_point()

CI <- confint(wht)

tidy(CI)

ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
  geom_pointrange()

tidy(summary(wht))
ggplot(mapping = aes(lhs, estimate)) +
  geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
  geom_point(aes(size = p), data = summary(wht)) +
  scale_size(trans = "reverse")

cld <- cld(wht)
```

```
tidy(cld)
```

`tidy.confusionMatrix` *Tidy a(n) confusionMatrix object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'confusionMatrix'
tidy(x, by_class = TRUE, ...)
```

Arguments

`x`	An object of class `confusionMatrix` created by a call to `caret::confusionMatrix()`.
`by_class`	Logical indicating whether or not to show performance measures broken down by class. Defaults to `TRUE`. When `by_class = FALSE` only returns a tibble with accuracy, kappa, and McNemar statistics.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`class`	The class under consideration.
`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`term`	The name of the regression term.
`p.value`	P-value for accuracy and kappa statistics.

See Also

tidy(), caret::confusionMatrix()

Examples

```
# load libraries for models and data
library(caret)

set.seed(27)

# generate data
two_class_sample1 <- as.factor(sample(letters[1:2], 100, TRUE))
two_class_sample2 <- as.factor(sample(letters[1:2], 100, TRUE))

two_class_cm <- confusionMatrix(
  two_class_sample1,
  two_class_sample2
)

# summarize model fit with tidiers
tidy(two_class_cm)
tidy(two_class_cm, by_class = FALSE)

# multiclass example
six_class_sample1 <- as.factor(sample(letters[1:6], 100, TRUE))
six_class_sample2 <- as.factor(sample(letters[1:6], 100, TRUE))

six_class_cm <- confusionMatrix(
  six_class_sample1,
  six_class_sample2
)

# summarize model fit with tidiers
tidy(six_class_cm)
tidy(six_class_cm, by_class = FALSE)
```

---

tidy.coxph *Tidy a(n) coxph object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'coxph'
tidy(x, exponentiate = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A coxph object returned from survival::coxph().
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	For tidy(), additional arguments passed to summary(x, ...). Otherwise ignored.

Value

A tibble::tibble() with columns:

estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.

See Also

tidy(), survival::coxph()

Other coxph tidiers: augment.coxph(), glance.coxph()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
cfit <- coxph(Surv(time, status) ~ age + sex, lung)

# summarize model fit with tidiers
```

```
tidy(cfit)
tidy(cfit, exponentiate = TRUE)

lp <- augment(cfit, lung)
risks <- augment(cfit, lung, type.predict = "risk")
expected <- augment(cfit, lung, type.predict = "expected")

glance(cfit)

# also works on clogit models
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(
  logan[indx, ],
  id = indx,
  tocc = factor(rep(resp, each = n))
)

logan2$case <- (logan2$occupation == logan2$tocc)

cl <- clogit(case ~ tocc + tocc:education + strata(id), logan2)

tidy(cl)
glance(cl)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(risks, aes(age, .fitted, color = sex)) +
  geom_point()

ggplot(expected, aes(time, .fitted, color = sex)) +
  geom_point()
```

`tidy.crr` *Tidy a(n) cmprsk object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'crr'
tidy(x, exponentiate = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A crr object returned from cmprsk::crr().
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.

See Also

tidy(), cmprsk::crr()

Other cmprsk tidiers: glance.crr()

Examples

```
library(cmprsk)

# time to loco-regional failure (lrf)
lrf_time <- rexp(100)
lrf_event <- sample(0:2, 100, replace = TRUE)
trt <- sample(0:1, 100, replace = TRUE)
strt <- sample(1:2, 100, replace = TRUE)

# fit model
x <- crr(lrf_time, lrf_event, cbind(trt, strt))

# summarize model fit with tidiers
tidy(x, conf.int = TRUE)
glance(x)
```

`tidy.cv.glmnet` *Tidy a(n) cv.glmnet object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'cv.glmnet'
tidy(x, ...)
```

Arguments

`x`	A `cv.glmnet` object returned from `glmnet::cv.glmnet()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

lambda	Value of penalty parameter lambda.
nzero	Number of non-zero coefficients for the given lambda.
std.error	The standard error of the regression term.
conf.low	lower bound on confidence interval for cross-validation estimated loss.
conf.high	upper bound on confidence interval for cross-validation estimated loss.
estimate	Median loss across all cross-validation folds for a given lamdba

See Also

tidy(), glmnet::cv.glmnet()

Other glmnet tidiers: glance.cv.glmnet(), glance.glmnet(), tidy.glmnet()

Examples

```
# load libraries for models and data
library(glmnet)

set.seed(27)

nobs <- 100
nvar <- 50
real <- 5

x <- matrix(rnorm(nobs * nvar), nobs, nvar)
beta <- c(rnorm(real, 0, 1), rep(0, nvar - real))
y <- c(t(beta) %*% t(x)) + rnorm(nvar, sd = 3)

cvfit1 <- cv.glmnet(x, y)

tidy(cvfit1)
glance(cvfit1)

library(ggplot2)

tidied_cv <- tidy(cvfit1)
glance_cv <- glance(cvfit1)

# plot of MSE as a function of lambda
g <- ggplot(tidied_cv, aes(lambda, estimate)) +
  geom_line() +
  scale_x_log10()
g

# plot of MSE as a function of lambda with confidence ribbon
g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
g
```

```
# plot of MSE as a function of lambda with confidence ribbon and choices
# of minimum lambda marked
g <- g +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
g

# plot of number of zeros for each choice of lambda
ggplot(tidied_cv, aes(lambda, nzero)) +
  geom_line() +
  scale_x_log10()

# coefficient plot with min lambda shown
tidied <- tidy(cvfit1$glmnet.fit)

ggplot(tidied, aes(lambda, estimate, group = term)) +
  scale_x_log10() +
  geom_line() +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
```

`tidy.density` *(Deprecated) Tidy density objects*

Description

(Deprecated) Tidy density objects

Usage

```
## S3 method for class 'density'
tidy(x, ...)
```

Arguments

`x`	A `density` object returned from `stats::density()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble with two columns: points x where the density is estimated, and estimated density y.

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.dist(), tidy.ftable(), tidy.numeric()

---

tidy.dist *(Deprecated) Tidy dist objects*

Description

(Deprecated) Tidy dist objects

Usage

```
## S3 method for class 'dist'
tidy(x, diagonal = attr(x, "Diag"), upper = attr(x, "Upper"), ...)
```

Arguments

x	A dist object returned from stats::dist().
diagonal	Logical indicating whether or not to tidy the diagonal elements of the distance matrix. Defaults to whatever was based to the diag argument of stats::dist().
upper	Logical indicating whether or not to tidy the upper half of the distance matrix. Defaults to whatever was based to the upper argument of stats::dist().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

If the distance matrix does not include an upper triangle and/or diagonal, the tidied version will not either.

Value

A tibble::tibble with one row for each pair of items in the distance matrix, with columns:

item1	First item
item2	Second item
distance	Distance between items

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.ftable(), tidy.numeric()

Examples

```
cars_dist <- dist(t(mtcars[, 1:4]))
cars_dist

tidy(cars_dist)
tidy(cars_dist, upper = TRUE)
tidy(cars_dist, diagonal = TRUE)
```

---

tidy.drc *Tidy a(n) drc object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'drc'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A drc object produced by a call to drc::drm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The tibble has one row for each curve and term in the regression. The `curveid` column indicates the curve.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.
`curve`	Index identifying the curve.

See Also

`tidy()`, `drc::drm()`

Other drc tidiers: `augment.drc()`, `glance.drc()`

Examples

```
# load libraries for models and data
library(drc)

# fit model
mod <- drm(dead / total ~ conc, type,
  weights = total, data = selenium, fct = LL.2(), type = "binomial"
)

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)
```

```
glance(mod)

augment(mod, selenium)
```

`tidy.emmGrid` *Tidy a(n) emmGrid object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'emmGrid'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `emmGrid` object.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int` = `TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`. **Cautionary note**: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.

df	Degrees of freedom used by this term in the model.
p.value	The two-sided p-value associated with the observed statistic.
std.error	The standard error of the regression term.
estimate	Expected marginal mean
statistic	T-ratio statistic

See Also

tidy(), emmeans::ref_grid(), emmeans::emmeans(), emmeans::contrast()

Other emmeans tidiers: tidy.lsmobj(), tidy.ref.grid(), tidy.summary_emm()

Examples

```
# load libraries for models and data
library(emmeans)

# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)
td <- tidy(oranges_rg1)
td

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)

ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)

by_price
```

```
tidy(by_price)

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))
```

`tidy.epi.2by2` *Tidy a(n) epi.2by2 object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'epi.2by2'
tidy(x, parameters = c("moa", "stat"), ...)
```

Arguments

`x`	A `epi.2by2` object produced by a call to `epiR::epi.2by2()`
`parameters`	Return measures of association (`moa`) or test statistics (`stat`), default is `moa` (measures of association)
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The tibble has a column for each of the measures of association or tests contained in `massoc` or `massoc.detail` when `epiR::epi.2by2()` is called.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
df	Degrees of freedom used by this term in the model.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
term	The name of the regression term.
estimate	Estimated measure of association

See Also

tidy(), epiR::epi.2by2()

Examples

```
# load libraries for models and data
library(epiR)

# generate data
dat <- matrix(c(13, 2163, 5, 3349), nrow = 2, byrow = TRUE)

rownames(dat) <- c("DF+", "DF-")
colnames(dat) <- c("FUS+", "FUS-")

# fit model
fit <- epi.2by2(
  dat = as.table(dat), method = "cross.sectional",
  conf.level = 0.95, units = 100, outcome = "as.columns"
)

# summarize model fit with tidiers
tidy(fit, parameters = "moa")
tidy(fit, parameters = "stat")
```

---

tidy.ergm *Tidy a(n) ergm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

The methods should work with any model that conforms to the **ergm** class, such as those produced from weighted networks by the **ergm.count** package.

Usage

```
## S3 method for class 'ergm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	An `ergm` object returned from a call to `ergm::ergm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments to pass to `ergm::summary()`. **Cautionary note**: Misspecified arguments may be silently ignored.

Value

A tibble::tibble with one row for each coefficient in the exponential random graph model, with columns:

`term`	The term in the model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error
`mcmc.error`	The MCMC error
`p.value`	The two-sided p-value

References

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008b). **ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. *Journal of Statistical Software*, 24(3). https://www.jstatsoft.org/v24/i03/.

See Also

tidy(), ergm::ergm(), ergm::control.ergm(), ergm::summary()

Other ergm tidiers: glance.ergm()

Examples

```
# load libraries for models and data
library(ergm)

# load the Florentine marriage network data
data(florentine)

# fit a model where the propensity to form ties between
# families depends on the absolute difference in wealth
gest <- ergm(flomarriage ~ edges + absdiff("wealth"))

# show terms, coefficient estimates and errors
tidy(gest)

# show coefficients as odds ratios with a 99% CI
tidy(gest, exponentiate = TRUE, conf.int = TRUE, conf.level = 0.99)

# take a look at likelihood measures and other
# control parameters used during MCMC estimation
glance(gest)
glance(gest, deviance = TRUE)
glance(gest, mcmc = TRUE)
```

---

tidy.factanal *Tidy a(n) factanal object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'factanal'
tidy(x, ...)
```

Arguments

x	A factanal object created by stats::factanal().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

variable	Variable under consideration.
uniqueness	Proportion of residual, or unexplained variance
flX	Factor loading for level X.

See Also

tidy(), stats::factanal()

Other factanal tidiers: augment.factanal(), glance.factanal()

Examples

```
set.seed(123)

# generate data
library(dplyr)
library(purrr)

m1 <- tibble(
  v1 = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 4, 5, 6),
  v2 = c(1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 3, 4, 3, 3, 3, 4, 6, 5),
  v3 = c(3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 4, 6),
  v4 = c(3, 3, 4, 3, 3, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 5, 6, 4),
  v5 = c(1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 6, 4, 5),
  v6 = c(1, 1, 1, 2, 1, 3, 3, 3, 4, 3, 1, 1, 1, 2, 1, 6, 5, 4)
)

# new data
m2 <- map_dfr(m1, rev)

# factor analysis objects
fit1 <- factanal(m1, factors = 3, scores = "Bartlett")
fit2 <- factanal(m1, factors = 3, scores = "regression")
```

```
# tidying the object
tidy(fit1)
tidy(fit2)

# augmented dataframe
augment(fit1)
augment(fit2)

# augmented dataframe (with new data)
augment(fit1, data = m2)
augment(fit2, data = m2)
```

`tidy.felm` *Tidy a(n) felm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'felm'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  fe = FALSE,
  se.type = c("default", "iid", "robust", "cluster"),
  ...
)
```

Arguments

`x`	A `felm` object returned from `lfe::felm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`fe`	Logical indicating whether or not to include estimates of fixed effects. Defaults to `FALSE`.

se.type	Character indicating the type of standard errors. Defaults to using those of the underlying felm() model object, e.g. clustered errors for models that were provided a cluster specification. Users can override these defaults by specifying an appropriate alternative: "iid" (for homoskedastic errors), "robust" (for Eicker-Huber-White robust errors), or "cluster" (for clustered standard errors; if the model object supports it).
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), lfe::felm()

Other felm tidiers: augment.felm()

Examples

```
# load libraries for models and data
library(lfe)

# use built-in `airquality` dataset
head(airquality)

# no FEs; same as lm()
est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)

# summarize model fit with tidiers
```

```
tidy(est0)
augment(est0)

# add month fixed effects
est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)

# summarize model fit with tidiers
tidy(est1)
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# the "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)

# summarize model fit with tidiers
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")
```

`tidy.fitdistr` *Tidy a(n) fitdistr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'fitdistr'
tidy(x, ...)
```

Arguments

`x`	A `fitdistr` object returned by `MASS::fitdistr()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `MASS::fitdistr()`

Other fitdistr tidiers: `glance.fitdistr()`

Examples

```
# load libraries for models and data
library(MASS)

# generate data
set.seed(2015)
x <- rnorm(100, 5, 2)

#  fit models
fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

---

tidy.fixest *Tidy a(n) fixest object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'fixest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A fixest object returned from any of the fixest estimators
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments passed to summary and confint. Important arguments are se and cluster. Other arguments are dof, exact_dof, forceCovariance, and keepBounded. See summary.fixest.

Details

The fixest package provides a family of functions for estimating models with arbitrary numbers of fixed-effects, in both an OLS and a GLM context. The package also supports robust (i.e. White) and clustered standard error reporting via the generic summary.fixest() command. In a similar vein, the tidy() method for these models allows users to specify a desired standard error correction either 1) implicitly via the supplied fixest object, or 2) explicitly as part of the tidy call. See examples below.

Note that fixest confidence intervals are calculated assuming a normal distribution – this assumes infinite degrees of freedom for the CI. (This assumption is distinct from the degrees of freedom used to calculate the standard errors. For more on degrees of freedom with clusters and fixed effects, see https://github.com/lrberge/fixest/issues/6 and https://github.com/sgaure/lfe/issues/1#issuecomment-530646990)

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), fixest::feglm(), fixest::fenegbin(), fixest::feNmlm(), fixest::femlm(), fixest::feols(), fixest::fepois()

Other fixest tidiers: augment.fixest()

Examples

```
# load libraries for models and data
library(fixest)

gravity <-
  feols(
    log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade
  )

tidy(gravity)
glance(gravity)
augment(gravity, trade)

# to get robust or clustered SEs, users can either:

# 1) specify the arguments directly in the `tidy()` call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))

tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) or, feed tidy() a summary.fixest object that has already accepted
# these arguments

gravity_summ <- summary(gravity, cluster = c("Product", "Year"))

tidy(gravity_summ, conf.int = TRUE)

# approach (1) is preferred.
```

---

tidy.ftable *(Deprecated) Tidy ftable objects*

Description

This function is deprecated. Please use tibble::as_tibble() instead.

Usage

```
## S3 method for class 'ftable'
tidy(x, ...)
```

Arguments

x	An ftable object returned from stats::ftable().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

An ftable contains a "flat" contingency table. This melts it into a tibble::tibble with one column for each variable, then a Freq column.

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.numeric()

---

tidy.Gam *Tidy a(n) Gam object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'Gam'
tidy(x, ...)
```

Arguments

x	A Gam object returned from a call to gam::gam().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

Tidy gam objects created by calls to `mgcv::gam()` with `tidy.gam()`.

Value

A `tibble::tibble()` with columns:

`df`	Degrees of freedom used by this term in the model.
`meansq`	Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`sumsq`	Sum of squares explained by this term.
`term`	The name of the regression term.

See Also

`tidy()`, `gam::gam()`, `tidy.anova()`, `tidy.gam()`

Other gam tidiers: `glance.Gam()`

Examples

```
# load libraries for models and data
library(gam)

# fit model
g <- gam(mpg ~ s(hp, 4) + am + qsec, data = mtcars)

# summarize model fit with tidiers
tidy(g)
glance(g)
```

`tidy.gam` *Tidy a(n) gam object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'gam'
tidy(
  x,
  parametric = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  ...
)
```

Arguments

`x`	A `gam` object returned from a call to `mgcv::gam()`.
`parametric`	Logical indicating if parametric or smooth terms should be tidied. Defaults to `FALSE`, meaning that smooth terms are tidied by default.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

When `parametric = FALSE` return columns `edf` and `ref.df` rather than `estimate` and `std.error`.

Value

A `tibble::tibble()` with columns:

`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.
`edf`	The effective degrees of freedom. Only reported when 'parametric = FALSE'
`ref.df`	The reference degrees of freedom. Only reported when 'parametric = FALSE'

See Also

`tidy()`, `mgcv::gam()`

Other mgcv tidiers: `glance.gam()`

Examples

```
# load libraries for models and data
library(mgcv)

# fit model
g <- gam(mpg ~ s(hp) + am + qsec, data = mtcars)

# summarize model fit with tidiers
tidy(g)
tidy(g, parametric = TRUE)
glance(g)
augment(g)
```

---

tidy.garch *Tidy a(n) garch object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'garch'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A garch object returned by tseries::garch().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), tseries::garch()

Other garch tidiers: glance.garch()

Examples

```
# load libraries for models and data
library(tseries)
```

```
# load data
data(EuStockMarkets)

# fit model
dax <- diff(log(EuStockMarkets))[, "DAX"]
dax.garch <- garch(dax)
dax.garch

# summarize model fit with tidiers
tidy(dax.garch)
glance(dax.garch)
```

`tidy.geeglm` *Tidy a(n) geeglm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'geeglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `geeglm` object returned from a call to `geepack::geeglm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

If `conf.int = TRUE`, the confidence interval is computed with the an internal `confint.geeglm()` function.

If you have missing values in your model data, you may need to refit the model with `na.action = na.exclude` or deal with the missingness in the data beforehand.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `geepack::geeglm()`

Examples

```
# load modeling library
library(geepack)

# load data
data(state)


ds <- data.frame(state.region, state.x77)

# fit model
geefit <- geeglm(Income ~ Frost + Murder,
  id = state.region,
  data = ds,
  corstr = "exchangeable"
)

# summarize model fit with tidiers
tidy(geefit)
tidy(geefit, conf.int = TRUE)
```

`tidy.glht` *Tidy a(n) glht object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'glht'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `glht` object returned by `multcomp::glht()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `summary()` and `tidy.summary.glht()`.

Value

A `tibble::tibble()` with columns:

`contrast`	Levels being compared.
`estimate`	The estimated value of the regression term.
`null.value`	Value to which the estimate is compared.

See Also

`tidy()`, `multcomp::glht()`

Other multcomp tidiers: `tidy.cld()`, `tidy.confint.glht()`, `tidy.summary.glht()`

Examples

```
# load libraries for models and data
library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))
```

```
tidy(wht)

ggplot(wht, aes(lhs, estimate)) +
  geom_point()

CI <- confint(wht)

tidy(CI)

ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
  geom_pointrange()

tidy(summary(wht))
ggplot(mapping = aes(lhs, estimate)) +
  geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
  geom_point(aes(size = p), data = summary(wht)) +
  scale_size(trans = "reverse")

cld <- cld(wht)
tidy(cld)
```

`tidy.glm` *Tidy a(n) glm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'glm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `glm` object returned from `stats::glm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

See Also

`stats::glm()`

Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.summary.lm()`, `glance.svyglm()`, `tidy.lm()`, `tidy.lm.beta()`, `tidy.mlm()`, `tidy.summary.lm()`

`tidy.glmnet` *Tidy a(n) glmnet object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'glmnet'
tidy(x, return_zeros = FALSE, ...)
```

Arguments

`x` A `glmnet` object returned from `glmnet::glmnet()`.

`return_zeros` Logical indicating whether coefficients with value zero zero should be included in the results. Defaults to `FALSE`.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

Note that while this representation of GLMs is much easier to plot and combine than the default structure, it is also much more memory-intensive. Do not use for large, sparse matrices.

No `augment` method is yet provided even though the model produces predictions, because the input data is not tidy (it is a matrix that may be very wide) and therefore combining predictions with it is not logical. Furthermore, predictions make sense only with a specific choice of lambda.

Value

A `tibble::tibble()` with columns:

`dev.ratio`	Fraction of null deviance explained at each value of lambda.
`estimate`	The estimated value of the regression term.
`lambda`	Value of penalty parameter lambda.
`step`	Which step of lambda choices was used.
`term`	The name of the regression term.

See Also

`tidy()`, `glmnet::glmnet()`

Other glmnet tidiers: `glance.cv.glmnet()`, `glance.glmnet()`, `tidy.cv.glmnet()`

Examples

```
# load libraries for models and data
library(glmnet)

set.seed(2014)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)
fit1 <- glmnet(x, y)

# summarize model fit with tidiers + visualization
tidy(fit1)
glance(fit1)

library(dplyr)
library(ggplot2)

tidied <- tidy(fit1) |> filter(term != "(Intercept)")

ggplot(tidied, aes(step, estimate, group = term)) +
  geom_line()

ggplot(tidied, aes(lambda, estimate, group = term)) +
  geom_line() +
  scale_x_log10()
```

```
ggplot(tidied, aes(lambda, dev.ratio)) +
  geom_line()

# works for other types of regressions as well, such as logistic
g2 <- sample(1:2, 100, replace = TRUE)
fit2 <- glmnet(x, g2, family = "binomial")
tidy(fit2)
```

`tidy.glmRob` *Tidy a(n) glmRob object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'glmRob'
tidy(x, ...)
```

Arguments

`x`	A `glmRob` object returned from `robust::glmRob()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

See Also

`robust::glmRob()`

Other robust tidiers: `augment.lmRob()`, `glance.glmRob()`, `glance.lmRob()`, `tidy.lmRob()`

Examples

```
# load libraries for models and data
library(robust)

# fit model
gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")

# summarize model fit with tidiers
tidy(gm)
glance(gm)
```

`tidy.glmrob` *Tidy a(n) glmrob object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'glmrob'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `glmrob` object returned from `robustbase::glmrob()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`robustbase::glmrob()`

Other robustbase tidiers: `augment.glmrob()`, `augment.lmrob()`, `glance.lmrob()`, `tidy.lmrob()`

Examples

```
if (requireNamespace("robustbase", quietly = TRUE)) {
  # load libraries for models and data
  library(robustbase)

  data(coleman)
  set.seed(0)

  m <- lmrob(Y ~ ., data = coleman)
  tidy(m)
  augment(m)
  glance(m)

  data(carrots)

  Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
    family = binomial, data = carrots, method = "Mqle",
    control = glmrobMqle.control(tcc = 1.2)
  )

  tidy(Rfit)
  augment(Rfit)
}
```

`tidy.gmm` *Tidy a(n) gmm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'gmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `gmm` object returned from `gmm::gmm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.

statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), gmm::gmm()

Other gmm tidiers: glance.gmm()

Examples

```
# load libraries for models and data
library(gmm)

# examples come from the "gmm" package
# CAPM test with GMM
data(Finance)
r <- Finance[1:300, 1:10]
rm <- Finance[1:300, "rm"]
rf <- Finance[1:300, "rf"]

z <- as.matrix(r - rf)
t <- nrow(z)
zm <- rm - rf
h <- matrix(zm, t, 1)
res <- gmm(z ~ zm, x = h)

# tidy result
tidy(res)
tidy(res, conf.int = TRUE)
tidy(res, conf.int = TRUE, conf.level = .99)

# coefficient plot
library(ggplot2)
library(dplyr)

tidy(res, conf.int = TRUE) |>
  mutate(variable = reorder(term, estimate)) |>
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)

# from a function instead of a matrix
g <- function(theta, x) {
  e <- x[, 2:11] - theta[1] - (x[, 1] - theta[1]) %*% matrix(theta[2:11], 1, 10)
  gmat <- cbind(e, e * c(x[, 1]))
  return(gmat)
}
```

```
x <- as.matrix(cbind(rm, r))
res_black <- gmm(g, x = x, t0 = rep(0, 11))

tidy(res_black)
tidy(res_black, conf.int = TRUE)

# APT test with Fama-French factors and GMM

f1 <- zm
f2 <- Finance[1:300, "hml"] - rf
f3 <- Finance[1:300, "smb"] - rf
h <- cbind(f1, f2, f3)
res2 <- gmm(z ~ f1 + f2 + f3, x = h)

td2 <- tidy(res2, conf.int = TRUE)
td2

# coefficient plot
td2 |>
  mutate(variable = reorder(term, estimate)) |>
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)
```

`tidy.htest` *Tidy/glance a(n) htest object*

Description

For models that have only a single component, the `tidy()` and `glance()` methods are identical. Please see the documentation for both of those methods.

Usage

```
## S3 method for class 'htest'
tidy(x, ...)

## S3 method for class 'htest'
glance(x, ...)
```

Arguments

`x`	An `htest` objected, such as those created by `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`, etc.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`alternative`	Alternative hypothesis (character).
`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`estimate1`	Sometimes two estimates are computed, such as in a two-sample t-test.
`estimate2`	Sometimes two estimates are computed, such as in a two-sample t-test.
`method`	Method used.
`p.value`	The two-sided p-value associated with the observed statistic.
`parameter`	The parameter being modeled.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.

See Also

`tidy()`, `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`

Other htest tidiers: `augment.htest()`, `tidy.pairwise.htest()`, `tidy.power.htest()`

Examples

```
tt <- t.test(rnorm(10))

tidy(tt)

# the glance output will be the same for each of the below tests
glance(tt)

tt <- t.test(mpg ~ am, data = mtcars)

tidy(tt)

wt <- wilcox.test(mpg ~ am, data = mtcars, conf.int = TRUE, exact = FALSE)

tidy(wt)
```

```
ct <- cor.test(mtcars$wt, mtcars$mpg)

tidy(ct)

chit <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))

tidy(chit)
augment(chit)
```

`tidy.ivreg` *Tidy a(n) ivreg object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ivreg'
tidy(x, conf.int = FALSE, conf.level = 0.95, instruments = FALSE, ...)
```

Arguments

`x`	An `ivreg` object created by a call to `AER::ivreg()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`instruments`	Logical indicating whether to return coefficients from the second-stage or diagnostics tests for each endogenous regressor (F-statistics). Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

This tidier currently only supports `ivreg`-classed objects outputted by the `AER` package. The `ivreg` package also outputs objects of class `ivreg`, and will be supported in a later release.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`p.value.Sargan`	p-value for Sargan test of overidentifying restrictions.
`p.value.weakinst`	p-value for weak instruments test.
`p.value.Wu.Hausman`	p-value for Wu-Hausman weak instruments test for endogeneity.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`statistic.Sargan`	Statistic for Sargan test of overidentifying restrictions.
`statistic.weakinst`	Statistic for Wu-Hausman test.
`statistic.Wu.Hausman`	Statistic for Wu-Hausman weak instruments test for endogeneity.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `AER::ivreg()`

Other ivreg tidiers: `augment.ivreg()`, `glance.ivreg()`

Examples

```
# load libraries for models and data
library(AER)

# load data
data("CigarettesSW", package = "AER")

# fit model
ivr <- ivreg(
  log(packs) ~ income | population,
  data = CigarettesSW,
  subset = year == "1995"
```

```
)

# summarize model fit with tidiers
tidy(ivr)
tidy(ivr, conf.int = TRUE)
tidy(ivr, conf.int = TRUE, instruments = TRUE)

augment(ivr)
augment(ivr, data = CigarettesSW)
augment(ivr, newdata = CigarettesSW)

glance(ivr)
```

`tidy.kappa` *Tidy a(n) kappa object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'kappa'
tidy(x, ...)
```

Arguments

`x`	A kappa object returned from `psych::cohen.kappa()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

Note that confidence level (alpha) for the confidence interval cannot be set in `tidy`. Instead you must set the `alpha` argument to `psych::cohen.kappa()` when creating the kappa object.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
type	Either ‘weighted‘ or ‘unweighted‘.

See Also

tidy(), psych::cohen.kappa()

Examples

```
# load libraries for models and data
library(psych)

# generate example data
rater1 <- 1:9
rater2 <- c(1, 3, 1, 6, 1, 5, 5, 6, 7)

# fit model
ck <- cohen.kappa(cbind(rater1, rater2))

# summarize model fit with tidiers + visualization
tidy(ck)

# graph the confidence intervals
library(ggplot2)

ggplot(tidy(ck), aes(estimate, type)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
```

---

tidy.kde *Tidy a(n) kde object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'kde'
tidy(x, ...)
```

Arguments

x	A kde object returned from ks::kde().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

Returns a data frame in long format with four columns. Use tidyr::pivot_wider(..., names_from = variable, values_from = value) on the output to return to a wide format.

Value

A tibble::tibble() with columns:

estimate	The estimated value of the regression term.
obs	weighted observed number of events in each group.
value	The value/estimate of the component. Results from data reshaping.
variable	Variable under consideration.

See Also

tidy(), ks::kde()

Examples

```
# load libraries for models and data
library(ks)

# generate data
dat <- replicate(2, rnorm(100))
k <- kde(dat)

# summarize model fit with tidiers + visualization
td <- tidy(k)
td
```

```
library(ggplot2)
library(dplyr)
library(tidyr)

td |>
  pivot_wider(c(obs, estimate),
    names_from = variable,
    values_from = value
  ) |>
  ggplot(aes(x1, x2, fill = estimate)) +
  geom_tile() +
  theme_void()

# also works with 3 dimensions
dat3 <- replicate(3, rnorm(100))
k3 <- kde(dat3)

td3 <- tidy(k3)
td3
```

`tidy.Kendall` *Tidy a(n) Kendall object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'Kendall'
tidy(x, ...)
```

Arguments

`x`	A `Kendall` object returned from a call to `Kendall::Kendall()`, `Kendall::MannKendall()`, or `Kendall::SeasonalMannKendall()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

kendall_score	Kendall score.
p.value	The two-sided p-value associated with the observed statistic.
var_kendall_score	Variance of the kendall_score.
statistic	Kendall's tau statistic
denominator	The denominator, which is tau=kendall_score/denominator.

See Also

tidy(), Kendall::Kendall(), Kendall::MannKendall(), Kendall::SeasonalMannKendall()

Examples

```
# load libraries for models and data
library(Kendall)

A <- c(2.5, 2.5, 2.5, 2.5, 5, 6.5, 6.5, 10, 10, 10, 10, 10, 14, 14, 14, 16, 17)
B <- c(1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2)

# fit models and summarize results
f_res <- Kendall(A, B)
tidy(f_res)

s_res <- MannKendall(B)
tidy(s_res)

t_res <- SeasonalMannKendall(ts(A))
tidy(t_res)
```

---

tidy.kmeans *Tidy a(n) kmeans object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'kmeans'
tidy(x, col.names = colnames(x$centers), ...)
```

Arguments

x	A kmeans object created by stats::kmeans().
col.names	Dimension names. Defaults to the names of the variables in x. Set to NULL to get names x1, x2, ...
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

cluster	A factor describing the cluster from 1:k.
size	Number of points assigned to cluster.
withinss	The within-cluster sum of squares.

See Also

tidy(), stats::kmeans()

Other kmeans tidiers: augment.kmeans(), glance.kmeans()

Examples

```
library(cluster)
library(modeldata)
library(dplyr)

data(hpc_data)

x <- hpc_data[, 2:5]

fit <- pam(x, k = 4)

tidy(fit)
glance(fit)
augment(fit, x)
```

`tidy.lavaan` *Tidy a(n) lavaan object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lavaan'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `lavaan` object, such as those returned from `lavaan::cfa()`, and `lavaan::sem()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `lavaan::parameterEstimates()`. **Cautionary note**: Misspecified arguments may be silently ignored.

Value

A `tibble::tibble()` with one row for each estimated parameter and columns:

`term`	The result of paste(lhs, op, rhs)
`op`	The operator in the model syntax (e.g. ~~ for covariances, or ~ for regression parameters)
`group`	The group (if specified) in the lavaan model
`estimate`	The parameter estimate (may be standardized)
`std.error`	
`statistic`	The z value returned by `lavaan::parameterEstimates()`
`p.value`	
`conf.low`	
`conf.high`	
`std.lv`	Standardized estimates based on the variances of the (continuous) latent variables only
`std.all`	Standardized estimates based on both the variances of both (continuous) observed and latent variables.
`std.nox`	Standardized estimates based on both the variances of both (continuous) observed and latent variables, but not the variances of exogenous covariates.

See Also

tidy(), lavaan::cfa(), lavaan::sem(), lavaan::parameterEstimates()

Other lavaan tidiers: glance.lavaan()

Examples

```
# load libraries for models and data
library(lavaan)

cfa.fit <- cfa("F =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9",
  data = HolzingerSwineford1939, group = "school"
)

tidy(cfa.fit)
```

---

tidy.lm *Tidy a(n) lm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

x	An lm object created by stats::lm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

If the linear model is an `mlm` object (multiple linear model), there is an additional column `response`. See `tidy.mlm()`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `stats::summary.lm()`

Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.summary.lm()`, `glance.svyglm()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.mlm()`, `tidy.summary.lm()`

Examples

```
library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)

tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)
```

```
ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# aside: There are tidy() and glance() methods for lm.summary objects too.
# this can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars |>
  head(6) |>
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval

# simpler bivariate model since we're plotting in 2D
mod2 <- lm(mpg ~ wt, data = mtcars)

au <- augment(mod2, newdata = newdata, interval = "prediction")

ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata |>
  select(-mpg)

augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)

ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
```

```
  geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
b <- a + rnorm(length(a))
result <- lm(b ~ a)

tidy(result)
```

`tidy.lm.beta` *Tidy a(n) lm.beta object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lm.beta'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `lm.beta` object created by lm.beta::lm.beta.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

If the linear model is an `mlm` object (multiple linear model), there is an additional column `response`.

If you have missing values in your model data, you may need to refit the model with `na.action = na.exclude`.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.mlm(), tidy.summary.lm()

Examples

```
# load libraries for models and data
library(lm.beta)

# fit models
mod <- stats::lm(speed ~ ., data = cars)
std <- lm.beta(mod)

# summarize model fit with tidiers
tidy(std, conf.int = TRUE)

# generate data
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)

# fit models
mod2 <- lm(weight ~ group)
std2 <- lm.beta(mod2)

# summarize model fit with tidiers
tidy(std2, conf.int = TRUE)
```

`tidy.lmodel2` *Tidy a(n) lmodel2 object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lmodel2'
tidy(x, ...)
```

Arguments

`x`	A `lmodel2` object returned by `lmodel2::lmodel2()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

There are always only two terms in an `lmodel2`: `"Intercept"` and `"Slope"`. These are computed by four methods: OLS (ordinary least squares), MA (major axis), SMA (standard major axis), and RMA (ranged major axis).

The returned p-value is one-tailed and calculated via a permutation test. A permutational test is used because distributional assumptions may not be valid. More information can be found in `vignette("mod2user", package = "lmodel2")`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`term`	The name of the regression term.
`method`	Either OLS/MA/SMA/RMA

See Also

tidy(), lmodel2::lmodel2()

Other lmodel2 tidiers: glance.lmodel2()

Examples

```
# load libraries for models and data
library(lmodel2)

data(mod2ex2)
Ex2.res <- lmodel2(Prey ~ Predators, data = mod2ex2, "relative", "relative", 99)
Ex2.res

# summarize model fit with tidiers + visualization
tidy(Ex2.res)
glance(Ex2.res)

# this allows coefficient plots with ggplot2
library(ggplot2)

ggplot(tidy(Ex2.res), aes(estimate, term, color = method)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
```

---

tidy.lmRob *Tidy a(n) lmRob object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lmRob'
tidy(x, ...)
```

Arguments

x	A lmRob object returned from robust::lmRob().

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

See Also

`robust::lmRob()`

Other robust tidiers: `augment.lmRob()`, `glance.glmRob()`, `glance.lmRob()`, `tidy.glmRob()`

Examples

```
# load modeling library
library(robust)

# fit model
m <- lmRob(mpg ~ wt, data = mtcars)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

---

tidy.lmrob *Tidy a(n) lmrob object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lmrob'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A lmrob object returned from robustbase::lmrob().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

For tidiers for robust models from the **MASS** package see tidy.rlm().

See Also

robustbase::lmrob()

Other robustbase tidiers: augment.glmrob(), augment.lmrob(), glance.lmrob(), tidy.glmrob()

Examples

```
if (requireNamespace("robustbase", quietly = TRUE)) {
  # load libraries for models and data
  library(robustbase)

  data(coleman)
  set.seed(0)

  m <- lmrob(Y ~ ., data = coleman)
  tidy(m)
  augment(m)
  glance(m)

  data(carrots)

  Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
    family = binomial, data = carrots, method = "Mqle",
    control = glmrobMqle.control(tcc = 1.2)
  )

  tidy(Rfit)
  augment(Rfit)
```

```
}
```

`tidy.lsmobj` *Tidy a(n) lsmobj object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'lsmobj'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `lsmobj` object.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`. **Cautionary note**: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`contrast`	Levels being compared.
`df`	Degrees of freedom used by this term in the model.
`null.value`	Value to which the estimate is compared.

p.value	The two-sided p-value associated with the observed statistic.
std.error	The standard error of the regression term.
estimate	Expected marginal mean
statistic	T-ratio statistic

See Also

tidy(), emmeans::ref_grid(), emmeans::emmeans(), emmeans::contrast()

Other emmeans tidiers: tidy.emmGrid(), tidy.ref.grid(), tidy.summary_emm()

Examples

```
# load libraries for models and data
library(emmeans)

# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)
td <- tidy(oranges_rg1)
td

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)

ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)

by_price

tidy(by_price)
```

```
ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))
```

`tidy.manova` *Tidy a(n) manova object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'manova'
tidy(x, test = "Pillai", ...)
```

Arguments

`x`	A manova object return from `stats::manova()`.
`test`	One of "Pillai" (Pillai's trace), "Wilks" (Wilk's lambda), "Hotelling-Lawley" (Hotelling-Lawley trace) or "Roy" (Roy's greatest root) indicating which test statistic should be used. Defaults to "Pillai".
`...`	Arguments passed on to `stats::summary.manova` `object` An object of class `"manova"` or an aov object with multiple responses. `intercept` logical. If `TRUE`, the intercept term is included in the table. `tol` tolerance to be used in deciding if the residuals are rank-deficient: see `qr`.

Details

Depending on which test statistic is specified only one of `pillai`, `wilks`, `hl` or `roy` is included.

Value

A `tibble::tibble()` with columns:

`den.df`	Degrees of freedom of the denominator.
`num.df`	Degrees of freedom.
`p.value`	The two-sided p-value associated with the observed statistic.

statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
term	The name of the regression term.
pillai	Pillai's trace.
wilks	Wilk's lambda.
hl	Hotelling-Lawley trace.
roy	Roy's greatest root.

See Also

tidy(), stats::summary.manova()

Other anova tidiers: glance.anova(), glance.aov(), tidy.TukeyHSD(), tidy.anova(), tidy.aov(), tidy.aovlist()

Examples

```
npk2 <- within(npk, foo <- rnorm(24))
m <- manova(cbind(yield, foo) ~ block + N * P * K, npk2)
tidy(m)
```

---

tidy.map *Tidy a(n) map object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'map'
tidy(x, ...)
```

Arguments

x	A map object returned from maps::map().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

term	The name of the regression term.
long	Longitude.
lat	Latitude.

Remaining columns give information on geographic attributes and depend on the inputted map object. See ?maps::map for more information.

See Also

tidy(), maps::map()

Examples

```
# load libraries for models and data
library(maps)
library(ggplot2)

ca <- map("county", "ca", plot = FALSE, fill = TRUE)

tidy(ca)

qplot(long, lat, data = ca, geom = "polygon", group = group)

tx <- map("county", "texas", plot = FALSE, fill = TRUE)
tidy(tx)
qplot(long, lat,
  data = tx, geom = "polygon", group = group,
  colour = I("white")
)
```

---

tidy.margins *Tidy a(n) margins object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'margins'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A margins object returned from margins::margins().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

The margins package provides a way to obtain coefficient marginal effects for a variety of (non-linear) models, such as logit or models with multiway interaction terms. Note that the glance.margins() method requires rerunning the underlying model again, which can take some time. Similarly, an augment.margins() method is not currently supported, but users can simply run the underlying model to obtain the same information.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), margins::margins()

Examples

```
# load libraries for models and data
library(margins)

# example 1: logit model
mod_log <- glm(am ~ cyl + hp + wt, data = mtcars, family = binomial)

# get tidied "naive" model coefficients
tidy(mod_log)

# convert to marginal effects with margins()
marg_log <- margins(mod_log)

# get tidied marginal effects
tidy(marg_log)
tidy(marg_log, conf.int = TRUE)

# requires running the underlying model again. quick for this example
glance(marg_log)

# augmenting `margins` outputs isn't supported, but
# you can get the same info by running on the underlying model
augment(mod_log)

# example 2: threeway interaction terms
mod_ie <- lm(mpg ~ wt * cyl * disp, data = mtcars)

# get tidied "naive" model coefficients
tidy(mod_ie)

# convert to marginal effects with margins()
marg_ie0 <- margins(mod_ie)
# get tidied marginal effects
tidy(marg_ie0)
glance(marg_ie0)

# marginal effects evaluated at specific values of a variable (here: cyl)
marg_ie1 <- margins(mod_ie, at = list(cyl = c(4,6,8)))

# summarize model fit with tidiers
tidy(marg_ie1)

# marginal effects of one interaction variable (here: wt), modulated at
# specific values of the two other interaction variables (here: cyl and drat)
marg_ie2 <- margins(mod_ie,
                    variables = "wt",
                    at = list(cyl = c(4,6,8), drat = c(3, 3.5, 4)))

# summarize model fit with tidiers
tidy(marg_ie2)
```

`tidy.Mclust` *Tidy a(n) Mclust object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'Mclust'
tidy(x, ...)
```

Arguments

`x`	An `Mclust` object return from `mclust::Mclust()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`proportion`	The mixing proportion of each component
`size`	Number of points assigned to cluster.
`mean`	The mean for each component. In case of 2+ dimensional models, a column with the mean is added for each dimension. NA for noise component
`variance`	In case of one-dimensional and spherical models, the variance for each component, omitted otherwise. NA for noise component
`component`	Cluster id as a factor.

See Also

`tidy()`, `mclust::Mclust()`

Other mclust tidiers: `augment.Mclust()`

Examples

```
# load library for models and data
library(mclust)

# load data manipulation libraries
library(dplyr)
library(tibble)
library(purrr)
library(tidyr)

set.seed(27)

centers <- tibble(
  cluster = factor(1:3),
  # number points in each cluster
  num_points = c(100, 150, 50),
  # x1 coordinate of cluster center
  x1 = c(5, 0, -3),
  # x2 coordinate of cluster center
  x2 = c(-1, 1, -2)
)

points <- centers |>
  mutate(
    x1 = map2(num_points, x1, rnorm),
    x2 = map2(num_points, x2, rnorm)
  ) |>
  select(-num_points, -cluster) |>
  unnest(c(x1, x2))

# fit model
m <- Mclust(points)

# summarize model fit with tidiers
tidy(m)
augment(m, points)
glance(m)
```

---

tidy.mediate *Tidy a(n) mediate object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'mediate'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A mediate object produced by a call to mediation::mediate().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

The tibble has four rows. The first two indicate the mediated effect in the control and treatment groups, respectively. And the last two the direct effect in each group.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), mediation::mediate()

Examples

```
# load libraries for models and data
library(mediation)

data(jobs)

# fit models
b <- lm(job_seek ~ treat + econ_hard + sex + age, data = jobs)
c <- lm(depress2 ~ treat + job_seek + econ_hard + sex + age, data = jobs)
mod <- mediate(b, c, sims = 50, treat = "treat", mediator = "job_seek")

# summarize model fit with tidiers
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)
```

`tidy.mfx` *Tidy a(n) mfx object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

The particular functions below provide generic tidy methods for objects returned by the `mfx` package, preserving the calculated marginal effects instead of the naive model coefficients. The returned tidy tibble will also include an additional "atmean" column indicating how the marginal effects were originally calculated (see Details below).

Usage

```
## S3 method for class 'mfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'logitmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'negbinmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'poissonmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'probitmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A logitmfx, negbinmfx, poissonmfx, or probitmfx object. (Note that betamfx objects receive their own set of tidiers.)
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

The mfx package provides methods for calculating marginal effects for various generalized linear models (GLMs). Unlike standard linear models, estimated model coefficients in a GLM cannot be directly interpreted as marginal effects (i.e., the change in the response variable predicted after a one unit change in one of the regressors). This is because the estimated coefficients are multiplicative, dependent on both the link function that was used for the estimation and any other variables that were included in the model. When calculating marginal effects, users must typically choose whether they want to use i) the average observation in the data, or ii) the average of the sample marginal effects. See vignette("mfxarticle") from the mfx package for more details.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
atmean	TRUE if the marginal effects were originally calculated as the partial effects for the average observation. If FALSE, then these were instead calculated as average partial effects.

See Also

tidy(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.betamfx(), glance.mfx(), tidy.betamfx()

Examples

```
# load libraries for models and data
library(mfx)

# get the marginal effects from a logit regression
mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)

tidy(mod_logmfx, conf.int = TRUE)

# compare with the naive model coefficients of the same logit call
tidy(
  glm(am ~ cyl + hp + wt, family = binomial, data = mtcars),
  conf.int = TRUE
)

augment(mod_logmfx)
glance(mod_logmfx)

# another example, this time using probit regression
mod_probmfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)

tidy(mod_probmfx, conf.int = TRUE)
augment(mod_probmfx)
glance(mod_probmfx)
```

---

tidy.mjoint *Tidy a(n) mjoint object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'mjoint'
tidy(
  x,
```

```
  component = "survival",
  conf.int = FALSE,
  conf.level = 0.95,
  boot_se = NULL,
  ...
)
```

Arguments

x	An mjoint object returned from joineRML::mjoint().
component	Character specifying whether to tidy the survival or the longitudinal component of the model. Must be either "survival" or "longitudinal". Defaults to "survival".
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
boot_se	Optionally a bootSE object from joineRML::bootSE(). If specified, calculates confidence intervals via the bootstrap. Defaults to NULL, in which case standard errors are calculated from the empirical information matrix.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), joineRML::mjoint(), joineRML::bootSE()

Other mjoint tidiers: glance.mjoint()

Examples

```
# broom only skips running these examples because the example models take a
# while to generate—they should run just fine, though!
## Not run:


# load libraries for models and data
library(joineRML)

# fit a joint model with bivariate longitudinal outcomes
data(heart.valve)

hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
  heart.valve$num <= 50, ]

fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
  inits = list("gamma" = c(0.11, 1.51, 0.80)),
  timeVar = "time"
)

# extract the survival fixed effects
tidy(fit)

# extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# augment original data with fitted longitudinal values and residuals
```

```
hvd2 <- augment(fit)

# extract model statistics
glance(fit)

## End(Not run)
```

`tidy.mle2` *Tidy a(n) mle2 object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'mle2'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `mle2` object created by a call to `bbmle::mle2()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.

conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), bbmle::mle2(), tidy_optim()

Examples

```
# load libraries for models and data
library(bbmle)

# generate data
x <- 0:10
y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)
d <- data.frame(x, y)

# fit model
fit <- mle2(y ~ dpois(lambda = ymean),
  start = list(ymean = mean(y)), data = d
)

# summarize model fit with tidiers
tidy(fit)
```

---

tidy.mlm *Tidy a(n) mlm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'mlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	An mlm object created by stats::lm() with a matrix as the response.
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

In contrast to lm object (simple linear model), tidy output for mlm (multiple linear model) objects contain an additional column response.

If you have missing values in your model data, you may need to refit the model with na.action = na.exclude.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.summary.lm()

Examples

```
# fit model
mod <- lm(cbind(mpg, disp) ~ wt, mtcars)

# summarize model fit with tidiers
tidy(mod, conf.int = TRUE)
```

`tidy.mlogit` *Tidying methods for logit models*

Description

These methods tidy the coefficients of mnl and nl models generated by the functions of the `mlogit` package.

Usage

```
## S3 method for class 'mlogit'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	an object returned from `mlogit::mlogit()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.

p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), mlogit::mlogit()

Other mlogit tidiers: augment.mlogit(), glance.mlogit()

Examples

```
# load libraries for models and data
library(mlogit)

data("Fishing", package = "mlogit")
Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")

# fit model
m <- mlogit(mode ~ price + catch | income, data = Fish)

# summarize model fit with tidiers
tidy(m)
augment(m)
glance(m)
```

---

tidy.muhaz *Tidy a(n) muhaz object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'muhaz'
tidy(x, ...)
```

Arguments

x	A muhaz object returned by muhaz::muhaz().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

time	Point in time.
estimate	Estimated hazard rate.

See Also

tidy(), muhaz::muhaz()

Other muhaz tidiers: glance.muhaz()

Examples

```
# load libraries for models and data
library(muhaz)
library(survival)

# fit model
x <- muhaz(ovarian$futime, ovarian$fustat)

# summarize model fit with tidiers
tidy(x)
glance(x)
```

---

tidy.multinom *Tidying methods for multinomial logistic regression models*

Description

These methods tidy the coefficients of multinomial logistic regression models generated by multinom of the nnet package.

Usage

```
## S3 method for class 'multinom'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

x	A multinom object returned from nnet::multinom().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
y.value	The response level.

See Also

tidy(), nnet::multinom()

Other multinom tidiers: glance.multinom()

Examples

```
# load libraries for models and data
library(nnet)
library(MASS)

example(birthwt)

bwt.mu <- multinom(low ~ ., bwt)

tidy(bwt.mu)
glance(bwt.mu)

# or, for output from a multinomial logistic regression
fit.gear <- multinom(gear ~ mpg + factor(am), data = mtcars)
tidy(fit.gear)
glance(fit.gear)
```

`tidy.negbin` *Tidy a(n) negbin object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'negbin'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `glm.nb` object returned by `MASS::glm.nb()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	For `tidy()`, additional arguments passed to `summary()`. Otherwise ignored.

See Also

MASS::glm.nb()

Other glm.nb tidiers: glance.negbin()

Examples

```
# load libraries for models and data
library(MASS)

# fit model
r <- glm.nb(Days ~ Sex / (Age + Eth * Lrn), data = quine)

# summarize model fit with tidiers
tidy(r)
glance(r)
```

---

tidy.nlrq *Tidy a(n) nlrq object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'nlrq'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A nlrq object returned from quantreg::nlrq().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `quantreg::nlrq()`

Other quantreg tidiers: `augment.nlrq()`, `augment.rq()`, `augment.rqs()`, `glance.nlrq()`, `glance.rq()`, `tidy.rq()`, `tidy.rqs()`

Examples

```
# load modeling library
library(quantreg)

# build artificial data with multiplicative error
set.seed(1)
dat <- NULL
dat$x <- rep(1:25, 20)
dat$y <- SSlogis(dat$x, 10, 12, 2) * rnorm(500, 1, 0.1)

# fit the median using nlrq
mod <- nlrq(y ~ SSlogis(x, Asym, mid, scal),
  data = dat, tau = 0.5, trace = TRUE
)

# summarize model fit with tidiers
tidy(mod)
glance(mod)
augment(mod)
```

`tidy.nls` *Tidy a(n) nls object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'nls'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `nls` object returned from `stats::nls()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

tidy, stats::nls(), stats::summary.nls()

Other nls tidiers: augment.nls(), glance.nls()

Examples

```
# fit model
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

# summarize model fit with tidiers + visualization
tidy(n)
augment(n)
glance(n)

library(ggplot2)

ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1

augment(n, newdata = newdata)
```

---

tidy.numeric *Tidy atomic vectors*

Description

Vector tidiers are deprecated and will be removed from an upcoming release of broom.

Usage

```
## S3 method for class 'numeric'
tidy(x, ...)

## S3 method for class 'character'
tidy(x, ...)

## S3 method for class 'logical'
tidy(x, ...)
```

Arguments

x	An object of class "numeric", "integer", "character", or "logical". Most likely a named vector
...	Extra arguments (not used)

Details

Turn atomic vectors into data frames, where the names of the vector (if they exist) are a column and the values of the vector are a column.

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable()

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable()

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable()

Examples

```
## Not run:
x <- 1:5
names(x) <- letters[1:5]
tidy(x)

## End(Not run)
```

---

tidy.pairwise.htest *Tidy a(n) pairwise.htest object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'pairwise.htest'
tidy(x, ...)
```

Arguments

x	A pairwise.htest object such as those returned from stats::pairwise.t.test() or stats::pairwise.wilcox.test().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

Note that in one-sided tests, the alternative hypothesis of each test can be stated as "group1 is greater/less than group2".

Note also that the columns of group1 and group2 will always be a factor, even if the original input is (e.g.) numeric.

Value

A tibble::tibble() with columns:

group1	First group being compared.
group2	Second group being compared.
p.value	The two-sided p-value associated with the observed statistic.

See Also

stats::pairwise.t.test(), stats::pairwise.wilcox.test(), tidy()

Other htest tidiers: augment.htest(), tidy.htest(), tidy.power.htest()

Examples

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
ptt <- pairwise.t.test(Ozone, Month)
tidy(ptt)

library(modeldata)
data(hpc_data)
attach(hpc_data)
ptt2 <- pairwise.t.test(compounds, class)
tidy(ptt2)

tidy(pairwise.t.test(compounds, class, alternative = "greater"))
```

```
tidy(pairwise.t.test(compounds, class, alternative = "less"))

tidy(pairwise.wilcox.test(compounds, class))
```

`tidy.pam` *Tidy a(n) pam object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'pam'
tidy(x, col.names = paste0("x", 1:ncol(x$medoids)), ...)
```

Arguments

`x`	An `pam` object returned from `cluster::pam()`
`col.names`	Column names in the input data frame. Defaults to the names of the variables in x.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

For examples, see the pam vignette.

Value

A `tibble::tibble()` with columns:

`size`	Size of each cluster.
`max.diss`	Maximal dissimilarity between the observations in the cluster and that cluster's medoid.

avg.diss	Average dissimilarity between the observations in the cluster and that cluster's medoid.
diameter	Diameter of the cluster.
separation	Separation of the cluster.
avg.width	Average silhouette width of the cluster.
cluster	A factor describing the cluster from 1:k.

See Also

tidy(), cluster::pam()

Other pam tidiers: augment.pam(), glance.pam()

Examples

```
# load libraries for models and data
library(dplyr)
library(ggplot2)
library(cluster)
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)

# summarize model fit with tidiers + visualization
tidy(p)
glance(p)
augment(p, x)

augment(p, x) |>
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)
```

---

tidy.plm *Tidy a(n) plm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'plm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A plm objected returned by plm::plm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), plm::plm(), tidy.lm()

Other plm tidiers: augment.plm(), glance.plm()

Examples

```
# load libraries for models and data
library(plm)
```

```
# load data
data("Produc", package = "plm")

# fit model
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
  data = Produc, index = c("state", "year")
)

# summarize model fit with tidiers
summary(zz)

tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = 0.9)

augment(zz)
glance(zz)
```

`tidy.poLCA` *Tidy a(n) poLCA object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'poLCA'
tidy(x, ...)
```

Arguments

`x`	A `poLCA` object returned from `poLCA::poLCA()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

class	The class under consideration.
outcome	Outcome of manifest variable.
std.error	The standard error of the regression term.
variable	Manifest variable
estimate	Estimated class-conditional response probability

See Also

tidy(), poLCA::poLCA()

Other poLCA tidiers: augment.poLCA(), glance.poLCA()

Examples

```
# load libraries for models and data
library(poLCA)
library(dplyr)

# generate data
data(values)

f <- cbind(A, B, C, D) ~ 1

# fit model
M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

M1

# summarize model fit with tidiers + visualization
tidy(M1)
augment(M1)
glance(M1)

library(ggplot2)

ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
  geom_bar(stat = "identity", width = 1) +
  facet_wrap(~variable)

# three-class model with a single covariate.
data(election)

f2a <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ PARTY
```

```
nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)
td

ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
  geom_line() +
  facet_wrap(~variable, nrow = 2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)

au

count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)

au2

dim(au2)
```

`tidy.polr` *Tidy a(n) polr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'polr'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  p.values = FALSE,
  ...
)
```

Arguments

x	A polr object returned from MASS::polr().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
p.values	Logical. Should p-values be returned, based on chi-squared tests from MASS::dropterm(). Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

In broom 0.7.0 the coefficient_type column was renamed to coef.type, and the contents were changed as well. Now the contents are coefficient and scale, rather than coefficient and zeta.

Calculating p.values with the dropterm() function is the approach suggested by the MASS package author. This approach is computationally intensive so that p.values are only returned if requested explicitly. Additionally, it only works for models containing no variables with more than two categories. If this condition is not met, a message is shown and NA is returned instead of p-values.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy, MASS::polr()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clm(), glance.clmm(), glance.polr(), glance.svyolr(), tidy.clm(), tidy.clmm(), tidy.svyolr()

Examples

```
# load libraries for models and data
library(MASS)

# fit model
fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

# summarize model fit with tidiers
tidy(fit, exponentiate = TRUE, conf.int = TRUE)

glance(fit)
augment(fit, type.predict = "class")

fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)

tidy(fit, p.values = TRUE)
```

---

tidy.power.htest *Tidy a(n) power.htest object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'power.htest'
tidy(x, ...)
```

Arguments

x	A power.htest object such as those returned from stats::power.t.test().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`delta`	True difference in means.
`n`	Number of observations by component.
`power`	Power achieved for given value of n.
`sd`	Standard deviation.
`sig.level`	Significance level (Type I error probability).

See Also

`stats::power.t.test()`

Other htest tidiers: `augment.htest()`, `tidy.htest()`, `tidy.pairwise.htest()`

Examples

```
ptt <- power.t.test(n = 2:30, delta = 1)
tidy(ptt)

library(ggplot2)

ggplot(tidy(ptt), aes(n, power)) +
  geom_line()
```

`tidy.prcomp` *Tidy a(n) prcomp object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'prcomp'
tidy(x, matrix = "u", ...)
```

Arguments

x	A prcomp object returned by stats::prcomp().
matrix	Character specifying which component of the PCA should be tidied. • "u", "samples", "scores", or "x": returns information about the map from the original space into principle components space. • "v", "rotation", "loadings" or "variables": returns information about the map from principle components space back into the original space. • "d", "eigenvalues" or "pcs": returns information about the eigenvalues.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

See https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If matrix is "u", "samples", "scores", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:

row	ID of the original observation (i.e. rowname from original data).
PC	Integer indicating a principal component.
value	The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If matrix is "v", "rotation", "loadings" or "variables", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

row	The variable labels (colnames) of the data set on which PCA was performed.
PC	An integer vector indicating the principal component.
value	The value of the eigenvector (axis score) on the indicated principal component.

If matrix is "d", "eigenvalues" or "pcs", the columns are:

PC	An integer vector indicating the principal component.
std.dev	Standard deviation explained by this PC.

percent	Fraction of variation explained by this component (a numeric value between 0 and 1).
cumulative	Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).

See Also

stats::prcomp(), svd_tidiers

Other svd tidiers: augment.prcomp(), tidy_irlba(), tidy_svd()

Examples

```
pc <- prcomp(USArrests, scale = TRUE)

# information about rotation
tidy(pc)

# information about samples (states)
tidy(pc, "samples")

# information about PCs
tidy(pc, "pcs")

# state map
library(dplyr)
library(ggplot2)
library(maps)

pc |>
  tidy(matrix = "samples") |>
  mutate(region = tolower(row)) |>
  inner_join(map_data("state"), by = "region") |>
  ggplot(aes(long, lat, group = group, fill = value)) +
  geom_polygon() +
  facet_wrap(~PC) +
  theme_void() +
  ggtitle("Principal components of arrest data")

au <- augment(pc, data = USArrests)

au

ggplot(au, aes(.fittedPC1, .fittedPC2)) +
  geom_point() +
  geom_text(aes(label = .rownames), vjust = 1, hjust = 1)
```

`tidy.pyears` *Tidy a(n) pyears object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'pyears'
tidy(x, ...)
```

Arguments

`x`	A `pyears` object returned from `survival::pyears()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

`expected` is only present in the output when if a `ratetable` term is present.

If the `data.frame = TRUE` argument is supplied to `pyears`, this is simply the contents of `x$data`.

Value

A `tibble::tibble()` with columns:

`expected`	Expected number of events.
`pyears`	Person-years of exposure.
`n`	number of subjects contributing time
`event`	observed number of events

See Also

tidy(), survival::pyears()

Other pyears tidiers: glance.pyears()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# generate and format data
temp.yr <- tcut(mgus$dxyr, 55:92, labels = as.character(55:91))
temp.age <- tcut(mgus$age, 34:101, labels = as.character(34:100))
ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
pfit <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus,
  data.frame = TRUE
)

# summarize model fit with tidiers
tidy(pfit)
glance(pfit)

# if data.frame argument is not given, different information is present in
# output
pfit2 <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus)

tidy(pfit2)
glance(pfit2)
```

---

tidy.rcorr *Tidy a(n) rcorr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rcorr'
tidy(x, diagonal = FALSE, ...)
```

Arguments

x	An rcorr object returned from Hmisc::rcorr().
diagonal	Logical indicating whether or not to include diagonal elements of the correlation matrix, or the correlation of a column with itself. For the elements, estimate is always 1 and p.value is always NA. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

Suppose the original data has columns A and B. In the correlation matrix from rcorr there may be entries for both the cor(A, B) and cor(B, A). Only one of these pairs will ever be present in the tidy output.

Value

A tibble::tibble() with columns:

column1	Name or index of the first column being described.
column2	Name or index of the second column being described.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
n	Number of observations used to compute the correlation

See Also

tidy(), Hmisc::rcorr()

Examples

```
# load libraries for models and data
library(Hmisc)

mat <- replicate(52, rnorm(100))

# add some NAs
mat[sample(length(mat), 2000)] <- NA

# also, column names
colnames(mat) <- c(LETTERS, letters)
```

```
# fit model
rc <- rcorr(mat)

# summarize model fit with tidiers  + visualization
td <- tidy(rc)
td

library(ggplot2)
ggplot(td, aes(p.value)) +
  geom_histogram(binwidth = .1)

ggplot(td, aes(estimate, p.value)) +
  geom_point() +
  scale_y_log10()
```

`tidy.ref.grid` *Tidy a(n) ref.grid object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ref.grid'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `ref.grid` object created by `emmeans::ref_grid()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`. **Cautionary note**: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`df`	Degrees of freedom used by this term in the model.
`p.value`	The two-sided p-value associated with the observed statistic.
`std.error`	The standard error of the regression term.
`estimate`	Expected marginal mean
`statistic`	T-ratio statistic

See Also

`tidy()`, `emmeans::ref_grid()`, `emmeans::emmeans()`, `emmeans::contrast()`

Other emmeans tidiers: `tidy.emmGrid()`, `tidy.lsmobj()`, `tidy.summary_emm()`

Examples

```
# load libraries for models and data
library(emmeans)

# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)
td <- tidy(oranges_rg1)
td

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)
```

```
ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)

by_price

tidy(by_price)

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))
```

`tidy.regsubsets` *Tidy a(n) regsubsets object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'regsubsets'
tidy(x, ...)
```

Arguments

`x`	A `regsubsets` object created by `leaps::regsubsets()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`r.squared`	R squared statistic, or the percent of variation explained by the model.
`adj.r.squared`	Adjusted R squared statistic
`BIC`	Bayesian information criterion for the component.
`mallows_cp`	Mallow's Cp statistic.

See Also

`tidy()`, `leaps::regsubsets()`

Examples

```
# load libraries for models and data
library(leaps)

# fit model
all_fits <- regsubsets(hp ~ ., mtcars)

# summarize model fit with tidiers
tidy(all_fits)
```

`tidy.ridgelm` *Tidy a(n) ridgelm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ridgelm'
tidy(x, ...)
```

Arguments

x	A ridgelm object returned from MASS::lm.ridge().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

GCV	Generalized cross validation error estimate.
lambda	Value of penalty parameter lambda.
term	The name of the regression term.
estimate	estimate of scaled coefficient using this lambda
scale	Scaling factor of estimated coefficient

See Also

tidy(), MASS::lm.ridge()

Other ridgelm tidiers: glance.ridgelm()

Examples

```
# load libraries for models and data
library(MASS)

names(longley)[1] <- "y"

# fit model and summarizd results
fit1 <- lm.ridge(y ~ ., longley)
tidy(fit1)

fit2 <- lm.ridge(y ~ ., longley, lambda = seq(0.001, .05, .001))
td2 <- tidy(fit2)
g2 <- glance(fit2)

# coefficient plot
library(ggplot2)
ggplot(td2, aes(lambda, estimate, color = term)) +
  geom_line()
```

```
# GCV plot
ggplot(td2, aes(lambda, GCV)) +
  geom_line()

# add line for the GCV minimizing estimate
ggplot(td2, aes(lambda, GCV)) +
  geom_line() +
  geom_vline(xintercept = g2$lambdaGCV, col = "red", lty = 2)
```

`tidy.rlm` *Tidy a(n) rlm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `rlm` object returned by `MASS::rlm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

See Also

`MASS::rlm()`

Other rlm tidiers: `augment.rlm()`, `glance.rlm()`

`tidy.rma` *Tidy a(n) rma object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rma'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  include_studies = FALSE,
  measure = "GEN",
  ...
)
```

Arguments

`x`	An `rma` object such as those created by `metafor::rma()`, `metafor::rma.uni()`, `metafor::rma.glmm()`, `metafor::rma.mh()`, `metafor::rma.mv()`, or `metafor::rma.peto()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`include_studies`	Logical. Should individual studies be included in the output? Defaults to `FALSE`.
`measure`	Measure type. See `metafor::escalc()`
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.

- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the individual study
`type`	The estimate type (summary vs individual study)

Examples

```
# load libraries for models and data
library(metafor)

df <-
  escalc(
    measure = "RR",
    ai = tpos,
    bi = tneg,
    ci = cpos,
    di = cneg,
    data = dat.bcg
  )

meta_analysis <- rma(yi, vi, data = df, method = "EB")

tidy(meta_analysis)
```

`tidy.roc` *Tidy a(n) roc object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'roc'
tidy(x, ...)
```

Arguments

x	An roc object returned from a call to AUC::roc().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

cutoff	The cutoff used for classification. Observations with predicted probabilities above this value were assigned class 1, and observations with predicted probabilities below this value were assigned class 0.
fpr	False positive rate.
tpr	The true positive rate at the given cutoff.

See Also

tidy(), AUC::roc()

Examples

```
# load libraries for models and data
library(AUC)

# load data
data(churn)

# fit model
r <- roc(churn$predictions, churn$labels)

# summarize with tidiers + visualization
td <- tidy(r)
td

library(ggplot2)
```

```
ggplot(td, aes(fpr, tpr)) +
  geom_line()

# compare the ROC curves for two prediction algorithms
library(dplyr)
library(tidyr)

rocs <- churn |>
  pivot_longer(contains("predictions"),
    names_to = "algorithm",
    values_to = "value"
  ) |>
  nest(data = -algorithm) |>
  mutate(tidy_roc = purrr::map(data, \(x) tidy(roc(x$value, x$labels)))) |>
  unnest(tidy_roc)

ggplot(rocs, aes(fpr, tpr, color = algorithm)) +
  geom_line()
```

`tidy.rq` *Tidy a(n) rq object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rq'
tidy(x, se.type = NULL, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `rq` object returned from `quantreg::rq()`.
`se.type`	Character specifying the method to use to calculate standard errors. Passed to `quantreg::summary.rq()` `se` argument. Defaults to `"rank"` if the sample size is less than 1000, otherwise defaults to `"nid"`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `quantreg::summary.rq()`.

Details

If `se.type = "rank"` confidence intervals are calculated by `summary.rq` and `statistic` and `p.value` values are not returned. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

`tidy()`, `quantreg::rq()`

Other quantreg tidiers: `augment.nlrq()`, `augment.rq()`, `augment.rqs()`, `glance.nlrq()`, `glance.rq()`, `tidy.nlrq()`, `tidy.rqs()`

Examples

```
# load modeling library and data
library(quantreg)

data(stackloss)

# median (l1) regression fit for the stackloss data.
mod1 <- rq(stack.loss ~ stack.x, .5)

# weighted sample median
mod2 <- rq(rnorm(50) ~ 1, weights = runif(50))

# summarize model fit with tidiers
tidy(mod1)
glance(mod1)
augment(mod1)

tidy(mod2)
glance(mod2)
augment(mod2)

# varying tau to generate an rqs object
mod3 <- rq(stack.loss ~ stack.x, tau = c(.25, .5))
```

```
tidy(mod3)
augment(mod3)

# glance cannot handle rqs objects like `mod3`--use a purrr
# `map`-based workflow instead
```

`tidy.rqs` *Tidy a(n) rqs object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rqs'
tidy(x, se.type = "rank", conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `rqs` object returned from `quantreg::rq()`.
`se.type`	Character specifying the method to use to calculate standard errors. Passed to `quantreg::summary.rq()` `se` argument. Defaults to `"rank"`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `quantreg::summary.rqs()`

Details

If `se.type = "rank"` confidence intervals are calculated by `summary.rq`. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.

estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
quantile	Linear conditional quantile.

See Also

tidy(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rq(), augment.rqs(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rq()

Examples

```
# load modeling library and data
library(quantreg)

data(stackloss)

# median (l1) regression fit for the stackloss data.
mod1 <- rq(stack.loss ~ stack.x, .5)

# weighted sample median
mod2 <- rq(rnorm(50) ~ 1, weights = runif(50))

# summarize model fit with tidiers
tidy(mod1)
glance(mod1)
augment(mod1)

tidy(mod2)
glance(mod2)
augment(mod2)

# varying tau to generate an rqs object
mod3 <- rq(stack.loss ~ stack.x, tau = c(.25, .5))

tidy(mod3)
augment(mod3)

# glance cannot handle rqs objects like `mod3`--use a purrr
# `map`-based workflow instead
```

`tidy.sarlm` *Tidying methods for spatially autoregressive models*

Description

These methods tidy the coefficients of spatial autoregression models generated by functions in the `spatialreg` package.

Usage

```
## S3 method for class 'sarlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An object returned from `spatialreg::lagsarlm()` or `spatialreg::errorsarlm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

tidy(), spatialreg::lagsarlm(), spatialreg::errorsarlm(), spatialreg::sacsarlm()

Other spatialreg tidiers: augment.sarlm(), glance.sarlm()

Examples

```
# load libraries for models and data
library(spatialreg)
library(spdep)

# load data
data(oldcol, package = "spdep")

listw <- nb2listw(COL.nb, style = "W")

# fit model
crime_sar <-
  lagsarlm(CRIME ~ INC + HOVAL,
    data = COL.OLD,
    listw = listw,
    method = "eigen"
  )

# summarize model fit with tidiers
tidy(crime_sar)
tidy(crime_sar, conf.int = TRUE)
glance(crime_sar)
augment(crime_sar)

# fit another model
crime_sem <- errorsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sem)
tidy(crime_sem, conf.int = TRUE)
glance(crime_sem)
augment(crime_sem)

# fit another model
crime_sac <- sacsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)

# summarize model fit with tidiers
tidy(crime_sac)
tidy(crime_sac, conf.int = TRUE)
glance(crime_sac)
augment(crime_sac)
```

`tidy.spec` *Tidy a(n) spec object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'spec'
tidy(x, ...)
```

Arguments

`x`	A spec object created by `stats::spectrum()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`freq`	Vector of frequencies at which the spectral density is estimated.
`spec`	Vector (for univariate series) or matrix (for multivariate series) of estimates of the spectral density at frequencies corresponding to freq.

See Also

`tidy()`, `stats::spectrum()`

Other time series tidiers: `tidy.acf()`, `tidy.ts()`, `tidy.zoo()`

Examples

```
spc <- spectrum(lh)
tidy(spc)

library(ggplot2)
ggplot(tidy(spc), aes(freq, spec)) +
  geom_line()
```

`tidy.speedglm` *Tidy a(n) speedglm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'speedglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `speedglm` object returned from `speedglm::speedglm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

speedglm::speedglm()

Other speedlm tidiers: augment.speedlm(), glance.speedglm(), glance.speedlm(), tidy.speedlm()

Examples

```
# load libraries for models and data
library(speedglm)

# generate data
clotting <- data.frame(
  u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
  lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18)
)

# fit model
fit <- speedglm(lot1 ~ log(u), data = clotting, family = Gamma(log))

# summarize model fit with tidiers
tidy(fit)
glance(fit)
```

---

tidy.speedlm *Tidy a(n) speedlm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'speedlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A speedlm object returned from speedglm::speedlm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

speedglm::speedlm(), tidy.lm()

Other speedlm tidiers: augment.speedlm(), glance.speedglm(), glance.speedlm(), tidy.speedglm()

Examples

```
# load modeling library
library(speedglm)
```

```
# fit model
mod <- speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)

# summarize model fit with tidiers
tidy(mod)
glance(mod)
augment(mod)
```

`tidy.summary.glht` *Tidy a(n) summary.glht object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'summary.glht'
tidy(x, ...)
```

Arguments

`x`	A `summary.glht` object created by calling `multcomp::summary.glht()` on a `glht` object created with `multcomp::glht()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`contrast`	Levels being compared.
`estimate`	The estimated value of the regression term.
`null.value`	Value to which the estimate is compared.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.

See Also

tidy(), multcomp::summary.glht(), multcomp::glht()

Other multcomp tidiers: tidy.cld(), tidy.confint.glht(), tidy.glht()

Examples

```
# load libraries for models and data
library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

tidy(wht)

ggplot(wht, aes(lhs, estimate)) +
  geom_point()

CI <- confint(wht)

tidy(CI)

ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
  geom_pointrange()

tidy(summary(wht))
ggplot(mapping = aes(lhs, estimate)) +
  geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
  geom_point(aes(size = p), data = summary(wht)) +
  scale_size(trans = "reverse")

cld <- cld(wht)
tidy(cld)
```

---

tidy.summary.lm *Tidy a(n) summary.lm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'summary.lm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A summary.lm object created by stats::summary.lm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

The tidy.summary.lm() method is a potentially useful alternative to tidy.lm(). For instance, if users have already converted large lm objects into their leaner summary.lm equivalents to conserve memory.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), stats::summary.lm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.lm.beta(), tidy.mlm()

Examples

```
# fit model
mod <- lm(mpg ~ wt + qsec, data = mtcars)
modsumm <- summary(mod)

# summarize model fit with tidiers
tidy(mod, conf.int = TRUE)

# equivalent to the above
tidy(modsumm, conf.int = TRUE)

glance(mod)

# mostly the same, except for a few missing columns
glance(modsumm)
```

`tidy.summary_emm` *Tidy a(n) summary_emm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'summary_emm'
tidy(x, null.value = NULL, ...)
```

Arguments

`x`	A `summary_emm` object.
`null.value`	Value to which estimate is compared.
`...`	Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`. **Cautionary note**: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
contrast	Levels being compared.
den.df	Degrees of freedom of the denominator.
df	Degrees of freedom used by this term in the model.
null.value	Value to which the estimate is compared.
num.df	Degrees of freedom.
p.value	The two-sided p-value associated with the observed statistic.
std.error	The standard error of the regression term.
level1	One level of the factor being contrasted
level2	The other level of the factor being contrasted
term	Model term in joint tests
estimate	Expected marginal mean
statistic	T-ratio statistic or F-ratio statistic

See Also

tidy(), emmeans::ref_grid(), emmeans::emmeans(), emmeans::contrast()

Other emmeans tidiers: tidy.emmGrid(), tidy.lsmobj(), tidy.ref.grid()

Examples

```
# load libraries for models and data
library(emmeans)

# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)
td <- tidy(oranges_rg1)
td

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
```

```
library(ggplot2)

ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)

by_price

tidy(by_price)

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))
```

`tidy.survdiff` *Tidy a(n) survdiff object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survdiff'
tidy(x, ...)
```

Arguments

`x`	An `survdiff` object returned from `survival::survdiff()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`exp`	Weighted expected number of events in each group.
`N`	Number of subjects in each group.
`obs`	weighted observed number of events in each group.

See Also

`tidy()`, `survival::survdiff()`

Other survdiff tidiers: `glance.survdiff()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
# load libraries for models and data
library(survival)

# fit model
s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

# summarize model fit with tidiers
tidy(s)
glance(s)
```

---

tidy.survexp *Tidy a(n) survexp object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survexp'
tidy(x, ...)
```

Arguments

x	An survexp object returned from survival::survexp().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

n.risk	Number of individuals at risk at time zero.
time	Point in time.
estimate	Estimate survival

See Also

tidy(), survival::survexp()

Other survexp tidiers: glance.survexp()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survfit(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
sexpfit <- survexp(
  futime ~ 1,
  rmap = list(
    sex = "male",
    year = accept.dt,
    age = (accept.dt - birth.dt)
  ),
```

```
  method = "conditional",
  data = jasa
)

# summarize model fit with tidiers
tidy(sexpfit)
glance(sexpfit)
```

`tidy.survfit` *Tidy a(n) survfit object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survfit'
tidy(x, ...)
```

Arguments

`x`	An `survfit` object returned from `survival::survfit()`.
`...`	For `glance.survfit()`, additional arguments passed to `summary()`. Otherwise ignored.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`n.censor`	Number of censored events.
`n.event`	Number of events at time t.
`n.risk`	Number of individuals at risk at time zero.
`std.error`	The standard error of the regression term.
`time`	Point in time.
`estimate`	estimate of survival or cumulative incidence rate when multistate
`state`	state if multistate survfit object input
`strata`	strata if stratified survfit object input

See Also

tidy(), survival::survfit()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survreg()

Examples

```
# load libraries for models and data
library(survival)

# fit model
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)

# summarize model fit with tidiers + visualization
tidy(sfit)
glance(sfit)

library(ggplot2)

ggplot(tidy(sfit), aes(time, estimate)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)

# multi-state
fitCI <- survfit(Surv(stop, status * as.numeric(event), type = "mstate") ~ 1,
  data = mgus1, subset = (start == 0)
)

td_multi <- tidy(fitCI)

td_multi

ggplot(td_multi, aes(time, estimate, group = state)) +
  geom_line(aes(color = state)) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
```

---

tidy.survreg *Tidy a(n) survreg object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survreg'
tidy(x, conf.level = 0.95, conf.int = FALSE, ...)
```

Arguments

x	An survreg object returned from survival::survreg().
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), survival::survreg()

Other survreg tidiers: augment.survreg(), glance.survreg()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit()

Examples

```
# load libraries for models and data
library(survival)

# fit model
sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx,
  ovarian,
  dist = "exponential"
)

# summarize model fit with tidiers + visualization
tidy(sr)
augment(sr, ovarian)
glance(sr)

# coefficient plot
td <- tidy(sr, conf.int = TRUE)

library(ggplot2)

ggplot(td, aes(estimate, term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
```

`tidy.svyglm` *Tidy a(n) svyglm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'svyglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A `svyglm` object returned from `survey::svyglm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.

conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

See Also

survey::svyglm(), stats::glm()

---

tidy.svyolr *Tidy a(n) svyolr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'svyolr'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

x	A svyolr object returned from survey::svyolr().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are:

- `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored.
- `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

The `tidy.svyolr()` tidier is a light wrapper around `tidy.polr()`. However, the implementation for p-value calculation in `tidy.polr()` is both computationally intensive and specific to that model, so the `p.values` argument to `tidy.svyolr()` is currently ignored, and will raise a warning when passed.

Value

A `tibble::tibble()` with columns:

`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`estimate`	The estimated value of the regression term.
`p.value`	The two-sided p-value associated with the observed statistic.
`statistic`	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
`std.error`	The standard error of the regression term.
`term`	The name of the regression term.

See Also

tidy, `survey::svyolr()`

Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clm()`, `glance.clmm()`, `glance.polr()`, `glance.svyolr()`, `tidy.clm()`, `tidy.clmm()`, `tidy.polr()`

Examples

```
library(broom)
library(survey)

data(api)
dclus1 <- svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
dclus1 <- update(dclus1, mealcat = cut(meals, c(0, 25, 50, 75, 100)))

m <- svyolr(mealcat ~ avg.ed + mobility + stype, design = dclus1)

m
```

```
tidy(m, conf.int = TRUE)
```

`tidy.systemfit` *Tidy a(n) systemfit object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'systemfit'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

`x`	A `systemfit` object produced by a call to `systemfit::systemfit()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

This tidy method works with any model objects of class `systemfit`. Default returns a tibble of six columns.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
std.error	The standard error of the regression term.
term	The name of the regression term.

See Also

tidy(), systemfit::systemfit()

Examples

```
set.seed(27)

# load libraries for models and data
library(systemfit)

# generate data
df <- data.frame(
  X = rnorm(100),
  Y = rnorm(100),
  Z = rnorm(100),
  W = rnorm(100)
)

# fit model
fit <- systemfit(formula = list(Y ~ Z, W ~ X), data = df, method = "SUR")

# summarize model fit with tidiers
tidy(fit)
tidy(fit, conf.int = TRUE)
```

---

tidy.table *Tidy a(n) table object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Deprecated. Please use tibble::as_tibble() instead.

Usage

```
## S3 method for class 'table'
tidy(x, ...)
```

Arguments

x	A base::table object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

Directly calls `tibble::as_tibble()` on a base::table object.

Value

A tibble::tibble in long-form containing frequency information for the table in a `Freq` column. The result is much like what you get from `tidyr::pivot_longer()`.

See Also

`tibble::as_tibble.table()`

---

tidy.ts *Tidy a(n) ts object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ts'
tidy(x, ...)
```

Arguments

x	A univariate or multivariate ts times series object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

series column is only present for multivariate ts objects.

Value

A tibble::tibble() with columns:

index	Index (i.e. date or time) for a 'ts' or 'zoo' object.
series	Name of the series (present only for multivariate time series).
value	The value/estimate of the component. Results from data reshaping.

See Also

tidy(), stats::ts()

Other time series tidiers: tidy.acf(), tidy.spec(), tidy.zoo()

Examples

```
set.seed(678)

tidy(ts(1:10, frequency = 4, start = c(1959, 2)))

z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
colnames(z) <- c("Aa", "Bb", "Cc")

tidy(z)
```

`tidy.TukeyHSD` *Tidy a(n) TukeyHSD object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'TukeyHSD'
tidy(x, ...)
```

Arguments

`x`	A `TukeyHSD` object return from `stats::TukeyHSD()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A `tibble::tibble()` with columns:

`adj.p.value`	P-value adjusted for multiple comparisons.
`conf.high`	Upper bound on the confidence interval for the estimate.
`conf.low`	Lower bound on the confidence interval for the estimate.
`contrast`	Levels being compared.
`estimate`	The estimated value of the regression term.
`null.value`	Value to which the estimate is compared.
`term`	The name of the regression term.

See Also

`tidy()`, `stats::TukeyHSD()`

Other anova tidiers: `glance.anova()`, `glance.aov()`, `tidy.anova()`, `tidy.aov()`, `tidy.aovlist()`, `tidy.manova()`

Examples

```
fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)
thsd <- TukeyHSD(fm1, "tension", ordered = TRUE)
tidy(thsd)

# may include comparisons on multiple terms
fm2 <- aov(mpg ~ as.factor(gear) * as.factor(cyl), data = mtcars)
tidy(TukeyHSD(fm2))
```

---

tidy.varest *Tidy a(n) varest object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'varest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	A varest object produced by a call to vars::VAR().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	For glance(), additional arguments passed to summary(). Otherwise ignored.

Details

The tibble has one row for each term in the regression. The component column indicates whether a particular term was used to model either the "mean" or "precision". Here the precision is the inverse of the variance, often referred to as phi. At least one term will have been used to model the precision phi.

The vars package does not include a confint method and does not report confidence intervals for varest objects. Setting the tidy argument conf.int = TRUE will return a warning.

Value

A tibble::tibble() with columns:

conf.high	Upper bound on the confidence interval for the estimate.
conf.low	Lower bound on the confidence interval for the estimate.
estimate	The estimated value of the regression term.
p.value	The two-sided p-value associated with the observed statistic.
statistic	The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error	The standard error of the regression term.
term	The name of the regression term.
component	Whether a particular term was used to model the mean or the precision in the regression. See details.

See Also

tidy(), vars::VAR()

Examples

```
# load libraries for models and data
library(vars)

# load data
data("Canada", package = "vars")

# fit models
mod <- VAR(Canada, p = 1, type = "both")

# summarize model fit with tidiers
tidy(mod)
glance(mod)
```

---

tidy.zoo *Tidy a(n) zoo object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'zoo'
tidy(x, ...)
```

Arguments

x	A zoo object such as those created by zoo::zoo().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble() with columns:

index	Index (i.e. date or time) for a ‘ts‘ or ‘zoo‘ object.
series	Name of the series (present only for multivariate time series).
value	The value/estimate of the component. Results from data reshaping.

See Also

tidy(), zoo::zoo()

Other time series tidiers: tidy.acf(), tidy.spec(), tidy.ts()

Examples

```
# load libraries for models and data
library(zoo)
library(ggplot2)

set.seed(1071)

# generate data
Z.index <- as.Date(sample(12450:12500, 10))
Z.data <- matrix(rnorm(30), ncol = 3)
colnames(Z.data) <- c("Aa", "Bb", "Cc")
Z <- zoo(Z.data, Z.index)

# summarize model fit with tidiers + visualization
tidy(Z)

ggplot(tidy(Z), aes(index, value, color = series)) +
```

```
  geom_line()

ggplot(tidy(Z), aes(index, value)) +
  geom_line() +
  facet_wrap(~series, ncol = 1)

Zrolled <- rollmean(Z, 5)
ggplot(tidy(Zrolled), aes(index, value, color = series)) +
  geom_line()
```

`tidy_irlba` *Tidy a(n) irlba object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `interp::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, they throw an error.

Usage

```
tidy_irlba(x, ...)
```

Arguments

`x`	A list returned from `irlba::irlba()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Details

A very thin wrapper around `tidy_svd()`.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If `matrix` is `"u"`, `"samples"`, `"scores"`, or `"x"` each row in the tidied output corresponds to the original data in PCA space. The columns are:

`row`	ID of the original observation (i.e. rowname from original data).
`PC`	Integer indicating a principal component.
`value`	The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If `matrix` is `"v"`, `"rotation"`, `"loadings"` or `"variables"`, each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

`row`	The variable labels (colnames) of the data set on which PCA was performed.
`PC`	An integer vector indicating the principal component.
`value`	The value of the eigenvector (axis score) on the indicated principal component.

If `matrix` is `"d"`, `"eigenvalues"` or `"pcs"`, the columns are:

`PC`	An integer vector indicating the principal component.
`std.dev`	Standard deviation explained by this PC.
`percent`	Fraction of variation explained by this component (a numeric value between 0 and 1).
`cumulative`	Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).

See Also

`tidy()`, `irlba::irlba()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_optim()`, `tidy_svd()`, `tidy_xyz()`

Other svd tidiers: `augment.prcomp()`, `tidy.prcomp()`, `tidy_svd()`

Examples

```
library(modeldata)
data(hpc_data)

mat <- scale(as.matrix(hpc_data[, 2:5]))
s <- svd(mat)

tidy_u <- tidy(s, matrix = "u")
tidy_u

tidy_d <- tidy(s, matrix = "d")
tidy_d
```

```
tidy_v <- tidy(s, matrix = "v")
tidy_v

library(ggplot2)
library(dplyr)

ggplot(tidy_d, aes(PC, percent)) +
  geom_point() +
  ylab("% of variance explained")

tidy_u |>
  mutate(class = hpc_data$class[row]) |>
  ggplot(aes(class, value)) +
  geom_boxplot() +
  facet_wrap(~PC, scale = "free_y")
```

`tidy_optim` *Tidy a(n) optim object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `interp::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, they throw an error.

Usage

```
tidy_optim(x, ...)
```

Arguments

`x`	A list returned from `stats::optim()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Two exceptions here are: • `tidy()` methods will warn when supplied an `exponentiate` argument if it will be ignored. • `augment()` methods will warn when supplied a `newdata` argument if it will be ignored.

Value

A tibble::tibble() with columns:

parameter	The parameter being modeled.
std.error	The standard error of the regression term.
value	The value/estimate of the component. Results from data reshaping.

std.error is only provided as a column if the Hessian is calculated.

Note

This function assumes that the provided objective function is a negative log-likelihood function. Results will be invalid if an incorrect function is supplied.

tidy(o) glance(o)

See Also

tidy(), stats::optim()

Other list tidiers: glance_optim(), list_tidiers, tidy_irlba(), tidy_svd(), tidy_xyz()

Examples

```
f <- function(x) (x[1] - 2)^2 + (x[2] - 3)^2 + (x[3] - 8)^2
o <- optim(c(1, 1, 1), f)
```

---

tidy_svd *Tidy a(n) svd object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), svd() and interp::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, they throw an error.

Usage

```
tidy_svd(x, matrix = "u", ...)
```

Arguments

x	A list with components u, d, v returned by base::svd().
matrix	Character specifying which component of the PCA should be tidied. • "u", "samples", "scores", or "x": returns information about the map from the original space into principle components space. • "v", "rotation", "loadings" or "variables": returns information about the map from principle components space back into the original space. • "d", "eigenvalues" or "pcs": returns information about the eigenvalues.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Details

See https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If matrix is "u", "samples", "scores", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:

row	ID of the original observation (i.e. rowname from original data).
PC	Integer indicating a principal component.
value	The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If matrix is "v", "rotation", "loadings" or "variables", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

row	The variable labels (colnames) of the data set on which PCA was performed.
PC	An integer vector indicating the principal component.
value	The value of the eigenvector (axis score) on the indicated principal component.

If matrix is "d", "eigenvalues" or "pcs", the columns are:

PC	An integer vector indicating the principal component.
std.dev	Standard deviation explained by this PC.

percent	Fraction of variation explained by this component (a numeric value between 0 and 1).
cumulative	Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).

See Also

base::svd()

Other svd tidiers: augment.prcomp(), tidy.prcomp(), tidy_irlba()

Other list tidiers: glance_optim(), list_tidiers, tidy_irlba(), tidy_optim(), tidy_xyz()

Examples

```
library(modeldata)
data(hpc_data)

mat <- scale(as.matrix(hpc_data[, 2:5]))
s <- svd(mat)

tidy_u <- tidy(s, matrix = "u")
tidy_u

tidy_d <- tidy(s, matrix = "d")
tidy_d

tidy_v <- tidy(s, matrix = "v")
tidy_v

library(ggplot2)
library(dplyr)

ggplot(tidy_d, aes(PC, percent)) +
  geom_point() +
  ylab("% of variance explained")

tidy_u |>
  mutate(class = hpc_data$class[row]) |>
  ggplot(aes(class, value)) +
  geom_boxplot() +
  facet_wrap(~PC, scale = "free_y")
```

---

tidy_xyz *Tidy a(n) xyz object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), svd() and interp::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, they throw an error.

xyz lists (lists where x and y are vectors of coordinates and z is a matrix of values) are typically used by functions such as graphics::persp() or graphics::image() and returned by interpolation functions such as interp::interp().

Usage

```
tidy_xyz(x, ...)
```

Arguments

x	A list with component x, y and z, where x and y are vectors and z is a matrix. The length of x must equal the number of rows in z and the length of y must equal the number of columns in z.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Two exceptions here are: • tidy() methods will warn when supplied an exponentiate argument if it will be ignored. • augment() methods will warn when supplied a newdata argument if it will be ignored.

Value

A tibble::tibble with vector columns x, y and z.

See Also

tidy(), graphics::persp(), graphics::image(), interp::interp()

Other list tidiers: glance_optim(), list_tidiers, tidy_irlba(), tidy_optim(), tidy_svd()

Examples

```
A <- list(x = 1:5, y = 1:3, z = matrix(runif(5 * 3), nrow = 5))
image(A)
tidy(A)
```

Index