Package 'CNSigs'

January 8, 2026

Type Package

Title Analysis of Copy Number Signatures

Version 0.1.0

Maintainer Shawn Striker <striker.35@osu.edu>

Description A workflow to generate and analyze signatures based on copy number data using non-negative matrix factorization (NMF) in an approach similar to that used in mutational signatures. It can be used to extract features from Copy number segment data and use that to find a subset of copy number signatures which can be further used to correlate with other relevant data. For more on 'NMF' see Gaujoux (2013) <doi:10.1186/1471-2105-11-367>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 2.10), NMF

Imports methods, doParallel, foreach, flexmix, limSolve, ggplot2, snow, cowplot, pheatmap, RColorBrewer, viridisLite, colorspace, stats, utils

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author David Tallman [aut], Shawn Striker [cre, ctb], Daniel Stover [cph]

Repository CRAN

Date/Publication 2026-01-08 19:30:19 UTC

Contents

addPloidyData 3

cancerComps . . . 3
cancerSigs . . . 4
checkCompOverlap . . . 4
collapsedSigs . . . 5
compareExposures . . . 5
compsExp . . . 6
createSigs . . . 6
defaultFeats . . . 7
determineNumSigs . . . 7
detSigNumPipeline . . . 8
diffCompSigSim . . . 9
extractCNFeats . . . 10
extractSegsize . . . 11
featsExp . . . 12
findExposures . . . 12
fitComponent . . . 13
fitModels . . . 14
generateSCM . . . 15
matchSigs . . . 16
plotComp . . . 17
plotComps . . . 17
plotScm . . . 18
plotSegs . . . 19
plotSig . . . 20
plotSigExposure . . . 20
plotSigExposureMat . . . 22
plotSigMat . . . 23
plotSigs . . . 23
postProb . . . 24
readSegs . . . 24
reducePeaks . . . 25
referenceExp . . . 25
remapResults . . . 26
runPipeline . . . 27
scmExp . . . 28
segDataExp . . . 29
segStats . . . 29
sigExposExp . . . 30
sigsExp . . . 30
sigSim . . . 31
smoothSegs . . . 31
sumOfPosteriors . . . 32
validateSegData . . . 32

Index **34**

`addPloidyData` *addPloidyData*

Description

This function is used to append your ploidy data onto the sample component matrix. The function is called in runPipeline if you specify ploidy as a desired feature and give a vector of ploidy values.

Usage

```
addPloidyData(scm, ploidyData)
```

Arguments

`scm`	The scm to append the ploidy to
`ploidyData`	Vector of ploidy date to append to segs

Value

Returns scm including the ploidy data

`cancerComps` *Components derived from TCGA*

Description

These generated components were derived from all of the copy number data available from TCGA. It is meant to be used to look for signatures in cancer data so that you do not have to model new data, and so that you can easily compare signatures

Usage

```
cancerComps
```

Format

A list with 6 flexmix objects:

segsize Mixture of normal distributions

bp10MB Mixture of normal distributions

osCN Mixture of poisson distributions

changepoint Mixture of normal distributions

copynumber Mixture of normal distributions

bpchrarm Mixture of poisson distributions

cancerSigs *Signatures derived from TCGA*

Description

These signatures were derived from all of the copy number data available from TCGA. It is meant to be used so that you can compare newly found signatures to these described signatures. There are a total of 25 signatures. The components used to derive these signatures can be found in the cancerComps variable.

Usage

```
cancerSigs
```

Format

An object of class `matrix` (inherits from `array`) with 28 rows and 25 columns.

checkCompOverlap *checkCompOverlap*

Description

This function is used to check for overlapping components and remove them if they overlap. The mixed modeling can sometimes run into an error in which it produces multiple essentially identical components. This function attempts to find and remove these duplicates.

Usage

```
checkCompOverlap(comps, pois = FALSE)
```

Arguments

`comps`	Component parameters
`pois`	Whether or not the components are poisson or normal distributions

Value

Returns a components with no overlaps

collapsedSigs *Signatures derived from TCGA and collapsed.*

Description

These signatures were derived from all of the copy number data available from TCGA. These pan-cancer signatures were collapsed by similarity to a total of 13 signatures and were built using ploidy data. It is meant to be used so that you can compare newly found signatures to these described signatures.

Usage

```
collapsedSigs
```

Format

An object of class `data.frame` with 29 rows and 13 columns.

compareExposures *compareExposures*

Description

This function is used to check two signature sets in order to compare how the samples exposures differ across the two runs.

Usage

```
compareExposures(reference, toCompare)
```

Arguments

`reference`	Results from your reference analysis
`toCompare`	Results from run that you want to compare to reference

Value

Prints out the difference in signature exposures

Examples

```
compareExposures(referenceExp, referenceExp)
```

compsExp *Components fitted onto featsExp*

Description

The generated components for the segDataExp dataset. Generated using the function fitModels(featsExp). Each data frame has a value and a ID column. The ID tells you which sample the observed value is from.

Usage

```
compsExp
```

Format

A list with 6 flexmix objects:

segsize Mixture of normal distributions

bp10MB Mixture of normal distributions

osCN Mixture of poisson distributions

changepoint Mixture of normal distributions

copynumber Mixture of normal distributions

bpchrarm Mixture of poisson distributions

createSigs *createSigs*

Description

This function is used to create the final signatures and generates the resulting NMF object, from which you can extract the feature contribution to each signature using NMF::basis(), and the signature contribution of each sample by using NMF::scoef()

Usage

```
createSigs(scm, nsig, cores = 1, runName = "", saveRes = FALSE, saveDir = NULL)
```

Arguments

scm	The sample_by_component matrix to run NMF on
nsig	Number of signatures for the NMF to create
cores	Number of cores to use in parallel process
runName	Name of the run used in file names, Default is ""
saveRes	Whether or not to save the results, Default is FALSE
saveDir	Where to save the results, must be provided if using saveDir

Value

Returns the resulting NMF object

Examples

```
createSigs(scmExp,5) #Generates 5 signatures from the SCM
```

---

defaultFeats *Default features to use for copy number signatures*

Description

These are the default features that are used in the package. Use this to get a list of the feature names and you can remove values from this and pass it in to the package using the featsToUse parameter seen in multiple functions.

Usage

```
defaultFeats
```

Format

An object of class `character` of length 6.

---

determineNumSigs *determineNumSigs*

Description

This function uses the extracted features and modelled components, and it performs NMF on these ranging from the minimum number of signatures to the max. It repeats this on randomized data and computes various measures to help inform the user on how many signatures to proceed with. This function may take a while to run since it repeats the NMF process many times. It is suggested to give it multiple cores to allow for parallel processing.

Usage

```
determineNumSigs(
  scm,
  rmin = 3,
  rmax = 12,
  cores = 1,
  nrun = 250,
  saveRes = FALSE,
  saveDir = NULL,
  runName = ""
)
```

Arguments

scm	Sample by component matrix used to find signatures
rmin	The lower bound of signature numbers to check. Default is 2.
rmax	The upper bound of signature numbers to check. Default is 12.
cores	The number of cores to use for parallel analysis. Default is 1.
nrun	Number of runs for NMF. Default is 250.
saveRes	Whether or not to save the plot. Default is FALSE.
saveDir	Directory to save plot in, must be provided if using saveDir
runName	Used to title plots and files when saving results

Value

Creates a series of plots to help user decide

Examples

```
determineNumSigs(generateSCM(featsExp,compsExp))
```

---

detSigNumPipeline *detSigNumPipeline*

Description

This function allows you to run the Copy number signature pipeline up until the determineSigNum call. This is useful if you want to repeteadly check the optimal number of signatures for different sample sets. May take a while, especially if not given multiple cores.

Usage

```
detSigNumPipeline(
  segData,
  cores = 1,
  components = NULL,
  saveRes = FALSE,
  runName = "Run",
  rmin = 3,
  rmax = 12,
  max_comps = NULL,
  min_comps = NULL,
  saveDir = NULL,
  smooth = FALSE,
  colMap = NULL,
  pR = FALSE,
  gbuild = "hg19",
  featsToUse = NULL,
  ploidyData = NULL
)
```

Arguments

segData	The data to be analyzed. If a path name, readSegs is used to make the list. Otherwise the list must be formatted correctly. Refer to ?readSegs for format information.
cores	The number of computer cores to be used for parallel processing
components	Can be used when fixing components. Default is NULL.
saveRes	Whether or not to save the resulting tables and plots. Default is FALSE
runName	Used to title plots and files when saving results
rmin	Minimum number of signatures to look for. Default is 3.
rmax	Maximum number of signatures to look for. Default is 12.
max_comps	vector of length 6 specifying the max number of components for each feature. Passed to fitModels. Default is 10 for all features
min_comps	vector of length 6 specifying the min number of components for each feature. Passed to fitModels. Default is 2 for all features
saveDir	Used to specify where to save the results, must be provided if using saveDir
smooth	Whether or not to smooth the input data. Default is FALSE.
colMap	Mapping of column names when reading from text file. Default column names are ID, chromosome, start, end, segVal.
pR	Peak Reduction
gbuild	The reference genome build. Default is hg19. Also supports hg18 and hg38.
featsToUse	Vector of feature names that you wish to use
ploidyData	The ploidy data to use as a feature

Value

Returns a list with all of the results from the pipeline

Examples

```
#Runs the entire pipeline on the example data giving it 6 cores and specifying
#5 signatures with a name of "TCGA Test"

detSigNumPipeline(segDataExp, cores = 6, saveRes = FALSE, runName = "TCGA Test")
```

---

diffCompSigSim *diffCompSigSim*

Description

This function is used to determine the similarity between two signatures that have different underlying components. Uses ks-statistic based measure to estimate similarity for normal distribution based components and uses a correlation measure when comparing poisson distribution based components.

Usage

```
diffCompSigSim(refComps, refWeights, valComps, valWeights)
```

Arguments

refComps	Reference component parameters
refWeights	Reference component weights
valComps	Component parameters to compare against
valWeights	Component weights to compare agaisnt

Value

Returns a correlation value

---

extractCNFeats *extractCNFeats*

Description

This function is used to extract the six copy number features that are eventually used in order to make the signatures. It does this using six sub functions to extract each feature. Before extracting the features, the segments are passed through a validation function to make sure the data is formatted correctly and there are no invalid segments. Can be done in parallel using the cores parameter.

Usage

```
extractCNFeats(
  segData,
  gbuild = "hg19",
  cores = 1,
  featsToExtract = CNSigs::defaultFeats
)
```

Arguments

segData	The copy number segment data
gbuild	The reference genome build. Default is hg19. Also supports hg18 and hg38.
cores	The number of cores to use for parallel processing. Default 1.
featsToExtract	The names of the features to extract.

Value

list of dataframes containing results of six copy number features

Examples

```
extractCNFeats(segDataExp)
```

`extractSegsize` *featureFuncs*

Description

This group of functions return vectors of the corresponding features for the samples passed in. Some of the functions use internal datasets to the CNSig package that specify the chromosome lengths and the centromere positions.

Usage

```
extractSegsize(segData)

extractBP10MB(segData, chrlen)

extractOscillations(segData, chrlen)

extractBPChrArm(segData, centromeres, chrlen)

extractChangepoints(segData, centromeres, chrlen)

extractCN(segData)
```

Arguments

`segData`	The samples to extract data from
`chrlen`	The lengths of the chromosomes from reference genome
`centromeres`	The positions of the centromeres in reference genome

extractSegsize

This function returns a vector of all the segment sizes for all for all of the samples.

extractBP10MB

This function returns a vector of the average number of breakpoints in a per 10MB for each chromosome.

extractOscillations

This function returns a vector of number of oscillation events found on each of the chromosomes.

getBPChrArm

This function returns a vector of number of total breakpoints per chromosome arm.

extractChangepoints

This function returns a vector of average size of changepoints per chromosome

extractCN

This function returns a vector of average copynumber per chromosome

---

`featsExp` *Features from segDataExp*

Description

The extracted features from the segDataExp dataset. Generated using the function extractCN-Feats(segDataExp). Each data frame has a value and a ID column. The ID tells you which sample the observed value is from.

Usage

```
featsExp
```

Format

A list with 6 data frames:

segsize Size of every segment

bp10MB Average # of breakpoints per 10MB per chromosome

osCN Number of oscillation events per chromosome

changepoint Average changepoint per chromosome

copynumber Average copy number per chromosome

bpchrarm Number of breakpoints per chromosome arm

`findExposures` *findExposures*

Description

This function is used to find the signature exposures of a set of samples using fixed signatures found earlier. It does this using the least squares optimization method with constraints to keep the output as non-negative using the lsei function from the package limSolve.

Usage

```
findExposures(scm, fixedSigs, runName = "", saveRes = FALSE, saveDir = NULL)
```

Arguments

scm	Sample by component matrix
fixedSigs	The fixed signatures
runName	Name of the run used in file names, Default is ""
saveRes	Whether or not to save the results, Default is FALSE
saveDir	Where to save the results, must be provided if using saveDir

Value

Returns the resulting matrix of exposures

Examples

```
findExposures(t(scmExp), sigsExp)
```

---

fitComponent *fitComponent*

Description

This function is used to fit a mixture model of either normal or poisson distributions to the inpput data. This function is mainly used by the fitModels function to create the components from the extracted features.

Usage

```
fitComponent(
  toFit,
  min_prior = 0.001,
  min_comp = 2,
  max_comp = 10,
  dist = "norm",
  pR = FALSE,
  seed = 77777,
  model_sel = "BIC",
  niter = 10000,
  nrep = 1
)
```

Arguments

toFit	Extracted features to fit models to.
min_prior	Minimum prior probability of a cluster. Default is 0.001.
min_comp	Minimum number of models to fit. Default is 2.
max_comp	Maximum number of models to fit. Default is 10.

dist	Type of distribution to fit. Either "norm" or "pois". Default "norm"
pR	Peak Reduction reduces peaks in modeling to make modeling easier. Default is FALSE.
seed	Seed to be used for modeling. Default is 77777
model_sel	Type of model_selection method to be used. Default "BIC". See flexmix package for more options.
niter	Max number of iterations for modeling. Default is 1000.
nrep	Number of repetitions for modeling attempts. Default is 1.

Value

Returns the flexmix object for the fit model.

Examples

```
fitComponent(featsExp$bp10MB[,2]) #Fits 2-10 normal distributions

#Tries to fit exactly 4 poisson distributions
fitComponent(featsExp$osCN[,2],dist="pois",min_comp = 4, max_comp = 4)
```

`fitModels` *fitModels*

Description

This function takes all of the extracted copy number features and attempts to fit a mixture of poisson and normal distributions to the data, and returns a mixture of components that can be used to build the signatures. The order of features is "segsize","bp10MB","osCN","changepoint","copynumber","bpchrarm". Therefore if you only want to change the maximum number of components for osCN to 5 then you would use max_comps = c(10,10,5,10,10,10).

Usage

```
fitModels(
  CN_features,
  max_comps = NULL,
  min_comps = NULL,
  cores = 1,
  pR = FALSE,
  min_prior = NULL,
  featsToModel = CNSigs::defaultFeats
)
```

Arguments

CN_features	List of features received from extractCopynumberFeatures
max_comps	vector of length 6 specifying the max number of components for each feature. default is 10 for all features
min_comps	vector of length 6 specifying the min number of components for each feature. default is 2 for all features
cores	Number of parallel cores to use. Default is 1.
pR	Peak Reduction reduces peaks in modeling to make modeling easier. Default is FALSE.
min_prior	Used to override the minimum prior probabilty of a cluster
featsToModel	The names of the features to extract.

Value

Returns a list of the different components that contain flexmix objects for each feature

Examples

```
fitModels(featsExp)

#Models an exact number of components, useful when comparing two different
#datasets
min_comps = c(7, 3, 3, 2, 2, 3)
max_comps = c(7, 3, 3, 2, 10, 3)
fitModels(featsExp, max_comps, min_comps)
```

---

generateSCM *generateSCM*

Description

This function takes in an extracted set of features and a defined set of components, and calculates the sum of the posterior probabilities for each feature. This sum represents how much each component contributes to a sample and corresponds to one column in the matrix.

Usage

```
generateSCM(feats, comps, runName = "", saveRes = FALSE, saveDir = NULL)
```

Arguments

feats	List of features received from extractCopynumberFeatures
comps	List of components modelled using fitModels
runName	Name of the run used in file names, Default is ""
saveRes	Whether or not to save the results, Default is FALSE
saveDir	Where to save the results, Default is getwd()

Value

Creates a sample by component matrix

Examples

```
generateSCM(featsExp,compsExp)
```

`matchSigs` *matchSigs*

Description

This function is used to check to find a mapping between two similar sets of signatures. It compares the signature values to see how similar the proposed signatures are and shows you the best matches. It uses the measure of cosine similarity to compare signatures. The two signature sets must have the same underlying components to be matched.

Usage

```
matchSigs(referenceSigs, toCompareSigs)
```

Arguments

`referenceSigs`	Signature matrix from your reference analysis
`toCompareSigs`	Signature matrix from run that you want to compare to reference

Value

Prints out the signature mapping and returns the avg similarity

Examples

```
matchSigs(referenceExp$sigs, referenceExp$sigs)
```

plotComp *plotComp*

Description

This function plots the specified mixed model so that it can be visualized. It utilizes the gamma function to allow approximations of the poisson distributions, allowing for a smooth plot.

Usage

```
plotComp(comps, compName, saveRes = FALSE, saveDir = NULL, runName = "")
```

Arguments

comps	List of components to be plotted. Output from fitModels.
compName	Name of the component to plot
saveRes	Whether or not to save results. Default is F.
saveDir	Where to save plots, must be provided if using saveDir
runName	Used to add a runName to the file output. Default is "".

Value

Plots the components to allow visualization

Examples

```
plotComp(compsExp, compName = "segsize")
```

plotComps *plotComps*

Description

This function plots all of the mixed models so that it can be visualized. It utilizes the gamma function to allow approximations of the poisson distributions, allowing for a smooth plot.

Usage

```
plotComps(comps, saveRes = FALSE, saveDir = NULL, runName = "")
```

Arguments

comps	List of components to be plotted. Output from fitModels.
saveRes	Whether or not to save results. Default is F.
saveDir	Where to save plots. Default is getwd()
runName	Used to add a runName to the file output. Default is "".

Value

Plots all the components to allow visualization

Examples

```
plotComps(compsExp)
```

`plotScm` *plotScm*

Description

This function is used to generate the sample by component matrix plot.

Usage

```
plotScm(
  scm,
  runName = "",
  saveRes = FALSE,
  saveDir = NULL,
  rowOrder = FALSE,
  colOrder = TRUE
)
```

Arguments

`scm`	Sample by component matrix
`runName`	Name of the run used in plot titles, Default is ""
`saveRes`	Whether or not to save the plots, Default is FALSE
`saveDir`	Where to save the plots, must be provided if using saveDir
`rowOrder`	Ordering specification for the rows of the heatmap. Three possible options: * TRUE: Uses hierarchical clustering to determine row order. * FALSE: (default) Leaves rows in the order they were given. * A numeric vector the same length as the number of rows specifying the indices of the input matrix
`colOrder`	Ordering specification for the columns of the heatmap. See above for options. Default value is T.

Value

pheatmap figure of component result by sample

Examples

```
plotScm(scmExp)

plotScm(scmExp, rowOrder = FALSE, colOrder = FALSE)

newOrder = sample(1:ncol(scmExp), ncol(scmExp))
plotScm(scmExp, colOrder = newOrder)
```

---

plotSegs *plotSegs*

Description

This function is used to create a plot of a samples segs. The input samples can either be a single data.frame of the segs of one patient or a list of data.frames for multiple samples.

Usage

```
plotSegs(
  samples,
  name = "",
  chrom = -1,
  gbuild = "hg19",
  sep = FALSE,
  alpha = 1
)
```

Arguments

samples	The samples to plot. If a list it plots both on the same plot
name	The name of the sample. Used for plot title
chrom	Which chromosome to plot. Default plots all of them.
gbuild	The reference genome build. Default is hg19. Also supports hg18 and hg38.
sep	Whether or not to place different members of the list on the same or different axis
alpha	Allows you to adjust the transparency of the lines. 0-1

Value

displays a plot of the segments

Examples

```
plotSegs(segDataExp[[1]]) #Plots all of the first sample's segments
plotSegs(segDataExp[[1]],1) #Only plots the first chromosome segments
plotSegs(segDataExp[1:2]) #Plots first two samples on same axis
plotSegs(segDataExp[1:2], sep = TRUE) #Plots first two samples seperately
```

plotSig *plotSig*

Description

This function is used to create a plot for the specified signature to look at the contribution of each of the components to the signatures

Usage

```
plotSig(sigs, sigNum)
```

Arguments

sigs	The dataset of component contribution to each signature
sigNum	The signature number to plot

Value

displays a plot of the signature

Examples

```
plotSig(referenceExp$sigs, 1) #Plots first signature
```

plotSigExposure *plotSigExposure*

Description

This function plots the signature exposure for all of the samples as a stacked bar plot. There are a number of different options for how to sort the resulting plot.

Usage

```
plotSigExposure(
  sigExposure,
  saveRes = FALSE,
  saveDir = NULL,
  runName = "",
  trackData = NULL,
  sort = FALSE,
  sortOrder = "m",
  method = NULL,
  colors = NULL
)
```

Arguments

sigExposure	Signature exposure matrix to be plotted
saveRes	Whether or not to save results. Default is FALSE.
saveDir	Where to save plots, must be provided if using saveDir
runName	Used to add a runName to the file output. Default is "".
trackData	Data used to plot tracks
sort	Whether or not to sort the plot
sortOrder	The order in which to sort the plot
method	The method by which to sort the main plot
colors	Colors used in plotting

Details

Adding data tracks to the plot: One of the major features of this function is that it allows the user to add in some additional data for the samples to be plotted as a track alongside the main signature exposure stacked bar plot. These additional data points can be passed in as a vector of corresponding values in the same order. If you want to plot multiple tracks you can pass in a list of vectors using the trackData parameter.

Specifying how to sort the plot: When you give the function a set of trackData, it allows you to begin to specify the sortOrder. This allows your to sort the main plot in a different order. "m" represents the main plot, and "t" followed by the number of the track (ie: "t1","t2" ...) represents the tracks. By chaining the values together you can specify a variety of ways to sort the final plot. As an example, the sortOrder of "mt1t2" specifies the the plot should be sorted by the signature exposures first followed by the first track and finally the second track. In another example, the sortOrder of "t2mt1" specifies the plot to be sorted by track number 2 first followed by the signature exposures and lastly by track number 1.

Sorting method: The two methods of sorting the signature exposure are either "hclust" or "group". The hclust uses the ward.D method to cluster the exposures and then cuts the tree to split the data. The group method splits the samples into groups based on which signatures they had the highest exposure to.

Value

Plots the signature exposure to allow visualization

Examples

```
plotSigExposure(sigExposExp)
```

plotSigExposureMat *plotSigExposureMat*

Description

This function is used to generate the signature exposure matrix heatmap plot.

Usage

```
plotSigExposureMat(
  sigExposure,
  runName = "",
  saveRes = FALSE,
  saveDir = NULL,
  rowOrder = FALSE,
  colOrder = TRUE
)
```

Arguments

sigExposure	Sample by signature matrix
runName	Name of the run used in plot titles, Default is ""
saveRes	Whether or not to save the plots, Default is FALSE
saveDir	Where to save the plots, must be provided if using saveDir
rowOrder	Ordering specification for the rows of the heatmap. Three possible options: * TRUE: Uses hierarchical clustering to determine row order. * FALSE: (default) Leaves rows in the order they were given. * A numeric vector the same length as the number of rows specifying the indices of the input matrix
colOrder	Ordering specification for the columns of the heatmap. See above for options. Default value is T.

Value

pheatmap figure of signature exposure by patient

Examples

```
plotSigExposureMat(sigExposExp)

plotSigExposureMat(sigExposExp, rowOrder = FALSE, colOrder = FALSE)

newOrder = sample(1:ncol(sigExposExp), ncol(sigExposExp))
plotSigExposureMat(sigExposExp, colOrder = newOrder)
```

plotSigMat *plotSigMat*

Description

This function is used to generate the signature by component matrix plot.

Usage

```
plotSigMat(sigs, runName = "", saveRes = FALSE, saveDir = NULL)
```

Arguments

sigs	Signature by component matrix
runName	Name of the run used in plot titles, Default is ""
saveRes	Whether or not to save the plots, Default is FALSE
saveDir	Where to save the plots, must be provided if using saveDir

Value

pheatmap figure of component weights by sample

Examples

```
plotSigMat(sigsExp)
```

plotSigs *plotSigs*

Description

This function plots all of the signatures so that they can be visualized. It does this by looping through the signatures and calling the plotSig function

Usage

```
plotSigs(sigs, saveRes = FALSE, saveDir = NULL, runName = "")
```

Arguments

sigs	The dataset of component contribution to each signature
saveRes	Whether or not to save results. Default is FALSE.
saveDir	Where to save plots, must be provided if using saveDir
runName	Used to add a runName to the file output. Default is "".

Value

Plots all the signatures to allow visualization

Examples

```
plotSigs(referenceExp$sigs)
```

`postProb` *postProb*

Description

This function calculates the probabilities that each of the new data point falls into the distributions defined by the parameters. Used when calculating the sample by component matrix.

Usage

```
postProb(params, newData)
```

Arguments

`params`	A vector of the distribution parameters
`newData`	The new data to calculate the probabilities for

Value

Returns the probability that the newData is in the distributions

---

`readSegs` *readSegs*

Description

This function is used to read in the segments. It can either take a file path to a csv to read in the data, or it can take in a long data frame and convert it to the format needed for the pipeline. The variable colMap is used in order to map your column names to what the pipeline expects. For instance, if your column that has the chromosome numbers in it is titled "chrom" instead of the expected "chromosome" then you would specify the colMap as c("ID","chrom","start","end","segVal"). If your data is seperated into major and minor allele copy numbers then for the segVal part of the colMap should be formatted as "nMajor+nMinor" to let the function know to add them together.

Usage

```
readSegs(path, colMap = NULL, readPloidy = FALSE)
```

Arguments

path	The path to the .txt file with the data in it, or a folder containing the .txt files
colMap	The mapping of column names. The default is c("ID","chromosome","start","end","segVal"). If your column names vary from this please pass a vector similar to the above with the changes.
readPloidy	Whether or not the input file has ploidy and should be read

Value

Returns a segments in a list formatted to be run through the pipeline

---

reducePeaks *reducePeaks*

Description

This function is used to reduce the peaks within a feature distribution so that the models can be fitted properly. The flexmix package can struggle to converge on a solution if there are large spikes in the distribution.

Usage

```
reducePeaks(toReduce)
```

Arguments

toReduce	Input feature distribution

Value

Returns the input feature with the peaks reduced

---

referenceExp *The result object from Pipeline using segDataExp.*

Description

The generated result object from the entire pipelin using the segDataExp. Function used to create: referenceExp = runPipeline(segDataExp,nsigs = 5)

Usage

```
referenceExp
```

Format

A list with 7 elements:

func The function call used in the pipeline run

Input_data The data the features were extracted from

CN_features The extracted features

CN_components The fitted component models

scm The sample by component matrix

nmf_Results The results of the NMF run

sigs The signature by component matrix.

---

remapResults *remapResults*

Description

This function is used to remap the results from a runPipeline run with a different order of the signatures or different names. You can either give the function a new mapping of the signatures which is just a new order in which you want the signatures in. For instance, if you want to just swap the first and second signatures and you have a total of 4 signatures, you would pass in c(2,1,3,4) for the sigMap parameters. The other use case is if you want to rename the signatures. To do this you just have to pass a vector of names that is of the same length as the number of signatures.

Usage

```
remapResults(path, sigMap = NULL, sigNames = NULL, saveRes = FALSE)
```

Arguments

path	The path to the results folder to remap
sigMap	The new order for the signatures
sigNames	New signature names
saveRes	Whether or not to save results. Default is FALSE.

Details

Overall, this function will create a duplicate results folder in the same directory and regenerate all of the plots and result files into the new order or with the new names. This means that you don't have to regenerate all of the plots manually.

Value

no return

runPipeline *runPipeline*

Description

This function allows you to run the entire Copy number signature pipeline in one go. May take a while, especially if not given multiple cores. For more information on what actually happens in the pipeline, refer to the CNSigs vignette.

Usage

```
runPipeline(
  segData,
  cores = 1,
  nsigs = 0,
  saveRes = FALSE,
  runName = "Run",
  rmin = 3,
  rmax = 12,
  components = NULL,
  max_comps = NULL,
  min_comps = NULL,
  fixedSigs = NULL,
  saveDir = NULL,
  smooth = FALSE,
  colMap = NULL,
  pR = FALSE,
  gbuild = "hg19",
  featsToUse = NULL,
  ploidyData = NULL,
  plot = TRUE
)
```

Arguments

segData	The data to be analyzed. If a path name, readSegs is used to make the list. Otherwise the list must be formatted correctly. Refer to ?readSegs for format information.
cores	The number of computer cores to be used for parallel processing
nsigs	The number of signatures to look for. Value of 0 runs the determineSigNum function to look for optimal number. Default is 0.
saveRes	Whether or not to save the resulting tables and plots. Default is FALSE
runName	Used to title plots and files when saving results
rmin	Minimum number of signatures to look for. Default is 3.
rmax	Maximum number of signatures to look for. Default is 12.

components	Can be used when fixing components. Default is NULL.
max_comps	vector of length 6 specifying the max number of components for each feature. Passed to fitModels. Default is 10 for all features
min_comps	vector of length 6 specifying the min number of components for each feature. Passed to fitModels. Default is 2 for all features
fixedSigs	Signature x Component matrix. Used when fixing signatures. Default is NULL
saveDir	Used to specify where to save the results, must be provided if using saveDir
smooth	Whether or not to smooth the input data. Default is F.
colMap	Mapping of column names when reading from text file. Default column names are ID, chromosome, start, end, segVal.
pR	Peak Reduction
gbuild	The reference genome build. Default is hg19. Also supports hg18 and hg38.
featsToUse	Vector of feature names that you wish to use
ploidyData	The ploidy data to use as a feature
plot	Whether or not to generate the plots. Default is T.

Value

Returns a list with all of the results from the pipeline

Examples

```
#Runs the entire pipeline on the example data giving it 6 cores and specifying
#5 signatures with a name of "TCGA Test"

runPipeline(segDataExp, cores = 6, nsigs = 5, saveRes = FALSE, "TCGA Test")
```

---

scmExp *Sample by component matrix for segDataExp*

Description

The generated scm for the segDataExp dataset. Generated using the function generateSCM(featsExp,compsExp). It is a matrix showing how much each extracted component contributes to each sample. Is what is put into NMF and used to create the signatures.

Usage

```
scmExp
```

Format

An object of class `matrix` (inherits from `array`) with 20 rows and 20 columns.

segDataExp *Segmentation Data from TCGA BRCA samples*

Description

A small example subset of BRCA samples from TCGA in list format. Each item contains the segmentation data for that sample.

Usage

```
segDataExp
```

Format

A list with 20 elements and 5 variables:

ID ID number for the sample

chromosome chromosome the segment is found on

start starting position of the segment

end end position of the segment

segVal copynumber value for the segment

Source

https://portal.gdc.cancer.gov/

segStats *segStats*

Description

This function allows you to get an overview of some of the features of your samples. It outputs a summary of stats for the segments, including size, number per sample, along with various other measures.

Usage

```
segStats(segTabs)
```

Arguments

segTabs	The list of samples copy number segments

Value

Outputs a summary of the statistics

Examples

```
segStats(segDataExp)
```

`sigExposExp` *The generated sigExposure from the segDataExp run*

Description

The generated signature exposure matrix for the segDataExp dataset. Extracted from the referenceExp data object using referenceExp$sigExposure. This matrix shows how much each signature contributes to the patient samples.

Usage

```
sigExposExp
```

Format

An object of class `data.frame` with 5 rows and 20 columns.

`sigsExp` *The generated Signatures from the segDataExp run*

Description

The generated signature by component matrix for the segDataExp dataset. Extracted from the referenceExp data object using referenceExp$sigs. This matrix shows how much each component contributes to the signatures.

Usage

```
sigsExp
```

Format

An object of class `matrix` (inherits from `array`) with 20 rows and 5 columns.

sigSim *sigSim*

Description

This function is used to compare two sets of signatures by finding the similarity matrix across both signature sets. If the signatures have the same underlying components similarity is calculated using the cosine similarity. If the signatures have different underlying components the similarity is estimated using a ks-statistic based measure. See package vignette for more information.

Usage

```
sigSim(reference, toCompare, plot = TRUE, text = TRUE)
```

Arguments

reference	Results from your reference analysis
toCompare	Results from run that you want to compare to reference
plot	If T, displays the heatmap plot
text	If T, displays the similarity value on the plot

Value

Plots signature similarity and returns the avg similarity

Examples

```
sigSim(referenceExp, referenceExp)
```

smoothSegs *smoothSegs*

Description

This function is used to attempt to smooth the input copy number segments in order to reduce the biasing affect of the technology and copy number caller used to make the segments. It does this by trying to join together close segs and removing small abbarent segments.

Usage

```
smoothSegs(segData, cores = 1)
```

Arguments

segData	The segData to be smoothed
cores	Number of cores to be used for parallel smoothing. Default is 1.

Value

Returns the smoothed segments

Examples

```
smoothSegs(segDataExp)
```

`sumOfPosteriors` *sumOfPosteriors*

Description

This function is used to calculate the sum of posteriors for a given feature. It returns a vector of posterior probabilities which describe how much each component contributes to the distribution of features passed in.

Usage

```
sumOfPosteriors(feat, comps, name)
```

Arguments

`feat`	Feature to calculate the sum from.
`comps`	Component parameters for that feature
`name`	Name of feature to sum across

Value

Returns the sum of the posteriors for the specified feature.

`validateSegData` *validateSegData*

Description

This function is used to validate and clean up all of the input data. It converts all the columns to numeric that need to be, and filters out any invalid segments, like ones with 0 or negative length. It also converts the chromosome tags to the proper format for feature extraction.

Usage

```
validateSegData(segData, cores = 1)
```

Arguments

segData	The copy number segment data
cores	The number of cores to use for parallel processing. Default 1.

Value

list of dataframes containing converted seg data

Examples

```
validateSegData(segDataExp)
```

Index

∗ **NMF**
createSigs, 6
∗ **bp10MB**
extractCNFeats, 10
∗ **bpchrarm**
extractCNFeats, 10
∗ **by**
generateSCM, 15
∗ **changepoint**
extractCNFeats, 10
∗ **components**
fitModels, 14
∗ **component**
generateSCM, 15
plotComp, 17
plotComps, 17
∗ **datasets**
cancerComps, 3
cancerSigs, 4
collapsedSigs, 5
compsExp, 6
defaultFeats, 7
featsExp, 12
referenceExp, 25
scmExp, 28
segDataExp, 29
sigExposExp, 30
sigsExp, 30
∗ **detSigNum**
detSigNumPipeline, 8
∗ **exposure**
plotSigExposureMat, 22
∗ **features**
extractCNFeats, 10
∗ **fixed**
findExposures, 12
∗ **lsei**
findExposures, 12
∗ **matrix**
generateSCM, 15
plotScm, 18
plotSigExposureMat, 22
plotSigMat, 23
∗ **mixed**
fitModels, 14
∗ **models**
fitModels, 14
∗ **number**
determineNumSigs, 7
∗ **of**
determineNumSigs, 7
∗ **osCN**
extractCNFeats, 10
∗ **plot**
plotComp, 17
plotComps, 17
plotScm, 18
plotSegs, 19
plotSig, 20
plotSigExposure, 20
plotSigExposureMat, 22
plotSigMat, 23
plotSigs, 23
∗ **run**
runPipeline, 27
∗ **sample**
generateSCM, 15
∗ **scm**
generateSCM, 15
plotScm, 18
∗ **segData**
validateSegData, 32
∗ **segmentation**
segStats, 29
∗ **segments**
plotSegs, 19
∗ **segsize**
extractCNFeats, 10

∗ **sigExposure**
plotSigExposure, 20
plotSigExposureMat, 22
∗ **signatures**
createSigs, 6
determineNumSigs, 7
findExposures, 12
plotSigs, 23
∗ **signature**
plotSig, 20
plotSigExposureMat, 22
plotSigMat, 23
∗ **signum**
determineNumSigs, 7
∗ **statistics**
segStats, 29
∗ **stats**
segStats, 29
∗ **summary**
segStats, 29
∗ **validate**
validateSegData, 32

addPloidyData, 3

cancerComps, 3
cancerSigs, 4
checkCompOverlap, 4
collapsedSigs, 5
compareExposures, 5
compsExp, 6
createSigs, 6

defaultFeats, 7
determineNumSigs, 7
detSigNumPipeline, 8
diffCompSigSim, 9

extractBP10MB *(*extractSegsize*)*, 11
extractBPChrArm *(*extractSegsize*)*, 11
extractChangepoints *(*extractSegsize*)*, 11
extractCN *(*extractSegsize*)*, 11
extractCNFeats, 10
extractOscillations *(*extractSegsize*)*, 11
extractSegsize, 11

featsExp, 12
findExposures, 12
fitComponent, 13
fitModels, 14

generateSCM, 15

matchSigs, 16

plotComp, 17
plotComps, 17
plotScm, 18
plotSegs, 19
plotSig, 20
plotSigExposure, 20
plotSigExposureMat, 22
plotSigMat, 23
plotSigs, 23
postProb, 24

readSegs, 24
reducePeaks, 25
referenceExp, 25
remapResults, 26
runPipeline, 27

scmExp, 28
segDataExp, 29
segStats, 29
sigExposExp, 30
sigsExp, 30
sigSim, 31
smoothSegs, 31
sumOfPosteriors, 32

validateSegData, 32