Package 'ALDEx2'

November 30, 2022

Type Package

Title Analysis Of Differential Abundance Taking Sample Variation Into Account

Version 1.30.0

Date 2022-05-09

Author Greg Gloor, Andrew Fernandes, Jean Macklaim, Arianne Albert, Matt Links, Thomas Quinn, Jia Rong Wu, Ruth Grace Wong, Brandon Lieng

Maintainer Greg Gloor <ggloor@uwo.ca>

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

Description A differential abundance analysis for the comparison of two or more conditions. Useful for analyzing data from standard RNA-seq or meta-RNA-seq assays as well as selected and unselected values from in-vitro sequence selections. Uses a Dirichlet-multinomial model to infer abundance from counts, optimized for three or more experimental replicates. The method infers biological and sampling variation to calculate the expected false discovery rate, given the variation, based on a Wilcoxon Rank Sum test and Welch's t-test (via aldex.ttest), a Kruskal-Wallis test (via aldex.kw), a generalized linear model (via aldex.glm), or a correlation test (via aldex.corr). All tests report p-values and Benjamini-Hochberg corrected p-values. ALDEx2 also calculates expected standardized effect sizes for paired or unpaired study designs.

License GPL (>= 3)

URL https://github.com/ggloor/ALDEx_bioc

BugReports https://github.com/ggloor/ALDEx_bioc/issues

RoxygenNote 7.2.0

VignetteBuilder knitr

Depends methods, stats, zCompositions,

Imports Rfast, BiocParallel, GenomicRanges, IRanges, S4Vectors, SummarizedExperiment, multtest

Suggests testthat, BiocStyle, knitr, rmarkdown

git_url https://git.bioconductor.org/packages/ALDEx2

git_branch RELEASE_3_16

git_last_commit cb66705

git_last_commit_date 2022-11-01

Date/Publication 2022-11-30

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

Welcome to the `ALDEx2` package!

The `aldex` function is a wrapper that performs log-ratio transformation and statistical testing in a single line of code. Specifically, this function: (a) generates Monte Carlo samples of the Dirichlet distribution for each sample, (b) converts each instance using a log-ratio transform, then (c) returns test results for two sample (Welch's t, Wilcoxon) or multi-sample (glm, Kruskal-Wallace) tests. This function also estimates effect size for two sample analyses.

Usage

```
aldex(
  reads,
  conditions,
  mc.samples = 128,
  test = "t",
```

```
  effect = TRUE,
  include.sample.summary = FALSE,
  verbose = FALSE,
  denom = "all",
  paired.test = FALSE,
  iterate = FALSE,
  ...
)
```

Arguments

reads	A non-negative, integer-only data.frame or matrix with unique names for all rows and columns. Rows should contain genes and columns should contain sequencing read counts (i.e., sample vectors). Rows with 0 reads in each sample are deleted prior to analysis.
conditions	A character vector. A description of the data structure used for testing. Typically, a vector of group labels. For aldex.glm, use a model.matrix.
mc.samples	An integer. The number of Monte Carlo samples to use when estimating the underlying distributions. Since we are estimating central tendencies, 128 is usually sufficient.
test	A character string. Indicates which tests to perform. "t" runs Welch's t and Wilcoxon tests. "kw" runs Kruskal-Wallace and glm tests. "glm" runs a generalized linear model using a model.matrix. "corr" runs a correlation test using cor.test.
effect	A boolean. Toggles whether to calculate abundances and effect sizes. Applies to test = "t" and test = "iterative".
include.sample.summary	A boolean. Toggles whether to include median clr values for each sample. Applies to effect = TRUE.
verbose	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets. Applies to effect = TRUE.
denom	A character string. Indicates 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 non-zero 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.
paired.test	A boolean. Toggles whether to do paired-sample tests. Applies to effect = TRUE and test = "t".
iterate	A boolean. Toggles whether to iteratively perform a test. For example, this will use the results from an initial "t" routine to seed the reference (i.e., denominator of Geometric Mean calculation) for a second "t" routine.
...	Arguments to embedded method (e.g., glm or cor.test).

Details

See "Examples" below for a description of the sample input.

Value

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

Author(s)

Greg Gloor, Andrew Fernandes, and Matt Links contributed to the original package. Thom Quinn added the "glm" test method, the "corr" test method, and the "iterate" procedure.

References

Please use the citation given by `citation(package="ALDEx2")`.

See Also

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

---

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

Description

Generate Monte Carlo samples of the Dirichlet distribution for each sample. Convert each instance using a 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 sample. 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 instances to use to estimate the underlying distributions; since we are estimating central tendencies, 128 is usually sufficient, but larger numbers may be .
denom	An any variable (all, iqlr, zero, lvha, median, user) indicating features to use as the denominator for the Geometric Mean calculation The default "all" uses the geometric mean abundance of all features. Using "median" returns the median abundance of all features. Using "iqlr" uses the features that are between the first and third quartile of the variance of the clr values across all samples. 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. Using "lvha" uses features that have low variance (bottom quartile) and high relative abundance (top quartile in every sample). 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. This should be used with caution because the offsets may be different in the original data and in the data used by the function because features that are 0 in all samples are removed by aldex.clr.
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. In practice serial and multicore are nearly the same speed because of overhead in setting up the parallel processes.

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 log-ratio 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, Thom Quinn, 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 centre-log ratio transformed Monte Carlo Dirichlet 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 eight slots: the reads, the condition information, the number of instances, the denominator, whether it was invoked as verbose, and if multi-cores was used, the Dirichlet Monte-Carlo probabilities, and the centre-log

ratio transformed Monte Carlo probabilities. These can be accessed along with information about the length of some attributes.

Value

The aldex.clr object contains the raw data, the estimated probabilities drawn from a Dirichlet distribution, and the clr transformed values for each Monte-Carlo instance. These can be accessed through getters outlined below.

Methods

In the code below, `x` is an `aldex.clr` object, and `i`, is a positive integer. There are N samples, D features, and M Monte-Carlo instances.

`getMonteCarloInstances(x)`: Returns the clr transformed Monte Carlo Dirichlet instances as a list where each list entry is a single sample containing a D x M matrix.

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

`getFeatureNames(x)`: Returns the names of the `keys`. These can be used to subset the data rows.

`getFeatures(x)`: Returns the clr transformed values for the features in the first sample.

`numFeatures(x)`: Returns the number of features that were non-0 in at least one sample.

`numMCInstances(x)`: Returns the number of Monte-Carlo instances.

`getReads(x)`: Returns the input data as used by the method.

`numConditions(x)`: Returns the number of samples in the `conditions` analysis.

`getMonteCarloReplicate(x, i)`: Returns the D x M matrix containing the Monte-Carlo instances for one sample.

`getMonteCarloSample(x, i)`: Returns the N x D matrix containing Monte-Carlo instance `i` for for all samples.

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))
i=1

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

# get reads plus uniform prior
reads <- getReads(x)

# get a list containing all of the clr transformed instances
monteCarloInstances <- getMonteCarloInstances(x)

# get a list containing all of the Monte-Carlo Dirichlet instances
monteCarloDirInstances <- getDirichletInstances(x)

# retrieve the clr transformed instances for sample i.
monteCarloInstance <- getMonteCarloReplicate(x,i)

# retrieve the Monte-Carlo Dirichlet instances for sample i.
monteCarloDirInstance <- getDirichletReplicate(x,i)

# retrieve the clr transformed instance i for all samples
monteCarloSample <- getMonteCarloSample(x,i)

# retrieve the Monte-Carlo Dirichlet instance i for all samples
monteCarloDirSample <- getDirichletSample(x,i)

# get sample names
sampleIDs <- getSampleIDs(x)

# get features
features <- getFeatures(x)

# get number of features with at least one count
```

```
numFeatures <- numFeatures(x)

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

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

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

# get the offset of the features in the log-ratio  denominator
denom <- getDenom(x)
```

`aldex.corr` *Calculate correlation with a continuous variable*

Description

`aldex.corr` calculates the expected values for the correlation between each feature and a continuous variable, using data returned returned by `aldex.clr` and a vector of the continuous variable. Returns results of Pearson, Spearman and Kendall tests.

Usage

```
aldex.corr(clr, cont.var)
```

Arguments

`clr`	An `ALDEx2` object. The output of `aldex.clr`.
`cont.var`	A continuous numeric vector

Value

Returns a data.frame of the average Pearson, Spearman and Kendall coefficients and their p-values for each feature, with FDR appended as a `BH` column.

Author(s)

Arianne Albert, Greg Gloor, Thom Quinn

References

Please use the citation given by `citation(package="ALDEx2")`.

See Also

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

Examples

```
data(selex)
#subset for efficiency
selex <- selex[1:100,]
conds <- c(rep("N", 7), rep("S",7))
cont.var <-  c(rep(1,7), rep(2,7))
x <- aldex.clr(selex, conds)
corr.test <- aldex.corr(x, cont.var)
```

`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, verbose = TRUE, include.sample.summary = FALSE, useMC=FALSE,
  CI=FALSE, glm.conds=NULL, paired.test=FALSE)
```

Arguments

`clr`	`clr` is the data output of `aldex.clr`
`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)
`CI`	give effect 95
`glm.conds`	give effect for glm contrasts, note: saved as list
`paired.test`	calculate effect size for paired samples, defaults to 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)
```

aldex.expectedDistance *Calculate the expected values of distances between samples, given an* aldex *Object*

Description

Calculates the expected value of distances between samples, given an aldex Object, using the median value of distances derived from n Monte-Carlo replicates.

Usage

```
    ## S3 method for class 'expectedDistance'
aldex(clrData)
```

Arguments

clrData	an object of class aldex produced by the aldex function

Details

Generates a distance matrix for each Monte-Carlo instance in an aldex Object. Calculates the median distance value across all instances.

Value

Returns a dist Object.

References

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

See Also

aldex, aldex.clr dist

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 = 128, denom = "all", verbose = FALSE)
x.dist <- aldex.expectedDistance(x)
```

aldex.glm *Calculate glm test statistics using a* model.matrix

Description

aldex.glm calculates the expected values for each coefficient of a glm model on the data returned by aldex.clr. This function requires the user to define a model with model.matrix.

Usage

```
aldex.glm(clr, verbose = FALSE, ...)
```

Arguments

clr	An ALDEx2 object. The output of aldex.clr.
verbose	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets. Applies to effect = TRUE.
...	Arguments passed to glm.

Value

Returns a data.frame of the average coefficients and their p-values for each feature, with FDR appended as a holm column.

Author(s)

Thom Quinn

References

Please use the citation given by citation(package="ALDEx2").

See Also

aldex, aldex.clr, aldex.ttest, aldex.kw, aldex.glm, aldex.effect, aldex.corr, selex

Examples

```
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
covariates <- data.frame("A" = sample(0:1, 14, replace = TRUE),
                         "B" = c(rep(0, 7), rep(1, 7)))
mm <- model.matrix(~ A + B, covariates)
x <- aldex.clr(selex, mm, mc.samples=4, denom="all")
glm.test <- aldex.glm(x)
```

`aldex.glm.effect` *calculate effect sizes and differences between all constrasts for the aldex.glm model matrix*

Description

data for this function is saved in a list with entries named by contrast 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.glm.effect(clr, verbose = TRUE, include.sample.summary = FALSE,
  useMC=FALSE, CI=FALSE)
```

Arguments

`clr`	`clr` is the data output of `aldex.clr`
`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)
`CI`	give effect 95

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.effect`, `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,]
covariates <- data.frame("A" = sample(0:1, 14, replace = TRUE),
              "B" = c(rep(0, 7), rep(1, 7)),
              "Z" = sample(c(1,2,3), 14, replace=TRUE))
mm <- model.matrix(~ A + Z + B, covariates)
x <- aldex.clr(selex, mm, mc.samples=8, denom="all")
glm.effect <- aldex.glm.effect(x)
```

`aldex.kw` *Calculate the Kruskal-Wallis test and glm statistics*

Description

`aldex.kw` calculates the expected values of the Kruskal-Wallis test and a glm on the data returned by `aldex.clr`.

Usage

```
aldex.kw(clr, useMC = FALSE, verbose = FALSE)
```

Arguments

`clr`	An `ALDEx2` object. The output of `aldex.clr`.
`useMC`	Toggles whether to use multi-core.
`verbose`	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets. Applies to `effect = TRUE`.

Details

use the aldex.glm function unless you really need the nonparametric KW test

Value

Returns a `data.frame` 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 corresponding expected value of the Benjamini-Hochberg corrected p-value for each feature
`glm.ep`	a vector containing the expected p-value of the glm ANOVA for each feature
`glm.eBH`	a vector containing the corresponding expected value of the Benjamini-Hochberg corrected p-value for each feature. Note, you should use the aldex.glm function for better post-hoc test statistics.

Author(s)

Arianne Albert

References

Please use the citation given by `citation(package="ALDEx2")`.

See Also

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

Examples

```
data(selex)
#subset for efficiency
selex <- selex[1201:1600,]
conds <- c(rep("A", 4), rep("B", 3), rep("C", 7))
x <- aldex.clr(selex, conds, mc.samples=1, denom="all")
kw.test <- aldex.kw(x)
```

---

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.pval=0.1, cutoff.effect=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 with 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: http://dx.doi.org/10.1080/10618600.2015.113
test	the method of calculating significance, one of: welch = welch's t test; wilcox = wilcox rank test; glm = glm; kruskal = Kruskal-Wallace test
cutoff.pval	the Benjamini-Hochberg fdr cutoff, default 0.1
cutoff.effect	the effect size cutoff for plotting, default 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.plotFeature *Show dispersion of the expected values returned by* `aldex.effect`

Description

`aldex.plotFeature` generates density plots showing the dispersion of the expected values given in the output from `aldex.effect`. The expected values are shown in the plots. This is a diagnostic visualization to help determine if the expected values are trustworthy

Usage

```
aldex.plotFeature(
  clrData,
  featureName,
  pooledOnly = FALSE,
  densityOnly = FALSE
)
```

Arguments

`clrData`	the output object from `aldex.clr`
`featureName`	the name of the feature from the input data
`pooledOnly`	show only the pooled plots, default FALSE, shows all plots
`densityOnly`	show only the density plots, default FALSE includes expected values

Author(s)

Brandon Lieng, Greg Gloor

References

Please use the citation given by `citation(package="ALDEx2")`.

See Also

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

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=4, denom="all")
aldex.plotFeature(x, "S:D:A:D")
```

---

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. Check that the same number of features are in the input and output datasets.

Details

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

Value

Outputs a vector containing indicies per condition, or a single vector in some cases.

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 Wilcoxon Rank Sum test and Welch's t-test statistics*

Description

`aldex.ttest` calculates the expected values of the Wilcoxon Rank Sum test and Welch's t-test on the data returned by `aldex.clr`.

Usage

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

Arguments

`clr`	An `ALDEx2` object. The output of `aldex.clr`.
`paired.test`	Toggles whether to calculate paired tests.
`hist.plot`	Toggles whether to plot a histogram of p-values for the first Dirichlet Monte Carlo instance.
`verbose`	A boolean. Toggles whether to print diagnostic information while running. Useful for debugging errors on large datasets. Applies to `effect` = `TRUE`.

Value

Returns a `data.frame` with the following information:

`we.ep`	a vector containing the expected p-value of Welch's t-test for each feature
`we.eBH`	a vector containing the corresponding expected value of the Benjamini-Hochberg corrected p-value for each feature
`wi.ep`	a vector containing the expected p-value of the Wilcoxon Rank Sum test for each feature
`wi.eBH`	a vector containing the corresponding 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="ALDEx2")`.

See Also

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

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")
ttest.test <- aldex.ttest(x)
```

`getDenom` *getDenom*

Description

Returns the offset of the features used as the denominator as the basis for the log-ratio, for an `aldex.clr` object.

Usage

```
getDenom(.object)
```

Arguments

`.object`	A `aldex.clr` object.

Details

Returns the offset of the features used as the denominator as the basis for the log-ratio. A vector of numbers is the offset of the non-0 features used in the denominator.

Value

A vector of integer row offsets.

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 = "iqlr", verbose = FALSE)
Denom <- getDenom(x)

# to find the names of housekeeping genes used
getFeatureNames(x)[getDenom(x)]
```

`getDirichletInstances` *getDirichletInstances*

Description

Returns a list of the Monte Carlo Dirichlet instances created by the `aldex.clr` function.

Usage

```
getDirichletInstances(.object)
```

Arguments

`.object`	A `aldex.clr` object containing the Monte Carlo Dirichlet instances derived from estimating.

Details

Returns a list of the raw Monte Carlo Dirichlet instances created by the `aldex.clr` function. These are probability estimates.

Value

A list of data frames.

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)
monteCarloDirInstances <- getDirichletInstances(x)
```

---

getDirichletReplicate *getDirichletReplicate*

Description

Returns the raw per-sample Monte Carlo Dirichlet replicates generated from analysis, for an aldex.clr object.

Usage

```
getDirichletReplicate(.object,i)
```

Arguments

.object	A aldex.clr object containing the Monte Carlo Dirichlet replicates 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 sample replicate.

Details

Returns the raw per-sample Monte Carlo Dirichlet replicates. These are estimated probabilities.

Value

A numeric matrix.

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)
DirichletReplicate <- getDirichletReplicate(x,1)
```

`getDirichletSample` *getDirichletSample*

Description

Returns a single Monte Carlo Dirichlet instance for all samples for an `aldex.clr` object.

Usage

```
getDirichletSample(.object,i)
```

Arguments

`.object`	A `aldex.clr` object containing the raw Monte Carlo Dirichlet 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 Monte-Carlo instance.

Details

Returns the designated Monte Carlo Dirichlet instance for all samples generated from analysis.

Value

A matrix representing the designated Monte Carlo Dirichlet instance for all samples.

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)
DirichletSample <- getDirichletSample(x,1)
```

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.

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.

Details

Returns the features from the first sample and first Monte-Carlo replicate as a vector, for an aldex.clr object. Used only for troubleshooting purposes.

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 a list of the log-ratio transformed Monte Carlo Dirichlet instances created by the `aldex.clr` function.

Usage

```
getMonteCarloInstances(.object)
```

Arguments

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

Details

Returns a list of the log-ratio transformed Monte Carlo Dirichlet instances created by the `aldex.clr` function.

Value

A list of data frames.

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 log-ratio transformed per-sample Monte Carlo Dirichlet replicates generated from analysis, for an `aldex.clr` object.

Usage

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

Arguments

`.object`	A `aldex.clr` object containing the Monte Carlo Dirichlet replicates 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 sample replicate.

Details

Returns the log-ratio transformed per-sample Monte Carlo Dirichlet replicates.

Value

A numeric matrix.

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)
monteCarloReplicate <- getMonteCarloReplicate(x,1)
```

getMonteCarloSample *getMonteCarloSample*

Description

Returns a single Monte Carlo Dirichlet instance for all samples for an aldex.clr object.

Usage

```
getMonteCarloSample(.object,i)
```

Arguments

.object	A aldex.clr object containing the log-ratio transformed Monte Carlo Dirichlet 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 Monte-Carlo instance.

Details

Returns the designated Monte Carlo Dirichlet instance for all samples generated from analysis.

Value

A matrix representing the designated log-ratio transformed Monte Carlo Dirichlet instance for all samples.

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)
monteCarloSample <- getMonteCarloSample(x,1)
```

getReads *getReads*

Description

Returns the count table used as input for analysis, for aldex.clr object. Note this count table has features that are 0 in all samples removed, and a uniform prior of 0.5 is applied.

Usage

```
getReads(.object)
```

Arguments

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

Details

Returns the count table.Note this count table has features that are 0 in all samples removed, and a uniform prior of 0.5 is applied.

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.

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.

Details

Returns the number of samples compared.

Value

A numeric representing the number of samples 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 non-0 features associated with the data, for an `aldex.clr` object.

Usage

```
numFeatures(.object)
```

Arguments

.object	A aldex.clr object.

Details

Returns the number of features associated with the data that are not 0 in all samples.

Value

A numeric representing the number of non-0 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 Dirichlet instances generated for analysis, for an aldex.clr object.

Usage

```
numMCInstances(.object)
```

Arguments

.object	A aldex.clr object.

Details

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

Value

A numeric representing the number of Monte Carle Dirichlet 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

```
data(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

`synth2` *Synthetic asymmetric dataset*

Description

This synthetic dataset contains 2 percent sparsity as 0 values asymmetrically distributed. It is used as a test dataset.

Usage

```
data(synth2)
```

Format

A dataframe of 1000 features and 16 samples. The first 8 samples contain 20 features set to 0, the last 8 samples contain counts.

Source

Gloor et al (2017) notes

Index