Package 'ALDEx2'

April 14, 2017

Type Package

Title Analysis Of Differential Abundance Taking Sample Variation Into Account

Version 1.6.0

Date 2016-07-27

Author Greg Gloor, Ruth Grace Wong, Andrew Fernandes, Arianne Albert, Matt Links, Jia Rong Wu

Maintainer Greg Gloor <ggloor@uwo.ca>

Depends methods

Imports S4Vectors, IRanges, GenomicRanges, SummarizedExperiment, BiocParallel

biocViews DifferentialExpression, RNASeq, DNASeq, ChIPSeq, GeneExpression, Bayesian, Sequencing, Software, Microbiome, Metagenomics

Description A differential abundance analysis for the comparison of two or more conditions. For example, single-organism and meta-RNA-seq high-throughput sequencing assays, or of selected and unselected values from in-vitro sequence selections. Uses a Dirichlet-multinomial model to infer abundance from counts, that has been optimized for three or more experimental replicates. Infers sampling variation and calculates the expected false discovery rate given the biological and sampling variation using the Wilcox rank test or Welches t-test (aldex.ttest) or the glm and Kruskal Wallis tests (aldex.glm). Reports both P and fdr values calculated by the Benjamini Hochberg correction.

License file LICENSE

NeedsCompilation no

R topics documented:

`ALDEx2m-package` *Analysis of differential abundance taking sample variation into account*

Description

A differential abundance analysis for the comparison of two or more conditions. For example, single-organism and meta-RNA-seq high-throughput sequencing assays, or of selected and unselected values from in-vitro sequence selections. Uses a Dirichlet-multinomial model to infer abundance from counts, that has been optimized for three or more experimental replicates. Infers sampling variation and calculates the expected false discovery rate given the biological and sampling variation using the Wilcox rank test or Welches t-test (aldex.ttest) or the glm and Kruskal Wallis tests (aldex.glm). Reports both P and fdr values calculated by the Benjamini Hochberg correction.

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex.clr`, `aldex.ttest`, `aldex.glm`, `aldex.effect`, `selex`

Examples

```
# see examples for the aldex.clr, aldex.ttest, aldex.effect, aldex.glm functions
```

aldex *Compute an* aldex *Object*

Description

Generate Monte Carlo samples of the Dirichlet distribution for each sample. Convert each instance using the centred log-ratio transform Return two sample test values (Welch's t, Wilcoxon) or multi-sample test values (glm or Kruskal Wallace). Returns effect size values by defaulte.

Usage

```
aldex(reads, conditions, mc.samples=128, test="t",
    effect=TRUE, include.sample.summary=FALSE, verbose=FALSE, denom="all")
```

Arguments

reads	a non-negative, integer-only containing data.frame that has unique names for all rows and columns, where each row is a different gene and each column represents a sequencing read-count. Rows with 0 reads in each sample are deleted prior to analysis
conditions	a description of the data structure to be used for testing
mc.samples	the number of Monte Carlo samples to use to estimate the underlying distributions; since we are estimating central tendencies, 128 is usually sufficient
denom	A character variable default "all" indicating which features to retain as the denominator for the Geometric Mean calculation. Using "iqlr" accounts for data with systematic variation and centers the features on the set features that have variance that is between the lower and upper quartile of variance. Using "zero" is a more extreme case where there are many nonzero features in one condition but many zeros in another. In this case the geometric mean of each group is calculated using the set of per-group non-zero features.
test	which tests to perform: t = Welch's t and Wilcoxon, glm = Kruskal Wallace and glm
effect	calculate abundances and effect sizes
include.sample.summary	include median clr values for each sample, defaults to FALSE
verbose	Print diagnostic information while running. Useful only for debugging if fails on large datasets

Details

An explicit description of the input format for the reads object is shown under 'Examples', below. This is not intended to be the generic function. The system is intended to be used for demonstration or instructional purposes.

Value

returns a number of values that depends on the set of options. See the return values of aldex.ttest, aldex.glm, and aldex.effect for explanations and example

Author(s)

Greg Gloor, Andrew Fernandes and Matt Links contributed to this code

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex.ttest`, `aldex.glm`, `aldex.effect`, `aldex.corr`, `selex`

Examples

```
# The 'reads' data.frame should have row
# and column names that are unique, and
# looks like the following:
#
#              T1a T1b  T2  T3  N1  N2  Nx
#   Gene_00001   0   0   2   0   0   1   0
#   Gene_00002  20   8  12   5  19  26  14
#   Gene_00003   3   0   2   0   0   0   1
#   Gene_00004  75  84 241 149 271 257 188
#   Gene_00005  10  16   4   0   4  10  10
#   Gene_00006 129 126 451 223 243 149 209
#       ... many more rows ...

data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex(selex, conds, mc.samples=2 ,denom="all", test="t",
effect=FALSE, verbose=FALSE)
```

`aldex.clr` *Compute an* `aldex.clr` *Object*

Description

Generate Monte Carlo samples of the Dirichlet distribution for each sample. Convert each instance using the centred log-ratio transform This is the input for all further analyses.

Usage

```
aldex.clr(reads, conds, mc.samples = 128, denom="all", verbose=FALSE, useMC=FALSE)
```

Arguments

`reads`	A `data.frame` or `RangedSummarizedExperiment` object containing non-negative integers only and with unique names for all rows and columns, where each row is a different gene and each column represents a sequencing read-count. Rows with 0 reads in each sample are deleted prior to analysis.

conds	A vector containing a descriptor for the samples, allowing them to be grouped and compared.
mc.samples	The number of Monte Carlo samples to use to estimate the underlying distributions; since we are estimating central tendencies, 128 is usually sufficient.
denom	A character variable default "all" indicating which features to use as the denominator for the Geometric Mean calculation. Using "all" uses the geometric mean abundance of all features. Using "iqlr" accounts for data with systematic variation and uses the features that are between the first and third quartile of the variance of the clr values across all sample. Using "zero" uses the non-zero features in each grop as the denominator. This approach is an extreme case where there are many nonzero features in one condition but many zeros in another. It is also possible to supply a vector of row indices to use as the denominator. Here, the experimentalist is determining a-priori which rows are thought to be invariant. In the case of RNA-seq, this could include ribosomal protein genes and and other house-keeping genes.
verbose	Print diagnostic information while running. Useful only for debugging if fails on large datasets.
useMC	Use multicore by default (FALSE). Multi core processing will be attempted with the BiocParallel package. Serial processing will be used if this is not possible.

Details

An explicit description of the input format for the reads object is shown under 'Examples', below.

Value

The object produced by the clr function contains the clr transformed values for each Monte-Carlo Dirichlet instance, which can be accessed through getMonteCarloInstances(x), where x is the clr function output. Each list element is named by the sample ID. getFeatures(x) returns the features, getSampleIDs(x) returns sample IDs, and getFeatureNames(x) returns the feature names.

Author(s)

Greg Gloor, Ruth Grace Wong, Andrew Fernandes, Matt Links and Jia Rong Wu contributed to this code.

References

Please use the citation given by citation(package="ALDEx").

See Also

aldex.ttest, aldex.glm, aldex.effect, selex

Examples

```
# The 'reads' data.frame or
# RangedSummarizedExperiment object should
# have row and column names that are unique,
# and looks like the following:
#
#             T1a T1b  T2  T3  N1  N2  Nx
```

```
#   Gene_00001   0   0   2   0   0   1   0
#   Gene_00002  20   8  12   5  19  26  14
#   Gene_00003   3   0   2   0   0   0   1
#   Gene_00004  75  84 241 149 271 257 188
#   Gene_00005  10  16   4   0   4  10  10
#   Gene_00006 129 126 451 223 243 149 209
#       ... many more rows ...

data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples=2, denom="all", verbose=FALSE)
```

`aldex.clr-class` *The aldex.clr class*

Description

The aldex.clr S4 class is a class which stores the data generated by the aldex.clr method.

Details

An aldex.clr object contains the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data. It is created by the `aldex.clr.function`, which is invoked by the aldex.clr method. It consists of four attributes: the sample names, the feature names, the conditions vector (assigns each sample to a condition), and the Monte Carlo Dirochlet instances themselves. These can be accessed, along with information about the length of some attributes. A single Monte Carlo instance can also be retrieved.

Value

The aldex.clr object contains the clr transformed values for each Monte-Carlo Dirichlet instance, which can be accessed through `getMonteCarloInstances(x)`, where `x` is the `clr` function output. Each list element is named by the sample ID. `getFeatures(x)` returns the features, `getSampleIDs(x)` returns sample IDs, and `getFeatureNames(x)` returns the feature names.

Methods

In the code below, `x` is an `aldex.clr` object, and `i` is a numeric whole number.

`getMonteCarloInstances(x)`: Returns `x`'s Monte Carlo Dirichlet instances.

`getSampleIDs(x)`: Returns the names of the samples. These can be used to access the original reads, as in `reads$sampleID` (if the reads are a data frame).

`getFeatures(x)`: Returns the names of the features as a vector.

`numFeatures(x)`: Returns the number of features associated with the data.

`numMCInstances(x)`: Returns the names of the `keys` that can be used to subset the data rows. The `keys` values are the rsid's.

`getFeatureNames(x)`: Returns the names of the `keys` that can be used to subset the data rows. The `keys` values are the rsid's.

`getReads(x)`: Returns the names of the `keys` that can be used to subset the data rows. The `keys` values are the rsid's.

`numConditions(x)`: Returns the names of the `keys` that can be used to subset the data rows. The `keys` values are the rsid's.

`getMonteCarloReplicate(x, i)`: Returns the names of the `keys` that can be used to subset the data rows. The `keys` values are the rsid's.

Author(s)

Greg Gloor, Ruth Grace Wong, Andrew Fernandes, Jia Rong Wu and Matt Links contributed to this code

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex.clr.function`

Examples

```
# The 'reads' data.frame or
# SummarizedExperiment object should have
# row and column names that are unique,
# and looks like the following:
#
#              T1a T1b  T2  T3  N1  N2  Nx
#   Gene_00001   0   0   2   0   0   1   0
#   Gene_00002  20   8  12   5  19  26  14
#   Gene_00003   3   0   2   0   0   0   1
#   Gene_00004  75  84 241 149 271 257 188
#   Gene_00005  10  16   4   0   4  10  10
#   Gene_00006 129 126 451 223 243 149 209
#       ... many more rows ...

data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))

# x is an object of type aldex.clr
x <- aldex.clr(selex, conds, mc.samples = 2, denom="all", verbose = FALSE)

# get all of the Monte Carlo Dirochlet instances
monteCarloInstances <- getMonteCarloInstances(x)

# get sample names
sampleIDs <- getSampleIDs(x)

# get features
features <- getFeatures(x)

# get number of features
numFeatures <- numFeatures(x)
```

```
# get number of Monte Carlo Dirochlet instances
numInstances <- numMCInstances(x)

# get names of features
featureNames <- getFeatureNames(x)

# get number of conditions
conditions <- numConditions(x)

# get number of conditions
reads <- getReads(x)

# retrieve the first Monte Carlo Dirochlet instance.
monteCarloInstance <- getMonteCarloReplicate(x,1)
```

`aldex.corr` *calculate Pearson's Product moment and Spearman's rank correlations*

Description

calculates expected values of Pearson's Product moment and Spearman's rank correlations on the data returned by aldex.clr. NOTE: this function will be replaced by a compositionally correct method in the next release cycle.

Usage

```
aldex.corr(clr, covar)
```

Arguments

`clr`	`clr` is the data output of the aldex.clr function
`covar`	a per-sample continuous variable to be correlated with the clr values

Details

An explicit example for two conditions is shown in the 'Examples' below.

Value

Outputs a dataframe with the following information:

`pearson.ecor`	a vector containing the expected Pearson's Product moment value for each feature
`pearson.ep`	a vector containing the expected P value of the Pearson Product moment value for each feature
`pearson.eBH`	a vector containing the expected Benjamini-Hochberg corrected P value of the Pearson Product moment value for each feature
`spearman.erho`	a vector containing the expected Spearman's rank correlation value for each feature

spearman.ep	a vector containing the expected P value of Spearman's rank correlation value for each feature
spearman.eBH	a vector containing the expected Benjamini-Hochberg corrected P value of Spearman's rank correlation value for each feature

Author(s)

Arianne Albert

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

aldex.clr, aldex.glm, aldex.effect, selex

Examples

```
# x is the output of the \code{x <- aldex.clr(data, conds, mc.samples, denom="all", useMC)} function
 # conditions is a description of the data
 # aldex.ttest(clr, covar)
```

---

aldex.effect *calculate effect sizes and differences between conditions*

Description

determines the median clr abundance of the feature in all samples and in groups determines the median difference between the two groups determines the median variation within each two group determines the effect size, which is the median of the ratio of the between group difference and the larger of the variance within groups

Usage

```
aldex.effect(clr, conditions, verbose = TRUE, include.sample.summary = FALSE,
    useMC=FALSE)
```

Arguments

clr	clr is the data output of aldex.clr
conditions	a description of the data structure to be used for testing
verbose	Print diagnostic information while running. Useful only for debugging if fails on large datasets
include.sample.summary	include median clr values for each sample, defaults to FALSE
useMC	use multicore by default (FALSE)

Details

An explicit example for two conditions is shown in the 'Examples' below.

Value

returns a dataframe with the following information:

`rab.all`	a vector containing the median clr value for each feature
`rab.win.conditionA`	a vector containing the median clr value for each feature in condition A
`rab.win.conditionB`	a vector containing the median clr value for each feature in condition B
`diff.btw`	a vector containing the per-feature median difference between condition A and B
`diff.win`	a vector containing the per-feature maximum median difference between Dirichlet instances within conditions
`effect`	a vector containing the per-feature effect size
`overlap`	a vector containing the per-feature proportion of effect size that is 0 or less

Author(s)

Greg Gloor, Andrew Fernandes, Matt Links

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex.clr`, `aldex.ttest`, `aldex.glm`, `selex`

Examples

```
# x is the output of the \code{x <- clr(data, mc.samples)} function
# conditions is a description of the data
# for the selex dataset, conditions <- c(rep("N", 7), rep("S", 7))
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples=2, denom="all")
effect.test <- aldex.effect(x, conds)
```

`aldex.glm` *calculate glm and Kruskal Wallis test statistics*

Description

calculates expected values of the glm and Kruskal Wallis functions on the data returned by clr_function.r

Usage

```
aldex.glm(clr, conditions, useMC=FALSE)
```

Arguments

clr	clr is the data output of aldex.clr
conditions	a description of the data structure to be used for testing
useMC	use multicore by default (FALSE)

Details

An explicit example for two conditions is shown in the 'Examples' below.

Value

Outputs a dataframe with the following information:

kw.ep	a vector containing the expected P value of the Kruskal Wallis test for each feature
kw.eBH	a vector containing the expected value of the Benjamini Hochberg corrected P value for each feature
glm.ep	a vector containing the expected P value of the glm test for each feature
glm.eBH	a vector containing the expected value of the Benjamini Hochberg corrected P value for each feature

Author(s)

Arianne Albert

References

Please use the citation given by citation(package="ALDEx").

See Also

aldex.clr, aldex.ttest, aldex.effect, selex

Examples

```
# x is the output of the \code{x <- aldex.clr(data, mc.samples)} function
# conditions is a description of the data
# for the selex dataset, conditions <- c(rep("N", 7), rep("S", 7))
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples=1, denom="all")
glm.test <- aldex.glm(x, conds)
```

aldex.plot *Plot an* aldex *Object*

Description

Create 'MW'- or 'MA'-type plots from the given aldex object.

Usage

```
    ## S3 method for class 'plot'
aldex(  x, ..., type=c("MW","MA"),
    xlab=NULL, ylab=NULL, xlim=NULL, ylim=NULL,
    all.col=rgb(0,0,0,0.2), all.pch=19, all.cex=0.4,
    called.col=red, called.pch=20, called.cex=0.6,
    thres.line.col=darkgrey, thres.lwd=1.5,
    test=welch, cutoff=0.1, rare.col=black, rare=0,
    rare.pch=20, rare.cex=0.2 )
```

Arguments

x	an object of class aldex produced by the aldex function
...	optional, unused arguments included for compatibility withe the S3 method signature
type	which type of plot is to be produced. MA is a Bland-Altman style plot; MW is a difference between to a variance within plot as described in the paper
test	the method of calculating significance, one of: welch = welch's t test; wilcox = wilcox rank test; glm = glm; kruskal = Kruskal-Wallace test
cutoff	the Benjamini-Hochberg fdr cutoff, default 0.1
xlab	the x-label for the plot, as per the parent plot function
ylab	the y-label for the plot, as per the parent plot function
xlim	the x-limits for the plot, as per the parent plot function
ylim	the y-limits for the plot, as per the parent plot function
all.col	the default colour of the plotted points
all.pch	the default plotting symbol
all.cex	the default symbol size
called.col	the colour of points with false discovery rate, q <= 0.1
called.pch	the symbol of points with false discovery rate, q <= 0.1
called.cex	the character expansion of points with false discovery rate, q <= 0.1
thres.line.col	the colour of the threshold line where within and between group variation is equivalent
thres.lwd	the width of the threshold line where within and between group variation is equivalent
rare	relative abundance cutoff for rare features, default 0 or the mean abundance
rare.col	color for rare features, default black
rare.pch	the default symbol of rare features
rare.cex	the default symbol size of rare points

Details

This particular specialization of the `plot` function is relatively simple and provided for convenience. For more advanced control of the plot is is best to use the values returned by `summary(x)`.

Value

None.

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex`, `aldex.effect`, `aldex.ttest`, `aldex.glm`

Examples

```
# See the examples for 'aldex'.
```

`aldex.set.mode` *identify set of denominator features for log-ratio calculation*

Description

calculate the features that are to be used as the denominator for the Geometric Mean calculation in clr_function.R

Usage

```
aldex.set.mode(reads, conds, denom="all")
```

Arguments

`reads`	A data frame containing the samples and features per sample.
`conds`	A vector describing which samples belong to what condition.
`denom`	Character argument specifying which indicies to return. 'all' returns all features in both conditons. 'zero' returns the nonzero count features per condition. 'iqlr' returns the features whose variance falls within the inter-quantile range of the CLR-transformed data. In cases of malformed or null queries, input defaults to 'all'. Additionally, the input can be a numeric vector, which contains a set of row indicies to center the data against. Only for advanced users who can pre-determine the invariant set of features within their data.

Details

An explicit example for two conditions is shown in the 'Examples' below.

Value

Outputs a vector containing indicies per condition.

Author(s)

Jia Rong Wu

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

`aldex.clr`, `aldex.ttest`, `aldex.effect`, `selex`

Examples

```
# x is the output of the \code{x <- clr(data, mc.samples)} function
# conditions is a description of the data
# for the selex dataset, conditions <- c(rep("N", 7), rep("S", 7))
# input can be "all", "iqlr", "zero" or numeric for advanced users
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples=2, denom="all")
```

`aldex.ttest` *calculate Welch's t-test and Wilcoxon test statistics*

Description

calculates expected values of the Welch's t-test and Wilcoxon rank test on the data returned by clr_function.r

Usage

```
aldex.ttest(clr, conditions, paired.test = FALSE, hist.plot=FALSE)
```

Arguments

`clr`	`clr` is the data output of the aldex.clr function
`conditions`	a description of the data structure to be used for testing
`paired.test`	whether the Welch's test should be paired or not
`hist.plot`	whether to plot a histogram of P values for an individual Dirichlet Monte-Carlo instance. Plot is output to the standard R plotting device.

Details

An explicit example for two conditions is shown in the 'Examples' below.

Value

Outputs a dataframe with the following information:

we.ep	a vector containing the expected P value of the Welch's t-test for each feature
we.eBH	a vector containing the expected value of the Benjamini Hochberg corrected P value for each feature
wi.ep	a vector containing the expected P value of the Wilcoxon test for each feature
wi.eBH	a vector containing the expected value of the Benjamini Hochberg corrected P value for each feature

Author(s)

Greg Gloor

References

Please use the citation given by `citation(package="ALDEx")`.

See Also

aldex.clr, aldex.glm, aldex.effect, selex

Examples

```
# x is the output of the \code{x <- aldex.clr(data, mc.samples)} function
# conditions is a description of the data
# for the selex dataset, conditions <- c(rep("N", 7), rep("S", 7))
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples=2, denom="all")
ttest.test <- aldex.ttest(x, conds)
```

---

getFeatureNames *getFeatureNames*

Description

Returns the names of the features as a vector, for an `aldex.clr` object.

Usage

```
getFeatureNames(.object)
```

Arguments

.object	A `aldex.clr` object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the names of the keys that can be used to subset the data rows. The keys values are the rsid's.

Value

A vector of feature names.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom="all", verbose = FALSE)
featureNames <- getFeatureNames(x)
```

---

getFeatures *getFeatures*

Description

Returns the features as a vector, for an aldex.clr object.

Usage

```
getFeatures(.object)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the features as a vector, for an aldex.clr object.

Value

A vector of features.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom="all", verbose = FALSE)
features <- getFeatures(x)
```

`getMonteCarloInstances` *getMonteCarloInstances*

Description

Returns the Monte Carlo Dirochlet instances used to create an `aldex.clr` object.

Usage

```
getMonteCarloInstances(.object)
```

Arguments

`.object`	A `aldex.clr` object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the Monte Carlo Dirochlet instances used to create an `aldex.clr` object.

Value

A list of data frames of Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data.

See Also

`aldex.clr`

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
monteCarloInstances <- getMonteCarloInstances(x)
```

getMonteCarloReplicate *getMonteCarloReplicate*

Description

Returns the designated Monte Carlo Dirochlet replicate generated from analysis, for an aldex.clr object.

Usage

```
getMonteCarloReplicate(.object,i)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.
i	The numeric index of the desired replicate.

Details

Returns the designated Monte Carlo Dirochlet replicate generated from analysis.

Value

A data frame representing the designated Monte Carlo Dirochlet replicate generated from analysis.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
monteCarloInstance <- getMonteCarloReplicate(x,1)
```

getReads *getReads*

Description

Returns the count table used as input for analysis, for an aldex.clr object.

Usage

```
getReads(.object)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the count table.

Value

A data frame representing the count table used as input for analysis.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
reads <- getReads(x)
```

getSampleIDs *getSampleIDs*

Description

Returns the names of the samples for an aldex.clr object. These can be used to access the original reads, as in reads$sampleID (if the reads are a data frame).

Usage

```
getSampleIDs(.object)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the names of the samples. These can be used to access the original reads, as in reads$sampleID (if the reads are a data frame).

Value

A vector of sample names.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
sampleIDs <- getSampleIDs(x)
```

---

numConditions *numConditions*

Description

Returns the number of conditions compared for analysis, for an aldex.clr object.

Usage

```
numConditions(.object)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the number of conditions compared.

Value

A numeric representing the number of conditions compared.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
conditions <- numConditions(x)
```

---

numFeatures *numFeatures*

Description

Returns the number of features associated with the data, for an aldex.clr object.

Usage

```
numFeatures(.object)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the number of features associated with the data.

Value

A numeric representing the number of features associated with the data.

See Also

aldex.clr

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
numFeatures <- numFeatures(x)
```

numMCInstances *numMCInstances*

Description

Returns the number of Monte Carle Dirochlet instances generated for analysis, for an `aldex.clr` object.

Usage

```
numMCInstances(.object)
```

Arguments

`.object`	A `aldex.clr` object containing the Monte Carlo Dirochlet instances derived from estimating the technical variance of the raw read count data, along with sample and feature information.

Details

Returns the number of Monte Carle Dirochlet instances generated for analysis.

Value

A numeric representing the number of Monte Carle Dirochlet instances generated for analysis.

See Also

`aldex.clr`

Examples

```
data(selex)
    #subset for efficiency
    selex <- selex[1201:1600,]
conds <- c(rep("NS", 7), rep("S", 7))
x <- aldex.clr(selex, conds, mc.samples = 2, denom = "all", verbose = FALSE)
numInstances <- numMCInstances(x)
```

selex *Selection-based differential sequence variant abundance dataset*

Description

This data set gives the differential abundance of 1600 enzyme variants grown under selective (NS) and selective (S) conditions

Usage

```
selex
```

Format

A dataframe of 1600 features and 14 samples. The first 7 samples are non-selected, the last 7 are selected.

Source

McMurrough et al (2014) PNAS doi:10.1073/pnas.1322352111

References

McMurrough et al (2014) PNAS doi:10.1073/pnas.1322352111

Index