Package 'mSigAct'

October 31, 2021

Title Mutational Signature Activity Analysis ('mSigAct')

Version 2.1.3.9000

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

biocViews

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

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

Suggests BSgenome.Hsapiens.1000genomes.hs37d5,
devtools,
htmlwidgets,
ipc,
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.

`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,
  drop.low.mut.samples = TRUE
)
```

Arguments

`spectra`	The spectra (multiple spectra) to be reconstructed.
`sigs`	A numerical matrix, possibly an `ICAMS` catalog.
`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.
`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.

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.

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: 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.
- 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 MAPAssignActivity1 for each sample in spectra.
- error.messages: Only appearing if there are errors running MAPAssignActivity.

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")
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)
```

`PlotExposure` *Plot exposures in multiple plots each with a manageable number of samples*

Description

Plot exposures in multiple plots each with a manageable number of samples

Usage

```
PlotExposure(
  exposure,
  samples.per.line = 30,
  plot.proportion = FALSE,
  xlim = NULL,
  ylim = NULL,
  legend.x = NULL,
  legend.y = NULL,
  cex.legend = 0.9,
  cex.yaxis = 1,
  cex.xaxis = NULL,
  plot.sample.names = TRUE,
  yaxis.labels = NULL,
  ...
)
```

Arguments

`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. If you want `exposure` sorted from largest to smallest, use `SortExposure`. Do not use column names that start with multiple underscores. The exposures will often be mutation counts, but could also be e.g. mutations per megabase.
`samples.per.line`	Number of samples to show in each plot.
`plot.proportion`	Plot exposure proportions rather than counts.

`xlim, ylim` Limits for the x and y axis. If `NULL`(default), the function tries to do something reasonable.

`legend.x, legend.y`
The x and y co-ordinates to be used to position the legend.

`cex.legend` A numerical value giving the amount by which legend plotting text and symbols should be magnified relative to the default.

`cex.yaxis` A numerical value giving the amount by which y axis values should be magnified relative to the default.

`cex.xaxis` A numerical value giving the amount by which x axis values should be magnified relative to the default. If `NULL`(default), the function tries to do something reasonable.

`plot.sample.names`
Whether to plot sample names below the x axis. Default is TRUE.

`yaxis.labels` User defined y axis labels to be plotted. If `NULL`(default), the function tries to do something reasonable.

`...` Other arguments passed to `barplot`. If `ylab` is not included, it defaults to a value depending on `plot.proportion`. If `col` is not supplied the function tries to do something reasonable.

Value

An **invisible** list whose first element is a logic value indicating whether the plot is successful. The second element is a numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph.

Examples

```
file <- system.file("extdata",
                    "Liver-HCC.exposure.csv",
                    package = "mSigAct")
exposure <- ReadExposure(file)
old.par <- par(mar = c(8, 5, 1, 1))
PlotExposure(exposure[, 1:30], main = "Liver-HCC exposure", cex.yaxis = 0.8,
             plot.proportion = TRUE)
par(old.par)
```

PlotExposureToPdf *Plot exposures in multiple plots each with a manageable number of samples to PDF*

Description

Plot exposures in multiple plots each with a manageable number of samples to PDF

Usage

```
PlotExposureToPdf(
  exposure,
  file,
  mfrow = c(2, 1),
```

```
  mar = c(6, 4, 3, 2),
  oma = c(3, 2, 0, 2),
  samples.per.line = 30,
  plot.proportion = FALSE,
  xlim = NULL,
  ylim = NULL,
  legend.x = NULL,
  legend.y = NULL,
  cex.legend = 0.9,
  cex.yaxis = 1,
  cex.xaxis = NULL,
  plot.sample.names = TRUE,
  yaxis.labels = NULL,
  width = 8.2677,
  height = 11.6929,
  ...
)
```

Arguments

`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. If you want `exposure` sorted from largest to smallest, use `SortExposure`. Do not use column names that start with multiple underscores. The exposures will often be mutation counts, but could also be e.g. mutations per megabase.
`file`	The name of the PDF file to be produced.
`mfrow`	A vector of the form `c(nr,nc)`. Subsequent figures will be drawn in an `nr`-by-`nc` array on the device by rows.
`mar`	A numerical vector of the form `c(bottom,left,top,right)` which gives the number of lines of margin to be specified on the four sides of the plot.
`oma`	A vector of the form `c(bottom,left,top,right)` giving the size of the outer margins in lines of text.
`samples.per.line`	Number of samples to show in each plot.
`plot.proportion`	Plot exposure proportions rather than counts.
`xlim, ylim`	Limits for the x and y axis. If `NULL`(default), the function tries to do something reasonable.
`legend.x, legend.y`	The x and y co-ordinates to be used to position the legend.
`cex.legend`	A numerical value giving the amount by which legend plotting text and symbols should be magnified relative to the default.
`cex.yaxis`	A numerical value giving the amount by which y axis values should be magnified relative to the default.
`cex.xaxis`	A numerical value giving the amount by which x axis values should be magnified relative to the default. If `NULL`(default), the function tries to do something reasonable.
`plot.sample.names`	Whether to plot sample names below the x axis. Default is TRUE.

`yaxis.labels` User defined y axis labels to be plotted. If `NULL`(default), the function tries to do something reasonable.

`width, height`
The width and height of the graphics region in inches.

`...` Other arguments passed to `barplot`. If `ylab` is not included, it defaults to a value depending on `plot.proportion`. If `col` is not supplied the function tries to do something reasonable.

Value

An **invisible** list whose first element is a logic value indicating whether the plot is successful. The second element is a numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph.

Examples

```
file <- system.file("extdata",
                    "Liver-HCC.exposure.csv",
                    package = "mSigAct")
exposure <- ReadExposure(file)
PlotExposureToPdf(exposure, file = file.path(tempdir(), "Liver-HCC.exposure.pdf"),
                  cex.yaxis = 0.8, plot.proportion = TRUE)
```

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()
```

ReadExposure *Read an exposure matrix from a file*

Description

Read an exposure matrix from a file

Usage

```
ReadExposure(file, check.names = FALSE)
```

Arguments

file	CSV file containing an exposure matrix.
check.names	Passed to read.csv. **IMPORTANT**: If TRUE this will replace the double colon in identifiers of the form <tumor_type>::<sample_id> with two periods (i.e. <tumor_type>..<sample_id>. If check.names is true, generate a warning if double colons were present.

Value

Matrix of exposures.

Examples

```
file <- system.file("extdata",
                    "Liver-HCC.exposure.csv",
                    package = "mSigAct")
exposure <- ReadExposure(file)
```

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 = DefaultManyOpts(),
  seed = NULL,
  mc.cores = 2
)
```

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. It can also be the signature id (e.g. "SBS22").
`m.opts`	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.presence.test.out <- SignaturePresenceTest(spectra = spectra,
                                               sigs = sigs.to.use,
                                               target.sig.index = "SBS17a",
                                               seed = 2581,
                                               mc.cores = 2)
```

SortExposure — *Sort columns of an exposure matrix from largest to smallest (or vice versa)*

Description

Sort columns of an exposure matrix from largest to smallest (or vice versa)

Usage

```
SortExposure(exposure, decreasing = TRUE)
```

Arguments

`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.
`decreasing`	If `TRUE`, sort from largest to smallest.

Value

The original `exposure` with columns sorted.

Examples

```
file <- system.file("extdata",
                    "Liver-HCC.exposure.csv",
                    package = "mSigAct")
exposure <- ReadExposure(file)
exposure.sorted <- SortExposure(exposure)
```

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.reconstruction.
- 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)
```

`WriteExposure` *Write an exposure matrix to a file*

Description

Write an exposure matrix to a file

Usage

```
WriteExposure(exposure, file, row.names = TRUE)
```

Arguments

`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.
`file`	File to which to write the exposure matrix (as a CSV file).
`row.names`	Either a logical value indicating whether the row names of `exposure` are to be written along with `exposure`, or a character vector of row names to be written.

Examples

```
file <- system.file("extdata",
                    "Liver-HCC.exposure.csv",
                    package = "mSigAct")
exposure <- ReadExposure(file)
WriteExposure(exposure, file = file.path(tempdir(), "Liver-HCC.exposure.csv"))
```

Index