Package: localScore (via r-universe)

November 3, 2024

Type Package

Title Package for Sequence Analysis by Local Score

Version 1.0.11

Date 2023-11-02

Copyright See the file COPYRIGHTS for various embedded Eigen library copyright details

Maintainer David Robelin <david.robelin@inrae.fr>

Description Functionalities for calculating the local score and calculating statistical relevance (p-value) to find a local Score in a sequence of given distribution (S. Mercier and J.-J. Daudin (2001) <https://hal.science/hal-00714174/>) ; S. Karlin and S. Altschul (1990) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC53667/> ; S. Mercier, D. Cellier and F. Charlot (2003) <https://hal.science/hal-00937529v1/> ; A. Lagnoux, S. Mercier and P. Valois (2017) <doi:10.1093/bioinformatics/btw699>).

License GPL (>= 2) | file LICENSE

Imports Rcpp (>= 0.12.16), utils

LinkingTo Rcpp

RoxygenNote 7.2.3

LazyData true

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Depends R (>= 3.2.0)

NeedsCompilation yes

Author Sebastian Simon [aut], David Robelin [aut, cre], Sabine Mercier [aut], Sebastien Dejean [aut], The authors of Eigen the library for the included version of Eigen [cph]

Repository CRAN

Date/Publication 2023-11-02 22:20:06 UTC

Contents

`localScore-package` *Package for sequence analysis by local score*

Description

Provides functionalities for:

- calculating the local score
- calculating statistical relevance (p-value) to find a local Score in a sequence of given distribution.

Details

Please refer to the vignette of this package or the manual for details on how to use this package.

Author(s)

Sebastian Simon, Sabine Mercier, David Robelin, Anne-Benedicte Urbano

Maintainer: David Robelin david.robelin@inra.fr

References

p-value:

- An Improved Approximation For Assessing The Statistical Significance of molecular Sequence Features, Mercier and al 2003
- Exact distribution for the local score of one i.i.d. random sequence, Sabine Mercier and JJ Daudin, 2001
- Limit Distributions of Maximal Segmental Score among Markov-Dependent Partial Sums, Karlin and Dembo 1992
- Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, Karlin and al 1990

local Score:

- Detection de courts segments inverses dans les genomes: methodes et applications, David Robelin 2005
- A Linear Time Algorithm for Finding All Maximal Scoring Subsequences, Constable and Bates 1985

Aeso *Congenital oesophageal atresia data*

Description

The data consists of individual dates of birth over n=35 cases of the birth defects oesophageal and tracheo-oesophagean fistula observed in a hospital in Birmingham, U.K., over 2191 days from 1950 through 1955, with Day one set as 1 January 1950

Usage

```
Aeso
```

Format

A matrix of 2191 lines and 2 columns. Each line is a day on the first column, and associated to a case (0/1) on the second column.

Source

Dolk H. Secular pattern of congenital oesophageal atresia–George Knox, 1959. J Epidemiol Community Health. 1997;51(2):114-115. <doi:10.1136/jech.51.2.114>

Examples

```
data(Aeso)
head(Aeso)
p <- sum(Aeso[,2]) / dim(Aeso)[1]
print(p)
```

aeso.data *Deprecated. Use 'Aeso' instead.*

Description

Deprecated. Use 'Aeso' instead.

Usage

```
aeso.data
```

Format

Deprecated. Use 'Aeso' instead.

`automatic_analysis` *Automatic analysis*

Description

Calculates local score and p-value for sequence(s) with integer scores.

Usage

```
automatic_analysis(
  sequences,
  model,
  scores,
  transition_matrix,
  distribution,
  method_limit = 2000,
  score_extremes,
  modelFunc,
  simulated_sequence_length = 1000,
  ...
)
```

Arguments

`sequences`	sequences to be analysed (named list)
`model`	the underlying model of the sequence (either "iid" for identically independently distributed variable or "markov" for Markov chains)
`scores`	vector of minimum and maximum score range
`transition_matrix`	if the sequences are markov chains, this is their transition matrix
`distribution`	vector of probabilities in ascending score order (iid sequences). Note that names of the vector must be the associated scores.
`method_limit`	limit length from which on computation-intensive exact calculation methods for p-value are replaced by approximative methods
`score_extremes`	a vector with two elements: minimal score value, maximal score value
`modelFunc`	function to create similar sequences. In this case, Monte Carlo is used to calculate p-value
`simulated_sequence_length`	if a modelFunc is provided and the sequence happens to be longer than method_limit, the method karlinMonteCarlo is used. This method requires the length of the sequences that will be created by the modelFunc for estimation of Gumble parameters.
`...`	parameters for modelFunc

Details

This method picks the adequate p-value method for your input.
If no sequences are passed to this function, it will let you pick a FASTA file.
If this is the case, and if you haven't provided any score system (as you can do by passing a named list with the appropriate scores for each character), the second file dialog which will pop up is for choosing a file containing the score (and if you provide an extra column for the probabilities, they will be used, too - see section File Formats in the vignette for details).
The function then either uses empirical distribution based on your input - or if you provided a distribution, then yours - to calculate the p-value based on the length of each of the sequences given as input.
You can influence the choice of the method by providing the modelFunc argument. In this case, the function uses exclusively simulation methods (monteCarlo, karlinMonteCarlo).
By setting the method_limit you can further decide to which extent computation-intensive methods (daudin, exact_mc) should be used to calculate the p-value. Remark that the warnings of the localScoreC() function have be deleted when called by automatic_analysis() function

Value

A list object containing

`Local score`	local score...
`p-value`	p-value ...
`Method`	the method used for the calculus of the p-value

Examples

```
# Minimal example
l = list()
seq1 = sample(-2:1, size = 3000, replace = TRUE)
seq2 = sample(-3:1, size = 150, replace = TRUE)
l[["hello"]] = seq1
l[["world"]] = seq2
automatic_analysis(l, "iid")
# Example with a given distribution
automatic_analysis(l,"iid",scores=c(-3,1),distribution=c(0.3,0.3,0.1,0.1,0.2))
# forcing the exact method for the longest sequence
aa1=automatic_analysis(l,"iid")
aa1$hello$`method applied`
aa1$hello$`p-value`
aa2=automatic_analysis(l,"iid",method_limit=3000)
aa2$hello$`method applied`
aa2$hello$`p-value`
# Markovian example
MyTransMat <-
matrix(c(0.3,0.1,0.1,0.1,0.4, 0.3,0.2,0.2,0.2,0.1, 0.3,0.4,0.1,0.1,0.1, 0.3,0.3,0.3,0.0,0.1,
        0.1,0.3,0.2,0.3,0.1), ncol = 5, byrow=TRUE)
MySeq.CM=transmatrix2sequence(matrix = MyTransMat,length=150, score =-2:2)
MySeq.CM2=transmatrix2sequence(matrix = MyTransMat,length=110, score =-2:2)
automatic_analysis(sequences = list("x1" = MySeq.CM, "x2" = MySeq.CM2), model = "markov")
```

`CharSequence2ScoreSequence`
Convert a character sequence into a score sequence

Description

Convert a character sequence into a score sequence. See CharSequences2ScoreSequences() fonction for several sequences

Usage

```
CharSequence2ScoreSequence(sequence, dictionary)
```

Arguments

`sequence`	a character sequence
`dictionary`	a dictionary

Value

a vector of a score sequence

Examples

```
data(ShortSeq)
ShortSeq
data(dico)
CharSequence2ScoreSequence(ShortSeq,dico)
```

`CharSequences2ScoreSequences`
Convert several character sequences into score sequences

Description

Convert several character sequence into score sequences. For only one sequence see CharSequence2ScoreSequence() function.

Usage

```
CharSequences2ScoreSequences(sequences, dictionary)
```

Arguments

`sequences`	a list of character sequences
`dictionary`	a dictionary

Value

a list of score sequences

Examples

```
data(ShortSeq)
ShortSeq
data(MidSeq)
MidSeq
data(dico)
MySequences=list("A1"=ShortSeq,"A2"=MidSeq)
CharSequences2ScoreSequences(MySequences,dico)
```

`daudin` *Daudin [p-value] [iid]*

Description

Calculates the exact p-value in the identically and independantly distributed of a given local score, a sequence length that 'must not be too large' and for a given score distribution

Usage

```
daudin(
  localScore,
  sequence_length,
  score_probabilities,
  sequence_min,
  sequence_max
)
```

Arguments

`localScore`	the observed local score
`sequence_length`	length of the sequence
`score_probabilities`	the probabilities for each score from lowest to greatest
`sequence_min`	minimum score
`sequence_max`	maximum score

Details

Small in this context depends heavily on your machine. On a 3,7GHZ machine this means for daudin(1000, 5000, c(0.2, 0.2, 0.2, 0.1, 0.2, 0.1), -2, 3) an execution time of ~2 seconds. This is due to the calculation method using matrix exponentation which becomes very fast very slow. The size of the matrix of the exponentiation is equal to a+1 with a the local score value. The matrix must be put at the power n, with n the sequence length. Moreover, it is known that the local score value is expected to be in mean of order log(n).

Value

A double representing the probability of a local score as high as the one given as argument

Examples

```
daudin(localScore = 4, sequence_length = 50,
score_probabilities = c(0.2, 0.3, 0.1, 0.2, 0.1, 0.1), sequence_min = -3, sequence_max = 2)
```

---

dico *Deprecated. Use 'HydroScore' instead.*

Description

Deprecated. Use 'HydroScore' instead.

Usage

```
dico
```

Format

Deprecated. Use 'HydroScore' instead.

---

exact_mc *Exact method for p-value [Markov chains]*

Description

Calculates the exact p-value for short numerical Markov chains. Memory usage and time computation can be too large for a high local score value and high score range (see details).

Usage

```
exact_mc(localScore, m, sequence_length, score_values = NULL, prob0 = NULL)
```

Arguments

`localScore`	Integer local score for which the p-value should be calculated
`m`	Transition matrix [matrix object]. Optionnaly, rownames can be corresponding score values. m should be a transition matrix of an ergodic Markov chain.
`sequence_length`	Length of the sequence
`score_values`	A integer vector of sequence score values (optional). If not set, the rownames of m are used if they are numeric and set.
`prob0`	Vector of probability distribution of the first score of the sequence (optional). If not set, the stationnary distribution of m is used.

Details

This method computation needs to allocate a square matrix of size localScore^(range(score_values)). This matrix is then exponentiated to sequence_length.

Value

A double representing the probability of a localScore as high as the one given as argument

Examples

```
mTransition <- t(matrix(c(0.2, 0.3, 0.5, 0.3, 0.4, 0.3, 0.2, 0.4, 0.4), nrow = 3))
scoreValues <- -1:1
initialProb <- stationary_distribution(mTransition)
exact_mc(localScore = 12, m = mTransition, sequence_length = 100,
        score_values = scoreValues, prob0 = initialProb)
exact_mc(localScore = 150, m = mTransition, sequence_length = 1000,
         score_values = scoreValues, prob0 = initialProb)
rownames(mTransition) <- scoreValues
exact_mc(localScore = 12, m = mTransition, sequence_length = 100, prob0 = initialProb)
# Minimal specification
exact_mc(localScore = 12, m = mTransition, sequence_length = 100)
```

`HydroScore` *Dictionnaire*

Description

Provides integer scores related to an hydrophobicity level of each amino acid. This score function is inspired by the Kyte and Doolittle (1982) scale.

Usage

```
HydroScore
```

Format

A score function for the 20 amino acid

Source

Kyte & Doolittle (1982) J. Mol. Biol. 157, 105-132

Examples

```
data(HydroScore)
HydroScore
data(Seq219)
Seq219
seqScore=CharSequence2ScoreSequence(Seq219,HydroScore)
seqScore[1:30]
localScoreC(seqScore)$localScore
```

---

karlin *Karlin [p-value] [iid]*

Description

Calculates an approximated p-value of a given local score value and a long sequence length in the identically and independantly distributed model for the sequence. See also mcc() function for another approximated method in the i.i.d. model

Usage

```
karlin(
  localScore,
  sequence_length,
  score_probabilities,
  sequence_min,
  sequence_max
)
```

Arguments

`localScore`	the observed local score
`sequence_length`	length of the sequence (at least several hundreds)
`score_probabilities`	the probabilities for each unique score from lowest to greatest
`sequence_min`	minimum score
`sequence_max`	maximum score

Details

This method works the better the longer the sequence is. Important note : the calculus of the parameter of the distribution uses the resolution of a polynome which is a function of the score distribution, of order max(score)-min(score). There exists only empirical methods to solve a polynome of order greater that 5 with no warranty of reliable solution. The found roots are checked internally to the function and an error message is throw in case of inconsistent. In such case, you could try to change your score scheme (in case of discretization) or use the function `karlinMonteCarlo` .

Value

A double representing the probability of a localScore as high as the one given as argument

Examples

```
karlin(150, 10000, c(0.08, 0.32, 0.08, 0.00, 0.08, 0.00, 0.00, 0.08, 0.02, 0.32, 0.02), -5, 5)
```

`karlinMonteCarlo` *Monte Carlo - Karlin [p-value]*

Description

Estimates p-value, for integer scores, based on a Monte Carlo estimation of Gumble parameters from simulations of smaller sequences with same distribution. Appropriate for great sequences with length > 10^3, for i.i.d and markovian sequence models.

Usage

```
karlinMonteCarlo(
  local_score,
  sequence_length,
  simulated_sequence_length,
  FUN,
  ...,
  numSim = 1000,
  plot = TRUE
)
```

Arguments

`local_score`	local score observed in a segment.
`sequence_length`	length of the sequence
`simulated_sequence_length`	length of simulated sequences produced by FUN
`FUN`	function to simulate similar sequences with.
`...`	parameters for FUN
`numSim`	number of sequences to create for estimation
`plot`	boolean value if to display plots for cumulated function and density

Details

The length of the simulated sequences is an argument specific to the function provided for simulation. Thus, it has to be provided also in the parameter simulated_sequence_length in the arguments of the "Monte Carlo - Karlin" function. It is a crucial detail as it influences precision and computation time of the result. Note that to get an appropriate estimation, the given average score must be non-positive.

Value

Floating value corresponding to the probability to obtain a local score with a value greater or equal to the parameter local_score

Examples

```
new = sample(-7:6, replace = TRUE, size = 1000)
#MonteCarlo taking random sample from the input sequence itself

karlinMonteCarlo(local_score = 66, sequence_length = 1000,
             FUN = function(x, simulated_sequence_length) {return(sample(x = x,
             size = simulated_sequence_length, replace = TRUE))},
             x=new, simulated_sequence_length = 1000,  numSim = 1000)
```

`karlinMonteCarlo_double`

Monte Carlo - Karlin for real scores[p-value]

Description

Estimates p-value, for integer scores, based on a Monte Carlo estimation of Gumble parameters from simulations of smaller sequences with same distribution. Appropriate for great sequences with length > 10^3, for i.i.d and markovian sequence models.

Usage

```
karlinMonteCarlo_double(
  local_score,
  sequence_length,
  simulated_sequence_length,
  FUN,
  ...,
  numSim = 1000,
  plot = TRUE
)
```

Arguments

`local_score`	local score observed in a segment.
`sequence_length`	length of the sequence
`simulated_sequence_length`	length of simulated sequences produced by FUN
`FUN`	function to simulate similar sequences with.
`...`	parameters for FUN

`numSim`	number of sequences to create for estimation
`plot`	boolean value if to display plots for cumulated function and density

Details

The length of the simulated sequences is an argument specific to the function provided for simulation. Thus, it has to be provided also in the parameter simulated_sequence_length in the arguments of the "Monte Carlo - Karlin" function. It is a crucial detail as it influences precision and computation time of the result. Note that to get an appropriate estimation, the given average score must be non-positive.

Value

Floating value corresponding to the probability to obtain a local score with value greater or equal the parameter

Examples

```
new = sample(-7:6, replace = TRUE, size = 1000)
#MonteCarlo taking random sample from the input sequence itself

karlinMonteCarlo_double(local_score = 66, sequence_length = 1000,
               FUN = function(x, simulated_sequence_length) {return(sample(x = x,
               size = simulated_sequence_length, replace = TRUE))},
               x=new, simulated_sequence_length = 1000,  numSim = 1000)

 #Markovian example (longer computation)
 MyTransMat_reels <- matrix(c(0.3,0.1,0.1,0.1,0.4, 0.2,0.2,0.1,0.2,0.3, 0.3,0.4,0.1,0.1,0.1,
0.3,0.3,0.1,0.2,0.1, 0.2,0.1,0.2,0.4,0.1),ncol = 5, byrow=TRUE)

karlinMonteCarlo(local_score = 10.5,sequence_length=200,FUN = transmatrix2sequence,
matrix = MyTransMat_reels, score =c(-1,-0.5,0,0.5,1),length=1500,
plot=FALSE, numSim = 1500, simulated_sequence_length =1500)
```

`lindley` *Lindley process*

Description

Creates a sequence of a Lindley process, also called CUSUM process, on a given sequence. For a sequence (X_k)k, the Lindley process is defined as follows: W_0:=0 and W_(k+1)=max(0,W_k+X_(k+1)). It defines positive excursions above 0.

Usage

```
lindley(sequence)
```

Arguments

`sequence`	numeric sequence of a Lindley process, eg service time per customer

Value

a vector with the Lindley process steps

Examples

```
seq=c(1,2,3,-4,1,-3,-1,2,3,-4,1)
lindley(seq)
plot(1:length(seq),lindley(seq),type='b')
```

`loadMatrixFromFile` *Loads matrix from csv-File*

Description

Reads a csv file without header and returns the matrix. For file formats please see section "File Formats" in vignette.

Usage

```
loadMatrixFromFile(filepath)
```

Arguments

`filepath`	optional: Location of file on disk. If not provided, a file picker dialog will be opened.

Value

A Matrix Object

`loadScoreFromFile` *Load score from file*

Description

Reads a csv file with 2-3 columns and returns it as a list object of vectors, with names corresponding to the first column of the file. For details view section "File Formats" in vignette.

Usage

```
loadScoreFromFile(filepath, ...)
```

Arguments

`filepath`	optional: location of file on disk. If not provided, a file picker dialog will be opened.
`...`	optional: use arguments from read.csv

Value

A List Object - Names correspond to the first column, usually Letters. Associated numerical scores are in the second column. If probabilities are provided, they will be loaded too and presumed to be in the third column

---

`localScoreC` *Local score*

Description

Calculates the local score for a sequence of integer scores. Only provides the first occurrence of the local score. Use function suboptimalSegment() or Lindley() to obtain the others localizations of the different realizations of the local score.

Usage

```
localScoreC(v, supressWarnings = FALSE)
```

Arguments

`v`	: a sequence of integer values as vector.
`supressWarnings`	: if warnings should not be displayed

Value

A structure containing: the local score value and the begin and end index of the segment realizing this optimal score ; all the local maxima of the Lindley process (non negative excursion) and their begin and ens index ; the record times of the Lindley process but only the ones corresponding to the begin index of non negative excursions

Examples

```
seq.OneSegment=c(1,-2,3,1,-1,2)
# one segment realizing the local score value
localScoreC(seq.OneSegment)
seq.TwoSegments=c(1,-2,3,1,2,-2,-2,-1,1,-2,3,1,2,-1,-2,-2,-1,1)
# two segments realizing the local score value
localScoreC(seq.TwoSegments)
# only the first realization
localScoreC(seq.TwoSegments)$localScore
# all the realization of the local together with the suboptimal ones
```

```
localScoreC(seq.TwoSegments)$suboptimalSegmentScores
# for small sequences, you can also use lindley() fonction to check if
# several segments achieve the local Score
lindley(seq.TwoSegments)
plot(1:length(seq.TwoSegments),lindley(seq.TwoSegments),type='b')
seq.TwoSegments.InSameExcursion=c(1,-2,3,2,-1,0,1,-2,-2,-4,1)
localScoreC(seq.TwoSegments.InSameExcursion)
# lindley() shows two realizations in the same excursion (no 0 value between the two LS values)
lindley(seq.TwoSegments.InSameExcursion)
# same beginning index but two possible ending indexes
# only one excursion realizes the local score even in there is two possible length of segment
localScoreC(seq.TwoSegments.InSameExcursion)$suboptimalSegmentScores
plot(1:length(seq.TwoSegments.InSameExcursion),lindley(seq.TwoSegments.InSameExcursion),type='b')
```

`localScoreC_double` *Local score for sequences of floating values*

Description

Calculates the local score for a sequence of doubles. Only provides the first occurrence. Use function suboptimalSegment() or Lindley() to obtain the others localizations of the different realizations of the local score.

Usage

```
localScoreC_double(v, supressWarnings = FALSE)
```

Arguments

`v`	A sequence of values as vector.
`supressWarnings`	if warnings should be displayed

Value

A structure containing: the local score value and the begin and end index of the segment realizing this optimal score ; all the local maxima of the Lindley process (non negative excursion) and their begin and ens index ; the record times of the Lindley process but only the ones corresponding to the begin index of non negative excursions

Examples

```
localScoreC_double(c(1.2,-2.1,3.5,1.7,-1.1,2.3))
seq.TwoSegments=c(1.2,-2.1,3.5,1.7,2,-2,-2,-3.5,1,3.5,1.7,1,-2,-2)
# two segments realizing the local score value
localScoreC(seq.TwoSegments)
# only the first realization
localScoreC(seq.TwoSegments)$localScore
# all the realization of the local together with the suboptimal ones
```

```
localScoreC(seq.TwoSegments)$suboptimalSegmentScores
# for small sequences, you can also use lindley() fonction to check if
# several segments achieve the local score
lindley(seq.TwoSegments)
plot(1:length(seq.TwoSegments),lindley(seq.TwoSegments),type='b')
seq.TwoSegments.InSameExcursion=c(1,-2,3,2,-1,0,1,-2,-2)
localScoreC(seq.TwoSegments.InSameExcursion)
# lindley() shows two realizations in the same excursion (no 0 value between the two LS values)
lindley(seq.TwoSegments.InSameExcursion)
plot(1:length(seq.TwoSegments.InSameExcursion),lindley(seq.TwoSegments.InSameExcursion),type='b')
# same beginning index but two possible ending indexes
# only one excursion realizes the local score even in there is two possible length of segment
localScoreC(seq.TwoSegments.InSameExcursion)$suboptimalSegmentScores
```

LongSeq *Deprecated. Use 'Seq1093' instead.*

Description

Deprecated. Use 'Seq1093' instead.

Usage

```
LongSeq
```

Format

Deprecated. Use 'Seq1093' instead.

---

maxPartialSumd *Maximum of the partial sum [probability] [iid]*

Description

Calculates the distribution of the maximum of the partial sum process for a given value in the identically and independantly distributed model

Usage

```
maxPartialSumd(k, score_probabilities, sequence_min, sequence_max)
```

Arguments

`k`	value at which calculates the probability
`score_probabilities`	the probabilities for each unique score from lowest to greatest
`sequence_min`	minimum score
`sequence_max`	maximum score

Details

Implement the formula (4) of the article Mercier, S., Cellier, D., & Charlot, D. (2003). An improved approximation for assessing the statistical significance of molecular sequence features. Journal of Applied Probability, 40(2), 427-441. doi:10.1239/jap/1053003554
Important note : the calculus of the parameter of the distribution uses the resolution of a polynome which is a function of the score distribution, of order max(score)-min(score). There exists only empirical methods to solve a polynome of order greater that 5 with no warranty of reliable solution. The found roots are checked internally to the function and an error message is throw in case of inconsistency.

Value

A double representing the probability of the maximum of the partial sum process equal to k

Examples

```
maxPartialSumd(10, c(0.08, 0.32, 0.08, 0.00, 0.08, 0.00, 0.00, 0.08, 0.02, 0.32, 0.02), -6, 4)
```

---

mcc *MCC [p-value] [iid]*

Description

Calculates an approximated p-value for a given local score value and a medium to long sequence length in the identically and independantly distributed model

Usage

```
mcc(
  localScore,
  sequence_length,
  score_probabilities,
  sequence_min,
  sequence_max
)
```

Arguments

`localScore`	the observed local score
`sequence_length`	length of the sequence (up to one hundred)
`score_probabilities`	the probabilities for each unique score from lowest to greatest
`sequence_min`	minimum score
`sequence_max`	maximum score

Details

This methods is actually an improved method of Karlin and produces more precise results. It should be privileged whenever possible.
As with karlin, the method works the better the longer the sequence. Important note : the calculus of the parameter of the distribution uses the resolution of a polynome which is a function of the score distribution, of order max(score)-min(score). There exists only empirical methods to solve a polynome of order greater that 5 with no warranty of reliable solution. The found roots are checked internally to the function and an error message is throw in case of inconsistency. In such case, you could try to change your score scheme (in case of discretization) or use the function `karlinMonteCarlo`.

Value

A double representing the probability of a local score as high as the one given as argument

Examples

```
mcc(40, 100, c(0.08, 0.32, 0.08, 0.00, 0.08, 0.00, 0.00, 0.08, 0.02, 0.32, 0.02), -6, 4)
mcc(40, 10000, c(0.08, 0.32, 0.08, 0.00, 0.08, 0.00, 0.00, 0.08, 0.02, 0.32, 0.02), -6, 4)
```

`MidSeq` *Deprecated. Use 'Seq219' instead.*

Description

Deprecated. Use 'Seq219' instead.

Usage

```
MidSeq
```

Format

Deprecated. Use 'Seq219' instead.

`monteCarlo` *Monte Carlo method [p-value]*

Description

Calculates an empirical p-value based on simulations of similar integer sequences of the same length. Perfect for small sequences (both markov chains and identically and independantly distributed) with length ~ 10^3. See function monteCarlo_double() for possible real scores.

Usage

```
monteCarlo(local_score, FUN, ..., plot = TRUE, numSim = 1000)
```

Arguments

`local_score`	local score observed in a segment.
`FUN`	function to simulate similar sequences with.
`...`	parameters for FUN
`plot`	boolean value if to display plots for cumulated function and density
`numSim`	number of sequences to generate during simulation

Value

Floating value corresponding to the probability to obtain a local score with value greater or equal to the parameter

Examples

```
monteCarlo(120, FUN = rbinom, n = 100, size = 5, prob=0.2)

new = sample(-7:6, replace = TRUE, size = 1000)
#MonteCarlo taking random sample from the input sequence itself

monteCarlo(local_score = 20, FUN = function(x) {return(sample(x = x,
size = length(x), replace = TRUE))}, x=new)

# Markovian example
MyTransMat <-
+   matrix(c(0.3,0.1,0.1,0.1,0.4, 0.2,0.2,0.1,0.2,0.3, 0.3,0.4,0.1,0.1,0.1, 0.3,0.3,0.1,0.0,0.3,
+              0.1,0.1,0.2,0.3,0.3), ncol = 5, byrow=TRUE)

monteCarlo(local_score = 50,
          FUN = transmatrix2sequence, matrix = MyTransMat,
          length=150, score = c(-2,-1,0,2,3), plot=FALSE, numSim = 5000)
```

`monteCarlo_double` *Monte Carlo method for real score case [p-value]*

Description

Calculates an empirical p-value based on simulations of similar sequences of the same length. Perfect for small sequences (both markov chains and identically and independantly distributed) with length ~ 10^3. Function dedicated for real score case.

Usage

```
monteCarlo_double(local_score, FUN, ..., plot = TRUE, numSim = 1000)
```

Arguments

`local_score`	local score observed in a segment.
`FUN`	function to simulate similar sequences with.
`...`	parameters for FUN
`plot`	Boolean value if to display plots for cumulated function and density
`numSim`	number of sequences to generate during simulation

Value

Floating value corresponding to the probability to obtain a local score with value greater or equal to the parameter

Examples

```
score_reels=c(-1,-0.5,0,0.5,1)
proba_score_reels=c(0.2,0.3,0.1,0.2,0.2)
sample_from_model <- function(score.sple,proba.sple, length.sple){sample(score.sple,
                                size=length.sple, prob=proba.sple, replace=TRUE)}

monteCarlo_double(5.5,FUN=sample_from_model, plot = TRUE,
score.sple=score_reels,proba.sple=proba_score_reels, length.sple=100, numSim = 1000)
```

`MySeqList` *Deprecated. Use 'SeqListSCOPe' instead.*

Description

Deprecated. Use 'SeqListSCOPe' instead.

Usage

```
MySeqList
```

Format

Deprecated. Use 'SeqListSCOPe' instead.

`RealScores2IntegerScores` *Convert a real scores vector into an integer scores vector*

Description

Convert real scores into integer scores

Usage

```
RealScores2IntegerScores(RealScore, ProbRealScore, coef = 10)
```

Arguments

`RealScore`	vector of real scores
`ProbRealScore`	vector of probability
`coef`	coefficient

Details

Convert real scores into integer scores by multiplying real scores by a coefficient (default 10) and then assigning probability to corresponding extended (from the minimum to the maximum) integer scores

Value

list containing ExtendedIntegerScore and ProbExtendedIntegerScore

Examples

```
score <- c(-1,-0.5,0,0.5,1)
prob.score <- c(0.2,0,0.4,0.1,0.3)
(res1 <- RealScores2IntegerScores(score, prob.score, coef=10))
prob.score.err <- c(0.1,0,0.4,0.1,0.3)
(res2 <- RealScores2IntegerScores(score, prob.score.err, coef=10))
# When coef=1, the function can handle integer scores
ex.integer.score <- c(-3,-1,0,1, 5)
(res3 <- RealScores2IntegerScores(ex.integer.score, prob.score, coef=1))
```

scoreDictionnary2probabilityVector

Check for missing scores values in the score distribution

Description

Get extremes scores, then create a complete list and set a probability equal to zeros for not present scores

Usage

```
scoreDictionnary2probabilityVector(list, score_extremes)
```

Arguments

`list`	vector of scores
`score_extremes`	vector of probability

Value

vector containing all score values between extremes and the probability equal to 0 for missing score

Examples

```
Mylist=list("x1"=c(-2,0.1),"x2"=c(0,0.7),"x3"=c(1,0.2))
scoreDictionnary2probabilityVector(list=Mylist,score_extremes=c(-2,1))
```

scoreSequences2probabilityVector

Empirical distribution from sequences

Description

Builds empirical distribution from a list of numerical sequences

Usage

```
scoreSequences2probabilityVector(sequences)
```

Arguments

`sequences`	list of numerical sequences

Details

By determining the extreme scores in the sequences, this function creates a vector of probabilities including values that do not occur at all. In this it differs from table(). For example, two sequences containing values from 1:2 and 5:6 will produce a vector of size 6.

Value

empirical distribution from minimum score to maximum score as a vector of floating numbers.

Examples

```
seq1 = sample(7:8, size = 10, replace = TRUE)
seq2 = sample(2:3, size = 15, replace = TRUE)
l = list(seq1, seq2)
scoreSequences2probabilityVector(l)
```

Seq1093 *Long protein sequence*

Description

A long protein sequence.

Usage

```
Seq1093
```

Format

A character string with 1093 characters corresponding to Q60519.fasta in UniProt Data base.

Source

https://www.uniprot.org/

Examples

```
data(Seq1093)
Seq1093
nchar(Seq1093)
data(HydroScore)
seqScore=CharSequence2ScoreSequence(Seq1093,HydroScore)
seqScore[1:50]
localScoreC(seqScore)$localScore
LS=localScoreC(seqScore)$localScore[1]
prob1 = scoreSequences2probabilityVector(list(seqScore))
daudin(localScore = LS, sequence_length = nchar(Seq1093),
               score_probabilities = prob1,
               sequence_min = min(seqScore),
               sequence_max = max(seqScore))
karlin(localScore = LS, sequence_length = nchar(Seq1093),
score_probabilities = prob1,
sequence_min = min(seqScore),
sequence_max = max(seqScore))
```

Seq219 *Protein sequence*

Description

A protein sequence.

Usage

```
Seq219
```

Format

A character string with 219 characters corresponding to P49755.fasta query in UniProt Data base.

Source

https://www.uniprot.org/

Examples

```
data(Seq219)
Seq219
nchar(Seq219)
data(HydroScore)
seqScore=CharSequence2ScoreSequence(Seq219,HydroScore)
seqScore[1:30]
localScoreC(seqScore)$localScore
prob1 = scoreSequences2probabilityVector(list(seqScore))
daudin(localScore = 52, sequence_length = nchar(Seq219),
               score_probabilities = prob1,
               sequence_min = min(seqScore),
               sequence_max = max(seqScore))
score=-5:5
prob2=c(0.15,0.15,0.1,0.1,0.0,0.05,0.15,0.05,0.2,0.0,0.05)
daudin(localScore = 52, sequence_length = nchar(Seq219),
       score_probabilities = prob2,
       sequence_min = min(seqScore),
       sequence_max = max(seqScore))
```

Seq31 *Short protein sequence*

Description

A short protein sequence of 31 amino acids corresponding to Q09FU3.fasta query in UniProt Data base.

Usage

```
Seq31
```

Format

A character string with 31 characters "MLTITSYFGFLLAALTITSVLFIGLNKIRLI"

Source

https://www.uniprot.org/

Examples

```
data(Seq31)
Seq31
nchar(Seq31)
data(HydroScore)
SeqScore=CharSequence2ScoreSequence(Seq31,HydroScore)
SeqScore
localScoreC(SeqScore)$localScore
LS=localScoreC(SeqScore)$localScore[1]
prob1 = scoreSequences2probabilityVector(list(SeqScore))
daudin(localScore = LS, sequence_length = nchar(Seq31),
               score_probabilities = prob1,
               sequence_min = min(SeqScore),
               sequence_max = max(SeqScore))
score=-5:5
prob2=c(0.15,0.15,0.1,0.1,0.0,0.05,0.15,0.05,0.2,0.0,0.05)
sum(prob2*score)
karlin(localScore = LS, sequence_length = nchar(Seq31),
score_probabilities = prob2,
sequence_min = min(SeqScore),
sequence_max = max(SeqScore))
```

SeqListSCOPe *Several sequences*

Description

A vector of character strings

Usage

```
SeqListSCOPe
```

Format

A list of 285 character strings with their entry codes as names

Source

Structural Classification Of Proteins database (SCOP). More precisely this data contain the 285 protein sequences of the data called "CF_scop2dom_20140205aa" with length from 31 to 404.

Examples

```
data(SeqListSCOPe)
head(SeqListSCOPe)
SeqListSCOPe[1]
nchar(SeqListSCOPe[1])
summary(sapply(SeqListSCOPe, nchar))
data(HydroScore)
MySeqScoreList=lapply(SeqListSCOPe, FUN=CharSequence2ScoreSequence, HydroScore)
head(MySeqScoreList)
AA=automatic_analysis(sequences=MySeqScoreList, model='iid')
AA[[1]]
# the p-value of the first 10 sequences
sapply(AA, function(x){x$`p-value`})[1:10]
# the 20th smallest p-values
sort(sapply(AA, function(x){x$`p-value`}))[1:20]
which(sapply(AA, function(x){x$`p-value`})<0.05)
table(sapply(AA, function(x){x$`method`}))
# The maximum sequence length equals 404 so it here normal that the exact method is used for
# all the 606 sequences of the data base
# Score distribution learnt on the data set
scoreSequences2probabilityVector(MySeqScoreList)
```

`sequences2transmatrix` *Transition matrix from sequence(s)*

Description

Calculates the transition matrix by counting occurences of tuples in given vector list

Usage

```
sequences2transmatrix(sequences)
```

Arguments

`sequences`	Sequences to be analyzed, as list of numeric vectors

Details

The transition matrix will be structured so that the lowest score corresponds to the first column and row and the highest score corresponds to the last column and row. Note that the resulting matrix is not stochastic because it can occur rows filled up with only 0 for not observed score in Min Value Max Value interval.

Value

A list object containing

`Transition Matrix`	Transition Matrix
`Min Value`	minimal score found in supplied sequences
`Max Value`	maximal score found in supplied sequences

Examples

```
seq = sample(-1:1, size = 20, replace = TRUE)
seq2 = sample(-6:1, size = 20, replace = TRUE)
seq3 = sample(3:6, size = 50, replace = TRUE)
sequences2transmatrix(list(seq, seq2, seq3))
```

`ShortSeq` *Deprecated. Use 'Seq31' instead.*

Description

Deprecated. Use 'Seq31' instead.

Usage

```
ShortSeq
```

Format

Deprecated. Use 'Seq31' instead.

`SJSyndrome.data` *Stevens-Johnson syndrome data*

Description

The Stevens-Johnson syndrome is an acute and serious dermatological disease due to a drug allergy. The syndrome appearance is life-threatening emergency. They are very rare, around 2 cases per million people per year.

Usage

```
SJSyndrome.data
```

Format

A data.frame of 824 lines, each describing a syndrome appearance described by 15 covariates: Case ID, Initial FDA Received Date, days since last fda, Event Date, Latest FDA Received Date, Suspect Product Names, Suspect Product Active Ingredients, Reason for Use, Reactions, Serious, Outcomes, Sex, Patient Age, Sender, Concomitant Product Names The third column correspond to the number of days between two adverse events.

Source

FDA open data.

Examples

```
data(SJSyndrome.data)
summary(SJSyndrome.data)
```

`stationary_distribution` *Stationary distribution [Markov chains]*

Description

Calculates stationary distribution of markov transition matrix by use of eigenvectors of length 1

Usage

```
stationary_distribution(m)
```

Arguments

`m`	Transition Matrix [matrix object]

Value

A vector with the probabilities

Examples

```
B = t(matrix (c(0.2, 0.8, 0.4, 0.6), nrow = 2))
stationary_distribution(B)
```

`transmatrix2sequence` *Sampling function for Markov chains*

Description

Creates Markov chains based on a transition matrix. Can be used as parameter for the Monte Carlo function.

Usage

```
transmatrix2sequence(matrix, length, initialIndex, score)
```

Arguments

`matrix`	transition matrix of Markov process
`length`	length of sequence to be sampled
`initialIndex`	(optional) index of matrix which should be initial value of sequence. If none supplied, a value from the stationary distribution is sampled as initial value.
`score`	(optional) a vector representing the scores (in ascending order) of the matrix index. If supplied, the result will be a vector of these values.

Details

The transition matrix is considered representing the transition from one score to another such that the score in the first row is the lowest and the last row are the transitions from the highest score to another. The matrix must be stochastic (no rows filled up with only '0' values).

Value

a Markov chain sampled from the transition matrix

Examples

```
B = t(matrix (c(0.2, 0.8, 0.4, 0.6), nrow = 2))
transmatrix2sequence(B, length = 10)
MyTransMat <-
  matrix(c(0.3,0.1,0.1,0.1,0.4, 0.2,0.2,0.1,0.2,0.3, 0.3,0.4,0.1,0.1,0.1, 0.3,0.3,0.1,0.0,0.3,
    0.1,0.1,0.2,0.3,0.3), ncol = 5, byrow=TRUE)
MySeq.CM=transmatrix2sequence(matrix = MyTransMat,length=90, score =c(-2,-1,0,2,3))
MySeq.CM
```

Index