Package ‘SparseSignatures’

May 4, 2024

Version 2.14.0
Date 2024-03-21
Title SparseSignatures
Depends R (>= 4.1.0), NMF
Imports nnlasso, nnls, parallel, data.table, Biostrings,
GenomicRanges, IRanges, BSgenome, GenomeInfoDb, ggplot2,
gridExtra, reshape2
Suggests BiocGenerics, BSgenome.Hsapiens.1000genomes.hs37d5,
BiocStyle, testthat, knitr,
Name An R package for the extraction of sparse mutational signatures
from whole genome sequencing data
Description Point mutations occurring in a genome can be divided into 96 categories based on the base being mutated, the base it is mutated into and its two flanking bases. Therefore, for any patient, it is possible to represent all the point mutations occurring in that patient's tumor as a vector of length 96, where each element represents the count of mutations for a given category in the patient. A mutational signature represents the pattern of mutations produced by a mutagen or mutagenic process inside the cell. Each signature can also be represented by a vector of length 96, where each element represents the probability that this particular mutagenic process generates a mutation of the 96 above mentioned categories. In this R package, we provide a set of functions to extract and visualize the mutational signatures that best explain the mutation counts of a large number of patients.

Encoding UTF-8
License file LICENSE
URL https://github.com/danro9685/SparseSignatures
BugReports https://github.com/danro9685/SparseSignatures
biocViews BiomedicalInformatics, SomaticMutation
RoxygenNote 7.3.1
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/SparseSignatures
Contents

background ............................................................... 3
background2 .............................................................. 3
cv_example ............................................................... 4
import.trinucleotides.counts ........................................ 4
imported_data ........................................................... 5
lambdaRangeAlphaEvaluation .......................................... 5
lambdaRangeBetaEvaluation ........................................... 7
lambda_range_example .................................................. 9
mutation_categories .................................................... 9
nmfLasso ................................................................. 10
nmfLassoBootstrap ..................................................... 11
nmfLassoCV .............................................................. 13
nmf_LassoK_example ................................................... 15
patients ................................................................. 16
patients.plot ............................................................ 17
sigAssignmentCV ........................................................ 17
sigAssignmentEvaluation .............................................. 19
sigAssignmentLasso .................................................... 21
signatures.plot .......................................................... 22
ssm560_reduced .......................................................... 22
startingBetaEstimation ............................................... 23
starting_betas_example ................................................. 24

Index ........................................................................... 25
**background**

---

**Description**

germline replication error estimated in Rahbari, Raheleh, et al. (2016).

**Usage**

data(background)

**Format**

vector of rates

**Value**

vector of rates for the 96 trinucleotides

**Source**


---

**background2**

**COSMIC replication error**

---

**Description**

background replication error signature derived from COSMIC SBS5.

**Usage**

data(background2)

**Format**

vector of rates

**Value**

vector of rates for the 96 trinucleotides

**Source**

cv_example

example of results obtained with the function nmf.LassoCV on the counts input from Nik-Zainal, Serena, et al. (2016).

Description

example of results obtained with the function nmf.LassoCV on the counts input from Nik-Zainal, Serena, et al. (2016).

Usage

data(cv_example)

Format

results obtained with the function nmf.LassoCV on the counts input from Nik-Zainal, Serena, et al. (2016)

Value

results obtained with the function nmf.LassoCV on the counts input from Nik-Zainal, Serena, et al. (2016)

import.trinucleotides.counts

Description

Make trinucleotides counts matrix from input data for a given reference genome.

Usage

import.trinucleotides.counts(data, reference = NULL)

Arguments

data a data.frame with variants having 6 columns: sample name, chromosome, start position, end position, ref, alt.

reference a BSgenome object with the reference genome to be used to retrieve flanking bases.

Value

A matrix with trinucleotides counts per patient.
**Examples**

```r
data(ssm560_reduced)
library("BSgenome.Hsapiens.1000genomes.hs37d5")
trinucleotides_counts = import.trinucleotides.counts(data=ssm560_reduced,
                                                  reference=BSgenome.Hsapiens.1000genomes.hs37d5)
```

**Description**

example of imported data from Nik-Zainal, Serena, et al. (2016).

**Usage**

```r
data(imported_data)
```

**Format**

results obtained with the function import.counts.data on the data from Nik-Zainal, Serena, et al. (2016)

**Value**

results obtained with the function import.counts.data on the data from Nik-Zainal, Serena, et al. (2016)

**lambdaRangeAlphaEvaluation**

**Description**

Estimate the range of lambda values for alpha to be considered in the signature inference. Note that too small values of lambda result in dense exposures, but too large values lead to bad fit of the counts.
Usage

```r
lambdaRangeAlphaEvaluation(
  x, 
  K = 5, 
  beta = NULL, 
  background_signature = NULL, 
  normalize_counts = TRUE, 
  nmf_runs = 10, 
  lambda_values = c(0.01, 0.05, 0.1, 0.2), 
  iterations = 30, 
  max_iterations_lasso = 10000, 
  num_processes = Inf, 
  seed = NULL, 
  verbose = TRUE, 
  log_file = ""
)
```

Arguments

- **x**: count matrix for a set of n patients and 96 trinucleotides.
- **K**: numeric value (minimum 2) indicating the number of signatures to be discovered.
- **beta**: starting beta for the estimation. If it is NULL, starting beta is estimated by NMF.
- **background_signature**: background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
- **normalize_counts**: if true, the input count matrix x is normalize such that the patients have the same number of mutation.
- **nmf_runs**: number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
- **lambda_values**: value of LASSO to be used for alpha between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_values is, the sparser are the resulting exposures, but too large values may result in a reduced fit of the observed counts.
- **iterations**: Number of iterations to be performed. Each iteration corresponds to a first step where beta is fitted and a second step where alpha is fitted.
- **max_iterations_lasso**: Number of maximum iterations to be performed during the sparsification via Lasso.
- **num_processes**: Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.
- **seed**: Seed for reproducibility.
- **verbose**: boolean; Shall I print all messages?
log_file log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

Value
A list corresponding to results of the function nmflasso for each value of lambda to be tested. This function allows to test a good range of lambda values for alpha to be considered. One should keep in mind that too small values generate dense solution, while too high ones leads to poor fit. This behavior is resampled in the values of loglik_progression, which should be increasing: too small values of lambda results in unstable log-likelihood through the iterations, while too large values make log-likelihood drop.

Examples
```r
data(background)
data(patients)
res = lambdaRangeAlphaEvaluation(x=patients[1:100,],
    K=5,background_signature=background,
    nmf_runs=1,lambda_values=c(0.01,0.05),
    num_processes=NA,
    seed=12345)
```

Description
Estimate the range of lambda values for beta to be considered in the signature inference. Note that too small values of lambda result in dense signatures, but too large values lead to bad fit of the counts.

Usage
```r
lambdaRangeBetaEvaluation(
    x,
    K = 5,
    beta = NULL,
    background_signature = NULL,
    normalize_counts = TRUE,
    nmf_runs = 10,
    lambda_values = c(0.01, 0.05, 0.1, 0.2),
    iterations = 30,
    max_iterations_lasso = 10000,
    num_processes = Inf,
    seed = NULL,
    verbose = TRUE,
)```

Arguments

- **x**: count matrix for a set of n patients and 96 trinucleotides.
- **K**: numeric value (minimum 2) indicating the number of signatures to be discovered.
- **beta**: starting beta for the estimation. If it is NULL, starting beta is estimated by NMF.
- **background_signature**: background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
- **normalize_counts**: if true, the input count matrix x is normalize such that the patients have the same number of mutation.
- **nmf_runs**: number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
- **lambda_values**: value of LASSO to be used for beta between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_values is, the sparser are the resulting signatures, but too large values may result in a reduced fit of the observed counts.
- **iterations**: Number of iterations to be performed. Each iteration corresponds to a first step where beta is fitted and a second step where alpha is fitted.
- **max_iterations_lasso**: Number of maximum iterations to be performed during the sparsification via Lasso.
- **num_processes**: Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.
- **seed**: Seed for reproducibility.
- **verbose**: boolean; Shall I print all messages?
- **log_file**: log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

Value

A list corresponding to results of the function nmfLasso for each value of lambda to be tested. This function allows to test a good range of lambda values for beta to be considered. One should keep in mind that too small values generate dense solution, while too high ones leads to poor fit. This behavior is resampled in the values of loglik_progression, which should be increasing: too small values of lambda results in unstable log-likelihood through the iterations, while too large values make log-likelihood drop.
Examples

```r
data(background)
data(patients)
res = lambdaRangeBetaEvaluation(
x=patients[1:100,],
    K=5,
    background_signature=background,
    nmf_runs=1,
    lambda_values=c(0.01,0.05),
    num_processes=NA,
    seed=12345)
```

lambda_range_example  example of results obtained with the function evaluate.lambda.range on the counts input from Nik-Zainal, Serena, et al. (2016).

Description

example of results obtained with the function evaluate.lambda.range on the counts input from Nik-Zainal, Serena, et al. (2016).

Usage

```r
data(lambda_range_example)
```

Format

results obtained with the function evaluate.lambda.range on the counts input from Nik-Zainal, Serena, et al. (2016)

Value

results obtained with the function evaluate.lambda.range on the counts input from Nik-Zainal, Serena, et al. (2016)

mutation_categories  trinucleotides mutation categories

Description

96 trinucleotides mutation categories

Usage

```r
data(mutation_categories)
```
Format

matrix of 96 trinucleotides mutation categories

Value

matrix of 96 trinucleotides mutation categories

Description

Perform the discovery of $K$ somatic mutational signatures given a set of observed counts $x$.

Usage

```r
nmfLasso(
  x,  # count matrix for a set of n patients and 96 trinucleotides.
  K,  # numeric value (minimum 2) indicating the number of signatures to be discovered.
  beta = NULL,  # starting beta for the estimation. If it is NULL, starting beta is estimated by NMF.
  background_signature = NULL,  # background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
  normalize_counts = TRUE,  # if true, the input count matrix x is normalize such that the patients have the same number of mutation.
  nmf_runs = 10,  # number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
  lambda_rate_alpha = 0.05,
  lambda_rate_beta = 0.05,
  iterations = 30,
  max_iterations_lasso = 10000,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

- **x**: count matrix for a set of $n$ patients and 96 trinucleotides.
- **K**: numeric value (minimum 2) indicating the number of signatures to be discovered.
- **beta**: starting beta for the estimation. If it is NULL, starting beta is estimated by NMF.
- **background_signature**: background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
- **normalize_counts**: if true, the input count matrix $x$ is normalize such that the patients have the same number of mutation.
- **nmf_runs**: number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
lambda_rate_alpha
value of LASSO to be used for alpha between 0 and 1. This value should be
greater than 0. 1 is the value of LASSO that would shrink all the exposure
values to 0 within one step. The higher lambda_rate_alpha is, the sparser are the
resulting exposure values, but too large values may result in a reduced fit of the
observed counts.

lambda_rate_beta
value of LASSO to be used for beta between 0 and 1. This value should be
greater than 0. 1 is the value of LASSO that would shrink all the signatures to
0 within one step. The higher lambda_rate_beta is, the sparser are the resulting
signatures, but too large values may result in a reduced fit of the observed counts.

iterations
Number of iterations to be performed. Each iteration corresponds to a first step
where beta is fitted and a second step where alpha is fitted.

max_iterations_lasso
Number of maximum iterations to be performed during the sparsification via
Lasso.

seed
Seed for reproducibility.

verbose
boolean; Shall I print all messages?

Value
A list with the discovered signatures. It includes 6 elements: alpha: matrix of the discovered ex-
posure values beta: matrix of the discovered signatures starting_alpha: initial alpha on which the
method has been applied starting_beta: initial beta on which the method has been applied log-
lik_progression: log-likelihood values during the iterations. This values should be increasing, if not
the selected value of lambda is too high best_loglik: log-likelihood of the best signatures configu-
ration

Examples

data(patients)
data(starting_betas_example)
beta = starting_betas_example["5_signatures","Value"]
res = nmfLasso(x=patients[1:100,],
K=5,
beta=beta,
lambda_rate_alpha=0.05,
lambda_rate_beta=0.05,
iterations=5,
seed=12345)
Description

Perform the evaluation of different nmfLasso solutions by bootstrap for $K$ (unknown) somatic mutational signatures given a set of observations $x$. The estimation can slow down because of memory usage and intensive computations, when a big number of bootstrap repetitions is asked and when the analysis is performed for a big range of signatures ($K$). In this case, an advice may be to split the computation into multiple smaller sets.

Usage

```r
nmfLassoBootstrap(
  x,
  K = 3:10,
  starting_beta = NULL,
  background_signature = NULL,
  normalize_counts = TRUE,
  nmf_runs = 10,
  bootstrap_repetitions = 50,
  iterations = 30,
  max_iterations_lasso = 10000,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE,
  log_file = ""
)
```

Arguments

- **x** count matrix for a set of $n$ patients and 96 trinucleotides.
- **K** a range of numeric value (each of them greater than 1) indicating the number of signatures to be discovered.
- **starting_beta** a list of starting beta value for each configuration of $K$. If it is NULL, starting betas are estimated by NMF.
- **background_signature** background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
- **normalize_counts** if true, the input count matrix $x$ is normalize such that the patients have the same number of mutation.
- **nmf_runs** number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
- **bootstrap_repetitions** Number of time bootstrap should be repeated. Higher values result in better estimate, but are computationally more expensive.
- **iterations** Number of iterations to be performed. Each iteration corresponds to a first step where beta is fitted and a second step where alpha is fitted.
max_iterations_lasso
Number of maximum iterations to be performed during the sparsification via Lasso.

num_processes
Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.

seed
Seed for reproducibility.

verbose
boolean; Shall I print all messages?

log_file
log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

Value
A list of 3 elements: stability, RSS and evar. Here, stability reports the estimated cosine similarity for alpha and beta at each bootstrap repetition; RSS reports for each configuration the estimated residual sum of squares; finally, evar reports the explained variance.

Examples
```
data(background)
data(patients)
res = nmfLassoBootstrap(x=patients[1:100,],
K=3:5,
background_signature=background,
nmf_runs=1,
bootstrap_repetitions=2,
um_processes=NA,
seed=12345)
```

Description
Perform the assessment of different nmfLasso solutions by cross validation for K (unknown) somatic mutational signatures given a set of observations x. The estimation can slow down because of memory usage and intensive computations, when a big number of cross validation repetitions is asked and when the grid search is performed for a lot of configurations. In this case, an advice may be to split the computation into multiple smaller sets.

Usage
```
nmfLassoCV(
  x,
  K = 3:10,
  starting_beta = NULL,
  background_signature = NULL,
)```
normalize_counts = TRUE,
nmf_runs = 10,
lambda_values_alpha = c(0, 0.01, 0.05, 0.1),
lambda_values_beta = c(0, 0.01, 0.05, 0.1),
cross_validation_entries = 0.01,
cross_validation_iterations = 5,
cross_validation_repetitions = 50,
iterations = 30,
max_iterations_lasso = 10000,
um_processes = Inf,
seed = NULL,
verbose = TRUE,
log_file = ""
)

Arguments

x  count matrix for a set of n patients and 96 trinucleotides.
K  a range of numeric value (each of them greater than 1) indicating the number of signatures to be discovered.
starting_beta a list of starting beta value for each configuration of K. If it is NULL, starting betas are estimated by NMF.
background_signature background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
normalize_counts if true, the input count matrix x is normalize such that the patients have the same number of mutation.
nmf_runs number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
lambda_values_alpha value of LASSO to be used for alpha between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_rate_alpha is, the sparser are the resulting exposures, but too large values may result in a reduced fit of the observed counts.
lambda_values_beta value of LASSO to be used for beta between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_rate_beta is, the sparser are the resulting exposures, but too large values may result in a reduced fit of the observed counts.
cross_validation_entries Percentage of cells in the count matrix to be replaced by 0s during cross validation.
cross_validation_iterations For each configuration, the first time the signatures are discovered form a matrix with a percentage of values replaced by 0s. This may result in poor fit/results.
Then, we perform predictions of these entries and replace them with such predicted values. This parameter is the number of restarts to be performed to improve this estimate and obtain more stable solutions.

**cross_validation_repetitions**
Number of time cross-validation should be repeated. Higher values result in better estimate, but are computationally more expensive.

**iterations**
Number of iterations to be performed. Each iteration corresponds to a first step where beta is fitted and a second step where alpha is fitted.

**max_iterations_lasso**
Number of maximum iterations to be performed during the sparsification via Lasso.

**num_processes**
Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.

**seed**
Seed for reproducibility.

**verbose**
boolean; Shall I print all messages?

**log_file**
log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

**Value**
A list of 2 elements: grid_search_mse and and grid_search_loglik. Here, grid_search_mse reports the mean squared error for each configuration of performed cross validation; grid_search_loglik reports for each configuration the number of times the algorithm converged.

**Examples**
```r
data(background)
data(patients)
res = nmfLassoCV(x=patients[1:100,], K=3:5,
background_signature=background,
nmf_runs=1,
lambda_values_alpha=c(0.00),
lambda_values_beta=c(0.00),
cross_validation_repetitions=2,
num_processes=NA,
seed=12345)
```

**Description**
example of results obtained with the function nmf.LassoK on the counts input from Nik-Zainal, Serena, et al. (2016).
Usage
data(nmf_LassoK_example)

Format
results obtained with the function nmf.LassoK on the counts input from Nik-Zainal, Serena, et al. (2016)

Value
results obtained with the function nmf.LassoK on the counts input from Nik-Zainal, Serena, et al. (2016)

Description

Usage
data(patients)

Format
counts of the point mutations

Value
counts of point mutations for 560 tumors and 96 trinucleotides

Source
Description

Plot trinucleotides counts for a set of given patients.

Usage

```r
patients.plot(
  trinucleotides_counts,
  samples = rownames(trinucleotides_counts),
  freq = FALSE,
  xlabels = FALSE
)
```

Arguments

- `trinucleotides_counts`  
  trinucleotides counts matrix.
- `samples`  
  name of the samples. This should match a rownames in trinucleotides_counts.
- `freq`  
  boolean value; shall I display rates instead of counts?
- `xlabels`  
  boolean value; shall I display x labels?

Value

A ggplot2 object.

Examples

```r
data(patients)
patients.plot(trinucleotides_counts=patients,samples=c("PD10010a","PD10011a","PD10014a"))
```

Description

Perform the assessment of different sigAssignmentLasso solutions by cross validation for a given set of somatic mutational signatures and observations x. The estimation can slow down because of memory usage and intensive computations, when a big number of cross validation repetitions is asked and when the grid search is performed for a lot of configurations. In this case, an advice may be to split the computation into multiple smaller sets.
Usage

```r
sigAssignmentCV(
  x,
  beta,
  normalize_counts = TRUE,
  lambda_values_alpha = c(0, 0.01, 0.05, 0.1),
  cross_validation_entries = 0.01,
  cross_validation_iterations = 5,
  cross_validation_repetitions = 50,
  max_iterations_lasso = 10000,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE,
  log_file = ""
)
```

Arguments

- **x**
  count matrix for a set of n patients and 96 trinucleotides.
- **beta**
  beta to be fixed during the estimation of alpha.
- **normalize_counts**
  if true, the input count matrix x is normalize such that the patients have the same number of mutation.
- **lambda_values_alpha**
  value of LASSO to be used for alpha between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_rate_alpha is, the sparser are the resulting exposures, but too large values may result in a reduced fit of the observed counts.
- **cross_validation_entries**
  Percentage of cells in the count matrix to be replaced by 0s during cross validation.
- **cross_validation_iterations**
  For each configuration, the first time the signatures are discovered form a matrix with a percentage of values replaced by 0s. This may result in poor fit/results. Then, we perform predictions of these entries and replace them with such predicted values. This parameter is the number of restarts to be performed to improve this estimate and obtain more stable solutions.
- **cross_validation_repetitions**
  Number of time cross-validation should be repeated. Higher values result in better estimate, but are computationally more expensive.
- **max_iterations_lasso**
  Number of maximum iterations to be performed during the sparsification via Lasso.
- **num_processes**
  Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.
- **seed**
  Seed for reproducibility.
verbose: boolean; Shall I print all messages?

log_file: log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

Value

A list of 2 elements: grid_search_mse and grid_search_loglik. Here, grid_search_mse reports the mean squared error for each configuration of performed cross validation; grid_search_loglik reports for each configuration the number of times the algorithm converged.

Examples

data(patients)
data(starting_betas_example)
beta = starting_betas_example[["5_signatures","Value"]]
res = sigAssignmentCV(x=patients[1:100,],
                      beta=beta,
                      lambda_values_alpha=c(0.00),
                      cross_validation_repetitions=2,
                      num_processes=NA,
                      seed=12345)

Description

Estimate the range of lambda values for alpha to be considered in the signature assignment. Note that too small values of lambda result in dense exposures, but too large values lead to bad fit of the counts.

Usage

sigAssignmentEvaluation(
  x, beta,
  normalize_counts = TRUE,
  lambda_values = c(0.01, 0.05, 0.1, 0.2),
  max_iterations_lasso = 10000,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE,
  log_file = ""
)
**Arguments**

- **x**: count matrix for a set of n patients and 96 trinucleotides.
- **beta**: beta to be fixed during the estimation of alpha.
- **normalize_counts**: if true, the input count matrix x is normalize such that the patients have the same number of mutation.
- **lambda_values**: value of LASSO to be used for alpha between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the signatures to 0 within one step. The higher lambda_values is, the sparser are the resulting exposures, but too large values may result in a reduced fit of the observed counts.
- **max_iterations_lasso**: Number of maximum iterations to be performed during the sparsification via Lasso.
- **num_processes**: Number of processes to be used during parallel execution. To execute in single process mode, this parameter needs to be set to either NA or NULL.
- **seed**: Seed for reproducibility.
- **verbose**: boolean; Shall I print all messages?
- **log_file**: log file where to print outputs when using parallel. If parallel execution is disabled, this parameter is ignored.

**Value**

A list corresponding to results of the function sigAssignmentLasso for each value of lambda to be tested. This function allows to test a good range of lambda values for alpha to be considered. One should keep in mind that too small values generate dense solution, while too high ones leads to poor fit. This behavior is resampled in the values of loglik_progression, which should be increasing: too small values of lambda results in unstable log-likelihood through the iterations, while too large values make log-likelihood drop.

**Examples**

```r
data(patients)
data(starting_betas_example)
beta = starting_betas_example[["5_signatures","Value"]]
res = sigAssignmentEvaluation(x=patients[1:100,],
                           beta=beta,
                           lambda_values=c(0.01,0.05),
                           num_processes=NA,
                           seed=12345)
```
#### Description

Perform the assignment of somatic mutational signatures to patients given a set of observed counts $x$ and signatures $\beta$.

#### Usage

```r
sigAssignmentLasso(
  x,
  beta,
  normalize_counts = TRUE,
  lambda_rate_alpha = 0.05,
  max_iterations_lasso = 10000,
  seed = NULL,
  verbose = TRUE
)
```

#### Arguments

- **x**: count matrix for a set of $n$ patients and 96 trinucleotides.
- **beta**: beta to be fixed during the estimation of alpha.
- **normalize_counts**: if true, the input count matrix $x$ is normalize such that the patients have the same number of mutation.
- **lambda_rate_alpha**: value of LASSO to be used for alpha between 0 and 1. This value should be greater than 0. 1 is the value of LASSO that would shrink all the exposure values to 0 within one step. The higher lambda_rate_alpha is, the sparser are the resulting exposure values, but too large values may result in a reduced fit of the observed counts.
- **max_iterations_lasso**: Number of maximum iterations to be performed during the sparsification via Lasso.
- **seed**: Seed for reproducibility.
- **verbose**: boolean; Shall I print all messages?

#### Value

A list with the discovered signatures and their assignment to patients. It includes 2 elements: 
- **alpha**: matrix of the assigned exposure values
- **beta**: matrix of the discovered signatures
Examples

```r
data(patients)
data(starting_betas_example)
beta = starting_betas_example["5_signatures","Value"]
res = sigAssignmentLasso(x=patients[1:100,],beta=beta,lambda_rate_alpha=0.05,seed=12345)
```

---

**signatures.plot**

Description

Plot the inferred mutational signatures.

Usage

```
signatures.plot(beta, useRowNames = FALSE, xlabels = FALSE)
```

Arguments

- `beta`: matrix with the inferred mutational signatures.
- `useRowNames`: boolean value; shall I use the rownames from beta as names for the signatures?
- `xlabels`: boolean value; shall I display x labels?

Value

A ggplot2 object.

Examples

```r
data(nmf_LassoK_example)
signatures.plot(beta=nmf_LassoK_example$beta)
```

---

**ssm560_reduced**

Description

a reduced version of the point mutations for 560 breast tumors in the format compatible with the import function

Usage

```
data(ssm560_reduced)
```
**startingBetaEstimation**

**Format**

reduced versione of the counts of the point mutations in the format compatible with the import function

**Value**

reduced versione of the counts of point mutations for 560 tumors and 96 trinucleotides in the format compatible with the import function

**Source**


---

**startingBetaEstimation**

**startingBetaEstimation**

**Description**

Perform a robust estimation of the starting betas for the nmfLasso method

**Usage**

```r
startingBetaEstimation(
  x,  # count matrix for a set of n patients and 96 trinucleotides.
  K = 3:10,  # numeric value (minimum 2) indicating the number of signatures to be discovered.
  background_signature = NULL,  # background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
  normalize_counts = TRUE,  # if true, the input count matrix x is normalize such that the patients have the same number of mutation.
  nmf_runs = 10,  # numeric value indicating the number of runs to be performed by NMF.
  seed = NULL,  # numeric value indicating the seed for the random number generator.
  verbose = TRUE  # if true, the progress of the algorithm is printed to the console.
)
```

**Arguments**

- **x**: count matrix for a set of n patients and 96 trinucleotides.
- **K**: numeric value (minimum 2) indicating the number of signatures to be discovered.
- **background_signature**: background signature to be used. If not provided, a warning is thrown and an initial value for it is estimated by NMF. If beta is not NULL, this parameter is ignored.
- **normalize_counts**: if true, the input count matrix x is normalize such that the patients have the same number of mutation.
starting_betas_example

nmf_runs  
number of iteration (minimum 1) of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.

seed  
Seed for reproducibility.

verbose  
boolean; Shall I print all messages?

Value

A list containing the starting beta values for each configuration of K.

Examples

data(background)
data(patients)
res = startingBetaEstimation(x=patients[1:100,],
    K=3:5,
    background_signature=background,
    nmf_runs=1,
    seed=12345)

starting_betas_example

example of results obtained with the function starting.betas.estimation on the counts input from Nik-Zainal, Serena, et al. (2016).

Description

example of results obtained with the function starting.betas.estimation on the counts input from Nik-Zainal, Serena, et al. (2016).

Usage

data(starting_betas_example)

Format

results obtained with the function starting.betas.estimation on the counts input from Nik-Zainal, Serena, et al. (2016)

Value

results obtained with the function starting.betas.estimation on the counts input from Nik-Zainal, Serena, et al. (2016)
Index

background, 3
background2, 3

cv_example, 4

import.trinucleotides.counts, 4
imported_data, 5

lambda_range_example, 9
lambdaRangeAlphaEvaluation, 5
lambdaRangeBetaEvaluation, 7

mutation_categories, 9

nmf_LassoK_example, 15
nmfLasso, 10
nmfLassoBootstrap, 11
nmfLassoCV, 13

patients, 16
patients.plot, 17

sigAssignmentCV, 17
sigAssignmentEvaluation, 19
sigAssignmentLasso, 21
signatures.plot, 22
ssm560_reduced, 22
starting_betas_example, 24
startingBetaEstimation, 23