Package 'DGEobj.utils'

July 21, 2025

Type Package

Title Differential Gene Expression (DGE) Analysis Utility Toolkit

Version 1.0.6

Description Provides a function toolkit to facilitate reproducible RNA-Seq Differential Gene Expression (DGE) analysis (Law (2015) <doi:10.12688/f1000research.9005.3>). The tools include both analysis work-flow and utility functions: mapping/unit conversion, count normalization, accounting for unknown covariates, and more. This is a complement/cohort to the 'DGEobj' package that provides a flexible container to manage and annotate Differential Gene Expression analysis results.

Depends R (>= 3.5.0)

License GPL-3

Language en-US

Encoding UTF-8

Imports assertthat, DGEobj (>= 1.0.3), dplyr, methods, stats, stringr

Suggests biomaRt, canvasXpress, conflicted, edgeR, glue, ggplot2, IHW, limma, knitr, qvalue, RNASeqPower, rmarkdown, statmod, sva, testthat, zFPKM

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author John Thompson [aut], Connie Brett [aut, cre], Isaac Neuhaus [aut], Ryan Thompson [aut]

Maintainer Connie Brett <connie@aggregate-genius.com>

Repository CRAN

Date/Publication 2022-05-19 23:50:02 UTC

Contents

`DGEobj.utils-package` *DGEobj.utils Package Overview*

Description

This package implements a set of utility functions to enable a limma/voom workflow capturing the results in DGEobj data structure. Aside from implementing a well developed and popular workflow in DGEobj format, the run* functions in the package illustrate how to wrap the individual processing steps in a workflow in functions that capture important metadata, processing parameters, and intermediate data items in the DGEobj data structure. This function- based approach to utilizing the DGEobj data structure insures consistency among a collection of projects processed by these methods and thus facilitates downstream automated meta-analysis.

More Information

```
browseVignettes(package = 'DGEobj.utils')
```

convertCounts *Convert count matrix to CPM, FPKM, FPK, or TPM*

Description

Takes a count matrix as input and converts to other desired units. Supported units include CPM, FPKM, FPK, and TPM. Output units can be logged and/or normalized. Calculations are performed using edgeR functions except for the conversion to TPM which is converted from FPKM using the formula provided by Harold Pimental.

Usage

```
convertCounts(
  countsMatrix,
  unit,
  geneLength,
  log = FALSE,
  normalize = "none",
  prior.count = NULL
)
```

Arguments

countsMatrix	A numeric matrix or dataframe of N genes x M Samples. All columns must be numeric.
unit	Required. One of CPM, FPKM, FPK or TPM.
geneLength	A vector or matrix of gene lengths. Required for length-normalized units (TPM, FPKM or FPK). If geneLength is a matrix, the rowMeans are calculated and used.
log	Default = FALSE. Set TRUE to return Log2 values. Employs edgeR functions which use an prior.count of 0.25 scaled by the library size.
normalize	Default = "none". Invokes edgeR's calcNormFactors() for normalization. Other options are: "TMM", "RLE", "upperquartile" (uses 75th percentile), "TMMwzp" and are case-insensitive.
prior.count	Average count to be added to each observation to avoid taking log of zero. Used only if log = TRUE. (Default dependent on method; 0 for TPM, 0.25 for CPM and FPKM) The prior.count is passed to edgeR cpm and rpkm functions and applies to logTPM, logCPM, and logFPKM calculations.

Details

geneLength is a vector where length(geneLength) == nrow(countsMatrix). If a RSEM effective-Length matrix is passed as input, rowMeans(effectiveLength) is used (because edgeR functions only accept a vector for effectiveLength).

Note that log2 values for CPM, TPM, and FPKM employ edgeR's prior.count handling to avoid divide by zero.

Value

A matrix in the new unit space

Examples

```
## Not run:
  # NOTE: Requires the edgeR package

  # Simulate some data
  counts <- trunc(matrix(runif(6000, min=0, max=2000), ncol=6))
  geneLength <- rowMeans(counts)

  # TMM normalized Log2FPKM
  Log2FPKM <- convertCounts(counts,
                            unit       = "fpkm",
                            geneLength = geneLength,
                            log        = TRUE,
                            normalize  = "tmm")

  # Non-normalized CPM (not logged)
  RawCPM <- convertCounts(counts,
                          unit      = "CPM",
                          log       = FALSE,
                          normalize = "none")

## End(Not run)
```

---

extractCol *Extract a named column from a series of df or matrices*

Description

Take a named list of dataframes where each dataframe has the same column names (e.g. a list of topTable dataframes), then extract the named column from each dataframe and return a matrix. The name of each dataframe is used as the column name in the resulting table.

Usage

```
extractCol(contrastList, colName, robust = TRUE)
```

Arguments

contrastList	A list of data.frames which all have the same colnames and same row counts. The dataframes in the list should have geneIDs as rownames.
colName	The name of the data column to extract into a matrix.

robust — Default = TRUE; TRUE forces use of a joins to merge columns which is more reliable and allows for combination of contrasts from different projects, but may not return items in the same row order as the source table. Setting to FALSE invokes a cbind() approach that requires all data.frames to have the same row count and row order but preserves the original row order.

Details

The common use case for this is to provide a list of topTable data frames and extract one column from each file to create a matrix of LogRatios or P-values (genes x contrasts)..

This should work as long as the requested column name is present in every dataframe. The default robust = TRUE should be used unless it has been verified that each dataframe in the input list has the same row count and row order.

Value

A dataframe containing the extracted columns

Examples

```
dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
TopTableList <- DGEobj::getType(dgeObj, type = "topTable")
MyPvalues    <- extractCol(TopTableList, colName = "P.Value")
head(MyPvalues)
```

---

lowIntFilter *Apply low intensity filters to a DGEobj*

Description

Takes a DGEobj as input and applies a combination of low intensity filters as specified by the user. Raw count, zFPKM, TPM, and/or FPK filters are supported. A gene must pass all active filters. Not setting a threshold argument inactivates that threshold.

Usage

```
lowIntFilter(
  dgeObj,
  countThreshold,
  zfpkmThreshold,
  fpkThreshold,
  tpmThreshold,
  sampleFraction = 0.5,
  geneLength,
  verbose = FALSE
)
```

Arguments

dgeObj	A DGEobj with RNA-Seq (counts) data (Required)
countThreshold	Genes below this threshold are removed (10 is recommended).
zfpkmThreshold	Genes below this threshold are removed. (-3.0 is recommended)
fpkThreshold	Genes below this threshold are removed. (5 is recommended)
tpmThreshold	Genes below this threshold are removed.
sampleFraction	The proportion of samples that must meet the thresholds (Default = 0.5). Range > 0 and <= 1.
geneLength	Vector of geneLengths for rows of dgeObj. Required for FPK and zFPKM filters, unless dgeObj is a DGEobj. If a DGEobj is supplied, geneLength is retrieved from the DGEobj, unless supplied by the geneLength argument.
verbose	Prints messages about the filtering process.

Value

Same class as input object with low intensity rows removed

Examples

```
## Not run:
  myDGEobj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
  dim(myDGEobj)

  # Simple count threshold in at least 3/4ths the samples
  myDGEobj <- lowIntFilter(myDGEobj,
                           countThreshold = 10,
                           sampleFraction = 0.5)
  dim(myDGEobj)

  # Count and FPK thresholds
  myDGEobj <- lowIntFilter(myDGEobj,
                           countThreshold = 10,
                           fpkThreshold = 5,
                           sampleFraction = 0.5)
  dim(myDGEobj)

## End(Not run)
```

---

rsqCalc *Calculate R-squared for each gene fit*

Description

Takes a Log2CPM numeric matrix and MArrayLM fit object from limma's lmFit() and calculates R-squared for each gene fit.

Usage

```
rsqCalc(normMatrix, fit)
```

Arguments

normMatrix	A normalized log2cpm matrix
fit	A MArrayLM object from limma's lmFit()

Value

A vector of R-squared values for each gene fit.

Examples

```
## Not run:
   # NOTE: Requires the edgeR package

   dgeObj    <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   log2cpm   <- convertCounts(dgeObj$counts, unit = "cpm", log=TRUE, normalize = "tmm")
   fitObject <- dgeObj$ReplicateGroupDesign_fit
   rsq       <- rsqCalc (log2cpm, fitObject)

## End(Not run)
```

---

runContrasts *Build contrast matrix and calculate contrast fits*

Description

Takes a DGEobj and a named list of contrasts to build. The DGEobj must contain a limma Fit object and associated designMatrix. Returns the DGEobj with contrast fit(s), contrast matrix, and topTable/topTreat dataframes added.

Usage

```
runContrasts(
  dgeObj,
  designMatrixName,
  contrastList,
  contrastSetName = NULL,
  runTopTable = TRUE,
  runTopTreat = FALSE,
  foldChangeThreshold = 1.5,
  runEBayes = TRUE,
  robust = TRUE,
  proportion = 0.01,
  qValue = FALSE,
```

```
  IHW = FALSE,
  verbose = FALSE
)
```

Arguments

dgeObj	A DGEobj object containing a Fit object and design matrix. (Required)
designMatrixName	The name of the design matrix within dgeObj to use for contrast analysis. (Required)
contrastList	A named list of contrasts. (Required)
contrastSetName	Name for the set of contrasts specified in contrastList. Defaults to "<fitName>_cf". Only needed to create 2 or more contrast sets from the same initial fit.
runTopTable	Runs topTable on the specified contrasts. (Default = TRUE)
runTopTreat	Runs topTreat on the specified contrasts. (Default = FALSE)
foldChangeThreshold	Only applies to topTreat (Default = 1.5)
runEBayes	Runs eBayes after contrast.fit (Default = TRUE)
robust	eBayes robust option. 'statmod' package must be installed to perform required calculations if robust = TRUE (Default = TRUE)
proportion	Proportion of genes expected to be differentially expressed. (used by eBayes) (Default = 0.01)
qValue	Set TRUE to include Q-values in topTable output. (Default = FALSE)
IHW	Set TRUE to add FDR values from the IHW package. (Default = FALSE)
verbose	Set TRUE to print some information during processing. (Default = FALSE)

Details

The contrastList is a named list. The values are composed of column names from the designMatrix of the DGEobj. Each contrast is named to give it a short, recognizable name to be used for display purposes.

Example contrastList

```
contrastList = list(
T1 = "treatment1 - control",
T2 = "treatment2 - control"
)
```

where treatment1, treatment2, and control are column names in the designMatrix.

The returned DGEobj list contains the new items:

- "contrastMatrix" a matrix
- "Fit.Contrasts" a Fit object
- "topTableList" a List of dataframes
- "topTreatList" a List of dataframes: if enabled

Value

The DGEobj with contrast matrix, fit and topTable/topTreat dataframes added.

Examples

```
## Not run:
  # NOTE: Requires the limma and statmod packages

   myDGEobj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))

   # Name the design matrix to be used (see inventory(myDGEobj))
   designMatrixName <- "ReplicateGroupDesign"

   # Define the named contrasts from design matrix column names
   contrastList  <- list(BDL_v_Sham = "ReplicateGroupBDL - ReplicateGroupSham",
                         EXT1024_v_BDL = "ReplicateGroupBDL_EXT.1024  - ReplicateGroupBDL",
                            Nint_v_BDL = "ReplicateGroupBDL_Nint - ReplicateGroupBDL",
                            Sora_v_BDL = "ReplicateGroupBDL_Sora - ReplicateGroupBDL")


   myDGEobj <- runContrasts(myDGEobj,
                            designMatrixName=designMatrixName,
                            contrastList=contrastList,
                            contrastSetName = "SecondContrastSet",
                            qValue = TRUE,
                            IHW = FALSE,
                            runTopTable = TRUE,
                            runTopTreat = TRUE,
                            foldChangeThreshold = 1.25)
   DGEobj::inventory(myDGEobj)

## End(Not run)
```

---

runEdgeRNorm *Run edgeR normalization on DGEobj*

Description

Takes a DGEobj and adds a normalized DGEList object representing the result of edgeR normalization (calcNormFactors).

Usage

```
runEdgeRNorm(
  dgeObj,
  normMethod = "TMM",
  itemName = "DGEList",
  includePlot = FALSE,
```

```
  plotLabels = NULL
)
```

Arguments

dgeObj	A DGEobj containing counts, design data, and gene annotation.
normMethod	One of "TMM", "RLE", "upperquartile", or "none". (Default = "TMM")
itemName	optional string represents the name of the new DGEList. It must be unique and not exist in the passed DGEobj (Default = "DGEList")
includePlot	Enable returning a "canvasXpress" or "ggplot" bar plot of the norm.factors produced (Default = FALSE). Possible values to pass: • **FALSE or NULL**: Disable plot • **TRUE or "canvasXpress"**: returns "canvasXpress" bar plot of the norm.factors produced. • **"ggplot"**: returns "ggplot" bar plot of the norm.factors produced.
plotLabels	Sample text labels for the plot. Length must equal the number of samples. (Default = NULL; sample number will be displayed)

Value

A DGEobj with a normalized DGEList added or a list containing the normalized DGEobj and a plot

Examples

```
## Not run:
   # NOTE: Requires the edgeR package

   myDGEobj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   myDGEobj <- DGEobj::resetDGEobj(myDGEobj)

   # Default TMM normalization
   myDGEobj <- runEdgeRNorm(myDGEobj)

   # With some options and plot output
   require(canvasXpress)
   myDGEobj <- DGEobj::resetDGEobj(myDGEobj)
   obj_plus_plot <- runEdgeRNorm(myDGEobj,
                                 normMethod = "upperquartile",
                                 includePlot = TRUE)
   myDGEobj <- obj_plus_plot[[1]]
   obj_plus_plot[[2]]

## End(Not run)
```

runIHW *Apply Independent Hypothesis Weighting (IHW) to a list of topTable dataframes*

Description

This is a wrapper around the independent hypothesis weighting package that takes a list of topTable data frames and applies Independent Hypothesis Weighting (IHW) to each topTable data frame in the list.

Usage

```
runIHW(contrastList, alpha = 0.1, FDRthreshold = 0.1, ...)
```

Arguments

contrastList	A named list of topTable dataframes.
alpha	Alpha should be the desired FDR level to interrogate (range 0-1; Default = 0.1)
FDRthreshold	Threshold value for the p-values of a dataframe (Default = 0.1)
...	other arguments are passed directly to the ihw function (see ?ihw)

Details

IHW is a method developed by N. Ignatiadis (http://dx.doi.org/10.1101/034330) to weight FDR values based on a covariate (AveExpr in this case).

The IHW FDR values are added as additional columns to the topTable data frames.

Function runIHW is normally called by runContrasts with argument IHW=T. It can also be used independently on a list of topTable dataframes. A list of topTable dataframes is conveniently retrieved with the DGEobj::getType function with the type argument set to "topTable".

This function expects the following columns are present in each data frame: P.value, adj.P.Val, AveExpr.

Note that it is impractical to run IHW on a list of genes less than ~5000. Operationally, IHW breaks the data into bins of 1500 genes for the analysis. If bins = 1, IHW converges on the BH FDR value. Instead, run IHW on the complete set of detected genes from topTable (not topTreat) results.

Value

A list of lists. The first element is the original contrastList with additional IHW columns added to each dataframe. The topTable dataframes will contain additional columns added by the IHW analysis and prefixed with "ihw." The second list element is the IHW result dataframe.

Examples

```
## Not run:
   # NOTE: Requires the IHW package

   dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   contrastList <- DGEobj::getType(dgeObj, type = "topTable")
   contrastList <- lapply(contrastList, dplyr::select,
                          -ihw.adj_pvalue,
                          -ihw.weight,
                          -ihw.weighted_pvalue)
   colnames(contrastList[[1]])
   contrastList <- runIHW(contrastList)

   # note new columns added
   colnames(contrastList[["contrasts"]][[1]])

## End(Not run)
```

`runPower` *Run a power analysis on counts and design matrix*

Description

Take a counts matrix and design matrix and return a power analysis using the RNASeqPower package. The counts matrix should be pre-filtered to remove non-expressed genes using an appropriate filtering criteria. The design matrix should describe the major sources of variation so the procedure can dial out those known effects for the power calculations.

Usage

```
runPower(
  countsMatrix,
  designMatrix,
  depth = c(10, 100, 1000),
  N = c(3, 6, 10, 20),
  FDR = c(0.05, 0.1),
  effectSize = c(1.2, 1.5, 2),
  includePlots = FALSE
)
```

Arguments

`countsMatrix`	A counts matrix or dataframe of numeric data. (Required)
`designMatrix`	A design matrix or dataframe of numeric data. (Required)
`depth`	A set of depth to use in the calculations. The default depths of c(10, 100, 1000) respectively represent a detection limit, below average expression, and median expression levels, expressed in read count units.

N	A set of N value to report power for. (Default = c(3, 6, 10, 20))
FDR	FDR thresholds to filter for for FDR vs. Power graph. (Default = c(0.05, 0.1))
effectSize	A set of fold change values to test. (Default = c(1.2, 1.5, 2))
includePlots	controls adding tow plots to the returned dataframe (Default = FALSE). The two plots are; a ROC curve (FDR vs. Power) and a plot of N vs. Power. Possible values to pass: • **FALSE or NULL**: Disable plots • **TRUE or "canvasXpress"**: returns "canvasXpress" plots. • **"ggplot"**: returns "ggplot" plots.

Details

Note, both 'RNASeqPower' and 'statmod' packages are required to run this function as follow:

- 'RNASeqPower' package is required to run power analysis on the given counts matrix and design matrix.
- 'statmod' package is required to run estimate dispersion calculations

If includePlots = FALSE (the default) or NULL, the function will return a tall skinny dataframe of power calculations for various requested combinations of N and significance thresholds.

If includePlots = TRUE, "canvasXpress" or "ggplot", a list is returned with an additional two "canvasXpress" or ggplots (plots) to the dataframe.

Value

a dataframe of power calculations or a list of the dataframe and defined plots as defined by the "includePlots" argument.

Examples

```
## Not run:
    # NOTE: Requires the RNASeqPower, statmod, and edgeR packages

    dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
    counts <- dgeObj$counts
    dm     <- DGEobj::getType(dgeObj, type = "designMatrix")[[1]]

    resultList <- runPower(countsMatrix = counts,
                           designMatrix = dm,
                           includePlots = TRUE)

    head(resultList[[1]]) # dataframe
    resultList[[2]]       # ROC Curves Plot
    resultList[[3]]       # N vs Power Plot

## End(Not run)
```

`runQvalue` *Calculate and add q-value and lFDR to dataframe*

Description

Takes an list of contrasts (e.g. topTable output or other dataframes that contain a p-value column). Adds a q-value and local FDR (lFDR) column to each dataframe.

Usage

```
runQvalue(contrastList, pvalField = "P.Value", ...)
```

Arguments

`contrastList`	A list of dataframes with a p-value column (all tables must use the same colname for the p-value column.)
`pvalField`	Define the colname of the p-value field in each dataframe. Not needed if using topTable output. (Optional. Default = "P.Value")
`...`	Optional arguments passed to the qvalue function (See ?qvalue)

Details

The qvalue package from John Storey at Princeton takes a list of p-values and calculates a q-value and a Local FDR (lFDR). The q-value is essentially a less conservative FDR estimate compared to the default Benjamini-Hochberg FDR produced by topTable analysis (i.e. will give more differential genes at the same nominal cutoff). The q-value function also produces a Local FDR (lFDR) column which answers a slightly different and possibly more relevant question. The BH FDR (adj.P.Val in topTable data.frames) and q-value gives the false discovery rate is for a list of genes at a given threshold. The local FDR attempts to answer the question: what is the probability that this particular gene is a false discovery? See doi:10.1007/9783642048982_248 for a brief introduction to FDRs and q-values.

Value

The input contrastList now containing q-value and lFDR columns in each dataframe.

Examples

```
## Not run:
   # NOTE: Requires the qvalue package

   dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   contrastList <- DGEobj::getType(dgeObj, type = "topTable")
   contrastList <- lapply(contrastList, dplyr::select,
                          -Qvalue,
                          -qvalue.lfdr)
   colnames(contrastList[[1]])
```

```
   contrastList <- runQvalue(contrastList)

   # note new columns added
   colnames(contrastList[[1]])

## End(Not run)
```

---

runSVA *Test for surrogate variables*

Description

Takes a DGEobj from runVoom and tests for surrogate variables. Adds a new design matrix to the DGEobj with the surrogate variable columns appended using cbind. runVoom should then be run again with the new design matrix to complete the analysis.

Usage

```
runSVA(dgeObj, designMatrixName, n.sv, method = "leek")
```

Arguments

dgeObj	A DGEobj with normalized counts and a designMatrix.
designMatrixName	The itemName of the design matrix in DGEobj.
n.sv	Optional; Use to override the default n.sv returned by num.sv for the number of SV to analyze.
method	Method passed to num.sv. Supports "leek" or "be". (Default = "leek")

Value

dgeObj containing an updated design table, the svobj and a new design matrix.

Examples

```
## Not run:
   # NOTE: Requires the sva package

    dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))

   ### Create a model based on surgery status, intentionally omitting the compound treatments
    dgeObj$design$SurgeryStatus <- "BDL"
    dgeObj$design$SurgeryStatus[dgeObj$design$ReplicateGroup == "Sham"] <- "Sham"
    formula <- '~ 0 + SurgeryStatus'
    designMatrix <- model.matrix (as.formula(formula), dgeObj$design)

    # Make sure the column names in the design matrix are legal
```

```
    colnames(designMatrix) <- make.names(colnames(designMatrix))

    #capture the formula as an attribute of the design matrix
    attr(designMatrix, "formula") <- formula

    #add the designMatrix to the DGEobj
    dgeObj <- DGEobj::addItem(dgeObj,
                              item      = designMatrix,
                              itemName  = "SurgeryStatusDesign",
                              itemType  = "designMatrix",
                              parent    = "design",
                              overwrite = TRUE)

    dgeObj <- runSVA(dgeObj, designMatrixName = "SurgeryStatusDesign")

## End(Not run)
```

---

runVoom *Run functions in a typical voom/lmFit workflow*

Description

In the default workflow, this function runs voomWithQualityWeights followed by lmFit and optionally eBayes. If the contrasts of interest are already represented in the model, enable eBayes. To use contrasts.fit downstream, run eBayes after that step instead. eBayes should always be run last.

Usage

```
runVoom(
  dgeObj,
  designMatrixName,
  dupCorBlock,
  runDupCorTwice = TRUE,
  qualityWeights = TRUE,
  var.design,
  mvPlot = TRUE,
  runEBayes = TRUE,
  robust = TRUE,
  proportion = 0.01
)
```

Arguments

dgeObj	A DGEobj containing a DGEList (e.g. from runEdgeRNorm) or counts (Required)
designMatrixName	Name of a design matrix within dgeObj. (Required)

dupCorBlock	Supply a block argument to trigger duplicateCorrelation. (Optional) Should be a vector the same length as ncol with values to indicate common group membership for duplicateCorrelation. Also, 'statmod' package must be installed to run duplicate correlation calculations.
runDupCorTwice	Default = TRUE. Gordon Smyth recommends running duplicateCorrelation twice. Set this to false to run duplicateCorrelation just once.
qualityWeights	Runs limma's voomWithQualityWeights() if set to TRUE (Default = TRUE). This should normally be set to TRUE.
var.design	Provide a design matrix (from model.matrix) to identify replicate groups (e.g. "~ ReplicateGroup") for quality weight determination. Causes quality weights to be determined on a group basis. If omitted limma's voomWithQualityWeights() treats each sample individually.
mvPlot	Enables the voom mean-variance plot. (Default = TRUE)
runEBayes	Runs eBayes after lmFit. (Default = TRUE) Note, 'statmod' package must be installed to run eBayes calculations.
robust	Used by eBayes. (Default = TRUE) Note, 'statmod' package must be installed to run eBayes calculations.
proportion	Proportion of genes expected to be differentially expressed (used by eBayes) (Default = 0.01) Modify the prior accordingly if the 1st pass analysis shows a significantly higher or lower proportion of genes regulated than the default.

Details

Input is minimally a DGEobj containing a DGEList (which is typically TMM-normalized) and a formula (character representation). If a DGEList is missing on the object the counts are used as-is. Other arguments can invoke the duplicateCorrelation method and modify use of quality weights.

Returns a DGEobj class object containing the VoomElist (voom output), and Fit object (lmFit output).

Quality weights should be enabled unless there is a good reason to turn them off. If all samples are equal quality, the weights will all approach 1.0 with no consequence on the results. More typically, some samples are better than others and using quality weights improves the overall result.

Use var.design if the quality weights are correlated with some factor in the experiment. This will cause the quality weights to be calculated as a group instead of individually.

Use duplicate correlation (dupCorBlock) when there are subjects that have been sampled more than once (e.g. before and after some treatment). This calculates a within- subject correlation and includes this in the model.

Value

A DGEobj now containing designMatrix, Elist, and fit object.

Examples

```
## Not run:
  # NOTE: Requires the limma package
```

```
dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
for (name in names(dgeObj)[11:length(dgeObj)]) {
    dgeObj <- DGEobj::rmItem(dgeObj, name)
}

dgeObj <- runVoom(dgeObj,
                  designMatrixName = "ReplicateGroupDesign",
                  mvPlot = TRUE)

# Note the Elist and fit objects have been added
DGEobj::inventory(dgeObj)

## End(Not run)
```

`summarizeSigCounts` *Summarize a contrast list*

Description

Takes a contrast list produced by runContrasts. Defaults are provided to specify columns to summarize and thresholds for each column, though they can be adjusted. A fold change threshold can optionally be specified. The function queries the topTable results and returns a dataframe with the summary results, but only includes gene counts that meet the specified conditions.

Usage

```
summarizeSigCounts(
  contrastList,
  columns = c("P.Value", "adj.P.Val", "Qvalue", "qvalue.lfdr", "ihw.adj_pvalue"),
  sigThresholds = c(0.01, 0.05, 0.05, 0.05, 0.05),
  fcThreshold = 0
)
```

Arguments

`contrastList`	A list of topTable dataframes.
`columns`	Vector of column names to summarize from topTable dataframes. Default = c("P.Value", "adj.P.Val", "Qvalue", "qvalue.lfdr", "ihw.adj_pvalue")
`sigThresholds`	Thresholds to use for each column specified in columns Must be same length at columns argument. Default = c(0.01, 0.05, 0.05, 0.05, 0.05)
`fcThreshold`	Fold-change threshold (absolute value, not logged.)

Details

Any specified column names that don't exist will be ignored. Normally the defaults cover all the p-value and FDR related columns. However, a fcThreshold can be added and the p-value/FDR thresholds can be modified using the fcThreshold and sigThresholds arguments, respectively.

Value

data.frame with one summary row per contrast.

Examples

```
dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
contrastList <- DGEobj::getType(dgeObj, type = "topTable")

#all defaults
sigSummary <- summarizeSigCounts(contrastList)

#add the fold-chage threshold
sigSummary <- summarizeSigCounts(contrastList, fcThreshold = 2)

#change the significance thresholds
sigSummary <- summarizeSigCounts(contrastList,
                                 sigThresholds = c(0.01, 0.1, 0.1, 0.1, 0.1))
```

---

topTable.merge *Merge specified topTable df cols*

Description

Take a named list of topTable dataframes and cbinds the requested columns from each file. To avoid column name conflicts the names are used as suffixes to the colnames. Although written for topTable data, this should work on any named list of dataframes where each member of the list has the same columns.

Usage

```
topTable.merge(
  contrastList,
  colNames = c("logFC", "AveExpr", "P.Value", "adj.P.Val"),
  digits = c(2, 2, 4, 3)
)
```

Arguments

contrastList	A named list of topTable data.frames which all have the same colnames and same row counts. The dataframes in the list should have rownames (geneIDs).
colNames	The list of column names of the data column to extract to a matrix (Default = c("logFC", "AveExpr", "P.Value", "adj.P.Val"))
digits	Number of decimal places for specified columns. Should be same length as colNames. (Default = c(2, 2, 4, 3)). If one value supplied, it is used for all columns.

Value

A matrix containing the extracted columns.

Examples

```
dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
contrastList <- DGEobj::getType(dgeObj, type = "topTable")

mergedData <- topTable.merge(contrastList)
colnames(mergedData)
```

---

tpm.direct *Convert countsMatrix and geneLength to TPM units*

Description

Takes a countsMatrix and geneLength as input and converts to TPM units using the equation from Harold Pimental.

Usage

```
tpm.direct(countsMatrix, geneLength, collapse = FALSE)
```

Arguments

countsMatrix	A numeric matrix of N genes x M samples. All columns must be numeric.
geneLength	Numeric matrix of gene lengths. Often the ExonLength item of a DGEobj.
collapse	Default = FALSE. TRUE or FALSE determines whether to use rowMeans on the geneLength matrix.

Details

The result should be the same as using convertCounts with normalize = 'tpm' and log = FALSE.

geneLength can be a vector (length == nrow(countsMatrix)) or a matrix (same dim as countsMatrix). The geneLength is used as is, or optionally collapsed to a vector by rowMeans.

Value

A matrix of TPM values

Examples

```
dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))

counts <- DGEobj::getItem(dgeObj, "counts")
exonLength <- dgeObj$geneData$ExonLength
tpm <- tpm.direct(counts, geneLength = exonLength)
```

`tpm.on.subset` *Calculate TPM for a subsetted DGEobj*

Description

Calculates TPM for a heavily subsetted DGEobj. The function will calculate TPM using the original data but returns a DGEobj with the subset.

Usage

```
tpm.on.subset(dgeObj, applyFilter = TRUE)
```

Arguments

`dgeObj`	A DGEobj data structure
`applyFilter`	Default = TRUE. If TRUE, reduces to the filtered gene list. FALSE returns all genes in the raw data.

Details

TPM should be calculated on a full dataset with only low signal genes removed. tpm.on.subset therefore allows calculation of TPM after heavy filtering of a DGEobj.

Internally, convertCounts uses edgeR's fpkm() to calculate FPKM and converts to TPM using the formula provided by [Harold Pimental](https://haroldpimentel.wordpress.com/2014/05/08/what-the-fpkm-a-review-rna-seq-expression-units/).

Value

A matrix of TPM values

Examples

```
## Not run:
   # NOTE: Requires the edgeR package

   dgeObj <- readRDS(system.file("exampleObj.RDS", package = "DGEobj"))
   tpm    <- tpm.on.subset(dgeObj)

## End(Not run)
```

Index