

# Package ‘yarn’

May 30, 2024

**Title** YARN: Robust Multi-Condition RNA-Seq Preprocessing and Normalization

**Version** 1.30.0

**Description** Expedite large RNA-Seq analyses using a combination of previously developed tools. YARN is meant to make it easier for the user in performing basic mis-annotation quality control, filtering, and condition-aware normalization. YARN leverages many Bioconductor tools and statistical techniques to account for the large heterogeneity and sparsity found in very large RNA-seq experiments.

**Depends** Biobase

**License** Artistic-2.0

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown, testthat (>= 0.8)

**Imports** biomaRt, downloader, edgeR, gplots, graphics, limma, matrixStats, preprocessCore, readr, RColorBrewer, stats, quantro

**VignetteBuilder** knitr

**biocViews** Software, QualityControl, GeneExpression, Sequencing, Preprocessing, Normalization, Annotation, Visualization, Clustering

**git\_url** <https://git.bioconductor.org/packages/yarn>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 54b819d

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-29

**Author** Joseph N Paulson [aut, cre],  
 Cho-Yi Chen [aut],  
 Camila Lopes-Ramos [aut],  
 Marieke Kuijjer [aut],  
 John Platig [aut],  
 Abhijeet Sonawane [aut],  
 Maud Fagny [aut],  
 Kimberly Glass [aut],  
 John Quackenbush [aut]

**Maintainer** Joseph N Paulson <paulson.joseph@gene.com>

## Contents

annotateFromBiomart . . . . .	2
bladder . . . . .	3
checkMisAnnotation . . . . .	4
checkTissuesToMerge . . . . .	5
downloadGTEx . . . . .	6
extractMatrix . . . . .	6
filterGenes . . . . .	7
filterLowGenes . . . . .	8
filterMissingGenes . . . . .	8
filterSamples . . . . .	9
normalizeTissueAware . . . . .	10
plotCMDS . . . . .	11
plotDensity . . . . .	12
plotHeatmap . . . . .	12
qsmooth . . . . .	13
qstats . . . . .	14
skin . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

annotateFromBiomart    *Annotate your Expression Set with biomaRt*

---

## Description

Annotate your Expression Set with biomaRt

## Usage

```
annotateFromBiomart(obj, genes = featureNames(obj),
  filters = "ensembl_gene_id", attributes = c("ensembl_gene_id",
  "hgnc_symbol", "chromosome_name", "start_position", "end_position"),
  biomaRt = "ensembl", dataset = "hsapiens_gene_ensembl", ...)
```

**Arguments**

obj	ExpressionSet object.
genes	Genes or rownames of the ExpressionSet.
filters	getBM filter value, see getBM help file.
attributes	getBM attributes value, see getBM help file.
biomart	BioMart database name you want to connect to. Possible database names can be retrieved with the function listMarts.
dataset	Dataset you want to use. To see the different datasets available within a biomart you can e.g. do: mart = useMart('ensembl'), followed by listDatasets(mart).
...	Values for useMart, see useMart help file.

**Value**

ExpressionSet object with a fuller featureData.

**Examples**

```
data(skin)
# subsetting and changing column name just for a silly example
skin <- skin[1:10,]
colnames(fData(skin)) = paste("names",1:6)
biomart<-"ENSEMBL_MART_ENSEMBL";
genes <- sapply(strsplit(rownames(skin),split="\\."),function(i)i[1])
newskin <-annotateFromBiomart(skin,genes=genes,biomart=biomart)
head(fData(newskin)[,7:11])
```

---

bladder

*Bladder RNA-seq data from the GTEx consortium*


---

**Description**

Bladder RNA-seq data from the GTEx consortium. V6 release.

**Usage**

```
data(bladder)
```

**Format**

An object of class "ExpressionSet"; see [ExpressionSet](#).

**Value**

ExpressionSet object

**Source**

GTEEx Portal

**References**

GTEEx Consortium, 2015. The Genotype-Tissue Expression (GTEx) pilot analysis: Multitissue gene regulation in humans. *Science*, 348(6235), pp.648-660. ([PubMed](#))

**Examples**

```
data(bladder);
checkMissAnnotation(bladder);
```

---

checkMisAnnotation	<i>Check for wrong annotation of a sample using classical MDS and control genes.</i>
--------------------	--

---

**Description**

Check for wrong annotation of a sample using classical MDS and control genes.

**Usage**

```
checkMisAnnotation(obj, phenotype, controlGenes = "all",
  columnID = "chromosome_name", plotFlag = TRUE,
  legendPosition = NULL, ...)
```

**Arguments**

obj	ExpressionSet object.
phenotype	phenotype column name in the phenoData slot to check.
controlGenes	Name of controlGenes, ie. 'Y' chromosome. Can specify 'all'.
columnID	Column name where controlGenes is defined in the featureData slot if other than 'all'.
plotFlag	TRUE/FALSE Whether to plot or not
legendPosition	Location for the legend.
...	Extra parameters for <a href="#">plotCMDS</a> function.

**Value**

Plots a classical multi-dimensional scaling of the 'controlGenes'. Optionally returns co-ordinates.

**Examples**

```
data(bladder)
checkMisAnnotation(bladder, 'GENDER', controlGenes='Y', legendPosition='topleft')
```

---

checkTissuesToMerge    *Check tissues to merge based on gene expression profile*

---

### Description

Check tissues to merge based on gene expression profile

### Usage

```
checkTissuesToMerge(obj, majorGroups, minorGroups, filterFun = NULL,  
  plotFlag = TRUE, ...)
```

### Arguments

obj	ExpressionSet object.
majorGroups	Column name in the phenoData slot that describes the general body region or site of the sample.
minorGroups	Column name in the phenoData slot that describes the specific body region or site of the sample.
filterFun	Filter group specific genes that might disrupt PCoA analysis.
plotFlag	TRUE/FALSE whether to plot or not
...	Parameters that can go to <a href="#">checkMisAnnotation</a>

### Value

CMDS Plots of the majorGroupss colored by the minorGroupss. Optional matrix of CMDS loadings for each comparison.

### See Also

checkTissuesToMerge

### Examples

```
data(skin)  
checkTissuesToMerge(skin, 'SMTS', 'SMTSD')
```

---

downloadGTEX	<i>Download GTEX files and turn them into ExpressionSet object</i>
--------------	--

---

**Description**

Downloads the V6 GTEX release and turns it into an ExpressionSet object.

**Usage**

```
downloadGTEX(type = "genes", file = NULL, ...)
```

**Arguments**

type	Type of counts to download - default genes.
file	File path and name to automatically save the downloaded GTEX expression set. Saves as a RDS file.
...	Does nothing currently.

**Value**

Organized ExpressionSet set.

**Examples**

```
# obj <- downloadGTEX(type='genes',file='~/Desktop/gtex.rds')
```

---

extractMatrix	<i>Extract the appropriate matrix</i>
---------------	---------------------------------------

---

**Description**

This returns the raw counts, log<sub>2</sub>-transformed raw counts, or normalized expression. If normalized = TRUE then the log parameter is ignored.

**Usage**

```
extractMatrix(obj, normalized = FALSE, log = TRUE)
```

**Arguments**

obj	ExpressionSet object or objrix.
normalized	TRUE / FALSE, use the normalized matrix or raw counts
log	TRUE/FALSE log <sub>2</sub> -transform.

**Value**

matrix

**Examples**

```
data(skin)
head(yarn:::extractMatrix(skin,normalized=FALSE,log=TRUE))
head(yarn:::extractMatrix(skin,normalized=FALSE,log=FALSE))
```

---

filterGenes

*Filter specific genes*

---

**Description**

The main use case for this function is the removal of sex-chromosome genes. Alternatively, filter genes that are not protein-coding.

**Usage**

```
filterGenes(obj, labels = c("X", "Y", "MT"),
            featureName = "chromosome_name", keepOnly = FALSE)
```

**Arguments**

obj	ExpressionSet object.
labels	Labels of genes to filter or keep, eg. X, Y, and MT
featureName	FeatureData column name, eg. chr
keepOnly	Filter or keep only the genes with those labels

**Value**

Filtered ExpressionSet object

**Examples**

```
data(skin)
filterGenes(skin,labels = c('X','Y','MT'),featureName='chromosome_name')
filterGenes(skin,labels = 'protein_coding',featureName='gene_biotype',keepOnly=TRUE)
```

---

filterLowGenes	<i>Filter genes that have less than a minimum threshold CPM for a given group/tissue</i>
----------------	--

---

**Description**

Filter genes that have less than a minimum threshold CPM for a given group/tissue

**Usage**

```
filterLowGenes(obj, groups, threshold = 1, minSamples = NULL, ...)
```

**Arguments**

obj	ExpressionSet object.
groups	Vector of labels for each sample or a column name of the phenoData slot. for the ids to filter. Default is the column names.
threshold	The minimum threshold for calling presence of a gene in a sample.
minSamples	Minimum number of samples - defaults to half the minimum group size.
...	Options for <a href="#">cpm</a> .

**Value**

Filtered ExpressionSet object

**See Also**

[cpm](#) function defined in the edgeR package.

**Examples**

```
data(skin)
filterLowGenes(skin, 'SMTSD')
```

---

filterMissingGenes	<i>Filter genes not expressed in any sample</i>
--------------------	---

---

**Description**

The main use case for this function is the removal of missing genes.

**Usage**

```
filterMissingGenes(obj, threshold = 0)
```



**Arguments**

obj                    ExpressionSet object.  
 threshold            Minimum sum of gene counts across samples – defaults to zero.

**Value**

Filtered ExpressionSet object

**Examples**

```
data(skin)
filterMissingGenes(skin)
```

---

filterSamples	<i>Filter samples</i>
---------------	-----------------------

---

**Description**

Filter samples

**Usage**

```
filterSamples(obj, ids, groups = colnames(obj), keepOnly = FALSE)
```

**Arguments**

obj                    ExpressionSet object.  
 ids                    Names found within the groups labels corresponding to samples to be removed  
 groups                Vector of labels for each sample or a column name of the phenoData slot for the  
                          ids to filter. Default is the column names.  
 keepOnly             Filter or keep only the samples with those labels.

**Value**

Filtered ExpressionSet object

**Examples**

```
data(skin)
filterSamples(skin,ids = "Skin - Not Sun Exposed (Suprapubic)",groups="SMTSD")
filterSamples(skin,ids=c("GTEX-OHPL-0008-SM-4E3I9","GTEX-145MN-1526-SM-5SI9T"))
```

---

normalizeTissueAware *Normalize in a tissue aware context*

---

### Description

This function provides a wrapper to various normalization methods developed. Currently it only wraps qsmooth and quantile normalization returning a log-transformed normalized matrix. qsmooth is a normalization approach that normalizes samples in a condition aware manner.

### Usage

```
normalizeTissueAware(obj, groups, normalizationMethod = c("qsmooth",  
  "quantile"), ...)
```

### Arguments

obj	ExpressionSet object
groups	Vector of labels for each sample or a column name of the phenoData slot for the ids to filter. Default is the column names
normalizationMethod	Choice of 'qsmooth' or 'quantile'
...	Options for <a href="#">qsmooth</a> function or <a href="#">normalizeQuantiles</a>

### Value

ExpressionSet object with an assayData called normalizedMatrix

### Source

The function qsmooth comes from the qsmooth packages currently available on github under user 'kokrah'.

### Examples

```
data(skin)  
normalizeTissueAware(skin, "SMTSD")
```

---

`plotCMDS`*Plot classical MDS of dataset*

---

**Description**

This function plots the MDS coordinates for the "n" features of interest. Potentially uncovering batch effects or feature relationships.

**Usage**

```
plotCMDS(obj, comp = 1:2, normalized = FALSE, distFun = dist,  
  distMethod = "euclidian", n = NULL, samples = TRUE, log = TRUE,  
  plotFlag = TRUE, ...)
```

**Arguments**

<code>obj</code>	ExpressionSet object or objrix.
<code>comp</code>	Which components to display.
<code>normalized</code>	TRUE / FALSE, use the normalized matrix or raw counts.
<code>distFun</code>	Distance function, default is <code>dist</code> .
<code>distMethod</code>	The distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
<code>n</code>	Number of features to make use of in calculating your distances.
<code>samples</code>	Perform on samples or genes.
<code>log</code>	TRUE/FALSE log2-transform raw counts.
<code>plotFlag</code>	TRUE/FALSE whether to plot or not.
<code>...</code>	Additional plot arguments.

**Value**

coordinates

**Examples**

```
data(skin)  
res <- plotCMDS(skin, pch=21, bg=factor(pData(skin)$SMTSD))  
  
# library(calibrate)  
# textxy(X=res[,1], Y=res[,2], labs=rownames(res))
```

---

plotDensity                      *Density plots of columns in a matrix*

---

### Description

Plots the density of the columns of a matrix. Wrapper for [matdensity](#).

### Usage

```
plotDensity(obj, groups = NULL, normalized = FALSE, legendPos = NULL,
  ...)
```

### Arguments

obj	ExpressionSet object
groups	Vector of labels for each sample or a column name of the phenoData slot for the ids to filter. Default is the column names.
normalized	TRUE / FALSE, use the normalized matrix or log2-transformed raw counts
legendPos	Legend title position. If null, does not create legend by default.
...	Extra parameters for <a href="#">matdensity</a> .

### Value

A density plot for each column in the ExpressionSet object colored by groups

### Examples

```
data(skin)
filtData <- filterLowGenes(skin, "SMTSD")
plotDensity(filtData, groups="SMTSD", legendPos="topleft")
# to remove the legend
plotDensity(filtData, groups="SMTSD")
```

---

plotHeatmap                      *Plot heatmap of most variable genes*

---

### Description

This function plots a heatmap of the gene expressions for the "n" features of interest.

### Usage

```
plotHeatmap(obj, n = NULL, fun = stats::sd, normalized = TRUE,
  log = TRUE, ...)
```

**Arguments**

obj	ExpressionSet object or objrix.
n	Number of features to make use of in plotting heatmap.
fun	Function to sort genes by, default <a href="#">sd</a> .
normalized	TRUE / FALSE, use the normalized matrix or raw counts.
log	TRUE/FALSE log2-transform raw counts.
...	Additional plot arguments for <a href="#">heatmap.2</a> .

**Value**

coordinates

**Examples**

```
data(skin)
tissues <- pData(skin)$SMTSD
plotHeatmap(skin, normalized=FALSE, log=TRUE, trace="none", n=10)
# Even prettier

# library(RColorBrewer)
data(skin)
tissues <- pData(skin)$SMTSD
heatmapColColors <- brewer.pal(12, "Set3")[as.integer(factor(tissues))]
heatmapCols <- colorRampPalette(brewer.pal(9, "RdBu"))(50)
plotHeatmap(skin, normalized=FALSE, log=TRUE, trace="none", n=10,
  col = heatmapCols, ColSideColors = heatmapColColors, cexRow = 0.6, cexCol = 0.6)
```

---

qsmooth

*Quantile shrinkage normalization*


---

**Description**

This function was modified from github user kokrah.

**Usage**

```
qsmooth(obj, groups, norm.factors = NULL, plot = FALSE,
  window = 0.05, log = TRUE)
```

**Arguments**

obj	for counts use $\log_2(\text{raw counts} + 1)$ , for MA use $\log_2(\text{raw intensities})$
groups	groups to which samples belong (character vector)
norm.factors	scaling normalization factors
plot	plot weights? (default=FALSE)

window            window size for running median (a fraction of the number of rows of exprs)  
log                Whether or not the data should be log transformed before normalization, TRUE  
                    = YES.

**Value**

Normalized expression

**Source**

[Kwame Okrah's qsmooth R package](#)

**Examples**

```
data(skin)
head(yarn:::qsmooth(skin, groups=pData(skin)$SMTSD))
```

---

qstats

*Compute quantile statistics*

---

**Description**

This function was directly borrowed from github user kokrah.

**Usage**

```
qstats(exprs, groups, window)
```

**Arguments**

exprs            for counts use  $\log_2(\text{raw counts} + 1)$ , for MA use  $\log_2(\text{raw intensities})$   
groups           groups to which samples belong (character vector)  
window          window size for running median as a fraction on the number of rows of exprs

**Value**

list of statistics

**Source**

[Kwame Okrah's qsmooth R package](#) Compute quantile statistics

---

skin

*Skin RNA-seq data from the GTEx consortium*

---

**Description**

Skin RNA-seq data from the GTEx consortium. V6 release. Random selection of 20 skin samples. 13 of the samples are fibroblast cells, 5 Skin sun exposed, 2 sun unexposed.

**Usage**

```
data(skin)
```

**Format**

An object of class "ExpressionSet"; see [ExpressionSet](#).

**Value**

ExpressionSet object

**Source**

GTEx Portal

**References**

GTEx Consortium, 2015. The Genotype-Tissue Expression (GTEx) pilot analysis: Multitissue gene regulation in humans. *Science*, 348(6235), pp.648-660. ([PubMed](#))

**Examples**

```
data(skin);  
checkMissAnnotation(skin, "GENDER");
```

# Index

## \* datasets

bladder, [3](#)

skin, [15](#)

annotateFromBiomart, [2](#)

bladder, [3](#)

checkMisAnnotation, [4](#), [5](#)

checkTissuesToMerge, [5](#)

cpm, [8](#)

downloadGTEx, [6](#)

ExpressionSet, [3](#), [15](#)

extractMatrix, [6](#)

filterGenes, [7](#)

filterLowGenes, [8](#)

filterMissingGenes, [8](#)

filterSamples, [9](#)

heatmap.2, [13](#)

matdensity, [12](#)

normalizeQuantiles, [10](#)

normalizeTissueAware, [10](#)

plotCMDS, [4](#), [11](#)

plotDensity, [12](#)

plotHeatmap, [12](#)

qsmooth, [10](#), [13](#)

qstats, [14](#)

sd, [13](#)

skin, [15](#)