Package 'tcgsaseq'

September 10, 2020

Type Package

Title Time-Course Gene Set Analysis for RNA-Seq Data

Version 2.0.5

Date 2020-09-10

Depends R (>= 3.0.2)

Imports CompQuadForm, ggplot2, graphics, GSA, KernSmooth, parallel, pbapply, stats, statmod, utils

Suggests limma, edgeR, DESeq2, S4Vectors, knitr, rmarkdown, testthat, covr

Description Analyze RNA-seq data with variance component score test accounting for data heteroscedasticity through precision weights. Perform both gene-wise and gene set analyses, and can deal with longitudinal data. Method is detailed in: Agniel & Hejblum (2017) <doi: 10.1093/biostatistics/kxx005>, Variance component score test for time-course gene set analysis of longitudinal RNA-seq data, Biostatistics, 18(4):589-604.

LazyData true

License GPL-2 | file LICENSE

BugReports https://github.com/denisagniel/tcgsaseq/issues

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author Denis Agniel [aut],
Boris P. Hejblum [aut, cre],
Marine Gauthier [aut]

Maintainer Boris P. Hejblum <boris.hejblum@u-bordeaux.fr>

Repository CRAN

Date/Publication 2020-09-10 16:40:02 UTC

R topics documented:

`baduel_5gs` *Small portion of RNA-seq data from plant physiology study.*

Description

A subsample of the RNA-seq data from Baduel *et al.* studying Arabidopsis Arenosa physiology.

Usage

```
data(baduel_5gs)
```

Format

3 objects

- `design`: a design matrix for the 48 measured samples, containing the following variables:
  - `SampleName` corresponding column names from `expr_norm_corr`
  - `Intercept` an intercept variable
  - `Population` a factor identifying the plant population
  - `Age_weeks` numeric age of the plant at sampling time (in weeks)
  - `Replicate` a purely technical variable as replicates are not from the same individual over weeks. Should not be used in analysis.
  - `Vernalized` a logical variable indicating whether the plant had undergone vernalization (exposition to cold and short day photoperiods)
  - `Vernalized` a binary variable indicating whether the plant belonged to the KA population
  - `AgeWeeks_Population` interaction variable between the `AgeWeeks` and `Population` variables
  - `AgeWeeks_Vernalized` interaction variable between the `AgeWeeks` and `Vernalized` variables

  - Vernalized_Population interaction variable between the Vernalized and Population variables
  - AgeWeeks_Vernalized_Population interaction variable between the AgeWeeks, Vernalized and Population variables
- baduel_gmt: a gmt object containing 5 gene sets of interest (see GSA.read.gmt)
- expr_norm_corr: a numeric matrix containing the normalized batch corrected expression for the 2454 genes included in either of the 5 gene sets of interests

Source

https://www.ncbi.nlm.nih.gov/bioproject/PRJNA312410/

References

Baduel P, Arnold B, Weisman CM, Hunter B & Bomblies K (2016). Habitat-Associated Life History and Stress-Tolerance Variation in Arabidopsis Arenosa. *Plant Physiology*, 171(1):437-51. 10.1104/pp.15.01875.

Agniel D & Hejblum BP (2017). Variance component score test for time-course gene set analysis of longitudinal RNA-seq data, *Biostatistics*, 18(4):589-604. 10.1093/biostatistics/kxx005. arXiv:1605.02351.

Examples

```
## Not run:
rm(list=ls())
data("baduel_5gs")

set.seed(54321)
KAvsTBG <- tcgsa_seq(y=log2(expr_norm_corr+1), x=apply(as.matrix(design[, c("Intercept",
  "Vernalized", "Age_weeks", "Vernalized_Population", "AgeWeeks_Population"), drop=FALSE]),
       2, as.numeric),
                     phi=as.matrix(design[, c("PopulationKA"), drop=FALSE]),
                     genesets=baduel_gmt$genesets[c(3,5)],
                     which_test = "permutation", which_weights = "loclin",
                     n_perm=1000, preprocessed = TRUE, doPlot = TRUE)

set.seed(54321)
Cold <- tcgsa_seq(y=log2(expr_norm_corr+1), x=apply(as.matrix(design[, c("Intercept",
   "Age_weeks", "PopulationKA", "AgeWeeks_Population"), drop=FALSE]), 2, as.numeric),
                 phi=as.matrix(design[, c("Vernalized", "Vernalized_Population")]),
                 genesets=baduel_gmt$genesets[c(3,5)],
                 which_test = "permutation", which_weights = "loclin",
                 n_perm=1000, preprocessed = TRUE, doPlot = TRUE)

## End(Not run)
```

dsFDR *Estimating the False Discovery Rate*

Description

This function uses the permutation plug-in method to estimate the FDR. It requires scaled-test statistics so that permutation are comparable from one test to another.

Usage

```
dsFDR(gene_scores_perm, gene_scores_obs, use_median = TRUE, doPlot = FALSE)
```

Arguments

gene_scores_perm	a numeric matrix of size G x n_perm containing the permuted gene-wise scores for G genes with n_perm permutations.
gene_scores_obs	a vector of length n containing the observed gene-wise scores.
use_median	a logical flag indicating whether the median should be used to estimate the true proportion of null features. If not, we use a range of quantiles of the permuted gene-wise scores and the true proportion of null features is extrapolated from the limit of a smoothed estimate using the natural cubic spline. Default is TRUE. See *Storey et al.* for details
doPlot	a logical flag indicating whether the plot of the natural cubic spline fit should be drawn. Default is FALSE. Ignored if use_median is TRUE.

Value

A vector of estimating discrete false discovery rates

References

J. Li and R. Tibshirani (2013). Finding consistent patterns: A nonparametric approach for identifying differential expression in RNA-seq data, *Statistical Methods in Medical Research*, 22(5): 519-536

Storey, J. D., & Tibshirani, R. (2003). Statistical significance for genome-wide studies. *Proceedings of the National Academy of Sciences*, 100(16), 9440-9445.

Examples

```
## Not run:
#rm(list=ls())
G <- 1000
nperm <- 500
G1 <- 0.3*G
G0 <- G-G1
```

```
gene_scores_perm <- matrix(rchisq(G*nperm, df=1), ncol=nperm, nrow=G)
gene_scores_obs <- c(rchisq(G1, df=1), rchisq(G0, df=1))

qvals <- dsFDR(gene_scores_perm, gene_scores_obs, use_median = FALSE, doPlot = TRUE)
summary(qvals)

qvals <- qvals[!is.na(qvals)]
eFDR_5pct <- sum(qvals[-(1:G1)]<0.05)/sum(qvals < 0.05)
eTDR_5pct <- sum(qvals[1:G1]<0.05)/sum(qvals < 0.05)
cat("FDR:", eFDR_5pct, " TDR:", eTDR_5pct, "\n")
plot(y = sapply(seq(0, 1, by=0.001), function(x){sum(qvals[-(1:G1)] < x)/sum(qvals < x)}),
    x = seq(0, 1, by=0.001),
    type = "l", xlab = "Nominal FDR level", ylab = "Empirical FDR", col = "red", lwd = 2,
    ylim = c(0,1))
abline(a = 0, b = 1, lty = 2)

res <- list()
G <- 1000
nperm <- 1000
G1 <- 0.3*G
G0 <- G-G1
for(i in 1:100){
cat(i, "/100\n", sep="")
gene_scores_obs <- c(rchisq(G1, df=1), rchisq(G0, df=1))
gene_scores_perm <- matrix(rchisq(G*nperm, df=1), ncol=nperm, nrow=G)
qvals <- dsFDR(gene_scores_perm, gene_scores_obs, use_median = TRUE, doPlot = FALSE)
res[[i]] <- sapply(seq(0, 1, by=0.01), function(x){sum(qvals < x)})
}

## End(Not run)
```

PBT_gmt *PBT gene sets related to kidney transplant*

Description

9 Pathogenesis Based Transcripts (PBT) gene sets specifically related to kidney transplant

Usage

```
data(PBT_gmt)
```

Format

a gmt object containing 9 gene sets specific to kidney transplant (see GSA.read.gmt)

Source

https://www.ualberta.ca/medicine/institutes-centres-groups/atagc/research/gene-lists.html

References

Halloran PF, De Freitas DG, Einecke G, *et al.*, The molecular phenotype of kidney transplants: Personal viewpoint, *Am J Transplant*, 10: 2215-2222, 2010. .

Sellares J, Reeve J, Loupy A, et al., Molecular diagnosis of antibody-mediated rejection in human kidney transplants, *Am J Transplant*, 13:971-983, 2013.

Broin PO, Hayde N, Bao Y, et al., A pathogenesis-based transcript signature in donor-specific antibody-positive kidney transplant patients with normal biopsies, *Genomics Data* 2: 357-60, 2014.

Examples

```
data("PBT_gmt")
PBT_gmt
```

---

perm_pe *Exact permutation p-values*

Description

Calculates exact p-values for permutation tests when permutations are randomly drawn with replacement. This implementation is based on

Usage

```
perm_pe(nperm_supobs, nperm_eff, total_possible_nperm)
```

Arguments

`nperm_supobs`	number of permutations that yielded test statistics at least as extreme as the observed data. Can be a vector or an array of values.
`nperm_eff`	number of permutations effectively computed.
`total_possible_nperm`	total number of permutations possible.

Author(s)

Belinda Phipson and Gordon Smyth (adapted by Boris Hejblum)

References

Phipson B, and Smyth GK (2010). Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, Volume 9, Issue 1, Article 39. http://www.statsci.org/smyth/pubs/PermPValuesPreprint.pdf

qAbundanceDist *Gene abundance proportion distribution of RNA-seq data.*

Description

An example of gene abundance proportion distribution function of RNA-seq data, generated from a real dataset. See supplementary material of Law *et al.*

Usage

```
data(qAbundanceDist)
```

Format

A function: `qAbundanceDist`.

Source

http://bioinf.wehi.edu.au/voom/

References

Law CW, Chen Y, Shi W & Smyth GK, voom: Precision weights unlock linear model analysis tools for RNA-seq read counts, *Genome Biology*, 15(2), R29, 2014.

Examples

```
## Not run:
# Get distribution function of abundance proportions
# This distribution was generated from a real dataset
#load(url("http://bioinf.wehi.edu.au/voom/qAbundanceDist.RData"))
data("qAbundanceDist")
curve(qAbundanceDist, from=0, to =0.99)

# Generate baseline proportions for desired number of genes
ngenes <- 10000
baselineprop <- qAbundanceDist( (1:ngenes)/(ngenes+1) )
baselineprop <- baselineprop/sum(baselineprop)

## End(Not run)
```

sp_weights *Non parametric local heteroscedasticity weights*

Description

Computes precision weights that account for heteroscedasticity in RNA-seq count data based on non-parametric local linear regression estimates.

Usage

```
sp_weights(
  y,
  x,
  phi,
  use_phi = TRUE,
  preprocessed = FALSE,
  doPlot = FALSE,
  gene_based = FALSE,
  bw = c("nrd", "ucv", "SJ", "nrd0", "bcv"),
 kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight",
    "tricube", "cosine", "optcosine"),
  exact = FALSE,
  transform = TRUE,
  verbose = TRUE,
  na.rm = FALSE
)
```

Arguments

y	a numeric matrix of size G x n containing the raw RNA-seq counts or preprocessed expression from n samples for G genes.
x	a numeric matrix of size n x p containing the model covariate(s) from n samples (design matrix).
phi	a numeric design matrix of size n x K containing the K variable(s) of interest(e.g. bases of time).
use_phi	a logical flag indicating whether conditional means should be conditioned on phi and on covariate(s) x, or on x alone. Default is TRUE in which case conditional means are estimated conditionally on both x and phi.
preprocessed	a logical flag indicating whether the expression data have already been preprocessed (e.g. log2 transformed). Default is FALSE, in which case y is assumed to contain raw counts and is normalized into log(counts) per million.
doPlot	a logical flag indicating whether the mean-variance plot should be drawn. Default is FALSE.
gene_based	a logical flag indicating whether to estimate weights at the gene-level. Default is FALSE, when weights will be estimated at the observation-level.

bw	a character string indicating the smoothing bandwidth selection method to use. See bandwidth for details. Possible values are "ucv", "SJ", "bcv", "nrd" or "nrd0". Default is "nrd".
kernel	a character string indicating which kernel should be used. Possibilities are "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "tricube", "cosine", "optcosine". Default is "gaussian" (NB: "tricube" kernel corresponds to the loess method).
exact	a logical flag indicating whether the non-parametric weights accounting for the mean-variance relationship should be computed exactly or extrapolated from the interpolation of local regression of the mean against the variance. Default is FALSE, which uses interpolation (faster).
transform	a logical flag indicating whether values should be transformed to uniform for the purpose of local linear smoothing. This may be helpful if tail observations are sparse and the specified bandwidth gives suboptimal performance there. Default is TRUE.
verbose	a logical flag indicating whether informative messages are printed during the computation. Default is TRUE.
na.rm	logical: should missing values (including NA and NaN) be omitted from the calculations? Default is FALSE.

Value

a n x G matrix containing the computed precision weights.

See Also

bandwidth density

Examples

```
#rm(list = ls())
set.seed(123)

G <- 10000
n <- 12
p <- 2
y <- sapply(1:G, FUN = function(x){rnbinom(n = n, size = 0.07, mu = 200)})

x <- sapply(1:p, FUN = function(x){rnorm(n = n, mean = n, sd = 1)})
```

`tcgsaseq` *tcgsaseq: a package to perform Time-course Gene Set Analysis and General Gene-Wise Analysis of RNA-seq data*

Description

Analysis of RNA-seq data with variance component score test accounting for data heteroscedasticity through precision weights.

Details

Package:	tcgsaseq
Type:	Package
Version:	2.0.4
Date:	2020-09-10
License:	GPL-2

The two main functions of the `tcgsaseq` package are `varcompseq` and `tcgsa_seq`.

Author(s)

Boris P. Hejblum, Denis Agniel — Maintainer: Boris P. Hejblum

References

Agniel D & Hejblum BP (2017). Variance component score test for time-course gene set analysis of longitudinal RNA-seq data, *Biostatistics*, 18(4):589-604. 10.1093/biostatistics/kxx005. arXiv:1605.02351.

`tcgsa_seq` *Time-course Gene Set Analysis*

Description

Wrapper function for performing gene set analysis of (potentially longitudinal) RNA-seq data

Usage

```
tcgsa_seq(
  y,
  x,
  phi,
  weights_phi_condi = TRUE,
  genesets,
```

```
  indiv = NULL,
  Sigma_xi = diag(ncol(phi)),
  which_test = c("permutation", "asymptotic"),
  which_weights = c("loclin", "voom", "none"),
  n_perm = 1000,
  progressbar = TRUE,
  parallel_comp = TRUE,
  nb_cores = parallel::detectCores() - 1,
  preprocessed = FALSE,
  doPlot = TRUE,
  gene_based_weights = TRUE,
  bw = "nrd",
  kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight",
    "tricube", "cosine", "optcosine"),
  exact = FALSE,
  transform = TRUE,
  padjust_methods = c("BH", "BY", "holm", "hochberg", "hommel", "bonferroni"),
  lowess_span = 0.5,
  R = NULL,
  homogen_traj = FALSE,
  na.rm_tcgsaseq = TRUE,
  verbose = TRUE
)
```

Arguments

y	a numeric matrix of size G x n containing the raw RNA-seq counts or preprocessed expressions from n samples for G genes.
x	a numeric matrix of size n x p containing the model covariates from n samples (design matrix). Usually, its first column is the intercept (full of 1s).
phi	a numeric design matrix of size n x K containing the K variables to be tested
weights_phi_condi	a logical flag indicating whether heteroscedasticity weights computation should be conditional on both the variable(s) to be tested phi and on covariate(s) x, or on x alone. #'Default is TRUE in which case conditional means are estimated conditionally on both x and phi.
genesets	either a vector of index or subscripts that defines which rows of y constitute the investigated gene set (when only 1 gene set is being tested). Can also be a list of index (or rownames of y) when several gene sets are tested at once, such as the first element of a gmt object. If NULL, then gene-wise p-values are returned.
indiv	a vector of length n containing the information for attributing each sample to one of the studied individuals. Coerced to be a factor. Default is NULL in which case each sample is considered as coming from independent subjects.
Sigma_xi	a matrix of size K x K containing the covariance matrix of the K random effects. Only used if homogen_traj is FALSE. Default assume diagonal correlation matrix, i.e. independence of random effects.

which_test — a character string indicating which method to use to approximate the variance component score test, either "permutation" or "asymptotic". Default is "permutation".

which_weights — a character string indicating which method to use to estimate the mean-variance relationship weights. Possibilities are "loclin", "voom" or "none" (in which case no weighting is performed). Default is "loclin". See sp_weights and voom_weights for details.

n_perm — the number of perturbations. Default is 1000.

progressbar — logical indicating whether a progress bar should be displayed when computing permutations (only in interactive mode).

parallel_comp — a logical flag indicating whether parallel computation should be enabled. Only Linux and MacOS are supported, this is ignored on Windows. Default is TRUE.

nb_cores — an integer indicating the number of cores to be used when parallel_comp is TRUE. Only Linux and MacOS are supported, this is ignored on Windows. Default is parallel::detectCores() -1.

preprocessed — a logical flag indicating whether the expression data have already been preprocessed (e.g. log2 transformed). Default is FALSE, in which case y is assumed to contain raw counts and is normalized into log(counts) per million.

doPlot — a logical flag indicating whether the mean-variance plot should be drawn. Default is FALSE.

gene_based_weights — a logical flag used for "loclin" weights, indicating whether to estimate weights at the gene-level, or rather at the observation-level. Default is TRUE, and weights are then estimated at the gene-level.

bw — a character string indicating the smoothing bandwidth selection method to use. See bandwidth for details. Possible values are "ucv", "SJ", "bcv", "nrd" or "nrd0"

kernel — a character string indicating which kernel should be used. Possibilities are "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "tricube", "cosine", "optcosine". Default is "gaussian" (NB: "tricube" kernel corresponds to the loess method).

exact — a logical flag indicating whether the non-parametric weights accounting for the mean-variance relationship should be computed exactly or extrapolated from the interpolation of local regression of the mean against the variance. Default is FALSE, which uses interpolation (faster computation).

transform — a logical flag used for "loclin" weights, indicating whether values should be transformed to uniform for the purpose of local linear smoothing. This may be helpful if tail observations are sparse and the specified bandwidth gives suboptimal performance there. Default is TRUE.

padjust_methods — multiple testing correction method used if genesets is a list. Default is "BH", i.e. Benjamini-Hochberg procedure for controlling the FDR. Other possibilities are: "holm", "hochberg", "hommel", "bonferroni" or "BY" (for Benjamini-Yekutieli procedure).

lowess_span — smoother span for the lowess function, between 0 and 1. This gives the proportion of points in the plot which influence the smooth at each value. Larger

	values give more smoothness. Only used if `which_weights` is `"voom"`. Default is `0.5`.
`R`	library.size (optional, important to provide if `preprocessed = TRUE`). Default is `NULL`
`homogen_traj`	a logical flag indicating whether trajectories should be considered homogeneous. Default is `FALSE` in which case trajectories are not only tested for trend, but also for heterogeneity.
`na.rm_tcgsaseq`	logical: should missing values in `y` (including `NA` and `NaN`) be omitted from the calculations? Default is `TRUE`.
`verbose`	logical: should informative messages be printed during the computation? Default is `TRUE`.

Value

A list with the following elements:

- `which_test`: a character string carrying forward the value of the '`which_test`' argument indicating which test was perform (either "asymptotic" or "permutation").
- `preprocessed`: a logical flag carrying forward the value of the '`preprocessed`' argument indicating whether the expression data were already preprocessed, or were provided as raw counts and transformed into log-counts per million.
- `n_perm`: an integer carrying forward the value of the '`n_perm`' argument indicating the number of perturbations performed (`NA` if asymptotic test was performed).
- `genesets`: carrying forward the value of the '`genesets`' argument defining the gene sets of interest (`NULL` for gene-wise testing).
- `pval`: computed p-values. A `data.frame` with one raw for each each gene set, or for each gene if `genesets` argument is `NULL`, and with 2 columns: the first one '`rawPval`' contains the raw p-values, the second one contains the FDR adjusted p-values (according to the '`padjust_methods`' argument) and is named '`adjPval`'.

References

Agniel D & Hejblum BP (2017). Variance component score test for time-course gene set analysis of longitudinal RNA-seq data, *Biostatistics*, 18(4):589-604. 10.1093/biostatistics/kxx005. arXiv:1605.02351.

Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2), R29.

See Also

`sp_weights` `vc_test_perm` `vc_test_asym` `p.adjust`

Examples

```
#rm(list=ls())
nsims <- 2 #100
res_quant <- list()
for(i in 1:2){
```

```
n <- 2000#0
nr <- 3
r <- nr*20#4*nr#100*nr
t <- matrix(rep(1:nr), r/nr, ncol=1, nrow=r)
sigma <- 0.4
b0 <- 1

#under the null:
b1 <- 0

y.tilde <- b0 + b1*t + rnorm(r, sd = sigma)
y <- t(matrix(rnorm(n*r, sd = sqrt(sigma*abs(y.tilde))), ncol=n, nrow=r) +
      matrix(rep(y.tilde, n), ncol=n, nrow=r))
x <- matrix(1, ncol=1, nrow=r)

#run test
res <- tcgsa_seq(y, x, phi=t, genesets=lapply(0:9, function(x){x*10+(1:10)}),
                Sigma_xi=matrix(1), indiv=rep(1:(r/nr), each=nr), which_test="asymptotic",
                        which_weights="none", preprocessed=TRUE)
res_genes <- tcgsa_seq(y, x, phi=cbind(t),#, rnorm(r)), #t^2
                       genesets=NULL,
                 Sigma_xi=diag(1), indiv=rep(1:(r/nr), each=nr), which_test="asymptotic",
                       which_weights="none", preprocessed=TRUE)
length(res_genes$pvals[, "rawPval"])
quantile(res_genes$pvals[, "rawPval"])
res_quant[[i]] <- res_genes$pvals[, "rawPval"]
}
#round(rowMeans(sapply(res_quant, quantile)), 3)
#plot(density(unlist(res_quant)))
#mean(unlist(res_quant)<0.05)

## Not run:
res_genes <- tcgsa_seq(y, x, phi=t, genesets=NULL,
               Sigma_xi=matrix(1), indiv=rep(1:(r/nr), each=nr), which_test="permutation",
               which_weights="none", preprocessed=TRUE, n_perm=1000, parallel_comp = FALSE)

mean(res_genes$pvals$rawPval < 0.05)
summary(res_genes$pvals$FDR)

## End(Not run)
```

---

varcompseq *Variance component testing for RNA-seq data analysis*

Description

Wrapper function for gene-by-gene association testing of RNA-seq data

Usage

```
varcompseq(
  exprmat,
  covariates,
  variables2test,
  sample_group = NULL,
  weights_var2test_condi = TRUE,
  cov_variables2test_eff = diag(ncol(variables2test)),
  which_test = c("permutation", "asymptotic"),
  which_weights = c("loclin", "voom", "none"),
  n_perm = 1000,
  progressbar = TRUE,
  parallel_comp = TRUE,
  nb_cores = parallel::detectCores() - 1,
  preprocessed = FALSE,
  doPlot = TRUE,
  gene_based_weights = FALSE,
  bw = "nrd",
 kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight",
    "tricube", "cosine", "optcosine"),
  exact = FALSE,
  transform = TRUE,
  padjust_methods = c("BH", "BY", "holm", "hochberg", "hommel", "bonferroni"),
  lowess_span = 0.5,
  na.rm_varcompseq = TRUE,
  homogen_traj = FALSE
)
```

Arguments

exprmat	a numeric matrix of size G x n containing the raw RNA-seq counts or preprocessed expressions from n samples for G genes.
covariates	a numeric matrix of size n x p containing the model covariates from n samples (design matrix). Usually, its first column is the intercept (full of 1s).
variables2test	a numeric design matrix of size n x K containing the K variables to be tested
sample_group	a vector of length n indicating whether the samples should be grouped (e.g. paired samples or longitudinal data). Coerced to be a factor. Default is NULL in which case no grouping is performed.
weights_var2test_condi	a logical flag indicating whether heteroscedasticity weights computation should be conditional on both the variables to be tested variables2test and on the covariates, or on covariates alone. Default is TRUE in which case conditional means are estimated conditionally on both variables2test and covariates.
cov_variables2test_eff	a matrix of size K x K containing the covariance matrix of the K random effects. Only used if homogen_traj is FALSE. Default assume diagonal correlation matrix, i.e. independence of random effects.

which_test — a character string indicating which method to use to approximate the variance component score test, either "permutation" or "asymptotic". Default is "permutation".

which_weights — a character string indicating which method to use to estimate the mean-variance relationship weights. Possibilities are "loclin", "voom" or "none" (in which case no weighting is performed). Default is "loclin". See sp_weights and voom_weights for details.

n_perm — the number of perturbations. Default is 1000.

progressbar — logical indicating whether a progress bar should be displayed when computing permutations (only in interactive mode).

parallel_comp — a logical flag indicating whether parallel computation should be enabled. Only Linux and MacOS are supported, this is ignored on Windows. Default is TRUE.

nb_cores — an integer indicating the number of cores to be used when parallel_comp is TRUE. Only Linux and MacOS are supported, this is ignored on Windows. Default is parallel::detectCores() -1.

preprocessed — a logical flag indicating whether the expression data have already been preprocessed (e.g. log2 transformed). Default is FALSE, in which case y is assumed to contain raw counts and is normalized into log(counts) per million.

doPlot — a logical flag indicating whether the mean-variance plot should be drawn. Default is FALSE.

gene_based_weights — a logical flag used for "loclin" weights, indicating whether to estimate weights at the gene-level, or rather at the observation-level. Default is FALSE, which is what it should be for gene-wise analysis.

bw — a character string indicating the smoothing bandwidth selection method to use. See bandwidth for details. Possible values are "ucv", "SJ", "bcv", "nrd" or "nrd0"

kernel — a character string indicating which kernel should be used. Possibilities are "gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "tricube", "cosine", "optcosine". Default is "gaussian" (NB: "tricube" kernel corresponds to the loess method).

exact — a logical flag indicating whether the non-parametric weights accounting for the mean-variance relationship should be computed exactly or extrapolated from the interpolation of local regression of the mean against the variance. Default is FALSE, which uses interpolation (faster computation).

transform — a logical flag used for "loclin" weights, indicating whether values should be transformed to uniform for the purpose of local linear smoothing. This may be helpful if tail observations are sparse and the specified bandwidth gives suboptimal performance there. Default is TRUE.

padjust_methods — multiple testing correction method used if genesets is a list. Default is "BH", i.e. Benjamini-Hochberg procedure for controlling the FDR. Other possibilities are: "holm", "hochberg", "hommel", "bonferroni" or "BY" (for Benjamini-Yekutieli procedure).

lowess_span — smoother span for the lowess function, between 0 and 1. This gives the proportion of points in the plot which influence the smooth at each value. Larger

values give more smoothness. Only used if which_weights is "voom". Default is 0.5.

na.rm_varcompseq
: logical: should missing values in y (including NA and NaN) be omitted from the calculations? Default is FALSE.

homogen_traj
: a logical flag indicating whether trajectories should be considered homogeneous. Default is FALSE in which case trajectories are not only tested for trend, but also for heterogeneity.

Value

A list with the following elements:

- which_test: a character string carrying forward the value of the 'which_test' argument indicating which test was perform (either "asymptotic" or "permutation").
- preprocessed: a logical flag carrying forward the value of the 'preprocessed' argument indicating whether the expression data were already preprocessed, or were provided as raw counts and transformed into log-counts per million.
- n_perm: an integer carrying forward the value of the 'n_perm' argument indicating the number of perturbations performed (NA if asymptotic test was performed).
- genesets: carrying forward the value of the 'genesets' argument defining the gene sets of interest (NULL for gene-wise testing).
- pval: computed p-values. A data.frame with one raw for each each gene set, or for each gene if genesets argument is NULL, and with 2 columns: the first one 'rawPval' contains the raw p-values, the second one contains the FDR adjusted p-values (according to the 'padjust_methods' argument) and is named 'adjPval'.

References

Agniel D & Hejblum BP (2017). Variance component score test for time-course gene set analysis of longitudinal RNA-seq data, *Biostatistics*, 18(4):589-604. 10.1093/biostatistics/kxx005. arXiv:1605.02351.

See Also

sp_weights vc_test_perm vc_test_asym p.adjust

Examples

```
#rm(list=ls())
nsims <- 2 #100
res <- numeric(nsims)
for(i in 1:nsims){
n <- 1000
nr=5
ni=50
r <- nr*ni
t <- matrix(rep(1:nr), ni, ncol=1, nrow=r)
sigma <- 0.5
b0 <- 1
```

```
#under the null:
b1 <- 0

y.tilde <- b0 + b1*t + rnorm(r, sd = sigma)
y <- t(matrix(rnorm(n*r, sd = sqrt(sigma*abs(y.tilde))), ncol=n, nrow=r) +
      matrix(rep(y.tilde, n), ncol=n, nrow=r))
x <- matrix(1, ncol=1, nrow=r)

#run test
res_genes <- varcompseq(exprmat=y, covariates=x, variables2test=t, sample_group=rep(1:ni, each=nr),
                  which_test="asymptotic",
                  which_weights="none", preprocessed=TRUE)
mean(res_genes$pvals[, "rawPval"]>0.05)
quantile(res_genes$pvals[, "rawPval"])
res[i] <- mean(res_genes$pvals[, "rawPval"]<0.05)
cat(i,"\n")
}
mean(res)

## Not run:
b0 <- 1
b1 <- 0
y.tilde <- b0 + b1*t + rnorm(r, sd = sigma)
y <- t(matrix(rnorm(n*r, sd = sqrt(sigma*abs(y.tilde))), ncol=n, nrow=r) +
      matrix(rep(y.tilde, n), ncol=n, nrow=r))
res_genes <- varcompseq(exprmat=y, covariates=x, variables2test=t, sample_group=rep(1:ni, each=nr),
                  which_weights="none", preprocessed=TRUE)
summary(res_genes$pvals)

## End(Not run)
```

`vc_test_asym` *Computes variance component test statistic for longitudinal*

Description

This function computes an approximation of the variance component test based on the asymptotic distribution of a mixture of χ^2s using Davies method from `davies`

Usage

```
vc_test_asym(
  y,
  x,
  indiv = rep(1, nrow(x)),
  phi,
  w,
  Sigma_xi = diag(ncol(phi)),
```

```
  genewise_pvals = FALSE,
  homogen_traj = FALSE,
  na.rm = FALSE
)
```

Arguments

y	a numeric matrix of dim g x n containing the raw or normalized RNA-seq counts for g genes from n samples.
x	a numeric design matrix of dim n x p containing the p covariates to be adjusted for
indiv	a vector of length n containing the information for attributing each sample to one of the studied individuals. Coerced to be a factor.
phi	a numeric design matrix of size n x K containing the K longitudinal variables to be tested (typically a vector of time points or functions of time)
w	a vector of length n containing the weights for the n samples, corresponding to the inverse of the diagonal of the estimated covariance matrix of y.
Sigma_xi	a matrix of size K x K containing the covariance matrix of the K random effects corresponding to phi.
genewise_pvals	a logical flag indicating whether gene-wise p-values should be returned. Default is FALSE in which case gene set p-value is computed and returned instead.
homogen_traj	a logical flag indicating whether trajectories should be considered homogeneous. Default is FALSE in which case trajectories are not only tested for trend, but also for heterogeneity.
na.rm	logical: should missing values (including NA and NaN) be omitted from the calculations? Default is FALSE.

Value

A list with the following elements when the set p-value is computed :

- set_score_obs: the approximation of the observed set score
- set_pval: the associated set p-value

or a list with the following elements when gene-wise p-values are computed:

- gene_scores_obs: vector of approximating the observed gene-wise scores
- gene_pvals: vector of associated gene-wise p-values

See Also

davies

Examples

```
#rm(list=ls())
set.seed(123)

##generate some fake data
########################
n <- 100
r <- 12
t <- matrix(rep(1:(r/4)), 4, ncol=1, nrow=r)
sigma <- 0.4
b0 <- 1

#under the null:
b1 <- 0
#under the alternative:
#b1 <- 0.5
y.tilde <- b0 + b1*t + rnorm(r, sd = sigma)
y <- t(matrix(rnorm(n*r, sd = sqrt(sigma*abs(y.tilde))), ncol=n, nrow=r) +
      matrix(rep(y.tilde, n), ncol=n, nrow=r))
x <- matrix(1, ncol=1, nrow=r)

#run test
asymTestRes <- vc_test_asym(y, x, phi=cbind(t, t^2), w=matrix(1, ncol=ncol(y), nrow=nrow(y)),
                            Sigma_xi=diag(2), indiv=1:r, genewise_pvals=TRUE)
plot(density(asymTestRes$gene_pvals))
quantile(asymTestRes$gene_pvals)
```

`vc_test_perm` *Permutation-based variance component test statistic*

Description

This function computes an approximation of the Variance Component test for a mixture of χ^2s using permutations. This is preferable to the asymptotic approximation for small sample sizes. We rely on exact p-values following Phipson and Smyth, 2010 (see References).

Usage

```
vc_test_perm(
  y,
  x,
  indiv = rep(1, nrow(x)),
  phi,
  w,
  Sigma_xi = diag(ncol(phi)),
  n_perm = 1000,
  progressbar = TRUE,
  parallel_comp = TRUE,
```

```
  nb_cores = parallel::detectCores() - 1,
  genewise_pvals = FALSE,
  homogen_traj = FALSE,
  na.rm = FALSE
)
```

Arguments

y	a numeric matrix of dim G x n containing the raw RNA-seq counts for G genes from n samples.
x	a numeric design matrix of dim n x p containing the p covariates to be adjusted for
indiv	a vector of length n containing the information for attributing each sample to one of the studied individuals. Coerced to be a factor.
phi	a numeric design matrix of size n x K containing the K variables to be tested
w	a vector of length n containing the weights for the n samples.
Sigma_xi	a matrix of size K x K containing the covariance matrix of the K random effects.
n_perm	the number of perturbations. Default is 1000.
progressbar	logical indicating whether a progress bar should be displayed when computing permutations (only in interactive mode).
parallel_comp	a logical flag indicating whether parallel computation should be enabled. Only Linux and MacOS are supported, this is ignored on Windows. Default is TRUE.
nb_cores	an integer indicating the number of cores to be used when parallel_comp is TRUE. Only Linux and MacOS are supported, this is ignored on Windows. Default is parallel::detectCores() -1.
genewise_pvals	a logical flag indicating whether gene-wise p-values should be returned. Default is FALSE in which case gene-set p-value is computed and returned instead.
homogen_traj	a logical flag indicating whether trajectories should be considered homogeneous. Default is FALSE in which case trajectories are not only tested for trend, but also for heterogeneity.
na.rm	logical: should missing values (including NA and NaN) be omitted from the calculations? Default is FALSE.

Value

A list with the following elements when the set p-value is computed :

- set_score_obs: the approximation of the observed set score
- set_pval: the associated set p-value

or a list with the following elements when gene-wise p-values are computed:

- gene_scores_obs: vector of approximating the observed gene-wise scores
- gene_pvals: vector of associated gene-wise p-values
- ds_fdr: vector of associated gene-wise discrete false discovery rates

References

Phipson B, and Smyth GK (2010). Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, Volume 9, Issue 1, Article 39. http://www.statsci.org/smyth/pubs/PermPValuesPreprint.pdf

See Also

davies

Examples

```
#rm(list=ls())
set.seed(123)

##generate some fake data
########################
n <- 100
r <- 12
t <- matrix(rep(1:3), 4, ncol=1, nrow=r)
sigma <- 0.4
b0 <- 1

#under the null:
b1 <- 0
#under the alternative:
b1 <- 0.5
y.tilde <- b0 + b1*t + rnorm(r, sd = sigma)
y <- t(matrix(rnorm(n*r, sd = sqrt(sigma*abs(y.tilde))), ncol=n, nrow=r) +
      matrix(rep(y.tilde, n), ncol=n, nrow=r))
x <- matrix(1, ncol=1, nrow=r)

#run test
permTestRes <- vc_test_perm(y, x, phi=t, w=matrix(1, ncol=ncol(y), nrow=nrow(y)),
                            indiv=rep(1:4, each=3), n_perm=50, #1000,
                            parallel_comp = FALSE)
permTestRes$set_pval
```

---

voom_weights *Precision weights accounting for heteroscedasticity in RNA-seq count data*

Description

Implementation of the procedure described in Law *et al.* for estimating precision weights from RNA-seq data.

Usage

```
voom_weights(
  y,
  x,
  preprocessed = FALSE,
  doPlot = FALSE,
  lowess_span = 0.5,
  R = NULL
)
```

Arguments

y	a matrix of size G x n containing the raw RNA-seq counts or preprocessed expressions from n samples for G genes.
x	a matrix of size n x p containing the model covariates from n samples (design matrix).
preprocessed	a logical flag indicating whether the expression data have already been preprocessed (e.g. log2 transformed). Default is FALSE, in which case y is assumed to contain raw counts and is normalized into log(counts) per million.
doPlot	a logical flag indicating whether the mean-variance plot should be drawn. Default is FALSE.
lowess_span	smoother span for the lowess function, between 0 and 1. This gives the proportion of points in the plot which influence the smooth at each value. Larger values give more smoothness. Default is 0.5.
R	library.size (optional, important to provide if preprocessed = TRUE). Default is NULL

Value

a vector of length n containing the computed precision weights

References

Law, C. W., Chen, Y., Shi, W., & Smyth, G. K. (2014). voom: Precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15(2), R29.

See Also

lowess approxfun voom

Examples

```
#rm(list=ls())
set.seed(123)

G <- 10000
n <- 12
p <- 2
```

```
y <- sapply(1:n, FUN=function(x){rnbinom(n=G, size=0.07, mu=200)})
x <- sapply(1:p, FUN=function(x){rnorm(n=n, mean=n, sd=1)})

my_w <-  voom_weights(y, x, doPlot=TRUE)
if (requireNamespace("limma", quietly = TRUE)) {
 w_voom <- limma::voom(counts=y, design=x, plot=TRUE) #slightly faster - same results
 all.equal(my_w, w_voom$weights)
}

## Not run:
microbenchmark::microbenchmark(limma::voom(counts=t(y), design=x, plot=FALSE),
                               voom_weights(x, y, doPlot=FALSE), times=30)
## End(Not run)
```

Index