

Package ‘SynSigRun’

September 20, 2021

Type Package

Title Run Mutational Signature Analysis Software Packages Using Mutational Spectra generated by SynSigGen

Version 0.1.2

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

biocViews

Imports data.table,
devtools,
dplyr,
ICAMS,
lsa,
magrittr,
rlang,
stats,
SynSigGen,
tibble,
utils

Remotes github::steverozen/SynSigGen

Depends R (>= 3.5)

RoxygenNote 7.1.1

Suggests BiocManager,
decompTumor2Sig,
deconstructSigs,
DelayedArray,
hdp,
knitr,
maftools,
mSigAct,

MutationalPatterns,
 mutSignatures,
 NMF,
 rmarkdown,
 rstan,
 sigfit,
 SignatureEstimation,
 signeR,
 SparseSignatures,
 testthat,
 YAPSA

R topics documented:

CopyBestSignatureAnalyzerResult	3
Diff4SynDataSets	3
FixSASigNames	4
InstalldecompTumor2Sig	4
InstalldeconstructSigs	4
InstallmutSignatures	5
MapSPToSASignatureNamesInExposure	5
MutationalSignatures	6
RealExposures	7
RundecompTumor2SigAttributeOnly	8
RundeconstructSigsAttributeOnly	9
RunhdpLessHier	10
Runmaftools	11
RunmSigActAttributeOnly	12
RunMutationalPatterns	13
RunMutationalPatternsAttributeOnly	15
RunmutSignatures	16
RunmutSignaturesAttributeOnly	17
RunmutSpec	18
Runsigfit	19
RunsigfitAttributeOnly	20
Runsigminer	21
RunSignatureAnalyzerAttribution	22
RunSignatureAnalyzerOnFile	24
RunSignatureEstimationQPAttributeOnly	25
RunSignatureEstimationSAAttributeOnly	26
RunsigneR	27
RunSomaticSignatures	28
RunSparseSignatures	30
Runtcsm	31
RunYAPSAAttributeOnly	32
SAMultiRunOneCatalog	33
SignatureAnalyzer4MatchedCatalogs	34
SignatureAnalyzerOneRun	35
SignatureAnalyzerPrepHyper1Secondary	37
SignatureAnalyzerPrepHyper4	38
SourceSignatureAnalyzerCode	38
SynSigRun	39

CopyBestSignatureAnalyzerResult	<i>Find the SignatureAnalyzer results directory with the best results and make a copy of it as sa.results.dir/best.run/</i>
---------------------------------	---

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.

Diff4SynDataSets	<i>diff new directory / files against regression data for testing.</i>
------------------	--

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.

InstalldecompTumor2Sig

Install decompTumor2Sig from Bioconductor

Description

Install decompTumor2Sig from Bioconductor

Usage

```
InstalldecompTumor2Sig()
```

InstalldeconstructSigs

Install deconstructSigs from CRAN

Description

Install deconstructSigs from CRAN

Usage

```
InstalldeconstructSigs()
```

InstallmutSignatures *Install mutSignatures from github*

Description

Install mutSignatures from github

Usage

```
InstallmutSignatures()
```

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

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>

RealExposures	<i>Real exposure (signature attributions) from SignatureAnalyzer and SigProfiler</i>
---------------	--

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>

RundecompTumor2SigAttributeOnly

Run decompTumor2Sig attribution on a spectra catalog file and known signatures.

Description

Run decompTumor2Sig attribution on a spectra catalog file and known signatures.

Usage

```
RundecompTumor2SigAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  seedNumber = 1,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

input.catalog	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
gt.sigs.file	File containing input mutational signatures. Columns are signatures, rows are mutation types.
out.dir	Directory that will be created for the output; abort if it already exists. Log files will be in paste0(out.dir, "/tmp").
seedNumber	Specify the pseudo-random seed number used to run deconstructSigs. Setting seed can make the attribution of deconstructSigs repeatable. Default: 1.
test.only	If TRUE, only analyze the first 10 columns read in from input.catalog. Default: FALSE
overwrite	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in paste0(out.dir, "/sa.output.rdata"). These are TODO(Steve): list the files

Value

The inferred exposure of deconstructSigs, invisibly.

`RundeconstructSigsAttributeOnly`*Run deconstructSigs attribution on a spectra catalog file and known signatures.*

Description

Run deconstructSigs attribution on a spectra catalog file and known signatures.

Usage

```
RundeconstructSigsAttributeOnly(  
  input.catalog,  
  gt.sigs.file,  
  out.dir,  
  seedNumber = 1,  
  test.only = FALSE,  
  overwrite = FALSE  
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run deconstructSigs. Setting seed can make the attribution of deconstructSigs repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of deconstructSigs, invisibly.

RunhdpLessHier

*Run hdp extraction and attribution on a spectra catalog file***Description**

Run hdp extraction and attribution on a spectra catalog file

Usage

```
RunhdpLessHier(
  input.catalog,
  out.dir,
  CPU.cores = 1,
  seedNumber = 1,
  K.guess,
  multi.types = FALSE,
  remove.noise = FALSE,
  num.posterior = 4,
  post.burnin = 4000,
  post.n = 50,
  post.space = 50,
  post.cptiter = 3,
  test.only = FALSE,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

<code>input.catalog</code>	File containing a spectra catalog in ICAMS format.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running hdp_posterior .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run hdp. Setting seed can make the attribution of hdp repeatable. Default: 1.
<code>K.guess</code>	Suggested initial value of the number of signatures, passed to dp_activate as <code>initcc</code> .
<code>multi.types</code>	<p>A logical scalar or a character vector. If FALSE, hdp will regard all input spectra as one tumor type, and will allocate them to one single dirichlet process node.</p> <p>If TRUE, hdp will infer tumor types based on the string before ":" in their names. e.g. Tumor type for "SA.Syn.Ovary-AdenoCA::S.500" would be "SA.Syn.Ovary-AdenoCA"</p> <p>If it is a character vector, it should be a vector of case-sensitive tumor types. e.g. <code>c("SA.Syn.Ovary-AdenoCA", "SA.Syn.Ovary-AdenoCA", "SA.Syn.Kidney-RCC")</code>.</p>
<code>remove.noise</code>	<p>Whether to remove noise signature "hdp.0"? In normal cases scenarios, only few mutations will be assigned to noise signature.</p> <p>For result visualization and assessment of hdp package, select TRUE; for diagnostic purposes, select FALSE.</p>

num.posterior	Number of posterior sampling chains; can set to 1 for testing.
post.burnin	Pass to <code>hdp_posterior</code> burnin.
post.n	Pass to <code>hdp_posterior</code> n.
post.space	Pass to <code>hdp_posterior</code> space.
post.cptiter	Pass to <code>hdp_posterior</code> cptiter.
test.only	If TRUE, only analyze the first 10 columns in <code>input.catalog</code> .
overwrite	If TRUE, overwrite existing output.
verbose	If TRUE then message progress information.

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files
 TODO(Wuyang)

Value

The inferred exposure of hdp, invisibly.

Runmaftools	<i>Run maftools extraction ONLY on a spectra catalog file</i>
-------------	---

Description

WARNING: maftools can only do signature extraction!

Usage

```
Runmaftools(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  K.exact = NULL,
  K.range = NULL,
  nrun.est.K = 10,
  pConstant = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

input.catalog	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
out.dir	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
CPU.cores	Number of CPUs to use in running maftools. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (CPU.cores = NULL), the CPU.cores would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.

`K.exact`, `K.range`

`K.exact` is the exact value for the number of signatures active in spectra (`K`). Specify `K.exact` if you know exactly how many signatures are active in the `input.catalog`, which is the ICAMS-formatted spectra file.

`K.range` is A numeric vector (`K.min`, `K.max`) of length 2 which tell maftools to search the best signature number active in spectra, `K`, in this range of `Ks`. Specify `K.range` if you don't know how many signatures are active in the `input.catalog`.

WARNING: You must specify only one of `K.exact` or `K.range`!

Default: NULL

`nrun.est.K`

Number of NMF runs for each possible number of signature. This is used in the step to estimate the most plausible number of signatures in input spectra catalog.

NOTE: Unlike other NMF-based packages, parameter `nrun.extract` is hard-coded as 1.

`pConstant`

A small positive value (a.k.a. pseudocount) to add to every entry in the `input.catalog`. Specify a value ONLY if an "non-conformable arrays error" is raised.

`test.only`

If TRUE, only analyze the first 10 columns read in from `input.catalog`.

`overwrite`

If TRUE, overwrite existing output.

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files

TODO(Wuyang)

Value

The extracted signatures of maftools, invisibly.

RunmSigActAttributeOnly

Run mSigAct attribution on a spectra catalog file and known signatures.

Description

Run mSigAct attribution on a spectra catalog file and known signatures.

Usage

```
RunmSigActAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  CPU.cores = NULL,
  seedNumber = 1,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running sigfit. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (<code>CPU.cores = NULL</code>), the <code>CPU.cores</code> would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.
<code>seedNumber</code>	Specify the pseudo-random seed number used to run <code>mSigAct</code> . Setting seed can make the attribution of <code>mSigAct</code> repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of `mSigAct`, invisibly.

RunMutationalPatterns *Run MutationalPatterns extraction and attribution on a spectra catalog file*

Description

WARNING: MutationalPatterns can only do exposure attribution using SBS96 spectra catalog and signature catalog!

Usage

```
RunMutationalPatterns(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  K.exact = NULL,
  K.range = NULL,
  nrun.est.K = 10,
  nrun.extract = 200,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running <code>MutationalPatterns</code> . For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (<code>CPU.cores = NULL</code>), the <code>CPU.cores</code> would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.
<code>K.exact, K.range</code>	<p><code>K.exact</code> is the exact value for the number of signatures active in spectra (<code>K</code>). Specify <code>K.exact</code> if you know exactly how many signatures are active in the <code>input.catalog</code>, which is the ICAMS-formatted spectra file.</p> <p><code>K.range</code> is A numeric vector (<code>K.min, K.max</code>) of length 2 which tell <code>MutationalPatterns</code> to search the best signature number active in spectra, <code>K</code>, in this range of <code>Ks</code>. Specify <code>K.range</code> if you don't know how many signatures are active in the <code>input.catalog</code>.</p> <p>WARNING: You must specify only one of <code>K.exact</code> or <code>K.range</code>!</p> <p>Default: <code>NULL</code></p>
<code>nrun.est.K</code>	Number of NMF runs for each possible number of signature. This is used in the step to estimate the most plausible number of signatures in input spectra catalog.
<code>nrun.extract</code>	number of NMF runs for extracting signatures and inferring exposures.
<code>test.only</code>	If <code>TRUE</code> , only analyze the first 10 columns read in from <code>input.catalog</code> . Default: <code>FALSE</code>
<code>overwrite</code>	If <code>TRUE</code> , overwrite existing output. Default: <code>FALSE</code>

Details

Creates several files in `out.dir`. These are: `TODO(Steve)`: list the files

`TODO(Wuyang)`

NOTE: The seed is hard-coded in `MutationalPatterns` as 123456.

`pConstant` is hard-coded as 1e-04.

Value

A list contains:

- `$signature` extracted signatures,
- `$exposure` inferred exposures,

of `MutationalPatterns`, invisibly.

`RunMutationalPatternsAttributeOnly`*Run MutationalPatterns attribution on a spectra catalog file and known signatures.*

Description

Run MutationalPatterns attribution on a spectra catalog file and known signatures.

Usage

```
RunMutationalPatternsAttributeOnly(  
  input.catalog,  
  gt.sigs.file,  
  out.dir,  
  seedNumber = 1,  
  test.only = FALSE,  
  overwrite = FALSE  
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run MutationalPatterns. Setting seed can make the attribution of MutationalPatterns repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of MutationalPatterns, invisibly.

RunmutSignatures

*Run mutSignatures extraction and attribution on a spectra catalog file***Description**

Run mutSignatures extraction and attribution on a spectra catalog file

Usage

```
RunmutSignatures(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  seedNumber = 12345,
  K.exact = NULL,
  nrun.exact = 1000,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running decipherMutationalProcesses .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run mutSignatures. Setting seed can make the attribution of mutSignatures repeatable. Default: 1.
<code>K.exact</code>	<code>K.exact</code> is the exact value for the number of signatures active in spectra (<code>K</code>). Specify <code>K.exact</code> if you know exactly how many signatures are active in the <code>input.catalog</code> , which is the ICAMS-formatted spectra file.
<code>nrun.exact</code>	number of NMF runs for extracting signatures and inferring exposures.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files
 TODO(Wuyang)

Value

The inferred exposure of mutSignatures, invisibly.

`RunmutSignaturesAttributeOnly`*Run mutSignatures attribution on a spectra catalog file and known signatures.*

Description

Run mutSignatures attribution on a spectra catalog file and known signatures.

Usage

```
RunmutSignaturesAttributeOnly(  
  input.catalog,  
  gt.sigs.file,  
  out.dir,  
  seedNumber = 1,  
  test.only = FALSE,  
  overwrite = FALSE  
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run mutSignatures. Setting seed can make the attribution of mutSignatures repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of mutSignatures, invisibly.

RunmutSpec

*Run mutSpec extraction and attribution on a spectra catalog file***Description**

NOTE: mutSpec can only do exposure attribution using SBS96 spectra catalog and signature catalog!

Usage

```
RunmutSpec(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  seedNumber = 1,
  K.exact = NULL,
  K.range = NULL,
  nrun.est.K = 50,
  nrun.extract = 200,
  pConstant = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

input.catalog	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
out.dir	Directory that will be created for the output; abort if it already exists. Log files will be in paste0(out.dir, "/tmp").
CPU.cores	Number of CPUs to use in running mutSpec. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (CPU.cores = NULL), the CPU.cores would be equal to (parallel::detectCores())/2, total number of CPUs divided by 2.
seedNumber	Specify the pseudo-random seed number used to run mutSpec. Setting seed can make the attribution of mutSpec repeatable. Default: 1.
K.exact, K.range	<p>K.exact is the exact value for the number of signatures active in spectra (K). Specify K.exact if you know exactly how many signatures are active in the input.catalog, which is the ICAMS-formatted spectra file.</p> <p>K.range is A numeric vector (K.min, K.max) of length 2 which tell mutSpec to search the best signature number active in spectra, K, in this range of Ks. Specify K.range if you don't know how many signatures are active in the input.catalog.</p> <p>WARNING: You must specify only one of K.exact or K.range!</p> <p>Default: NULL</p>
nruntime.est.K	Number of NMF runs for each possible number of signature. This is used in the step to estimate the most plausible number of signatures in input spectra catalog.
nruntime.extract	number of NMF runs for extracting signatures and inferring exposures.

pConstant	A small positive value (a.k.a. pseudocount) to add to every entry in the <code>input.catalog</code> . Specify a value ONLY if an "non-conformable arrays error" is raised.
test.only	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
overwrite	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `out.dir`. These are:

- `extracted.signatures.csv`
- `inferred.exposures.csv`
- `sessionInfo`

Value

The inferred exposure of `mutSpec`, invisibly.

Runsigfit	<i>Run sigfit extraction and attribution on a spectra catalog file</i>
-----------	--

Description

WARNING: sigfit can only do exposure attribution using SBS96 spectra catalog and signature catalog!

Usage

```
Runsigfit(
  input.catalog,
  out.dir,
  model = "nmf",
  CPU.cores = NULL,
  seedNumber = 1,
  K.exact = NULL,
  K.range = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>model</code>	Algorithm to be used to extract signatures and attribute exposures. Only "nmf" or "emu" is valid. Default: "nmf".

CPU.cores	Number of CPUs to use in running sigfit. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (CPU.cores = NULL), the CPU.cores would be equal to (parallel::detectCores())/2, total number of CPUs divided by 2.
seedNumber	Specify the pseudo-random seed number used to run sigfit. Setting seed can make the attribution of sigfit repeatable. Default: 1.
K.exact, K.range	<p>K.exact is the exact value for the number of signatures active in spectra (K). Specify K.exact if you know exactly how many signatures are active in the input.catalog, which is the ICAMS-formatted spectra file.</p> <p>K.range is A numeric vector (K.min,K.max) of length 2 which tell sigfit to search the best signature number active in spectra, K, in this range of Ks. Specify K.range if you don't know how many signatures are active in the input.catalog. K.max - K.min >= 3, otherwise an error will be thrown.</p> <p>WARNING: You must specify only one of K.exact or K.range!</p> <p>Default: NULL</p>
test.only	If TRUE, only analyze the first 10 columns read in from input.catalog. Default: FALSE
overwrite	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in out.dir. These are: TODO(Steve): list the files
 TODO(Wuyang)

Value

The inferred exposure of sigfit, invisibly.

RunsigfitAttributeOnly

Run sigfit attribution on a spectra catalog file and known signatures.

Description

Run sigfit attribution on a spectra catalog file and known signatures.

Usage

```
RunsigfitAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  model = "nmf",
  seedNumber = 1,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>model</code>	Algorithm to be used to extract signatures and attribute exposures. Only "nmf" or "emu" is valid. Default: "nmf".
<code>seedNumber</code>	Specify the pseudo-random seed number used to run sigfit. Setting seed can make the attribution of sigfit repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of sigfit, invisibly.

Runsigminer

Run sigminer extraction and attribution on a spectra catalog file

Description

Run sigminer extraction and attribution on a spectra catalog file

Usage

```
Runsigminer(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  seedNumber = 1,
  K.max = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running sigminer. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (<code>CPU.cores = NULL</code>), the <code>CPU.cores</code> would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.
<code>seedNumber</code>	Specify the pseudo-random seed number used to run SomaticSignatures. Setting seed can make the attribution of SomaticSignatures repeatable. Default: 1.
<code>K.max</code>	<code>K.max</code> is the maximum number of signatures users expect to active in <code>input.catalog</code> . As this approach cannot specify <code>K.exact</code> , you can specify <code>K.max = K.exact</code> If you know exactly how many signatures are active in the <code>input.catalog</code> . On the other hand, you may specify <code>max(K.range)</code> if you don't know how many signatures are active in the <code>input.catalog</code> .
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `out.dir`. These are: `TODO(Steve)`: list the files
`TODO(Wuyang)`

Value

A list contains:

- `$signature` extracted signatures,
- `$exposure` inferred exposures,

of sigminer, invisibly.

RunSignatureAnalyzerAttribution

Run SignatureAnalyzer attribution on a catalog file and output by RunSignatureAnalyzerOnFile().

Description

Normally, please call [SignatureAnalyzerOneRun](#) instead of this function.

Usage

```
RunSignatureAnalyzerAttribution(
  input.catalog,
  read.catalog.function,
  extracted.signature.file,
  raw.exposures.file,
  write.signature.function,
  out.dir,
  test.only = FALSE,
  input.exposures = NULL,
  delete.tmp.files = TRUE,
  overwrite = FALSE,
  verbose = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input catalog. Columns are samples (tumors), rows are signatures. SignatureAnalyzer does not care about the row names (I think) TODO(Steve): check this.
<code>read.catalog.function</code>	Function taking a file path as its only argument and returning a catalog as a numeric matrix.
<code>extracted.signature.file</code>	A .csv file containing extracted signatures. Normally, this file is named "sa.output.sigs.csv" and is generated by function RunSignatureAnalyzerOnFile(). It expects to have the same format as the <code>input.catalog</code> , thus it will be read by <code>read.catalog.function</code> too.
<code>raw.exposures.file</code>	A .csv file containing raw attributions of exposures. Normally, this file is named "sa.output.raw.exp.csv" and is generated by function RunSignatureAnalyzerOnFile().
<code>write.signature.function</code>	Function with first argument the signatures generated by SignatureAnalyzer and second argument the file to write to.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> .
<code>input.exposures</code>	A file with the synthetic exposures used to generate <code>input.catalog</code> ; if provided here, this is copied over to the output directory for downstream analysis.
<code>delete.tmp.files</code>	If TRUE delete the many temporary files generated by SignatureAnalyzer.
<code>overwrite</code>	If TRUE, overwrite existing output.
<code>verbose</code>	If TRUE cat a message regarding progress.

Details

Save the final attribution of a catalog matrix into a file named "sa.output.fine.exp.csv" under the folder `out.dir`.

Value

The final attribution matrix. (i.e. `exp.fine.tuned`)

RunSignatureAnalyzerOnFile

Run SignatureAnalyzer on a file containing a catalog AFTER the SignatureAnalyzer code has been source'ed.

Description

Normally, please call [SignatureAnalyzerOneRun](#) instead of this function.

Usage

```
RunSignatureAnalyzerOnFile(
  input.catalog,
  out.dir,
  input.exposures = NULL,
  maxK = 30,
  tol = 1e-07,
  test.only = FALSE,
  delete.tmp.files = TRUE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input catalog. Columns are samples (tumors), rows are signatures. SignatureAnalyzer does not care about the row names (I think) TODO(Steve): check this.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>input.exposures</code>	A file with the synthetic exposures used to generate <code>input.catalog</code> ; if provided here, this is copied over to the output directory for downstream analysis.
<code>maxK</code>	The maximum number of signatures to consider extracting.
<code>tol</code>	Controls when SignatureAnalyzer will terminate its search; <code>tol</code> was 1.e-05 for the PCAWG7 analysis.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> .
<code>delete.tmp.files</code>	If TRUE delete the many temporary files generated by SignatureAnalyzer.
<code>overwrite</code>	If TRUE, overwrite existing output

Details

Creates several files in `out.dir`:

1. `sa.output.sigs.csv` Normalized signatures (no all-0 signatures, column sums all 0)
2. `sa.output.raw.exp.csv` Raw exposures (attributions)
3. `sa.output.exp.csv` Same as `sa.output.raw.exp.csv`
4. `sa.output.other.data.csv`, contains a summary of important information, including the number of signatures extracted.
5. `input.syn.exp.csv` Optional, a copy of `input.exposures`, if it was provided.

Value

A list with the following elements:

1. `signatures.W` The raw signature matrix, *including* columns of all zeros.
2. `exposures.H` The raw exposure matrix, *excluding* rows of all zeros. The matrix product of the non-zero columns of `signatures.w` and `exposures.H` approximates the input spectrum matrix.
3. `likelihood` The likelihood as returned by `SignatureAnalyzer`.
4. `evidence` $-1 \times$ the posterior probability as returned by `SignatureAnalyzer`.
5. `relevance` One for each column of the `signatures.W`, as returned by `SignatureAnalyzer`.
6. `error` A measure of reconstruction error (?) as returned by `SignatureAnalyzer`
7. `normalized.sigs` The non-0 columns of `signatures.W` normalized so that each column sum is 1.

RunSignatureEstimationQPAttributeOnly

Run SignatureEstimation Quadratic Programming (QP) attribution on a spectra catalog file and known signatures.

Description

Run SignatureEstimation Quadratic Programming (QP) attribution on a spectra catalog file and known signatures.

Usage

```
RunSignatureEstimationQPAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  seedNumber = 1,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run SignatureEstimation. Setting seed can make the attribution of SignatureEstimation repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Value

Invisibly returns a list which contains:

- `$exposuresCounts`: the exposure counts inferred in ICAMSxtra format,
- `$exposureErrors`: the MSE in ICAMSxtra format,
- `$SEoutput`: A list which contains:
 - `$exposures`: exposure proportion in SignatureEstimation format, and errors invisibly.
 - `$errors`: mean squared error (MSE) between normalized reconstructed spectra and normalized ground-truth mutational spectra.

RunSignatureEstimationSAAttributeOnly

Run SignatureEstimation Simulated Annealing (SA) attribution on a spectra catalog file and known signatures.

Description

Run SignatureEstimation Simulated Annealing (SA) attribution on a spectra catalog file and known signatures.

Usage

```
RunSignatureEstimationSAAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  seedNumber = 1,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run <code>SignatureEstimation</code> . Setting seed can make the attribution of <code>SignatureEstimation</code> repeatable. Default: 1.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Value

Invisibly returns a list which contains:

- `$exposuresCounts`: the exposure counts inferred in ICAMSxtra format,
- `$exposureErrors`: the MSE in ICAMSxtra format,
- `$SEoutput`: A list which contains:
 - `$exposures`: exposure proportion in `SignatureEstimation` format, and errors invisibly.
 - `$errors`: mean squared error (MSE) between normalized reconstructed spectra and normalized ground-truth mutational spectra.

RunsigneR

Run signeR extraction and attribution on a spectra catalog file

Description

Run signeR extraction and attribution on a spectra catalog file

Usage

```
RunsigneR(
  input.catalog,
  out.dir,
  seedNumber = 1,
  K.exact = NULL,
  K.range = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run <code>signeR</code> . Setting seed can make the attribution of <code>signeR</code> repeatable. Default: 1.
<code>K.exact</code> , <code>K.range</code>	<code>K.exact</code> is the exact value for the number of signatures active in spectra (<code>K</code>). Specify <code>K.exact</code> if you know exactly how many signatures are active in the <code>input.catalog</code> , which is the ICAMS-formatted spectra file. <code>K.range</code> is A numeric vector (<code>K.min</code> , <code>K.max</code>) of length 2 which tell <code>signeR</code> to search the best signature number active in spectra, <code>K</code> , in this range of <code>Ks</code> . Specify <code>K.range</code> if you don't know how many signatures are active in the <code>input.catalog</code> . WARNING: You must specify only one of <code>K.exact</code> or <code>K.range</code> ! Default: NULL
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files
TODO(Wuyang)

Value

The inferred exposure of `signeR`, invisibly.

RunSomaticSignatures	<i>Run SomaticSignatures.NMF extraction and attribution on a spectra catalog file</i>
----------------------	---

Description

Run SomaticSignatures.NMF extraction and attribution on a spectra catalog file

Usage

```
RunSomaticSignatures(
  input.catalog,
  out.dir,
  CPU.cores = NULL,
  seedNumber = 1,
  K.exact = NULL,
  K.range = NULL,
  nrun.est.K = 30,
  nrun.extract = 1,
  pConstant = NULL,
```

```

    save.diag = FALSE,
    test.only = FALSE,
    overwrite = FALSE
)

```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>CPU.cores</code>	Number of CPUs to use in running <code>SomaticSignatures.NMF</code> . For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (<code>CPU.cores = NULL</code>), the <code>CPU.cores</code> would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.
<code>seedNumber</code>	Specify the pseudo-random seed number used to run <code>SomaticSignatures</code> . Setting seed can make the attribution of <code>SomaticSignatures</code> repeatable. Default: 1.
<code>K.exact, K.range</code>	<p><code>K.exact</code> is the exact value for the number of signatures active in spectra (<code>K</code>). Specify <code>K.exact</code> if you know exactly how many signatures are active in the <code>input.catalog</code>, which is the ICAMS-formatted spectra file.</p> <p><code>K.range</code> is A numeric vector (<code>K.min, K.max</code>) of length 2 which tell <code>SomaticSignatures.NMF</code> to search the best signature number active in spectra, <code>K</code>, in this range of <code>Ks</code>. Specify <code>K.range</code> if you don't know how many signatures are active in the <code>input.catalog</code>.</p> <p>WARNING: You must specify only one of <code>K.exact</code> or <code>K.range</code>!</p> <p>Default: <code>NULL</code></p>
<code>nrun.est.K</code>	Number of NMF runs for each possible number of signature. This is used in the step to estimate the most plausible number of signatures in input spectra catalog.
<code>nrun.extract</code>	number of NMF runs for extracting signatures and inferring exposures.
<code>pConstant</code>	A small positive value (a.k.a. pseudocount) to add to every entry in the <code>input.catalog</code> . Specify a value ONLY if an "non-conformable arrays error" is raised.
<code>save.diag</code>	<p>Save object of class <code>MutationalSignatures</code> which stores full results from multiple NMF decomposition runs into files below:</p> <ul style="list-style-type: none"> • <code>assess.K.pdf</code> RSS and explained variance at each <code>K</code> in <code>K.range</code>. Used for manual selection of number of signatures (<code>K</code>). • <code>assess.K.Rdata</code> Full results for each <code>K</code> in <code>K.range</code>. Used for diagnosing goodness of fit and stability. • <code>extract.given.K.Rdata</code> Full results when <code>K</code> is specified by <code>K.exact</code> or selected by elbow-point method. Used for diagnosing accuracy of signature extraction. <p>Set to <code>TRUE</code> for diagnostic purposes, set to <code>FALSE</code> for cleaner results.</p>
<code>test.only</code>	If <code>TRUE</code> , only analyze the first 10 columns read in from <code>input.catalog</code> . Default: <code>FALSE</code>
<code>overwrite</code>	If <code>TRUE</code> , overwrite existing output. Default: <code>FALSE</code>

Details

SomaticSignatures.NMF used approach in Hutchins et al. (2008) to estimate K: it selects the first inflection point of residual sum of squares (RSS) function by finding the smallest K where the second derivate of RSS at its neighbouring Ks have opposite signs.

This requires calculation of second derivative of residual sum of squares (RSS) at >2 integers, and thus requires at least 3 Ks to be assessed.

Value

A list contains:

- \$signature extracted signatures,
- \$exposure inferred exposures,

of SomaticSignatures.NMF, invisibly.

References

<http://dx.doi.org/10.1093/bioinformatics/btn526>

RunSparseSignatures	<i>Run SparseSignatures extraction and attribution on a spectra catalog file</i>
---------------------	--

Description

Run SparseSignatures extraction and attribution on a spectra catalog file

Usage

```
RunSparseSignatures(
  input.catalog,
  out.dir,
  seedNumber = 1,
  K.exact = NULL,
  K.range = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```

Arguments

input.catalog	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
out.dir	Directory that will be created for the output; abort if it already exists. Log files will be in paste0(out.dir, "/tmp").
seedNumber	Specify the pseudo-random seed number used to run SparseSignatures. Setting seed can make the attribution of SparseSignatures repeatable. Default: 1.

`K.exact`, `K.range`

`K.exact` is the exact value for the number of signatures active in spectra (`K`). Specify `K.exact` if you know exactly how many signatures are active in the `input.catalog`, which is the ICAMS-formatted spectra file.

`K.range` is A numeric vector (`K.min`,`K.max`) of length 2 which tell SparseSignatures to search the best signature number active in spectra, `K`, in this range of `Ks`. Specify `K.range` if you don't know how many signatures are active in the `input.catalog`.

WARNING: You must specify only one of `K.exact` or `K.range`!

Default: NULL

`test.only` If TRUE, only analyze the first 10 columns read in from `input.catalog`. Default: FALSE

`overwrite` If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files

TODO(Wuyang)

Value

The inferred exposure of SparseSignatures, invisibly.

Runtcsm

Run tcsn extraction and attribution on a spectra catalog file

Description

Run tcsn extraction and attribution on a spectra catalog file

Usage

```
Runtcsm(
  input.catalog,
  out.dir,
  seedNumber = 1,
  CPU.cores = 1,
  K.exact = NULL,
  K.range = NULL,
  covariates = NULL,
  test.only = FALSE,
  overwrite = FALSE,
  feature.file = NULL,
  effect.output.file = NULL,
  sigma.output.file = NULL,
  gamma.output.file = NULL
)
```

Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run tcsn. Setting seed can make the attribution of tcsn repeatable.
<code>CPU.cores</code>	Number of CPUs to use in running MutationalPatterns. For a server, 30 cores would be a good choice; while for a PC, you may only choose 2-4 cores. By default (<code>CPU.cores = NULL</code>), the <code>CPU.cores</code> would be equal to <code>(parallel::detectCores())/2</code> , total number of CPUs divided by 2.
<code>K.exact, K.range</code>	<code>K.exact</code> is the exact value for the number of signatures active in spectra (<code>K</code>). Specify <code>K.exact</code> if you know exact how many signatures are active in the <code>input.catalog</code> , which is the ICAMS-formatted spectra file. <code>K.range</code> is A numeric vector (<code>K.min, K.max</code>) of length 2 which tell tcsn to search the best signature number active in spectra, <code>K</code> , in this range of <code>Ks</code> . Specify <code>K.range</code> if you don't know how many signatures are active in the <code>input.catalog</code> . <code>K.max - K.min >= 3</code> , otherwise an error will be thrown. WARNING: You must specify only one of <code>K</code> or <code>K.range</code> !
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> .
<code>overwrite</code>	If TRUE, overwrite existing output.

Details

Creates several files in `out.dir`. These are: TODO(Steve): list the files
TODO(Wuyang)

Value

The inferred exposure of tcsn, invisibly.

RunYAPSAAttributeOnly *Run YAPSA attribution on a spectra catalog file and known signatures.*

Description

Run YAPSA attribution on a spectra catalog file and known signatures.

Usage

```
RunYAPSAAttributeOnly(
  input.catalog,
  gt.sigs.file,
  out.dir,
  seedNumber = 1,
  signature.cutoff = NULL,
  test.only = FALSE,
  overwrite = FALSE
)
```


Arguments

<code>input.catalog</code>	File containing input spectra catalog. Columns are samples (tumors), rows are mutation types.
<code>gt.sigs.file</code>	File containing input mutational signatures. Columns are signatures, rows are mutation types.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run YAPSA. Setting seed can make the attribution of YAPSA repeatable. Default: 1.
<code>signature.cutoff</code>	A numeric vector of values less than 1. Signatures from within W with an overall exposure less than the respective value in <code>in_cutoff_vector</code> will be discarded. Default: vector length of number of sigs with all zeros
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> . Default: FALSE
<code>overwrite</code>	If TRUE, overwrite existing output. Default: FALSE

Details

Creates several files in `paste0(out.dir, "/sa.output.rdata")`. These are TODO(Steve): list the files

Value

The inferred exposure of YAPSA, invisibly.

<code>SAMultiRunOneCatalog</code>	<i>Run SignatureAnalyzer many times on one catalog and put results in specified location.</i>
-----------------------------------	---

Description

Run SignatureAnalyzer many times on one catalog and put results in specified location.

Usage

```
SAMultiRunOneCatalog(
  num.runs,
  signatureanalyzer.code.dir,
  input.catalog,
  out.dir,
  maxK = 30,
  tol = 1e-07,
  test.only = FALSE,
  delete.tmp.files = TRUE,
  overwrite = FALSE,
  mc.cores = 1,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

num.runs	The number of times run SignatureAnalyzer on each catalog (matrix of mutational spectra).
signatureanalyzer.code.dir	The directory holding the SignatureAnalyzer code.
input.catalog	The catalog to analyze.
out.dir	Root of directory tree that will contain the results.
maxK	The maximum number of signatures to consider extracting.
tol	Controls when SignatureAnalyzer will terminate its search; tol was 1.e-05 for the PCAWG7 analysis.
test.only	If TRUE, only analyze the first 10 columns read in from input.catalog.
delete.tmp.files	If TRUE delete the many temporary files generated by SignatureAnalyzer.
overwrite	If TRUE overwrite previous results in same directory tree.
mc.cores	Number of cores to use for mclapply; ignored on Windows.
verbose	If TRUE cat a message regarding progress.
seed	If not NULL call RNGkind(kind = "L'Ecuyer-CMRG"); set.seed(seed).

SignatureAnalyzer4MatchedCatalogs

Run SignatureAnalyzer on 4 coordinated data sets and put results in specified location.

Description

Run SignatureAnalyzer on 4 coordinated data sets and put results in specified location.

Usage

```
SignatureAnalyzer4MatchedCatalogs(
  num.runs = 20,
  signatureanalyzer.code.dir,
  dir.root,
  maxK = 30,
  tol = 1e-07,
  test.only = FALSE,
  delete.tmp.files = TRUE,
  slice = 1:4,
  overwrite = FALSE,
  mc.cores = 1
)
```

Arguments

num.runs	Number of SignatureAnalyzer runs per data set.
signatureanalyzer.code.dir	The directory holding the SignatureAnalyzer code.
dir.root	Root of directory tree that contains the input data and to which the results will be written.
maxK	The maximum number of signatures to consider extracting.
tol	Controls when SignatureAnalyzer will terminate its search; tol was 1.e-05 for the PCAWG7 analysis.
test.only	If TRUE, only analyze the first 10 columns read in from input.catalog.
delete.tmp.files	If TRUE delete the many temporary files generated by SignatureAnalyzer.
slice	Vector of integers from 1:4. Only run on the corresponding data set (see Details).
overwrite	if TRUE overwrite preexisting results.
mc.cores	The number of cores to use with mclapply; automatically overridden to 1 on Windows.

Details

The 4 coordinated data sets are

1. sa.sa.96
2. sp.sp
3. sa.sa.COMPOSITE
4. sp.sa.COMPOSITE

which are described elsewhere.

SignatureAnalyzerOneRun

Source SignatureAnalyzer and run it once on a single data set and put results in specified location.

Description

Source SignatureAnalyzer and run it once on a single data set and put results in specified location.

Usage

```
SignatureAnalyzerOneRun(
  signatureanalyzer.code.dir,
  input.catalog,
  out.dir,
  seedNumber = NULL,
  input.exposures = NULL,
  maxK = 30,
  tol = 1e-07,
```

```

    test.only = FALSE,
    delete.tmp.files = TRUE,
    verbose = 0,
    overwrite = FALSE
)

```

Arguments

<code>signatureanalyzer.code.dir</code>	The directory holding the SignatureAnalyzer code.
<code>input.catalog</code>	File containing input catalog. Columns are samples (tumors), rows are signatures. SignatureAnalyzer does not care about the row names (I think) TODO(Steve): check this.
<code>out.dir</code>	Directory that will be created for the output; abort if it already exists. Log files will be in <code>paste0(out.dir, "/tmp")</code> .
<code>seedNumber</code>	Specify the pseudo-random seed number used to run SignatureAnalyzer. Setting seed can make the attribution of SignatureAnalyzer repeatable. If NULL, this function will not specify seed number. Default: NULL.
<code>input.exposures</code>	A file with the synthetic exposures used to generate <code>input.catalog</code> ; if provided here, this is copied over to the output directory for downstream analysis.
<code>maxK</code>	The maximum number of signatures to consider extracting.
<code>tol</code>	Controls when SignatureAnalyzer will terminate its search; <code>tol</code> was 1.e-05 for the PCAWG7 analysis.
<code>test.only</code>	If TRUE, only analyze the first 10 columns read in from <code>input.catalog</code> .
<code>delete.tmp.files</code>	If TRUE delete the many temporary files generated by SignatureAnalyzer.
<code>verbose</code>	If TRUE, then print various messages.
<code>overwrite</code>	If TRUE, overwrite existing output

Details

Creates several files in `out.dir`:

1. `sa.output.sigs.csv` Normalized signatures (no all-0 signatures, column sums all 0)
2. `sa.output.raw.exp.csv` Raw exposures (attributions)
3. `sa.output.exp.csv` Same as `sa.output.raw.exp.csv`
4. `sa.output.other.data.csv`, contains a summary of important information, including the number of signatures extracted.
5. `input.syn.exp.csv` Optional, a copy of `input.exposures`, if it was provided.

Value

A list with the following elements:

1. `signatures.W` The raw signature matrix, **including** columns of all zeros.
2. `exposures.H` The raw exposure matrix, **excluding** rows of all zeros. The matrix product of the non-zero columns of `signatures.w` and `exposures.H` approximates the input spectrum matrix.

3. likelihood The likelihood as returned by SignatureAnalyzer.
4. evidence -1 * the posterior probability as returned by SignatureAnalyzer.
5. relevance One for each column of the signatures.W, as returned by SignatureAnalyzer.
6. error A measure of reconstruction error (?) as returned by SignatureAnalyzer
7. normalized.sigs The non-0 columns of signatures.W normalized so that each column sum is 1.

SignatureAnalyzerPrepHyper1Secondary

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

Description

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

Description

Prepare the "hypermuted" 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-hypermuted 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.

SourceSignatureAnalyzerCode

Source SignatureAnalyzer Codes.

Description

Source SignatureAnalyzer Codes.

Usage

SourceSignatureAnalyzerCode(signatureanalyzer.code.dir)

Arguments

- signatureanalyzer.code.dir

The directory which stores SignatureAnalyzer program files. It must include a folder named INPUT_SignatureAnalyzer and a R script named SignatureAnalyer.PCAWG.function

SynSigRun	<i>SynSigRun: An easy-to-use package for non-experts which runs software packages reproducibly with synthetic tumors generated by SynSigGen.</i>
-----------	--

Description

SynSigRun gives necessary information to mutational-signature analysis programs. These programs used catalogs of synthetic mutational spectra created by package SynSigGen, and results were assessed by SynSigEval.

Workflow

Typical workflow for conducting a mutational signature analysis with mutational spectra is as follows.

Input mutational spectra:

Mutational spectra can be obtained from vcf files of real samples (see "Importing mutational spectra from ICAMS"). Mutational spectra can also be generated in-silico by R package SynSigGen, and then imported by ICAMS (see "(In SynSigGen) Creating Synthetic Mutational Spectra").

Importing mutational spectra from ICAMS:

Relevant functions are:

1. ReadCatalog
2. StrelkaSBSVCFFilesToCatalog
3. StrelkaIDVCFFilesToCatalog
4. MutectVCFFilesToCatalog

See ICAMS package documentation for more details.

(In SynSigGen) Creating Synthetic Mutational Spectra:

These functions create synthetic mutational spectra based on parameters derived from mutational signature profiles and exposures.

Relevant functions for generate exposures are:

1. GenerateSynFromReal
2. GenerateSyntheticExposures
3. GenSBS1SBS5Exposure

After generating exposures for spectra dataset, SynSigGen used these functions to generate mutational spectra:

1. CreateFromReal
2. CreateMixedTumorTypeSyntheticData
3. CreateRandomSyn

See SynSigGen package documentation for more details.

(In SynSigRun) Run mutational analysis computational approaches:

Relevant functions are:

1. [RunhdpLessHier](#)
2. [Runmaftools](#)
3. [RunMutationalPatterns](#)
4. [Runsigner](#)

5. Runtcsm

(In SynSigEval) Summarize results:

Summarize results of of signature extraction and exposure inference (a.k.a. signature attribution):

Relevant functions are:

1. SummarizeSigOnehelmsmanSubdir
2. SignatureAnalyzerSummarizeTopLevel
3. SignatureAnalyzerSummarizeSBS1SBS5
4. SummarizeSigOneSigProExtractorSubdir
5. SummarizeSigProExtractor
6. SummarizeSigOneExtrAttrSubdir

Package SynSigEval uses functions in ICAMSxtra to compare two sets of mutational signatures. Often we will be interested in comparing signature profiles extracted from synthetic data to the ground-truth signature profiles:

1. Match1Sig
2. MatchSigs1Direction
3. MatchSigs2Directions
4. MatchSigsAndRelabel

Folder structure for epcamcze5xRPzOT5TWNiW5FJs4S8D5Qy-38- and epcamcze5xRPzOT5TWNiW5FJs4S8D5Qy-39-:

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.pancreas"). 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`:

1. It can be ("Attr") for storing results of packages which can only do signature attribution of known signatures ("Attr");
2. It can be ("ExtrAttr"), folder to store results of computational approaches which can do de-novo extraction and following attribution, without knowing the number of ground-truth mutational signatures active in the spectra data set.
3. It can also be ("ExtrAttrExact"), folder to store results of computational approaches which can do de-novo extraction and following attribution, given the number of ground-truth mutational signatures active in the spectra data set.

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

Fifth level - `run.dir`: contains results from a run of the computational approach using a specific number of seed. (e.g. "seed.1")

Index

* datasets

MutationalSignatures, 6
RealExposures, 7

BestSignatureAnalyzerResult, 3

CopyBestSignatureAnalyzerResult, 3

decipherMutationalProcesses, 16

Diff4SynDataSets, 3

dp_activate, 10

FixSASigNames, 4

hdp_posterior, 10, 11

ICAMS, 10

InstalldecompTumor2Sig, 4

InstalldeconstructSigs, 4

InstallmutSignatures, 5

MapSPToSASignatureNamesInExposure, 5

MutationalSignatures, 6

RealExposures, 7

RundecompTumor2SigAttributeOnly, 8

RundeconstructSigsAttributeOnly, 9

RunhdpLessHier, 10, 39

Runmaftools, 11, 39

RunmSigActAttributeOnly, 12

RunMutationalPatterns, 13, 39

RunMutationalPatternsAttributeOnly, 15

RunmutSignatures, 16

RunmutSignaturesAttributeOnly, 17

RunmutSpec, 18

Runsifit, 19

RunsifitAttributeOnly, 20

Runsifminer, 21

RunSignatureAnalyzerAttribution, 22

RunSignatureAnalyzerOnFile, 24

RunSignatureEstimationQPAttributeOnly,
25

RunSignatureEstimationSAAttributeOnly,
26

RunsigneR, 27, 39

RunSomaticSignatures, 28

RunSparseSignatures, 30

Runtcsm, 31, 40

RunYAPSAAttributeOnly, 32

sa.96.sigs, 5

sa.96.sigs (MutationalSignatures), 6

sa.all.real.exposures (RealExposures), 7

sa.COMPOSITE.sigs
(MutationalSignatures), 6

sa.DBS.sigs (MutationalSignatures), 6

sa.ID.sigs (MutationalSignatures), 6

sa.no.hyper.real.exposures
(RealExposures), 7

SAMultiRunOneCatalog, 33

SignatureAnalyzer4MatchedCatalogs, 34

SignatureAnalyzerOneRun, 22, 24, 35

SignatureAnalyzerPrepHyper1Secondary,
37

SignatureAnalyzerPrepHyper4, 38

SourceSignatureAnalyzerCode, 38

sp.all.real.exposures (RealExposures), 7

sp.no.hyper.real.exposures
(RealExposures), 7

sp.sigs, 5

sp.sigs (MutationalSignatures), 6

SynSigRun, 39