Package ‘signeR’

April 12, 2018

Type Package

Title Empirical Bayesian approach to mutational signature discovery

Version 1.4.0

Author Rafael Rosales, Rodrigo Drummond, Renan Valieris, Israel Tojal da Silva

Maintainer Renan Valieris <renan.valieris@cipe.accamargo.org.br>

Description The signeR package provides an empirical Bayesian approach to mutational signature discovery. It is designed to analyze single nucleotide variaton (SNV) counts in cancer genomes, but can also be applied to other features as well. Functionalities to characterize signatures or genome samples according to exposure patterns are also provided.

License GPL-3

Imports BiocGenerics, Biostrings, BSgenome (>= 1.36.3), class, graphics, grDevices, GenomicRanges, nloptr, methods, NMF, stats, utils, VariantAnnotation, PMCMR

Depends NMF

LinkingTo Rcpp, RcppArmadillo (>= 0.7.100)

SystemRequirements C++11

NeedsCompilation yes

ByteCompile TRUE

biocViews GenomicVariation, SomaticMutation, StatisticalMethod, Visualization

Suggests knitr, rtracklayer, BSgenome.Hsapiens.UCSC.hg19

VignetteBuilder knitr

R topics documented:

signeR-package *Empirical Bayesian approach to mutational signature discovery*

Description

The signeR package provides an empirical Bayesian approach to mutational signature discovery. It is designed to analyze single nucleotide variaton (SNV) counts in cancer genomes, but can also be applied to other features as well. Functionalities to characterize signatures or genome samples according to exposure patterns are also provided.

Details

signeR package focus on the characterization and analysis of mutational processes. Its functionalities can be divided in three steps. Firstly, it provides tools to process VCF files and generate matrices of SNV mutation counts and mutational opportunities, both divided according to a 3bp context (mutation site and its neighboring bases). Secondly, the main part of the package takes those matrices as input and applies a Bayesian approach to estimate the number of underlying signatures and their mutational profiles. Thirdly, the package provides tools to correlate the activities of those signatures with other relevant information, e.g. clinical data, in order to infer conclusions about the analyzed genome samples, which can be useful for clinical applications.

Author(s)

Rodrigo Drummond, Rafael Rosales, Renan Valieris, Israel Tojal da Silva

Maintainer: Renan Valieris <renan.valieris@cipe.accamargo.org.br>

References

This work has been submitted to Bioinformatics under the title "signeR: An empirical Bayesian approach to mutational signature discovery".

L. B. Alexandrov, S. Nik-Zainal, D. C. Wedge, P. J. Campbell, and M. R. Stratton. Deciphering Signatures of Mutational Processes Operative in Human Cancer. Cell Reports, 3(1):246-259, Jan. 2013. doi:10.1016/j.celrep.2012.12.008.

A. Fischer, C. J. Illingworth, P. J. Campbell, and V. Mustonen. EMu: probabilistic inference of mutational processes and their localization in the cancer genome. Genome biology, 14(4):R39, Apr. 2013. doi:10.1186/gb-2013-14-4-r39.

Examples

```
vignette(package="signeR")
```

Classify *Classify unknown samples*

Description

Classify: Assign unknown samples to previously defined groups.

Usage

```
## S4 method for signature 'SignExp,character'
Classify(signexp_obj, labels, method="knn",
    k=3, weights=NA, plot_to_file=FALSE, file="Classification_barplot.pdf",
    colors=NA_character_, min_agree=0.75, ...)
```

Arguments

`signexp_obj` A SignExp object returned by signeR function.

`labels` Sample labels. Every sample labeled as NA will be classified according to its mutational profile and the profiles of labeled samples.

`method` Classification algorithm used. Default is k-Nearest Neighbors (kNN). Any other algorithm may be used, as long as it is customized to satisfy the following conditions:
Input: a matrix of labeled samples, with one sample per line and one feature per column; a matrix of unlabeled samples to classify, with the same structure; an array of labels, with one entry for each labeled sample.
Output: an array of assigned labels, one for each unlabeled sample.

`k` Number of nearest neighbors considered for classification, used only if method="kNN". Default is 3.

`weights` Vector of weights applied to the signatures when performing classification. Default is NA, which leads all the signatures to have weight=1.

`plot_to_file` Whether to save the plot to the file parameter. Default is FALSE.

`file` File that will be generated with classification graphic output.

`colors` Array of color names, one for each sample class. Colors will be recycled if the length of this array is less than the number of classes.

`min_agree` Minimum frequency of agreement among individual classifications. Samples showing a frequency of agreement below this value are considered as "undefined". Default is 0.75.

`...` additional parameters for classification algorithm (defined by "method" above).

Value

A list with the following items:

`class` The assigned classes for each unlabeled sample.

`freq` Classification agreement for each unlabeled sample: the relative frequency of assignment of each sample to the group specified in "class".

`allfreqs` Matrix with one column for each unlabeled sample and one row for each group label. Contains the assignment frequencies of each sample to each group.

Examples

```
# assuming signatures is the return value of signeR()


my_labels <- c("a","a",NA,"b","b",NA)
Class <- Classify(signatures$SignExposures, labels=my_labels)

# see also
vignette(package="signeR")
```

DiffExp *Differential Exposure Analysis*

Description

DiffExp : Identify signatures with significantly different activities among sample groups.

Usage

```
## S4 method for signature 'SignExp,character'
DiffExp(signexp_obj, labels,
    method=kruskal.test, contrast="all", quant=0.5, cutoff=0.05,
    p.adj= "BH",plot_to_file=FALSE, file="Diffexp_boxplot.pdf",
    colored=TRUE, relative = FALSE, ...)
```

Arguments

`signexp_obj` a SignExp object returned by signeR function.

`labels` sample labels used to define sample groups.

`method` algorithm used to compare each signature exposures among sample groups. Default is kruskal.test, which leads to the use of Kruskal-Wallis Rank Sum Test.

`contrast` defines which sample groups will be considered in the analysis. Default is "all", which leads the algorithm to evaluates the null hypothesis of exposure levels beeing constant in all groups. Instead, if this parameter contains a list of group labels, the algorithm will evaluate the null hypothesis of exposure levels beeing constant among those groups.

`quant` the p-values quantile which, after log-transform, will be used as DES (Diferental Exposure Score). Deafult is 0.5, which means the median log-transformed p-value will be considered as DES.

`p.adj` correction method for p-values adjust at the post-hoc tests performed when there are more then two group labels. See p.adjust for options.

`cutoff` threshold for p-values quantile for signatures to be considered as showing differential exposure.

`plot_to_file` Whether to save the plot to the file parameter. Default is FALSE.

`file` Output file to export p-values boxplot.

`colored` Boolean variable, if TRUE boxplots of differentially exposed signatures will be colored in green, cutoff line will be colored in red and line segments showing the transformed p-value quantile used for DE evaluation will be colored in blue. Otherwise the plot will be black & white.

`relative`	Whether tests should be performed on absolute or relative signature contributions to each sample mutations. Default is FALSE (absolute contributions will be tested).
`...`	additional parameters for test algorithm defined by the method parameter.

Value

A list with the following items:

`Pvquant`	boolean array with one entry for each signature, indicating whether it shows differential exposure.
`Pvalues`	matrix containing all computed p-values, with one row for each signature.
`MostExposed`	for each differentially exposed signature, this array contains the label of the group where it showed higher levels of exposure. Contains NA for signatures not showing differential exposure.
`Differences`	List of matrices, exported only when there are more than two groups in the analysis and any signature is found to be differentially active. Each matrix corresponds to one of the highlighted signatures and show the results of comparisons among groups, with the significant ones marked as TRUE.

Examples

```
# assuming signatures is the return value of signeR()

# labels vector, one for each sample
my_labels <- c("a","a","b","b")

diff_exposure <- DiffExp(signatures$SignExposures,labels=my_labels)

# see also
vignette(package="signeR")
```

---

generateMatrix *count matrix and opportinity matrix generators*

Description

genCountMatrixFromVcf : generate count matrix from a VCF file.
genOpportunityFromGenome : generate opportunity matrix from a target regions set.

Usage

```
genCountMatrixFromVcf(bsgenome, vcfobj)
genOpportunityFromGenome(bsgenome, target_regions, nsamples=1)
```

Arguments

`bsgenome`	A BSgenome object, equivalent to the genome used for the variant call.
`vcfobj`	A VCF object. See VCF-class from the VariantAnnotation package.
`target_regions`	A GRanges object, describing the target region analyzed by the variant caller.
`nsamples`	Number of samples to generate the matrix, should be the same number as rows of the count matrix.

Value

A matrix of samples x (96 features).
Each feature is a SNV change with a 3bp context.

Examples

```
library(rtracklayer)
library(VariantAnnotation)

# input files, variant call and target
vcf_file <- system.file("extdata","example.vcf", package="signeR")
bed_file <- system.file("extdata","example.bed", package="signeR")

# BSgenome, will depend on your variant call
library(BSgenome.Hsapiens.UCSC.hg19)

vcfobj <- readVcf(vcf_file, "hg19")
mut <- genCountMatrixFromVcf(BSgenome.Hsapiens.UCSC.hg19, vcfobj)

target_regions <- import(con=bed_file, format="bed")
opp <- genOpportunityFromGenome(BSgenome.Hsapiens.UCSC.hg19,
    target_regions, nsamples=nrow(mut))


# see also
vignette(package="signeR")
```

`methods` *SignExp class methods*

Description

setSamples: Define sample names for a SignExp object, according to the "names" argument.

setMutations: Define mutation names for a SignExp object, according to the "mutations" argument.

Normalize: Normalize a SignExp object so that the entries of each signature sum up to one.

Reorder_signatures: Change the order of the signatures in a SignExp object. New signature order wil be defined by the "ord" argument.

Reorder_samples: Change samples order, according to ord parameter.

Reorder_mutations: Change mutations order, according to ord parameter.

Average_sign: Exports an aproximation of the signatures obtained by the averages of the samples for the signature matrix P.

Median_sign: Exports an aproximation of the signatures obtained by the medians of the samples for signature matrix P.

Average_exp: Exports an aproximation of the exposures obtained by the averages of the samples for exposure matrix E.

Median_exp: Exports an aproximation of the exposures obtained by the medians of the samples for exposure matrix E.

Usage

```
## S4 method for signature 'SignExp'
setSamples(signexp_obj, names)
## S4 method for signature 'SignExp'
setMutations(signexp_obj, mutations)
## S4 method for signature 'SignExp'
Normalize(signexp_obj)
## S4 method for signature 'SignExp,numeric'
Reorder_signatures(signexp_obj, ord)
## S4 method for signature 'SignExp,numeric'
Reorder_samples(signexp_obj, ord)
## S4 method for signature 'SignExp,numeric'
Reorder_mutations(signexp_obj, ord)
## S4 method for signature 'SignExp'
Average_sign(signexp_obj, normalize=TRUE)
## S4 method for signature 'SignExp'
Median_sign(signexp_obj, normalize=TRUE)
## S4 method for signature 'SignExp'
Average_exp(signexp_obj, normalize=TRUE)
## S4 method for signature 'SignExp'
Median_exp(signexp_obj, normalize=TRUE)
```

Arguments

signexp_obj	a SignExp object returned by signeR function. e.g.: sig$SignExposures
names	Vector of sample names.
mutations	Vector of mutations, e.g. "C>A:TCG".
normalize	Whether the signatures should be normalized before extracting approximations. Default is TRUE.
ord	Vector with the new signature order.

Value

setSamples, setMutations, Normalize and Reorder_* returns a modified SignExp object.
Average_sign, Median_sign, Average_exp and Median_exp return a matrix with the corresponding approximation.

Examples

```
# each function needs the SignExposures object
# which is part of the result of the signeR() call
signexp  <- Normalize(signatures$SignExposures)
signexp  <- Reorder_signatures(signatures$SignExposures,ord=c(2,1))
```

```
matrix_p <- Median_sign(signatures$SignExposures)
# etc ...

# see also
vignette(package="signeR")
```

plots *signeR plot functions*

Description

BICboxplot: Plot the measured values of the Bayesian Information Criterion (BICs) for tested model dimensions.

Paths: Plot the convergence of the Gibbs sampler for signatures and exposures on separate charts.

SignPlot: Plot the mutational signatures in a barchart, with error bars according to the variation of individual entries along generated Gibbs samples.

SignHeat: Plot the mutations signatures in a heatmap.

ExposureBarplot: Barplot of estimated exposure values, showing the contribution of the signatures to the mutation counts of each genome sample.

ExposureBoxplot: Boxplot of exposure values, showing their variation along generated Gibbs samples.

ExposureHeat: Plot a heatmap of the exposures, along with a dendrogram of the samples grouped by exposure levels.

Usage

```
BICboxplot(signeRout, plot_to_file=FALSE, file="Model_selection_BICs.pdf")
## S4 method for signature 'SignExp'
Paths(signexp_obj, plot_to_file=FALSE,
    file_suffix="plot.pdf", plots_per_page=4, ...)
## S4 method for signature 'SignExp'
SignPlot(signexp_obj, plot_to_file=FALSE,
    file="Signature_plot.pdf", pal="bcr1", threshold=0, plots_per_page=4,
    gap=1, reord=NA, ...)
## S4 method for signature 'SignExp'
SignHeat(signexp_obj, plot_to_file=FALSE,
    file="Signature_heatmap.pdf", nbins=20, pal="roh", ...)
## S4 method for signature 'SignExp'
ExposureBarplot(signexp_obj, plot_to_file=FALSE,
    file="Exposure_barplot.pdf", col='tan2', threshold=0, relative=FALSE,
    title="", samplenames=TRUE, ...)
## S4 method for signature 'SignExp'
ExposureBoxplot(signexp_obj, plot_to_file=FALSE,
    file="Exposure_boxplot.pdf", col='tan2', threshold=0,
    plots_per_page=4, reord=NA, ...)
## S4 method for signature 'SignExp'
ExposureHeat(signexp_obj, plot_to_file=FALSE,
    file="Exposure_heatmap.pdf", nbins=20, pal="roh", distmethod="euclidean",
        clustermethod="complete", ...)
```

Arguments

`signexp_obj`	A SignExp object returned by signeR function. e.g.: sig$SignExposures
`signeRout`	The list returned by the signeR function.
`plot_to_file`	Whether to save the plot to the file parameter. Default is FALSE.
`file`	Output pdf file of the plots.
`pal`	Color palette used. Options are: "brew","lba","bcr1", "bcr2","bw","roh".
`threshold`	Entries below this value will be rounded to 0. Default is 0 (all entries are kept).
`plots_per_page`	How many plots in a single page, default is 4.
`gap`	Distance between consecutive bars on the plot.
`reord`	Order of signatures for plotting. Should be a permutation of 1:nsig, where nsig is the number of signatures. By default, signatures are ordered by the total exposure, in decreasing order.
`nbins`	The range of signature entries is divided in this number of bins for plotting, each bin corresponding to a different color.
`file_suffix`	The suffix of the output file.
`col`	Single color name for boxplots.
`distmethod`	Distance measure used for grouping samples. Default is "euclidean", see the documentation of the dist function for other options.
`clustermethod`	Agglomeration method used for grouping samples. Default is "complete", see the documentation of the hclust function for other options.
`relative`	Whether to normalize exposures of each sample so that they sum up to one. Default is TRUE, thus generating a barplot of relative contributions of the signatures to mutation counts. Otherwise, absolute contributions to mutation counts will be displayed.
`title`	Main title added to the plot. Default is no title.
`samplenames`	Whether sample names will be shown in the plot. Default is TRUE, however sample names are not shown if there are more then 50 samples.
`...`	.

Value

The plot result is exported to the current graphic device. If plot_to_file=TRUE, the plot is saved in the file defined by the file argument.

Examples

```
# each plot function need the SignExposures object
# which is part of the result of the signeR() call
```

```
SignPlot(signatures$SignExposures)
Paths(signatures$SignExposures)
# etc ...

# BICboxplot needs the returned list itself
BICboxplot(signatures)

# see also
vignette(package="signeR")
```

signeR *signeR*

Description

Generates the signatures.

Usage

```
signeR(M, Mheader = TRUE, samples = "rows", Opport = NA,
    Oppheader = FALSE, nsig = NA, nlim = c(NA, NA),
    try_all = FALSE, ap = NA, bp = NA, ae = NA, be = NA,
    lp = NA, le = NA, var.ap = 10, var.ae = 10,
    testing_burn = 1000, testing_eval = 1000, EM_eval = 100,
    main_burn = 10000, main_eval = 2000, start = "lee",
    estimate_hyper = FALSE, EMit_lim=100)
```

Arguments

`M` mutation counts matrix of samples x features.

`Mheader` if M have colnames defined use TRUE, if FALSE a default order will be assumed.

`samples` if the samples are row-wise or column-wise in M, default is "row".

`Opport` context count matrix of samples x features in the target genome or region.

`Oppheader` if Opport have header defined.

`nsig` number of signatures, which can be provided or estimated by the algorithm.

`nlim` define a interval to search for the optimal number of signatures.

`try_all` if true, all possible values for nsig will be tested

`ap` shape parameter of the gamma distribution used to generate the entries of a matrix of rate parameters of the gamma distributions which generate signatures.

`bp` rate parameter of the gamma distribution used to generate the entries of a matrix of rate parameters of the gamma distributions which generate signatures.

`ae` shape parameter of the gamma distribution used to generate the entries of a matrix of rate parameters of the gamma distributions which generate exposures.

`be` rate parameter of the gamma distribution used to generate the entries of a matrix of rate parameters of the gamma distributions which generate exposures.

`lp` parameter of the exponential distribution used to generate the entries of a matrix of shape parameters of the gamma distributions which generate signatures.

le parameter of the exponential distribution used to generate the entries of a matrix of shape parameters of the gamma distributions which generate exposures.

var.ap variance of the gamma distribution used to generate proposals for shape parameters of signatures

var.ae variance of the gamma distribution used to generate proposals for shape parameters of exposures

testing_burn number of burning iterations of the Gibbs sampler used to estimate the number of signatures in data. Corresponds to R0 at Algorithm 1 on signeR paper.

testing_eval number of iterations of the Gibbs sampler used to estimate the number of signatures in data. Corresponds to R2 at Algorithm 1 on signeR paper.

EM_eval number of samples generated at each iteration of the EM algorithm. Corresponds to R1 at Algorithm 1 on signeR paper.

main_burn number of burning iterations of the final Gibbs sampler.

main_eval number of iterations of the final Gibbs sampler.

start NMF algorithm used to generate initial values for signatures and exposures,options: "brunet","KL","lee","Frobenius","offset","nsNMF", "ls-nmf","pe-nmf","siNMF","snmf/r" or "snmf/l".

estimate_hyper if TRUE, algorithm estimates optimal values of ap,bp,ae,be,lp,le. Start values can still be provided.

EMit_lim limit of EM iterations for the estimation of hyper-hyperparameters ap,bp,ae,be,lp,le. Default is 100. Corresponds to U at Algorithm 1 on signeR paper.

Value

signeR output is a list with the following items:

Nsign selected number of signatures.

tested_n array containing the numbers of signatures tested by the algorithm.

Test_BICs list of measured BIC values when testing different numbers of signatures.

Phat Estimated signatures, median of P samples.

Ehat Estimated exposures, median of E samples.

SignExposures SignExp object which contain the set of samples for the model parameters.

Bics measured BIC values on the final run of the sampler.

HyperParam evolution of estimated hyperparameters when testing different numbers of signatures.

Examples

```
vignette(package="signeR")
```

`SignExp` *SignExp class*

Description

Keep samples for signature and exposure matrices.

Value

Object fields:

`@Sign` array of signature matrix samples.

`@Exp` array of exposure matrix samples.

`@sigSums` Signature sums for each sample, organized by row. Normalizing factors.

`@samples` Genome sample IDs.

`@mutations` mutation names.

`@normalized` boolean variable, indicating whether Sign array has been normalized.

Index