Package 'SparseSignatures'

October 17, 2020

Version 1.8.0

Date 2020-04-14

Title SparseSignatures

Maintainer Luca De Sano <luca.desano@gmail.com>

Depends R (>= 4.0), NMF

Imports BiocGenerics (>= 0.31.6), nnlasso, nnls, parallel, data.table, Biostrings, GenomicRanges, IRanges, BSgenome, BSgenome.Hsapiens.1000genomes.hs37d5, GenomeInfoDb, ggplot2, gridExtra

Suggests 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

LazyData TRUE

License file LICENSE

URL https://github.com/danro9685/SparseSignatures

BugReports https://github.com/danro9685/SparseSignatures

biocViews BiomedicalInformatics, SomaticMutation

RoxygenNote 7.1.0

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/SparseSignatures

git_branch RELEASE_3_11

git_last_commit ad76117

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

Author Daniele Ramazzotti [cre, aut],
Avantika Lal [aut],
Keli Liu [ctb],
Luca De Sano [ctb],
Robert Tibshirani [ctb],
Arend Sidow [aut]

R topics documented:

`as.alpha` *as.alpha*

Description

Return the discovered alpha values from an input object as the one generated by the function nmf.LassoK.

Usage

```
as.alpha(nmf.LassoK.result)
```

Arguments

`nmf.LassoK.result`
: discovered signatures into an object as the one generated by the function nmf.LassoK

Value

A matrix with alpha values

Examples

```
data(nmf_LassoK_example)
res = as.alpha(nmf_LassoK_example)
```

---

as.alpha.in.range *as.alpha.in.range*

Description

Return the discovered alpha values for a given configuration from an input object as the one generated by the function evaluate.lambda.range.

Usage

```
as.alpha.in.range(lambda.range.result, lambda_value = 0.2)
```

Arguments

`lambda.range.result`
: discovered signatures into an object as the one generated by the function evaluate.lambda.range

`lambda_value`
: value of LASSO used for the estimation whose configuration has to be returned

Value

A matrix with alpha values

Examples

```
data(lambda_range_example)
res = as.alpha.in.range(lambda_range_example,lambda_value=0.10)
```

as.beta *as.beta*

Description

Return the discovered signatures (beta values) from an input object as the one generated by the function nmf.LassoK.

Usage

```
as.beta(nmf.LassoK.result)
```

Arguments

nmf.LassoK.result	discovered signatures into an object as the one generated by the function nmf.LassoK

Value

A matrix with beta values

Examples

```
data(nmf_LassoK_example)
res = as.beta(nmf_LassoK_example)
```

as.beta.in.range *as.beta.in.range*

Description

Return the discovered signatures (beta values) for a given configuration from an input object as the one generated by the function evaluate.lambda.range.

Usage

```
as.beta.in.range(lambda.range.result, lambda_value = 0.2)
```

Arguments

lambda.range.result	discovered signatures into an object as the one generated by the function evaluate.lambda.range
lambda_value	value of LASSO used for the estimation whose configuration has to be returned

Value

A matrix with beta values

Examples

```
data(lambda_range_example)
res = as.beta.in.range(lambda_range_example,lambda_value=0.10)
```

---

as.loglik.progression *as.loglik.progression*

Description

Return the log-likelihood values for each iteration during the estimation of the signatures from an input object as the one generated by the function nmf.LassoK.

Usage

```
as.loglik.progression(nmf.LassoK.result)
```

Arguments

nmf.LassoK.result	discovered signatures into an object as the one generated by the function nmf.LassoK

Value

A vector with log-likelihood values for each iteration during the estimation of the signatures

Examples

```
data(nmf_LassoK_example)
res = as.loglik.progression(nmf_LassoK_example)
```

---

as.loglik.progression.in.range *as.loglik.progression.in.range*

Description

Return the log-likelihood values for a given configuration for each iteration during the estimation of the signatures from an input object as the one generated by the function evaluate.lambda.range.

Usage

```
as.loglik.progression.in.range(lambda.range.result, lambda_value = 0.2)
```

Arguments

lambda.range.result	discovered signatures into an object as the one generated by the function evaluate.lambda.range
lambda_value	value of LASSO used for the estimation whose configuration has to be returned

Value

A vector with log-likelihood values for each iteration during the estimation of the signatures

Examples

```
data(lambda_range_example)
res = as.loglik.progression.in.range(lambda_range_example,lambda_value=0.10)
```

`as.mean.squared.error` *as.mean.squared.error*

Description

Return the mean squared error between the observed counts and the predicted ones for all the configurations considered during cross validation. This function expects an input object as the one generated by the function nmf.LassoCV.

Usage

```
as.mean.squared.error(nmf.LassoCV.result)
```

Arguments

`nmf.LassoCV.result`
: results of cross validation into an object as the one generated by the function nmf.LassoCV

Value

A list with distributions and median values of the mean squared error values computed between the observed counts and the predicted ones at each iteration of cross validation

Examples

```
data(cv_example)
res = as.mean.squared.error(cv_example)
```

`as.starting.beta` *as.starting.beta*

Description

Return the initial values of beta used for the estimation of the signatures from an input object as the one generated by the function nmf.LassoK.

Usage

```
as.starting.beta(nmf.LassoK.result)
```

Arguments

`nmf.LassoK.result`
: discovered signatures into an object as the one generated by the function nmf.LassoK

Value

A matrix with the initial values of beta used for the estimation of the signatures

Examples

```
data(nmf_LassoK_example)
res = as.starting.beta(nmf_LassoK_example)
```

`as.starting.beta.in.range` *as.starting.beta.in.range*

Description

Return the initial values of beta used for the estimation of the signatures for a given configuration from an input object as the one generated by the function evaluate.lambda.range.

Usage

```
as.starting.beta.in.range(lambda.range.result, lambda_value = 0.2)
```

Arguments

`lambda.range.result`
: discovered signatures into an object as the one generated by the function nmf.LassoK

`lambda_value`
: value of LASSO used for the estimation whose configuration has to be returned

Value

A matrix with the initial values of beta used for the estimation of the signatures

Examples

```
data(lambda_range_example)
res = as.starting.beta.in.range(lambda_range_example,lambda_value=0.10)
```

background *germline replication error*

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

Rahbari, Raheleh, et al. "Timing, rates and spectra of human germline mutation." Nature genetics 48.2 (2016): 126.

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)

evaluate.lambda.range *evaluate.lambda.range*

Description

Estimate the range of lambda values 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

```
evaluate.lambda.range(
  x,
  K = 6,
  beta = NULL,
  background_signature = NULL,
  nmf_runs = 10,
  lambda_values = c(0.1, 0.2, 0.3, 0.4, 0.5),
  iterations = 20,
  max_iterations_lasso = 10000,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	count matrix.
K	numeric value (greater than 1) 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.
nmf_runs	number of iteration of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
lambda_values	range of values of LASSO to be used 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 is, the sparser are the resulting signatures, but too large values result in a poor fit of the counts.
iterations	Number of iterations to be performed. Each iteration correspond to a first step where the counts are fitted and a second step where sparsity is enhanced.
max_iterations_lasso	Number of maximum iterations to be performed during the sparsification.
num_processes	Number of processes to be used during parallel execution. If executing in single process mode, this is ignored.
seed	Seed for reproducibility.
verbose	boolean; Shall I print all messages?

Value

A list corresponding to results of the function nmf.LassoK for each value of lambda to be tested. This function allows to test a good range of lambda values 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 lamda results in unstable log-likelihood through the iterations, while too large values make log-likelihood drop.

`import.counts.data` *import.counts.data*

Description

Import point mutations data to build the count matrix to extract mutatational signatues.

Usage

```
import.counts.data(input, bsg, mutation_categories)
```

Arguments

`input`	either a data.frame/data.table object or a file with 5 columns: sample name, chromosome, position, ref, alt.
`bsg`	a BSgenome object for the reference genome. Chromosome names have to match the input table.
`mutation_categories`	array with the 96 mutational categories to be considered. It is provided along with the package by data(mutation_categories).

Value

A count matrix to extract mutatational signatues

Examples

```
data(ssm560_reduced)
library("BSgenome.Hsapiens.1000genomes.hs37d5")
bsg = BSgenome.Hsapiens.1000genomes.hs37d5
data(mutation_categories)
imported_data = import.counts.data(input=ssm560_reduced,bsg=bsg,mutation_categories=mutation_categories)
```

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

```
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

```
data(mutation_categories)
```

Format

matrix of 96 trinucleotides mutation categories

Value

matrix of 96 trinucleotides mutation categories

nmf.LassoCV *nmf.LassoCV*

Description

Perform the discovery by cross validation of K (unknown) somatic mutational signatures given a set of observations x. The estimation can slow down because of memory usage, when I high number of cross validation repetitions is asked and when the grid search is performed for a lot of configurations. In this case, we advice to split the computation into multiple smaller sets.

Usage

```
nmf.LassoCV(
  x,
  K = 3:10,
  starting_beta = NULL,
  background_signature = NULL,
  nmf_runs = 10,
  lambda_values = c(0.1, 0.2, 0.3),
  cross_validation_entries = 0.05,
  cross_validation_iterations = 5,
  cross_validation_repetitions = 10,
  iterations = 20,
  max_iterations_lasso = 10000,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	count matrix.
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.
nmf_runs	number of iteration of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
lambda_values	range of values of LASSO to be used 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 is, the sparser are the resulting signatures, but too large values result in a poor fit of the counts.
cross_validation_entries	Percentage of cells in the count matrix to be replaced by 0s.
cross_validation_iterations	For each configuration, the first time the signatures are discovered form a matrix with a ercentage of values replaced by 0s. This may result in a poor. This parameter is the number of restarts to be performed to improve this estimate.

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

iterations
: Number of iterations to be performed. Each iteration correspond to a first step where the counts are fitted and a second step where sparsity is enhanced.

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

num_processes
: Number of processes to be used during parallel execution. If executing in single process mode, this is ignored.

seed
: Seed for reproducibility.

verbose
: boolean; Shall I print all messages?

Value

A list corresponding with 3 elements: grid_search, starting_beta and mean_squared_error. Here, grid_search provides all the results of the executions within the grid search; starting_beta is the set of initial values of beta used for each configuration and mean_squared_error is the mean squared error between the observed counts and the predicted ones for each configuration.

`nmf.LassoK` *nmf.LassoK*

Description

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

Usage

```
nmf.LassoK(
  x,
  K,
  beta = NULL,
  background_signature = NULL,
  nmf_runs = 10,
  lambda_rate = 0.2,
  iterations = 20,
  max_iterations_lasso = 10000,
  num_processes = Inf,
  parallel = NULL,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x
: count matrix.

K
: numeric value (greater than 1) 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.

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

lambda_rate
: value of LASSO to be used 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 is, the sparser are the resulting signatures, but too large values result in a poor fit of the counts.

iterations
: Number of iterations to be performed. Each iteration correspond to a first step where the counts are fitted and a second step where sparsity is enhanced.

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

num_processes
: Number of processes to be used during parallel execution. If executing in single process mode, this is ignored.

parallel
: Cluster object for parallel execution.

seed
: Seed for reproducibility.

verbose
: boolean; Shall I print all messages?

Value

A list with the discovered signatures. It includes 5 elements: alpha: matrix of the discovered alpha values beta: matrix of the discovered signatures starting_beta: initial signatures on which the method has been apploid best_loglik: log-likelihood of the best signatures configuration log-lik_progression: log-likelihood values during the iterations. This values should be increasing, if not the selected value of lambda is too high

Examples

```
data(starting_betas_example)
beta = starting_betas_example[["5_signatures","Value"]]
res = nmf.LassoK(x=patients,K=5,beta=beta,background=background,lambda_rate=0.10,iterations=5,num_processes
```

---

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

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)

`patient.plot` *patient.plot*

Description

Plot the counts for a patient.

Usage

```
patient.plot(countMatrix, patientName, xlabels = TRUE, freq = FALSE)
```

Arguments

`countMatrix`	count matrix as the one generated by the import.data function.
`patientName`	name of the patient. This should match a rowname of the countMatrix.
`xlabels`	boolean value; shall I display x labels?
`freq`	boolean value; shall I display rates instead of counts?

Value

A ggplot2 object

Examples

```
data(patients)
patient.plot(countMatrix=patients,patientName="PD18775a")
```

`patients` *point mutations for 560 breast tumors*

Description

dataset of counts of the point mutations detected in 560 breast tumors published in Nik-Zainal, Serena, et al. (2016).

Usage

```
data(patients)
```

Format

counts of the point mutations

Value

counts of point mutations for 560 tumors and 96 trinucleotides

Source

Nik-Zainal, Serena, et al. "Landscape of somatic mutations in 560 breast cancer whole-genome sequences." Nature 534.7605 (2016): 47.

`signatures.plot` *signatures.plot*

Description

Plot the discovered signatures.

Usage

```
signatures.plot(
  beta,
  useColNames = TRUE,
  mutation_categories = NULL,
  firstBackground = TRUE,
  xlabels = TRUE
)
```

Arguments

`beta`	discovered signatures
`useColNames`	boolean value; shall I use the colnames from beta as names for the signatures?
`mutation_categories`	if useColNames is FALSE, the trinucleotides categories to be considered can be specified. An example is provided in the package and can be loaded by data(mutation_categories)
`firstBackground`	boolean value; shall I display the background signature as the first element?
`xlabels`	boolean value; shall I display x labels?

Value

A ggplot2 object

Examples

```
data(nmf_LassoK_example)
beta = as.beta(nmf_LassoK_example)
signatures.plot(beta=beta)
```

ssm560_reduced *a reduced version of the point mutations for 560 breast tumors in the format compatible with the import function*

Description

reduced versione of the dataset of counts of the point mutations detected in 560 breast tumors published in Nik-Zainal, Serena, et al. (2016).

Usage

```
data(ssm560_reduced)
```

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

Nik-Zainal, Serena, et al. "Landscape of somatic mutations in 560 breast cancer whole-genome sequences." Nature 534.7605 (2016): 47.

starting.betas.estimation *starting.betas.estimation*

Description

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

Usage

```
starting.betas.estimation(
  x,
  K = 3:10,
  background_signature = NULL,
  nmf_runs = 10,
  num_processes = Inf,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

x	count matrix.
K	range of numeric values (each of them greater than 1) indicating the number of signatures to be discovered.
background_signature	background signature to be used. If not provided, a warning is thrown.
nmf_runs	number of iteration of NMF to be performed for a robust estimation of starting beta. If beta is not NULL, this parameter is ignored.
num_processes	Number of processes to be used during parallel execution. If executing in single process mode, this is ignored.
seed	Seed for reproducibility.
verbose	boolean; Shall I print all messages?

Value

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

---

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