Package 'speckle'

November 12, 2026

Type Package

Title Statistical methods for analysing single cell RNA-seq data

Version 1.12.0

Date 2025-10-14

LazyData FALSE

Depends R (>= 4.2.0)

Imports limma, edgeR, SingleCellExperiment, Seurat, ggplot2, methods, stats, grDevices, graphics

VignetteBuilder knitr

Suggests BiocStyle, knitr, rmarkdown, statmod, CellBench, scater, patchwork, jsonlite, vdiffr, testthat (>= 3.0.0)

Description The speckle package contains functions for the analysis of single cell RNA-seq data. The speckle package currently contains functions to analyse differences in cell type proportions. There are also functions to estimate the parameters of the Beta distribution based on a given counts matrix, and a function to normalise a counts matrix to the median library size. There are plotting functions to visualise cell type proportions and the mean-variance relationship in cell type proportions and counts. As our research into specialised analyses of single cell data continues we anticipate that the package will be updated with new functions.

License GPL-3

biocViews SingleCell, RNASeq, Regression, GeneExpression

RoxygenNote 7.2.2

Encoding UTF-8

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit cced36d

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Belinda Phipson [aut, cre]

Maintainer Belinda Phipson <phipson.b@wehi.edu.au>

Contents

speckle-package . . . 2
.extractSCE . . . 3
.extractSeurat . . . 3
convertDataToList . . . 4
estimateBetaParam . . . 5
estimateBetaParamsFromCounts . . . 6
getTransformedProps . . . 7
ggplotColors . . . 9
normCounts . . . 10
pbmc_props . . . 11
plotCellTypeMeanVar . . . 12
plotCellTypeProps . . . 13
plotCellTypePropsMeanVar . . . 14
propeller . . . 15
propeller.anova . . . 18
propeller.ttest . . . 20
speckle_example_data . . . 22

Index **24**

`speckle-package` *speckle: Statistical methods for analysing single cell RNA-seq data*

Description

The speckle package contains functions for the analysis of single cell RNA-seq data. The speckle package currently contains functions to analyse differences in cell type proportions. There are also functions to estimate the parameters of the Beta distribution based on a given counts matrix, and a function to normalise a counts matrix to the median library size. There are plotting functions to visualise cell type proportions and the mean-variance relationship in cell type proportions and counts. As our research into specialised analyses of single cell data continues we anticipate that the package will be updated with new functions.

Author(s)

Maintainer: Belinda Phipson <phipson.b@wehi.edu.au>

.extractSCE *Extract metadata from* SingleCellExperiment *object*

Description

This is an accessor function that extracts cluster, sample and group information for each cell.

Usage

```
.extractSCE(x)
```

Arguments

x	object of class SingleCellExperiment

Value

a dataframe containing clusters, sample and group

Author(s)

Belinda Phipson

.extractSeurat *Extract metadata from* Seurat *object*

Description

This is an accessor function that extracts cluster, sample and group information for each cell.

Usage

```
.extractSeurat(x)
```

Arguments

x	object of class Seurat

Value

a dataframe containing clusters, sample and group

Author(s)

Belinda Phipson

convertDataToList *Convert counts or proportions matrix to list object for propeller*

Description

This function takes a matrix of counts or proportions, and returns a list object that is expected from the `propeller.ttest` and `propeller.anova` functions. This allows the `propeller` framework to be applied to any proportions data, not just single cell data.

Usage

```
convertDataToList(
  x,
  data.type = c("proportions", "counts"),
  transform = NULL,
  scale.fac = NULL
)
```

Arguments

`x`	a matrix of counts or proportions, where the columns correspond to samples and the rows correspond to cell types, or entities for which proportions are calculated.
`data.type`	a character scalar specifying whether the data matrix contains counts or proportions. Possible values include "proportions" or "counts". Defaults to "proportions".
`transform`	a character scalar specifying which transformation of the proportions to perform. Possible values include "asin" or "logit". Defaults to "logit".
`scale.fac`	the total number of cells N for each sample. Can be a scalar or a vector of the same length as the number of samples. If NULL, a default of 5000 cells per sample is assumed.

Value

outputs a list object with the following components

`Counts`	A matrix of cell type counts with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.
`TransformedProps`	A matrix of transformed cell type proportions with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.
`Proportions`	A matrix of cell type proportions with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.

Author(s)

Belinda Phipson

See Also

getTransformedProps propeller.ttest propeller.anova

Examples

```
library(speckle)
library(limma)
# Make up some data with two groups, two biological replicates in each
# group and three cell types
# True cell type proportions for 4 samples
props <- matrix(c(0.5,0.3,0.2,0.6,0.3,0.1,0.3,0.4,0.3,0.4,0.3,0.3),
ncol=4, nrow=3, byrow=FALSE)
rownames(props) <- c("C0","C1","C2")
colnames(props) <- paste("S",c(1,2,3,4),sep="")
# Total numbers of cells per sample
numcells <- c(1000,1500,900,1200)

# Get data into list object to use as input to propeller.ttest
propslist <- convertDataToList(props, data.type="proportions",
transform="asin",scale.fac=numcells)

# Run propeller.ttest to test for differences between 2 groups
# Assume s1 and s2 belong to group 1 and s3 and s4 belong to group 2
grp <- rep(c("A","B"), each=2)

design <- model.matrix(~0+grp)
design

# Compare Grp A to B
contrasts <- c(1,-1)

propeller.ttest(propslist, design=design, contrasts=contrasts,
robust=TRUE, trend=FALSE, sort=TRUE)
```

---

estimateBetaParam *Estimate the parameters of a Beta distribution*

Description

This function estimates the two parameters of the Beta distribution, alpha and beta, given a vector of proportions. It uses the method of moments to do this.

Usage

```
estimateBetaParam(x)
```

Arguments

x	a vector of proportions.

Value

a list object with the estimate of alpha in a and beta in b.

Author(s)

Belinda Phipson

Examples

```
# Generate proportions from a beta distribution
props <- rbeta(1000, shape1=2, shape2=10)
estimateBetaParam(props)
```

---

estimateBetaParamsFromCounts

Estimate parameters of a Beta distribution from counts

Description

This function estimates the two parameters of the Beta distribution, alpha and beta for each cell type. The input is a matrix of cell type counts, where the rows are the cell types/clusters and the columns are the samples.

Usage

```
estimateBetaParamsFromCounts(x)
```

Arguments

x	a matrix of counts

Details

This function is called from the plotting function plotCellTypeMeanVar in order to estimate the variance for the Beta-Binomial distribution for each cell type.

Value

outputs a list object with the following components

n	Normalised library size
alpha	a vector of alpha parameters for the Beta distribution for each cell type
beta	vector of beta parameters for the Beta distribution for each cell type
pi	Estimate of the true proportion for each cell type
dispersion	Dispersion estimates for each cell type
var	Variance estimates for each cell type

Author(s)

Belinda Phipson

Examples

```
data <- speckle_example_data()
x <- table(data$clusters, data$samples)
estimateBetaParamsFromCounts(x)
```

---

getTransformedProps *Calculates and transforms cell type proportions*

Description

Calculates cell types proportions based on clusters/cell types and sample information and performs a variance stabilising transformation on the proportions.

Usage

```
getTransformedProps(clusters = clusters, sample = sample, transform = NULL)
```

Arguments

clusters	a factor specifying the cluster or cell type for every cell.
sample	a factor specifying the biological replicate for every cell.
transform	a character scalar specifying which transformation of the proportions to perform. Possible values include "asin" or "logit". Defaults to "asin".

Details

This function is called by the propeller function and calculates cell type proportions and performs an arcsin-square root transformation.

Value

outputs a list object with the following components

Counts	A matrix of cell type counts with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.
TransformedProps	A matrix of transformed cell type proportions with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.
Proportions	A matrix of cell type proportions with the rows corresponding to the clusters/cell types and the columns corresponding to the biological replicates/samples.

Author(s)

Belinda Phipson

See Also

propeller

Examples

```
library(speckle)
library(ggplot2)
library(limma)

# Make up some data

# True cell type proportions for 4 samples
p_s1 <- c(0.5,0.3,0.2)
p_s2 <- c(0.6,0.3,0.1)
p_s3 <- c(0.3,0.4,0.3)
p_s4 <- c(0.4,0.3,0.3)

# Total numbers of cells per sample
numcells <- c(1000,1500,900,1200)

# Generate cell-level vector for sample info
biorep <- rep(c("s1","s2","s3","s4"),numcells)
length(biorep)

# Numbers of cells for each of 3 clusters per sample
n_s1 <- p_s1*numcells[1]
n_s2 <- p_s2*numcells[2]
n_s3 <- p_s3*numcells[3]
n_s4 <- p_s4*numcells[4]

cl_s1 <- rep(c("c0","c1","c2"),n_s1)
cl_s2 <- rep(c("c0","c1","c2"),n_s2)
cl_s3 <- rep(c("c0","c1","c2"),n_s3)
cl_s4 <- rep(c("c0","c1","c2"),n_s4)
```

```
# Generate cell-level vector for cluster info
clust <- c(cl_s1,cl_s2,cl_s3,cl_s4)
length(clust)

getTransformedProps(clusters = clust, sample = biorep)
```

---

ggplotColors *Output a vector of colours based on the ggplot colour scheme*

Description

This function takes as input the number of colours the user would like, and outputs a vector of colours in the ggplot colour scheme.

Usage

```
ggplotColors(g)
```

Arguments

g	the number of colours to be generated.

Value

a vector with the names of the colours.

Author(s)

Belinda Phipson

Examples

```
# Generate a palette of 6 colours
cols <- ggplotColors(6)
cols

# Generate some count data
y <- matrix(rnbinom(600, mu=100, size=1), ncol=6)

par(mfrow=c(1,1))
boxplot(y, col=cols)
```

normCounts *Normalise a counts matrix to the median library size*

Description

This function takes a DGEList object or matrix of counts and normalises the counts to the median library size. This puts the normalised counts on a similar scale to the original counts.

Usage

```
normCounts(x, log = FALSE, prior.count = 0.5, lib.size = NULL)
```

Arguments

x	a DGEList object or matrix of counts.
log	logical, indicates whether the output should be on the log2 scale or counts scale. Default is FALSE.
prior.count	The prior count to add if the data is log2 normalised. Default is a small count of 0.5.
lib.size	a vector of library sizes to be used during the normalisation step. Default is NULL and will be computed from the counts matrix.

Details

If the input is a DGEList object, the normalisation factors in norm.factors are taken into account in the normalisation. The prior counts are added proportionally to the library size

Value

a matrix of normalised counts

Author(s)

Belinda Phipson

Examples

```
# Simulate some data from a negative binomial distribution with mean equal
# to 100 and dispersion set to 1. Simulate 1000 genes and 6 samples.
y <- matrix(rnbinom(6000, mu = 100, size = 1), ncol = 6)

# Normalise the counts
norm.y <- normCounts(y)

# Return log2 normalised counts
lnorm.y <- normCounts(y, log=TRUE)

# Return log2 normalised counts with prior.count = 2
```

```
lnorm.y2 <- normCounts(y, log=TRUE, prior.count=2)

par(mfrow=c(1,2))
boxplot(norm.y, main="Normalised counts")
boxplot(lnorm.y, main="Log2-normalised counts")
```

---

pbmc_props *Cell type proportions from single cell PBMC data*

Description

This dataset is from a paper published in PNAS that looked at differences in immune functioning between young and old, male and female samples: \ Huang Z. et al. (2021) Effects of sex and aging on the immune cell landscape as assessed by single-cell transcriptomic analysis. Proc. Natl. Acad. Sci. USA, 118, e2023216118.

Usage

```
pbmc_props
```

Format

'pbmc_props' A list object with the following components:

proportions A data frame of cell type proportions, where the rows are cell types and the columns are the samples. There are 24 rows and 20 columns

sample_info A data frame with age and sex information for each sample

total_cells Numeric, the total number of cells profiled across all samples in the single cell experiment

Details

The cell type proportions were extracted from the Supplementary Material "Dataset_S02". The sample information was extracted from the sample names (Y=young, M=male, O=old, F=female). The total number of cells profiled across all samples is 174684, but the number of cells per sample is unknown.

Source

<https://www.pnas.org/doi/10.1073/pnas.2023216118>, <https://www.pnas.org/doi/suppl/10.1073/pnas.2023216118/suppl_

plotCellTypeMeanVar *Plot cell type counts means versus variances*

Description

This function returns a plot of the log10(mean) versus log10(variance) of the cell type counts. The function takes a matrix of cell type counts as input. The rows are the clusters/cell types and the columns are the samples.

Usage

```
plotCellTypeMeanVar(x)
```

Arguments

x	a matrix or table of counts

Details

The expected variance under a binomial distribution is shown in the solid line, and the points represent the observed variance for each cell type/row in the counts table. The expected variance under different model assumptions are shown in the different coloured lines.

The mean and variance for each cell type is calculated across all samples.

Value

a base R plot

Author(s)

Belinda Phipson

Examples

```
library(edgeR)
# Generate some data
# Total number of samples
nsamp <- 10
# True cell type proportions
p <- c(0.05, 0.15, 0.35, 0.45)

# Parameters for beta distribution
a <- 40
b <- a*(1-p)/p

# Sample total cell counts per sample from negative binomial distribution
numcells <- rnbinom(nsamp,size=20,mu=5000)
true.p <- matrix(c(rbeta(nsamp,a,b[1]),rbeta(nsamp,a,b[2]),
          rbeta(nsamp,a,b[3]),rbeta(nsamp,a,b[4])),byrow=TRUE, ncol=nsamp)
```

```
counts <- matrix(NA,ncol=nsamp, nrow=nrow(true.p))
rownames(counts) <- paste("c",0:(nrow(true.p)-1), sep="")
for(j in 1:length(p)){
    counts[j,] <- rbinom(nsamp, size=numcells, prob=true.p[j,])
}

plotCellTypeMeanVar(counts)
```

`plotCellTypeProps` *Plot cell type proportions for each sample*

Description

This is a plotting function that shows the cell type composition for each sample as a stacked barplot. The `plotCellTypeProps` returns a `ggplot2` object enabling the user to make style changes as required.

Usage

```
plotCellTypeProps(x = NULL, clusters = NULL, sample = NULL)
```

Arguments

`x`	object of class `SingleCellExperiment` or `Seurat`
`clusters`	a factor specifying the cluster or cell type for every cell. For `SingleCellExperiment` objects this should correspond to a column called `clusters` in the `colData` assay. For `Seurat` objects this will be extracted by a call to `Idents(x)`.
`sample`	a factor specifying the biological replicate for each cell. For `SingleCellExperiment` objects this should correspond to a column called `sample` in the `colData` assay and for `Seurat` objects this should correspond to `x$sample`.

Value

a ggplot2 object

Author(s)

Belinda Phipson

Examples

```
library(speckle)
library(ggplot2)
library(limma)

# Generate some fake data from a multinomial distribution
```

```
# Group A, 4 samples, 1000 cells in each sample
countsA <- rmultinom(4, size=1000, prob=c(0.1,0.3,0.6))
colnames(countsA) <- paste("s",1:4,sep="")

# Group B, 3 samples, 800 cells in each sample

countsB <- rmultinom(3, size=800, prob=c(0.2,0.05,0.75))
colnames(countsB) <- paste("s",5:7,sep="")
rownames(countsA) <- rownames(countsB) <- paste("c",0:2,sep="")

allcounts <- cbind(countsA, countsB)
sample <- c(rep(colnames(allcounts),allcounts[1,]),
          rep(colnames(allcounts),allcounts[2,]),
          rep(colnames(allcounts),allcounts[3,]))
clust <- rep(rownames(allcounts),rowSums(allcounts))

plotCellTypeProps(clusters=clust, sample=sample)
```

`plotCellTypePropsMeanVar` *Plot cell type proportions versus variances*

Description

This function returns a plot of the log10(proportion) versus log10(variance) given a matrix of cell type counts. The rows are the clusters/cell types and the columns are the samples.

Usage

```
plotCellTypePropsMeanVar(x)
```

Arguments

x	a matrix or table of counts

Details

The expected variance under a binomial distribution is shown in the solid line, and the points represent the observed variance for each cell type/row in the counts table. The blue line shows the empirical Bayes variance that is used in `propeller`.

The mean and variance for each cell type is calculated across all samples.

Value

a base R plot

Author(s)

Belinda Phipson

Examples

```
library(limma)
# Generate some data
# Total number of samples
nsamp <- 10
# True cell type proportions
p <- c(0.05, 0.15, 0.35, 0.45)

# Parameters for beta distribution
a <- 40
b <- a*(1-p)/p

# Sample total cell counts per sample from negative binomial distribution
numcells <- rnbinom(nsamp,size=20,mu=5000)
true.p <- matrix(c(rbeta(nsamp,a,b[1]),rbeta(nsamp,a,b[2]),
          rbeta(nsamp,a,b[3]),rbeta(nsamp,a,b[4])),byrow=TRUE, ncol=nsamp)

counts <- matrix(NA,ncol=nsamp, nrow=nrow(true.p))
rownames(counts) <- paste("c",0:(nrow(true.p)-1), sep="")
for(j in 1:length(p)){
    counts[j,] <- rbinom(nsamp, size=numcells, prob=true.p[j,])
}

plotCellTypePropsMeanVar(counts)
```

`propeller` *Finding statistically significant differences in cell type proportions*

Description

Calculates cell type proportions, performs a variance stabilising transformation on the proportions and determines whether the cell type proportions are statistically significant between different groups using linear modelling.

Usage

```
propeller(
  x = NULL,
  clusters = NULL,
  sample = NULL,
  group = NULL,
  trend = FALSE,
  robust = TRUE,
  transform = "logit"
)
```

Arguments

x	object of class SingleCellExperiment or Seurat
clusters	a factor specifying the cluster or cell type for every cell. For SingleCellExperiment objects this should correspond to a column called clusters in the colData assay. For Seurat objects this will be extracted by a call to Idents(x).
sample	a factor specifying the biological replicate for each cell. For SingleCellExperiment objects this should correspond to a column called sample in the colData assay and for Seurat objects this should correspond to x$sample.
group	a factor specifying the groups of interest for performing the differential proportions analysis. For SingleCellExperiment objects this should correspond to a column called group in the colData assay. For Seurat objects this should correspond to x$group.
trend	logical, if true fits a mean variance trend on the transformed proportions
robust	logical, if true performs robust empirical Bayes shrinkage of the variances
transform	a character scalar specifying which transformation of the proportions to perform. Possible values include "asin" or "logit". Defaults to "logit".

Details

This function will take a SingleCellExperiment or Seurat object and extract the group, sample and clusters cell information. The user can either state these factor vectors explicitly in the call to the propeller function, or internal functions will extract them from the relevants objects. The user must ensure that group and sample are columns in the metadata assays of the relevant objects (any combination of upper/lower case is acceptable). For Seurat objects the clusters are extracted using the Idents function. For SingleCellExperiment objects, clusters needs to be a column in the colData assay.

The propeller function calculates cell type proportions for each biological replicate, performs a variance stabilising transformation on the matrix of proportions and fits a linear model for each cell type or cluster using the limma framework. There are two options for the transformation: arcsin square root or logit. Propeller tests whether there is a difference in the cell type proportions between multiple groups. If there are only 2 groups, a t-test is used to calculate p-values, and if there are more than 2 groups, an F-test (ANOVA) is used. Cell type proportions of 1 or 0 are accommodated. Benjamini and Hochberg false discovery rates are calculated to account to multiple testing of cell types/clusters.

Value

produces a dataframe of results

Author(s)

Belinda Phipson

References

Smyth, G.K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, Volume **3**, Article 3.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series*, B, **57**, 289-300.

See Also

propeller.ttest propeller.anova lmFit, eBayes, getTransformedProps

Examples

```
library(speckle)
library(ggplot2)
library(limma)

# Make up some data
# True cell type proportions for 4 samples
p_s1 <- c(0.5,0.3,0.2)
p_s2 <- c(0.6,0.3,0.1)
p_s3 <- c(0.3,0.4,0.3)
p_s4 <- c(0.4,0.3,0.3)

# Total numbers of cells per sample
numcells <- c(1000,1500,900,1200)

# Generate cell-level vector for sample info
biorep <- rep(c("s1","s2","s3","s4"),numcells)
length(biorep)

# Numbers of cells for each of the 3 clusters per sample
n_s1 <- p_s1*numcells[1]
n_s2 <- p_s2*numcells[2]
n_s3 <- p_s3*numcells[3]
n_s4 <- p_s4*numcells[4]

# Assign cluster labels for 4 samples
cl_s1 <- rep(c("c0","c1","c2"),n_s1)
cl_s2 <- rep(c("c0","c1","c2"),n_s2)
cl_s3 <- rep(c("c0","c1","c2"),n_s3)
cl_s4 <- rep(c("c0","c1","c2"),n_s4)

# Generate cell-level vector for cluster info
clust <- c(cl_s1,cl_s2,cl_s3,cl_s4)
length(clust)

# Assume s1 and s2 belong to group 1 and s3 and s4 belong to group 2
grp <- rep(c("grp1","grp2"),c(sum(numcells[1:2]),sum(numcells[3:4])))

propeller(clusters = clust, sample = biorep, group = grp,
```

```
robust = FALSE, trend = FALSE, transform="asin")
```

`propeller.anova` *Performs F-tests for transformed cell type proportions*

Description

This function is called by `propeller` and performs F-tests between multiple experimental groups or conditions (> 2) on transformed cell type proportions.

Usage

```
propeller.anova(
  prop.list = prop.list,
  design = design,
  coef = coef,
  robust = robust,
  trend = trend,
  sort = sort
)
```

Arguments

`prop.list`	a list object containing two matrices: `TransformedProps` and `Proportions`
`design`	a design matrix with rows corresponding to samples and columns to coefficients to be estimated
`coef`	a vector specifying which the columns of the design matrix corresponding to the groups to test
`robust`	logical, should robust variance estimation be used. Defaults to TRUE.
`trend`	logical, should a trend between means and variances be accounted for. Defaults to FALSE.
`sort`	logical, should the output be sorted by P-value.

Details

In order to run this function, the user needs to run the `getTransformedProps` function first. The output from `getTransformedProps` is used as input. The `propeller.anova` function expects that the design matrix is not in the intercept format. This `coef` vector will identify the columns in the design matrix that correspond to the groups being tested. Note that additional confounding covariates can be accounted for as extra columns in the design matrix, but need to come after the group-specific columns.

The `propeller.anova` function uses the `limma` functions `lmFit` and `eBayes` to extract F statistics and p-values. This has the additional advantage that empirical Bayes shrinkage of the variances are performed.

Value

produces a dataframe of results

Author(s)

Belinda Phipson

See Also

propeller, getTransformedProps, lmFit, eBayes

Examples

```
library(speckle)
library(ggplot2)
library(limma)

# Make up some data

# True cell type proportions for 4 samples
p_s1 <- c(0.5,0.3,0.2)
p_s2 <- c(0.6,0.3,0.1)
p_s3 <- c(0.3,0.4,0.3)
p_s4 <- c(0.4,0.3,0.3)
p_s5 <- c(0.8,0.1,0.1)
p_s6 <- c(0.75,0.2,0.05)

# Total numbers of cells per sample
numcells <- c(1000,1500,900,1200,1000,800)

# Generate cell-level vector for sample info
biorep <- rep(c("s1","s2","s3","s4","s5","s6"),numcells)
length(biorep)

# Numbers of cells for each of 3 clusters per sample
n_s1 <- p_s1*numcells[1]
n_s2 <- p_s2*numcells[2]
n_s3 <- p_s3*numcells[3]
n_s4 <- p_s4*numcells[4]
n_s5 <- p_s5*numcells[5]
n_s6 <- p_s6*numcells[6]

cl_s1 <- rep(c("c0","c1","c2"),n_s1)
cl_s2 <- rep(c("c0","c1","c2"),n_s2)
cl_s3 <- rep(c("c0","c1","c2"),n_s3)
cl_s4 <- rep(c("c0","c1","c2"),n_s4)
cl_s5 <- rep(c("c0","c1","c2"),n_s5)
cl_s6 <- rep(c("c0","c1","c2"),n_s6)

# Generate cell-level vector for cluster info
clust <- c(cl_s1,cl_s2,cl_s3,cl_s4,cl_s5,cl_s6)
length(clust)
```

```
prop.list <- getTransformedProps(clusters = clust, sample = biorep)

# Assume s1 and s2 belong to group A, s3 and s4 belong to group B, s5 and
# s6 belong to group C
grp <- rep(c("A","B","C"), each=2)

# Make sure design matrix does not have an intercept term
design <- model.matrix(~0+grp)
design

propeller.anova(prop.list, design=design, coef=c(1,2,3), robust=TRUE,
trend=FALSE, sort=TRUE)
```

`propeller.ttest` *Performs t-tests of transformed cell type proportions*

Description

This function is called by `propeller` and performs t-tests between two experimental groups or conditions on the transformed cell type proportions.

Usage

```
propeller.ttest(
  prop.list = prop.list,
  design = design,
  contrasts = contrasts,
  robust = robust,
  trend = trend,
  sort = sort
)
```

Arguments

`prop.list`	a list object containing two matrices: `TransformedProps` and `Proportions`
`design`	a design matrix with rows corresponding to samples and columns to coefficients to be estimated
`contrasts`	a vector specifying which columns of the design matrix correspond to the two groups to test
`robust`	logical, should robust variance estimation be used. Defaults to TRUE.
`trend`	logical, should a trend between means and variances be accounted for. Defaults to FALSE.
`sort`	logical, should the output be sorted by P-value.

Details

In order to run this function, the user needs to run the `getTransformedProps` function first. The output from `getTransformedProps` is used as input. The `propeller.ttest` function expects that the design matrix is not in the intercept format and a contrast vector needs to be supplied. This contrast vector will identify the two groups to be tested. Note that additional confounding covariates can be accounted for as extra columns in the design matrix after the group-specific columns.

The `propeller.ttest` function uses the `limma` functions `lmFit`, `contrasts.fit` and `eBayes` which has the additional advantage that empirical Bayes shrinkage of the variances are performed.

Value

produces a dataframe of results

Author(s)

Belinda Phipson

See Also

`propeller`, `getTransformedProps`, `lmFit`, `contrasts.fit`, `eBayes`

Examples

```
library(speckle)
library(ggplot2)
library(limma)

# Make up some data

# True cell type proportions for 4 samples
p_s1 <- c(0.5,0.3,0.2)
p_s2 <- c(0.6,0.3,0.1)
p_s3 <- c(0.3,0.4,0.3)
p_s4 <- c(0.4,0.3,0.3)

# Total numbers of cells per sample
numcells <- c(1000,1500,900,1200)

# Generate cell-level vector for sample info
biorep <- rep(c("s1","s2","s3","s4"),numcells)
length(biorep)

# Numbers of cells for each of 3 clusters per sample
n_s1 <- p_s1*numcells[1]
n_s2 <- p_s2*numcells[2]
n_s3 <- p_s3*numcells[3]
n_s4 <- p_s4*numcells[4]

cl_s1 <- rep(c("c0","c1","c2"),n_s1)
cl_s2 <- rep(c("c0","c1","c2"),n_s2)
cl_s3 <- rep(c("c0","c1","c2"),n_s3)
```

```
cl_s4 <- rep(c("c0","c1","c2"),n_s4)

# Generate cell-level vector for cluster info
clust <- c(cl_s1,cl_s2,cl_s3,cl_s4)
length(clust)

prop.list <- getTransformedProps(clusters = clust, sample = biorep)

# Assume s1 and s2 belong to group 1 and s3 and s4 belong to group 2
grp <- rep(c("A","B"), each=2)

design <- model.matrix(~0+grp)
design

# Compare Grp A to B
contrasts <- c(1,-1)

propeller.ttest(prop.list, design=design, contrasts=contrasts, robust=TRUE,
trend=FALSE, sort=TRUE)

# Pretend additional sex variable exists and we want to control for it
# in the linear model
sex <- rep(c(0,1),2)
des.sex <- model.matrix(~0+grp+sex)
des.sex

# Compare Grp A to B
cont.sex <- c(1,-1,0)

propeller.ttest(prop.list, design=des.sex, contrasts=cont.sex, robust=TRUE,
trend=FALSE, sort=TRUE)
```

`speckle_example_data` *Generate example data*

Description

Generate example data

Usage

```
speckle_example_data()
```

Value

a dataframe containing cell-level information for sample, group and clusters

Examples

```
speckle_example_data()
```

Index

∗ **datasets**
pbmc_props, 11
∗ **internal**
speckle-package, 2
.extractSCE, 3
.extractSeurat, 3

contrasts.fit, *21*
convertDataToList, 4

eBayes, *17*, *19*, *21*
estimateBetaParam, 5
estimateBetaParamsFromCounts, 6

getTransformedProps, *5*, 7, *17*, *19*, *21*
ggplotColors, 9

lmFit, *17*, *19*, *21*

normCounts, 10

pbmc_props, 11
plotCellTypeMeanVar, 12
plotCellTypeProps, 13
plotCellTypePropsMeanVar, 14
propeller, *8*, 15, *19*, *21*
propeller.anova, *5*, *17*, 18
propeller.ttest, *5*, *17*, 20

speckle *(*speckle-package*)*, 2
speckle-package, 2
speckle_example_data, 22