

Exact distribution of excursions height

Sébastien Déjean - Charly Marti - Sabine Mercier - Sebastian Simon - David Robelin

2026-04-21

Contents

Introduction	1
Computation method	1
Definition of a mathematical excursion	2
Toy examples	3
A study case	4
Optimal excursion, the local score	5
Sub optimal excursions	9
With a reverse lecture of the protein	13
What about the excursion realising the local score	17

Introduction

The goal of this part of the package is to calculate the theoretical probability that the i -th excursion reach the threshold score a , for a Markov chain with a known transition matrix and a given score function. The main function for this objective is `proba_theoretical_ith_excursion_markov()`.

This section of the localScore package is the direct implementation of the pre publication: “Exact distribution of excursion heights of the Lindley process in a Markovian model” written by Carlos Cortés Rojas, Simona Grusea and Sabine Mercier.

Computation method

The main goal of this function is to calculate the probability that the first excursion of the Lindley sequence associated to the studied sequence is greater or equal to a , conditionally to α a potential beginning of this sequence. It also computes the transition probability matrix of the beginnings of excursions: `matrix_M`.

The product of this matrix, elevated to the power $i - 1$, with the first vector of the probabilities conditionally to α , gives the probabilities conditionally to α for the i -th excursion. To return the global probability, the function multiplies the conditional vector with the distribution of the first letter of the sequence.

Definition of a mathematical excursion

The number of an excursion is given by the mathematical definition of Karlin and Altschul (1990). Its corresponds to the number of record times of the Lindley sequence associated to the score sequence $(X_k)_{1 \leq k \leq n}$ and recursively defined as follows:

$$T_0 := 0 \quad \text{and} \quad T_{(k+1)} := \inf\{i \geq T_k + 1, \sum_{j=T_k+1}^i X_j < 0\}.$$

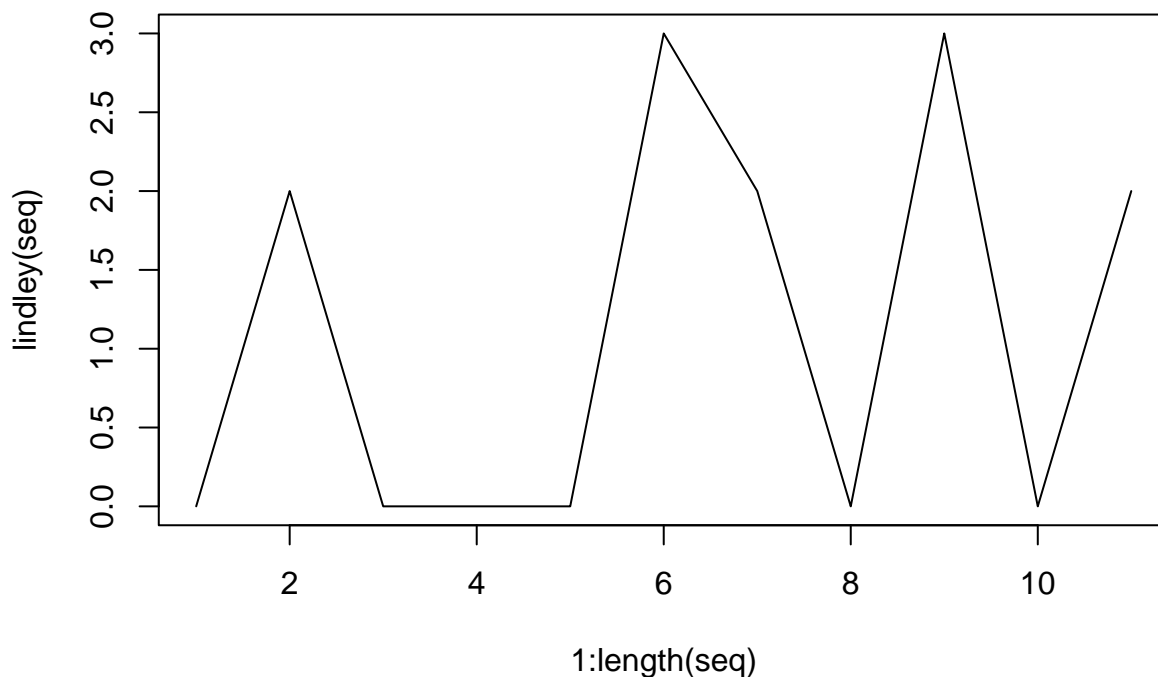
Please note that the sum $\sum_{j=T_k+1}^i X_j$ must be non positive. If it is equal to zero, it will mathematically be still the same Lindley excursion.

$$(A_k)_k \quad \text{with} \quad T_{(i-1)} + 1 \leq k < T_i$$

is called the i -th positive excursion of the sequence in the term of Karlin and Altschul. This mathematical definition includes the index with a Lindley score equal to 0, whereas visually we “see” the excursion beginning at the following index.

The mathematical number of an excursion must be distinguished from the one of the corresponding visible excursion because of the possible appearance of flat excursions. See below for an example.

```
seq <- c(-1,2,-3,-4,-1,3,-1,-2,3,-4,2)
lindley(seq)
#> [1] 0 2 0 0 0 3 2 0 3 0 2
plot(1:length(seq), lindley(seq), type = 'l')
```



```
localScoreC(seq)
#> $localScore
#> value begin end
#> 3 6 6
#>
#> $suboptimalSegmentScores
```

```
#>   value begin end
#> 1     2     2  2
#> 2     3     6  6
#> 3     3     9  9
#> 4     2    11 11
#>
#> $RecordTime
#> [1] 0 1 3 4 5 10
```

We can “see” three positive excursions, with top at index $\{2; 6; 9\}$; plus one last not achieved at index $\{11\}$. There exists also five flats excursions: at index $\{1; 3; 4; 5; 10\}$. Note also that the Lindley excursion 6-10 reach 0 at index 8 without going in non positive values, so it is still the same excursion.

In summary, the excursions list is: $\{1; 2 - 3; 4; 5; 6 - 10; 11\}$. The following function gives the record time of the mathematical excursion that must be used to compute the statistical significance. Note that we choose to omit the conventional first record time which is always 0.

```
recordTimes(seq)
#> [1] 1 3 4 5 10
```

So the sub optimal excursion values given by `$suboptimalSegmentScores` 2 3 3 2 are achieved by the excursion number 2, 5 (twice) and 6. At this point, we wish to lighten a difference between the notion of “excursion” defined mathematically by Karlin and Dembo (1992), and the results given in `$suboptimalSegmentScores`. The segment [6-10] is an excursion in mathematical sense as the cumulative process starting at position 7 never hit a negative value before the 10th position. Nevertheless this process hit a null value indicating that this single excursion contain at least two positive score segments (here : at position 6 and 9). We chose to indicate in `$suboptimalSegmentScores` all positive consecutive sub-sequences, as it can be more meaningful when applied on real data.

To calculate the p -value of a given excursion, we have to know which number of excursion it is in sequential order. Notice that it will not bring any problem of making a mistake in the number of the excursion for a number exceeding 10, and a small difference for a lower number, as the sequence composed by the begin-component of each excursion, which is also a Markov chain, can reach the stationary distribution. For the first excursions, it is far more easy to recover the record time of the corresponding mathematical excursion in the list.

Toy examples

As the function calculates a theoretical probability, we don’t need a sequence of scores but the transition matrix of the markov chain: Λ . It also requires a score function with: integer scores, a negative expectation and at least one possible positive score. We also need θ , an alphabet (can contains numbers) with unique values ; and i : the rank of the excursion on which we calculate the probabilities. The optional parameters are `epsilon` and `prob0`. `epsilon` is a threshold for the computation of matrix M and `prob0` is the probability distribution of the first letter of the sequence. Obviously, `theta`, the score function and `prob0` should have the same length p , and the dimension of `lambda` should be (p, p) .

In this example, we want to know the probability that the second excursion reach a score greater or equal than 5 with the matrix Λ and a score function of $(-1, 1)$, given that the first score of the sequence have 50 percent chances to be -1.

```
proba_theoretical_ith_excursion_markov(a = 5,
                                         theta = c("A", "B"),
                                         lambda = matrix(c(0.5, 0.5, 0.8, 0.2),
```

```

                                ncol = 2, byrow = TRUE),
score_function = c(-1,1),
i = 2,
epsilon = 1e-16,
prob0 = c(0.5, 0.5))

#> $proba_q_i_geq_a
#> [1] 0.009661836
#>
#> $P_alpha
#>      A      B
#> 0.009661836 0.009661836

```

In this following example, we see that this is not a problem to have missing score values in the score function, and that theta can contain numbers. Initial value of `epsilon` is `1e-16`, and if we don't precise the value of `prob0`, the function compute the stationary distribution of Λ , and use it for the distribution of the first letter of the sequence.

```

proba_theoretical_ith_excursion_markov(a = 5,
                                theta = c("K", 12, 1.54),
                                lambda = matrix(c(0.9, 0, 0.1, 0.9, 0.1, 0, 0, 0.8, 0.2),
                                                ncol = 3, byrow = TRUE),
                                score_function = c(-3,-1,2),
                                i = 4)

#> $proba_q_i_geq_a
#> [1] 0.00405119
#>
#> $P_alpha
#>      K      12      1.54
#> 0.004054779 0.004050819 0.004022805

```

In the following example, the function takes longer as we increased the number of scores (complexity of $O(\text{length}(\theta)^3)$), but here with 20 scores, the computational time is acceptable.

```

transition_matrix <- matrix(runif(400, min = 0, max = 1), nrow = 20)
transition_matrix <- t(apply(transition_matrix, 1, function(x) x/sum(x)))
theta <- letters[1:20]
score_f <- c(-3,-1,2,-3,-1,2,-3,-1,2,-1,
            -3,-1,2,-3,-1,2,-3,-1,2,-1)
sum(stationary_distribution(transition_matrix)*score_f) #score expectation (stationary)
#> [1] -0.6445784
system.time(pv1 <- proba_theoretical_ith_excursion_markov(a=5, theta, transition_matrix, score_f, i = 4))
#> utilisateur      système      écoulé
#>      5.501      0.004      5.513
pv1$proba_q_i_geq_a
#> [1] 0.06826226

```

A study case

Let us consider the protein Seq1093 of 1093 amino acid proposed in the package `localScore` which corresponds to the Q60519.fasta in UniProt Data base.

Optimal excursion, the local score

Using the hydrophobic score scale called `HydroScore`, we compute the local score, corresponding to the height of the highest excursion.

```
library(localScore)
data("Seq1093")
Seq1093
#> [1] "MVVPGPLALSLSSLTLLVSHLSSSQDIASESSSEQQMCTREHPIVAFEDLKPWFNFNFTYPGVRDFSQALDPSRNQLIVGARNYLFRLSLAN
nchar(Seq1093)
#> [1] 1093
data(HydroScore)
LongSeqScore <- CharSequence2ScoreSequence(Seq1093, HydroScore)
table(LongSeqScore)
#> LongSeqScore
#>  -5  -4  -3  -2  -1   0   2   3   4   5
#>  81 222  22  82 232  85  78 103 157  31
localScoreC(LongSeqScore)
#> $localScore
#> value begin end
#>    65   956 1001
#>
#> $suboptimalSegmentScores
#>   value begin end
#> 1     40     1  20
#> 2     10    71  73
#> 3     23    80  99
#> 4      3   114 114
#> 5      3   124 124
#> 6      4   128 128
#> 7     23   130 150
#> 8     16   181 195
#> 9      2   217 217
#> 10     4   224 224
#> 11    33   229 243
#> 12     5   294 296
#> 13     4   301 301
#> 14     6   304 307
#> 15    36   312 337
#> 16     4   379 379
#> 17     4   384 384
#> 18     2   387 387
#> 19    14   390 397
#> 20     4   411 411
#> 21     3   413 413
#> 22    25   416 486
#> 23     9   523 527
#> 24     9   533 537
#> 25     4   548 548
#> 26     3   554 554
#> 27     3   563 563
#> 28     4   566 566
#> 29    10   574 576
#> 30    16   588 601
```

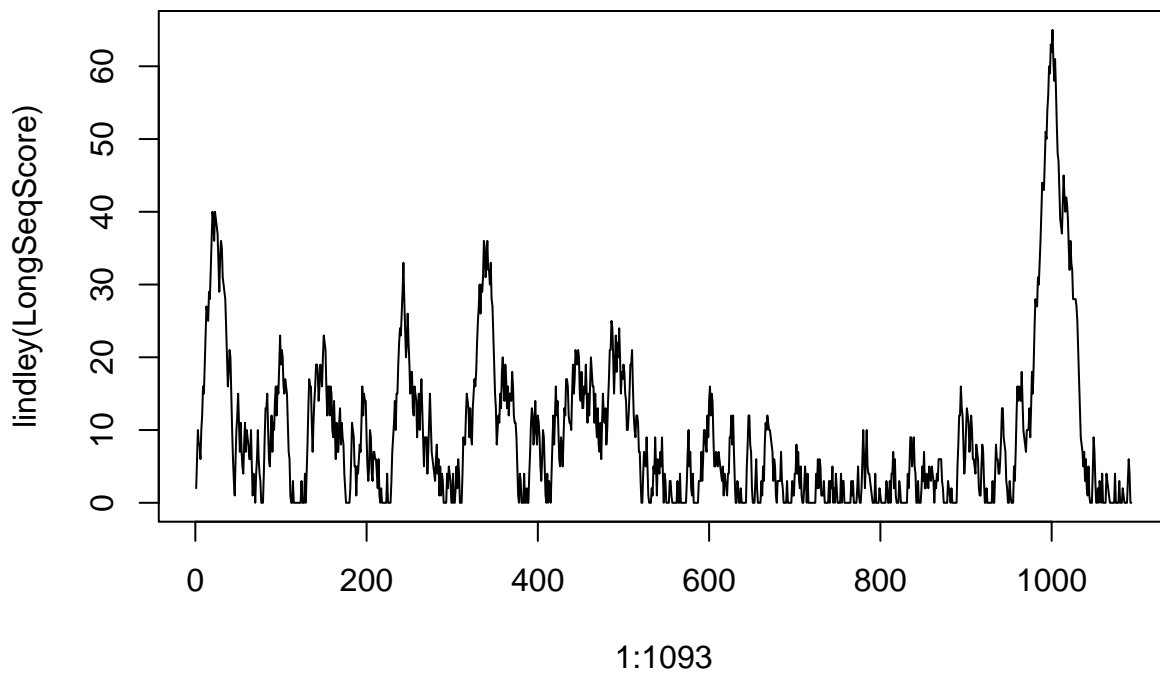
```

#> 31      3    633    633
#> 32      2    637    637
#> 33     12    644    646
#> 34      6    654    655
#> 35     12    661    668
#> 36      7    680    684
#> 37      3    691    691
#> 38      8    697    702
#> 39      5    711    712
#> 40      4    715    715
#> 41      6    725    727
#> 42      3    740    740
#> 43      5    742    744
#> 44      2    746    746
#> 45      4    748    748
#> 46      4    754    754
#> 47      3    756    756
#> 48      3    766    766
#> 49      5    773    774
#> 50     10    778    780
#> 51      4    794    794
#> 52      2    799    799
#> 53      3    807    807
#> 54      3    811    811
#> 55      7    813    815
#> 56      3    822    822
#> 57      9    832    835
#> 58      3    844    844
#> 59      7    848    851
#> 60      6    864    868
#> 61      3    879    879
#> 62      2    883    883
#> 63     16    890    894
#> 64      2    924    924
#> 65      3    931    931
#> 66     13    934    942
#> 67      3    951    951
#> 68     65    956   1001
#> 69      9   1048   1049
#> 70      2   1054   1054
#> 71      3   1056   1056
#> 72      4   1059   1059
#> 73      4   1064   1064
#> 74      4   1074   1074
#> 75      3   1079   1079
#> 76      2   1083   1083
#> 77      6   1089   1090
#>
#> $RecordTime
#>   [1]      0    70    77    78    79   112   113   115   116   117   119   120   121   122   123
#>  [16]   125   126   127   129   176   177   178   179   180   218   219   220   221   222   223
#>  [31]   226   227   228   290   291   293   299   300   303   310   311   378   381   382   383
#>  [46]   386   388   389   412   415   521   522   531   532   550   551   558   559   560   561

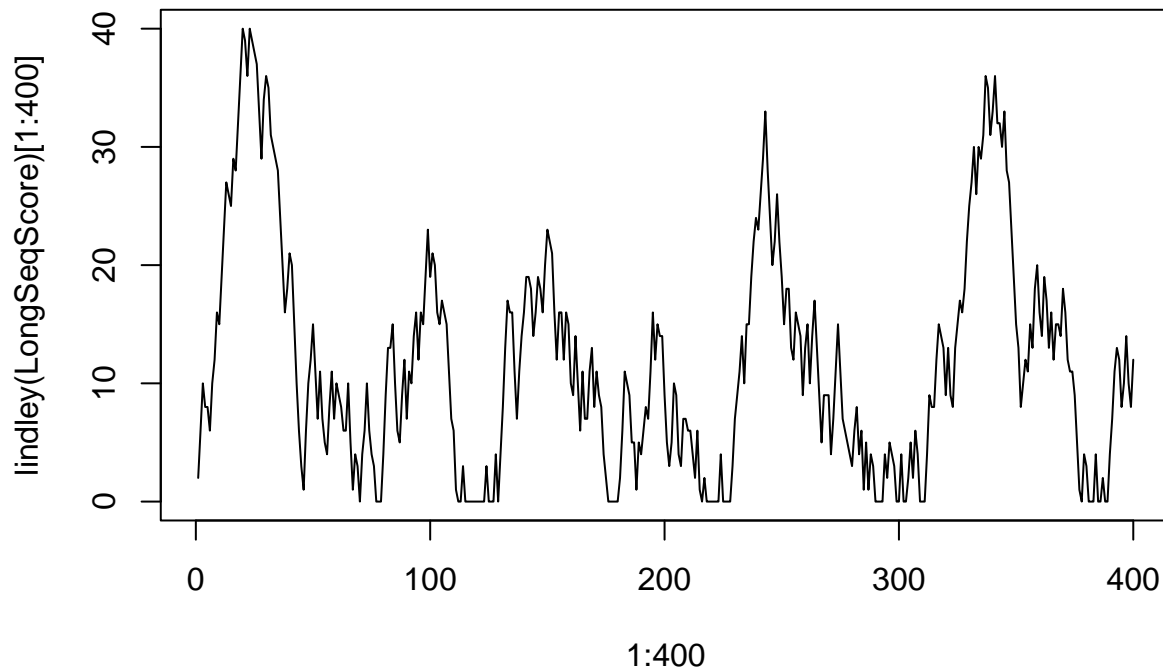
```

```
#> [61] 562 564 565 569 570 571 573 583 584 585 586 587 631 632 635
#> [76] 636 639 640 643 651 652 653 657 658 659 660 679 687 688 689
#> [91] 690 692 693 695 696 709 710 714 716 717 718 719 720 721 722
#> [106] 723 724 735 736 737 738 739 741 750 751 752 758 759 760 761
#> [121] 762 763 764 765 769 771 777 791 792 793 798 803 804 805 806
#> [136] 809 810 812 819 820 821 824 825 826 827 828 829 843 845 846
#> [151] 847 863 874 875 876 877 878 882 885 886 888 889 922 925 926
#> [166] 927 928 929 930 932 933 949 950 954 955 1045 1046 1047 1052 1053
#> [181] 1055 1061 1062 1063 1069 1070 1071 1072 1073 1076 1077 1078 1080 1081 1082
#> [196] 1085 1087 1088 1093
```

```
head(sort(localScoreC(LongSeqScore)$suboptimalSegmentScores[,1], decreasing = TRUE))
#> [1] 65 40 36 33 25 23
plot(1:1093,lindley(LongSeqScore), type = 'l')
```



```
plot(1:400,lindley(LongSeqScore)[1:400], type = 'l')
```



We get: local score = 65. Let us compute its p -value in a I.I.D. model.

```
LS <- localScoreC(LongSeqScore)$localScore["value"]
probl <- scoreSequences2probabilityVector(list(LongSeqScore))
probl
#>      -5      -4      -3      -2      -1      0      1
#> 0.07410796 0.20311070 0.02012809 0.07502287 0.21225984 0.07776761 0.00000000
#>      2      3      4      5
#> 0.07136322 0.09423605 0.14364135 0.02836231
daudin(local_score = LS, sequence_length = length(LongSeqScore),
       score_probabilities = probl,
       sequence_min = min(LongSeqScore),
       sequence_max = max(LongSeqScore))
#> [1] 0.07231933
```

In an independent model the p -value equals 7.2%, that is non significant using the nominal level 5%.

Let us compute it in a markovian model. We need the transition matrix.

```
tmpMarkovParameters <- sequences2transmatrix(LongSeqScore)
lambda <- tmpMarkovParameters$transition_matrix
apply(lambda, 1, sum) #to check stochasticity
#> -5 -4 -3 -2 -1 0 2 3 4 5
#> 1 1 1 1 1 1 1 1 1 1
prob0 <- stationary_distribution(lambda)
print(prob0)
#> [1] 0.07415553 0.20331534 0.02016219 0.07512607 0.21241680 0.07789931
#> [7] 0.07052191 0.09425453 0.14375820 0.02839011
score_values <- tmpMarkovParameters$score_value
print(score_values)
#> [1] -5 -4 -3 -2 -1 0 2 3 4 5
```

Notice that the score value 1 is not present in the sequence.


```
exact_mc(LS, lambda, sequence_length = length(LongSeqScore), score_values = score_values)
#> [1] 0.07695456
```

The p -value in a Markovian model, equal to 7.7% is very similar in this case with the one of the independent model.

Sub optimal excursions

Let us consider a study on the first and the last sub optimal excursions.

```
subOptSegment <- localScoreC(LongSeqScore)$suboptimalSegmentScores
o <- order(subOptSegment[,1], decreasing = TRUE)
subOptSegment <- subOptSegment[o,] # reordering segments by decreasing score values
print(subOptSegment)
```

#>	value	begin	end
#> 68	65	956	1001
#> 1	40	1	20
#> 15	36	312	337
#> 11	33	229	243
#> 22	25	416	486
#> 3	23	80	99
#> 7	23	130	150
#> 8	16	181	195
#> 30	16	588	601
#> 63	16	890	894
#> 19	14	390	397
#> 66	13	934	942
#> 33	12	644	646
#> 35	12	661	668
#> 2	10	71	73
#> 29	10	574	576
#> 50	10	778	780
#> 23	9	523	527
#> 24	9	533	537
#> 57	9	832	835
#> 69	9	1048	1049
#> 38	8	697	702
#> 36	7	680	684
#> 55	7	813	815
#> 59	7	848	851
#> 14	6	304	307
#> 34	6	654	655
#> 41	6	725	727
#> 60	6	864	868
#> 77	6	1089	1090
#> 12	5	294	296
#> 39	5	711	712
#> 43	5	742	744
#> 49	5	773	774
#> 6	4	128	128
#> 10	4	224	224

```

#> 13      4    301  301
#> 16      4    379  379
#> 17      4    384  384
#> 20      4    411  411
#> 25      4    548  548
#> 28      4    566  566
#> 40      4    715  715
#> 45      4    748  748
#> 46      4    754  754
#> 51      4    794  794
#> 72      4   1059 1059
#> 73      4   1064 1064
#> 74      4   1074 1074
#> 4       3    114  114
#> 5       3    124  124
#> 21      3    413  413
#> 26      3    554  554
#> 27      3    563  563
#> 31      3    633  633
#> 37      3    691  691
#> 42      3    740  740
#> 47      3    756  756
#> 48      3    766  766
#> 53      3    807  807
#> 54      3    811  811
#> 56      3    822  822
#> 58      3    844  844
#> 61      3    879  879
#> 65      3    931  931
#> 67      3    951  951
#> 71      3   1056 1056
#> 75      3   1079 1079
#> 9       2    217  217
#> 18      2    387  387
#> 32      2    637  637
#> 44      2    746  746
#> 52      2    799  799
#> 62      2    883  883
#> 64      2    924  924
#> 70      2   1054 1054
#> 76      2   1083 1083

```

The sub optimal scores are 40, 36 and 33 realized by the “visual” excursions number 1, 15 and 11. Such number of excursion, 11 and 15 are large enough to avoid considering the mathematical number. The first excursion is mathematically the first one (see below).

```

lindley(LongSeqScore)
#>      [1]  2  6 10  8  8  6 10 12 16 15 19 23 27 26 25 29 28 32 36 40 39 36 40 39
#>     [25] 38 37 33 29 34 36 35 31 30 29 28 24 20 16 18 21 20 15 10  6  3  1  6 10
#>     [49] 12 15 11  7 11  7  5  4  8 11  7 10  9  8  6  6 10  5  1  4  3  0  4  6
#>     [73] 10  6  4  3  0  0  0  4  9 13 13 15 10  6  5  9 12  7 11 10 14 16 12 16
#>     [97] 15 19 23 19 21 20 16 15 17 16 15 11  7  6  1  0  0  3  0  0  0  0  0  0
#>    [121]  0  0  0  3  0  0  0  4  0  4  8 13 17 16 16 11  7 11 14 16 19 19 18 14
#>    [145] 16 19 18 16 20 23 22 21 16 12 16 16 12 16 15 10  9 14 10  6 11  7  7 11

```

```

#> [169] 13 8 11 9 8 4 2 0 0 0 0 0 2 6 11 10 9 5 5 1 5 4 6 8
#> [193] 7 11 16 12 15 14 14 9 5 3 5 10 9 4 3 7 7 6 6 4 2 6 1 0
#> [217] 2 0 0 0 0 0 0 4 0 0 0 0 3 7 9 11 14 10 15 15 19 22 24 23
#> [241] 26 29 33 28 24 20 22 26 22 19 15 18 18 13 12 16 15 14 9 13 15 10 14 17
#> [265] 13 9 5 9 9 9 4 7 11 15 11 7 6 5 4 3 6 8 4 6 1 5 1 4
#> [289] 3 0 0 0 0 4 2 5 4 3 0 0 4 0 0 2 5 2 6 4 0 0 0 4
#> [313] 9 8 8 12 15 14 13 9 13 9 8 13 15 17 16 18 22 25 27 30 26 30 29 31
#> [337] 36 35 31 33 36 32 32 30 33 28 27 23 19 15 13 8 10 12 11 15 13 18 20 16
#> [361] 14 19 17 13 16 12 15 15 14 18 16 12 11 11 9 5 1 0 4 3 0 0 0 4
#> [385] 0 0 2 0 0 4 7 11 13 12 8 10 14 10 8 12 11 9 5 3 6 10 9 5
#> [409] 1 0 4 0 3 2 0 4 8 12 8 12 16 12 14 10 6 5 9 8 5 9 13 12
#> [433] 17 17 16 12 11 11 10 15 19 15 17 21 20 19 21 20 15 14 18 13 13 16 15 19
#> [457] 15 11 15 12 16 20 18 16 16 11 15 11 9 13 8 7 11 6 11 15 12 11 13 8
#> [481] 10 14 17 21 21 25 24 20 15 19 23 18 22 20 24 20 15 18 17 19 18 15 14 10
#> [505] 10 12 15 19 19 21 16 12 10 9 12 12 11 7 7 3 0 0 4 7 6 5 9 5
#> [529] 1 0 0 0 2 1 5 4 9 5 1 6 5 4 7 5 9 4 0 4 3 0 0 0
#> [553] 0 3 3 1 0 0 0 0 0 0 3 0 0 4 0 0 0 0 0 0 0 3 7 10
#> [577] 5 7 2 1 4 0 0 0 0 0 0 3 3 7 3 6 10 10 8 7 12 9 14
#> [601] 16 12 15 14 9 5 5 7 6 5 7 6 5 4 3 5 1 4 3 2 1 4 4 9
#> [625] 9 12 8 12 7 3 0 0 3 2 0 0 2 0 0 0 0 0 0 5 8 12 12 8
#> [649] 7 2 0 0 0 3 6 2 0 0 0 0 3 1 5 3 8 11 10 12 11 10 10 9
#> [673] 8 7 3 6 5 1 0 3 3 3 3 7 3 2 0 0 0 0 3 0 0 0 0 0
#> [697] 3 1 1 4 4 8 4 7 3 2 5 1 0 0 2 5 3 0 4 0 0 0 0 0
#> [721] 0 0 0 0 4 2 6 2 6 5 1 1 1 3 0 0 0 0 0 3 0 3 2 5
#> [745] 0 2 0 4 2 0 0 0 0 4 0 3 3 0 0 0 0 0 0 0 0 3 1 3
#> [769] 0 0 0 0 2 5 1 0 0 2 6 10 6 2 6 10 5 4 4 3 2 1 0 0
#> [793] 0 4 0 0 0 0 2 1 0 0 0 0 0 0 3 2 0 0 3 0 4 4 7 2
#> [817] 6 1 0 0 0 3 2 0 0 0 0 0 0 0 4 2 5 9 9 5 7 9 5
#> [841] 4 0 0 3 0 0 0 2 5 3 7 2 2 4 3 2 5 4 3 5 4 3 0 3
#> [865] 2 4 3 6 6 6 6 3 2 0 0 0 0 0 3 2 1 0 2 0 0 0 0 0
#> [889] 0 5 8 12 12 16 13 12 8 4 6 10 13 12 11 7 9 12 10 6 6 5 4 8
#> [913] 7 6 2 1 1 5 8 7 3 0 0 2 0 0 0 0 0 0 3 0 0 4 8 6
#> [937] 6 4 4 6 9 13 13 9 8 7 3 2 0 0 3 1 0 0 0 5 3 7 12 16
#> [961] 14 16 15 14 18 14 10 9 8 7 10 10 10 13 9 13 18 15 19 24 28 27 27 31
#> [985] 30 33 36 40 44 43 43 47 51 50 54 56 60 59 63 62 65 61 58 61 57 52 48 47
#> [1009] 43 39 38 37 41 45 42 40 42 41 39 35 32 36 33 32 28 28 28 28 27 25 21 17
#> [1033] 13 9 8 7 5 7 3 6 2 1 5 1 0 0 0 4 9 7 3 0 0 2 0 3
#> [1057] 2 0 4 0 0 0 0 4 3 2 1 0 0 0 0 4 0 0 0 0 0 3 0
#> [1081] 0 0 2 1 0 0 0 0 3 6 4 0 0

```

recordTimes(LongSeqScore)

```

#> [1] 70 77 78 79 112 113 115 116 117 119 120 121 122 123 125
#> [16] 126 127 129 176 177 178 179 180 218 219 220 221 222 223 226
#> [31] 227 228 290 291 293 299 300 303 310 311 378 381 382 383 386
#> [46] 388 389 412 415 521 522 531 532 550 551 558 559 560 561 562
#> [61] 564 565 569 570 571 573 583 584 585 586 587 631 632 635 636
#> [76] 639 640 643 651 652 653 657 658 659 660 679 687 688 689 690
#> [91] 692 693 695 696 709 710 714 716 717 718 719 720 721 722 723
#> [106] 724 735 736 737 738 739 741 750 751 752 758 759 760 761 762
#> [121] 763 764 765 769 771 777 791 792 793 798 803 804 805 806 809
#> [136] 810 812 819 820 821 824 825 826 827 828 829 843 845 846 847
#> [151] 863 874 875 876 877 878 882 885 886 888 889 922 925 926 927
#> [166] 928 929 930 932 933 949 950 954 955 1045 1046 1047 1052 1053 1055
#> [181] 1061 1062 1063 1069 1070 1071 1072 1073 1076 1077 1078 1080 1081 1082 1085

```

```
#> [196] 1087 1088 1093
LongSeqScore[1] > 0
#> [1] TRUE
```

Let us consider a study on the first and the last sub optimal excursions in the sequential order of the sequence. First excursion height equal to 40; last one, the 77th one, height equal to 6.

```
subOptSegment["1",] #First excursion
#>   value begin end
#> 1    40     1  20
subOptSegment[as.character(dim(subOptSegment)[1]),] #Last excursion
#>   value begin end
#> 77     6 1089 1090
```

Let us compute their p -values.

```
theta <- letters[1:length(score_values)] # arbitrary
score_function <- score_values           # defined earlier
a <- 40
i <- 1
if (FALSE) { # To avoid long computation on CRAN servers... (Comment if you
# want to try)
  system.time(pv2 <- proba_theoretical_ith_excursion_markov(a, theta, lambda,
score_function,i)$proba_q_i_geq_a)
} else {
  pv2 <- 0.1741784
}
#> utilisateur système écoulé
#> 38.061      0.033   38.137
pv2
#> [1] 0.1741784

a <- 6
i <- 77
system.time(pv3<-proba_theoretical_ith_excursion_markov(a, theta, lambda,
score_function, i)$proba_q_i_geq_a)
#> utilisateur système écoulé
#> 1.950      0.012   1.964
pv3
#> [1] 0.1741784
```

First excursion : height equal to 40 with p -value=0.45%; Last excursion, the 77th one, height equal to 6 and p -value=17%.

The time computation of the first p -value is larger because of the larger value of a .

We can consider that from the 20th mountain we reached the stationary distribution for the beginning of excursion. We shall therefore take a lower value for i for the last excursion to evaluate the difference. Let us try $i = 20$ instead of 77.

```
a <- 6
i <- 20
system.time(pv4 <- proba_theoretical_ith_excursion_markov(a, theta,
```

```

                                lambda,
                                score_function,i)$proba_q_i_geq_a)
#> utilisateur      système      écoulé
#>      1.872      0.002      1.875
pv4
#> [1] 0.1741784

i <- 10
system.time(pv5 <- proba_theoretical_ith_excursion_markov(a, theta,
                                lambda,
                                score_function,i)$proba_q_i_geq_a)
#> utilisateur      système      écoulé
#>      1.774      0.000      1.775
pv5
#> [1] 0.1741784

```

We obtain the same value as expected even for $i = 10$.

With a reverse lecture of the protein

As the lecture of a protein could be done in both direction, we also consider the reverse sequence.

```

LongSeqScore.inv <- rev(LongSeqScore)
localScoreC(LongSeqScore.inv)
#> $localScore
#> value begin  end
#>    65    93   138
#>
#> $suboptimalSegmentScores
#>   value begin  end
#> 1     6     4    5
#> 2     2    11   11
#> 3     3    15   15
#> 4     4    20   20
#> 5     4    30   30
#> 6     4    35   35
#> 7     2    40   40
#> 8     9    45   46
#> 9     4    51   51
#> 10    3    54   54
#> 11    2    56   56
#> 12    4    72   72
#> 13    2    77   77
#> 14    8    80   81
#> 15    3    90   90
#> 16   65    93  138
#> 17    7   243  246
#> 18    3   250  250
#> 19    4   255  256
#> 20    9   259  262
#> 21    3   272  272
#> 22    4   277  277

```

```

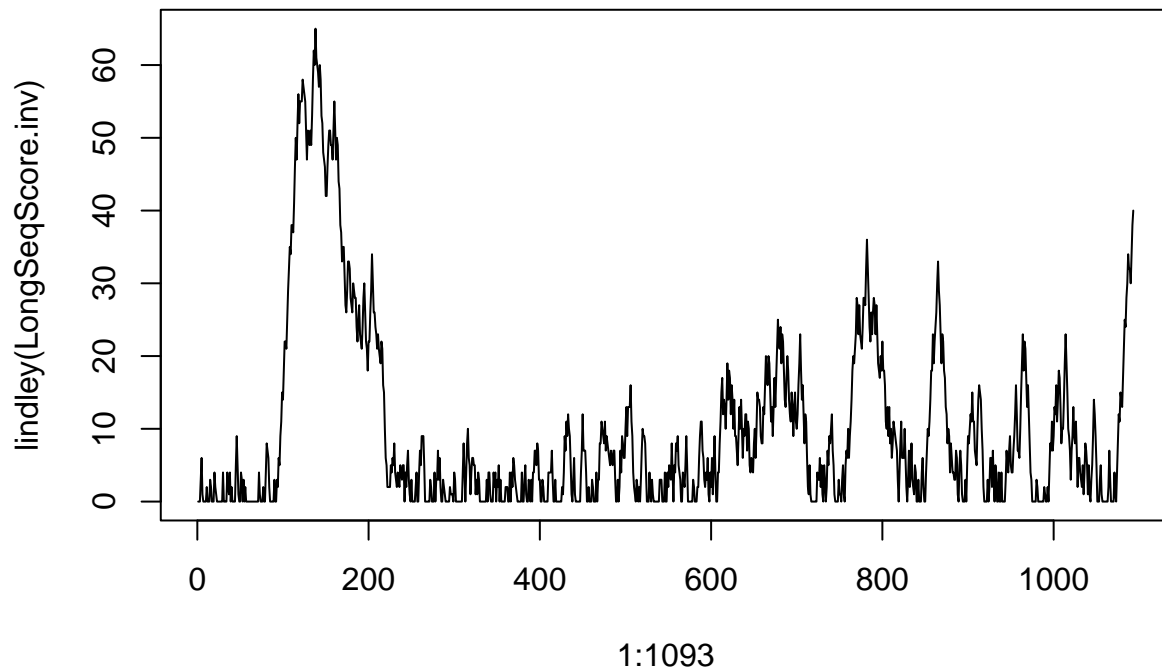
#> 23      7  279 281
#> 24      3  287 287
#> 25      2  295 295
#> 26      4  300 300
#> 27      8  310 311
#> 28     10  314 316
#> 29      3  338 338
#> 30      4  340 340
#> 31      4  346 346
#> 32      5  350 352
#> 33      3  354 354
#> 34      2  360 360
#> 35      4  365 365
#> 36      6  367 369
#> 37      4  379 379
#> 38      5  382 383
#> 39      3  387 387
#> 40      3  390 390
#> 41      8  392 397
#> 42      3  403 403
#> 43      7  410 414
#> 44      3  418 418
#> 45     12  426 433
#> 46      6  439 440
#> 47     12  448 450
#> 48      2  457 457
#> 49      3  461 461
#> 50      4  466 466
#> 51     11  468 472
#> 52      3  491 491
#> 53     16  493 506
#> 54      3  513 513
#> 55      2  516 516
#> 56     10  518 520
#> 57      4  528 528
#> 58      3  531 531
#> 59      3  540 540
#> 60      4  546 546
#> 61      8  549 554
#> 62      9  557 561
#> 63      9  567 571
#> 64      3  579 579
#> 65     11  584 588
#> 66      9  599 604
#> 67     25  608 678
#> 68      6  724 727
#> 69      5  732 732
#> 70     12  735 741
#> 71      3  749 749
#> 72      5  753 754
#> 73     36  757 782
#> 74     11  820 822
#> 75      8  830 834

```

```

#> 76      3  842  842
#> 77      6  846  847
#> 78     33  851  865
#> 79      7  890  891
#> 80      3  897  897
#> 81     16  899  913
#> 82      3  923  923
#> 83      7  925  929
#> 84      5  932  932
#> 85      4  936  936
#> 86      4  939  939
#> 87     23  944  964
#> 88      3  980  980
#> 89      2  989  989
#> 90      2  993  993
#> 91     23  995 1014
#> 92     14 1044 1047
#> 93      5 1054 1055
#> 94      7 1064 1065
#> 95      4 1071 1071
#> 96     40 1074 1093
#>
#> $RecordTime
#> [1]      0      1      2      3      7      9     10     12     13     14     18     19     23     24     25
#> [16]     26     27     28     29     32     33     34     41     42     43     44     49     50     53     55
#> [31]     58     59     60     61     62     63     64     65     69     70     71     74     75     76     79
#> [46]     85     86     87     88     89     92    249    251    252    253    254    267    268    269    270
#> [61]    271    274    275    276    278    285    286    291    293    294    299    303    304    305    306
#> [76]    308    309    330    331    332    333    334    335    336    339    343    344    345    349    355
#> [91]    356    357    358    359    363    364    374    375    376    377    378    381    385    386    389
#> [106]   391    401    402    405    406    407    408    409    416    417    419    420    421    422    424
#> [121]   425    438    442    443    444    445    446    455    456    459    460    463    464    465    489
#> [136]   490    492    512    515    517    527    530    533    534    535    536    537    538    543    544
#> [151]   545    548    565    566    574    576    577    582    583    598    607    717    718    719    721
#> [166]   722    723    731    734    745    746    747    748    752    819    829    840    843    844    845
#> [181]   849    850    894    896    898    920    921    922    924    931    934    935    941    942    943
#> [196]   975   977   978   979   982   983   984   985   986   987   988   991   992   994  1043
#> [211]  1052  1053  1057  1058  1059  1060  1061  1062  1063  1067  1068  1069  1070
head(sort(localScoreC(LongSeqScore.inv)$suboptimalSegmentScores[,1], decreasing = TRUE))
#> [1] 65 40 36 33 25 23
plot(1:1093,lindley(LongSeqScore.inv), type = 'l')

```



That leads to: first excursion equal to 6; last excursion, the 96th one, equal to 38.

The corresponding p -values are:

```
markov_parameters <- sequences2transmatrix(LongSeqScore.inv)
lambda.inv <- markov_parameters$transition_matrix
if (FALSE) {
  system.time(pv5 <- proba_theoretical_ith_excursion_markov(a = 38, theta,
                                                            lambda.inv,
                                                            score_function, i = 96
  )$proba_q_i_geq_a)
} else {
  pv5 <- 0.005460216
}
#> utilisateur système écoulé
#> 35.478      0.046    35.573
pv5
#> [1] 0.005460216
proba_theoretical_ith_excursion_markov(a = 6, theta, lambda.inv,
                                       score_function, i = 1
  )$proba_q_i_geq_a
#> [1] 0.1765172
```

Last excursion, the 96th one, equal to 38, with p -value equal to 0.55%. First excursion equal to 6, with p -value equal to 17%.

The last excursion is still significant. The first one is still non significant.

Remark : Even with a Bonferroni correction to take into account the multiple test (here two studies excursions), the sequence possesses a significant segment. Whereas considering the highest value over all the excursions of the whole sequence, the local score value, the sequence is not significant.

What about the excursion realising the local score

Let us consider the excursion 30 as an excursion among the others. For the first way to read the protein:

```
a <- 65
i <- 30
if (FALSE) { # To avoid long computation on CRAN servers... (Comment if you
                                     # want to try)
  system.time(pv6 <- proba_theoretical_ith_excursion_markov(a, theta,
                                                           lambda, score_function,
                                                           i)$proba_q_i_geq_a)
} else {
  pv6 <- 0.0004221151
}
#> utilisateur système écoulé
#> 104.861      0.101    105.062
pv6
#> [1] 0.0004221151
```

There is a less than 4 in 10,000 chance of having a first mountain over 65.

Or with the second way to read the protein:

```
subOptSegment.inv <- localScoreC(LongSeqScore.inv)$suboptimalSegmentScores
which.max(subOptSegment.inv[,1])
#> [1] 16
print(subOptSegment.inv[16,])
#>      value begin end
#> 16      65     93 138

a <- 65
i <- 16
if (FALSE) {# To avoid long computation on CRAN servers... (Comment if you
                                     # want to try)
  system.time(pv7 <- proba_theoretical_ith_excursion_markov(a, theta, lambda.inv,
                                                           score_function,i)$proba_q_i_geq_a)
} else {
  pv7 <- 0.0004379929
}
#> utilisateur système écoulé
#> 103.018      0.088    103.229
pv7
#> [1] 0.0004379929
```

As expected we obtain a similar p -value.

Although the highest mountain of height 65 in this sequence of length 1093 does not have a significant value at the 5% threshold, it does not mean that the sequence has not interesting segments. Observing a mountain exceeding 40 is significant for example, and it is the same for values 36 and 33. All these tests remain significant even if a correction of type Bonferroni is taken.