Package 'gphmm'

October 2, 2017

Version 0.99.0

Title Generalized Pair Hidden Markov Chain Model for Sequence Alignment

Description Implementation of a generalized pair hidden Markov chain model (GPHMM) that can be used to compute the probability of alignment between two sequences of nucleotides (e.g., a reference sequence and a noisy sequenced read). The model can be trained on a dataset where the noisy sequenced reads are known to have been sequenced from known reference sequences. If no training sets are available default parameters can be used.

License Artistic-2.0

LazyData TRUE

Imports Rcpp, parallel, dplyr, Biostrings, stringr, stringi, stats, docopt

LinkingTo Rcpp

Suggests testthat, knitr, jsonlite

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Author Fanny Perraudeau [aut, cre], James Bullard [aut]

Maintainer Fanny Perraudeau <perraudeau.f@gmail.com>

Repository CRAN

Date/Publication 2017-10-02 11:30:09 UTC

R topics documented:

calculategphmm 2
computeCounts 3
computegphmm 3
computeGphmmParam 4
generateRandomSequences 4

generateRead 5
initializeGphmm 6
makeGphmmPerRead 6

Index **7**

`calculategphmm` *Calculate GPHMM probability.*

Description

This function returns the GPHMM probability that a read x could have been sequenced from a reference sequence y.

Usage

```
calculategphmm(x, y, tau, pp, qX, qY, dX, dY, eX, eY)
```

Arguments

x	string, with the sequence of the read.
y	string, with the sequence of the reference.
tau	double, probability to transition from any state to end state.
pp	matrix, emission probabilities in the M state.
qX	vector, emission probabilities in the insertion state.
qY	vector, emission probabilities in the deletion state.
dX	double, transition probability from the M to the insertion state.
dY	double, transition probability from the M to the deletion state.
eX	double, transition probability from the insertion to the insertion state.
eY	double, transition probability from the deletion to the deletion state.

Examples

```
param <- initializeGphmm()
tau <- param[['tau']]
pp <- param[['pp']]
qX <- param[['qX']]
qY <- param[['qY']]
dX <- 1/(1+exp(-sum(param[['deltaX']] * c(1, 20))))
dY <- 1/(1+exp(-sum(param[['deltaY']] * c(1, 20))))
eX <- param[['epsX']]
eY <- param[['epsY']]
calculategphmm('ATCG', 'ATGG', tau, pp, qX, qY, dX, dY, eX, eY)
```

computeCounts *Compute counts for emissions and transitions.*

Description

computeCounts returns a list with emission and transition counts.

Usage

```
computeCounts(gphmm)
```

Arguments

gphmm	- output of computegphmm with arg output='long'.

Examples

```
gphmmOut <- computegphmm('ATCG', 'ATG', output = 'long')
computeCounts(gphmmOut)
```

computegphmm *Compute GPHMM log probability.*

Description

computegphmm returns GPHMM log probability for a read and a reference sequence.

Usage

```
computegphmm(read, ref, parameters = initializeGphmm(), qv = 20,
  output = "long")
```

Arguments

read	- chr str.
ref	- chr str.
parameters	- list of GPHMM parameters, can be created by initializeGphmm().
qv	- float, quality value, default is 20.
output	- if 'long', output is a list with path, read, ref and log GPHMM proba, else output is just the log GPHMM proba.

Examples

```
computegphmm('TAGC', 'AAG')
computegphmm('TAGC', 'AAG', qv = 30)
```

computeGphmmParam *Compute gphmm parameters from counts.*

Description

computeGphmmParam returns a list with gphmm parameters.

Usage

```
computeGphmmParam(emiTrans)
```

Arguments

emiTrans	- list with emission and transition counts.

Examples

```
library(Biostrings)
seqs <- DNAStringSet(c(a='ATGC', b = 'ATGG', c = 'ATGT'))
csv <- data.frame(queries = c('a', 'b'), refs = c('c', 'c'))
gphmmPerRead <- makeGphmmPerRead(seqs, csv)
parameters <- initializeGphmm()
counts <- lapply(1:nrow(csv), function(i) gphmmPerRead(i, parameters))
computeGphmmParam(counts)
```

generateRandomSequences *Split randomly any R object for which method length() has been defined into a train set and a test set.*

Description

generateRandomSequences returns a list with indices for train and test sets.

Usage

```
generateRandomSequences(n = 2, meanLen = 10, sdLen = 0, seed = NULL,
  ncores = NULL, prob = rep(0.25, 4))
```

Arguments

n	- int, number of sequences to generate.
meanLen	- float, mean of the length distribution (gaussian) of the generated sequences.
sdLen	- float, sd of the length distribution (gaussian) of the generated sequences.
seed	- int, when the same seed and parameters are used, exact same sequences are generated.
ncores	- int, number of cores to use.
prob	- vector of length 4 with the probability for the 4 nucleotides (A, C, G, T).

Examples

```
generateRandomSequences(n = 2, meanLen = 20, sdLen = 2)
generateRandomSequences(n = 4, meanLen = 5, sdLen = 0)
```

---

generateRead *Generate a noisy read from a true sequence.*

Description

generateRead returns a list with the noisy read, the true sequence, the path, and the phred quality score.

Usage

```
generateRead(seq = "ATGCGGATCG", qv = NULL, seed = NULL,
  paramgphmm = initializeGphmm())
```

Arguments

seq	- character vector of true sequence.
qv	- integer, wanted phred quality score for the read.
seed	- integer, seed for reproducibility.
paramgphmm	- list of parameters.

Examples

```
generateRead('ACGTGCA')
generateRead('ACGTGCA', qv = 10, seed = 1416)
```

initializeGphmm *Initial parameters for gphmm.*

Description

initializeGphmm returns initial set of parameters for gphmm.

Usage

```
initializeGphmm()
```

Examples

```
param <- initializeGphmm ()
param[['qR']] <- c(0.1, 0.3, 0.3, 0.1)
```

makeGphmmPerRead *Create a function to compute gphmm probabilities during the training.*

Description

makeGphmmPerRead returns a function to compute gphmm probabilities for each row of the csv file.

Usage

```
makeGphmmPerRead(seqs, csv)
```

Arguments

seqs	- DNAStringSet with DNA sequences used for the training.
csv	- data.frame with first column = queries, second column = reference sequences, third column = qv

Examples

```
library(Biostrings)
seqs <- DNAStringSet(c(a='ATGC', b = 'ATGG', c = 'ATGT'))
csv <- data.frame(queries = c('a', 'b'), refs = c('c', 'c'))
makeGphmmPerRead(seqs, csv)
```

Index

calculategphmm, 2
computeCounts, 3
computegphmm, 3
computeGphmmParam, 4

generateRandomSequences, 4
generateRead, 5

initializeGphmm, 6

makeGphmmPerRead, 6