Package 'NanoTube'

November 12, 2025

Type Package

Title An Easy Pipeline for NanoString nCounter Data Analysis

Version 1.16.0

Date 2023-05-31

Depends R (>= 4.1), Biobase, ggplot2, limma

Imports fgsea, methods, reshape, stats, utils

Suggests grid, kableExtra, knitr, NanoStringDiff, pheatmap, plotly, rlang, rmarkdown, ruv, RUVSeq, shiny, testthat, xlsx

VignetteBuilder knitr

Description NanoTube includes functions for the processing, quality control, analysis, and visualization of NanoString nCounter data. Analysis functions include differential analysis and gene set analysis methods, as well as postprocessing steps to help understand the results. Additional functions are included to enable interoperability with other Bioconductor NanoString data analysis packages.

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.2.1

biocViews Software, GeneExpression, DifferentialExpression, QualityControl

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

git_branch RELEASE_3_22

git_last_commit b78a412

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-11-11

Author Caleb Class [cre, aut] (ORCID: <https://orcid.org/0000-0003-3130-3613>), Caiden Lukan [ctb]

Maintainer Caleb Class <cclass@butler.edu>

Contents

`deVolcano` *Draw volcano plot of differential expression results*

Description

Draw a volcano plot for results of a differential expression analysis by limma.

Usage

```
deVolcano(limmaResults, plotContrast = NULL, y.var = c("p.value", "q.value"))
```

Arguments

limmaResults	Result from runLimmaAnalysis.
plotContrast	Contrast to select for volcano plot. Should be one of the columns in the limma coefficients matrix (for example, a sample group that was compared against the base group, or one of the contrasts in the design matrix). If NULL (default), will plot the first non-Intercept column from the limma coefficients matrix.
y.var	The variable to plot for the y axis, either "p.value" or "q.value" (the false discovery adjusted p-value)

Value

A volcano plot using ggplot2

Examples

```
data(ExampleResults) # Results from runLimmaAnalysis

deVolcano(ExampleResults, plotContrast = "Autoimmune.retinopathy")
```

ExamplePathways *Example pathway database*

Description

A list object containing example gene sets from WikiPathways.

Usage

```
data(ExamplePathways)
```

Format

A list object with 30 vectors of gene symbols, for 30 pathways

---

ExampleResults *Example results from runLimmaAnalysis*

Description

Results of runLimmaAnalysis using the example data set GSE117751 (in extdata).

Usage

```
data(ExampleResults)
```

Format

An MArrayLM object from limma

`fgseaPostprocessing` *Postprocessing for GSEA analyses*

Description

Clusters GSEA results by leading edge genes, and writes reports showing gene expression profiles of these genes.

Usage

```
fgseaPostprocessing(
  genesetResults,
  leadingEdge,
  limmaResults,
  join.threshold = 0.5,
  ngroups = NULL,
  dist.method = "binary",
  reportDir
)
```

Arguments

`genesetResults`	Results from pathway analysis using limmaToFGSEA.
`leadingEdge`	Results from fgseaToLEdge
`limmaResults`	Results from runLimmaAnalysis
`join.threshold`	The threshold distance to join gene sets. Gene sets with a distance below this value will be joined to a single "cluster."
`ngroups`	The desired number of gene set groups. Either 'join.threshold' or 'ngroups' must be specified, 'ngroups' takes priority if both are specified.
`dist.method`	Method for distance calculation (see options for dist()). We recommend the 'binary' (also known as Jaccard) distance.
`reportDir`	Directory for the GSEA reports (each comparison will be a separate txt file). Directory will be created if it does not exist.

Value

A table of gene set analysis results, as well as reports showing differential expression of leading edge genes.

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways)

leadingEdge <- fgseaToLEdge(fgseaResults, cutoff.type = "padj", cutoff = 0.1)


fgseaPostprocessing(fgseaResults, leadingEdge,
```

```
                    limmaResults = ExampleResults,
                    join.threshold = 0.5,
                    reportDir = "GSEAresults")
```

`fgseaPostprocessingXLSX` *Postprocessing for GSEA analyses for Excel*

Description

Clusters GSEA results by leading edge genes, and writes reports showing gene expression profiles of these genes (to Excel).

Usage

```
fgseaPostprocessingXLSX(
  genesetResults,
  leadingEdge,
  limmaResults,
  join.threshold = 0.5,
  ngroups = NULL,
  dist.method = "binary",
  filename
)
```

Arguments

`genesetResults`	Results from pathway analysis using limmaToFGSEA.
`leadingEdge`	Results from fgseaToLEdge
`limmaResults`	Results from runLimmaAnalysis
`join.threshold`	The threshold distance to join gene sets. Gene sets with a distance below this value will be joined to a single "cluster."
`ngroups`	The desired number of gene set groups. Either 'join.threshold' or 'ngroups' must be specified, 'ngroups' takes priority if both are specified.
`dist.method`	Method for distance calculation (see options for dist()). We recommend the 'binary' (also known as Jaccard) distance.
`filename`	File name for the output Excel file.

Value

An Excel file where the first sheet summarizes the gene set analysis results. Subsequent sheets are reports showing differential expression statistics of leading edge genes.

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways)

leadingEdge <- fgseaToLEdge(fgseaResults, cutoff.type = "padj", cutoff = 0.1)


fgseaPostprocessingXLSX(fgseaResults, leadingEdge,
                    limmaResults = ExampleResults,
                    join.threshold = 0.5,
                    filename = "Results.xlsx")
```

`fgseaToLEdge` *Generate leading edge matrix from fgsea results.*

Description

Extract leading edge genes from gene sets identified in fgsea analysis. Gene sets may be filtered by significance or NES.

Usage

```
fgseaToLEdge(
  fgsea.res,
  cutoff.type = c("padj", "pval", "NES", "none"),
  cutoff = 0.05,
  nes.abs.cutoff = TRUE
)
```

Arguments

`fgsea.res`	Result from limmaToFGSEA
`cutoff.type`	Filter gene sets by adjusted p-value ('padj'), nominal p-value ('pval'), normalized enrichment score ('NES'), or include all gene sets ('none')
`cutoff`	Numeric cutoff for filtering (not used if cutoff.type == "none")
`nes.abs.cutoff`	If cutoff.type == "NES", should we use extreme positive and negative values (TRUE), or only filter in the positive or negative direction (FALSE). If TRUE, will select gene sets with abs(NES) > cutoff. If FALSE, will select gene sets with NES > cutoff (if cutoff >= 0) or NES < cutoff (if cutoff < 0)

Value

a list containing the leading edge matrix for each comparison

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways)

# Generate the leading edge for pathways with padj < 0.25
leadingEdge <- fgseaToLEdge(fgseaResults,
                            cutoff.type = "padj", cutoff = 0.25)

# Generate the leading edge for pathways with abs(NES) > 2
leadingEdge <- fgseaToLEdge(fgseaResults, cutoff.type = "NES",
                            cutoff = 2, nes.abs.cutoff = TRUE)
```

---

gm_mean *Calculate the geometric mean*

Description

Calculates the geometric mean of a numeric vector

Usage

```
gm_mean(x, na.rm = TRUE)
```

Arguments

x	A numeric vector
na.rm	Logical (default TRUE). Should NA values be ignored in this calculation? If FALSE, a vector containing NA values will return a geometric mean of NA.

Value

The geometric mean

Examples

```
gm_mean(c(1, 3, 5))
```

---

groupedGSEAtoStackedReport *Build a report from gene set enrichment results.*

Description

After clustering FGSEA results by gene set similarity, this function builds a report containing the individual gene expression profiles for genes contained in each gene set cluster.

Usage

```
groupedGSEAtoStackedReport(grouped.gsea, leadingEdge, de.fit, outputDir = NULL)
```

Arguments

grouped.gsea	Output from groupFGSEA()
leadingEdge	Leading edge analysis results used in groupFGSEA()
de.fit	Differential Expression results from Limma or NanoStringDiff
outputDir	Directory for output files. If NULL (default), will return the stacked report instead of writing to a file.

Value

A stacked report containing statistics and gene expression profiles for genes contained in each cluster

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways,
                             min.set = 5, rank.by = "t")
leadingEdge <- fgseaToLEdge(fgseaResults, cutoff.type = "padj", cutoff = 0.1)

fgseaGrouped <- groupFGSEA(fgseaResults$Autoimmune.retinopathy,
                           leadingEdge$Autoimmune.retinopathy,
                           join.threshold = 0.5,
                           dist.method = "binary")

results.AR <- groupedGSEAtoStackedReport(
              fgseaGrouped,
              leadingEdge = leadingEdge$Autoimmune.retinopathy,
              de.fit = ExampleResults)
```

---

groupFGSEA *Cluster gene set analysis results*

Description

Groups the pathway analysis results (using limmaToFGSEA or nsdiffToFGSEA) based on the enriched gene sets' leading edges. If the calculated distance metric is lower than the given threshold (i.e. the gene sets have highly overlapping leading edge genes), these gene sets will be joined to a single gene set "cluster." Or if 'ngroups' is specified, gene sets will be clustered by similarity into that number of groups.

Usage

```
groupFGSEA(
  gsea.res,
  l.edge,
  join.threshold = NULL,
  ngroups = NULL,
  dist.method = "binary",
  returns = c("signif", "all")
)
```

Arguments

gsea.res	Results from pathway analysis for a single comparison, using limmaToFSEA.
l.edge	Leading edge result from fgseaToLEdge.
join.threshold	The threshold distance to join gene sets. Gene sets with a distance below this value will be joined to a single "cluster."
ngroups	The desired number of gene set groups. Either 'join.threshold' or 'ngroups' must be specified, 'ngroups' takes priority if both are specified.
dist.method	Method for distance calculation (see options for dist()). We recommend the 'binary' (also known as Jaccard) distance.
returns	Either "signif" or "all". This argument defines whether only significantly enriched gene sets are included in the output table, or if the full results are included. Regardless of this selection, only significantly enriched gene sets are clustered.

Value

A data frame including the FGSEA results, plus two additional columns for the clustering results:

Cluster	The cluster that the gene set was assigned to. Gene sets in the same cluster have a distance below the join.threshold.
best	Whether the gene set is the most enriched (by p-value) in a given cluster.

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways,
                             min.set = 5, rank.by = "t")

leadingEdge <- fgseaToLEdge(fgseaResults, cutoff.type = "padj",
                            cutoff = 0.25)

# Group the results, and only returns those satisfying the cutoff specified
# in leadingEdge()
groupedResults <- groupFGSEA(fgseaResults$Autoimmune.retinopathy,
                             leadingEdge$Autoimmune.retinopathy,
                             join.threshold = 0.5,
                             returns = "signif")
```

---

limmaToFGSEA *Run gene set enrichment analysis using DE results.*

Description

Use the fgsea library to run gene set enrichment analysis from the Limma analysis results. Genes will be ranked by their log2 fold changes or t-statistics (specified using 'rank.by').

Usage

```
limmaToFGSEA(
  limmaResults,
  gene.sets,
  sourceDB = NULL,
  min.set = 1,
  rank.by = c("coefficients", "t"),
  skip.first = TRUE
)
```

Arguments

limmaResults	Result from runLimmaAnalysis.
gene.sets	Gene set file name, in .rds (list), .gmt, or .tab format; or a list object containing the gene sets. Gene names must be in the same form as in the ranked.list.
sourceDB	Source database to include (only if using a .tab-format geneset.file from CPDB).
min.set	Number of genes required to conduct analysis on a given gene set (default = 1). If fewer than this number of genes from limmaResults are included in a gene set, that gene set will be skipped for this analysis.
rank.by	Rank genes by log2 fold changes ('coefficients', default) or t-statistics ('t').
skip.first	Logical: Skip the first factor for gene set analysis? Frequently the first factor is the 'Intercept', which is generally uninteresting for GSEA (default TRUE).

Details

Limma returns matrices of coefficients and t statistics with columns for each column in the design matrix. This function will conduct a separate enrichment analysis on each column from the relevant matrix. Because the first column may be an "intercept" term, which is generally not relevant for enrichment analysis, the user may want to skip analysis for that term (using skip.first = TRUE, the default).

Value

A list containing data frames with the fgsea results for each comparison.

Examples

```
data("ExamplePathways")
data("ExampleResults") # Results from runLimmaAnalysis

# Use the default settings
fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways)

# Only include gene sets with at least 5 genes in the NanoString data set,
# and rank genes by their "t" statistics.
fgseaResults <- limmaToFGSEA(ExampleResults, gene.sets = ExamplePathways,
                             min.set = 5, rank.by = "t")
```

makeDiffExprFile *Make differential expression results file.*

Description

Make a data frame or text file containing coefficients, p-, and q-values from Limma differential expression analysis. If returns == "all", will also center the log-expression data on the median of base.group expression, and include the expression data in the output.

Usage

```
makeDiffExprFile(
  limmaResults,
  filename = NULL,
  returns = c("all", "stats"),
  skip.first = TRUE
)
```

Arguments

limmaResults	Result from runLimmaAnalysis
filename	The desired name for the output tab-delimited text file. If NULL (default) the resulting table will be returned as an R data frame.
returns	If "all" (default), will center the log-expression data on median of base.group expression and include the expression data in the output. If "stats", will only include the differential expression statistics.
skip.first	Logical: Skip the first factor for gene set analysis? Frequently the first factor is the 'Intercept', which is generally uninteresting for GSEA (default TRUE).

Value

A table of differential expression results

Examples

```
data("ExampleResults") # Results from runLimmaAnalysis

# Include expression data in the results table
deResults <- makeDiffExprFile(ExampleResults, returns = "all")

# Only include statistics, and save to a .txt file

makeDiffExprFile(ExampleResults, file = "DE.txt",
                 returns = "stats")
```

makeFGSEAmasterTable *Make master table of all GSEA results*

Description

This function clusters GSEA results by leading edge similarity, and then combines to a data frame or text file.

Usage

```
makeFGSEAmasterTable(
  genesetResults,
  leadingEdge,
  join.threshold = 0.5,
  ngroups = NULL,
  dist.method = "binary",
  filename = NULL
)
```

Arguments

genesetResults	Results from pathway analysis using limmaToFGSEA.
leadingEdge	Results from fgseaToLEdge
join.threshold	The threshold distance to join gene sets. Gene sets with a distance below this value will be joined to a single "cluster."
ngroups	The desired number of gene set groups. Either 'join.threshold' or 'ngroups' must be specified, 'ngroups' takes priority if both are specified.
dist.method	Method for distance calculation (see options for dist()). We recommend the 'binary' (also known as Jaccard) distance.
filename	File name for the output text file. If NULL (default), data will be returned as an R data frame.

Value

A table of GSEA results, clustered by similarity of leading edge.

makeNanoStringSetFromEset *Convert NanoString ExpressionSet to NanoStringSet*

Description

Convert ExpressionSet from processNanoStringData to a NanoStringSet for use with the NanoStringDiff package.

Usage

```
makeNanoStringSetFromEset(eset, designs = NULL)
```

Arguments

eset	NanoString data ExpressionSet, from processNanostringData
designs	Design matrix. If NULL, will look for "groups" column in pData(eset).

Value

A NanoStringSet for NanoStringDiff

Examples

```
# Example data
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata", "GSE117751_sample_data.csv",
package = "NanoTube")

# Load data without normalization
dat <- processNanostringData(nsFiles = example_data,
                     sampleTab = sample_data, groupCol = "Sample_Diagnosis",
                     normalization = "none")

# Convert to NanoStringSet
dat.ns <- makeNanoStringSetFromEset(dat)
```

---

nanostringPCA *Plot PCA*

Description

Conduct principal components analysis and plot the results, using either ggplot2 or plotly.

Usage

```
nanostringPCA(
  ns,
  pc1 = 1,
  pc2 = 2,
  interactive.plot = FALSE,
  exclude.zeros = TRUE,
  codeclass.retain = "endogenous"
)
```

Arguments

ns	Processed NanoString data
pc1	Principal component to plot on x-axis (default 1)
pc2	Principal component to plot on y-axis (default 2)
interactive.plot	Plot using plotly? Default FALSE (in which case ggplot2 is used)
exclude.zeros	Exclude genes that are not detected in all samples (default TRUE)

codeclass.retain
: The CodeClasses to retain for principal components analysis.Generally we're interested in endogenous genes, so we keep "endogenous" only by default. Others can be included by entering a character vector for this option. Alternatively, all targets can be retained by setting this option to ".".

Value

A list containing:

pca	The PCA object
plt	The PCA plot

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

# Process and normalize data first
dat <- processNanostringData(example_data,
                             sampleTab = sample_data,
                             groupCol = "Sample_Diagnosis",
                             normalization = "nSolver",
                             bgType = "t.test", bgPVal = 0.01)

# Interactive PCA using plotly
nanostringPCA(dat, interactive.plot = TRUE)$plt

# Static plot using ggplot2, for the 3rd and 4th PC's.
nanostringPCA(dat, pc1 = 3, pc2 = 4, interactive.plot = FALSE)$plt
```

NanoTube *NanoTube*

Description

A package for NanoString nCounter gene expression data processing, analysis, and visualization.

---

negativeQC *Calculate negative control statistics*

Description

Provide a table the negative control statistics, and plot the counts of negative control genes in each sample.

Usage

```
negativeQC(ns, interactive.plot = FALSE)
```

Arguments

ns	NanoString data, processed by 'processNanostringData' with output.format set to 'list' and 'nSolver' normalization.
interactive.plot	Generate an interactive plot using plotly? Only recommended for fewer than 20 samples (default FALSE)

Value

A list object containing:

tab	The table of negative control statistics, including the mean & standard deviation of negative control genes, calculated background threshold, and number of endogenous genes below that threshold
plt	An object containing the negative control plots.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

# Process and normalize data first
dat <- processNanostringData(example_data,
                             sampleTab = sample_data,
                             groupCol = "Sample_Diagnosis",
                             normalization = "nSolver",
                             bgType = "threshold",
                             bgThreshold = 2, bgProportion = 0.5,
                             output.format = "list")

negQC <- negativeQC(dat, interactive.plot = FALSE)

# View negative QC table & plot
head(negQC$tab)
negQC$plt
```

---

normalize_housekeeping

Housekeeping gene normalization

Description

Scale endogenous and housekeeping genes by the geometric mean of housekeeping genes. This should be conducted after positive control normalization and background correction. This step is conducted within processNanostringData, when normalization is set to "nCounter".

Usage

```
normalize_housekeeping(dat, genes = NULL, logfile = "")
```

Arguments

dat	NanoString data, including expression matrix and gene dictionary.
genes	List of housekeeping genes to use for normalization. If NULL (default), will use all genes marked as "Housekeeping" in codeset.
logfile	Optional name of logfile to print messages, warnings or errors.

Value

NanoString data, with expression matrix now normalized by housekeeping gene expression.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")

# Load data, positive control normalization, and background filtering
dat <- read_merge_rcc(list.files(example_data, full.names = TRUE))
dat <- normalize_pos_controls(dat)
dat <- remove_background(dat, mode = "t.test", pval = 0.05)

# Normalize by genes marked "Housekeeping" in RCC files
dat <- normalize_housekeeping(dat)

# Normalize by specified housekeeping genes (gene symbol or accession)
dat <- normalize_housekeeping(dat,
                        genes = c("TUBB", "TBP", "POLR2A", "GUSB", "SDHA"))
```

---

normalize_pos_controls

Positive control gene normalization

Description

Scale genes by the geometric mean of positive control genes. This step is conducted within processNanostringData, when normalization is set to "nCounter".

Usage

```
normalize_pos_controls(dat, logfile = "")
```

Arguments

dat	NanoString data, including expression matrix and gene dictionary.
logfile	Optional name of logfile to print messages, warnings or errors.

Value

NanoString data, with expression matrix now normalized by positive control gene expression.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")

dat <- read_merge_rcc(list.files(example_data, full.names = TRUE))

# Positive controls are identified in the RCC files, and used to
# normalize the data
dat <- normalize_pos_controls(dat)
```

---

nsdiffToFGSEA *Run gene set enrichment analysis using DE results.*

Description

Use the fgsea library to run gene set enrichment analysis from the NanoStringDiff analysis results. Genes will be ranked by their log2 fold changes.

Usage

```
nsdiffToFGSEA(deResults, gene.sets, sourceDB = NULL, min.set = 1)
```

Arguments

deResults	Result from NanoStringDiff::glm.LRT.
gene.sets	Gene set file name, in .rds (list), .gmt, or .tab format; or a list object containing the gene sets. Gene names must be in the same form as in the ranked.list.
sourceDB	Source database to include, only if using a .tab-format geneset.file from CPDB.
min.set	Number of genes required to conduct analysis on a given gene set (default = 1). If fewer than this number of genes from limmaResults are included in a gene set, that gene set will be skipped for this analysis.

Value

A list containing data frames with the fgsea results.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

datNoNorm <- processNanostringData(nsFiles = example_data,
                                   sampleTab = sample_data,
                                   groupCol = "Sample_Diagnosis",
                                   normalization = "none")

# Convert to NanoString Set, retaining 2 samples per group for this example
# (will run faster, but still pretty slow)
nsDiffSet <- makeNanoStringSetFromEset(datNoNorm[,c(1,2,15,16,29,30)])
```

```
# Run NanoStringDiff analysis
nsDiffSet <- NanoStringDiff::estNormalizationFactors(nsDiffSet)
result <- NanoStringDiff::glm.LRT(nsDiffSet,
                                  design.full = as.matrix(pData(nsDiffSet)),
                                  contrast = c(1, -1, 0))
                                  #contrast: Autoimmune retinopathy vs. None

# FGSEA with example pathways, only for pathways with at least 5 genes
# analyzed in NanoString experiment
data("ExamplePathways")
fgseaResult <- nsdiffToFGSEA(result, gene.sets = ExamplePathways,
                             min.set = 5)
```

`positiveQC` *Calculate positive control statistics*

Description

Calculate the linearity and scale factors of positive control genes, and plot the expected vs. observed counts for each sample.

Usage

```
positiveQC(ns, samples = NULL, expected = NULL)
```

Arguments

`ns`	NanoString data, processed by 'processNanostringData' with normalization set to 'none' or with output.format set to 'list'.
`samples`	A subset of samples to analyze (either a vector of sample names, or column indexes). If NULL (default), will include all samples.
`expected`	The expected values of each positive control gene, as a numeric vector. These are frequently provided by NanoString in the 'Name' field of the genes, in which case those values will be read automatically and this option can be left as NULL (the default).

Value

A list object containing:

`tab`	The table of positive control statistics, included the positive scale factor and the R-squared value for the expected vs. measured counts
`plt`	An object containing the positive control plots. This gets cumbersome if there are lots of samples.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata",
                           "GSE117751_sample_data.csv",
                           package = "NanoTube")

# Process data first. Must be output as a "list" or without normalization to
# obtain positive control statistics
dat <- processNanostringData(example_data,
                             sampleTab = sample_data,
                             groupCol = "Sample_Diagnosis",
                             normalization = "nSolver",
                             bgType = "t.test",
                             bgPVal = 0.01,
                             output.format = "list")

# Generate positive QC metrics for all samples
posQC <- positiveQC(dat)

# View positive QC table & plot
head(posQC$tab)
posQC$plt

# Plot for only the first three samples
posQC <- positiveQC(dat, samples = 1:3)
posQC$plt
```

`processNanostringData` *Process NanoString nCounter gene expression data.*

Description

This function reads in a zip file or folder containing multiple .rcc files (or a txt/csv file containing raw count data), and then optionally conducts positive control normalization, background correction, and housekeeping normalization.

Usage

```
processNanostringData(
  nsFiles,
  sampleTab = NULL,
  idCol = NULL,
  groupCol = NULL,
  replicateCol = NULL,
  normalization = c("nSolver", "RUVIII", "RUVg", "none"),
  bgType = c("threshold", "t.test", "none"),
  bgThreshold = 2,
  bgProportion = 0.5,
  bgPVal = 0.001,
  bgSubtract = FALSE,
  n.unwanted = NULL,
  RUVg.drop = 0,
```

```
  housekeeping = NULL,
  skip.housekeeping = FALSE,
  includeQC = FALSE,
  sampIds = NULL,
  output.format = c("ExpressionSet", "list"),
  logfile = ""
)
```

Arguments

nsFiles	file path (or zip file) containing the .rcc files, or multiple directories in a character vector, or a single text/csv file containing the combined counts, or .rcc files in a character vector.
sampleTab	.txt (tab-delimited) or .csv (comma-delimited) file containing sample data table (optional, default NULL)
idCol	the column name of the sample identifiers in the sample table, which should correspond to the column names in the count table (default NULL: will assume the first column contains the sample identifiers)
groupCol	the column name of the group identifiers in the sample table.
replicateCol	the column name of the technical replicate identifiers (default NULL). Multiple replicates of the same sample will have the same value in this column. Replicates are used to improve normalization performance in the "RUVIII" method; otherwise they are averaged.
normalization	If "nSolver" (default), continues with background, positive control, and housekeeping control normalization steps to return a NanoStringSet of normalized data. If "RUVIII", runs RUV normalization using controls, housekeeping genes and technical replicates. If "RUVg", runs RUV normalization using housekeeping genes. If "none", returns a NanoStringSet with the raw counts, suitable for running NanoStringDiff.
bgType	(Only if normalization is not "none") Type of background correction to use: "threshold" sets a threshold for N standard deviations above the mean of negative controls. "t.test" conducts a one-sided t test for each gene against all negative controls. "none" to skip background removal
bgThreshold	If bgType=="threshold", number of sd's above the mean to set as threshold for background correction.
bgProportion	If bgType=="threshold", proportion of samples that a gene must be above threshold to be included in analysis.
bgPVal	If bgType=="t.test", p-value threshold to use for gene to be included in analysis.
bgSubtract	Should calculated background levels be subtracted from reported expressions? If TRUE, will subtract mean+numSD*sd of the negative controls from the endogenous genes, and then set negative values to zero (default FALSE)
n.unwanted	The number of unwanted factors to use (for RUVIII or RUVg normalization only). If NULL (default), the maximum possible value will be identified and used.
RUVg.drop	The number of singular values to drop for RUVg normalization (see RUVSeq::RUVg)
housekeeping	vector of genes (symbols or accession) to use for housekeeping correction ("nCounter" or "RUVg" normalization). If NULL, will use genes listed as "Housekeeping" under CodeClass.

skip.housekeeping
: Skip housekeeping normalization? (default FALSE)

includeQC
: Should we include the QC from the .rcc files? This can cause errors, particularly when reading in files from multiple experiments.

sampIds
: a vector of sample identifiers, important if there are technical replicates. Currently, this function averages technical replicates. sampIds will be extracted from the replicateCol in the sampleTab, if provided.

output.format
: If "list", will return the normalized (optional) and raw expression data, as well as various QC and relevant information tables. If "ExpressionSet" (default), will convert to an n*p ExpressionSet, with n rows representing genes and p columns representing samples. ExpressionSet objects are required for some steps, such as runLimmaAnalysis.

logfile
: a filename for the logfile (optional). If blank, will print warnings to screen.

Value

An list or ExpressionSet containing the raw and/or normalized counts, dictionary, and sample info if provided

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_data <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

# Process NanoString data from RCC files present in example_data folder.
# Use standard nCounter normalization, removing genes that do
# pass a t test against negative control genes with p < 0.05. Return the
# result as an "ExpressionSet".

dat <- processNanostringData(nsFiles = example_data,
                             sampleTab = sample_data,
                             groupCol = "Sample_Diagnosis",
                             normalization = "nSolver",
                             bgType = "t.test", bgPVal = 0.01,
                             output.format = "ExpressionSet")

# Load NanoString data from a csv file (from NanoString's RCC Collector tool,
# for example). Skip normalization by setting 'normalization = "none"'.

csv_data <- system.file("extdata", "GSE117751_expression_matrix.csv",
                        package = "NanoTube")
dat <- processNanostringData(nsFile = csv_data,
                             sampleTab = sample_data,
                             idCol = "GEO_Accession",
                             groupCol = "Sample_Diagnosis",
                             normalization = "none")

# Load NanoString data from RCC files, using a threshold background level for
# removing low-expressed genes. Also, specify which genes to use for
# housekeeping normalization. Save the result in "list" format (useful for
# some QC functions) instead of an "ExpressionSet".

dat <- processNanostringData(nsFiles = example_data,
                             sampleTab = sample_data,
```

```
groupCol = "Sample_Diagnosis",
normalization = "nSolver",
bgType = "threshold",
bgThreshold = 2, bgProportion = 0.5,
housekeeping = c("TUBB", "TBP", "POLR2A",
                 "GUSB", "SDHA"),
output.format = "list")
```

`read_cpdb_sourceDBs` *Identify source databases from a .tab file*

Description

Read in a .tab file from the Consensus Pathway Database (CPDB), and identify the source databases present.

Usage

```
read_cpdb_sourceDBs(file)
```

Arguments

`file`	The filename

Value

A table of the source databases, with the number of gene sets from each one.

`read_cpdb_tab` *Read .tab file.*

Description

Read in a .tab file from the Consensus Pathway Database (CPDB)

Usage

```
read_cpdb_tab(file, sourceDB = NULL)
```

Arguments

`file`	The filename
`sourceDB`	The source database to use. If NULL (default), retains gene sets from all source databases

Value

A list object, containing a character vector of genes for each gene set.

read_merge_rcc *Merge multiple .rcc files*

Description

Read in multiple .rcc files named in the fileList and merge the expression data. This step is conducted within processNanostringData.

Usage

```
read_merge_rcc(fileList, includeQC = FALSE, logfile = "")
```

Arguments

fileList	a character vector of .rcc file names
includeQC	include merged QC data (from the "Lane Attributes" part of file) in the output? Default FALSE
logfile	a filename for the logfile (optional). If blank, will print warnings to screen.

Value

A list object including:

exprs	The expression matrix
dict	The gene dictionary
qc	QC metrics included in the .rcc files, if includeQC == TRUE

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")

dat <- read_merge_rcc(list.files(example_data, full.names = TRUE))
```

read_rcc *Read .rcc file*

Description

This function reads in a single .rcc file and splits into expression, sample data, and qc components.

Usage

```
read_rcc(file)
```

Arguments

file	file name

Value

list containing expression data, sample attributes, and basic qc from the .rcc file.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")

# First file only
single_file <- list.files(example_data, full.names = TRUE)[1]
single_dat <- read_rcc(single_file)
```

`read_sampleData` *Read in a sample data table.*

Description

Read in a .txt or .csv file containing sample names, group identifiers, replicate identifiers, and any other sample data. Sample names must be in the first column and must correspond with sample names in the count data file(s).

Usage

```
read_sampleData(dat, file.name, idCol = NULL, groupCol, replicateCol = NULL)
```

Arguments

`dat`	expression data, read in by read_merge_rcc or read.delim
`file.name`	the path/name of the .txt or .csv file
`idCol`	the column name of the sample identifiers in the sample table, which should correspond to the column names in the count table (default NULL: will assume the first column contains the sample identifiers).
`groupCol`	the column name of the group identifiers.
`replicateCol`	the column name of the replicate identifiers (default NULL). Multiple replicates of the same sample will have the same value in this column.

Value

The list with the expression data, now combined with the sample information

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_info <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

dat <- read_merge_rcc(list.files(example_data, full.names = TRUE))

# Merge expression data with sample info
dat <- read_sampleData(dat, file.name = sample_info,
                       groupCol = "Sample_Diagnosis")
```

remove_background *Assess background expression*

Description

Compare endogenous gene expression data against negative control genes and remove data for genes that fail the comparison. This step is conducted within processNanostringData, when normalization is set to "nCounter".

Usage

```
remove_background(
  dat,
  mode = c("threshold", "t.test"),
  numSD,
  proportionReq,
  pval,
  subtract = FALSE
)
```

Arguments

dat	Positive control-scaled NanoString data
mode	Either "threshold" (default) or "t.test". If "threshold", requires proportionReq of samples to have expression numSD standard deviations among the mean of negative control genes. If "t.test", each gene will be compared with all negative control genes in a one-sided two-sample t-test.
numSD	Number of standard deviations above mean of negative control genes to used as background threshold for each sample: mean(negative_controls) + numSD * sd(negative_controls). Required if mode == "threshold" or subtract == TRUE
proportionReq	Required proportion of sample expressions exceeding the sample background threshold to include gene in further analysis. Required if mode == "threshold" or subtract == TRUE
pval	p-value (from one-sided t-test) threshold to declare gene expression above background expression level. Genes with p-values above this level are removed from further analysis. Required if mode == "t.test"
subtract	Should calculated background levels be subtracted from reported expressions? If TRUE, will subtract mean+numSD*sd of the negative controls from the endogenous genes, and then set negative values to zero (default FALSE).

Value

NanoString data, with genes removed that fail the comparison test against negative control genes. Expression levels are updated for all genes if subtract == TRUE.

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")

# Load data and positive control normalization
```

```
dat <- read_merge_rcc(list.files(example_data, full.names = TRUE))
dat <- normalize_pos_controls(dat)

# Remove endogenous genes that fail to reject the null hypothesis
# in a one-sided t test against negative control genes with p < 0.05.
dat <- remove_background(dat, mode = "t.test", pval = 0.05)

# Remove endogenous genes where fewer than 25% of samples have an expression
# 2 standard deviations above the average negative control gene. Also,
# subtract this background level (mean + 2*sd) from endogenous genes.
dat <- remove_background(dat, mode = "threshold",
                         numSD = 2, proportionReq = 0.25, subtract = TRUE)
```

`runLimmaAnalysis` *Conduct differential expression analysis*

Description

Use Limma to conduct a simple differential expression analysis. All groups are compared against the base.group, and empirical Bayes method is used to identify significantly differentially expressed genes. Alternatively, a design matrix can be supplied, as explained in limma::limmaUsersGuide()

Usage

```
runLimmaAnalysis(
  dat,
  groups = NULL,
  base.group = NULL,
  design = NULL,
  codeclass.retain = "endogenous",
  ...
)
```

Arguments

`dat`	NanoString data ExpressionSet, from processNanostringData
`groups`	character vector, in same order as the samples in dat. NULL if already included in 'dat'
`base.group`	the group against which other groups are compared (must be one of the levels in 'groups'). Will use the first group if NULL.
`design`	a design matrix for Limma analysis (default NULL, will do analysis based on provided 'group' data)
`codeclass.retain`	The CodeClasses to retain for Limma analysis. Generally we're interested in endogenous genes, so we keep "endogenous" only by default. Others can be included by entering a character vector for this option (see limmaResults3 example). Alternatively, all targets can be retained by setting this option to ".".
`...`	Optional arguments to be passed to limma::lmFit

Value

The fit Limma object

Examples

```
example_data <- system.file("extdata", "GSE117751_RAW", package = "NanoTube")
sample_info <- system.file("extdata", "GSE117751_sample_data.csv",
                           package = "NanoTube")

dat <- processNanostringData(nsFiles = example_data,
                             sampleTab = sample_info,
                             groupCol = "Sample_Diagnosis")

# Compare the two diseases against healthy controls ("None")
limmaResults <- runLimmaAnalysis(dat, base.group = "None")


# You can also supply a design matrix
# Generate fake batch labels
batch <- rep(c(0, 1), times = ncol(dat) / 2)

# Reorder groups ("None" first)
group <- factor(dat$groups, levels = c("None", "Autoimmune retinopathy",
                                       "Retinitis pigmentosa"))

# Design matrix including sample group and batch
design <- model.matrix(~group + batch)

# Analyze data
limmaResults2 <- runLimmaAnalysis(dat, design = design)

# Run Limma analysis including endogenous *and* housekeeping genes.
limmaResults3 <- runLimmaAnalysis(dat, design = design,
                     codeclass.retain = c("endogenous", "housekeeping"))
```

`untar_dirs` *Untar*

Description

Untars provided list of directories (analogous to unzip_dirs)

Usage

```
untar_dirs(fileDirs)
```

Arguments

`fileDirs`	character list of tar files

Value

Names of now-untarred directories

unzip_dirs *Unzip*

Description

Unzips provided list of directories

Usage

```
unzip_dirs(fileDirs)
```

Arguments

fileDirs	character list of zip files

Value

Names of now-unzipped directories

Index