Package ‘zenith’

May 30, 2024

Type Package

Title Gene set analysis following differential expression using linear
  (mixed) modeling with dream

Version 1.6.0

Date 2024-03-08

Description Zenith performs gene set analysis on the result of differential expression using linear
  (mixed) modeling with dream by considering the correlation between gene expression
  traits. This package implements the camera method from the limma package proposed by Wu and Smyth
  (2012). Zenith is a simple extension of camera to be compatible with linear mixed models implemented in
  variancePartition::dream().

VignetteBuilder knitr

License Artistic-2.0

Encoding UTF-8

URL https://DiseaseNeuroGenomics.github.io/zenith

BugReports https://github.com/DiseaseNeuroGenomics/zenith/issues

Suggests BiocStyle, BiocGenerics, knitr, pander, rmarkdown,
  tweedIEseqCountData, edgeR, kableExtra, RUnit

biocViews RNASeq, GeneExpression, GeneSetEnrichment,
  DifferentialExpression, BatchEffect, QualityControl,
  Regression, Epigenetics, FunctionalGenomics, Transcriptomics,
  Normalization, Preprocessing, Microarray, ImmunoOncology,
  Software

Depends R (>= 4.2.0), limma, methods

Imports variancePartition (>= 1.26.0), EnrichmentBrowser (>= 2.22.0),
  GSEABase (>= 1.54.0), msigdb (>= 7.5.1), Rfast, ggplot2,
  tidyverse, reshape2, progress, utils, Rdpack, stats

RdMacros Rdpack

RoxygenNote 7.2.3

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

git_branch RELEASE_3_19
.rankSumTestWithCorrelation

Two Sample Wilcoxon-Mann-Whitney Rank Sum Test Allowing For Correlation

Description

Same as limma::.rankSumTestWithCorrelation, but returns effect size.

Usage

.rankSumTestWithCorrelation(index, statistics, correlation = 0, df = Inf)

Arguments

index any index vector such that statistics[index] contains the values of the statistic for the test group.

statistics numeric vector giving values of the test statistic.

correlation numeric scalar, average correlation between cases in the test group. Cases in the second group are assumed independent of each other and other the first group.

df degrees of freedom which the correlation has been estimated.

Details

See limma::.rankSumTestWithCorrelation
**corInGeneSet**

**Value**

data.frame storing results of hypothesis test

---

**corInGeneSet**

Evaluate mean correlation between residuals in gene set

**Description**

Evaluate mean correlation between residuals in gene set based on results from dream

**Usage**

corInGeneSet(fit, idx, squareCorr = FALSE)

**Arguments**

- fit: result of differential expression with dream
- idx: indeces or rownames to extract
- squareCorr: compute the mean squared correlation instead

**Value**

list storing correlation and variance inflation factor

---

**get_GeneOntology**

Load Gene Ontology genesets

**Description**

Load Gene Ontology genesets

**Usage**

geneOntology(
    onto = c("BP", "MF", "CC"),
    to = "ENSEMBL",
    includeOffspring = TRUE,
    org = "hsa"
)
get_MSigDB

Load MSigDB genesets

**Arguments**

- **cat**: array of categories to load. Defaults to array of all MSigDB categories.
- **to**: convert gene names to this type using EnrichmentBrowser::idMap(). See EnrichmentBrowser::idTypes(org="hsa") for valid types.
- **includeOffspring**: if TRUE, follow the GO hierarchy down and include all genes in offspring sets for a given gene set.
- **org**: organism. human ('hsa'), mouse ('mmu'), etc.

**Details**

This function loads the GO gene sets using the packages EnrichmentBrowser and GO.db. It can take a minute to load because converting gene name type is slow.

**Value**

Gene sets stored as GeneSetCollection

**Examples**

```r
# load GO Biological Process
# gs = get_GeneOntology('BP')

# load all gene sets
# gs = get_GeneOntology()
```

---

get_MSigDB

Load MSigDB genesets

**Description**

Load MSigDB genesets

**Usage**

```r
get_MSigDB(
  cat = unique(msigdb_collections()$gs_cat),
  to = "ENSEMBL",
  org = "hsa"
)
```

**Arguments**

- **cat**: array of categories to load. Defaults to array of all MSigDB categories.
- **to**: convert gene names to this type using EnrichmentBrowser::idMap(). See EnrichmentBrowser::idTypes(org="hsa") for valid types.
- **org**: organism. human ('hsa'), mouse ('mmu'), etc.
**plotZenithResults**

**Details**

This function loads the MSigDB gene sets using the packages `EnrichmentBrowser` and `msigdbr`. It can take a minute to load because converting gene name type is slow.

**Value**

Gene sets stored as `GeneSetCollection`

**Examples**

```r
# load Hallmark gene sets
gs = get_MSigDB('H')

# load all gene sets
# gs = get_MSigDB()
```

---

**plotZenithResults**  
*Heatmap of zenith results using ggplot2*

**Description**

Heatmap of zenith results showing genesets that have the top and bottom t-statistics from each assay.

**Usage**

```r
plotZenithResults(
  df,  
nTop = 5,  
nBottom = 5,  
label.angle = 45,  
zmax = NULL,  
transpose = FALSE,  
sortByGeneset = TRUE
)
```

**Arguments**

- `df`  
  result data.frame from `zenith_gsa`
- `ntop`  
  number of gene sets with highest t-statistic to show
- `nbottom`  
  number of gene sets with lowest t-statistic to show
- `label.angle`  
  angle of x-axis label
- `zmax`  
  maximum of the color scales. If not specified, used range of the observed t-statistics
- `transpose`  
  transpose the axes of the plot
- `sortByGeneset`  
  use hierarchical clustering to sort gene sets. Default is `TRUE`
Value

Heatmap showing enrichment for gene sets and cell types

Examples

# Load packages
library(edgeR)
library(variancePartition)
library(tweeDEseqCountData)

# Load RNA-seq data from LCL's
data(pickrell)
geneCounts = exprs(pickrell.eset)
df_metadata = pData(pickrell.eset)

# Filter genes
# Note this is low coverage data, so just use as code example
dsgn = model.matrix(~ gender, df_metadata)
keep = filterByExpr(geneCounts, dsgn, min.count=5)

# Compute library size normalization
dge = DGEList(counts = geneCounts[keep,])
dge = calcNormFactors(dge)

# Estimate precision weights using voom
vobj = voomWithDreamWeights(dge, ~ gender, df_metadata)

# Apply dream analysis
fit = dream(vobj, ~ gender, df_metadata)
fit = eBayes(fit)

# Load Hallmark genes from MSigDB
# use gene 'SYMBOL', or 'ENSEMBL' id
# use get_GeneOntology() to load Gene Ontology
gs = get_MSigDB("H", to="ENSEMBL")

# Run zenith analysis
res.gsa = zenith_gsa(fit, gs, 'gendermale', progressbar=FALSE)

# Show top gene sets
head(res.gsa, 2)

# for each cell type select 3 genesets with largest t-statistic
# and 1 geneset with the lowest
# Grey boxes indicate the gene set could not be evaluated because
# to few genes were represented
plotZenithResults(res.gsa)
zenith  

**Gene set analysis following differential expression with dream**

**Description**

Perform gene set analysis on the result of differential expression using linear (mixed) modeling with variancePartition::dream by considering the correlation between gene expression traits. This package is a slight modification of limma::camera to 1) be compatible with dream, and 2) allow identification of gene sets with log fold changes with mixed sign.

**Usage**

```r
zenith(
  fit,  
  coef, 
  index, 
  use.ranks = FALSE, 
  allow.neg.cor = FALSE, 
  progressbar = TRUE, 
  inter.gene.cor = 0.01 
)
```

**Arguments**

- `fit`  
  result of differential expression with dream

- `coef`  
  coefficient to test using `topTable(fit, coef)`

- `index`  
  an index vector or a list of index vectors. Can be any vector such that `fit[index,]` selects the rows corresponding to the test set. The list can be made using `ids2indices`.

- `use.ranks`  
  do a rank-based test (`TRUE`) or a parametric test (`'FALSE'`)?

- `allow.neg.cor`  
  should reduced variance inflation factors be allowed for negative correlations?

- `progressbar`  
  if TRUE, show progress bar

- `inter.gene.cor`  
  if NA, estimate correlation from data. Otherwise, use specified value

**Details**

zenith gives the same results as `camera(..., inter.gene.cor=NA)` which estimates the correlation with each gene set.

For differential expression with dream using linear (mixed) models see Hoffman and Roussos (2020). For the original camera gene set test see Wu and Smyth (2012).
Value

- NGenes: number of genes in this set
- Correlation: mean correlation between expression of genes in this set
- delta: difference in mean t-statistic for genes in this set compared to genes not in this set
- se: standard error of delta
- p.less: p-value for hypothesis test of \( H_0: \delta < 0 \)
- p.greater: p-value for hypothesis test of \( H_0: \delta > 0 \)
- PValue: p-value for hypothesis test \( H_0: \delta \neq 0 \)
- Direction: direction of effect based on sign(delta)
- FDR: false discovery rate based on Benjamini-Hochberg method in p.adjust

References


Examples

```r
library(variancePartition)

# simulate meta-data
info <- data.frame(Age=c(20, 31, 52, 35, 43, 45), Group=c(0, 0, 0, 1, 1, 1))

# simulate expression data
y <- matrix(rnorm(1000*6), 1000, 6)
rownames(y) = paste0("gene", 1:1000)
colnames(y) = rownames(info)

# First set of 20 genes are genuinely differentially expressed
index1 <- 1:20
y[index1, 4:6] <- y[index1, 4:6] + 1

# Second set of 20 genes are not DE
index2 <- 21:40

# perform differential expression analysis with dream
fit = dream(y, ~ Age + Group, info)
fit = eBayes(fit)

# perform gene set analysis testing Age
res = zenith(fit, "Age", list(set1=index1, set2=index2))

head(res)
```
zenithPR_gsa

Gene set analysis using pre-computed test statistic

Description

Perform gene set analysis on the result of a pre-computed test statistic. Test whether statistics in a gene set are larger/smaller than statistics not in the set.

Usage

zenithPR_gsa(
    statistics,
    ids,
    geneSets,
    use.ranks = FALSE,
    n_genes_min = 10,
    progressbar = TRUE,
    inter.gene.cor = 0.01,
    coef.name = "zenithPR"
)

Arguments

- **statistics**: pre-computed test statistics
- **ids**: name of gene for each entry in statistics
- **geneSets**: GeneSetCollection
- **use.ranks**: do a rank-based test TRUE or a parametric test FALSE? default: FALSE
- **n_genes_min**: minimum number of genes in a geneset
- **progressbar**: if TRUE, show progress bar
- **inter.gene.cor**: correlation of test statistics with in gene set
- **coef.name**: name of column to store test statistic

Details

This is the same as zenith_gsa(), but uses pre-computed test statistics. Note that zenithPR_gsa() may give slightly different results for small samples sizes, if zenithPR_gsa() is fed t-statistics instead of z-statistics.

Value

- **NGenes**: number of genes in this set
- **Correlation**: mean correlation between expression of genes in this set
- **delta**: difference in mean t-statistic for genes in this set compared to genes not in this set
- **se**: standard error of delta
• **p.less**: p-value for hypothesis test of $H_0: \delta < 0$
• **p.greater**: p-value for hypothesis test of $H_0: \delta > 0$
• **PValue**: p-value for hypothesis test $H_0: \delta \neq 0$
• **Direction**: direction of effect based on $\text{sign}(\delta)$
• **FDR**: false discovery rate based on Benjamini-Hochberg method in `p.adjust`
• **coef.name**: name for pre-computed test statistics. Default: `zenithPR`

**See Also**

`zenith_gsa()`, `limma::cameraPR()`

**Examples**

```r
# Load packages
library(edgeR)
library(variancePartition)
library(tweeDEseqCountData)

# Load RNA-seq data from LCL's
data(pickrell)
geneCounts = exprs(pickrell.eset)
df_metadata = pData(pickrell.eset)

# Filter genes
# Note this is low coverage data, so just use as code example
dsgn = model.matrix(~ gender, df_metadata)
keep = filterByExpr(geneCounts, dsgn, min.count=5)

dge = DGEList(counts = geneCounts[keep,])
dge = calcNormFactors(dge)

# Estimate precision weights using voom
vobj = voomWithDreamWeights(dge, ~ gender, df_metadata)

# Apply dream analysis
fit = dream(vobj, ~ gender, df_metadata)
fit = eBayes(fit)

# Load Hallmark genes from MSigDB
# use gene 'SYMBOL', or 'ENSEMBL' id
# use `get_GeneOntology()` to load Gene Ontology
gs = get_MSigDB("H", to="ENSEMBL")

# Run zenithPR analysis with a test statistic for each gene
tab = topTable(fit, coef='gendermale', number=Inf)
res.gsa = zenithPR_gsa(tab$t, rownames(tab), gs)
```
Perform gene set analysis using zenith

Description

Perform a competitive gene set analysis accounting for correlation between genes.

Usage

```
zenith_gsa(
  fit, 
  geneSets, 
  coefs, 
  use.ranks = FALSE, 
  n_genes_min = 10, 
  inter.gene.cor = 0.01, 
  progressbar = TRUE, 
  ...
)
```

## S4 method for signature 'MArrayLM, GeneSetCollection'

```
zenith_gsa(
  fit, 
  geneSets, 
  coefs, 
  use.ranks = FALSE, 
  n_genes_min = 10, 
  inter.gene.cor = 0.01, 
  progressbar = TRUE, 
  ...
)
```

Arguments

- `fit`: results from `dream()`
- `geneSets`: `GeneSetCollection`
- `coefs`: list of coefficients to test using `topTable(fit, coef=coefs[[i]])`
- `use.ranks`: do a rank-based test `TRUE` or a parametric test `FALSE`? default: `FALSE`
- `n_genes_min`: minimum number of genes in a geneset
- `inter.gene.cor`: if NA, estimate correlation from data. Otherwise, use specified value
- `progressbar`: if `TRUE`, show progress bar
- `...`: other arguments
Details

This code adapts the widely used `camera()` analysis (Wu and Smyth 2012) in the `limma` package (Ritchie et al. 2015) to the case of linear (mixed) models used by `variancePartition::dream()`.

Value

data.frame of results for each gene set and cell type

References


See Also

`limma::camera`

Examples

```r
# Load packages
library(edgeR)
library(variancePartition)
library(tweeDEseqCountData)

# Load RNA-seq data from LCL's
data(pickrell)
geneCounts = exprs(pickrell.eset)
df_metadata = pData(pickrell.eset)

# Filter genes
# Note this is low coverage data, so just use as code example
dsgn = model.matrix(~ gender, df_metadata)
keep = filterByExpr(geneCounts, dsgn, min.count=5)

dge = DGEList(counts = geneCounts[keep,])
dge = calcNormFactors(dge)

# Estimate precision weights using voom
vobj = voomWithDreamWeights(dge, ~ gender, df_metadata)

# Apply dream analysis
fit = dream(vobj, ~ gender, df_metadata)
fit = eBayes(fit)

# Load Hallmark genes from MSigDB
# use gene 'SYMBOL', or 'ENSEMBL' id
```
# use get_GeneOntology() to load Gene Ontology
gs = get_MSigDB("H", to="ENSEMBL")

# Run zenith analysis
res.gsa = zenith_gsa(fit, gs, 'gendermale', progressbar=FALSE)

# Show top gene sets
head(res.gsa, 2)

# for each cell type select 3 genesets with largest t-statistic
# and 1 geneset with the lowest
# Grey boxes indicate the gene set could not be evaluated because
# to few genes were represented
plotZenithResults(res.gsa)
Index

.rankSumTestWithCorrelation, 2

corInGeneSet, 3

geneSet, 3

geneSet, 3

geneSet, 4

plotZenithResults, 5

zenith, 7

zenith_gsa, 5, 11

zenith_gsa_MArrayLM_GeneSetCollection ANY-method
(zenith_gsa), 11

zenith_gsa_MArrayLM_GeneSetCollection method
(zenith_gsa), 11

zenithPR_gsa, 9