Package 'mSigAct'

September 30, 2021

Title mutational Signature Activity analysis ('mSigAct')

Version 2.1.2

Author Steve Rozen, Alvin Wei Tian Ng, Arnoud Boot, Nanhai Jiang

Maintainer Steve Rozen <steverozen@gmail.com>

Description Analyze the ``activities'' of mutational signatures in one or more mutational spectra. 'mSigAct' stands for mutational Signature Activity. mSigAct uses a maximum likelihood approach to estimate (conservatively) whether there is evidence that a particular set of mutational signatures is present in a spectrum. It can also determine a *minimal* subset of signatures needed to plausibly reconstruct an observed spectrum. This sparse assign signatures functionality is *deliberately biased* toward using as few signatures as possible. There is also functionality to do a maximum a posteriori estimate of signature activity, which makes use of information on the proportion of tumors in a given type that have a particular signature combined with the likelihood that a particular combination of signatures generated an observed spectrum.

License GPL-3

URL https://github.com/steverozen/mSigAct

BugReports https://github.com/steverozen/mSigAct/issues

Encoding UTF-8

Language en-US

Depends R (>= 4.0),

RoxygenNote 7.1.2

VignetteBuilder knitr

biocViews

Imports dplyr,
    ICAMS (>= 2.3.5.9002),
    ICAMSxtra (>= 0.0.3.9999),
    lsa,
    nloptr,
    PCAWG7 (>= 0.1.0.9003),
    philentropy,
    quadprog,
    stats,
    sets,
    tibble,
    utils

1

Remotes github::steverozen/ICAMS@master,
    github::steverozen/ICAMSxtra@master,
    github::steverozen/PCAWG7@master

Suggests BSgenome.Hsapiens.1000genomes.hs37d5,
    devtools,
    htmlwidgets,
    knitr,
    profvis,
    rmarkdown,
    testthat (>= 2.1.0),
    usethis

R topics documented:

CancerTypes   *Return a character vector of some common cancer types*

Description

Return a character vector of some common cancer types

Usage

```
CancerTypes()
```

Examples

```
cancer.types <- CancerTypes()
```

---

cossim   *Cosine similarity with useful argument types*

Description

Calls [cosine](#).

Usage

```
cossim(v1, v2)
```

Arguments

v1	A vector or single-column matrix
v2	A vector or single-column matrix

Examples

```
spectrum <- PCAWG7::spectra$PCAWG$SBS96[, 1, drop = FALSE]
SBS96.sigs <- PCAWG7::signature$genome$SBS96
exposure <- PCAWG7::exposure$PCAWG$SBS96[, 1, drop = FALSE]
reconstructed.spectrum <- ReconstructSpectrum(sigs = SBS96.sigs,
                                              exp = exposure,
                                              use.sig.names = TRUE)
cosine <- cossim(spectrum, reconstructed.spectrum)
```

DefaultManyOpts   *Set default options for many functions, especially* [nloptr](#)

Description

Set default options for many functions, especially [nloptr](#)

Usage

```
DefaultManyOpts(likelihood.dist = "multinom")
```

Arguments

likelihood.dist

        The probability distribution used to calculate the likelihood, can be either "multinom" (multinomial distribution) or "neg.binom" (negative binomial distribution).

Value

A list with the following elements

global.opts A sub-list with several options for [nloptr](), q.v., for the global optimization phase.

local.opts A sub-list with several options for [nloptr](), q.v., for the local optimization phase.

nbinom.size Only appearing if `likelihood.dist = "neg.binom"`. The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See [NegBinomial]().

trace If > 0 print progress messages.

global_eval_f The objective function for the global optimization phase.

local_eval_f The objective function for the local optimization phase.

local_eval_g_ineq The inequality constraint function for the local optimization phase.

likelihood.dist The probability distribution used to calculate the likelihood.

Examples

```
my.opts <- DefaultManyOpts()
my.opts$trace <- 10
```

ExposureProportions	*Return the proportions of tumors of a given cancer type that have a particular signature*

Description

Return the proportions of tumors of a given cancer type that have a particular signature

Usage

```
ExposureProportions(
  mutation.type,
  cancer.type,
  all.sigs = NULL,
  drop.sigs.no.info = TRUE,
  must.include = character(),
  must.include.prop = 0.1
)
```

Arguments

mutation.type   A character string, one of "SBS96", "SBS192", "ID", "DBS78".

cancer.type   A character string. For some common cancer types, see [CancerTypes]() for more details.

all.sigs   An optional matrix of known signatures, with column names being signatures ids. Only used to drop signatures not present in `all.sigs`.

drop.sigs.no.info
                 If TRUE, drop signatures not present in the column names of `all.sigs`.

must.include   A character vector of signature IDs that must be included, even if they have not previously been observed in that cancer type. The associated proportion is specified by `must.include.prop`.

must.include.prop

> The value used for the expected proportion of signatures in must.include but not previously observed in the given cancer.type.

Value

A numerical vector of the proportion of tumors of type cancer.type with each signature for those signatures observed in cancer.type. The names are the signature ids.

Examples

```
cancer.types <- CancerTypes()
cancer.types
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                 cancer.type = "Lung-AdenoCA")
```

LLHSpectrumMultinom   *Likelihood that 1 observed spectrum was generated from a vector of expected mutation counts using multinomial distribution*

Description

Likelihood that 1 observed spectrum was generated from a vector of expected mutation counts using multinomial distribution

Usage

```
LLHSpectrumMultinom(spectrum, expected.counts, verbose = FALSE)
```

Arguments

spectrum   An observed spectrum (a numeric vector)

expected.counts

> A vector of expected mutation counts, one expected count for each mutation type. We want to know the likelihood that this model generated the observed spectrum, assuming each mutational type generates counts according to a multinomial distribution with the given expected.counts (argument prob to Multinom).

verbose   If TRUE print messages under some circumstances.

Value

log(likelihood(spectrum | expected.counts)), or, in more detail, the multinomial likelihood that each element of the spectrum (i.e., the count for each mutation type e.g. ACT > AAT) was generated from the expected count for that mutation type using multinomial distribution.

PossibleArtifacts	*Return a character vector of the IDs of possible SBS96 signature artifacts.*

Description

Return a character vector of the IDs of possible SBS96 signature artifacts.

Usage

```
PossibleArtifacts()
```

Examples

```
artifact.sigs <- PossibleArtifacts()
```

RareSignatures	*Return a character vector of the IDs of rare SBS96 signatures.*

Description

Return a character vector of the IDs of rare SBS96 signatures.

Usage

```
RareSignatures()
```

ReconstructSpectrum	*Given signatures (sigs) and exposures (exp), return a spectrum or spectra*

Description

Given signatures (sigs) and exposures (exp), return a spectrum or spectra

Usage

```
ReconstructSpectrum(sigs, exp, use.sig.names = FALSE)
```

Arguments

sigs	Signature as a matrix or data frame, with each row one mutation type (e.g. CCT > CAT or CC > TT) and each column a signature.
exp	The exposures for one or more samples as a matrix or data.frame, with each row a signature and each column a sample.
use.sig.names	If TRUE check that rownames(exp) is a subset of colnames(sigs), and use only the columns in sigs that are present in exp.

Details

Does not care or check if colSums(sigs) == 1. Error checking is minimal since this function is
called often.

Value

The matrix product sigs %*% exp after some error checking.

Examples

```
spectra <- PCAWG7::spectra$PCAWG$SBS96[, 1:2, drop = FALSE]
SBS96.sigs <- PCAWG7::signature$genome$SBS96
exposure <- PCAWG7::exposure$PCAWG$SBS96[, 1:2, drop = FALSE]
reconstructed.spectra <- ReconstructSpectrum(sigs = SBS96.sigs,
                                             exp = exposure,
                                             use.sig.names = TRUE)
```

SignaturePresenceTest   *Test whether a given signature is plausibly present in a catalog.*

Description

Test whether a given signature is plausibly present in a catalog.

Usage

```
SignaturePresenceTest(
  spectra,
  sigs,
  target.sig.index,
  m.opts = NULL,
  seed = NULL,
  mc.cores = 10
)
```

Arguments

spectra	The catalog (matrix) to analyze. This could be an [ICAMS](#) catalog or a numerical matrix.
sigs	A catalog of signatures from which to choose. This could be and [ICAMS](#) catalog or a numerical matrix.
target.sig.index	
	The index of the signature the presence of which we want to test.
m.opts	If NULL use the return from calling [DefaultManyOpts](#). For documentation see [DefaultManyOpts](#).
seed	Random seed; set this to get reproducible results. (The numerical optimization is in two phases; the first, global phase might rarely find different optima depending on the random seed.)
mc.cores	Number of cores to use. Always silently changed to 1 on Microsoft Windows.

Value

A list of test results for each sample in `spectra`. Each sublist contains the following elements:

- loglh.with: The maximum log likelihood of the reconstructed spectrum using all the signatures.

- loglh.without: The maximum log likelihood of the reconstructed spectrum without the target signature.

- statistic: Likelihood ratio test statistic.

- chisq.p: P-value of the likelihood ratio test. The null hypothesis is we can plausibly reconstruct the spectrum without the target signature.

- exp.with: The exposure using all the signatures which generates the maximum log likelihood `loglh.with`.

- exp.without: The exposure not using the target signature which generates the maximum log likelihood `loglh.without`.

Examples

```
indices <- grep("Lung-AdenoCA", colnames(PCAWG7::spectra$PCAWG$SBS96))
spectra <- PCAWG7::spectra$PCAWG$SBS96[, indices[1:2], drop = FALSE]
sigs <- PCAWG7::signature$genome$SBS96
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                 cancer.type = "Lung-AdenoCA")
sigs.to.use <- sigs[, names(sigs.prop), drop = FALSE]
# Test whether SBS17a is plausibly present in the spectra
sig.index <- which(colnames(sigs.to.use) == "SBS17a")
sig.presence.test.out <- SignaturePresenceTest(spectra = spectra,
                                               sigs = sigs.to.use,
                                               target.sig.index = sig.index,
                                               seed = 2581,
                                               mc.cores = 2)
```

    SparseAssignActivity   *Find known signatures that can most sparsely reconstruct each spectrum in a catalog.*

Description

Find known signatures that can most sparsely reconstruct each spectrum in a catalog.

Usage

```
SparseAssignActivity(
  spectra,
  sigs,
  output.dir,
  max.level = 5,
  p.thresh = 0.05,
  m.opts = DefaultManyOpts(),
  num.parallel.samples = 5,
  mc.cores.per.sample = min(20, 2^max.level),
```

```
    progress.monitor = NULL,
    seed = NULL,
    max.subsets = 1000,
    drop.low.mut.samples = TRUE
)
```

Arguments

spectra	The spectra (multiple spectra) to be reconstructed.
sigs	A numerical matrix, possibly an [ICAMS](#) catalog.
output.dir	Directory path to save the output file.
max.level	The maximum number of signatures to try removing.
p.thresh	If the p value for a better reconstruction with a set of signatures (as opposed to without that set of signatures) is > than this argument, then we can use exposures without this set.
m.opts	See [DefaultManyOpts](#).
num.parallel.samples	
	The (maximum) number of samples to run in parallel. On Microsoft Windows machines it is silently changed to 1. Each sample in turn can require multiple cores, as governed by mc.cores.per.sample.
mc.cores.per.sample	
	The maximum number of cores to use for each sample. On Microsoft Windows machines it is silently changed to 1.
progress.monitor	
	Function called at the start of each new level (number of signatures to try excluding). Must take named arguments value and detail, and no others. Designed for a [AsyncProgress](#) progress bar function.
seed	Random seed; set this to get reproducible results. (The numerical optimization is in two phases; the first, global phase might rarely find different optima depending on the random seed.)
max.subsets	This argument provides a way to heuristically limit the amount of time spent by this function. Larger values of this argument will tend to allow longer running times. The algorithm successively tries to remove all subsets of 1 signature, 2 signatures, 3 signatures, etc., down to max.level. (Not every subset is tested at each level; if a subset was already found to be necessary the algorithm does not test supersets of that subset.) If at any level the algorithm needs to test more than max.subsets this function will not proceed.
drop.low.mut.samples	
	Whether to exclude low mutation samples from the analysis. If TRUE(default), samples with SBS total mutations less than 100, DBS or ID total mutations less than 25 will be dropped.

Value

A list with the elements:

- proposed.assignment: The most sparse set of signatures that can plausibly explain spectra.
- proposed.reconstruction: The reconstruction based on sparse assignment.
- reconstruction.distances: Various distances and similarities between spectra and proposed.reconstructio
- all.tested: All tested possible ways to reconstruct each sample in spectra.

- alt.solutions: A tibble showing all the alternative solutions that are statistically as good as the proposed.assignment that can plausibly reconstruct spectra.

- time.for.assignment: Value from system.time for running SparseAssignActivity for each sample in spectra.

- error.messages: Only appearing if there are errors running SparseAssignActivity.

The elements proposed.assignment, proposed.reconstruction, reconstruction.distances, all.tested, time.for.assignment will be NULL if the algorithm could not find the optimal reconstruction or there are errors coming out for **all** samples.

Examples

```
## Not run:
# This is a long running example unless parallel computing is supported on your machine
indices <- grep("Lung-AdenoCA", colnames(PCAWG7::spectra$PCAWG$SBS96))
spectra <- PCAWG7::spectra$PCAWG$SBS96[, indices[1:2], drop = FALSE]
sigs <- PCAWG7::signature$genome$SBS96
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                 cancer.type = "Lung-AdenoCA")
sigs.to.use <- sigs[, names(sigs.prop), drop = FALSE]
sparse.out <- SparseAssignActivity(spectra = spectra,
                                   sigs = sigs.to.use,
                                   output.dir = file.path(tempdir(), "Lung-AdenoCA"),
                                   max.level = ncol(sigs.to.use) - 1,
                                   p.thresh = 0.05 / ncol(spectra),
                                   num.parallel.samples = 2,
                                   mc.cores.per.sample = 30,
                                   seed = 2561)

## End(Not run)
```

Index