Package ‘plotannotate’

August 22, 2025

Version 1.4-0

Date 2025-08-19

Title Annotate Plots

Depends R (>= 4.2.0)

Imports stats, graphics, grDevices, grid, conicfit

Description Interactively annotate 'base R graphics' plots with freehand drawing, symbols (points, lines, arrows, rectangles, circles, ellipses), and text. This is useful for teaching, for example to visually explain certain plot elements, and creating quick sketches.

License GPL-3

Encoding UTF-8

URL `https://github.com/wviechtb/plotannotate` `https://www.wvbauer.com`

BugReports `https://github.com/wviechtb/plotannotate/issues`

NeedsCompilation no

Author Wolfgang Viechtbauer [aut, cre] (ORCID: <`https://orcid.org/0000-0003-3463-4063`>)

Maintainer Wolfgang Viechtbauer <`wvb@wvbauer.com`>

Repository CRAN

Date/Publication 2025-08-22 18:50:02 UTC

Contents

`plotannotate-package` *plotannotate: Annotate Plots*

Description

The **plotannotate** package allows annotating plots created with 'base R graphics'. The function for this purpose is called `annotate`. After creating a plot, for example with `plot()`, the command `annotate()` enters a freehand drawing mode where one can draw on the plot with the mouse (or some other input device). Various symbols (points, lines, arrows, rectangles, circles, ellipses) and text can also be added. Pressing `q` or the right mouse button terminates the annotation mode. For further details, see the documentation of the `annotate` function.

The functionality provided by the package is especially useful when teaching (e.g., to visually explain certain plot elements), but can also be used for creating quick sketches (see also the `blank` function for creating a blank plot).

Note that the `annotate()` function makes extensive use of `getGraphicsEvent` to capture mouse movements and keyboard inputs. Only some graphics devices support this (currently only `windows` and `x11`). If you receive the error message 'The graphics device does not support event handling', then the graphics device that was opened does not provide this kind of functionality. For example, this will be the case for the `RStudioGD` graphics device that is used by RStudio. You can then try running `x11()` before creating the plot.

Also, `annotate()` does not work for plots based on the **grid** graphics engine (which includes plots created with the **ggplot2** or **lattice** packages), since such plots are not compatible with `getGraphicsEvent`.

Author(s)

Wolfgang Viechtbauer (<`wvb@wvbauer.com`>)

`annotate` *Annotate Plots*

Description

Function to add annotations to a plot.

Usage

```
annotate(col = c("black", "red", "green", "blue"),
         lwd = c(4, 4, 30),
         cex = c(1, 1),
         info = TRUE)
```

Arguments

`col`	a vector of colors (can be up to length 9).
`lwd`	a numeric vector to specify the line width for freehand drawing, for symbols, and the eraser.
`cex`	a numeric vector to specify the expansion factor for points and text.
`info`	a logical to specify whether visual information about the drawing mode and colors should be shown while annotating a plot. Can also be a numeric value to specify a multiplication factor for the visual information.

Details

This is the main function of the package. After creating a plot, for example with `plot()`, the command `annotate()` enters a freehand drawing mode where one can draw on the plot (by pressing the left mouse button and dragging the pointer across the plot). Use the `q` key (or the right mouse button) to quit the `annotate()` function.

Various keyboard keys can be used to select colors and to change the drawing mode (when `info=TRUE`, one can also click on the boxes at the top to select colors and drawing modes):

key	**function**
`1-9`	Select one of the colors from the ones specified via the `col` argument.
`d`	Change to freehand drawing mode as explained above.
`p`	Change to point drawing mode. Click on a location of the plot to add a point.
`l`	Change to line drawing mode. Click on a location, hold the mouse button, move the pointer to a different location, a
`a`	Change to arrow drawing mode. Works like the line drawing mode. The arrowhead is shown at the release location.
`A`	Change to arrow drawing mode with arrowheads at both ends.
`r`	Change to rectangle drawing mode. Works like the line drawing mode.
`c`	Change to circle drawing mode. Works like the line drawing mode. The larger of the x- or y-axis distance determine
`C`	Change to circle drawing mode. Works like the freehand drawing mode, except that the nearest circle corresponding
`o`	Change to ellipse drawing mode. Works like the freehand drawing mode, except that the nearest ellipse correspondi
`t`	Change to text drawing mode. Click on a location (which switches to the type mode) and type some text. Text entry
`e`	Change to eraser mode. Like the freehand drawing mode, except that the background color is used for drawing.

Various keyboard keys can be used to adjust the settings:

key	**function**
↓↑	Decrease/increase the line width or the point/text size. The default sizes are set by the `lwd` and `cex` arguments. Car
←→	Toggle through four different point symbols (circle, square, diamond, and open circle) in point drawing mode or th
`s`	For freehand drawing, toggle smooth mode, so that the line is drawn after running a smoother over the input coordi

Finally, some additional keyboard keys have the following functions:

key	**function**
`u`	Undo (i.e., remove) annotations that have been added to the plot by drawing on top of them with the background colo
`z`	Redraw the entire plot without the annotations. This relies on `recordPlot` and `replayPlot` working correctly.

`x`	Fix the current state of the plot (i.e., runs `recordPlot`) so `z` redraws the plot in this state.
`b`	Make the plot completely blank. See also the `blank` function for creating a blank plot.
`i`	Toggle `info` on/off.

When `info=TRUE` (the default), then visual information about the drawing mode and colors are shown at the top left while annotating a plot. This information may overlap with the plot title or other elements. One can use the `info` argument to specify a numeric value to rescale the visual information to better match the resolution of the plot. Note that the boxes for the drawing modes and colors are clickable.

Resizing plots while or after the `annotate` function has been run is not recommended (all annotations and visual elements need to be redrawn, which can take some time, especially when a lot of annotations have been added to the plot).

After annotating a plot, it should be possible to save the plot including the annotations with `dev.print`, for example using something like `dev.print(device=png, filename="plot.png", width=<width>, height=<height>)`. However, by default, the plot will have a transparent background color, which is typically not what is desired. To avoid this issue, use `par(bg="white")` before creating the plot (i.e., before running something like `plot()`) to set a white background color.

Value

No return value (called for side effects).

Author(s)

Wolfgang Viechtbauer (<`wvb@wvbauer.com`>).

Examples

```
dat <- mtcars
plot(dat$hp, dat$mpg, pch=21, bg="gray", cex=1.5, main="Horsepower versus Mileage")
annotate()
```

`blank` *Create a Blank Plot*

Description

Function to create a blank plot.

Usage

```
blank(annotate = TRUE, bg = "white", ...)
```

Arguments

`annotate`	a logical to specify whether to start the annotation mode after creating the plot (`TRUE` by default).
`bg`	a string to specify the background color of the device region (`"white"` by default).
`...`	arguments that are passed on to `annotate()` when `annotate = TRUE`.

Details

The function creates a blank plot. This is useful if one simply wants to make a quick sketch (and not add annotations to an existing plot).

Value

No return value (called for side effects).

Author(s)

Wolfgang Viechtbauer (`<wvb@wvbauer.com>`).

Examples

```
blank()
```

Index