Package 'COCOA'

November 11, 2026

Version 2.26.0

Date 2025-06-25

Title Coordinate Covariation Analysis

Description COCOA is a method for understanding epigenetic variation among samples. COCOA can be used with epigenetic data that includes genomic coordinates and an epigenetic signal, such as DNA methylation and chromatin accessibility data. To describe the method on a high level, COCOA quantifies inter-sample variation with either a supervised or unsupervised technique then uses a database of ``region sets'' to annotate the variation among samples. A region set is a set of genomic regions that share a biological annotation, for instance transcription factor (TF) binding regions, histone modification regions, or open chromatin regions. COCOA can identify region sets that are associated with epigenetic variation between samples and increase understanding of variation in your data.

Depends R (>= 3.5), GenomicRanges

Imports BiocGenerics, S4Vectors, IRanges, data.table, ggplot2, Biobase, stats, methods, ComplexHeatmap, MIRA, tidyr, grid, grDevices, simpleCache, fitdistrplus

Suggests knitr, parallel, testthat, BiocStyle, rmarkdown, AnnotationHub, LOLA

VignetteBuilder knitr

License GPL-3

biocViews Epigenetics, DNAMethylation, ATACSeq, DNaseSeq, MethylSeq, MethylationArray, PrincipalComponent, GenomicVariation, GeneRegulation, GenomeAnnotation, SystemsBiology, FunctionalGenomics, ChIPSeq, Sequencing, ImmunoOncology

URL http://code.databio.org/COCOA/

BugReports https://github.com/databio/COCOA

RoxygenNote 7.1.1

git_url https://git.bioconductor.org/packages/COCOA

git_branch RELEASE_3_23

git_last_commit 422ea42

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author John Lawson [aut, cre],
Nathan Sheffield [aut] (http://www.databio.org),
Jason Smith [ctb]

Maintainer John Lawson <jtl2hk@virginia.edu>

Contents

aggregateSignal . . . 3
aggregateSignalGRList . . . 5
atf3_chr1 . . . 8
brcaATACCoord1 . . . 9
brcaATACData1 . . . 9
brcaMCoord1 . . . 10
brcaMetadata . . . 10
brcaMethylData1 . . . 11
brcaPCScores . . . 11
brcaPCScores657 . . . 12
COCOA . . . 12
convertToFromNullDist . . . 13
esr1_chr1 . . . 14
gata3_chr1 . . . 14
getGammaPVal . . . 15
getMetaRegionProfile . . . 16
getPermStat . . . 18
getTopRegions . . . 20
nrf1_chr1 . . . 21
plotAnnoScoreDist . . . 21
regionQuantileByTargetVar . . . 23
rsRankingIndex . . . 25
rsScoreHeatmap . . . 26
rsScores . . . 28
runCOCOA . . . 28
runCOCOAPerm . . . 32
signalAlongAxis . . . 36

Index **39**

aggregateSignal *Score a region set using feature contribution scores*

Description

First, this function identifies which epigenetic features overlap the region set. Then the region set is scored using the feature contribution scores (‘signal‘ input) according to the ‘scoringMetric‘ parameter.

Usage

```
aggregateSignal(
  signal,
  signalCoord,
  regionSet,
  signalCol = c("PC1", "PC2"),
  signalCoordType = "default",
  scoringMetric = "default",
  verbose = FALSE,
  absVal = TRUE,
  rsOL = NULL,
  pOlap = NULL,
  returnCovInfo = TRUE,
  .checkInput = TRUE
)
```

Arguments

signal	Matrix of feature contribution scores (the contribution of each epigenetic feature to each target variable). One named column for each target variable. One row for each original epigenetic feature (should be same order as original data/signalCoord). For (an unsupervised) example, if PCA was done on epigenetic data and the goal was to find region sets associated with the principal components, you could use the x$rotation output of prcomp(epigenetic data) as the feature contribution scores/‘signal‘ parameter.
signalCoord	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
regionSet	A genomic ranges (GRanges) object with regions corresponding to the same biological annotation. Must be from the same reference genome as the coordinates for the actual data/samples (signalCoord).
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes).

signalCoordType

Character. Can be "default", "singleBase", or "multiBase". This describes whether the coordinates for ‘signal‘ (‘signalCoord‘) are each a single base (e.g. as for DNA methylation) or a region/multiple bases (e.g. as for chromatin accessibility). Different scoring options are available for each type of data. If "default" is given, the type of coordinates will be detected automatically. For "default", if each coordinate start value equals the coordinate end value (all(start(signalCoord) == end(signalCoord))), "singleBase" will be used. Otherwise, "multiBase" will be used.

scoringMetric

A character object with the scoring metric. There are different methods available for signalCoordType="singleBase" vs signalCoordType="multiBase". For "singleBase", the available methods are "regionMean", "regionMedian", "simpleMean", and "simpleMedian". The default method is "regionMean". For "multiBase", the methods are "proportionWeightedMean", "simpleMean", and "simpleMedian". The default is "proportionWeightedMean". "regionMean" is a weighted average of the signal, weighted by region (absolute value of signal if absVal=TRUE). First the signal is averaged within each regionSet region, then all the regions are averaged. With "regionMean" method, be cautious in interpretation for region sets with low number of regions that overlap signalCoord. The "regionMedian" method is the same as "regionMean" but the median is taken at each step instead of the mean. The "simpleMean" method is just the unweighted average of all (absolute) signal values that overlap the given region set. For multiBase data, this includes signal regions that overlap a regionSet region at all (1 base overlap or more) and the signal for each overlapping region is given the same weight for the average regardless of how much it overlaps. The "simpleMedian" method is the same as "simpleMean" but takes the median instead of the mean. "proportionWeightedMean" is a weighted average of all signalCoord regions that overlap with regionSet regions. For each signalCoord region that overlaps with a regionSet region, we calculate what proportion of the regionSet region is covered. Then this proportion is used to weight the signal value when calculating the mean. The denominator of the mean is the sum of all the proportion overlaps.

verbose

A "logical" object. Whether progress of the function should be shown. One bar indicates the region set is completed.

absVal

Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

rsOL

a "SortedByQueryHits" object (output of findOverlaps function). Should have the overlap information between signalCoord and one item of GRList (one unique region set). The region set must be the "subject" in findOverlaps and signalCoord must be the "query". E.g. findOverlaps(subject=regionSet, query=signalCoord). Providing this information can greatly improve permutation speed since the overlaps will not have to be calculated for each permutation. When using this parameter, signalCoord, genomicSignal, and the region set must be in the same order as they were when olList was created. Otherwise, the wrong genomic loci

will be referenced (e.g. if epigenetic features were filtered out of genomicSignal after rsOL was created.)

`pOlap` Numeric vector. Only used if rsOL is given and scoringMetric is "proportionWeightedMean". This vector should contain the proportion of each regionSet region that is overlapped by a signalCoord region. The order of pOlap should be the same as the overlaps in rsOL.

`returnCovInfo` logical. If TRUE, the following coverage and region set info will be calculated and included in function output: regionSetCoverage, signalCoverage, totalRegionNumber, and meanRegionSize. For the proportionWeightedMean scoring method, sumProportionOverlap will also be calculated.

`.checkInput` A "logical" object. For programmatic use only. Whether inputs to the function should be checked for correctness/appropriateness. This parameter may be used by some COCOA functions to prevent unnecessary checks of objects after arguments have already been checked once.

Value

A data.frame with one row and the following columns: one column for each item of signalCol with names given by signalCol. These columns have scores for the region set for each signalCol. Other columns: signalCoverage (formerly cytosine_coverage) which has number of epigenetic features that overlapped at all with regionSet, regionSetCoverage which has number of regions from regionSet that overlapped any of the epigenetic features, totalRegionNumber that has number of regions in regionSet, meanRegionSize that has average size in base pairs of regions in regionSet, the average is based on all regions in regionSet and not just ones that overlap. For "multiBase" data, if the "proportionWeightedMean" scoring metric is used, then the output will also have a "sumProportionOverlap" column. During this scoring method, the proportion overlap between each signalCoord region and overlapping regionSet region is calculated. This column is the sum of all those proportion overlaps and is another way to quantify coverage of regionSet in addition to regionSetCoverage.

Examples

```
data("brcaATACCoord1")
data("brcaATACData1")
data("esr1_chr1")
featureContributionScores <- prcomp(t(brcaATACData1))$rotation
rsScores <- aggregateSignal(signal=featureContributionScores,
                                signalCoord=brcaATACCoord1,
                                regionSet=esr1_chr1,
                                signalCol=c("PC1", "PC2"),
                                scoringMetric="default")
```

`aggregateSignalGRList` *Score many region sets*

Description

This function will give each region set a score for each target variable given by ‘signalCol‘ based on the ‘scoringMetric‘ parameter. Based on these scores, you can determine which region sets out of a region set database (given by ‘GRList‘) are most associated with the target variables. See the vignette "Introduction to Coordinate Covariation Analysis" for help interpreting your results.

Usage

```
aggregateSignalGRList(
  signal,
  signalCoord,
  GRList,
  signalCol = c("PC1", "PC2"),
  signalCoordType = "default",
  scoringMetric = "default",
  verbose = TRUE,
  absVal = TRUE,
  olList = NULL,
  pOlapList = NULL,
  returnCovInfo = TRUE
)
```

Arguments

signal	Matrix of feature contribution scores (the contribution of each epigenetic feature to each target variable). One named column for each target variable. One row for each original epigenetic feature (should be same order as original data/signalCoord). For (an unsupervised) example, if PCA was done on epigenetic data and the goal was to find region sets associated with the principal components, you could use the x$rotation output of prcomp(epigenetic data) as the feature contribution scores/‘signal‘ parameter.
signalCoord	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
GRList	GRangesList object. Each list item is a distinct region set to test (region set: regions that correspond to the same biological annotation). The region set database must be from the same reference genome as the coordinates for the actual data/samples (signalCoord).
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes).
signalCoordType	Character. Can be "default", "singleBase", or "multiBase". This describes whether the coordinates for ‘signal‘ (‘signalCoord‘) are each a single base (e.g. as for DNA methylation) or a region/multiple bases (e.g. as for chromatin accessibility). Different scoring options are available for each type of data.

If "default" is given, the type of coordinates will be detected automatically. For "default", if each coordinate start value equals the coordinate end value (all(start(signalCoord) == end(signalCoord))), "singleBase" will be used. Otherwise, "multiBase" will be used.

`scoringMetric` A character object with the scoring metric. There are different methods available for signalCoordType="singleBase" vs signalCoordType="multiBase". For "singleBase", the available methods are "regionMean", "regionMedian", "simpleMean", and "simpleMedian". The default method is "regionMean". For "multiBase", the methods are "proportionWeightedMean", "simpleMean", and "simpleMedian". The default is "proportionWeightedMean". "regionMean" is a weighted average of the signal, weighted by region (absolute value of signal if absVal=TRUE). First the signal is averaged within each regionSet region, then all the regions are averaged. With "regionMean" method, be cautious in interpretation for region sets with low number of regions that overlap signalCoord. The "regionMedian" method is the same as "regionMean" but the median is taken at each step instead of the mean. The "simpleMean" method is just the unweighted average of all (absolute) signal values that overlap the given region set. For multiBase data, this includes signal regions that overlap a regionSet region at all (1 base overlap or more) and the signal for each overlapping region is given the same weight for the average regardless of how much it overlaps. The "simpleMedian" method is the same as "simpleMean" but takes the median instead of the mean. "proportionWeightedMean" is a weighted average of all signalCoord regions that overlap with regionSet regions. For each signalCoord region that overlaps with a regionSet region, we calculate what proportion of the regionSet region is covered. Then this proportion is used to weight the signal value when calculating the mean. The denominator of the mean is the sum of all the proportion overlaps.

`verbose` A "logical" object. Whether progress of the function should be shown. One bar indicates the region set is completed.

`absVal` Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

`olList` list. Each list item should be a "SortedByQueryHits" object (output of findOverlaps function). Each hits object should have the overlap information between signalCoord and one item of GRList (one unique region set). The region sets from GRList must be the "subject" in findOverlaps and signalCoord must be the "query". E.g. findOverlaps(subject=regionSet, query=signalCoord). Providing this information can greatly improve permutation speed since the overlaps will not have to be calculated for each permutation. The "runCOCOAPerm" function calculates this information only once, internally, so this does not have to be provided when using that function. When using this parameter, signalCoord, genomicSignal, and each region set must be in the same order as they were when olList was created. Otherwise, the wrong genomic loci will be referenced (e.g. if epigenetic features were filtered out of genomicSignal after olList was created.)

pOlapList — list. This parameter is only used if the scoring metric is "proportionWeightedMean" and olList is also provided as an argument. Each item of the list should be a vector that contains the proportion overlap between signalCoord and regions from one region set (one item of GRList). Specifically, each value should be the proportion of the region set region that is overlapped by a signalCoord region. The proportion overlap values should be in the same order as the overlaps given by olList for the corresponding region set.

returnCovInfo — logical. If TRUE, the following coverage and region set info will be calculated and included in function output: regionSetCoverage, signalCoverage, totalRegionNumber, and meanRegionSize. For the proportionWeightedMean scoring method, sumProportionOverlap will also be calculated.

Value

Data.frame of results, one row for each region set. It has the following columns: one column for each item of signalCol with names given by signalCol. These columns have scores for the region set for each signalCol. Other columns: signalCoverage (formerly cytosine_coverage) which has number of epigenetic features that overlapped at all with regionSet, regionSetCoverage which has number of regions from regionSet that overlapped any of the epigenetic features, totalRegionNumber that has number of regions in regionSet, meanRegionSize that has average size in base pairs of regions in regionSet, the average is based on all regions in regionSet and not just ones that overlap. For "multiBase" data, if the "proportionWeightedMean" scoring metric is used, then the output will also have a "sumProportionOverlap" column. During this scoring method, the proportion overlap between each signalCoord region and overlapping regionSet region is calculated. This column is the sum of all those proportion overlaps and is another way to quantify coverage of regionSet in addition to regionSetCoverage.

Examples

```
data("brcaATACCoord1")
data("brcaATACData1")
data("esr1_chr1")
data("nrf1_chr1")
featureContributionScores <- prcomp(t(brcaATACData1))$rotation
GRList <- GRangesList(esr1_chr1, nrf1_chr1)
rsScores <- aggregateSignalGRList(signal=featureContributionScores,
                                  signalCoord=brcaATACCoord1,
                                  GRList= GRList,
                                  signalCol=c("PC1", "PC2"),
                                  scoringMetric="default")
```

---

atf3_chr1 *Atf3 binding regions.*

Description

Binding regions for Atf3. hg38 genome version. Only includes regions in chr1 to keep the example data small.

Usage

```
data(atf3_chr1)
```

Format

A GRanges object

`brcaATACCoord1` *A GRanges object with coordinates for select BRCA ATAC-seq peak regions from chr1.*

Description

Corresponds to the rows of brcaATACData1. The ATAC-seq data is from breast cancer patients from The Cancer Genome Atlas (TCGA-BRCA, Corces et. al, 2018, doi: 10.1126/science.aav1898, https://atacseq.xenahubs.net/download/brca/brca_peak_Log2Counts_dedup). Coordinates correspond to the hg38 genome version. Only select regions on chr1 are included to keep the example data small.

Usage

```
data(brcaATACCoord1)
```

Format

A GRanges object

`brcaATACData1` *A matrix with ATAC-seq counts in select peak regions from chromosome 1 for 37 patients.*

Description

Each row corresponds to one region and the coordinates for these regions are in the object brcaATACCoord1, (data("brcaATACCoord1"), hg38 genome). Only select regions on chr1 are included to keep the example data small. Columns are patients, with TCGA patient identifiers as column names. 4053 regions are included. ATAC-seq data is from breast cancer patients from The Cancer Genome Atlas (TCGA-BRCA, Corces et. al, 2018, doi: 10.1126/science.aav1898, https://atacseq.xenahubs.net/download/brca/brca_peak_Log2Counts_dedup).

Usage

```
data(brcaATACData1)
```

Format

A matrix object

brcaMCoord1 *A GRanges object with genomic coordinates for cytosines from chr1 for the package's built-in DNA methylation data*

Description

Corresponds to the rows of brcaMethylData1. DNA methlyation data is Illumina 450k microarray data from breast cancer patients from The Cancer Genome Atlas (TCGA-BRCA, https://portal.gdc.cancer.gov/). Coordinates correspond to the hg38 genome version. Only selected cytosines on chr1 are included to keep the example data small.

Usage

```
data(brcaMCoord1)
```

Format

A GRanges object

brcaMetadata *A data.frame with patient metadata for breast cancer patients.*

Description

Has metadata for patients for which DNA methylation or chromatin accessibility data was included as package data (329 patients). Rows are patients, with TCGA patient identifiers as row names and the column "subject_ID". Also includes columns: ER_status, ER_percent, age_diagnosis, days_to_death, and days_to_last_follow_up. Metadata is from The Cancer Genome Atlas (TCGA-BRCA, https://portal.gdc.cancer.gov/).

Usage

```
data(brcaMetadata)
```

Format

A data.frame object

brcaMethylData1 — *A matrix with DNA methylation levels from some CpGs on chromosome 1*

Description

This object contains methylation levels (0 to 1) for select cytosines in chromosome 1 for TCGA breast cancer patients from a DNA methylation microarray (Illumina 450k microarray). Each row corresponds to one cytosine and the coordinates for these cytosines are in the object brcaMCoord1, (data("brcaMCoord1"), hg38 genome). Only select cytosines on chr1 are included to keep the example data small. Columns are patients, with TCGA patient identifiers as column names. 6004 CpGs and 300 patients are included. DNA methlyation data is Illumina 450k microarray data from breast cancer patients from The Cancer Genome Atlas (TCGA-BRCA, https://portal.gdc.cancer.gov/).

Usage

```
data(brcaMethylData1)
```

Format

A matrix object

brcaPCScores — *A matrix with principal component scores for PCs 1-4 for four breast cancer patients.*

Description

This object contains PC scores for four patients for PCs 1-4. Columns are PCs. Rows are patients, with TCGA patient identifiers as row names. DNA methlyation data is Illumina 450k microarray data from breast cancer patients from The Cancer Genome Atlas (TCGA-BRCA, https://portal.gdc.cancer.gov/), hg38 genome.

Usage

```
data(brcaPCScores)
```

Format

A matrix object

`brcaPCScores657` *A data.frame with principal component scores for PCs 1-4 for 657 breast cancer patients as well as a column with estrogen receptor status.*

Description

This object contains PC scores for 657 patients for PCs 1-4. Columns are PCs as well as a column with estrogen receptor status. Rows are patients, with TCGA patient identifiers as row names. Patients were selected from all BRCA patients in TCGA based on having complete metadata information for estrogen receptor status and progesterone receptor status as well as having 450k microarray data. PCA was done on the Illumina 450k DNA methlyation microarray data (TCGA-BRCA, https://portal.gdc.cancer.gov/), hg38 genome.

Usage

```
data(brcaPCScores657)
```

Format

A data.frame object

`COCOA` *Coordinate Covariation Analysis (COCOA)*

Description

COCOA is a method for understanding epigenetic variation among samples. COCOA can be used with epigenetic data that includes genomic coordinates and an epigenetic signal, such as DNA methylation and chromatin accessibility data. To describe the method on a high level, COCOA quantifies inter-sample variation with either a supervised or unsupervised technique then uses a database of "region sets" to annotate the variation among samples. A region set is a set of genomic regions that share a biological annotation, for instance transcription factor (TF) binding regions, histone modification regions, or open chromatin regions. COCOA can identify region sets that are associated with epigenetic variation between samples and increase understanding of variation in your data.

Author(s)

John Lawson

Nathan Sheffield

References

http://github.com/databio

convertToFromNullDist *Converts COCOA permutation results to null distributions and vice versa*

Description

This function will take a list of results of permutation tests that included many region sets and return a list of data.frames where each data.frame contains the null distribution for a single region set. The function can also convert in the reverse order from a list of null distributions to a list of COCOA results.

Usage

```
convertToFromNullDist(rsScoresList)
```

Arguments

rsScoresList	each item in the list is a data.frame, one item for each permutation with the results of that permutation. Each row in the data.frame is a region set. All data.frames should be the same size and each data.frame's rows should be in the same order

Value

a list of data.frames. If given a list where each item is a data.frame with results from one COCOA permutation, this function will return a list of data.frames where each data.frame contains the null distributions for a single region set. The output data.frames will have the same columns as the input data.frames. If given a list where each item is a data.frame with the null distribution/s for a single region set, this function will return a list where each item is a data.frame with one row for each region set (e.g. a data.frame with results for a single COCOA permutation).

Examples

```
# six region sets (rows), 2 signals (columns)
fakePermScores <- data.frame(abs(rnorm(6)), abs(rnorm(6)))
fakePermScores2 <- data.frame(abs(rnorm(6)), abs(rnorm(6)))
# 2 fake COCOA results (i.e. nPerm=2)
permRSScores <- list(fakePermScores, fakePermScores2)
convertToFromNullDist(permRSScores)
```

esr1_chr1 *Estrogen receptor alpha binding regions.*

Description

Binding regions for estrogen receptor alpha (ESR1). hg 38 genome version. Only includes regions in chr1 to keep the example data small.

Usage

```
data(esr1_chr1)
```

Format

A GRanges object

gata3_chr1 *Gata3 binding regions.*

Description

Binding regions for gata3. hg38 genome version. Only includes regions in chr1 to keep the example data small.

Usage

```
data(gata3_chr1)
```

Format

A GRanges object

getGammaPVal *Get a p-value for region set scores based on a gamma distribution.*

Description

First fit a gamma distribution to each region set's null distribution/s (nullDistList). Then use this gamma distribution to convert scores in rsScores to p-values.

Usage

```
getGammaPVal(
  rsScores,
  nullDistList,
  signalCol,
  method = "mme",
  realScoreInDist = TRUE,
  force = FALSE
)
```

Arguments

rsScores	data.frame. A data.frame with region set scores. The output of the 'aggregateSignalGRList' function. Each row is a region set. One column for each sample variable of interest (e.g. PC or sample phenotype). Also can have columns with info on the overlap between the region set and the epigenetic data. Rows should be in the same order as the region sets in GRList (the list of region sets used to create rsScores.)
nullDistList	list of data.frames. Each list item has null distributions for a single region set (list items should be in the same order as rows of rsScores). Has same score columns as rsScores. Each column corresponds to a null distribution for that region set for a given sample variable of interest/target variable (e.g. PC or sample phenotype).
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). Must be column names of rsScores.
method	Character. Has the method to use to fit the gamma distribution to the null distribution. Options are "mme" (moment matching estimation), "mle" (maximum likelihood estimation), "qme" (quantile matching estimation), and "mge" (maximum goodness-of-fit estimation). See ?fitdistrplus::fitdist() for available options and meaning.
realScoreInDist	logical. Should the actual score (from test with no permutations) be included in the null distribution when fitting the gamma distribution. realScoreInDist=TRUE is recommended.
force	logical. If force=TRUE, when fitting the gamma distribution returns an error (as may happen when a method other than "mme" is used) then allow the error. If

force=FALSE, when fitting the gamma distribution returns an error then don't return an error but instead use the "mme" method for fitting that specific gamma distribution.

Value

Returns a data.frame with p values, one column for each signalCol in rsScores

---

`getMetaRegionProfile` *Create a "meta-region" profile*

Description

This profile can show enrichment of genomic signals with high feature contribution scores in the region set but not in the surrounding genome, suggesting that variation is linked specifically to that region set.

Usage

```
getMetaRegionProfile(
  signal,
  signalCoord,
  regionSet,
  signalCol = c("PC1", "PC2"),
  signalCoordType = "default",
  binNum = 21,
  verbose = TRUE,
  aggrMethod = "default",
  absVal = TRUE
)
```

Arguments

`signal`	Matrix of feature contribution scores (the contribution of each epigenetic feature to each target variable). One named column for each target variable. One row for each original epigenetic feature (should be same order as original data/signalCoord). For (an unsupervised) example, if PCA was done on epigenetic data and the goal was to find region sets associated with the principal components, you could use the x$rotation output of prcomp(epigenetic data) as the feature contribution scores/'signal' parameter.
`signalCoord`	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
`regionSet`	A genomic ranges (GRanges) object with regions corresponding to the same biological annotation.

signalCol A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes).

signalCoordType
Character. Can be "default", "singleBase", or "multiBase". This describes whether the coordinates for ‘signal‘ (‘signalCoord‘) are each a single base (e.g. as for DNA methylation) or a region/multiple bases (e.g. as for chromatin accessibility). Different scoring options are available for each type of data. If "default" is given, the type of coordinates will be detected automatically. For "default", if each coordinate start value equals the coordinate end value (all(start(signalCoord) == end(signalCoord))), "singleBase" will be used. Otherwise, "multiBase" will be used.

binNum Number of bins to split each region into when making the aggregate profile. More bins will give a higher resolution but perhaps more noisy profile.

verbose A "logical" object. Whether progress of the function should be shown. One bar indicates the region set is completed.

aggrMethod character. A character object with the aggregation method. Similar to aggregateSignalGRList ‘scoringMetric‘ parameter. There are different methods available for signalCoordType="singleBase" vs signalCoordType="multiBase". For "singleBase", the available methods are "regionMean", "regionMedian", "simpleMean", and "simpleMedian". The default method is "regionMean". For "multiBase", the methods are "proportionWeightedMean", "simpleMean", and "simpleMedian". The default is "proportionWeightedMean". "regionMean" is a weighted average of the signal, weighted by region (absolute value of signal if absVal=TRUE). First the signal is averaged within each regionSet region, then all the regions are averaged. With "regionMean" method, be cautious in interpretation for region sets with low number of regions that overlap signalCoord. The "regionMedian" method is the same as "regionMean" but the median is taken at each step instead of the mean. The "simpleMean" method is just the unweighted average of all (absolute) signal values that overlap the given region set. For multiBase data, this includes signal regions that overlap a regionSet region at all (1 base overlap or more) and the signal for each overlapping region is given the same weight for the average regardless of how much it overlaps. The "simpleMedian" method is the same as "simpleMean" but takes the median instead of the mean. "proportionWeightedMean" is a weighted average of all signalCoord regions that overlap with regionSet regions. For each signalCoord region that overlaps with a regionSet region, we calculate what proportion of the regionSet region is covered. Then this proportion is used to weight the signal value when calculating the mean. The denominator of the mean is the sum of all the proportion overlaps.

absVal Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

Details

All regions in a given region set are combined into a single aggregate profile. Regions in ‘regionSet‘ should be expanded on each side to include a wider area of the genome around the regions of interest (see example and vignettes). To make the profile, first we optionally take the absolute value of ‘signal‘ (‘absVal‘ parameter). Then each expanded regionSet region is split into ‘binNum‘ bins. The corresponding bins from each region (e.g. all bin1’s, all bin2’s, etc.) are grouped. All overlapping values from ‘signal‘ are aggregated in each bin group according to the ‘aggrMethod‘ parameter to get a meta-region profile. Since DNA strand information is not considered, the profile is averaged symmetrically around the center. A peak in the middle of this profile suggests that variability is specific to the region set of interest and is not a product of the surrounding genome. A region set can still be significant even if it does not have a peak. For example, some histone modification region sets may be in large genomic blocks and not show a peak, despite having variation across samples.

Value

A data.frame with the binned meta-region profile, one row per bin. columns: binID and one column for each target variable in signalCol. The function will return NULL if there is no overlap between signalCoord and any of the bin groups that come from regionSet (e.g. none of the bin1’s overlapped signalCoord, NULL returned).

Examples

```
data("brcaATACCoord1")
data("brcaATACData1")
data("esr1_chr1")
featureContributionScores <- prcomp(t(brcaATACData1))$rotation
esr1_chr1_expanded <- resize(esr1_chr1, 12000, fix="center")
mrProfile <- getMetaRegionProfile(signal=featureContributionScores,
                                  signalCoord=brcaATACCoord1,
                                  regionSet=esr1_chr1_expanded,
                                  signalCol=c("PC1", "PC2"),
                                  binNum=21)
```

---

getPermStat *Get p-value or z-score based on permutation results*

Description

This function starts with real COCOA scores for each region set and null distributions for each region set that come from running COCOA on permuted data. Then this function uses the null distributions to get an empirical p-value or z-score for each region set. See vignettes for the workflow that leads to this function. The calculation of the p-value/z-score does not include the real region set score in the null distribution.

Usage

```
getPermStat(
  rsScores,
  nullDistList,
  signalCol,
  testType = "greater",
  whichMetric = "pval"
)
```

Arguments

rsScores	data.frame. A data.frame with region set scores. The output of the 'aggregateSignalGRList' function. Each row is a region set. One column for each sample variable of interest (e.g. PC or sample phenotype). Also can have columns with info on the overlap between the region set and the epigenetic data. Rows should be in the same order as the region sets in GRList (the list of region sets used to create rsScores.)
nullDistList	List. one item per region set. Each item is a data.frame with the null distribution/s for a single region set. Each column in the data.frame is for a target variable (e.g. PC or phenotype), which is given by the 'signalCol' parameter (each target variable has a different null distribution for a given region set).
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). Must be column names of rsScores.
testType	Character. "greater", "lesser", "two-sided" Whether to create p values based on one sided test or not. Only applies when whichMetric="pval".
whichMetric	Character. Can be "pval" or "zscore"

Value

A data.table/data.frame. If whichMetric="pval", returns the empirical p-value for each region set in 'rsScores'. If the region set score is more extreme than all scores in the null distribution, a p-value of 0 is returned but this simply means the p-value is the minimum detectable p-value with the given number of permutations used to make the null distributions. If whichMetric="zscore", the function returns a z-score for each region set score: ((region set score) - mean(null distribution)) / sd(null distribution)

Examples

```
fakeOriginalScores <- data.frame(PC1=abs(rnorm(6)), PC2=abs(rnorm(6)))
fakePermScores <- data.frame(PC1=abs(rnorm(6)), PC2=abs(rnorm(6)))
fakePermScores2 <- data.frame(PC1=abs(rnorm(6)), PC2=abs(rnorm(6)))
fakePermScores3 <- data.frame(PC1=abs(rnorm(6)), PC2=abs(rnorm(6)))
permRSScores <- list(fakePermScores, fakePermScores2, fakePermScores3)
nullDistList <- convertToFromNullDist(permRSScores)
getPermStat(rsScores=fakeOriginalScores, nullDistList=nullDistList,
            signalCol=c("PC1", "PC2"), whichMetric="pval")
getPermStat(rsScores=fakeOriginalScores, nullDistList=nullDistList,
            signalCol=c("PC1", "PC2"), whichMetric="zscore")
```

`getTopRegions` *Get regions that are most associated with target variable*

Description

Get a GRanges with top regions from the region set based on average feature contribution scores for the regions or the quantile of the region's average feature contribution score based on the distribution of all feature contribution scores for the target variable. Returns average feature contribution score or quantile as GRanges metadata.

Usage

```
getTopRegions(
  signal,
  signalCoord,
  regionSet,
  signalCol = c("PC1", "PC2"),
  cutoff = 0.8,
  returnQuantile = TRUE
)
```

Arguments

`signal`	Matrix of feature contribution scores (the contribution of each epigenetic feature to each target variable). One named column for each target variable. One row for each original epigenetic feature (should be same order as original data/signalCoord). For (an unsupervised) example, if PCA was done on epigenetic data and the goal was to find region sets associated with the principal components, you could use the x$rotation output of prcomp(epigenetic data) as the feature contribution scores/'signal' parameter.
`signalCoord`	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
`regionSet`	A genomic ranges (GRanges) object with regions corresponding to the same biological annotation.
`signalCol`	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes).
`cutoff`	Numeric. Only regions with at least this value will be returned (either above this average 'signal' value or above this quantile if returnQuantile=TRUE).
`returnQuantile`	Logical. If FALSE, return region averages. If TRUE, for each region, return the quantile of that region's average value based on the distribution of individual feature values in 'signal' for that 'signalCol'.

Value

A GRanges object with region coordinates for regions with scores/quantiles above "cutoff" for any target variable in signalCol. The scores/quantiles for signalCol are given as metadata in the GRanges.

Examples

```
data("brcaATACCoord1")
data("brcaATACData1")
data("esr1_chr1")
featureContributionScores <- prcomp(t(brcaATACData1))$rotation
topRegions <- getTopRegions(signal=featureContributionScores,
                            signalCoord=brcaATACCoord1,
                            regionSet=esr1_chr1,
                            returnQuantile = TRUE)
```

---

nrf1_chr1 *Nrf1 binding regions.*

Description

Binding regions for Nrf1. hg38 genome version. Only includes regions in chr1 to keep the example data small.

Usage

```
data(nrf1_chr1)
```

Format

A GRanges object

---

plotAnnoScoreDist *Plot ranked region set scores, annotating groups of interest*

Description

Visualize the distribution of region set scores for region set groups of interest.

Usage

```
plotAnnoScoreDist(
  rsScores,
  colToPlot,
  pattern,
  patternName = pattern,
  rsNameCol = "rsName",
  alpha = 0.5,
  shape = 3
)
```

Arguments

rsScores	data.frame. Each row should be a region set. Columns should include score columns and a column that contains the name of each region set.
colToPlot	character. Name of the column with region set scores to plot.
pattern	character. Region sets that match each pattern will be given the same color. Multiple patterns can be given as character objects in a vector (each will have a different color). Regular expressions can be used (ignore.case=TRUE though). For example, to search for ER or GATA3 and color with a single color, this pattern can be given "ER|GATA3".
patternName	character. A name for each string in "pattern" that will be used for the legend.
rsNameCol	character. Column name of "rsScores" column that contains the name or description of each region set. This column will be searched for the pattern given by the "pattern" parameter.
alpha	numeric. Transparency of points. See ggplot documentation for further details.
shape	integer. Shape of the points. See ggplot documentation for options.

Details

If the same region set matches two patterns, the later group will be assigned to that region set.

Value

ggplot object that can be modified using ggplot syntax. E.g. plot + ggplot_function

Examples

```
data(rsScores)
rsScores$rsName <- c("ER", "GATA3", "ER", "GATA3", "AP1")
plotAnnoScoreDist(rsScores, colToPlot="PC1", pattern="ER", alpha=1)
plotAnnoScoreDist(rsScores, colToPlot="PC2", pattern=c("ER", "GATA3"))
```

`regionQuantileByTargetVar` *Visualize how individual regions are associated with target variable*

Description

Visualize how much each region in a region set is associated with each target variable. For each target variable ('signalCol'), the average (absolute) signal value is calculated for each region in the region set. Then for a given target variable, the average signal is converted to a percentile/quantile based on the distribution of all signal values for that target variable. These values are plotted in a heatmap.

Usage

```
regionQuantileByTargetVar(
  signal,
  signalCoord,
  regionSet,
  rsName = "",
  signalCol = paste0("PC", 1:5),
  maxRegionsToPlot = 8000,
  cluster_rows = TRUE,
  row_title = "Region",
  column_title = rsName,
  column_title_side = "top",
  cluster_columns = FALSE,
  name = "Percentile of Loading Scores in PC",
  col = c("skyblue", "yellow"),
  absVal = TRUE,
  ...
)
```

Arguments

`signal`	Matrix of feature contribution scores (the contribution of each epigenetic feature to each target variable). One named column for each target variable. One row for each original epigenetic feature (should be same order as original data/signalCoord). For (an unsupervised) example, if PCA was done on epigenetic data and the goal was to find region sets associated with the principal components, you could use the x$rotation output of prcomp(epigenetic data) as the feature contribution scores/'signal' parameter.
`signalCoord`	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).

`regionSet` A genomic ranges (GRanges) object with regions corresponding to the same biological annotation. Must be from the same reference genome as the coordinates for the actual data/samples (signalCoord). The regions that will be visualized.

`rsName` Character. Name of the region set. For use as a title for the heatmap.

`signalCol` A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes).

`maxRegionsToPlot` How many top regions from region set to include in heatmap. Including too many may slow down computation and increase memory use. If regionSet has more regions than maxRegionsToPlot, a number of regions equal to maxRegionsToPlot will be randomly sampled from the region set and these regions will be plotted. Clustering rows is a major limiting factor on how long it takes to plot the regions so if you want to plot many regions, you can also set cluster_rows to FALSE.

`cluster_rows` Logical object, whether to cluster rows or not (may increase computation time significantly for large number of rows)

`row_title` Character object, row title

`column_title` Character object, column title

`column_title_side` Character object, where to put the column title: "top" or "bottom"

`cluster_columns` Logical object, whether to cluster columns. It is recommended to keep this as FALSE so it will be easier to compare target variables that have a certain order such as PCs (with cluster_columns = FALSE, they will be in the same specified order in different heatmaps)

`name` Character object, legend title

`col` A vector of colors or a color mapping function which will be passed to the ComplexHeatmap::Heatmap() function. See ?Heatmap (the "col" parameter) for more details.

`absVal` Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

`...` Optional parameters for ComplexHeatmap::Heatmap()

Value

A heatmap. Columns are signalCol's, rows are regions. This heatmap allows you to see if some regions are associated with certain target variables but not others. Also, you can see if a subset of regions in the region set are associated with target variables while another subset are not associated with any target variables To color each region, first the (absolute) signal values within that region are averaged. Then this average is compared to the distribution of all (absolute) individual signal values for the given target variable to get a quantile/percentile for that region. Colors are based on this quantile/percentile. The output is a Heatmap object (ComplexHeatmap package).

Examples

```
data("brcaATACCoord1")
data("brcaATACData1")
data("esr1_chr1")
featureContributionScores <- prcomp(t(brcaATACData1))$rotation
regionByPCHM <- regionQuantileByTargetVar(signal = featureContributionScores,
                                  signalCoord = brcaATACCoord1,
                                  regionSet = esr1_chr1,
                                  rsName = "Estrogen Receptor Chr1",
                                  signalCol=paste0("PC", 1:2),
                                  maxRegionsToPlot = 8000,
                                  cluster_rows = TRUE,
                                  cluster_columns = FALSE,
                                  column_title = rsName,
                                  name = "Percentile of Loading Scores in PC")
```

`rsRankingIndex` *Get indices for top scored region sets*

Description

For each target variable, get index of original region sets but ordered by rsScores ranking for each target variable. The original index refers to that region set's position in the 'GRList' param given to 'aggregateSignalGRList' which is also that region set's row index in the COCOA output. The first number in a given column of this function's output will be the original index of the region set ranked first for that target variable. The second row for a column will be the original index of the region set that ranked second for that target variable, etc. You can use this function to make it easier when you want to select the top region sets for further analysis or just for sorting the results. Region set scores are sorted in decreasing or increasing order according to the 'decreasing' parameter.

Usage

```
rsRankingIndex(rsScores, signalCol, decreasing = TRUE, newColName = signalCol)
```

Arguments

`rsScores`	data.frame. A data.frame with region set scores. The output of the 'aggregateSignalGRList' function. Each row is a region set. One column for each sample variable of interest (e.g. PC or sample phenotype). Also can have columns with info on the overlap between the region set and the epigenetic data. Rows should be in the same order as the region sets in GRList (the list of region sets used to create rsScores.)
`signalCol`	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). The columns in rsScores for which you want the indices of the original region sets.

decreasing	Logical. Whether to sort rsScores in decreasing or increasing order.
newColName	Character. The names of the columns of the output data.frame. The order should correspond to the order of the input columns given by signalCol.

Value

A data.frame with one column for each ‘signalCol‘. Column names are given by ‘signalCol‘ or ‘newColName‘ (if used). Each column has been sorted by score for region sets for that target variable (order given by ‘decreasing‘ param). Original indices for region sets that were used to create rsScores are given. Region sets with a score of NA are counted as having the lowest scores and indices for these region sets will be at the bottom of the returned data.frame (na.last=TRUE in sorting)

Examples

```
data("rsScores")
rsRankInd = rsRankingIndex(rsScores=rsScores,
                           signalCol=c("PC1", "PC2"))
# region sets sorted by score for PC1
rsScores[rsRankInd$PC1, ]
# region sets sorted by score for PC2
rsScores[rsRankInd$PC2, ]
```

---

rsScoreHeatmap *Heatmap of region set scores*

Description

Heatmap of the ranking of region set scores across target variables. A visualization of the rank of region sets in each target variable, allowing the user to see if a region set is ranked highly for all target variables or only a subset. Region sets will be ranked from highest scoring to lowest based on their score for ‘orderByCol‘. The ComplexHeatmap package is used and additional parameters for the ComplexHeatmap::Heatmap function may be passed to this function to modify the heatmap.

Usage

```
rsScoreHeatmap(
  rsScores,
  signalCol = paste0("PC", 1:5),
  orderByCol = "PC1",
  rsNameCol = "rsName",
  topX = 20,
  col = c("red", "#EEEEEE", "blue"),
  row_title = "Region Set",
  column_title = "Principal Component",
  column_title_side = "bottom",
  cluster_rows = FALSE,
```

```
  cluster_columns = FALSE,
  show_row_names = TRUE,
  row_names_max_width = unit(10000, "mm"),
  name = "Rank within PC",
  ...
)
```

Arguments

rsScores	data.frame. A data.frame with region set scores. The output of the 'aggregateSignalGRList' function. Each row is a region set. One column for each sample variable of interest (e.g. PC or sample phenotype). Also can have columns with info on the overlap between the region set and the epigenetic data. Rows should be in the same order as the region sets in GRList (the list of region sets used to create rsScores.)
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). Must be column names of rsScores.
orderByCol	A character object. Target variable to order by in heatmap (arranged in decreasing order for scores so p values should be -log transformed). Must be the name of a column in rsScores.
rsNameCol	Character. Name of the column in rsScores that has the names/identifiers for the region sets so these can be included in the plot as row names.
topX	Number of top region sets to include in the heatmap
col	A vector of colors or a color mapping function which will be passed to the ComplexHeatmap::Heatmap() function. See ?Heatmap (the "col" parameter) for more details. "#EEEEEE" is the code for a color similar to white.
row_title	Character object, row title
column_title	Character object, column title
column_title_side	Character object, where to put the column title: "top" or "bottom"
cluster_rows	Logical object, whether rows should be clustered. This should be kept as FALSE to keep the correct ranking of region sets.
cluster_columns	Logical object, whether to cluster columns. It is recommended to keep this as FALSE so it will be easier to compare target variables that are ordered (such as principal components). With cluster_columns = FALSE, they will be in the same specified order in different heatmaps.
show_row_names	Logical object, display row names (ie region set names)
row_names_max_width	"unit" object. The amount of room to allocate for row names. See ?grid::unit for object type.
name	Character object, legend title
...	Optional parameters for ComplexHeatmap::Heatmap().

Value

A heatmap of region set scores across. Each row is a region set, each column is a target variable. The color corresponds to the relative rank of a region set's score for a given target variable out of all tested region sets.

Examples

```
data("rsScores")
scoreHeatmap <- rsScoreHeatmap(rsScores,
          signalCol=paste0("PC", 1:2), orderByCol = "PC2")
```

---

rsScores *Example COCOA Results (made up)*

Description

A data.frame with example COCOA results. 5 region sets with names given by rsScores$rsName. Each region set has a score for each PC. Scores for real region sets would normally be orders of magnitude smaller.

Usage

```
data(rsScores)
```

Format

A data.frame object

---

runCOCOA *Run COCOA: quantify inter-sample variation, score region sets*

Description

This is a convenience function that does the two steps of COCOA: quantifying the epigenetic variation and scoring the region sets. This function will return the real COCOA scores if using the default 'sampleOrder' parameter values. This function also makes it easy to generate null distributions in order to evaluate the statistical significance of the real COCOA results. You can use the sampleOrder parameter to shuffle the samples, then run COCOA to get fake scores for each region set. By doing this many times, you can build a null distribution for each region set composed of the region set's random scores from each permutation. There are multiple options for quantifying the epigenetic variation, specified by the 'variationMetric' parameter. Quantifying the variation for the real/non-permuted COCOA scores should be done with the same variation metric as is used for the random permutations. For an unsupervised analysis using dimensionality reduction, first, the dimensionality reduction is done outside 'runCOCOA', then the latent factors/principal components are input to 'runCOCOA' as the sample labels (targetVar parameter) when calculating both the real and also the permutated region set scores. For a supervised analysis, the target variables/phenotypes are the targetVar. See the vignettes for examples.

Usage

```
runCOCOA(
  genomicSignal,
  signalCoord,
  GRList,
  signalCol,
  targetVar,
  sampleOrder = 1:nrow(targetVar),
  variationMetric = "cor",
  scoringMetric = "default",
  verbose = TRUE,
  absVal = TRUE,
  olList = NULL,
  pOlapList = NULL,
  centerGenomicSignal = TRUE,
  centerTargetVar = TRUE,
  returnCovInfo = TRUE
)
```

Arguments

genomicSignal	Matrix/data.frame. The genomic signal (e.g. DNA methylation levels) Columns of genomicSignal should be samples/patients. Rows should be individual signal/features (each row corresponds to one genomic coordinate/range)
signalCoord	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
GRList	GRangesList object. Each list item is a distinct region set to test (region set: regions that correspond to the same biological annotation). The region set database must be from the same reference genome as the coordinates for the actual data/samples (signalCoord).
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). The columns in 'sampleLabels' for which to calculate the variation related to the epigenetic data (e.g. correlation) and then to run COCOA on.
targetVar	Matrix or data.frame. Rows should be samples. Columns should be the target variables (whatever variable you want to test for association with the epigenetic signal: e.g. PC scores),
sampleOrder	numeric. A vector of length (number of samples). If sampleOrder is 1:(number of samples) then this function will return the real COCOA scores. To generate random COCOA scores in order to make null distributions, shuffle the samples in a random order. E.g. sampleOrder = sample(1:ncol(genomicSignal), ncol(genomicSignal)) where ncol(genomicSignal) is the number of samples. Set the seed with set.seed() before making sampleOrder to ensure reproducibility.

variationMetric
: Character. The metric to use to quantify the association between each feature in genomicSignal and each target variable in sampleLabels. Either "cor" (Pearson correlation), "cov" (covariation), or "spearmanCor" (Spearman correlation).

scoringMetric
: A character object with the scoring metric. There are different methods available for signalCoordType="singleBase" vs signalCoordType="multiBase". For "singleBase", the available methods are "regionMean", "regionMedian", "simpleMean", and "simpleMedian". The default method is "regionMean". For "multiBase", the methods are "proportionWeightedMean", "simpleMean", and "simpleMedian". The default is "proportionWeightedMean". "regionMean" is a weighted average of the signal, weighted by region (absolute value of signal if absVal=TRUE). First the signal is averaged within each regionSet region, then all the regions are averaged. With "regionMean" method, be cautious in interpretation for region sets with low number of regions that overlap signalCoord. The "regionMedian" method is the same as "regionMean" but the median is taken at each step instead of the mean. The "simpleMean" method is just the unweighted average of all (absolute) signal values that overlap the given region set. For multiBase data, this includes signal regions that overlap a regionSet region at all (1 base overlap or more) and the signal for each overlapping region is given the same weight for the average regardless of how much it overlaps. The "simpleMedian" method is the same as "simpleMean" but takes the median instead of the mean. "proportionWeightedMean" is a weighted average of all signalCoord regions that overlap with regionSet regions. For each signalCoord region that overlaps with a regionSet region, we calculate what proportion of the regionSet region is covered. Then this proportion is used to weight the signal value when calculating the mean. The denominator of the mean is the sum of all the proportion overlaps.

verbose
: A "logical" object. Whether progress of the function should be shown. One bar indicates the region set is completed.

absVal
: Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

olList
: list. Each list item should be a "SortedByQueryHits" object (output of findOverlaps function). Each hits object should have the overlap information between signalCoord and one item of GRList (one unique region set). The region sets from GRList must be the "subject" in findOverlaps and signalCoord must be the "query". E.g. findOverlaps(subject=regionSet, query=signalCoord). Providing this information can greatly improve permutation speed since the overlaps will not have to be calculated for each permutation. The "runCOCOAPerm" function calculates this information only once, internally, so this does not have to be provided when using that function. When using this parameter, signalCoord, genomicSignal, and each region set must be in the same order as they were when olList was created. Otherwise, the wrong genomic loci will be referenced (e.g. if epigenetic features were filtered out of genomicSignal after olList was created.)

pOlapList list. This parameter is only used if the scoring metric is "proportionWeightedMean" and olList is also provided as an argument. Each item of the list should be a vector that contains the proportion overlap between signalCoord and regions from one region set (one item of GRList). Specifically, each value should be the proportion of the region set region that is overlapped by a signalCoord region. The proportion overlap values should be in the same order as the overlaps given by olList for the corresponding region set.

centerGenomicSignal
Logical. Should rows in genomicSignal be centered based on their means? (subtracting row mean from each row)

centerTargetVar
Logical. Should columns in targetVar be centered based on their means? (subtract column mean from each column)

returnCovInfo logical. If TRUE, the following coverage and region set info will be calculated and included in function output: regionSetCoverage, signalCoverage, totalRegionNumber, and meanRegionSize. For the proportionWeightedMean scoring method, sumProportionOverlap will also be calculated.

Value

data.frame. The output of aggregateSignalGRList for one permutation.

Examples

```
data("esr1_chr1")
data("nrf1_chr1")
data("brcaMethylData1")
data("brcaMCoord1")
pcScores <- prcomp(t(brcaMethylData1))$x
targetVarCols <- c("PC1", "PC2")
targetVar <- pcScores[, targetVarCols]

# give the actual order of samples to `runCOCOA` to get the real scores
correctSampleOrder=1:nrow(targetVar)
realRSScores <- runCOCOA(genomicSignal=brcaMethylData1,
                         signalCoord=brcaMCoord1,
                         GRList=GRangesList(esr1_chr1, nrf1_chr1),
                         signalCol=c("PC1", "PC2"),
                         targetVar=targetVar,
                         sampleOrder=correctSampleOrder,
                         variationMetric="cor")
realRSScores

# give random order of samples to get random COCOA scores
# so you start building a null distribution for each region set
# (see vignette for example of building a null distribution with `runCOCOA`)
randomOrder <- sample(1:nrow(targetVar),
                      size=nrow(targetVar),
                      replace=FALSE)
randomRSScores <- runCOCOA(genomicSignal=brcaMethylData1,
                           signalCoord=brcaMCoord1,
```

```
                          GRList=GRangesList(esr1_chr1, nrf1_chr1),
                          signalCol=c("PC1", "PC2"),
                          targetVar=targetVar,
                          sampleOrder=randomOrder,
                          variationMetric="cor")
randomRSScores
```

---

runCOCOAPerm *Run COCOA permutations to get p-values*

Description

This is a convenience function that runs multiple steps of the permutation process together: it runs COCOA permutations, converts these to null distributions, gets the empirical p value (which is limited by the number of permutations), gets z scores, and fits a gamma distribution to each null distribution to estimate p values (not limited by the number of permutations), Requires that the user has previously calculated the real COCOA scores. See these individual functions for more info on each step: runCOCOA, convertToFromNullDist, getPermStat, and getGammaPVal.

Usage

```
runCOCOAPerm(
  genomicSignal,
  signalCoord,
  GRList,
  rsScores,
  targetVar,
  signalCol = c("PC1", "PC2"),
  scoringMetric = "default",
  absVal = TRUE,
  olList = NULL,
  centerGenomicSignal = TRUE,
  centerTargetVar = TRUE,
  variationMetric = "cor",
  nPerm = 300,
  useSimpleCache = TRUE,
  cacheDir = getwd(),
  dataID = "",
  testType = "greater",
  gammaFitMethod = "mme",
  realScoreInDist = TRUE,
  force = FALSE,
  verbose = TRUE,
  returnCovInfo = FALSE,
  ...
)
```

Arguments

genomicSignal	Matrix/data.frame. The genomic signal (e.g. DNA methylation levels) Columns of genomicSignal should be samples/patients. Rows should be individual signal/features (each row corresponds to one genomic coordinate/range)
signalCoord	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
GRList	GRangesList object. Each list item is a distinct region set to test (region set: regions that correspond to the same biological annotation). The region set database must be from the same reference genome as the coordinates for the actual data/samples (signalCoord).
rsScores	data.frame. A data.frame with region set scores. The output of the 'aggregateSignalGRList' function. Each row is a region set. One column for each sample variable of interest (e.g. PC or sample phenotype). Also can have columns with info on the overlap between the region set and the epigenetic data. Rows should be in the same order as the region sets in GRList (the list of region sets used to create rsScores.)
targetVar	Matrix or data.frame. Rows should be samples. Columns should be the target variables (whatever variable you want to test for association with the epigenetic signal: e.g. PC scores),
signalCol	A character vector with the names of the sample variables of interest/target variables (e.g. PCs or sample phenotypes). The columns in 'sampleLabels' for which to calculate the variation related to the epigenetic data (e.g. correlation) and then to run COCOA on.
scoringMetric	A character object with the scoring metric. There are different methods available for signalCoordType="singleBase" vs signalCoordType="multiBase". For "singleBase", the available methods are "regionMean", "regionMedian", "simpleMean", and "simpleMedian". The default method is "regionMean". For "multiBase", the methods are "proportionWeightedMean", "simpleMean", and "simpleMedian". The default is "proportionWeightedMean". "regionMean" is a weighted average of the signal, weighted by region (absolute value of signal if absVal=TRUE). First the signal is averaged within each regionSet region, then all the regions are averaged. With "regionMean" method, be cautious in interpretation for region sets with low number of regions that overlap signalCoord. The "regionMedian" method is the same as "regionMean" but the median is taken at each step instead of the mean. The "simpleMean" method is just the unweighted average of all (absolute) signal values that overlap the given region set. For multiBase data, this includes signal regions that overlap a regionSet region at all (1 base overlap or more) and the signal for each overlapping region is given the same weight for the average regardless of how much it overlaps. The "simpleMedian" method is the same as "simpleMean" but takes the median instead of the mean. "proportionWeightedMean" is a weighted average of all signalCoord regions that overlap with regionSet regions. For each signalCoord region that overlaps with a regionSet region, we calculate what proportion of the

regionSet region is covered. Then this proportion is used to weight the signal value when calculating the mean. The denominator of the mean is the sum of all the proportion overlaps.

`absVal` Logical. If TRUE, take the absolute value of values in signal. Choose TRUE if you think there may be some genomic loci in a region set that will increase and others will decrease (if there may be anticorrelation between regions in a region set). Choose FALSE if you expect regions in a given region set to all change in the same direction (all be positively correlated with each other).

`olList` list. Each list item should be a "SortedByQueryHits" object (output of findOverlaps function). Each hits object should have the overlap information between signalCoord and one item of GRList (one unique region set). The region sets from GRList must be the "subject" in findOverlaps and signalCoord must be the "query". E.g. findOverlaps(subject=regionSet, query=signalCoord). Providing this information can greatly improve permutation speed since the overlaps will not have to be calculated for each permutation. The "runCOCOAPerm" function calculates this information only once, internally, so this does not have to be provided when using that function. When using this parameter, signalCoord, genomicSignal, and each region set must be in the same order as they were when olList was created. Otherwise, the wrong genomic loci will be referenced (e.g. if epigenetic features were filtered out of genomicSignal after olList was created.)

`centerGenomicSignal`
Logical. Should rows in genomicSignal be centered based on their means? (subtracting row mean from each row)

`centerTargetVar`
Logical. Should columns in targetVar be centered based on their means? (subtract column mean from each column)

`variationMetric`
Character. The metric to use to quantify the association between each feature in genomicSignal and each target variable in sampleLabels. Either "cor" (Pearson correlation), "cov" (covariation), or "spearmanCor" (Spearman correlation).

`nPerm` Numeric. The number of permutations to do.

`useSimpleCache` Logical. Whether to use save caches. Caches will be created for each permutation so that if the function is disrupted it can restart where it left off. The final results are also saved as a cache. See simpleCache package for more details.

`cacheDir` Character. The path for the directory in which the caches should be saved.

`dataID` Character. A unique identifier for this dataset (for saving results with simpleCache)

`testType` Character. Parameter for ‘getPermStat‘. Whether to create p values based on one a two sided test or a lesser/greater one sided test. Options are: "greater", "lesser", "two-sided"

`gammaFitMethod` Character. method to use for fitting the gamma distribution to null distribution. Options are "mme" (moment matching estimation), "mle" (maximum likelihood estimation), "qme" (quantile matching estimation), and "mge" (maximum goodness-of-fit estimation). See ?COCOA::getGammaPVal and ?fitdistrplus::fitdist() for more info.

realScoreInDist
: Logical. Should the actual score (from test with no permutations) be included in the null distribution when fitting the gamma distribution. realScoreInDist=TRUE is recommended.

force
: Logical. If force=TRUE, when fitting the gamma distribution returns an error (as may happen when a method other than "mme" is used) then allow the error. If force=FALSE, when fitting the gamma distribution returns an error then don't return an error but instead use the "mme" method for fitting that specific gamma distribution.

verbose
: A "logical" object. Whether progress of the function should be shown. One bar indicates the region set is completed.

returnCovInfo
: logical. If TRUE, the following coverage and region set info will be calculated and included in function output: regionSetCoverage, signalCoverage, totalRegionNumber, and meanRegionSize. For the proportionWeightedMean scoring method, sumProportionOverlap will also be calculated.

...
: Character. Optional additional arguments for simpleCache.

Details

For reproducibility, set seed with 'set.seed()' function before running.

Value

Returns a list with the following 4 items: 1. a list of length nPerm where each item is a data.frame of the COCOA scores from a single permutation. Each data.frame is the output of 'runCOCOA()' 2. a data.table/data.frame of empirical p-values (the output of 'getPermStat') 3. a data.table/data.frame of z-scores (the output of 'getPermStat'. 4. a data.frame of p-values based on the gamma approximation (the output of getGammaPVal().

Examples

```
data("esr1_chr1")
data("nrf1_chr1")
data("brcaMethylData1")
data("brcaMCoord1")
pcScores <- prcomp(t(brcaMethylData1))$x
targetVarCols <- c("PC1", "PC2")
targetVar <- pcScores[, targetVarCols]

# give the actual order of samples to `runCOCOA` to get the real scores
correctSampleOrder=1:nrow(targetVar)
realRSScores <- runCOCOA(genomicSignal=brcaMethylData1,
                         signalCoord=brcaMCoord1,
                         GRList=GRangesList(esr1_chr1, nrf1_chr1),
                         signalCol=c("PC1", "PC2"),
                         targetVar=targetVar,
                         sampleOrder=correctSampleOrder,
                         variationMetric="cor")

# give random order of samples to get random COCOA scores
```

```
# so you start building a null distribution for each region set
# (see vignette for example of building a null distribution with `runCOCOA`)
randomOrder <- sample(1:nrow(targetVar),
                      size=nrow(targetVar),
                      replace=FALSE)
randomRSScores <- runCOCOA(genomicSignal=brcaMethylData1,
                           signalCoord=brcaMCoord1,
                           GRList=GRangesList(esr1_chr1, nrf1_chr1),
                           signalCol=c("PC1", "PC2"),
                           targetVar=targetVar,
                           sampleOrder=randomOrder,
                           variationMetric="cor")

# runCOCOAPerm
permResults <- runCOCOAPerm(genomicSignal=brcaMethylData1,
                            signalCoord=brcaMCoord1,
                            GRList=GRangesList(esr1_chr1, nrf1_chr1),
                            rsScores=realRSScores,
                            targetVar=targetVar,
                            signalCol=c("PC1", "PC2"),
                            variationMetric="cor",
                            nPerm = 10,
                            useSimpleCache=FALSE)
permResults
```

`signalAlongAxis` *Visualize how genomic signal in a region set changes along a given axis*

Description

Look at genomic signal (e.g., DNA methylation values) in regions of interest across samples, with samples ordered according to a variable of interest (e.g. PC score). The ComplexHeatmap package is used and additional parameters for the ComplexHeatmap::Heatmap function may be passed to this function to modify the heatmap.

Usage

```
signalAlongAxis(
  genomicSignal,
  signalCoord,
  regionSet,
  sampleScores,
  orderByCol = "PC1",
  topXVariables = NULL,
  variableScores = NULL,
  decreasing = TRUE,
```

```
  cluster_columns = FALSE,
  cluster_rows = FALSE,
  row_title = "Sample",
  column_title = "Genomic Signal",
  column_title_side = "bottom",
  name = "Genomic Signal Value",
  col = c("blue", "#EEEEEE", "red"),
  ...
)
```

Arguments

genomicSignal	Matrix/data.frame. The genomic signal (e.g. DNA methylation levels) Columns of genomicSignal should be samples/patients. Rows should be individual signal/features (each row corresponds to one genomic coordinate/range) Must have sample names/IDs as column names, These same sample names must be row names of sampleScores.
signalCoord	A GRanges object or data frame with coordinates for the genomic signal/original epigenetic data. Coordinates should be in the same order as the original data and the feature contribution scores (each item/row in signalCoord corresponds to a row in signal). If a data.frame, must have chr and start columns (optionally can have end column, depending on the epigenetic data type).
regionSet	A genomic ranges (GRanges) object with regions corresponding to the same biological annotation. Must be from the same reference genome as the coordinates for the actual data/samples (signalCoord). The regions that will be visualized.
sampleScores	A matrix. Must contain a column for the variable of interest/target variable. E.g. The variable of interest could be the principal component scores for the samples. 'sampleScores' must have sample names/IDs as row names, These same sample names must be column names of genomicSignal.
orderByCol	A character object. A variable to order samples by (order rows of heatmap by variable, from high to low value). Must be the name of a column in sampleScores. For instance, if doing unsupervised COCOA with PCA, orderByCol might be the name of one of the PCs (e.g. "PC1"). If doing supervised COCOA, orderByCol might be the name of the target variable of the supervised analysis.
topXVariables	Numeric. The number of variables from genomicSignal to plot. The variables with the highest scores according to variableScores will be plotted. Can help to reduce the size of the plot.
variableScores	Numeric. A vector that has a numeric score for each variable in genomicSignal (length(variableScores) should equal nrow(genomicSignal)). Only used if topXVariables is given. The highest 'topXVariables' will be plotted.
decreasing	Logical. Whether samples should be sorted in decreasing order of 'orderByCol' or not (FALSE is increasing order).
cluster_columns	Logical. Whether to cluster columns (the genomic signal, e.g. DNA methylation values for each CpG).

cluster_rows	Logical. Whether rows should be clustered. This should be kept as FALSE to keep the correct ranking of samples/observations according to their target variable score.
row_title	Character object, row title
column_title	Character object, column title
column_title_side	Character object, where to put the column title: "top" or "bottom"
name	Character object, legend title
col	A vector of colors or a color mapping function which will be passed to the ComplexHeatmap::Heatmap() function. See ?Heatmap (the "col" parameter) for more details. "#EEEEEE" is the code for a color similar to white.
...	Optional parameters for ComplexHeatmap::Heatmap()

Value

A heatmap of genomic signal values (eg DNA methylation levels) in regions of interest (regionSet), with rows ordered by the column of sampleScores given with 'orderByCol'. Each row is a patient/sample and each column is an individual genomic signal value.

Examples

```
data("brcaMethylData1")
data("brcaMCoord1")
data("esr1_chr1")
data("brcaPCScores")
signalHM <- signalAlongAxis(genomicSignal=brcaMethylData1,
                            signalCoord=brcaMCoord1,
                            regionSet=esr1_chr1,
                            sampleScores=brcaPCScores,
                            orderByCol="PC1", cluster_columns=TRUE)
```

Index

∗ **datasets**
  atf3_chr1, 8
  brcaATACCoord1, 9
  brcaATACData1, 9
  brcaMCoord1, 10
  brcaMetadata, 10
  brcaMethylData1, 11
  brcaPCScores, 11
  brcaPCScores657, 12
  esr1_chr1, 14
  gata3_chr1, 14
  nrf1_chr1, 21
  rsScores, 28

aggregateSignal, 3
aggregateSignalGRList, 5
atf3_chr1, 8

brcaATACCoord1, 9
brcaATACData1, 9
brcaMCoord1, 10
brcaMetadata, 10
brcaMethylData1, 11
brcaPCScores, 11
brcaPCScores657, 12

COCOA, 12
convertToFromNullDist, 13

esr1_chr1, 14

gata3_chr1, 14
getGammaPVal, 15
getMetaRegionProfile, 16
getPermStat, 18
getTopRegions, 20

nrf1_chr1, 21

plotAnnoScoreDist, 21

regionQuantileByTargetVar, 23
rsRankingIndex, 25
rsScoreHeatmap, 26
rsScores, 28
runCOCOA, 28
runCOCOAPerm, 32

signalAlongAxis, 36