Package 'decorate'

March 2, 2020

Type Package

Title Differential Epigenetic Coregulation Test

Version 1.0.27

Date 2019-11-25

Author Gabriel E. Hoffman <gabriel.hoffman@mssm.edu>

Maintainer Gabriel E. Hoffman <gabriel.hoffman@mssm.edu>

Description Identify regions of the genome that show different patterns of coregulation of epigenetic signals between two datasets.

VignetteBuilder knitr

License GPL (>= 2)

Suggests BiocStyle, knitr, EnsDb.Hsapiens.v86

biocViews ChIPSeq, ATACSeq, DNAMethylation, DifferentialPeakCalling, DifferentialMethylation, DifferentialExpression, GeneSetEnrichment, Regression, Clustering, Software

Depends ggplot2, grid, Biobase, methods

Imports ClustGeo, sp, spdep, vegan, rlist, adjclust, limma, sLED, foreach, progress, data.table, BiocParallel, Rcpp, MASS, RcppArmadillo, labeling, cowplot, GenomicFeatures, grDevices, iterators, GGally, ggdendro, GenomeInfoDb, IRanges, psych, Matrix, rtracklayer, reshape2, heplots, GenomicRanges, utils, stats

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.0.0

NeedsCompilation yes

R topics documented:

.evalDiffCorr 3
boxM 4
boxM_fast 4
boxM_permute 5
collapseClusters 6
combineResults 7
corrMatrix.test 8
corSubsetPairs 9

countClusters . . . 10
createClusters . . . 10
createCorrelationMatrix . . . 11
decorateData . . . 12
delaneau.score . . . 13
delaneau.test . . . 14
epiclust-class . . . 15
epiclustDiscreteList-class . . . 15
epiclustDiscreteListContain-class . . . 15
epiclustList-class . . . 15
evalDiffCorr . . . 16
evaluateCorrDecay . . . 18
extractCorrelationScores . . . 19
filterClusters . . . 21
getClusterNames . . . 21
getClusterRanges . . . 22
getFeaturesInCluster . . . 23
getFeaturesInClusterList . . . 24
getPeakDistances . . . 25
getSubset . . . 25
get_exon_coords . . . 26
ggplot_by_sampling . . . 27
jaccard . . . 27
makeImageRect . . . 28
plotClusterSegments . . . 28
plotCompareCorr . . . 29
plotCorrDecay . . . 30
plotCorrTriangle . . . 31
plotDecorate . . . 32
plotDensityPoints . . . 33
plotEnsGenes . . . 34
plotGenes . . . 35
plotPairwiseScatter . . . 36
plotScatterPairs . . . 37
retainClusters . . . 38
runFastStat . . . 39
runOrderedClustering . . . 39
runOrderedClusteringGenome . . . 40
runPermutedData . . . 42
scoreClusters . . . 43
sle.score . . . 44
sle.test . . . 45
sLEDresults-class . . . 46
summary,sLEDresults-method . . . 46
whichCluster . . . 47
[,epiclustDiscreteListContain,ANY,ANY,ANY-method . . . 48

Index **49**

`.evalDiffCorr` *Internal .evalDiffCorr*

Description

Internal .evalDiffCorr

Usage

```
.evalDiffCorr(
  epiSignal,
  testVariable,
  gRanges,
  clustList,
  npermute = c(100, 10000, 1e+05),
  adj.beta = 0,
  rho = 0,
  sumabs.seq = 1,
  BPPARAM = bpparam(),
 method = c("sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich",
   "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U",
     "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)
```

Arguments

`epiSignal`	matrix or EList of epigentic signal. Rows are features and columns are samples
`testVariable`	factor indicating two subsets of the samples to compare
`gRanges`	GenomciRanges corresponding to the rows of epiSignal
`clustList`	list of cluster assignments
`npermute`	array of two entries with min and max number of permutations
`adj.beta`	parameter for sLED
`rho`	a large positive constant such that A(X)-A(Y)+diag(rep(rho,p)) is positive definite. Where p is the number of features
`sumabs.seq`	sparsity parameter
`BPPARAM`	parameters for parallel evaluation
`method`	"sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich", "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"
`method.corr`	Specify type of correlation: "pearson", "kendall", "spearman"

Value

list of result by chromosome and clustList

boxM *Box's M-test*

Description

Box's M-test

Usage

```
boxM(
  Y,
  group,
  tol = 1e-10,
  fxn = cor,
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

Y	response matrix
group	factor defining groups
tol	tolerance for eigen values
fxn	define function. Here default is cor to compare correlation structure. Use cov to compare covariance structure like in heplots::boxM
method	specify which correlation method: "pearson", "kendall" or "spearman"

Examples

```
data(iris)

boxM( iris[,1:4], iris[,5])
```

boxM_fast *Box's M-test*

Description

boxM performs the Box's (1949) M-test for homogeneity of covariance matrices obtained from multivariate normal data according to one or more classification factors. The test compares the product of the log determinants of the separate covariance matrices to the log determinant of the pooled covariance matrix, analogous to a likelihood ratio test. The test statistic uses a chi-square approximation. Uses permutations to estimate the degrees of freedom under the null

Usage

```
boxM_fast(Y, group, method = c("pearson", "spearman"))
```

Arguments

Y	response variable matrix
group	a factor defining groups, number of entries must equal nrow(Y)
method	Specify type of correlation: "pearson", "spearman"

See Also

heplots::boxM

Examples

```
data(iris)

boxM_fast( as.matrix(iris[, 1:4]), iris[, "Species"])
```

---

boxM_permute *Box's M-test*

Description

boxM performs the Box's (1949) M-test for homogeneity of covariance matrices obtained from multivariate normal data according to one or more classification factors. The test compares the product of the log determinants of the separate covariance matrices to the log determinant of the pooled covariance matrix, analogous to a likelihood ratio test. The test statistic uses a chi-square approximation. Uses permutations to estimate the degrees of freedom under the null

Usage

```
boxM_permute(
  Y,
  group,
  nperm = 200,
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

Y	response variable matrix
group	a factor defining groups, or a continuous variable, number of entries must equal nrow(Y)
nperm	number of permutations of group variable used to estimate degrees of freedom under the null
method	Specify type of correlation: "pearson", "kendall", "spearman"

Value

list of p.value, test statistic, and df.approx estimated by permutation

See Also

heplots::boxM

Examples

```
data(iris)

boxM_permute(iris[, 1:4], iris[, "Species"])
```

`collapseClusters` *Collapse clusters based on jaccard index*

Description

Collapse clusters if jaccard index between clusters exceeds a cutoff

Usage

```
collapseClusters(treeListClusters, featurePositions, jaccardCutoff = 0.9)
```

Arguments

`treeListClusters`	from createClusters()
`featurePositions`	GRanges object storing location of each feature
`jaccardCutoff`	cutoff value for jaccard index

Value

subset of clusters in treeListClusters that passes cutoff

Examples

```
library(GenomicRanges)

# load data
data('decorateData')

# Evaluate hierarchical clustering
# adjacentCount is the number of adjacent peaks considered in correlation
treeList = runOrderedClusteringGenome( simData, simLocation)

# Choose cutoffs and return cluster
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20, 30, 40, 50)

# Evaluate strength of correlation for each cluster
clstScore = scoreClusters(treeList, treeListClusters )

# Filter to retain only strong clusters
clustInclude = retainClusters( clstScore, "LEF", 0.30 )
```

```
# get retained clusters
treeListClusters_filter = filterClusters( treeListClusters, clustInclude)

# collapse similar clusters
treeListClusters_collapse = collapseClusters( treeListClusters_filter, simLocation)
```

`combineResults` *Combine results into a single data.frame*

Description

Combine results into a single data.frame for easy post processing

Usage

```
combineResults(
  sledRes,
  clstScore,
  treeListClusters,
  peakLocations,
  verbose = TRUE
)
```

Arguments

`sledRes`	sLEDresults from evalDiffCorr()
`clstScore`	cluster summary statistics from from scoreClusters()
`treeListClusters`	epiclustDiscreteListContain from createClusters()
`peakLocations`	GenomeRanges object
`verbose`	show messages

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clsutering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20) )

# Evaluate strength of correlation for each cluster
clstScore = scoreClusters(treeList, treeListClusters )
```

```
# Filter to retain only strong clusters
# If lead eigen value fraction (LEF) > 30% then keep clusters
# LEF is the fraction of variance explained by the first eigen-value
clustInclude = retainClusters( clstScore, "LEF", 0.30 )

# get retained clusters
treeListClusters_filter = filterClusters( treeListClusters, clustInclude)

# collapse redundant clusters
treeListClusters_collapse = collapseClusters( treeListClusters_filter, simLocation, jaccardCutoff=0.9)

# Evaluate Differential Correlation between two subsets of data
sledRes = evalDiffCorr( simData, metadata$Disease, simLocation, treeListClusters_collapse, npermute=c(20, 200

# Combine results for each cluster
df_results = combineResults( sledRes, clstScore, treeListClusters, simLocation)
```

`corrMatrix.test` *Test difference between two correlation matricies*

Description

Test difference between two correlation matricies using one of 5 tests

Usage

```
corrMatrix.test(
  Y,
  group,
 method = c("Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich", "Factor",
  "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj",
    "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)
```

Arguments

`Y`	data matrix
`group`	a factor defining groups
`method`	Specify test: "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich", "Factor", "Mann.Whitney" "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"
`method.corr`	Specify type of correlation: "pearson", "kendall", "spearman"

corSubsetPairs *Compute correlations between pairs of features*

Description

Compute correlations between pairs of features given in idxi and idxj

Usage

```
corSubsetPairs(
  Y,
  idxi,
  idxj,
  method = c("pearson", "spearman"),
  silent = FALSE,
  setNANtoZero = FALSE
)
```

Arguments

Y	matrix where rows are features
idxi	indecies
idxj	indecies
method	specify which correlation method: "pearson" or "spearman"
silent	suppress messages
setNANtoZero	replace NAN correlation values with a zero

Value

Compute local correlations between for all k: cor(Y[,idxi[k]], Y[,idxj[k]])

Examples

```
# Simulate simple dataset
N = 600
Y = matrix(rnorm(N*100), 100, N)

# select pairs to compute correlations between
i1 = sample.int(N, 200, replace=TRUE)
i2 = sample.int(N, 200, replace=TRUE)

# evaluate all piars
C = corSubsetPairs(t(Y), i1,i2)

# show value
C[i1[10], i2[10]]

# show values from evaluating this pair directly
cor(Y[,i1[10]], Y[,i2[10]])
```

countClusters *Count clusters on each chromosome*

Description

Count clusters on each chromosome

Usage

```
countClusters(treeListClusters)

## S4 method for signature 'epiclustDiscreteList'
countClusters(treeListClusters)

## S4 method for signature 'epiclustDiscreteListContain'
countClusters(treeListClusters)
```

Arguments

treeListClusters	from createClusters()

Value

count number of clusters on each chromsome

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Count clusters on each chromsome
countClusters( treeListClusters )
```

createClusters *Create cluster from list of hclust objects*

Description

Create cluster from list of hclust objects

Usage

```
createClusters(
  treeList,
  method = c("capushe", "bstick", "meanClusterSize"),
  meanClusterSize = 50,
  pct = 0.15
)
```

Arguments

treeList	list of hclust objects
method	'capushe': slope heuristic. 'bstick': broken stick. 'meanClusterSize': create clusters based on target mean value.
meanClusterSize	select target mean cluster size. Can be an array of values
pct	minimum percentage of points for the plateau selection in capushe selection. Can be an array of values

Value

Convert hierarchical clustering into discrete clusters based on selection criteria method

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Plot correlations and clusters in region defined by query
query = range(simLocation)

plotDecorate( ensdb, treeList, treeListClusters, simLocation, query)
```

---

createCorrelationMatrix

Create correlation matrix

Description

Create correlation matrix based on correlation between pairs of peaks

Usage

```
createCorrelationMatrix(
  query,
  regionQuant,
  adjacentCount = 500,
  windowSize = 1e+06,
  method = "adjacent",
  method.corr = c("pearson", "spearman"),
  quiet = FALSE,
  setNANtoZero = FALSE
)
```

Arguments

query	GRanges object of intervals to query
regionQuant	normalized quantifications of regions in query. Rows are features, like in limma
adjacentCount	number of adjacent entries to compute correlation against
windowSize	width of window in bp around each interval beyond which weight is zero
method	'adjacent': compute corr on fixed count sliding window define by adjacentCount. "distance": compute corr for entries within windowSize bp
method.corr	specify which correlation method: "pearson" or "spearman"
quiet	suppress messages
setNANtoZero	replace NAN correlation values with a zero

Value

for peak i and j with distance d_i,j, M[i,j] = cor(vobj$E[i,], vobj$E[j,])

return sparse symmatric matrix

Examples

```
data('decorateData')

C = createCorrelationMatrix(simLocation, simData)
```

---

decorateData *Simulated data to show correlation clustering*

Description

Simulated data to show correlation clustering

Simulated data to show correlation clustering. GRanges object indicating genomic position of data in rows of simData

Simulated disease status for differential analysis

Usage

```
data(decorateData)

data(decorateData)

data(decorateData)
```

Format

An object of class `matrix` with 448 rows and 1000 columns.

`delaneau.score` *Score impact of each sample on correlation sturucture*

Description

Score impact of each sample on correlation sturucture. Compute correlation using all samples (i.e. C), then compute correlation omitting sample i (i.e. Ci). The score the sample i is based on the difference between C and Ci.

Usage

```
delaneau.score(Y, method = c("pearson", "kendall", "spearman"))
```

Arguments

`Y`	data matrix with samples on rows and variables on columns
`method`	specify which correlation method: "pearson", "kendall" or "spearman"

Value

score for each sample measure impact on correlation structure

See Also

delaneau.test

Examples

```
# load iris data
data(iris)

# Evalaute score on each sample
delaneau.score( iris[,1:4] )
```

`delaneau.test` *Test association between correlation sturucture and variable*

Description

Score impact of each sample on correlation sturucture and then peform test of association with variable using Kruskal-Wallis test

Usage

```
delaneau.test(Y, variable, method = c("pearson", "kendall", "spearman"))
```

Arguments

`Y`	data matrix with samples on rows and variables on columns
`variable`	variable with number of entries must equal nrow(Y). Can be discrete or continuous.
`method`	specify which correlation method: "pearson", "kendall" or "spearman"

Details

The statistical test used depends on the variable specified. if variable is factor with multiple levels, use Kruskal-Wallis test if variable is factor with 2 levels, use Wilcoxon test if variable is continuous, use Wilcoxon test

Value

list of p-value, estimate and method used

See Also

delaneau.score sle.test

Examples

```
# load iris data
data(iris)

# variable is factor with multiple levels
# use kruskal.test
delaneau.test( iris[,1:4], iris[,5] )

# variable is factor with 2 levels
# use wilcox.test
delaneau.test( iris[1:100,1:4], iris[1:100,5] )

# variable is continuous
# use cor.test with spearman
delaneau.test( iris[,1:4], iris[,1] )
```

epiclust-class *Class epiclust*

Description

Class epiclust

epiclustDiscreteList-class *Class epiclustDiscreteList*

Description

Class epiclustDiscreteList

epiclustDiscreteListContain-class *Class epiclustDiscreteListContain*

Description

Class epiclustDiscreteListContain: is a list containing epiclustDiscreteList objects

epiclustList-class *Class epiclustList*

Description

Class epiclustList

`evalDiffCorr` *Evaluate Differential Correlation*

Description

Evaluate Differential Correlation between two subsets of data

Usage

```
evalDiffCorr(
  epiSignal,
  testVariable,
  gRanges,
  clustList,
  npermute = c(100, 10000, 1e+05),
  adj.beta = 0,
  rho = 0,
  sumabs.seq = 1,
  BPPARAM = bpparam(),
 method = c("sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich",
   "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U",
     "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)

## S4 method for signature 'EList,ANY,GRanges,list'
evalDiffCorr(
  epiSignal,
  testVariable,
  gRanges,
  clustList,
  npermute = c(100, 10000, 1e+05),
  adj.beta = 0,
  rho = 0,
  sumabs.seq = 1,
  BPPARAM = bpparam(),
 method = c("sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich",
   "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U",
     "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)

## S4 method for signature 'matrix,ANY,GRanges,list'
evalDiffCorr(
  epiSignal,
  testVariable,
  gRanges,
  clustList,
  npermute = c(100, 10000, 1e+05),
  adj.beta = 0,
  rho = 0,
```

```
  sumabs.seq = 1,
  BPPARAM = bpparam(),
 method = c("sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich",
  "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U",
    "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)

## S4 method for signature 'data.frame,ANY,GRanges,list'
evalDiffCorr(
  epiSignal,
  testVariable,
  gRanges,
  clustList,
  npermute = c(100, 10000, 1e+05),
  adj.beta = 0,
  rho = 0,
  sumabs.seq = 1,
  BPPARAM = bpparam(),
 method = c("sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich",
  "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U",
    "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)
```

Arguments

`epiSignal`	matrix or EList of epigentic signal. Rows are features and columns are samples
`testVariable`	factor indicating two subsets of the samples to compare
`gRanges`	GenomciRanges corresponding to the rows of epiSignal
`clustList`	list of cluster assignments
`npermute`	array of two entries with min and max number of permutations
`adj.beta`	parameter for sLED
`rho`	a large positive constant such that A(X)-A(Y)+diag(rep(rho,p)) is positive definite. Where p is the number of features
`sumabs.seq`	sparsity parameter
`BPPARAM`	parameters for parallel evaluation
`method`	"sLED", "Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich", "Factor", "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj", "Schott.Frob", "Delaneau", "deltaSLE"
`method.corr`	Specify type of correlation: "pearson", "kendall", "spearman"

Details

Correlation sturucture between two subsets of the data is evaluated with sparse-Leading-Eigenvalue-Driven (sLED) test:

Zhu, Lingxue, Jing Lei, Bernie Devlin, and Kathryn Roeder. 2017. Testing high-dimensional covariance matrices, with application to detecting schizophrenia risk genes. Annals of Applied Statistics. 11:3 1810-1831. doi:10.1214/17-AOAS1062

Value

list of result by chromosome and clustList

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clsutering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20) )

# Plot correlations and clusters in region defined by query
query = range(simLocation)

# Plot clusters
plotDecorate( ensdb, treeList, treeListClusters, simLocation, query)

# Evaluate Differential Correlation between two subsets of data
sledRes = evalDiffCorr( simData, metadata$Disease, simLocation, treeListClusters, npermute=c(20, 200, 2000))

# get summary of results
df = summary( sledRes )

# print results
head(df)

# extract peak ID's from most significant cluster
peakIDs = getFeaturesInCluster( treeListClusters, df$chrom[1], df$cluster[1], "20")

# plot comparison of correlation matrices for peaks in peakIDs
#  where data is subset by metadata$Disease
main = paste0(df$chrom[1], ': cluster ', df$cluster[1])
plotCompareCorr( simData, peakIDs, metadata$Disease) + ggtitle(main)
```

`evaluateCorrDecay` *Evaluate the decay of correlation versus distance between features*

Description

For pairs of features evaluate the physical distance and the correlation

Usage

```
evaluateCorrDecay(treeList, gr, chromArray = seqlevels(gr), verbose = TRUE)
```

Arguments

treeList	list of hclust objects
gr	GenomicRanges object corresponding to features clustered in treeList
chromArray	Use this only this set of chromosmes. Can substantially reduce memory usage
verbose	show progress

Value

a data.frame of distance and correlation value for all pairs of features already evalauted in treeList. Note that runOrderedClusteringGenome() that returns treeList only evalutes correlation between a specified number of adjacent peaks

Examples

```
library(GenomicRanges)
library(ggplot2)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Evaluate how correlation between features decays with distance
dfDist = evaluateCorrDecay( treeList, simLocation )

# make plot
plotCorrDecay( dfDist )
```

---

extractCorrelationScores

Extract sample-level correlation scores

Description

Extract sample-level correlation scores for each cluster

Usage

```
extractCorrelationScores(
  epiSignal,
  gRanges,
  clustList,
  method = c("deltaSLE", "Delaneau"),
  method.corr = c("pearson", "kendall", "spearman"),
  BPPARAM = bpparam(),
  rho = 0.1,
  sumabs = 1
)
```

Arguments

epiSignal	matrix or EList of epigentic signal. Rows are features and columns are samples
gRanges	GenomciRanges corresponding to the rows of epiSignal
clustList	list of cluster assignments
method	"deltaSLE", "Delaneau"
method.corr	Specify type of correlation: "pearson", "kendall", "spearman"
BPPARAM	parameters for parallel evaluation
rho	used only for sle.score(). A positive constant such that cor(Y) + diag(rep(rho,p)) is positive definite. See sLED::sLED()
sumabs	used only for sle.score(). regularization paramter. Value of 1 gives no regularization, sumabs*sqrt(p) is the upperbound of the L_1 norm of v, controlling the sparsity of solution. Must be between 1/sqrt(p) and 1. See sLED::sLED()

Value

matrix of scores of each sample for each cluster

See Also

sle.score delaneau.score

Examples

```
library(GenomicRanges)

# load data
data('decorateData')

# Evaluate hierarchical clustering
# adjacentCount is the number of adjacent peaks considered in correlation
treeList = runOrderedClusteringGenome( simData, simLocation)

# Choose cutoffs and return cluster
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20, 30, 40, 50)

# Evaluate strength of correlation for each cluster
clstScore = scoreClusters(treeList, treeListClusters )

# Filter to retain only strong clusters
clustInclude = retainClusters( clstScore, "LEF", 0.30 )

# get retained clusters
treeListClusters_filter = filterClusters( treeListClusters, clustInclude)

# collapse similar clusters
treeListClusters_collapse = collapseClusters( treeListClusters_filter, simLocation)

# get correlation scores for each sample for each cluster
corScores = extractCorrelationScores( simData, simLocation, treeListClusters_collapse )
```

filterClusters *Extract subset of clusters*

Description

Extract subset of clusters based on entries in chroms and clusters

Usage

```
filterClusters(treeListClusters, clustInclude)
```

Arguments

treeListClusters	from createClusters()
clustInclude	data.frame from retainClusters() indcating which clusters to include

Value

epiclustDiscreteList of specified clusters

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Evaluate score for each cluster
clstScore = scoreClusters(treeList, treeListClusters )

# Retain clusters that pass this criteria
clustInclude = retainClusters( clstScore, "LEF", 0.30 )

# get retained clusters
treeListClusters_filter = filterClusters( treeListClusters, clustInclude)
```

getClusterNames *Get name of each cluster*

Description

Get name of each cluster as parameter:chrom:cluster

Usage

```
getClusterNames(clustList)
```

Arguments

clustList	list of cluster assignments

Value

array of cluster names

Examples

```
library(GenomicRanges)

# load data
data('decorateData')

# Evaluate hierarchical clustering
# adjacentCount is the number of adjacent peaks considered in correlation
treeList = runOrderedClusteringGenome( simData, simLocation)

# Choose cutoffs and return cluster
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20, 30, 40, 50)

# Name of each cluster is parameter:chrom:cluster
getClusterNames( treeListClusters)
```

---

getClusterRanges *Get genome coordinates for each cluster*

Description

Get genome coordinates for each cluster as a GRanges object

Usage

```
getClusterRanges(gRanges, clustList, verbose = TRUE)
```

Arguments

gRanges	GenomciRanges corresponding to the rows of epiSignal
clustList	list of cluster assignments
verbose	write messages to screen

Value

GRanges object

Examples

```
library(GenomicRanges)

# load data
data('decorateData')

# Evaluate hierarchical clustering
# adjacentCount is the number of adjacent peaks considered in correlation
treeList = runOrderedClusteringGenome( simData, simLocation)

# Choose cutoffs and return cluster
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20, 30, 40, 50)

# Get start and end coordinates for each cluster
# cluster name is parameter:chrom:cluster
getClusterRanges( simLocation, treeListClusters)
```

---

getFeaturesInCluster *Get feature names in selected cluster*

Description

Get feature names in selected cluster given chrom and clusterid

Usage

```
getFeaturesInCluster(treeListClusters, chrom, clustID, id)
```

Arguments

treeListClusters	from createClusters()
chrom	chromosome name of cluster
clustID	cluster identifier
id	clustering parameter identifier

Value

array of feature names

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method='meanClusterSize', meanClusterSize = 50 )
```

```
# Find chromsome and cluster of peak_204
getFeaturesInCluster( treeListClusters, "chr20", 3, "50")
```

`getFeaturesInClusterList` *Get feature names in selected cluster*

Description

Get feature names in selected cluster given array of chrom and cluster ids

Usage

```
getFeaturesInClusterList(treeListClusters, chrom, clustID, id)
```

Arguments

`treeListClusters`	from createClusters()
`chrom`	chromosome name of cluster
`clustID`	cluster identifier
`id`	clustering parameter identifier

Value

list of array of feature names. Query with index i as returned in list index i

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method='meanClusterSize', meanClusterSize = 50 )

df_cluster = data.frame( chrom = c("chr20", "chr20"),
  cluster = c(3,5),
  id = c("50", "50"), stringsAsFactors=FALSE )

# Find features for the clusters in df_cluster
getFeaturesInClusterList( treeListClusters, chrom=df_cluster$chrom, clustID=df_cluster$cluster, id=df_cluste
```

getPeakDistances *Compute distance between peaks*

Description

Given a query set of genome intervals, report distance to all others within window

Usage

```
getPeakDistances(query, windowSize = 10000)
```

Arguments

query	GRanges object of intervals to query
windowSize	with of window around each interval in bp

Details

A smaller window size will create a covariance matrix that is faster to evaluate, but larger windows give a better approximation and are less likely to have negative eigen value

Value

for all pairs of peaks within windowSize, report distance

Examples

```
library(GenomicRanges)

query = GRanges(rep('chr1', 5), IRanges(1:5, 1:5))

getPeakDistances( query )
```

getSubset *Extract subset of data points*

Description

Extract subset of data points

Usage

```
getSubset(fit, query)

## S4 method for signature 'epiclust,GRanges'
getSubset(fit, query)

## S4 method for signature 'epiclustList,GRanges'
getSubset(fit, query)
```

Arguments

fit	epiclust or epiclustList object
query	GenomicRanges object of which data points to retain

Value

epiclust or epiclustList object

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# extract subset of data after clustering
res = getSubset( treeList, simLocation[1:10])
```

---

get_exon_coords *Get coordinates of exons*

Description

Get coordinates of exons from ENSEMBL database

Usage

```
get_exon_coords(ensdb, query, biotypes = c("protein_coding"))
```

Arguments

ensdb	ENSEMBL database object like EnsDb.Hsapiens.v86
query	GRranges ofject of one interval. "chr20" should be coded as "20"
biotypes	gene biotypes to return

Value

GRanges object of exon locations

Examples

```
library(EnsDb.Hsapiens.v86)
library(GenomicRanges)

# gene database
ensdb = EnsDb.Hsapiens.v86

# interval
```

```
query = GRanges("20", IRanges(62045027,62164563))

# get GRanges object of exon locations
get_exon_coords( ensdb, query)
```

`ggplot_by_sampling` *Plot by subsampling in each bin*

Description

Plot by subsampling in each bin in x-axis

Usage

```
ggplot_by_sampling(x, y, N, nbins = 1000)
```

Arguments

`x`	x values
`y`	y values
`N`	number of samples
`nbins`	number of bins on the x-axis

`jaccard` *Evaluate Jaccard index*

Description

Evaluate Jaccard index

Usage

```
jaccard(a, b)
```

Arguments

`a`	set 1
`b`	set 2

Value

Jaccard index

Examples

```
a = 1:10
b = 5:15
jaccard(a,b)
```

makeImageRect *Convert correlation matrix into triangle plot*

Description

Adapted from LDheatmap

Usage

```
makeImageRect(nrow, ncol, cols, name, byrow = TRUE)
```

Arguments

nrow	nrow(C)
ncol	ncol(C)
cols	entries of C converted to color
name	name of the plot
byrow	process C by row

Value

rectGrob

plotClusterSegments *Plot cluster segments*

Description

Plot bar of segments showing clusters

Usage

```
plotClusterSegments(clusterValues)
```

Arguments

clusterValues	array of names of cluster for each entry

Value

ggplot2 of cluster assignments

Examples

```
plotClusterSegments(c(rep(1, 5), rep(2,2), rep(3, 4)))
```

plotCompareCorr *Plot two correlation matrices together*

Description

Combined plot of correlation matricies from cases and controls

Usage

```
plotCompareCorr(
  epiSignal,
  peakIDs,
  testVariable,
  cols,
  size = 5,
  absCorr = FALSE
)
```

Arguments

epiSignal	matrix or EList of epigentic signal. Rows are features and columns are samples
peakIDs	feature names to extract from rows of epiSignal
testVariable	factor indicating two subsets of the samples to compare
cols	array of color values
size	size of text
absCorr	show absolute correlations

Value

ggplot2 of combined correlation matrix

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clsutering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c( 10, 20) )

# Simulate variable to split dataset by
set.seed(1)
metadata = data.frame( Disease = factor(sample(0:1, ncol(simData), replace=TRUE)))

# get peak ID's from chr1, cluster 1
peakIDs = getFeaturesInCluster( treeListClusters, "chr1", 1, "10")

# plot comparison of correlation matrices for peaks in peakIDs
```

```
#  where data is subset by metadata$Disease
plotCompareCorr( simData, peakIDs, metadata$Disease) + ggtitle("chr1: cluster 1")
```

`plotCorrDecay` *Plot correlation delay*

Description

Plot correlation delay using subsampling

Usage

```
plotCorrDecay(
  dfDist,
  method = c("R", "Rsq"),
  xlim = c(10, 1e+06),
  n = 100,
  outlierQuantile = 0.001,
  densityExponent = 0.25
)
```

Arguments

`dfDist`	data.frame of distance and correlation from from evaluateCorrDecay()
`method`	on show either R or Rsq on y-axis
`xlim`	min and max values for x-axis
`n`	the number of equally spaced points at which the density is to be estimated.
`outlierQuantile`	show points if density is less than this quantile
`densityExponent`	color based on density^densityExponent

Details

Plot correlation versus log10 distance. Sample equal number of points for each bin along the x-axis.

Examples

```
library(GenomicRanges)
library(ggplot2)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Evaluate how correlation between features decays with distance
dfDist = evaluateCorrDecay( treeList, simLocation )

# make plot
plotCorrDecay( dfDist )
```

`plotCorrTriangle` *Plot triangle of correlation matrix*

Description

Plot lower triangle of correlation matrix

Usage

```
plotCorrTriangle(
  C,
  size = 1,
  stroke = 1.5,
  cols = c("blue", "white", "red"),
  absCorr = FALSE
)
```

Arguments

`C`	correlation matrix
`size`	plotting argument to geom_point()
`stroke`	plotting argument to geom_point()
`cols`	array of two colors for gradient
`absCorr`	show absolute correlations

Details

Adjust size and stroke of points in the plot to fix look of plot depending on dimensions

Value

ggplot2 plot of correlation matrix

Examples

```
N = 1000
p = 100
X = matrix(rnorm(N*p), N,p)
C = cor(X)
plotCorrTriangle( C )
```

`plotDecorate` *Plot decorate analysis*

Description

Plot decorate analysis for clusters and correlations

Usage

```
plotDecorate(
  ensdb,
  treeList,
  treeListClusters,
  featurePositions,
  query,
  data,
  cols,
  showGenes = TRUE,
  splice_variants = FALSE,
  non_coding = FALSE,
  absCorr = FALSE,
  method.corr = c("pearson", "kendall", "spearman")
)
```

Arguments

`ensdb`	ENSEMBL database object like EnsDb.Hsapiens.v86
`treeList`	hierarchical clustering of each chromosome from runOrderedClusteringGenome()
`treeListClusters`	assign regions to clusters after cutting tree with createClusters()
`featurePositions`	GRanges object storing location of each feature
`query`	GRanges object indiecating region to plot
`data`	data to compute correlations stratified by testVariable
`cols`	array of color values
`showGenes`	plot genes
`splice_variants`	if TRUE, show multiple transcripts from the same gene
`non_coding`	if TRUE, also show non-coding genes
`absCorr`	show absolute correlations
`method.corr`	if data is specified, compute correlation using: "pearson", "kendall", "spearman"

Value

ggplot2 of cluster assignments and correlation between peaks

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clsutering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method='meanClusterSize', meanClusterSize=30)

# Plot correlations and clusters in region defined by query
query = range(simLocation)

plotDecorate( ensdb, treeList, treeListClusters, simLocation, query)
```

`plotDensityPoints` *Plot density as color, add outlier points*

Description

Plot density as color, add outlier points

Usage

```
plotDensityPoints(
  x,
  y,
  n = 100,
  outlierQuantile = 1e-05,
  densityExponent = 0.25
)
```

Arguments

`x`	x values
`y`	y values
`n`	the number of equally spaced points at which the density is to be estimated.
`outlierQuantile`	show points if density is less than this quantile
`densityExponent`	color based on density^densityExponent

Examples

```
x = rnorm(10000)
y = rnorm(10000)

plotDensityPoints( x, y)
```

`plotEnsGenes` *Plot ENSEMBL genes*

Description

Plot ENSEMBL genes in region

Usage

```
plotEnsGenes(
  ensdb,
  minRange,
  maxRange,
  chromosome,
  plot_lines_distance = 0.03,
  vp = viewport(x = 0, y = 0.95, just = c("left", "top")),
  splice_variants = TRUE,
  non_coding = TRUE
)
```

Arguments

`ensdb`	ENSEMBL database object like EnsDb.Hsapiens.v86
`minRange`	start genome coordinate
`maxRange`	end genome coordinate
`chromosome`	chrom
`plot_lines_distance`	veritcal distance between genes
`vp`	viewport
`splice_variants`	if TRUE, show multiple transcripts from the same gene
`non_coding`	if TRUE, also show non-coding genes

Value

GRanges object of exon locations

Examples

```
library(EnsDb.Hsapiens.v86)
library(GenomicRanges)
library(grid)

# gene database
ensdb = EnsDb.Hsapiens.v86

# interval
query = GRanges("20", IRanges(62045027,62164563))

# plot genes
fig = plotEnsGenes( ensdb, start(query), end(query), seqnames(query))
grid.draw( fig )
```

`plotGenes` *Plot genes from a specified region of the human genome.*

Description

Retrieves genes from the UCSC Genome Browser and generate the genes plot.

Usage

```
plotGenes(minRange, maxRange, chromosome, genome = "hg19", plot_lines_distance = 0.03,
vp = viewport(x = 0, y = 0.99, just = c("left", "top")), splice_variants = TRUE,
non_coding = TRUE)
```

Arguments

`minRange`	The sequence minimum range in base pairs.
`maxRange`	The sequence maximum range in base pairs.
`chromosome`	A character string identifying the chromosome.
`genome`	The genome assembly to use. The default is hg19, the most recent human genome assembly on the UCSC genome browser.
`plot_lines_distance`	The distance between the lines of genes plotted.
`vp`	A `viewport`.
`splice_variants`	If `FALSE`, exclude gene splice variants.
`non_coding`	If `FALSE`, exclude non-coding genes.

Details

The genes are color coded as follows: Black – feature has a corresponding entry in the Protein Data Bank (PDB) Dark blue – transcript has been reviewed or validated by either the RefSeq, SwissProt or CCDS staff Medium blue – other RefSeq transcripts Light blue – non-RefSeq transcripts

For assemblies older than hg18, all genes are plotted in grey.

Value

A grob of gene plots.

Author(s)

Sigal Blay <sblay@sfu.ca> and more

References

http://genome.ucsc.edu/cgi-bin/hgTrackUi?g=knownGene

Examples

```
## Not run:
grid.newpage()
plotGenes(149500000, 150000000, "chr7")

## End(Not run)
```

---

plotPairwiseScatter *Scatter plot of all pairs of variables stratified by test variable*

Description

Make a scatterplot for each pair of variables in X and Y. Dataset is divided in two based on value in testVariable

Usage

```
plotPairwiseScatter(
  X,
  Y,
  testVariable,
  size = 1,
  cols = c("#00ff40", "deepskyblue"),
  axisLabels = c("show", "internal", "none"),
  title = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

X	data.frame of variables
Y	data.frame of variables
testVariable	factor indicating two subsets of the samples to compare
size	size of points
cols	color to label samples in for two levels of testVariable
axisLabels	either "show" to display axisLabels, "internal" for labels in the diagonal plots, or "none" for no axis labels

title	title
xlab	xlab
ylab	xlab

Value

ggplot2 of combined pairwise scatter plots

---

plotScatterPairs *Scatter plot of all pairs of variables stratified by test variable*

Description

Scatter plot of all pairs of variables stratified by test variable

Usage

```
plotScatterPairs(epiSignal, peakIDs, testVariable, size = 1)
```

Arguments

epiSignal	matrix or EList of epigentic signal. Rows are features and columns are samples
peakIDs	feature names to extract from rows of epiSignal
testVariable	factor indicating two subsets of the samples to compare
size	size of points

Value

ggplot2 of combined pairwise scatter plots

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clsutering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method = "meanClusterSize", meanClusterSize=c(10, 30) )

# get peak ID's from chr1, cluster 1
peakIDs = getFeaturesInCluster( treeListClusters, "chr20", 2, "30")

# plot comparison of correlation matrices for peaks in peakIDs
#  where data is subset by metadata$Disease
plotScatterPairs( simData, peakIDs, metadata$Disease) + ggtitle("chr20: cluster 1")
```

`retainClusters` *Retain clusters by applying filter*

Description

Retain clusters by applying filter

Usage

```
retainClusters(clstScore, metric = "LEF", cutoff = 0.4)
```

Arguments

`clstScore`	score each cluster using scoreClusters()
`metric`	column of clstScore to use in filtering
`cutoff`	retain cluster than exceed the cutoff for metric. Can be array with one entry per entry in clstScore

Value

data.frame of chrom, clutser, id (the clustering parameter value), and the specified metric

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Evaluate score for each cluster
clstScore = scoreClusters(treeList, treeListClusters )

# Retain clusters that pass this criteria
clustInclude = retainClusters( clstScore, "LEF", 0.30 )

# Or filter by mean absolute correlation
# clustInclude = retainClusters( clstScore, "mean_abs_corr", 0.1 )

# get retained clusters
treeListClusters_filter = filterClusters( treeListClusters, clustInclude )
```

runFastStat *Test difference in correlation using closed form tests*

Description

Test difference in correlation using closed form tests

Usage

```
runFastStat(
  itObj,
 method = c("Box", "Box.permute", "Steiger.fisher", "Steiger", "Jennrich", "Factor",
  "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj",
   "Schott.Frob", "Delaneau", "deltaSLE"),
  method.corr = c("pearson", "kendall", "spearman")
)
```

Arguments

itObj	iterator
method	Specify test: "Box", "Box.permute", Steiger.fisher", "Steiger", "Jennrich", "Factor" "Mann.Whitney", "Kruskal.Wallis", "Cai.max", "Chang.maxBoot", "LC.U", "WL.randProj", "Schott.Frob"
method.corr	Specify type of correlation: "pearson", "kendall", "spearman"

runOrderedClustering *Run hierarchical clustering preserving order*

Description

Run hierarchical clustering preserving sequential order of entries

Usage

```
runOrderedClustering(X, gr, alpha = 0.5)
```

Arguments

X	data matrix were *rows* are features in sequential order
gr	GenomicRanges object corresponding to rows in X
alpha	mixture parameter weigting correlations (alpha=0) versus chromosome distances (alpha=1)

Details

Use hclustgeo in ClustGeo package to generate hierarchical clustering that preserves sequential order.

Chavent, et al. 2017. ClustGeo: an R package for hierarchical clustering with spatial constraints. arXiv:1707.03897v2 doi:10.1007/s00180-018-0791-1

Value

hclust tree

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Plot correlations and clusters in region defined by query
query = range(simLocation)

plotDecorate( ensdb, treeList, treeListClusters, simLocation, query)
```

---

runOrderedClusteringGenome

Run hierarchical clustering preserving order

Description

Run hierarchical clustering preserving sequential order of entries

Usage

```
runOrderedClusteringGenome(
  X,
  gr,
  method = c("adjclust", "hclustgeo"),
  quiet = FALSE,
  alpha = 0.5,
  adjacentCount = 500,
  setNANtoZero = FALSE,
  method.corr = c("pearson", "spearman")
)
```

Arguments

X	data matrix were *rows* are features in sequential order
gr	GenomicRanges object with entries corresponding to the *rows* of X

method	'adjclust': adjacency constrained clustering. 'hclustgeo': incorporate data correlation and distance in bp
quiet	suppress messages
alpha	use by 'hclustgeo': mixture parameter weighing correlations (alpha=0) versus chromosome distances (alpha=1)
adjacentCount	used by 'adjclust': number of adjacent entries to compute correlation against
setNANtoZero	replace NAN correlation values with a zero
method.corr	Specify type of correlation: "pearson", "kendall", "spearman"

Details

Use adjacency constrained clustering from adjclust package:

Alia Dehman, Christophe Ambroise and Pierre Neuvial. 2015. Performance of a blockwise approach in variable selection using linkage disequilibrium information. BMC Bioinformatics 16:148 doi.org:10.1186/s12859-015-0556-6

Or, use hclustgeo in ClustGeo package to generate hierarchical clustering that roughly preserves sequential order.

Chavent, et al. 2017. ClustGeo: an R package for hierarchical clustering with spatial constraints. arXiv:1707.03897v2 doi:10.1007/s00180-018-0791-1

Value

list hclust tree, one entry for each chromosome

Examples

```
library(GenomicRanges)
library(EnsDb.Hsapiens.v86)

# load data
data('decorateData')

# load gene locations
ensdb = EnsDb.Hsapiens.v86

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Plot correlations and clusters in region defined by query
query = range(simLocation)

plotDecorate( ensdb, treeList, treeListClusters, simLocation, query)
```

`runPermutedData` *Run hierarchical clustering permuting features*

Description

Run hierarchical clustering permuting features to get statistics under the null

Usage

```
runPermutedData(
  X,
  gr,
  method = c("adjclust", "hclustgeo"),
  quiet = FALSE,
  alpha = 0.5,
  adjacentCount = 500,
  setNANtoZero = FALSE,
  method.corr = c("pearson", "spearman"),
  meanClusterSize = c(5, 10)
)
```

Arguments

`X`	data matrix were *rows* are features in sequential order
`gr`	GenomicRanges object with entries corresponding to the *rows* of X
`method`	'adjclust': adjacency constrained clustering. 'hclustgeo': incorporate data correlation and distance in bp
`quiet`	suppress messages
`alpha`	use by 'hclustgeo': mixture parameter weighing correlations (alpha=0) versus chromosome distances (alpha=1)
`adjacentCount`	used by 'adjclust': number of adjacent entries to compute correlation against
`setNANtoZero`	replace NAN correlation values with a zero
`method.corr`	Specify type of correlation: "pearson", "kendall", "spearman"
`meanClusterSize`	select target mean cluster size. Can be an array of values

Value

list of clusterScores and cutoff values at 5

Examples

```
library(GenomicRanges)

# load data
data('decorateData')

# First, analysis of original data
# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )
```

```
# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method='meanClusterSize', meanClusterSize=c(5, 10) )

# Evaluate score for each cluster
clstScore = scoreClusters(treeList, treeListClusters )

# Then, analysis of permuted data
# Evaluate hierarchical clustering
res = runPermutedData( simData, simLocation, meanClusterSize=c(5, 10)  )

# LEF values for permuted data at 5% false positive rate
res$cutoffs$LEF

# Retain clusters that pass this criteria
clustInclude = retainClusters( clstScore, "LEF", res$cutoffs$LEF )
```

`scoreClusters` *Compute scores for each cluster*

Description

For each cluster compute summary statistics for the cluster to measure how strong the correlation structure is. Clusters with weak correlation structure can be dropped from downstream analysis.

Usage

```
scoreClusters(treeList, treeListClusters, BPPARAM = bpparam())
```

Arguments

`treeList`	list of hclust objects
`treeListClusters`	from createClusters()
`BPPARAM`	parameters for parallel evaluation

Details

For each cluster, extract the correlation matrix and return the mean absolute correlation; the 75th, 90th and 95th quantile absolute correlation, and LEF, the leading eigen-value fraction which is the fraction of variance explained by the leading eigen value of the matrix abs(C).

Value

for all pairs of peaks within windowSize, report distance

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )

# Evaluate score for each cluster
clstScore = scoreClusters(treeList, treeListClusters )
```

`sle.score` *Score impact of each sample on sparse leading eigen-value*

Description

Score impact of each sample on sparse leading eigen-value. Compute correlation using all samples (i.e. C), then compute correlation omitting sample i (i.e. Ci). The score the sample i is based on sparse leading eigen-value of the diffrence between C and Ci.

Usage

```
sle.score(
  Y,
  method = c("pearson", "kendall", "spearman"),
  rho = 0.05,
  sumabs = 1
)
```

Arguments

`Y`	data matrix with samples on rows and variables on columns
`method`	specify which correlation method: "pearson", "kendall" or "spearman"
`rho`	a positive constant such that cor(Y) + diag(rep(rho,p)) is positive definite.
`sumabs`	regularization paramter. Value of 1 gives no regularization, sumabs*sqrt(p) is the upperbound of the L_1 norm of v,controling the sparsity of solution. Must be between 1/sqrt(p) and 1.

Value

score for each sample measure impact on correlation structure

See Also

sle.test

Examples

```
# load iris data
data(iris)

# Evalaute score on each sample
sle.score( iris[,1:4] )
```

`sle.test` *Test association between sparse leading eigen-value and variable*

Description

Score impact of each sample on sparse leading eigen-value and then peform test of association with variable using non-parametric test

Usage

```
sle.test(
  Y,
  variable,
  method = c("pearson", "kendall", "spearman"),
  rho = 0,
  sumabs = 1
)
```

Arguments

`Y`	data matrix with samples on rows and variables on columns
`variable`	variable with number of entries must equal nrow(Y). Can be discrete or continuous.
`method`	specify which correlation method: "pearson", "kendall" or "spearman"
`rho`	a positive constant such that cor(Y) + diag(rep(rho,p)) is positive definite.
`sumabs`	regularization paramter. Value of 1 gives no regularization, sumabs*sqrt(p) is the upperbound of the L_1 norm of v,controling the sparsity of solution. Must be between 1/sqrt(p) and 1.

Details

The statistical test used depends on the variable specified. if variable is factor with multiple levels, use Kruskal-Wallis test if variable is factor with 2 levels, use Wilcoxon test if variable is continuous, use Wilcoxon test

Value

list of p-value, estimate and method used

See Also

sle.score delaneau.test

Examples

```
# load iris data
data(iris)

# variable is factor with multiple levels
# use kruskal.test
sle.test( iris[,1:4], iris[,5] )

# variable is factor with 2 levels
# use wilcox.test
sle.test( iris[1:100,1:4], iris[1:100,5] )

# variable is continuous
# use cor.test with spearman
sle.test( iris[,1:4], iris[,1] )
```

---

sLEDresults-class *An S4 class that stores results of sLED analysis*

Description

An S4 class that stores results of sLED analysis

Slots

`.Data` list of sLED results

---

summary,sLEDresults-method *Summarize sLED analysis*

Description

extract statistic and p-value for each cluster

Usage

```
## S4 method for signature 'sLEDresults'
summary(object)
```

Arguments

`object`	sLEDresults

Value

data.frame

`whichCluster` *Find which cluster a peak is in*

Description

Find which cluster a peak is in

Usage

```
whichCluster(treeListClusters, feature_id, id = NULL)
```

Arguments

`treeListClusters`	from createClusters()
`feature_id`	name of query feature, can also be array
`id`	clustering parameter identifier. After filtering by feature_id, filter by id

Value

data.frame of chromosome and cluster

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList, method='meanClusterSize', meanClusterSize = c(50, 100) )

# Find chromsome and cluster of peak_20
whichCluster( treeListClusters, 'peak_20')

# Find chromsome and cluster of peak_20 with clustering parameter 50
#  corresponding to meanClusterSize
whichCluster( treeListClusters, 'peak_20', "50")

# Search for multiple clusters
whichCluster( treeListClusters, c('peak_20', 'peak_21'), "50")
```

`[,epiclustDiscreteListContain,ANY,ANY,ANY-method`
Allow subsetting of epiclustDiscreteListContain

Description

Allow subsetting of epiclustDiscreteListContain

Usage

```
## S4 method for signature 'epiclustDiscreteListContain,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

x	epiclustDiscreteListContain
i	index 1
j	index 2
...	additional arguement
drop	TRUE/FALSE

Value

subset of epiclustDiscreteListContain

Examples

```
library(GenomicRanges)

data('decorateData')

# Evaluate hierarchical clustering
treeList = runOrderedClusteringGenome( simData, simLocation )

# Choose cutoffs and return clusters
treeListClusters = createClusters( treeList )
```

Index

∗Topic **decorateData**
    decorateData, 12
.evalDiffCorr, 3
[,epiclustDiscreteListContain,ANY,ANY,ANY-method, 48

boxM, 4
boxM_fast, 4
boxM_permute, 5

collapseClusters, 6
combineResults, 7
corrMatrix.test, 8
corSubsetPairs, 9
countClusters, 10
countClusters,epiclustDiscreteList-method *(countClusters)*, 10
countClusters,epiclustDiscreteListContain-method *(countClusters)*, 10
createClusters, 10
createCorrelationMatrix, 11

decorateData, 12
delaneau.score, 13
delaneau.test, 14

epiclust-class, 15
epiclustDiscreteList-class, 15
epiclustDiscreteListContain-class, 15
epiclustList-class, 15
evalDiffCorr, 16
evalDiffCorr,data.frame,ANY,GRanges,list,ANY,ANY,ANY,ANY,ANY,ANY,ANY-method *(evalDiffCorr)*, 16
evalDiffCorr,data.frame,ANY,GRanges,list-method *(evalDiffCorr)*, 16
evalDiffCorr,EList,ANY,GRanges,list,ANY,ANY,ANY,ANY,ANY,ANY,ANY-method *(evalDiffCorr)*, 16
evalDiffCorr,EList,ANY,GRanges,list-method *(evalDiffCorr)*, 16
evalDiffCorr,matrix,ANY,GRanges,list,ANY,ANY,ANY,ANY,ANY,ANY,ANY-method *(evalDiffCorr)*, 16
evalDiffCorr,matrix,ANY,GRanges,list-method *(evalDiffCorr)*, 16
evaluateCorrDecay, 18
extractCorrelationScores, 19

filterClusters, 21

get_exon_coords, 26
getClusterNames, 21
getClusterRanges, 22
getFeaturesInCluster, 23
getFeaturesInClusterList, 24
getPeakDistances, 25
getSubset, 25
getSubset,epiclust,GRanges-method *(getSubset)*, 25
getSubset,epiclustList,GRanges-method *(getSubset)*, 25
ggplot_by_sampling, 27

jaccard, 27

makeImageRect, 28
metadata *(decorateData)*, 12

plotClusterSegments, 28
plotCompareCorr, 29
plotCorrDecay, 30
plotCorrTriangle, 31
plotDecorate, 32
plotDensityPoints, 33
plotEnsGenes, 34
plotGenes, 35
plotPairwiseScatter, 36
plotScatterPairs, 37
retainClusters, 38
runFastStat, 39
runOrderedClustering, 39
runOrderedClusteringGenome, 40
runPermutedData, 42

scoreClusters, 43
simData *(decorateData)*, 12
simLocation *(decorateData)*, 12
sle.score, 44
sle.test, 45
sLEDresults-class, 46

summary,sLEDresults-method, 46

whichCluster, 47