Package ‘RNAseqQC’

July 15, 2024

Title  Quality Control for RNA-Seq Data
Version  0.2.1
Description  Functions for semi-automated quality control of bulk RNA-seq data.
License  Apache License (>= 2)
Encoding  UTF-8
LazyData  true
RoxygenNote  7.3.1
Depends  R (>= 4.0)
Imports  AnnotationHub, AnnotationFilter, BiocGenerics, ensembldb,
circlize, DESeq2, SummarizedExperiment, ComplexHeatmap,
matrixStats, grid, ggpointdensity, ggrepel, ggplot2, cowplot,
patchwork, purrr, dplyr, stringr, tidyrr, tibble, tidyselect,
magrittr
Suggests  rmarkdown, knitr, recount3, apeglm, ggrastr, gghighlight
VignetteBuilder  knitr
URL  https://github.com/frederikziebell/RNAseqQC
BugReports  https://github.com/frederikziebell/RNAseqQC/issues
NeedsCompilation  no
Author  Frederik Ziebell [cre],
Frederik Ziebell [aut] (https://orcid.org/0000-0003-3673-1721),
GlaxoSmithKline Research & Development Limited [cph] (GlaxoSmithKline
Research & Development Limited; registered address: 980 Great West
Road, Brentford, Middlesex TW8 9GS, United Kingdom)
Maintainer  Frederik Ziebell <f_ziebell@web.de>
Repository  CRAN
Date/Publication  2024-07-15 14:40:02 UTC
## Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>all_numeric</td>
<td>2</td>
</tr>
<tr>
<td>filter_genes</td>
<td>3</td>
</tr>
<tr>
<td>get_gene_id</td>
<td>3</td>
</tr>
<tr>
<td>make_dds</td>
<td>4</td>
</tr>
<tr>
<td>mean_sd_plot</td>
<td>5</td>
</tr>
<tr>
<td>plot_biotypes</td>
<td>5</td>
</tr>
<tr>
<td>plot_chromosome</td>
<td>6</td>
</tr>
<tr>
<td>plot_gene</td>
<td>7</td>
</tr>
<tr>
<td>plot_gene_detection</td>
<td>8</td>
</tr>
<tr>
<td>plot_library_complexity</td>
<td>8</td>
</tr>
<tr>
<td>plot_loadings</td>
<td>9</td>
</tr>
<tr>
<td>plot_ma</td>
<td>10</td>
</tr>
<tr>
<td>plot_pca</td>
<td>11</td>
</tr>
<tr>
<td>plot_pca_scatters</td>
<td>12</td>
</tr>
<tr>
<td>plot_sample_clustering</td>
<td>14</td>
</tr>
<tr>
<td>plot_sample_MAs</td>
<td>15</td>
</tr>
<tr>
<td>plot_total_counts</td>
<td>15</td>
</tr>
<tr>
<td>plot_within_level_sample_MAs</td>
<td>16</td>
</tr>
<tr>
<td>save_plots_to_pdf</td>
<td>17</td>
</tr>
<tr>
<td>T47D</td>
<td>18</td>
</tr>
<tr>
<td>T47D_diff_testing</td>
<td>19</td>
</tr>
</tbody>
</table>

## Index

- all_numeric

## Description

for a vector x, check if all non-NA elements of x can be converted to numeric

## Usage

```r
all_numeric(x)
```

## Arguments

- `x` A non-numeric vector
**filter_genes**  
*Filter genes with low counts*

**Description**
Filter genes with low counts

**Usage**
```r
filter_genes(dds, min_count = 5, min_rep = 3)
```

**Arguments**
- `dds` A DESeqDataSet
- `min_count`, `min_rep`  
  keep genes with at least `min_count` counts in at least `min_rep` replicates

**Value**
A DESeq2::DESeqDataSet object with only those genes that meet the filter criteria.

**Examples**
```r
library("DESeq2")
dds <- makeExampleDESeqDataSet()
filter_genes(dds)
```

---

**get_gene_id**  
*Get all gene IDs in a DESeqDataSet for a given gene name.*

**Description**
Get all gene IDs in a DESeqDataSet for a given gene name.

**Usage**
```r
get_gene_id(gene_name, dds)
```

**Arguments**
- `gene_name` A gene name
- `dds` A DESeqDataSet

**Value**
A character vector
make.dds

Examples

get_gene_id("HBA1", T47D)

---

**Description**

Make DESeqDataSet from counts matrix and metadata

**Usage**

```r
make.dds(counts, metadata, ah_record, design = ~1)
```

**Arguments**

- **counts**: The genes x samples counts matrix with row names. At least one row name must be an ENSEMBL gene ID, since gene annotation is done via the ENSEMBL database.
- **metadata**: data.frame of sample information. Order of rows corresponds to the order of columns in the counts matrix.
- **ah_record**: ID of AnnotationHub record used to retrieve an EnsDb object.
- **design**: The design formula specified in DESeqDataSet(). To view all valid record IDs, run

```r
library(AnnotationHub)
mcols(AnnotationHub()) %>%
as_tibble(rownames="ah_record") %>%
filter(rdataclass="EnsDb")
```

**Value**

A DESeq2::DESeqDataSet object containing the counts matrix and metadata.

**Examples**

```r
library("DESeq2")
count_mat <- counts(T47D)
meta <- data.frame(colData(T47D))
dds <- make.dds(counts = count_mat, metadata = meta, ah_record = "AH89426")
```
### mean_sd_plot

Create a mean-sd plot. Make a scatterplot that shows for each gene its standard deviation versus mean.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a mean-sd plot. Make a scatterplot that shows for each gene its standard deviation versus mean.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean_sd_plot(vsd)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ggplot object of the ggplot2 package that contains the mean-sd plot.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
</table>
| library("DESeq2")
dds <- makeExampleDESeqDataSet(interceptMean=10, n=5000)
vsd <- vst(dds)
mean_sd_plot(vsd) |

---

### plot_biotypes

Plot number of counts per sample and biotype

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plot the total number of counts for each sample and the major classes of ENSEMBL gene biotypes (protein coding, IncRNA, etc.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot_biotypes(dds)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>dds</td>
</tr>
</tbody>
</table>
Value

A ggplot object of the ggplot2 package.

Examples

plot_biotypes(T47D)

---

plot_chromosome

Plot gene expression along a chromosome

Description

Plot gene expression along a chromosome

Usage

plot_chromosome(vsd, chr, scale = FALSE, trunc_val = NULL)

Arguments

- `vsd`: An object generated by DESeq2::vst()
- `chr`: A string denoting a chromosome as annotated by ENSEMBL, e.g. '1', '2', 'X', 'Y', 'MT'
- `scale`: Whether to scale the columns of the heatmap
- `trunc_val`: Truncate the expression matrix to this value prior to plotting. This is useful if some very high expression values dominate the heatmap. By default, the heatmap is truncated to expression values at most 3 standard deviations from the mean.

Value

A Heatmap-class object of the ComplexHeatmap package that contains the heatmap of expression values.

Examples

library("DESeq2")
chr1 <- T47D[which(mcols(T47D)$chromosome=="1"),]
vsd <- vst(chr1)
plot_chromosome(vsd, chr="1")
plot_gene

Description
Plot a gene

Usage
plot_gene(
  gene,
  dds,
  x_var = NULL,
  color_by = NULL,
  point_alpha = 0.7,
  point_rel_size = 2,
  show_plot = TRUE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene</td>
<td>A gene ID or gene name, i.e. an element of rownames(dds) or of rowData(dds)$gene_name</td>
</tr>
<tr>
<td>dds</td>
<td>a DESeqDataSet</td>
</tr>
<tr>
<td>x_var</td>
<td>Variable to plot on the x-axis. If NULL, then each sample is plotted separately.</td>
</tr>
<tr>
<td>color_by</td>
<td>Variable (column in colData(dds)) to color points by.</td>
</tr>
<tr>
<td>point_alpha</td>
<td>alpha value of geom_point()</td>
</tr>
<tr>
<td>point_rel_size</td>
<td>relative size of geom_point()</td>
</tr>
<tr>
<td>show_plot</td>
<td>Whether to show the plot or not</td>
</tr>
</tbody>
</table>

Value
The function displays the plot and returns invisible the data frame of expression values and colData annotation for the gene.

Examples
library("DESeq2")
set.seed(1)
dds <- makeExampleDESeqDataSet()
colData(dds)$patient <- c("1", "2", "2", "3", "3")
dds <- estimateSizeFactors(dds)
plot_gene("gene1", dds)
plot_gene("gene1", dds, x_var="patient", color_by="type")
plot_gene_detection  
Plot number of detected genes for each sample

Description
For specified thresholds, the number of detected genes is shown for each sample.

Usage
plot_gene_detection(dds, thresholds = c(3, 10, 20, 50))

Arguments
- dds: A DESeqDataSet
- thresholds: Vector of thresholds for which the number of genes with counts greater or equal than the thresholds is plotted

Value
A ggplot object of the ggplot2 package that contains the gene detection plot.

Examples
library("DESeq2")
set.seed(1)
dds <- makeExampleDESeqDataSet()
plot_gene_detection(dds)

plot_library_complexity
Plot the library complexity

Description
Plot per sample the fraction of genes, versus the fraction of total counts.

Usage
plot_library_complexity(dds, show_progress = TRUE)

Arguments
- dds: A DESeqDataSet
- show_progress: Whether to show a progress bar of the computation.
plot_loadings

**Value**

A ggplot object of the ggplot2 package that contains the library complexity plot.

**Examples**

```r
library("DESeq2")
set.seed(1)
dds <- makeExampleDESeqDataSet()
plot_library_complexity(dds)
```

---

plot_loadings

*Plot loadings of a principal component*

**Description**

Plot loadings of a principal component

**Usage**

```r
plot_loadings(
  pca_res,
  PC = 1,
  square = FALSE,
  color_by = NULL,
  annotate_top_n = 0,
  highlight_genes = NULL,
  show_plot = TRUE
)
```

**Arguments**

- **pca_res**: A result returned from `plot_pca()`
- **PC**: Number of the principal component to plot
- **square**: Whether to plot squared loadings. The squared loading is equal to the fraction of variance explained by the respective feature in the given principal component.
- **color_by**: Variable (column in `pca_res$loadings`) to color points by. Can also be 'pc_sign' to color by the sign of the loading (useful in combination with the square = TRUE parameter).
- **annotate_top_n**: Annotate the top n features with positive or negative loading
- **highlight_genes**: Vector of gene names or gene IDs to highlight on the plot (overwrites top_n annotation)
- **show_plot**: Whether to show the plot
Value

The function displays the loadings plot and returns invisible a list of the plot, the data.frame of the PCA loadings.

Examples

```r
set.seed(1)
data <- matrix(rnorm(100*6), ncol=6)
data <- t(t(data)+c(-1, -1.1, -1.2, 1, 1.1, 1.2))
pca_res <- plot_pca(data)
plot_loadings(pca_res)
```

Description

MA-plot of a differential testing result

Usage

```r
plot_ma(de_res, dds, annotate_top_n = 5, highlight_genes = NULL)
```

Arguments

de_res An object returned by DESeq2::results() or DESeq2::lfcShrink()
dds The DESeqDataSet that was used to build the `de_res` object. This is needed for gene name annotation.
annotate_top_n Annotate the top n significant genes by fold change (up- and down-regulated)
highlight_genes Vector of gene names or gene IDs to highlight on the plot (overwrites top_n annotation)

Value

A ggplot object of the ggplot2 package that contains the MA-plot. The plot shows three classes of points: Light gray points are genes with low counts that are removed from the analysis by independent filtering. Darker gray points are not significant genes that show a density map to visualize where the majority of non-significant points are located. Finally, red point show significant genes.
Examples

library("DESeq2")
set.seed(1)
dds <- makeExampleDESeqDataSet(n=1500, m=6, betaSD=.3, interceptMean=6)
rowData(dds)$gene_name <- rownames(dds)
dds <- DESeq(dds)
de_res <- results(dds)
de_res <- results(dds)
plot_ma(de_res, dds)

plot_pca

Plot results of a principal component analysis

Description

Plot results of a principal component analysis

Usage

plot_pca(
  obj,
  PC_x = 1,
  PC_y = 2,
  n_feats = 500,
  scale_feats = FALSE,
  na_frac = 0.3,
  metadata = NULL,
  color_by = NULL,
  shape_by = NULL,
  point_alpha = 0.7,
  point_rel_size = 2,
  show_plot = TRUE,
  rasterise = FALSE,
  ...
)

Arguments

obj A (features x samples) matrix or SummarizedExperiment object
PC_x The PC to show on the x-axis.
PC_y The PC to show on the y-axis.
n_feats Number of top-variable features to include.
scale_feats Whether to scale the features.
na_frac Only consider features with the stated maximum fraction of NAs or NaNs. NA/NaNs will be mean-imputed for PCA.
metadata A data.frame used for annotating samples. rownames(metadata) must match colnames(obj).

color_by Variable by which to color points. Must be a column in metadata or in colData(obj). Alternatively, it can be the name of a feature (a rowname of obj) or a gene name (an element of rowData(obj)$gene_name).

shape_by Variable by which to color points. Must be a column in metadata or in colData(obj).

point_alpha alpha value of geom_point()

point_rel_size relative size of geom_point()

show_plot Whether to show the plot or not

rasterise Whether to rasterise the point, using ggrastr.

... Other parameters passed on to ggrastr::rasterize

Details

If the metadata or colData of obj contain a column colname, this column will be removed in the $pca_data slot, because this column contains the colnames of the data matrix. Similarly, for the $loadings slot, the column rowname is reserved for the rownames of the data matrix.

Value

The function displays the plot and returns invisible a list of the plot, the data.frame to make the plot, the vector of percentages of variance explained and the loadings matrix.

Examples

```
set.seed(1)
data <- matrix(rnorm(100*6), ncol=6)
data <- t(t(data)+c(-1, -1.1, -1.2, 1, 1.1, 1.2))
plot_pca(data)
```

Description

Plot matrix of PCA scatter plots

Usage

```
plot_pca_scatters(
obj,
n_PCs = min(10, nrow(obj), ncol(obj)),
show_var_exp = T,
n_feats = 500,
scale_feats = FALSE,
)```
Arguments

obj A (features x samples) matrix or SummarizedExperiment object
n_PCs Number of principal components to plot
show_var_exp Whether to show a plot of the percentage of variance explained by each PC in the bottom left corner.
n_feats Number of top-variable features to include.
scale_feats Whether to scale the features.
na_frac Only consider features with the stated maximum fraction of NAs or NaNs. NA/NaN values will be mean-imputed for PCA.
metadata A data.frame used for annotating samples. rownames(metadata) must match colnames(obj).
color_by Variable by which to color points. Must be a column in metadata or in colData(obj).
Alternatively, it can be the name of a feature (a rowname of obj) or a gene name (an element of rowData(obj)$gene_name).
shape_by Variable by which to color points. Must be a column in metadata or in colData(obj).
point_alpha Alpha value of geom_point()
point_rel_size Relative size of geom_point()
transpose Whether to transpose the whole matrix of scatter plots
rasterise Whether to rasterise the points using ggrastr.
... Other parameters passed on to ggrastr::rasterise

Value

The function displays the scatter plots of the PCs

Examples

```
set.seed(1)
data <- matrix(rnorm(100*6), ncol=6)
data <- t(t(data)+c(-1, -1.1, -1.2, 1, 1.1, 1.2))
plot_pca_scatters(data)
```

```
plot_sample_clustering

Plot clustering of samples in a distance heatmap

Description

Plot clustering of samples in a distance heatmap

Usage

plot_sample_clustering(
  se,
  n_feats = 500,
  anno_vars = NULL,
  anno_title = "group",
  distance = "euclidean",
  ...
)

Arguments

se A SummarizedExperiment object.
n_feats Number of top-variable features (genes) to consider
anno_vars Character vector of columns in colData(se) to annotate samples
anno_title The title of the color legend for anno_vars
distance The type of distance metric to consider. Either 'euclidean', 'pearson' or 'spearman'
... Other arguments passed on to ComplexHeatmap::Heatmap()

Value

A Heatmap-class object of the ComplexHeatmap package that contains the heatmap of pairwise sample distances.

Examples

library("DESeq2")
dds <- makeExampleDESeqDataSet(m=8, interceptMean=10)
vsd <- vst(dds)
plot_sample_clustering(vsd)
MA plots of samples

**Description**

For each level of the grouping variable, the gene-wise median over all samples is computed to obtain a reference sample. Then, each sample is plotted against the reference.

**Usage**

```r
plot_sample_MAs(vsd, group, y_lim = 3, rasterise = FALSE, ...)
```

**Arguments**

- `vsd`: An object generated by `DESeq2::vst()`
- `group`: A grouping variable, must be a column of `colData(vsd)`
- `y_lim`: Y-axis limits, the axis will run from `-y_lim` to `y_lim`
- `rasterise`: Whether to rasterise the points using `ggrastr`
- `...`: Other parameters passed on to `ggrastr::rasterise`

**Value**

A list of ggplot objects of the ggplot2 package, with each element corresponding to one MA-plot.

**Examples**

```r
library("DESeq2")
set.seed(1)
.dds <- makeExampleDESeqDataSet(n=1000, m=4, interceptMean=10)
colData(dds)$type <- c("A","A","B","B")
vsd <- vst(dds)
plot_sample_MAs(vsd, group="type")
```

---

Plot total counts per sample

**Description**

Plot the distribution of the total number of counts per sample as histogram.

**Usage**

```r
plot_total_counts(dds, n_bins = 50)
```
Arguments

  dds  A DESeqDataSet
  n_bins  Number of histogram bins

Value

  A ggplot object of the ggplot2 package that contains the histogram of total counts per sample.

Examples

  library("DESeq2")
  set.seed(1)
  dds <- makeExampleDESeqDataSet(m=30)
  plot_total_counts(dds)

Description

  For the given level, the gene-wise median over all samples is computed to obtain a reference sample.
  Then, each sample is plotted against the reference as MA-plot.

Usage

  plot_within_level_sample_MAs(
    vsd,
    group,
    level,
    y_lim = 4,
    rasterise = FALSE,
    ...
  )

Arguments

  vsd  An object generated by DESeq2::vst()
  group  A grouping variable, must be a column of colData(vsd)
  level  A level of the grouping variable
  y_lim  Y-axis limits, the axis will run from -y_lim to y_lim
  rasterise  Whether to rasterise the points using ggrastr.
  ...  Other parameters passed on to ggrastr::rasterise
Value

A list of ggplot objects of the ggplot2 package that contains for each sample of the specified level the sample vs reference MA-plot.

Examples

```r
library("DESeq2")
set.seed(1)
.dds <- makeExampleDESeqDataSet(n=1000, m=4, interceptMean=10)
colData(dds)$type <- c("A","A","B","B")
vsd <- vst(dds)
plot_within_level_sample_MAs(vsd, group="type", level="A")
```

Description

This function takes a list of plots as input and makes a pdf with \texttt{ncol} x \texttt{nrow} plots per page.

Usage

```r
save_plots_to_pdf(
    plots,
    file = "plots.pdf",
    ncol, 
    nrow, 
    subfig_width = subfig_height * 16/9, 
    subfig_height = 2.5, 
    legend_position = "original"
)
```

Arguments

- **plots**: List of plots that is passed to the \texttt{plotlist} argument of \texttt{cowplot::plot_grid}
- **file**: file where the plots are saved
- **ncol**: number of columns per page for the grid of plots
- **nrow**: number of rows per page for the grid of plots
- **subfig_width**: width of a plot of the grid in inches
- **subfig_height**: height of a plot of the grid in inches
- **legend_position**: either 'original' if the original legend of each sub-plot is shown, 'none', if no legend should be shown in any of the sub-plots, 'bottom', if no legend should be shown in the sub plots and one shared legend at the bottom or 'right', which is same as 'bottom', but shown on the right
Value

The function returns nothing but is called for its side effect, which is to save a pdf of plots to the filesystem.

Examples

```r
library("ggplot2")
manuf <- unique(mpg$manufacturer)
plots <- lapply(manuf, function(x){
df <- mpg[mpg$manufacturer==x,]
  ggplot(df, aes(cty, hwy)) +
    geom_point() +
    labs(title=x)
})
save_plots_to_pdf(plots, ncol=3, nrow=2)
```

T47D  
*The T47D cell line data of RNA-seq experiment GSE89888*

Description

The dataset contains the read counts of experiment GSE89888 in which T47D cells with different mutation statuses were treated with E2 (estradiol) or vehicle.

Usage

T47D

Format

A DESeqDataSet with 43576 rows (of genes) and 24 columns (of samples).

Source

doi:10.1101/2021.05.21.445138
Differential expression results corresponding to the T47D data set.

### Description
Differential expression results corresponding to the T47D data set.

### Usage
`T47D_diff_testing`

### Format
A DESeqResults object with 36562 rows and 3 columns.

### Source
See the 'data' vignette on how to reproduce this object.
Index

* datasets
  T47D, 18
  T47D_diff_testing, 19

all_numeric, 2

filter_genes, 3

gene_id, 3

make_dds, 4

mean_sd_plot, 5

plot_biotypes, 5

plot_chromosome, 6

plot_gene, 7

plot_gene_detection, 8

plot_library_complexity, 8

plot_loadings, 9

plot_ma, 10

plot_pca, 11

plot_pca_scatters, 12

plot_sample_clustering, 14

plot_sample_MAs, 15

plot_total_counts, 15

plot_within_level_sample_MAs, 16

save_plots_to_pdf, 17

T47D, 18

T47D_diff_testing, 19