Package ‘SynSigEval’

May 7, 2020

Type Package

Title Evaluate Mutational Signature Analysis Software
Using Synthetic Spectra Created by Package SynSigGen

Version 0.0.4.9001

Author Steven G. Rozen, Yang Wu

Maintainer Steven G. Rozen <steverozen@gmail.com>

Description Create catalogs of synthetic mutational spectra and assess the
performance of mutational signature analysis programs on these.

License GPL-3

Language en-US

Encoding UTF-8

LazyData true

Imports lsa,
data.table,
ICAMS,
SynSigGen,
stats,
devtools,
utils,
rlang,
graphics,
grDevices,
ggplot2,
ggpubr,
ggbeeswarm,
gtools

Remotes github::steverozen/SynSigGen

Depends R (>= 3.5)

RoxygenNote 7.1.0

Suggests testthat,
knitr,
rmarkdown,
DelayedArray,
BiocManager

R topics documented:

`CopyBestSignatureAnalyzerResult` *Find the SignatureAnalyzer results directory with the best results and make a copy of it as* `sa.results.dir/best.run/`

Description

Find the SignatureAnalyzer results directory with the best results and make a copy of it as `sa.results.dir/best.run/`

Usage

```
CopyBestSignatureAnalyzerResult(
  sa.results.dir,
  verbose = FALSE,
  overwrite = FALSE
)
```

Arguments

`sa.results.dir`	See `BestSignatureAnalyzerResult`
`verbose`	See `BestSignatureAnalyzerResult`
`overwrite`	If TRUE overwrite existing "best.run"

Value

The path of the best directory that was copied as a string, with the list directories examined as the attribute `run.directories`.

`CreateEMuOutput` *Prepare input file for EMu from a EMu formatted catalog file.*

Description

Prepare input file for EMu from a EMu formatted catalog file.

Usage

```
CreateEMuOutput(
  catalog,
  out.dir = paste0(dirname(catalog), "/ExtrAttr/EMu.results"),
  overwrite = FALSE
)
```

Arguments

`catalog`	a catalog in ICAMS format. It can be a .csv file, or a matrix or data.frame. Usually, it refers to `"ground.truth.syn.catalog.csv"`.
`out.dir`	Directory that will be created for the output; abort if it already exists. Usually, the `out.dir` will be a `EMu.results` folder directly under the folder storing `catalog`.
`overwrite`	If TRUE, overwrite existing output

Details

Creates folder named `EMu.results` containing catalogs in EMu-formatted catalogs: Rows are signatures; the first column is the name of the mutation type, while the remaining columns are samples (tumors). These EMu-formated catalogs will the input when running EMu program later on compiled binary.

Value

`invisible(catalog)`, original catalog in EMu format

`CreatehelmsmanOutput` *Prepare input file for helmsman from a helmsman formatted catalog file.*

Description

Prepare input file for helmsman from a helmsman formatted catalog file.

Usage

```
CreatehelmsmanOutput(
  catalog,
  out.dir = paste0(dirname(catalog), "/ExtrAttr/helmsman.results"),
  overwrite = FALSE
)
```

Arguments

`catalog`	a catalog in ICAMS format. It can be a .csv file, or a matrix or data.frame. Usually, it refers to `"ground.truth.syn.catalog.csv"`.
`out.dir`	Directory that will be created for the output; abort if it already exists. Usually, the `out.dir` will be a `helmsman.results` folder directly under the folder storing `catalog`.
`overwrite`	If TRUE, overwrite existing output

Details

Creates folder named `helmsman.results` containing catalogs in helmsman-formatted catalogs: Rows are signatures; the first column is the name of the mutation type, while the remaining columns are samples (tumors). These helmsman-formated catalogs will the input when running helmsman program later on Python platform.

Value

`invisible(catMatrix)`, original catalog in helmsman format

`CreateMultiModalMuSigOutput`

Prepare input file for MultiModalMuSig from a MultiModalMuSig formatted catalog file.

Description

Prepare input file for MultiModalMuSig from a MultiModalMuSig formatted catalog file.

Usage

```
CreateMultiModalMuSigOutput(
  catalog,
  read.catalog.function = NULL,
  out.dir = paste0(dirname(catalog), "/ExtrAttr/MultiModalMuSig.results"),
  overwrite = FALSE
)
```

Arguments

`catalog`	a catalog in ICAMS format. It can be a .csv file, or a matrix or data.frame. Usually, it refers to "`ground.truth.syn.catalog.csv`".
`read.catalog.function`	Function taking a file path as its only argument and returning a catalog as a numeric matrix.
`out.dir`	Directory that will be created for the output; abort if it already exists. Usually, the `out.dir` will be a `MultiModalMuSig.results` folder directly under the folder storing `catalog`.
`overwrite`	If TRUE, overwrite existing output

Details

Creates folder named `MultiModalMuSig.results` containing catalogs in MultiModalMuSig-formatted catalogs: Rows are signatures; the first column is the name of the mutation type, while the remaining columns are samples (tumors). These MM-formated catalogs will the input when running MultiModalMuSig program later on Julia platform.

Value

`invisible(catMatrix)`, original catalog in MultiModalMuSig format

`CreatetcsmOutput` *Prepare input file for tcsm from a tcsm formatted catalog file.*

Description

Prepare input file for tcsm from a tcsm formatted catalog file.

Usage

```
CreatetcsmOutput(
  catalog,
  out.dir = paste0(dirname(catalog), "/ExtrAttr/tcsm.results"),
  overwrite = FALSE
)
```

Arguments

`catalog`	a catalog in ICAMS format. It can be a .csv file, or a matrix or data.frame. Usually, it refers to `"ground.truth.syn.catalog.csv"`.
`out.dir`	Directory that will be created for the output; abort if it already exists. Usually, the `out.dir` will be a `tcsm.results` folder directly under the folder storing `catalog`.
`overwrite`	If TRUE, overwrite existing output

Details

Creates folder named `tcsm.results` containing catalogs in tcsm-formatted catalogs: Rows are signatures; the first column is the name of the mutation type, while the remaining columns are samples (tumors). These tcsm-formated catalogs will the input when running tcsm program later on Python platform.

Value

`invisible(catMatrix)`, original catalog in tcsm format

`Diff4SynDataSets` *diff new directory / files against regression data for testing.*

Description

diff new directory / files against regression data for testing.

Usage

```
Diff4SynDataSets(dirname, unlink)
```

Arguments

`dirname`	the root name of the directories to diff.
`unlink`	if `TRUE` unlink `tmpdirname`, but do not unlink if there are diffs.

Value

The output of the diff command.

`FixSASigNames` *Standardize SignatureAnalyzer signature names*

Description

For example, change `BI_COMPOSITE_SNV_SBS83_P` to `BI_COMPOSITE_SBS83_P`

Usage

```
FixSASigNames(sig.names)
```

Arguments

`sig.names`	Vector of signature names

Details

This is necessary because for COMPOSITE signatures we rbind coordinated "SNV", "DNP", and "INDEL" signatures.

Value

Vector of signatures names with "_SNV" removed.

`helmsmanCatalog2ICAMS` *Read Catalog files or matrices in helmsman format.*

Description

Read Catalog files or matrices in helmsman format.

Usage

```
helmsmanCatalog2ICAMS(
  cat,
  region = "unknown",
  catalog.type = "counts.signature"
)
```

Arguments

`cat`	Input catalog, can be a tab-delimited text file in helmsman format, or a matrix/data.frame object.
`region`	Catalog region. Can be a specific genomic or exonic region, or "unknown". Default: "unknown"
`catalog.type`	Is the catalog a signature catalog, or a spectrum catalog? Default: "counts.signature"

Value

a catalog matrix in ICAMS format.

`ICAMSCatalog2EMu` *Convert Catalogs from ICAMS format to EMu format*

Description

Convert Catalogs from ICAMS format to EMu format

Usage

```
ICAMSCatalog2EMu(catalog)
```

Arguments

`catalog`	A catalog matrix in ICAMS format. (SNS only!)

Value

a matrix without any dimnames, but the values are the transposition of the values in `catalog`.

`ICAMSCatalog2helmsman` *Convert Catalogs from ICAMS format to helmsman format*

Description

Convert Catalogs from ICAMS format to helmsman format

Usage

```
ICAMSCatalog2helmsman(catalog, type = "spectra")
```

Arguments

`catalog`	A catalog matrix in ICAMS format. (SNS only!)
`type`	Whether it is a spectra catalog ("spectra") or a signature catalog ("signature").

Value

a catalog matrix in helmsman format.

`ICAMSCatalog2MM` *Convert Catalogs from ICAMS format to MM format*

Description

Convert Catalogs from ICAMS format to MM format

Usage

```
ICAMSCatalog2MM(catalog)
```

Arguments

`catalog`	A catalog matrix in ICAMS format. (SNS/DNS/ID)

Value

a catalog matrix in MultiModalMuSig format.

`ICAMSCatalog2tcsm` *Convert Catalogs from ICAMS format to tcsm format*

Description

Convert Catalogs from ICAMS format to tcsm format

Usage

```
ICAMSCatalog2tcsm(catalog)
```

Arguments

`catalog`	A catalog matrix in ICAMS format. (SNS only!)

Value

a catalog matrix in tcsm format.

MapSPToSASignatureNamesInExposure

With the signatures represented in a matrix of exposures, find the nearest SignatureAnalyzer exposure.

Description

With the signatures represented in a matrix of exposures, find the nearest SignatureAnalyzer exposure.

Usage

```
MapSPToSASignatureNamesInExposure(
  sp.exposures,
  sa.sig.names.to.consider = colnames(sa.96.sigs)
)
```

Arguments

`sp.exposures`	The exposures
`sa.sig.names.to.consider`	A subset of the colnames of `sa.96.sigs`

Details

IMPORTANT: uses the package global variables `sa.96.sigs` and `sp.sigs`.

Value

A list with

1. exp2 Copy of `sp.exposures` with the rownames(signature names) updated according to the match.
2. `sp.to.sa.sig.match`
3. `sa.to.sp.sig.match` Best matches in the opposite direction

Match1Sig

Find the signature in other.sigs that is nearest (by cosine similarity) to query.sig.

Description

Find the signature in other.sigs that is nearest (by cosine similarity) to query.sig.

Usage

```
Match1Sig(query.sig, other.sigs)
```

Arguments

`query.sig`	A single signature
`other.sigs`	Matrix with each column being one signature

Value

The maximum similarity between `query.sig` and any signature in `other.sigs`

See Also

Other signature matching functions: `MatchSigs1Direction()`, `MatchSigs2Directions()`, `MatchSigsAndRelabel()`

`MatchSigs1Direction` *Find the closest match in* `other.sigs` *for each signature in* `query.sigs`

Description

Find the closest match in `other.sigs` for each signature in `query.sigs`

Usage

```
MatchSigs1Direction(query.sigs, other.sigs)
```

Arguments

`query.sigs`	A signature matrix; signatures for which to find the closest match in other.sigs. The colnames are used as the identifiers of the signatures.
`other.sigs`	A signature matrix; find the closest matches to a signature in this matrix. The colnames are used as the identifiers of the signatures.

Value

A list with one element for each signature in query.sigs. The names of the list elements are the colnames of query.sigs. Each list element is a vector of length 1, and the name of the vector element is the name of the closest matching signature in other.sigs, and the value is the cosine similarity between the given signature in query.sigs and the matching signature in other.sigs.

See Also

Other signature matching functions: `Match1Sig()`, `MatchSigs2Directions()`, `MatchSigsAndRelabel()`

MatchSigs2Directions *Calculate bidirectional closest similarities between two sets of signatures and the average of the similarities.*

Description

Calculate bidirectional closest similarities between two sets of signatures and the average of the similarities.

Usage

```
MatchSigs2Directions(sigs1, sigs2)
```

Arguments

`sigs1`	Matrix of signatures; colnames are used as signature identifiers, and the colnames in `sigs1` should be distinguishable from those in `sigs2`.
`sigs2`	Matrix of signatures; colnames are used as signature identifiers.

Value

A list with the elements:

`averCosSim`: the average of the cosine similarities between each signature in `sigs1` and its closest match in `sigs2` and the closest match between each signature in `sigs2` and its closest match in `sigs1`.

`match1`: a data frame with rownames being signature identifiers from `sigs1`, the signature identifier of the closest match in `sigs1` in the 1st column, and the cosine similarity between them in the 2nd column.

`match2`: a data frame with the rownames being signature identifiers from `sigs2`, the signature identifier of the closest match in `sigs1` in the 1st column, and the cosine similarity between them in the 2nd column.

`match1` and `match2` might not have the same number of rows.

See Also

Other signature matching functions: `Match1Sig()`, `MatchSigs1Direction()`, `MatchSigsAndRelabel()`

MatchSigsAndRelabel *Run* `MatchSigs2Directions`*, then plot its results and write them as .csv files.*

Description

Run `MatchSigs2Directions`, then plot its results and write them as .csv files.

Usage

```
MatchSigsAndRelabel(ex.sigs, gt.sigs, exposure)
```

Arguments

`ex.sigs`	Newly extracted signatures to be compared to gt.sigs
`gt.sigs`	"Ground truth" signatures.
`exposure`	"Ground truth" exposures used generate the synthetic data from which `ex.sigs` were extracted.

Value

A list with the elements `averCosSim`, `match1`, `match2` as for `SigSetSimilarity`, with `match1` being matches for the the extracted signatures (`ex.sigs`) and `match2` being the matches for the ground truth signatures (`gt.sigs`). The return list also echos the input arguments `ex.sigs` and `gt.sigs`.

See Also

Other signature matching functions: `Match1Sig()`, `MatchSigs1Direction()`, `MatchSigs2Directions()`

`MMCatalog2ICAMS` *Convert Catalogs (File or Matrix) from MM format to ICAMS format*

Description

Convert Catalogs (File or Matrix) from MM format to ICAMS format

Usage

```
MMCatalog2ICAMS(cat, region = "unknown", catalog.type = "counts.signature")
```

Arguments

`cat`	Input catalog, can be a tab-delimited file or matrix in MultiModalMuSig format.
`region`	Catalog region. Can be a specific genomic or exonic region, or "unknown". Default: "unknown"
`catalog.type`	Is the catalog a signature catalog, or a spectrum catalog? Default: "counts.signature"

Value

a catalog matrix in ICAMS format.

`MutationalSignatures` *Reference mutational signature profiles from PCAWG7.*

Description

Reference mutational signature profiles from PCAWG7.

Usage

```
sa.96.sigs

sa.COMPOSITE.sigs

sa.DBS.sigs

sa.ID.sigs

sp.sigs
```

Format

Numerical matrix with rows indicating mutation types and columns indicating signatures.

An object of class `matrix` (inherits from `array`) with 96 rows and 60 columns.

An object of class `matrix` (inherits from `array`) with 1697 rows and 60 columns.

An object of class `matrix` (inherits from `array`) with 78 rows and 15 columns.

An object of class `matrix` (inherits from `array`) with 83 rows and 29 columns.

An object of class `matrix` (inherits from `array`) with 96 rows and 65 columns.

Details

`sa.96.sigs` provides SignatureAnalyzer mutational signature profiles collapsed from COMPOSITE to 96-channel SNS signatures.

`sa.COMPOSITE.sigs` provides COMPOSITE mutational signature profiles extracted by SignatureAnalyzer. `sa.COMPOSITE.sigs` are an `rbind` of the contents of `https://www.synapse.org/#!Synapse:syn11738311` (SBS 1536), `https://www.synapse.org/#!Synapse:syn11738308` (DBS), and `https://www.synapse.org/#!Synapse:syn11738309` (ID).

`sa.DBS.sigs` provides the DBS signatures extracted by SignatureAnalyzer, from `https://www.synapse.org/#!Synapse:syn11738312`. These are not the DBS signatures that are part of `sa.COMPOSITE.sigs`; these were extracted from the ID catalogs alone.

`sa.ID.sigs` provides the ID signatures extracted by SignatureAnalyzer, from `https://www.synapse.org/#!Synapse:syn11738313`.These are not the ID signatures that are part of `sa.COMPOSITE.sigs`; these were extracted from the ID catalogs alone.

`sp.sigs` provides signatures extracted by SigProfiler.

Source

https://www.synapse.org/#!Synapse:syn11738310

https://www.synapse.org/#!Synapse:syn11738311

https://www.synapse.org/#!Synapse:syn11738308

https://www.synapse.org/#!Synapse:syn11738309

https://www.synapse.org/#!Synapse:syn11738312

https://www.synapse.org/#!Synapse:syn11738313

https://www.synapse.org/#!Synapse:syn11738319

`NumFromId` *Get the numerical parts of signature ids*

Description

Get the numerical parts of signature ids

Usage

```
NumFromId(s)
```

Arguments

`s`	A character vector

Value

A vector, each element of which is the integer corresponding to the first string of digits of an element of s

---

`PlotCatCOMPOSITE` *Plot the a SignatureAnalyzer COMPOSITE signature or catalog into separate pdfs*

Description

Plot the a SignatureAnalyzer COMPOSITE signature or catalog into separate pdfs

Usage

```
PlotCatCOMPOSITE(catalog, filename.header, type, id = colnames(catalog))
```

Arguments

`catalog`	Catalog or signature matrix
`filename.header`	Contain path and the beginning part of the file name. The name of the pdf files will be: `filename.header`.SNS.96.pdf `filename.header`.SNS.1536.pdf `filename.header`.DNS.78.pdf `filename.header`.ID.83.pdf
`type`	See `PlotCatalogToPdf`.
`id`	A vector containing the identifiers of the samples or signatures in `catalog`.

ReadAndAnalyzeExposures *Assess how well inferred exposures match input exposures We assume that in many cases attribution programs will be run outside of R on file inputs and will generate fill outputs.*

Description

Assess how well inferred exposures match input exposures

We assume that in many cases attribution programs will be run outside of R on file inputs and will generate fill outputs.

Usage

```
ReadAndAnalyzeExposures(
  extracted.sigs,
  ground.truth.sigs,
  inferred.exp.path,
  ground.truth.exposures
)
```

Arguments

`extracted.sigs`	Path to file containing the extracted signature profiles.
`ground.truth.sigs`	File containing signature profiles from which the synthetic data were generated.
`inferred.exp.path`	File containing mutation counts (exposures) of synthetic tumors which are inferred to extracted or input signatures.
`ground.truth.exposures`	File containing the exposures from which the synthetic catalogs were generated. This file is used to restrict assessment of signature exposures to only those signatures in `ground.truth.sigs` that were actually represented in the exposures.

Details

Generates output files by calling `MatchSigsAndRelabel`

Value

A `data.frame` recording:

`Ground.truth.exposure`: sum of ground truth exposures of all tumors to all ground-truth signatures.

`Inferred.exposure`: sum of inferred exposures of all tumors to all ground-truth signatures. Here, inferred exposure of a tumor to a ground-truth signature equals to the sum of the exposures of this tumor to all extracted signatures which are most similar to a ground-truth signature. If there is no extracted signature resembling an ground-truth signature, the inferred exposure of this ground-truth signature will be `0`.

`Absolute.difference`: sum of absolute difference between ground-truth exposure and inferred exposure of all tumors to all ground-truth signatures.

`ReadAndAnalyzeSigs` *Assess how well extracted signatures match input signatures We assume that in many cases extraction programs will be run outside of R on file inputs and will generate fill outputs.*

Description

Assess how well extracted signatures match input signatures

We assume that in many cases extraction programs will be run outside of R on file inputs and will generate fill outputs.

Usage

```
ReadAndAnalyzeSigs(extracted.sigs, ground.truth.sigs, ground.truth.exposures)
```

Arguments

`extracted.sigs`	Path to file containing the extracted signature profiles.
`ground.truth.sigs`	File containing signature profiles from which the synthetic data were generated.
`ground.truth.exposures`	File containing the exposures from which the synthetic catalogs were generated. This file is used to restrict assessment to only those signatures in `ground.truth.sigs` that were actually represented in the exposures.

Details

Generates output files by calling `MatchSigsAndRelabel`

Value

See `MatchSigsAndRelabel`

`ReadEMuCatalog` *Read Catalog files in EMu format.*

Description

Read Catalog files in EMu format.

Usage

```
ReadEMuCatalog(
  cat,
  mutTypes,
  sigOrSampleNames,
  region = "unknown",
  catalog.type = "counts.signature"
)
```

Arguments

`cat`	A tab-delimited catalog text file in EMu format; or a EMu formatted matrix or data.frame.
`mutTypes`	Types of mutations. They are usually from an `ICAMS:::catalog.row.header` object.
`sigOrSampleNames`	If input file is a counts signature file (`catalog.type == "counts.signature"`), signature names should be provided. If input file is a counts spectra file (`catalog.type == "counts"`), names of samples should be provided.
`region`	Catalog region. Can be a specific genomic or exonic region, or "unknown". Default: "unknown"
`catalog.type`	Is the catalog a signature catalog, or a spectrum catalog? Default: "counts"

Value

a catalog matrix in ICAMS format.

`ReadEMuExposureFile` *Read Exposure files in EMu format.*

Description

Read Exposure files in EMu format.

Usage

```
ReadEMuExposureFile(exposureFile, sigNames, sampleNames)
```

Arguments

`exposureFile`	Exposure file generated by EMu. Usually, it is called "W_components.txt".
`sigNames`	Names of signatures. These will be served as the rownames of the exposure matrix.
`sampleNames`	Names of samples in exposure file. Return ICAMS/SynSigEval formatted exposure matrix.

`ReadExposureMM` *Read Catalog files in MM format*

Description

Read Catalog files in MM format

Usage

```
ReadExposureMM(exposureFile)
```

Arguments

`exposureFile`	Input exposure file, can be a tab-delimited text file in MultiModalMuSig format.

Value

a exposure matrix in ICAMS format.

`ReadhelmsmanExposure` *Read Exposure files in helmsman format.*

Description

Read Exposure files in helmsman format.

Usage

```
ReadhelmsmanExposure(exposure, check.names = TRUE)
```

Arguments

`exposure`	Exposure file generated by helmsman. Usually, it is called "W_components.txt".
`check.names`	logical. If `TRUE` then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by `make.names`) so that they are, and also to ensure that there are no duplicates. Return ICAMS/SynSigEval formatted exposure matrix.

ReadSigProfilerExposure

Read a file containing exposures attributed by SigProfiler/Python

Description

Read a file containing exposures attributed by SigProfiler/Python

Usage

```
ReadSigProfilerExposure(file)
```

Arguments

`file`	The name of the file to read.

Value

The corresponding signature matrix in standard internal representation.

ReadSigProfilerSigDBS78

Read a file containing DBS78 signatures extracted by SigProfiler/Python

Description

Read a file containing DBS78 signatures extracted by SigProfiler/Python

Usage

```
ReadSigProfilerSigDBS78(file)
```

Arguments

`file`	The name of the file to read.

Value

The corresponding signature matrix in standard internal representation.

`ReadSigProfilerSigSBS96` *Read a file containing SBS96 signatures extracted by SigProfiler/Python*

Description

Read a file containing SBS96 signatures extracted by SigProfiler/Python

Usage

```
ReadSigProfilerSigSBS96(file)
```

Arguments

`file`	The name of the file to read.

Value

The corresponding signature matrix in standard internal representation.

`ReadtcsmExposure` *Read Exposure files in tcsm format.*

Description

Read Exposure files in tcsm format.

Usage

```
ReadtcsmExposure(exposure)
```

Arguments

`exposure`	Exposure file generated by tcsm. Usually, it is called "W_components.txt". Return ICAMS/SynSigEval formatted exposure matrix.

`RealExposures` *Real exposure (signature attributions) from SignatureAnalyzer and SigProfiler*

Description

Real exposure (signature attributions) from SignatureAnalyzer and SigProfiler

Usage

```
sa.all.real.exposures

sp.all.real.exposures

sa.no.hyper.real.exposures

sp.no.hyper.real.exposures
```

Format

Numerical matrix with rows indicating signatures and columns indicating (tumor) samples.

An object of class `matrix` (inherits from `array`) with 60 rows and 2780 columns.

An object of class `matrix` (inherits from `array`) with 65 rows and 2780 columns.

An object of class `matrix` (inherits from `array`) with 35 rows and 2624 columns.

An object of class `matrix` (inherits from `array`) with 65 rows and 2624 columns.

Note

Prefix `sa` indicates SignatureAnalyzers, `sp` indicates SigProfiler; `all` indicates all samples, `no.hyper` means that hypermutated tumors as defined for SignatureAnalyzer have been removed.

Source

https://dx.doi.org/10.7303/syn11761237.4

https://dx.doi.org/10.7303/syn11738669.5

https://dx.doi.org/10.7303/syn11761198.4

https://dx.doi.org/10.7303/syn11761237.4

`SignatureAnalyzerPrepHyper1Secondary`
Prepare the "hypermutated" segment (a.k.a "Secondary" segment of a split non-hyper and hyper data set.)

Description

Prepare the "hypermutated" segment (a.k.a "Secondary" segment of a split non-hyper and hyper data set.)

Usage

```
SignatureAnalyzerPrepHyper1Secondary(
  non.hyper.results,
  primary.catalog,
  hyper.catalog,
  secondary.catalog,
  overwrite = TRUE
)
```

Arguments

`non.hyper.results`	The directory containing the the results of the analysis of the non-hyper-mutated (a.k.a "PRIMARY") mutational spectra.
`primary.catalog`	The catalog of non-hyper-mutated mutational spectra from which the results in `non.hyper.results` were derived.
`hyper.catalog`	The catalog of hyper-mutated mutational spectra which will be part of the input for the secondary analysis.
`secondary.catalog`	The final output catalog on which the secondary analysis will be performed; this is a `cbind` of pseudo-spectra generated from the PRIMARY signatures with the `hyper.catalog`.
`overwrite`	If `TRUE` overwrite possible previously computed files and/or directories.

`SignatureAnalyzerPrepHyper4`
Prepare the "hypermutated" segment (a.k.a "Secondary" segment of a split non-hyper and hyper data set.)

Description

Prepare the "hypermutated" segment (a.k.a "Secondary" segment of a split non-hyper and hyper data set.)

Usage

```
SignatureAnalyzerPrepHyper4(parent.dir, overwrite = FALSE)
```

Arguments

`parent.dir`	A directory that must contain subdirectories `syn.SA.hyper.low` and `syn.SA.hyper.mixed`. `syn.SA.hyper.low` must contain the synthetic non-hypermutated data and the results of running SignatureAnalyzer on the non-hyper segment, with subdirectories `sa.sa.96`, `sa.sa.COMPOSITE`, `sp.sa.COMPOSITE`, and `sp.sp.` `syn.SA.hyper.mixed` must contain the synthetic hypermutated data. The results of the initial SignatureAnalyzer run will be placed here to prepare this directory for the second SignatureAnalyzer run.
`overwrite`	If `TRUE` overwrite existing directories and files.

`SignatureAnalyzerSummarizeSBS1SBS5` *Summarize all subdirectories of Signatureanalyzer results on the correlated SBS1 / SBS5.*

Description

This is special-purpose function to summarize results from one in-silico experiment that examines how well signatures can be extracted from synthetic tumors with correlated SBS1 and SBS5.

Usage

```
SignatureAnalyzerSummarizeSBS1SBS5(top.level.dir, overwrite = FALSE)
```

Arguments

`top.level.dir`	Path to top level directory.
`overwrite`	If TRUE overwrite existing directories and files.

`SignatureAnalyzerSummarizeTopLevel` *Summarize all subdirectories of SignatureAnalyzer results on a major dataset.*

Description

This function depends on a particular directory structure: see argument `top.level.dir`. This function finds the best of multiple SignatureAnalyzer extraction runs and summarizes the comparision of the best run with the ground truth.

Usage

```
SignatureAnalyzerSummarizeTopLevel(top.level.dir, overwrite = FALSE)
```

Arguments

`top.level.dir`	Path to top level directory, which must contain the following subdirectories: • `sa.sa.96/sa.results/` • `sp.sp/sa.results/` • `sa.sa.COMPOSITE/sa.results/` • `sp.sa.COMPOSITE/sa.results/` Each of the directories must contain additonal subdirectories, one for each SignatureAnalyzer run, names `sa.run.<n>`, where <n> is an integer (string of digits).
`overwrite`	If TRUE overwrite existing summary files.

`SplitCatCOMPOSITE` *Split COMPOSITE (SNS1536+DBS78+ID83) catalogs in ICAMS format into 3 individual catalogs.*

Description

Split COMPOSITE (SNS1536+DBS78+ID83) catalogs in ICAMS format into 3 individual catalogs.

Usage

```
SplitCatCOMPOSITE(catalog)
```

Arguments

`catalog`	Input catalog, can be a .csv file or matrix in ICAMS COMPOSITE format.

Value

a list, containing 3 catalog matrices in MultiModalMuSig format. Each matrix contains SNS1536, DBS78 and ID83 information, respectively.

`SummarizeMultiRuns` *Assess/evaluate multiple summarized runs for one dataset from one software approach.*

Description

Summarize results from each software approach in `tool.dir/run.names` (generated by running a software approach), combine them into `tool.dir`.

Usage

```
SummarizeMultiRuns(datasetName, toolName, tool.dir, run.names)
```

Arguments

`datasetName`	Name of the dataset. (e.g. "S.0.1.Rsq.0.1"). Usually, it is has the same name as `basename(top.dir)`.
`toolName`	Name of software approach. (e.g. "sigproextractor")
`tool.dir`	Fourth level path from the `top.dir`. Expected to have multiple runs with different names (e.g. "seed.1") That is, `top.dir`/sp.sp/ExtrAttr/sa.results/. or `top.dir`/sa.sa.96/Attr/deconstructSigs.results/ Here, `top.dir` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. However there should be a directory within the `tool.dir` which stores the software output.
`run.names`	A character vector records the list of `run.dir`, or fifth level directories from the dataset top-level folder. E.g., c("seed.1","seed.691")

Details

Also writes multiple files into folder `tool.dir`:

Value

A list contain c(`mean`,`sd`) of multiple runs: Cosine similarity True Positives(TP): Ground-truth signatures which are active in the spectra, and extracted. False Negatives(FN): Ground-truth signatures not extracted. False Positives(FP): Signatures wrongly extracted, not resembling any ground-truth signatures. True positive rate (TPR, Sensitivity): TP / (TP + FN) False discovery rate (FDR): FP / (FP + TP)

SummarizeMultiToolsMultiDatasets

Combine results for multiple datasets, from different software approaches.

Description

Summarize results from each software approach in `third.level.dir/tool.dirnames` (generated by `SummarizeMultiRuns`), combine them into `third.level.dir`.

Usage

```
SummarizeMultiToolsMultiDatasets(
  dataset.dirs,
  second.third.level.dirname,
  out.dir,
  overwrite = FALSE
)
```

Arguments

`dataset.dirs`	Paths of top-level dataset directories trees you want to investigate. E.g. "./S.0.1.Rsq.0.1"
`second.third.level.dirname`	Name of the second.level.dir (e.g. "sp.sp") and the third.level.dir (e.g. "ExtrAttr") to be investigated. Examples are: "sp.sp/ExtrAttr", "sa.sa.96/Attr" Note: `multiTools.RDa` are expected to be exist under `dataset.dirs/second.third.level.dirnam`
`out.dir`	Path of the output directory.
`overwrite`	Whether to overwrite the contents in out.dir if it already exists. (Default: FALSE)

`SummarizeMultiToolsOneDataset`
Combine results for a single dataset, from different software approaches.

Description

Summarize results from each software approach in `third.level.dir/tool.dirnames` (generated by `SummarizeMultiRuns`), combine them into `third.level.dir`.

Usage

```
SummarizeMultiToolsOneDataset(
  third.level.dir,
  toolNames,
  tool.dirnames,
  datasetGroup,
  datasetGroupName,
  datasetSubGroup,
  datasetSubGroupName
)
```

Arguments

`third.level.dir`	Third level path distinguishing de-novo extraction + attribution packages from attribution-only packages. Examples: `top.dir`/sp.sp/ExtrAttr/ `top.dir`/sa.sa/Attr/
`toolNames`	Names of software approach. (e.g. "sigproextractor")
`tool.dirnames`	Third level path from the `top.dir`. Expected to have summarized results generated by `SummarizeMultiRuns`. (multiRun.RDa, ManhattanDist.csv, meanSD.csv, meanSD.Manhattan.dist.csv) Examples: `"signeR.results"` (Under `third.level.dir` "ExtrAttr") `"deconstructSigs.results"` (Under `third.level.dir` "Attr") Here, `top.dir` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. However there should be a directory within the `tool.names` which stores the software output.
`datasetGroup`	Numeric or character vector specifying the group each dataset belong to. E.g. For SBS1-SBS5 correlated datasets, we can consider slope as the group: c("slope=0.1","slope=0.5","s Default: "Default"

`datasetGroupName`	Meaning or label of all datasetGroup. E.g. For SBS1-SBS5 correlated datasets, we can consider `"SBS1:SBS5 mutation count ratio"` as the label of the `datasetGroup` slope.
`datasetSubGroup`	Numeric or character vector differentiating datasets within each group. E.g. For SBS1-SBS5 correlated datasets, we can consider Pearson's R^2 as the subgroup: c("Rsq=0.1","Rsq=0.2","Rsq=0.3","Rsq=0.6") Default: Names of datasets, which are `basename(dataset.dirs)`
`datasetSubGroupName`	Meaning or label of all datasetSubGroup. E.g. For SBS1-SBS5 correlated datasets, we can consider `"Pearson's R squared"` as the label of the `datasetSubGroup` Pearson's R^2.

Value

A list contain c(`mean,sd`) of multiple runs: Cosine similarity True Positives(TP): Ground-truth signatures which are active in the spectra, and extracted. False Negatives(FN): Ground-truth signatures not extracted. False Positives(FP): Signatures wrongly extracted, not resembling any ground-truth signatures. True positive rate (TPR, Sensitivity): TP / (TP + FN) False discovery rate (FDR): FP / (FP + TP)

`SummarizeOneToolMultiDatasets`
Combine results for multiple datasets, from one software approaches.

Description

Summarize results from each software approach in `third.level.dir/tool.dirnames` (generated by `SummarizeMultiRuns`), combine them into `third.level.dir`.

Usage

```
SummarizeOneToolMultiDatasets(
  dataset.dirs,
  datasetGroup = NULL,
  datasetGroupName,
  datasetSubGroup = NULL,
  datasetSubGroupName,
  toolName,
  tool.dirname,
  out.dir,
  overwrite = FALSE
)
```

Arguments

`dataset.dirs`	Paths of top-level dataset directories trees you want to investigate. E.g. "./S.0.1.Rsq.0.1"
`datasetGroup`	Numeric or character vector specifying the group each dataset belong to. E.g. For SBS1-SBS5 correlated datasets, we can consider slope (SBS1:SBS5 count ratio) as the group: `c(0.1,0.5,1,2,5,10)` Default: "Default"

`datasetGroupName`	Meaning or label of all datasetGroup. E.g. For SBS1-SBS5 correlated datasets, we can consider `"SBS1:SBS5 mutation count ratio"` as the label of the `datasetGroup` slope.
`datasetSubGroup`	Numeric or character vector differentiating datasets within each group. E.g. For SBS1-SBS5 correlated datasets, we can consider Pearson's R^2 as the subgroup: c(0.1,0.2,0.3,0.6) Default: Names of datasets, which are `basename(dataset.dirs)`
`datasetSubGroupName`	Meaning or label of all datasetSubGroup. E.g. For SBS1-SBS5 correlated datasets, we can consider `"Pearson's R squared"` as the label of the `datasetSubGroup` Pearson's R^2.
`toolName`	Name of software approach to be investigated (e.g. "sigproextractor")
`tool.dirname`	Name of the second.level.dir (e.g. "sp.sp"), third.level.dir (e.g. "ExtrAttr") and tool.dir (e.g. "sigproextractor.results") to be investigated. One example: "sp.sp/ExtrAttr/sigproextractor.results" Note: this function expects the summary generated by `SummarizeSigOneSubdir` under `dataset.dirs/tool.dirname`
`out.dir`	Path of the output directory.
`overwrite`	Whether to overwrite the contents in out.dir if it already exists. (Default: FALSE)

`SummarizeSigOneAttrSubdir` *Assess/evaluate results from packages which can ONLY do exposure attribution.*

Description

Packages including but not limited to: deconstructSigs, YAPSA.

Usage

```
SummarizeSigOneAttrSubdir(
  run.dir,
  ground.truth.exposure.dir = paste0(run.dir, "/../../../"),
  overwrite = FALSE
)
```

Arguments

`run.dir`	Lowest level path to results, e.g. `<top.dir>/sa.sa.96/Attr/YAPSA.results/seed.1/` Here, `<top.dir>` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. For packages which can do both extraction and attribution, we expect two files, `ground.truth.signatures.csv` and `inferred.exposures.csv` are in the folder.
`ground.truth.exposure.dir`	Folder which stores ground-truth exposures. It defaults to be `sub.dir`, i.e. `run.dir`/../../
`overwrite`	If TRUE overwrite existing directories and files.

Details

Here, we excluded SignatureEstimation. Although it is also a package with only attribution, but it has two attribution algorithms. Therefore the naming of the results are slightly different from the other two packages.

`SummarizeSigOneExtrAttrSubdir`
Assess/evaluate results from packages which can do BOTH extraction and attribution, excluding SigProfiler-Python and SignatureAnalyzer.

Description

Packages including but not limited to: HDP, MutationalPatterns, sigfit, SigneR, SomaticSignatures.

Usage

```
SummarizeSigOneExtrAttrSubdir(
  run.dir,
  ground.truth.exposure.dir = paste0(run.dir, "/../../../"),
  overwrite = FALSE
)
```

Arguments

`run.dir`	Lowest level path to result of a run. E.g. `<top.dir>/sa.sa.96/ExtrAttr/SomaticSignatures.res` Here, `<top.dir>` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. For packages which can do both extraction and attribution, we expect two files, `extracted.signatures.csv` and `inferred.exposures.csv` are in the folder.
`ground.truth.exposure.dir`	Folder which stores ground-truth exposures. It defaults to be `sub.dir`, i.e. `run.dir`/../../../
`overwrite`	If TRUE overwrite existing directories and files.

`SummarizeSigOnehelmsmanSubdir`
Assess/evaluate results from SigProfiler-python (a.k.a. sigproextractor) Assessment is restricted to v0.0.5.43, because different version has different folder structure.

Description

Assess/evaluate results from SigProfiler-python (a.k.a. sigproextractor) Assessment is restricted to v0.0.5.43, because different version has different folder structure.

Usage

```
SummarizeSigOnehelmsmanSubdir(
  run.dir,
  ground.truth.exposure.dir = paste0(run.dir, "/../../../"),
  overwrite = FALSE,
  hierarchy = FALSE
)
```

Arguments

`run.dir`	Lowest level path to results, e.g. `<top.dir>/sa.sa.96/ExtrAttr/sigproextractor.results/see` Here, `<top.dir>` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. However there should be a directory `<run.dir>/SBS96` which stores SigProfiler results.
`ground.truth.exposure.dir`	Folder which stores ground-truth exposures. Usually, it refers to `sub.dir`, i.e. `run.dir`/../../../
`overwrite`	If TRUE overwrite existing directories and files.
`hierarchy`	Whether the user have enabled hierarchy = True when running sigproextractor. specifying True or False into sigproextractor will cause the program to generate different folder structure. (Default: `FALSE`)

SummarizeSigOneSigProExtractorSubdir

Assess/evaluate results from SigProExtractor SigProFiler-python de novo extraction and attribution package. Assessment is restricted to v0.0.5.43+, because different version has different folder structure.

Description

Assess/evaluate results from SigProExtractor SigProFiler-python de novo extraction and attribution package. Assessment is restricted to v0.0.5.43+, because different version has different folder structure.

Usage

```
SummarizeSigOneSigProExtractorSubdir(
  run.dir,
  ground.truth.exposure.dir = paste0(run.dir, "/../../../"),
  overwrite = FALSE,
  hierarchy = FALSE
)
```

Arguments

`run.dir`	Lowest level path to results, e.g. `<top.dir>/sa.sa.96/ExtrAttr/sigproextractor.results/see` Here, `<top.dir>` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. However there should be a directory `<run.dir>/SBS96` which stores SigProfiler results.

`ground.truth.exposure.dir`	TODO(Wu Yang): Fix this File name which stores ground-truth exposures; defaults to `"ground.truth.syn.exposures.csv"`. This file can be found in the `sub.dir`, i.e. `<run.dir>/../../../`
`overwrite`	If TRUE overwrite existing directories and files.
`hierarchy`	Whether the user have enabled hierarchy = True when running sigproextractor. specifying True or False into sigproextractor will cause the program to generate different folder structure. (Default: `FALSE`)

`SummarizeSigOneSigProSSSubdir`
Assess/evaluate results from sigproSS (a.k.a. SigProfiler Python attribution package)

Description

Assess/evaluate results from sigproSS (a.k.a. SigProfiler Python attribution package)

Usage

```
SummarizeSigOneSigProSSSubdir(
  run.dir,
  ground.truth.exposure.dir = paste0(run.dir, "/../../../"),
  overwrite = FALSE
)
```

Arguments

`run.dir`	Lowest level path to results, e.g. `<top.dir>/sa.sa.96/ExtrAttr/sigproextractor.results/see` Here, `<top.dir>` refers to a top-level directory which contains the full information of a synthetic dataset. (e.g. `syn.2.7a.7b.abst.v8`) This code depends on a conventional directory structure documented elsewhere. However there should be a directory `<run.dir>/SBS96` which stores SigProfiler results.
`ground.truth.exposure.dir`	TODO(Wu Yang): Fix this File name which stores ground-truth exposures; defaults to `"ground.truth.syn.exposures.csv"`. This file can be found in the `sub.dir`, i.e. `<run.dir>/../../../`
`overwrite`	If TRUE overwrite existing directories and files.

`SummarizeSigProExtractor`
Summarize SigProfiler results in the sa.sa.96 and/or sp.sp subdirectories.

Description

Summarize SigProfiler results in the sa.sa.96 and/or sp.sp subdirectories.

Usage

```
SummarizeSigProExtractor(
  top.dir,
  sub.dir = c("sa.sa.96", "sp.sp"),
  overwrite = FALSE
)
```

Arguments

`top.dir`	The top directory of a conventional data structure containing at least one of the subdirectories: sa.sa.96/sp.results and sp.sp/sp.results; see further documentation elsewhere.
`sub.dir`	The subdirectory under `top.dir`, and containing a folder named sp.results. By default, it contains both `c("sa.sa","sp.sp")`. But you should specify `sub.dir = "sp.sp"` for `top.dir` with only the `sp.sp` subdirectory (as is the case for the correlated SBS1-and-SBS5-containing data sets).
`overwrite`	whether to overwrite the existing `run.dir/summary` folder? If chosen to be FALSE and there is an existing summary folder, an error will be raised.

Details

Results are put in standardized subdirectories of `top.dir`.

`SynSigEval` *SynSigEval*

Description

Assess the performance of mutational-signature analysis programs Using catalogs of synthetic mutational spectra created by package `SynSigGen`.

Overview

The main focus is generating synthetic catalogs of mutational spectra (mutations in tumors) based on known mutational signature profiles and attributions (assignment of exposures to tumors) in the PCAWG7 data. We call this kind of synthetic data broadly "reality-based" synthetic data. The package also has a set of functions that generate random mutational signature profiles and then create synthetic catalogs based on these random signature profiles. We call this kind of synthetic data "random" synthetic data, while pointing out that much depends on the distributions from which the random signature profiles and attributions are generated.

Typical workflow for generating catalogs of "reality-based" synthetic mutational spectra is as follows.

```
In \code{SynSigGen}:

Input (based on SignatureAnalyzer or SigProfiler analysis of PCAWG tumors)
  A, matrix of attributions (signatures x samples)
  S, mutational signature profiles (mutation type x signature)
```

```
P <- GetSynSigParamsFromExposures(A, ...)

synthetic.exposures <- GenerateSyntheticExposures(P, ...)

synthetic.spectra <- CreateAndWRiteCatalog(S, synthetic.exposures, ...)

In \code{SynSigEval}:

T <- Signatures extracted by SignatureAnalzer or SigProfiler on synthetic.spectra

SummarizeResults(T, S, synthetic.exposures, ...)
```

(In SynSigGen) Creating Synthetic Mutational Catalogs

These functions create synthetic mutational catalogs based on parameters derived from signature profiles and attributions (exposures).

Summarize results (of signature extraction)

Relevant functions are:

1. `SummarizeSigProExtractor`
2. `SignatureAnalyzerSummarizeTopLevel`
3. `SignatureAnalyzerSummarizeSBS1SBS5`

Comparing two sets of mutational signatures

Functions for comparing mutational signatures and sets of mutational signatures. Often we will be interested in comparing signature profiles extracted from synthetic data to the ground-truth signature profiles.

`Match1Sig`, `MatchSigs1Direction`, `MatchSigs2Directions`, `MatchSigsAndRelabel`

Folder structure for SynSigEval v0.2

Summary function will fit to the new 5-level folder structure:

First Level - `top.level.dir`: dataset folder (e.g. "S.0.1.Rsq.0.1", "syn.prancreas"). All spectra datasets under any top.level.dir have the same exposure.

Second Level - `ground.truth.exposure.dir`: spectra folder: (e.g. "sp.sp", "sa.sa.96"). All spectra datasets under any second.level.dir have the same signature and the same exposure counts.

Third Level - `third.level.dir`: It can be ("Attr") for storing results of packages which can only do exposure attribution of known signatures ("Attr"); it can also be ("ExtrAttr"), folder to store results of software packages which can do de-novo extraction and following attribution.

Fourth Level - `tool.dir`: The results of a software package (e.g. "sigproextractor.results","SignatureEstimation.QP.resul
Under this level, `tool.dir` may contain multiple `run.dir`, each is a run of the software package using a specific number of seed.

Fifth level - `run.dir`: contains results from a run of the software package using a specific number of seed. (e.g. "seed.1") `tcsmCatalog2ICAMS` *Read Catalog files or matrices in tcsm format.*

Description

Read Catalog files or matrices in tcsm format.

Usage

```
tcsmCatalog2ICAMS(cat, region = "unknown", catalog.type = "counts.signature")
```

Arguments

`cat`	Input catalog, can be a tab-delimited text file in tcsm format, or a matrix/data.frame object.
`region`	Catalog region. Can be a specific genomic or exonic region, or "unknown". Default: "unknown"
`catalog.type`	Is the catalog a signature catalog, or a spectrum catalog? Default: "counts.signature"

Value

a catalog matrix in ICAMS format.

Index