Package 'EMDomics'

May 4, 2026

Title Earth Mover's Distance for Differential Analysis of Genomics Data

Description The EMDomics algorithm is used to perform a supervised multi-class analysis to measure the magnitude and statistical significance of observed continuous genomics data between groups. Usually the data will be gene expression values from array-based or sequence-based experiments, but data from other types of experiments can also be analyzed (e.g. copy number variation). Traditional methods like Significance Analysis of Microarrays (SAM) and Linear Models for Microarray Data (LIMMA) use significance tests based on summary statistics (mean and standard deviation) of the distributions. This approach lacks power to identify expression differences between groups that show high levels of intra-group heterogeneity. The Earth Mover's Distance (EMD) algorithm instead computes the ``work" needed to transform one distribution into another, thus providing a metric of the overall difference in shape between two distributions. Permutation of sample labels is used to generate q-values for the observed EMD scores. This package also incorporates the Komolgorov-Smirnov (K-S) test and the Cramer von Mises test (CVM), which are both common distribution comparison tests.

Version 2.43.0

biocViews Software, DifferentialExpression, GeneExpression, Microarray

Maintainer Sadhika Malladi <contact@sadhikamalladi.com> and Daniel Schmolze <emd@schmolze.com>

Depends R (>= 3.2.1)

Imports emdist, BiocParallel, matrixStats, ggplot2, CDFt, preprocessCore

Suggests knitr

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Sadhika Malladi [aut, cre], Daniel Schmolze [aut, cre], Andrew Beck [aut], Sheida Nabavi [aut]

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

git_branch devel

git_last_commit 5b37735

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-04

Contents

`emdomics-package` *Earth Mover's Distance algorithm for differential analysis of genomics data.*

Description

`calculate_emd`, `calculate_cvm`, or `calculate_ks` will usually be the only functions needed, depending on the type of distribution comparison test that is desired.

`calculate_cvm` *Cramer von Mises for differential analysis of genomics data*

Description

This is a main user interface to the **EMDomics** package, and will usually the only function needed when conducting an analysis using the CVM algorithm. Analyses can also be conducted with the Komolgorov-Smirnov Test using `calculate_ks` or the Earth Mover's Distance algorithm using `calculate_emd`.

The algorithm is used to compare genomics data between any number of groups. Usually the data will be gene expression values from array-based or sequence-based experiments, but data from other types of experiments can also be analyzed (e.g. copy number variation).

Traditional methods like Significance Analysis of Microarrays (SAM) and Linear Models for Microarray Data (LIMMA) use significance tests based on summary statistics (mean and standard deviation) of the two distributions. This approach tends to give non-significant results if the two distributions are highly heterogeneous, which can be the case in many biological circumstances (e.g sensitive vs. resistant tumor samples).

The Cramer von Mises (CVM) algorithm generates a test statistic that is the sum of the squared values of the differences between two cumulative distribution functions (CDFs). As a result, the test statistic tends to overestimate the similarity between two distributions and cannot effectively handle partial matching like EMD does. However, it is one of the most commonly referenced nonparametric two-class distribution comparison tests in non-genomic contexts.

The CVM-based algorithm implemented in **EMDomics** has two main steps. First, a matrix (e.g. of expression data) is divided into data for each of the groups. Every possible pairwise CVM score is then computed and stored in a table. The CVM score for a single gene is calculated by averaging all of the pairwise CVM scores. Next, the labels for each of the groups are randomly permuted a specified number of times, and an CVM score for each permutation is calculated. The median of the permuted scores for each gene is used as the null distribution, and the False Discovery Rate (FDR) is computed for a range of permissive to restrictive significance thresholds. The threshold that minimizes the FDR is defined as the q-value, and is used to interpret the significance of the CVM score analogously to a p-value (e.g. q-value < 0.05 is significant.)

Usage

```
calculate_cvm(data, outcomes, nperm = 100, pairwise.p = FALSE,
  seq = FALSE, quantile.norm = FALSE, verbose = TRUE, parallel = TRUE)
```

Arguments

`data`	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
`outcomes`	A vector containing group labels for each of the samples provided in the `data` matrix. The names should be the sample identifiers provided in `data`.
`nperm`	An integer specifying the number of randomly permuted CVM scores to be computed. Defaults to 100.
`pairwise.p`	Boolean specifying whether the permutation-based q-values should be computed for each pairwise comparison. Defaults to `FALSE`.

seq	Boolean specifying if the given data is RNA Sequencing data and ought to be normalized. Set to TRUE, if passing transcripts per million (TPM) data or raw data that is not scaled. If TRUE, data will be normalized by first multiplying by 1E6, then adding 1, then taking the log base 2. If FALSE, the data will be handled as is (unless quantile.norm is TRUE). Note that as a distribution comparison function, K-S will compute faster with scaled data. Defaults to FALSE.
quantile.norm	Boolean specifying is data should be normalized by quantiles. If TRUE, then the normalize.quantiles function is used. Defaults to FALSE.
verbose	Boolean specifying whether to display progress messages.
parallel	Boolean specifying whether to use parallel processing via the **BiocParallel** package. Defaults to TRUE.

Value

The function returns an CVMomics object.

See Also

CVMomics CramerVonMisesTwoSamples

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_cvm(dat, outcomes, nperm=10, parallel=FALSE)
head(results$cvm)
```

---

calculate_cvm_gene *Calculate CVM score for a single gene*

Description

Calculate CVM score for a single gene

Usage

```
calculate_cvm_gene(vec, outcomes, sample_names)
```

Arguments

vec	A named vector containing data (e.g. expression data) for a single gene.
outcomes	A vector of group labels for the samples. The names must correspond to the names of vec.
sample_names	A character vector with the names of the samples in vec.

Details

All possible combinations of the classes are used as pairwise comparisons. The data in vec is divided based on class labels based on the outcomes identifiers given. For each pairwise computation, the hist function is used to generate histograms for the two groups. The densities are then retrieved and passed to CramerVonMisesTwoSamples to compute the pairwise CVM score. The total CVM score for the given data is the average of the pairwise CVM scores.

Value

The cvm score is returned.

See Also

CramerVonMisesTwoSamples

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

calculate_cvm_gene(dat[1,], outcomes, colnames(dat))
```

---

calculate_emd *Earth Mover's Distance for differential analysis of genomics data*

Description

This is the main user interface to the **EMDomics** package, and will usually the only function needed when conducting an analysis using the EMD algorithm. Analyses can also be conducted with the Komolgorov-Smirnov Test using calculate_ks or the Cramer Von Mises algorithm using calculate_cvm.

The algorithm is used to compare genomics data between any number of groups. Usually the data will be gene expression values from array-based or sequence-based experiments, but data from other types of experiments can also be analyzed (e.g. copy number variation).

Traditional methods like Significance Analysis of Microarrays (SAM) and Linear Models for Microarray Data (LIMMA) use significance tests based on summary statistics (mean and standard deviation) of the two distributions. This approach tends to give non-significant results if the two distributions are highly heterogeneous, which can be the case in many biological circumstances (e.g sensitive vs. resistant tumor samples).

The Earth Mover's Distance algorithm instead computes the "work" needed to transform one distribution into another, thus capturing possibly valuable information relating to the overall difference in shape between two heterogeneous distributions.

The EMD-based algorithm implemented in **EMDomics** has two main steps. First, a matrix (e.g. of expression data) is divided into data for each of the groups. Every possible pairwise EMD score is

then computed and stored in a table. The EMD score for a single gene is calculated by averaging all of the pairwise EMD scores. Next, the labels for each of the groups are randomly permuted a specified number of times, and an EMD score for each permutation is calculated. The median of the permuted scores for each gene is used as the null distribution, and the False Discovery Rate (FDR) is computed for a range of permissive to restrictive significance thresholds. The threshold that minimizes the FDR is defined as the q-value, and is used to interpret the significance of the EMD score analogously to a p-value (e.g. q-value < 0.05 is significant.)

Because EMD is based on a histogram binning of the expression levels, data that cannot be binned will be discarded, and a message for that gene will be printed. The most likely reason for histogram binning failing is due to uniform values (e.g. all 0s).

Usage

```
calculate_emd(data, outcomes, binSize = 0.2, nperm = 100,
  pairwise.p = FALSE, seq = FALSE, quantile.norm = FALSE,
  verbose = TRUE, parallel = TRUE)
```

Arguments

`data`	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
`outcomes`	A vector containing group labels for each of the samples provided in the `data` matrix. The names should be the sample identifiers provided in `data`.
`binSize`	The bin size to be used when generating histograms of the data for each group. Defaults to 0.2.
`nperm`	An integer specifying the number of randomly permuted EMD scores to be computed. Defaults to 100.
`pairwise.p`	Boolean specifying whether the permutation-based q-values should be computed for each pairwise comparison. Defaults to `FALSE`.
`seq`	Boolean specifying if the given data is RNA Sequencing data and ought to be normalized. Set to `TRUE`, if passing transcripts per million (TPM) data or raw data that is not scaled. If `TRUE`, data will be normalized by first multiplying by 1E6, then adding 1, then taking the log base 2. If `FALSE`, the data will be handled as is (unless `quantile.norm` is `TRUE`). Note that as a distribution comparison function, EMD will compute faster with scaled data. Defaults to `FALSE`.
`quantile.norm`	Boolean specifying is data should be normalized by quantiles. If `TRUE`, then the `normalize.quantiles` function is used. Defaults to `FALSE`.
`verbose`	Boolean specifying whether to display progress messages.
`parallel`	Boolean specifying whether to use parallel processing via the **BiocParallel** package. Defaults to `TRUE`.

Value

The function returns an `EMDomics` object.

See Also

`EMDomics` `emd2d`

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_emd(dat, outcomes, nperm=10, parallel=FALSE)
head(results$emd)
```

---

calculate_emd_gene *Calculate EMD score for a single gene*

Description

Calculate EMD score for a single gene

Usage

```
calculate_emd_gene(vec, outcomes, sample_names, binSize = 0.2)
```

Arguments

vec	A named vector containing data (e.g. expression data) for a single gene. Names ought to correspond to samples.
outcomes	A vector of group labels for the samples. The names must correspond to the names of vec.
sample_names	A character vector with the names of the samples in vec.
binSize	The bin size to be used when generating histograms for each of the groups.

Details

All possible combinations of the classes are used as pairwise comparisons. The data in vec is divided based on class labels based on the outcomes identifiers given. For each pairwise computation, the hist function is used to generate histograms for the two groups. The densities are then retrieved and passed to emd2d to compute the pairwise EMD score. The total EMD score for the given data is the average of the pairwise EMD scores.

Value

The emd score is returned.

See Also

emd2d

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

calculate_emd_gene(dat[1,], outcomes, colnames(dat))
```

`calculate_ks` *Calculate the Komolgorov-Smirnov test statistic and q-values for differential gene expression analysis.*

Description

This is only function needed when conducting an analysis using the Komolgorov-Smirnov algorithm. Analyses can also be conducted with the EMD algorithm using `calculate_emd` or the Cramer Von Mises (CVM) algorithm using `calculate_cvm`.

The algorithm is used to compare genomics data between any number of groups. Usually the data will be gene expression values from array-based or sequence-based experiments, but data from other types of experiments can also be analyzed (e.g. copy number variation).

Traditional methods like Significance Analysis of Microarrays (SAM) and Linear Models for Microarray Data (LIMMA) use significance tests based on summary statistics (mean and standard deviation) of the two distributions. This approach tends to give non-significant results if the two distributions are highly heterogeneous, which can be the case in many biological circumstances (e.g sensitive vs. resistant tumor samples).

Komolgorov-Smirnov instead calculates a test statistic that is the maximum distance between two cumulative distribution functions (CDFs). Unlike the EMD score, the KS test statistic summarizes only the maximum difference (while EMD considers quantity and distance between all differences).

The KS algorithm implemented in **EMDomics** has two main steps. First, a matrix (e.g. of expression data) is divided into data for each of the groups. Every possible pairwise KS score is then computed and stored in a table. The KS score for a single gene is calculated by averaging all of the pairwise KS scores. If the user sets `pairwise.p` to true, then the p-values from the KS test are adjusted using the Benjamini-Hochberg method and stored in a table. Next, the labels for each of the groups are randomly permuted a specified number of times, and an EMD score for each permutation is calculated. The median of the permuted scores for each gene is used as the null distribution, and the False Discovery Rate (FDR) is computed for a range of permissive to restrictive significance thresholds. The threshold that minimizes the FDR is defined as the q-value, and is used to interpret the significance of the EMD score analogously to a p-value (e.g. q-value < 0.05 = significant). The q-values returned by the KS test (and adjusted for multiple significance testing) can be compared to the permuted q-values.

Usage

```
calculate_ks(data, outcomes, nperm = 100, pairwise.p = FALSE, seq = FALSE,
  quantile.norm = FALSE, verbose = TRUE, parallel = TRUE)
```

Arguments

data	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
outcomes	A vector containing group labels for each of the samples provided in the data matrix. The names should be the sample identifiers provided in data.
nperm	An integer specifying the number of randomly permuted EMD scores to be computed. Defaults to 100.
pairwise.p	Boolean specifying whether the user wants the pairwise p-values. Pairwise p-values returned by ks.test are adjusted within pairwise comparison using the Benjamini-Hochberg (BH) method. Defaults to FALSE.
seq	Boolean specifying if the given data is RNA Sequencing data and ought to be normalized. Set to TRUE, if passing transcripts per million (TPM) data or raw data that is not scaled. If TRUE, data will be normalized by first multiplying by 1E6, then adding 1, then taking the log base 2. If FALSE, the data will be handled as is (unless quantile.norm is TRUE). Note that as a distribution comparison function, K-S will compute faster with scaled data. Defaults to FALSE.
quantile.norm	Boolean specifying is data should be normalized by quantiles. If TRUE, then the normalize.quantiles function is used. Defaults to FALSE.
verbose	Boolean specifying whether to display progress messages.
parallel	Boolean specifying whether to use parallel processing via the **BiocParallel** package. Defaults to TRUE.

Value

The function returns an KSomics object.

See Also

EMDomics ks.test

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_ks(dat, outcomes, nperm=10, parallel=FALSE)
head(results$ks)
```

`calculate_ks_gene` *Calculate KS score for a single gene*

Description

Calculate KS score for a single gene

Usage

```
calculate_ks_gene(vec, outcomes, sample_names)
```

Arguments

`vec`	A named vector containing data (e.g. expression data) for a single gene.
`outcomes`	A vector of group labels for the samples. The names must correspond to the names of `vec`.
`sample_names`	A character vector with the names of the samples in `vec`.

Details

All possible combinations of the classes are used as pairwise comparisons. The data in `vec` is divided based on class labels based on the `outcomes` identifiers given. For each pairwise computation, `ks.test` is used to compute the pairwise KS scores. The total KS score for the given data is the average of the pairwise KS scores.

Value

The KS score is returned.

See Also

`ks.test`

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(100000), nrow=100, ncol=1000)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:1000, sep="")

# assign outcomes
outcomes <- c(rep(1,500), rep(2,300), rep(3,200))
names(outcomes) <- colnames(dat)

calculate_ks_gene(dat[1,], outcomes, colnames(dat))
```

CVMomics *Create an CVMomics object*

Description

This is the constructor for objects of class 'CVMomics'. It is used in calculate_cvm to construct the return value.

Usage

```
CVMomics(data, outcomes, cvm, cvm.perm, pairwise.cvm.table, pairwise.q.table)
```

Arguments

data	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
outcomes	A vector of group labels for each of the sample identifiers. The names of this vector must correspond to the column names of data.
cvm	A matrix containing a row for each gene in data, and with the following columns: • cvm The calculated cvm score. • q-value The calculated q-value. The row names should specify the gene identifiers for each row.
cvm.perm	A matrix containing a row for each gene in data, and with a column containing cvm scores for each random permutation calculated via calculate_cvm.
pairwise.cvm.table	A table containing the CVM scores for each pairwise comparison for each gene. For a two-class problem, there should be only one column comparing class 1 and class 2. The row names should be gene identifiers. The column names should be in the format "<class 1> vs <class 2>" (e.g. "1 vs 2" or "A vs B").
pairwise.q.table	A table containing the permutation-based q-values for each pairwise comparison for each gene. May be NULL if pairwise.p=F.

Value

The function combines its arguments in a list, which is assigned class 'CVMomics'. The resulting object is returned.

See Also

calculate_cvm

EMDomics *Create an EMDomics object*

Description

This is the constructor for objects of class 'EMDomics'. It is used in calculate_emd to construct the return value.

Usage

```
EMDomics(data, outcomes, emd, emd.perm, pairwise.emd.table, pairwise.q.table)
```

Arguments

data	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
outcomes	A vector of group labels for each of the sample identifiers. The names of this vector must correspond to the column names of data.
emd	A matrix containing a row for each gene in data, and with the following columns: • emd The calculated emd score. • q-value The calculated q-value. The row names should specify the gene identifiers for each row.
emd.perm	A matrix containing a row for each gene in data, and with a column containing emd scores for each random permutation calculated via calculate_emd.
pairwise.emd.table	A table containing the EMD scores for each pairwise comparison for each gene. For a two-class problem, there should be only one column comparing class 1 and class 2. The row names should be gene identifiers. The column names should be in the format "<class 1> vs <class 2>" (e.g. "1 vs 2" or "A vs B").
pairwise.q.table	A table containing the permutation-based q-values for each pairwise comparison for each gene. May be NULL if pairwise.p=F.

Value

The function combines its arguments in a list, which is assigned class 'EMDomics'. The resulting object is returned.

See Also

calculate_emd

KSomics *Create an KSomics object*

Description

This is the constructor for objects of class 'KSomics'. It is used in `calculate_ks` to construct the return value.

Usage

```
KSomics(data, outcomes, ks, ks.perm, pairwise.ks.score, pairwise.ks.q = NULL)
```

Arguments

`data`	A matrix containing genomics data (e.g. gene expression levels). The rownames should contain gene identifiers, while the column names should contain sample identifiers.
`outcomes`	A vector of group labels for each of the sample identifiers. The names of this vector must correspond to the column names of `data`.
`ks`	A matrix containing a row for each gene in `data`, and with the following columns: • `ks` The calculated KS score. • `q-value` The calculated q-value (by permutation analysis). The row names should specify the gene identifiers for each row.
`ks.perm`	A matrix containing a row for each gene in `data`, and with a column containing KS scores for each random permutation calculated via `calculate_ks`.
`pairwise.ks.score`	A table containing the KS scores for each pairwise comparison for each gene. For a two-class problem, there should be only one column comparing class 1 and class 2. The row names should be gene identifiers. The column names should be in the format "<class 1> vs <class 2>" (e.g. "1 vs 2" or "A vs B").
`pairwise.ks.q`	A table of the same dimensions as `pairwise.ks.score` with the q-values for the pairwise comparisons. Q-values are computed by adjusting the p-value using the Benjamini-Hochberg method within each pairwise comparison.

Value

The function combines its arguments in a list, which is assigned class 'KSomics'. The resulting object is returned.

See Also

`calculate_ks`

plot_cvmnull — *Plot null distribution of permuted cvm scores vs. calculated cvm scores.*

Description

The median of the randomly permuted CVM scores (i.e. the null distribution) is plotted on the x-axis, vs. the observed CVM scores on the y-axis. The line y=x is superimposed.

Usage

```
plot_cvmnull(cvmobj)
```

Arguments

cvmobj	An CVMomics object, typically returned via a call to calculate_cvm.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_cvm ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "group A" = first 50, "group B" = second 50
groups <- c(rep("A",50),rep("B",50))
names(groups) <- colnames(dat)

results <- calculate_cvm(dat, groups, nperm=10, parallel=FALSE)
plot_cvmnull(results)
```

plot_cvmperms — *Plot histogram of CVM scores calculated via random permutation.*

Description

The permuted CVM scores stored in cvmobj$cvm.perm are plotted as a histogram.

Usage

```
plot_cvmperms(cvmobj)
```

Arguments

cvmobj	An CVMomics object, typically returned via a call to calculate_cvm.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_cvm ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_cvm(dat, outcomes, nperm=10, parallel=FALSE)
plot_cvmperms(results)
```

---

plot_cvm_density *Plot distributions and CVM score for a gene.*

Description

The data for the specified gene is retrieved from cvmobj$data. outcomes is used to divide the data into distributions for each group, which are then visualized as density distributions. The calculated CVM score for the specified gene is displayed in the plot title.

Usage

```
plot_cvm_density(cvmobj, gene_name)
```

Arguments

cvmobj	An CVMomics object, typically returned via a call to calculate_cvm.
gene_name	The gene to visualize. The name should be defined as a row name in cvmobj$cvm.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_cvm ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_cvm(dat, outcomes, nperm=10, parallel=FALSE)
plot_cvm_density(results, "gene5")
```

---

plot_emdnull *Plot null distribution of permuted EMD scores vs. calculated EMD scores.*

Description

The median of the randomly permuted EMD scores (i.e. the null distribution) is plotted on the x-axis, vs. the observed EMD scores on the y-axis. The line y=x is superimposed.

Usage

```
plot_emdnull(emdobj)
```

Arguments

emdobj	An EMDomics object, typically returned via a call to calculate_emd.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_emd ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "group A" = first 50, "group B" = second 50
groups <- c(rep("A",50),rep("B",50))
names(groups) <- colnames(dat)

results <- calculate_emd(dat, groups, nperm=10, parallel=FALSE)
plot_emdnull(results)
```

plot_emdperms *Plot histogram of EMD scores calculated via random permutation.*

Description

The permuted EMD scores stored in `emdobj$emd.perm` are plotted as a histogram.

Usage

```
plot_emdperms(emdobj)
```

Arguments

emdobj	An EMDomics object, typically returned via a call to calculate_emd.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_emd ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_emd(dat, outcomes, nperm=10, parallel=FALSE)
plot_emdperms(results)
```

plot_emd_density *Plot distributions and EMD score for a gene.*

Description

The data for the specified gene is retrieved from `emdobj$data`. `outcomes` is used to divide the data into distributions for each group, which are then visualized as density distributions. The calculated EMD score for the specified gene is displayed in the plot title.

Usage

```
plot_emd_density(emdobj, gene_name)
```

Arguments

emdobj	An EMDomics object, typically returned via a call to calculate_emd.
gene_name	The gene to visualize. The name should be defined as a row name in emdobj$emd.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_emd ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_emd(dat, outcomes, nperm=10, parallel=FALSE)
plot_emd_density(results, "gene5")
```

---

plot_ksnull *Plot null distribution of permuted ks scores vs. calculated ks scores.*

Description

The median of the randomly permuted KS scores (i.e. the null distribution) is plotted on the x-axis, vs. the observed KS scores on the y-axis. The line y=x is superimposed.

Usage

```
plot_ksnull(ksobj)
```

Arguments

ksobj	An KSomics object, typically returned via a call to calculate_ks.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_ks ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "group A" = first 50, "group B" = second 50
groups <- c(rep("A",50),rep("B",50))
names(groups) <- colnames(dat)

results <- calculate_ks(dat, groups, nperm=10, parallel=FALSE)
plot_ksnull(results)
```

`plot_ksperms` *Plot histogram of KS scores calculated via random permutation.*

Description

The permuted KS scores stored in `ksobj$ks.perm` are plotted as a histogram.

Usage

```
plot_ksperms(ksobj)
```

Arguments

`ksobj`	An `KSomics` object, typically returned via a call to `calculate_ks`.

Value

A `ggplot` object is returned. If the value is not assigned, a plot will be drawn.

See Also

`calculate_ks` `ggplot`

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_ks(dat, outcomes, nperm=10, parallel=FALSE)
plot_ksperms(results)
```

plot_ks_density *Plot distributions and KS score for a gene.*

Description

The data for the specified gene is retrieved from `ksobj$data`. `outcomes` is used to divide the data into distributions for each group, which are then visualized as density distributions. The calculated KS score for the specified gene is displayed in the plot title.

Usage

```
plot_ks_density(ksobj, gene_name)
```

Arguments

ksobj	An KSomics object, typically returned via a call to calculate_ks.
gene_name	The gene to visualize. The name should be defined as a row name in ksobj$ks.

Value

A ggplot object is returned. If the value is not assigned, a plot will be drawn.

See Also

calculate_ks ggplot

Examples

```
# 100 genes, 100 samples
dat <- matrix(rnorm(10000), nrow=100, ncol=100)
rownames(dat) <- paste("gene", 1:100, sep="")
colnames(dat) <- paste("sample", 1:100, sep="")

# "A": first 50 samples; "B": next 30 samples; "C": final 20 samples
outcomes <- c(rep("A",50), rep("B",30), rep("C",20))
names(outcomes) <- colnames(dat)

results <- calculate_ks(dat, outcomes, nperm=10, parallel=FALSE)
plot_ks_density(results, "gene5")
```

Index