

Package ‘mSigAct’

July 15, 2021

Title mutational Signature Activity analysis ('mSigAct')

Version 2.1.1.9007

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

LazyData true

Language en-US

Depends R (>= 4.0),

RoxygenNote 7.1.1

VignetteBuilder knitr

biocViews

Imports dplyr,
ICAMS (>= 2.3.5.9002),
ICAMSextra (>= 0.0.3.9999),
lsa,
nloptr,
PCAWG7 (>= 0.1.0.9003),
philentropy,
quadprog,
rlang,
stats,

sets,
tibble,
utils

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:

AddSigActivity	3
CancerTypes	4
cossim	4
DefaultManyOpts	5
ExposureProportions	5
g_ineq_for_ObjFnBinomMaxLH2	6
g_ineq_for_ObjFnMultinomMaxLH	7
LLHSpectrumMAP	7
LLHSpectrumMultinom	8
LLHSpectrumNegBinom	9
LLHSpectrumNegBinomDebug	10
MAPAssignActivity	10
MAPAssignActivity1	12
ObjFnBinomMaxLHMustRound	14
ObjFnBinomMaxLHNoRoundOK	15
ObjFnBinomMaxLHRound	16
ObjFnMultinomMaxLH	16
OneMAPAssignTest	17
OptimizeExposure	18
OptimizeExposureQP	19
OptimizeExposureQPBootstrap	19
PCAWGMAPTest	20
PossibleArtifacts	21
RareSignatures	22
ReconstructSpectrum	22
ShowSigActivity	23
SignaturePresenceTest	24
SignaturePresenceTest1	24
SparseAssignActivity	25

Index

AddSigActivity	<i>Add contributing signature activity information for multiple spectra</i>
----------------	---

Description

Add contributing signature activity information for multiple spectra

Usage

```
AddSigActivity(spectra, exposure, sigs, sigs.presence.prop, nbinom.size = 5)
```

Arguments

spectra	The spectra (multiple spectra) to be reconstructed.
exposure	Exposures as a numerical matrix (or data.frame) with signatures in rows and samples in columns. Rownames are taken as the signature names and column names are taken as the sample IDs.
sigs	The signatures with which we are trying to reconstruct spectra. A numerical matrix, possibly an ICAMS catalog. The column names of sigs should be a superset of row names of exposure.
sigs.presence.prop	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being a subset of colnames(sigs). See ExposureProportions for more details.
nbinom.size	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .

Details

This function calls [ReconstructSpectrum](#) and [LLHSpectrumNegBinom](#).

Value

A list of lists containing output for each sample in spectra.

Each sublist has the following elements:

- `original.spect`: The original spectrum with total mutation counts added to its column name. An additional attribute "exposure" from `exposure` is also added.
- `reconstructed.spect`: The reconstructed spectrum using `sigs` and `exposure`. Its column name has the total mutation counts and cosine similarity with the original spectrum.
- `contributing.sigs`: The contributing signatures to the original spectrum. The column names of each contributing signature has mutation counts attributed to this signature, its contribution proportion and proposed etiology(if the etiology is unknown, then will be blank.)
- `distances`: Various distances and similarities between the original spectrum and `reconstructed.spect`.

Remark

The column names of `spectra` should be the same as the column name of `exposure`.

Examples

```
spectra <- PCAWG7::spectra$PCAWG$SBS96[, 1:2, drop = FALSE]
exposure <- PCAWG7::exposure$PCAWG$SBS96[, 1:2, drop = FALSE]
sigs <- PCAWG7::signature$genome$SBS96
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                cancer.type = "Biliary-AdenoCA")
retval <- AddSigActivity(spectra, exposure, sigs, sigs.prop)
```

CancerTypes	<i>Return a character vector of some common cancer types</i>
-------------	--

Description

Return a character vector of some common cancer types

Usage

```
CancerTypes()
```

cossim	<i>Cosine similarity with useful argument types..</i>
--------	---

Description

Calls [cosine](#).

Usage

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

Arguments

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

DefaultManyOpts	<i>Set default options for many functions, especially nloptr.</i>
-----------------	---

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 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.

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

<code>mutation.type</code>	A character string, one of "SBS96", "SBS192", "ID", "DBS78".
<code>cancer.type</code>	A character string. For some common cancer types, see CancerTypes for more details.
<code>all.sigs</code>	An optional matrix of known signatures, with column names being signature ids. Only used to drop signatures not present in <code>all.sigs</code> .
<code>drop.sigs.no.info</code>	If TRUE, drop signatures not present in the column names of <code>all.sigs</code> .
<code>must.include</code>	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 <code>must.include.prop</code> .
<code>must.include.prop</code>	The value used for the expected proportion of signatures in <code>must.include</code> but not previously observed in the given <code>cancer.type</code> .

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

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

`g_ineq_for_ObjFnBinomMaxLH2`

Function to constrain the sum of estimated exposures to the number of mutations in the spectrum.

Description

See [nloptr](#) to understand how this function is used.

Usage

```
g_ineq_for_ObjFnBinomMaxLH2(exp, spectrum, sigs, nbinom.size)
```

Arguments

<code>exp</code>	A numeric vector of exposures.
<code>spectrum</code>	The observed spectrum we are trying to reconstruct.
<code>sigs</code>	The signatures with which we are trying to reconstruct the spectrum. (Ignored in this function but used by nloptr .)
<code>nbinom.size</code>	Dispersion parameter. (Ignored in this function but used by nloptr .)

g_ineq_for_ObjFnMultinomMaxLH

Function to constrain the sum of estimated exposures to the number of mutations in the spectrum.

Description

See [nloptr](#) to understand how this function is used.

Usage

```
g_ineq_for_ObjFnMultinomMaxLH(exp, spectrum, sigs)
```

Arguments

exp	A numeric vector of exposures.
spectrum	The observed spectrum we are trying to reconstruct.
sigs	The signatures with which we are trying to reconstruct the spectrum. (Ignored in this function but used by nloptr .)

LLHSpectrumMAP

Likelihood that 1 observed spectrum was generated from a vector of expected mutation counts using prior information of the signature presence proportions

Description

Likelihood that 1 observed spectrum was generated from a vector of expected mutation counts using prior information of the signature presence proportions

Usage

```
LLHSpectrumMAP (
  spectrum,
  expected.counts,
  nbinom.size,
  model,
  sigs.presence.prop,
  likelihood.dist = "multinom",
  verbose = FALSE
)
```

Arguments

<code>spectrum</code>	An observed spectrum (a numeric vector).
<code>expected.counts</code>	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 types generates counts according to a probability distribution specified by <code>likelihood.dist</code> with the given <code>expected.counts</code> . See <code>LLHSpectrumMultinom</code> and <code>LLHSpectrumNegBinom</code> for more details.
<code>nbinom.size</code>	Only needed when <code>likelihood.dist = "neg.binom"</code>. The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .
<code>model</code>	Names of sigs present in the MAP exposure. Do not use indices.
<code>sigs.presence.prop</code>	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being a superset of <code>model</code> .
<code>likelihood.dist</code>	The probability distribution used to calculate the likelihood, can be either "multinom" (multinomial distribution) or "neg.binom" (negative binomial distribution).
<code>verbose</code>	If TRUE print messages under some circumstances.

Value

`log(likelihood(spectrum | expected.counts)) + log(probability(model | sigs.presence.prop))`, or, in more detail, the sum of the negative binomial likelihoods 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 plus the probability of the signature model used in the reconstruction given the prior `sigs.presence.prop`.

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

<code>spectrum</code>	An observed spectrum (a numeric vector)
<code>expected.counts</code>	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 <code>expected.counts</code> (argument <code>prob</code> to Multinom).
<code>verbose</code>	If <code>TRUE</code> 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.

LLHSpectrumNegBinom

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

Description

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

Usage

```
LLHSpectrumNegBinom(spectrum, expected.counts, nbinom.size, verbose = FALSE)
```

Arguments

<code>spectrum</code>	An observed spectrum (a numeric vector)
<code>expected.counts</code>	A vector of (integer) 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 types generates counts according to a negative binomial distribution with the given <code>expected.counts</code> (argument <code>mu</code> to NegBinomial) and dispersion parameter <code>nbinom.size</code> .
<code>nbinom.size</code>	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .
<code>verbose</code>	If <code>TRUE</code> print messages under some circumstances.

Value

`log(likelihood(spectrum | expected.counts))`, or, in more detail, the sum of the negative binomial likelihoods 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.

LLHSpectrumNegBinomDebug

A verbose version of [LLHSpectrumNegBinom](#) for testing

Description

We use a separate function so as not to slow down the heavily used [LLHSpectrumNegBinom](#) and to provide more information in the output

Usage

```
LLHSpectrumNegBinomDebug (
  spectrum,
  expected.counts,
  nbinom.size,
  verbose = FALSE
)
```

Arguments

spectrum	An observed spectrum (a numeric vector)
expected.counts	A vector of (integer) 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 types generates counts according to a negative binomial distribution with the given <code>expected.counts</code> (argument <code>mu</code> to NegBinomial) and dispersion parameter <code>nbinom.size</code> .
nbinom.size	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .
verbose	If TRUE print messages under some circumstances.

Value

A [tibble](#) with self-explanatory columns and rows.

MAPAssignActivity *Find Maximum A Posteriori (MAP) assignment of signature exposures that explain multiple spectra*

Description

This function also can do sparse assignment by specifying `use.sparse.assign = TRUE`.

Usage

```
MAPAssignActivity(
  spectra,
  sigs,
  sigs.presence.prop,
  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,
  use.sparse.assign = FALSE
)
```

Arguments

<code>spectra</code>	The spectra (multiple spectra) to be reconstructed.
<code>sigs</code>	A numerical matrix, possibly an ICAMS catalog.
<code>sigs.presence.prop</code>	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being a subset of <code>colnames(sigs)</code> . See ExposureProportions for more details.
<code>output.dir</code>	Directory path to save the output file.
<code>max.level</code>	The maximum number of signatures to try removing.
<code>p.thresh</code>	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.
<code>m.opts</code>	See DefaultManyOpts .
<code>num.parallel.samples</code>	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 <code>mc.cores.per.sample</code> .
<code>mc.cores.per.sample</code>	The maximum number of cores to use for each sample. On Microsoft Windows machines it is silently changed to 1.
<code>progress.monitor</code>	Function called at the start of each new level (number of signatures to try excluding). Must take named arguments <code>value</code> and <code>detail</code> , and no others. Designed for a AsyncProgress progress bar function.
<code>seed</code>	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.)
<code>max.subsets</code>	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 <code>max.level</code> . (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.

`use.sparse.assign`

Whether to use sparse assignment. If `TRUE`, arguments designed for Maximum A Posteriori assignment such as `sigs.presence.prop` will be ignored.

Value

A list with the elements:

- `proposed.assignment`: Proposed signature assignment for spectra with the highest MAP found. If `use.sparse.assign = TRUE`, this will be the most sparse set of signatures that can plausibly explain spectra.
- `proposed.reconstruction`: Proposed reconstruction of spectra based on MAP. If `use.sparse.assign = TRUE`, this will be the reconstruction based on sparse assignment.
- `reconstruction.distances`: Various distances and similarities between spectra and `proposed.reconstruction`.
- `error.messages`: Only appearing if there are errors running MAPAssignActivity.
- `results.details`: Detailed results for each sample in spectra.

The elements `proposed.assignment`, `proposed.reconstruction`, `reconstruction.distances` 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")
MAP.out <- MAPAssignActivity(spectra = spectra,
                             sigs = sigs,
                             sigs.presence.prop = sigs.prop,
                             output.dir = file.path(tempdir(), "Lung-AdenoCA"),
                             max.level = length(sigs.prop) - 1,
                             p.thresh = 0.05 / ncol(spectra),
                             num.parallel.samples = 2,
                             mc.cores.per.sample = 10)

## End(Not run)
```

MAPAssignActivity1 *Find a Maximum A Posteriori (MAP) assignment of signature exposures that explain one spectrum.*

Description

This function also can do sparse assignment by specifying `use.sparse.assign = TRUE`.

Usage

```
MAPAssignActivity1(
  spect,
  sigs,
  sigs.presence.prop,
  max.level = 5,
  p.thresh = 0.05,
  m.opts = DefaultManyOpts(),
  max.mc.cores = min(20, 2^max.level),
  progress.monitor = NULL,
  seed = NULL,
  max.subsets = 1000,
  use.sparse.assign = FALSE
)
```

Arguments

<code>spect</code>	A single spectrum.
<code>sigs</code>	A numerical matrix, possibly an ICAMS catalog.
<code>sigs.presence.prop</code>	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being a subset of <code>colnames(sigs)</code> . See ExposureProportions for more details.
<code>max.level</code>	The maximum number of signatures to try removing.
<code>p.thresh</code>	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.
<code>m.opts</code>	See DefaultManyOpts .
<code>max.mc.cores</code>	The maximum number of cores to use. On Microsoft Windows machines it is silently changed to 1.
<code>progress.monitor</code>	Function called at the start of each new level (number of signatures to try excluding). Must take named arguments <code>value</code> and <code>detail</code> , and no others. Designed for a AsyncProgress progress bar function.
<code>seed</code>	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.)
<code>max.subsets</code>	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 <code>max.level</code> . (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 <code>max.subsets</code> this function will not proceed.
<code>use.sparse.assign</code>	Whether to use sparse assignment. If TRUE, arguments designed for Maximum A Posteriori assignment such as <code>sigs.presence.prop</code> will be ignored.

Value

A list with the elements:

- `proposed.assignment`: Proposed signature assignment for `spect` with the highest MAP found. If `use.sparse.assign = TRUE`, this will be the most sparse set of signatures that can plausibly explain `spect`.
- `proposed.reconstruction`: Reconstruction based on MAP. If `use.sparse.assign = TRUE`, this will be the reconstruction based on sparse assignment.
- `reconstruction.distances`: Various distances and similarities between `spect` and `proposed.reconstruction`.
- `all.tested`: A tibble of all the search results.
- `time.for.MAP.assign`: Value from `system.time` for running `MAPAssignActivity1`.
- `error.messages`: Only present if there were errors running `MAPAssignActivity1`.

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

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))
spect <- PCAWG7::spectra$PCAWG$SBS96[, indices[1], drop = FALSE]
sigs <- PCAWG7::signature$genome$SBS96
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                cancer.type = "Lung-AdenoCA")
MAP.out <- MAPAssignActivity1(spect = spect,
                             sigs = sigs,
                             sigs.presence.prop = sigs.prop,
                             max.level = length(sigs.prop) - 1)

## End(Not run)
```

ObjFnBinomMaxLHMustRound

A deprecated negative binomial maximum likelihood objective function.

Description

Use [ObjFnBinomMaxLHRound](#) instead.

Usage

```
ObjFnBinomMaxLHMustRound(exp, spectrum, sigs, nbinom.size)
```

Arguments

<code>exp</code>	A vector of exposures ("activities").
<code>spectrum</code>	The spectrum to assess.
<code>sigs</code>	The matrix of signatures.
<code>nbinom.size</code>	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .

Details

This function will lead to errors in some situations when the rounded reconstructed signature contains 0s for mutations classes for which the target spectrum is > 0 .

ObjFnBinomMaxLHNoRoundOK

A deprecated negative binomial maximum likelihood objective function.

Description

Use [ObjFnBinomMaxLHRound](#) instead.

Usage

```
ObjFnBinomMaxLHNoRoundOK(exp, spectrum, sigs, nbinom.size)
```

Arguments

<code>exp</code>	A vector of exposures ("activities").
<code>spectrum</code>	The spectrum to assess.
<code>sigs</code>	The matrix of signatures.
<code>nbinom.size</code>	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .

Details

This function rounds sometimes, which leads to minor differences in log likelihoods of reconstructed spectra ([LLHSpectrumNegBinom](#)) compared to the value returned by this function.

ObjFnBinomMaxLHRound

The preferred negative binomial maximum likelihood objective function.

Description

Can be used as the objective function for [MAPAssignActivity](#), [MAPAssignActivity1](#), [SparseAssignActivity](#), [SparseAssignActivity1](#), [SignaturePresenceTest](#) and [SignaturePresenceTest1](#). (Internally used by [nloptr](#).)

Usage

```
ObjFnBinomMaxLHRound(exp, spectrum, sigs, nbinom.size)
```

Arguments

exp	A vector of exposures ("activities").
spectrum	The spectrum to assess.
sigs	The matrix of signatures.
nbinom.size	The dispersion parameter for the negative binomial distribution; smaller is more dispersed. See NegBinomial .

Value

$-1 * \log(\text{likelihood}(\text{spectrum} \mid \text{reconstruction}))$

[nloptr](#) minimizes the objective function, so the lower the objective function, the better.

ObjFnMultinomMaxLH *The multinomial likelihood objective function*

Description

Can be used as the objective function for [MAPAssignActivity](#), [MAPAssignActivity1](#), [SparseAssignActivity](#), [SparseAssignActivity1](#), [SignaturePresenceTest](#) and [SignaturePresenceTest1](#). (Internally used by [nloptr](#).)

Usage

```
ObjFnMultinomMaxLH(exp, spectrum, sigs)
```

Arguments

exp	The matrix of exposures ("activities").
spectrum	The spectrum to assess.
sigs	The matrix of signatures.

Value

$-1 * \log(\text{likelihood}(\text{spectrum} \mid \text{reconstruction}))$

`nloptr` minimizes the objective function, so the lower the objective function, the better.

OneMAPAssignTest *Run one test of `MAPAssignActivity1`.*

Description

Run one test of `MAPAssignActivity1`.

Usage

```
OneMAPAssignTest (
  spect,
  reference.exp,
  cancer.type,
  mutation.type,
  exposure.mutation.type,
  max.subsets = 1000,
  max.level = 5,
  max.mc.cores = 100,
  m.opts = DefaultManyOpts(),
  out.dir = NULL,
  p.thresh,
  max.presence.proportion,
  sigs.prop = NULL,
  sigs = NULL
)
```

Arguments

<code>spect</code>	A single spectrum.
<code>reference.exp</code>	Compare the inferred exposures to this.
<code>cancer.type</code>	Character string from a fixed set indicating different cancer types, used to look up the set of signatures known in that cancer type and the proportion of cancers of that type that have the signature. TODO: provide information on how to find the allowed cancer types.
<code>mutation.type</code>	One of "SBS96", "SBS192", "ID", "DBS78".
<code>exposure.mutation.type</code>	One of "SBS96", "ID", "DBS78".
<code>max.subsets</code>	The maximum number of subsets that can be tested for removal from the set of signatures.
<code>max.level</code>	The maximum number of signatures to try removing.
<code>max.mc.cores</code>	The maximum number of cores to use. On Microsoft Windows machines it is silently changed to 1.

<code>m.opts</code>	See DefaultManyOpts .
<code>out.dir</code>	If non-NULL create this directory if necessary and put results there.
<code>p.thresh</code>	If the p value for a better reconstruction with than without a set of signatures is > than <code>p.thresh</code> , then we can use exposures without this set.
<code>max.presence.proportion</code>	The maximum value of the proportion of tumors that must have a given signature. Used so that it is possible to exclude a signature from a spectrum, e.g. perhaps all examples of tumor types have SBS5, but we want to allow a small chance that SBS5 is not present.
<code>sigs.prop</code>	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being signature identifiers. Can be the return value from ExposureProportions .
<code>sigs</code>	Matrix of signatures.

OptimizeExposure	<i>Optimize the reconstruction of a spectrum from a set of signatures.</i>
------------------	--

Description

Optimize the reconstruction of a spectrum from a set of signatures.

Usage

```
OptimizeExposure(spectrum, sigs, m.opts, ...)
```

Arguments

<code>spectrum</code>	The spectrum to be reconstructed.
<code>sigs</code>	The available signatures.
<code>m.opts</code>	Options that govern the numerical optimization. For documentation see DefaultManyOpts .
<code>...</code>	Additional arguments for <code>eval_f</code> .

Value

A list with elements

`loglh` The log likelihood of the best solution (set of exposures) found.

`exposure` The vector of exposures that generated `loglh`, i.e. the number of mutations ascribed to each signature.

`objective` The final value of the objective function.

`solution` The optimum exposures. Deprecated.

`warnings` A character vector of warnings.

`global.search.diagnostics` Diagnostics from [nloptr](#).

`local.search.diagnostics` Diagnostics from [nloptr](#).

OptimizeExposureQP *Quadratic programming optimization of signature activities*

Description

Quadratic programming optimization of signature activities

Usage

```
OptimizeExposureQP(spectrum, signatures)
```

Arguments

spectrum	Mutational signature spectrum as a numeric vector or single column data frame or matrix.
signatures	Matrix or data frame of signatures from which reconstruct spectrum. Rows are mutation types and columns are signatures. Should have column names for interpretable results. Cannot be a vector because the column names are needed.

Value

A vector of exposures with names being the colnames from signatures. Code adapted from `SignatureEstimation::decomposeQP`.

OptimizeExposureQPBootstrap

Bootstrap [OptimizeExposureQP](#) and filter exposures by confidence intervals

Description

Bootstrap [OptimizeExposureQP](#) and filter exposures by confidence intervals

Usage

```
OptimizeExposureQPBootstrap(
  spectrum,
  signatures,
  num.replicates = 10000,
  conf.int = 0.95,
  mc.cores = 10,
  seed = NULL
)
```

Arguments

<code>spectrum</code>	Mutational signature spectrum as a numeric vector or single column data frame or matrix.
<code>signatures</code>	Matrix or data frame of signatures from which reconstruct <code>spectrum</code> . Rows are mutation types and columns are signatures. Should have column names for interpretable results. Cannot be a vector because the column names are needed.
<code>num.replicates</code>	Number of bootstrap replicates.
<code>conf.int</code>	Discard signatures with <code>conf.int</code> that overlaps 0.
<code>mc.cores</code>	The maximum number of cores to use. On MS Windows machines it defaults to 1.
<code>seed</code>	Random seed; set this to get reproducible results.
	<code>#' @return</code> A list with elements
	<code>exposure</code> The vector of exposures that generated <code>loglh</code> , i.e. the number of mutations ascribed to each signature. The names of <code>exposure</code> are a subset of the <code>colnames(signatures)</code> .
	<code>euclidean.dist</code> The final value of the objective function.
	<code>cosine.sim</code> The cosine similarity between <code>spectrum</code> and the reconstruction based on <code>signatures</code> .
	If the spectrum has 0 mutations, no bootstrapping is done, and in the return value all signatures have 0 exposures, <code>euclidean.dist</code> is 0, and <code>cosine.sim</code> is NaN.

PCAWGMAPTest

Run [MAPAssignActivity1](#) on one sample from the PCAWG platinum data set.

Description

Run [MAPAssignActivity1](#) on one sample from the PCAWG platinum data set.

Run [MAPAssignActivity1](#) on one sample from the PCAWG platinum data set with artifact signatures removed.

Usage

```
PCAWGMAPTest (
  cancer.type,
  sample.index,
  mutation.type,
  max.level = 5,
  max.mc.cores,
  out.dir = NULL,
  p.thresh = 0.01,
  m.opts = DefaultManyOpts(),
  max.presence.proportion = 0.99,
  sigs.prop = NULL
)
```

```

PCAWGMAPTest (
  cancer.type,
  sample.index,
  mutation.type,
  max.level = 5,
  max.mc.cores,
  out.dir = NULL,
  p.thresh = 0.01,
  m.opts = DefaultManyOpts(),
  max.presence.proportion = 0.99,
  sigs.prop = NULL
)

```

Arguments

<code>cancer.type</code>	A cancer type from the PCAWG exposures matrix.
<code>sample.index</code>	The index of the sample within the exposures matrix.
<code>mutation.type</code>	One of "SBS96", "SBS192", "ID", "DBS78"
<code>max.level</code>	The maximum number of signatures to try removing.
<code>max.mc.cores</code>	The maximum number of cores to use. On Microsoft Windows machines it is silently changed to 1.
<code>out.dir</code>	If non-NULL create this directory if necessary and put results there.
<code>p.thresh</code>	If the p value for a better reconstruction with than without a set of signatures is > than <code>p.thresh</code> , then we can use exposures without this set.
<code>m.opts</code>	See DefaultManyOpts .
<code>max.presence.proportion</code>	The maximum value of the proportion of tumors that must have a given signature. Used so that it is possible to exclude a signature from a spectrum, e.g. perhaps all examples of tumor types have SBS5, but we want to allow a small chance that SBS5 is not present.
<code>sigs.prop</code>	The proportions of samples that contain each signature. A numerical vector (values between 0 and 1), with names being signature identifiers. Can be the return value from ExposureProportions .

Value

See [OneMAPAssignTest](#).

A list with two elements, each the result for one call to [OneMAPAssignTest](#).

PossibleArtifacts	<i>Return a character vector of the IDs of possible SBS96 signature artifacts.</i>
-------------------	--

Description

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

Usage

```
PossibleArtifacts()
```

RareSignatures	<i>Return a character vector of the IDs of rare SBS96 signatures.</i>
----------------	---

Description

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

Usage

```
RareSignatures()
```

ReconstructSpectrum	<i>Given signatures (sigs) and exposures (exp), return a spectrum or spectra</i>
---------------------	--

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 (g.e. 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.

ShowSigActivity	<i>Show signature activity from the output generated by AddSigActivity</i>
-----------------	--

Description

Show signature activity from the output generated by AddSigActivity

Usage

```
ShowSigActivity(
  list.of.sig.activity,
  output.dir,
  base.filename = NULL,
  plot.all.samples.in.one.pdf = TRUE,
  plot.exposure.proportion = FALSE,
  ...
)
```

Arguments

<code>list.of.sig.activity</code>	A list of contributing signature activity information for multiple spectra. See the return value of AddSigActivity for more details.
<code>output.dir</code>	The directory to save the results. Create this directory if it does not exist.
<code>base.filename</code>	Optional. <code>base.filename</code> will be appended to the start of the names of files generated inside <code>output.dir</code> .
<code>plot.all.samples.in.one.pdf</code>	Whether to plot all the signature activity information within one PDF. Default is TRUE. If FALSE, then plot one PDF for each sample.
<code>plot.exposure.proportion</code>	Whether to plot exposure proportions rather than counts.
<code>...</code>	Other arguments passed to PlotCatalogToPdf .

Examples

```
spectra <- PCAWG7::spectra$PCAWG$SBS96[, 1:2, drop = FALSE]
exposure <- PCAWG7::exposure$PCAWG$SBS96[, 1:2, drop = FALSE]
sigs <- PCAWG7::signature$genome$SBS96
sigs.prop <- ExposureProportions(mutation.type = "SBS96",
                                cancer.type = "Biliary-AdenoCA")
retval <- AddSigActivity(spectra, exposure, sigs, sigs.prop)
ShowSigActivity(retval, output.dir = file.path(tempdir(), "SBS96"),
  base.filename = "Biliary-AdenoCA")
```

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,
    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 an 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 .
mc.cores	Number of cores to use. Always silently changed to 1 on Microsoft Windows.

SignaturePresenceTest1

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

Description

For backward compatibility. See also [AnySigSubsetPresent](#).

Usage

```
SignaturePresenceTest1(spectrum, sigs, target.sig.index, m.opts)
```

Arguments

spectrum	The spectrum to analyze.
sigs	A catalog of signatures from which to choose.
target.sig.index	The index of the signature the presence of which we want to test.
m.opts	For documentation see DefaultManyOpts .

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),
  seed = NULL
)
```

Arguments

spectra	The spectra (multiple spectra) to be reconstructed.
sigs	The known signatures to use in reconstruction.
output.dir	Directory path to save the output file.
max.level	The largest number of signatures to consider discarding in the reconstruction.
p.thresh	The maximum p value based on which it is decided to retain a signature in a reconstruction.
m.opts	For documentation see DefaultManyOpts .
num.parallel.samples	The (maximum) number of samples to run in parallel; each sample in turn can require multiple cores, as governed by <code>mc.cores.per.sample</code> .
mc.cores.per.sample	The maximum number of cores to use for each sample. On Microsoft Windows machines it is silently changed to 1.
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.)

Value

A list with the elements:

- `proposed.assignment`: Proposed sparse assignment for `spectra`.
- `proposed.reconstruction`: Proposed reconstruction of `spectra` based on sparse assignment.
- `reconstruction.distances`: Various distances and similarities between `spectra` and `proposed.reconstruction`.

Examples

[illegible]

Index

AddSigActivity, [3](#), [23](#)
AnySigSubsetPresent, [24](#)
AsyncProgress, [11](#), [13](#)

CancerTypes, [4](#), [6](#)
cosine, [4](#)
cossim, [4](#)

DefaultManyOpts, [5](#), [11](#), [13](#), [18](#), [21](#), [24](#), [25](#)

ExposureProportions, [3](#), [5](#), [11](#), [13](#), [18](#),
[21](#)

g_ineq_for_ObjFnBinomMaxLH2, [6](#)
g_ineq_for_ObjFnMultinomMaxLH, [7](#)

ICAMS, [3](#), [11](#), [13](#), [24](#)

LLHSpectrumMAP, [7](#)
LLHSpectrumMultinom, [8](#)
LLHSpectrumNegBinom, [3](#), [9](#), [10](#), [15](#)
LLHSpectrumNegBinomDebug, [10](#)

MAPAssignActivity, [10](#), [16](#)
MAPAssignActivity1, [12](#), [16](#), [17](#), [20](#)
Multinom, [9](#)

NegBinomial, [3](#), [5](#), [8–10](#), [15](#), [16](#)
nloptr, [5–7](#), [16–18](#)

ObjFnBinomMaxLHMustRound, [14](#)
ObjFnBinomMaxLHNoRoundOK, [15](#)
ObjFnBinomMaxLHRound, [14](#), [15](#), [16](#)
ObjFnMultinomMaxLH, [16](#)
OneMAPAssignTest, [17](#), [21](#)
OptimizeExposure, [18](#)
OptimizeExposureQP, [19](#), [19](#)
OptimizeExposureQPBootstrap, [19](#)

PCAWGMAPTest, [20](#)
PlotCatalogToPdf, [23](#)
PossibleArtifacts, [21](#)

RareSignatures, [22](#)
ReconstructSpectrum, [3](#), [22](#)

ShowSigActivity, [23](#)
SignaturePresenceTest, [16](#), [24](#)
SignaturePresenceTest1, [16](#), [24](#)
SparseAssignActivity, [16](#), [25](#)
SparseAssignActivity1, [16](#)

tibble, [10](#)