

# Package ‘BatchQC’

March 17, 2023

**Type** Package

**Title** Batch Effects Quality Control Software

**Version** 1.26.0

**Date** 2020-10-11

**Maintainer** Solaiappan Manimaran <manimaran\_1975@hotmail.com>

**Description** Sequencing and microarray samples often are collected or processed in multiple batches or at different times. This often produces technical biases that can lead to incorrect results in the downstream analysis. BatchQC is a software tool that streamlines batch preprocessing and evaluation by providing interactive diagnostics, visualizations, and statistical analyses to explore the extent to which batch variation impacts the data. BatchQC diagnostics help determine whether batch adjustment needs to be done, and how correction should be applied before proceeding with a downstream analysis. Moreover, BatchQC interactively applies multiple common batch effect approaches to the data, and the user can quickly see the benefits of each method. BatchQC is developed as a Shiny App. The output is organized into multiple tabs, and each tab features an important part of the batch effect analysis and visualization of the data. The BatchQC interface has the following analysis groups: Summary, Differential Expression, Median Correlations, Heatmaps, Circular Dendrogram, PCA Analysis, Shape, ComBat and SVA.

**Author** Solaiappan Manimaran <manimaran\_1975@hotmail.com>, W. Evan Johnson <wej@bu.edu>, Heather Selby <selbyh@bu.edu>, Claire Ruberman <claireruberman@gmail.com>, Kwame Okrah <kwame.okrah@gmail.com>, Hector Corrada Bravo <hcorrada@gmail.com>

**URL** <https://github.com/mani2012/BatchQC>

**BugReports** <https://github.com/mani2012/BatchQC/issues>

**License** GPL (>= 2)

**Depends** R (>= 3.5.0)

**Collate** 'simulate\_data.R' 'heatmap.R' 'pca.R' 'batchtest.R'  
'batchQC.R' 'correlation.R' 'utils.R' 'sva.R' 'Circos.R'  
'shapeAnalysis.R' 'lmlfitC.R'

**Suggests** testthat

**Imports** utils, rmarkdown, knitr, pander, gplots, MCMCpack, shiny, sva,  
corpcor, moments, matrixStats, ggvis, heatmaply, reshape2,  
limma, grDevices, graphics, stats, methods, Matrix

**biocViews** BatchEffect, GraphAndNetwork, Microarray,  
PrincipalComponent, Sequencing, Software, Visualization,  
QualityControl, RNASeq, Preprocessing, DifferentialExpression,  
ImmunoOncology

**SystemRequirements** pandoc (<http://pandoc.org/installing.html>) for  
generating reports from markdown files.

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Encoding** UTF-8

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

**git\_branch** RELEASE\_3\_16

**git\_last\_commit** c593445

**git\_last\_commit\_date** 2022-11-01

**Date/Publication** 2023-03-17

## R topics documented:

batchQC . . . . .	3
BatchQCOut-class . . . . .	4
batchQC_analyze . . . . .	5
batchqc_circosplot . . . . .	6
batchQC_condition_adjusted . . . . .	6
batchqc_correlation . . . . .	7
batchqc_corscatter . . . . .	8
batchqc_explained_variation . . . . .	9
batchQC_filter_genes . . . . .	9
batchQC_fsva_adjusted . . . . .	10
batchqc_heatmap . . . . .	11
batchQC_num.sv . . . . .	12
batchqc_pca . . . . .	12
batchqc_pca_svd . . . . .	13
batchqc_pc_explained_variation . . . . .	14
batchQC_shapeVariation . . . . .	15
batchQC_sva . . . . .	16

batchQC_svregress_adjusted . . . . .	16
batchtest . . . . .	17
combatPlot . . . . .	18
example_batchqc_data . . . . .	19
getShinyInput . . . . .	20
getShinyInputCombat . . . . .	20
getShinyInputOrig . . . . .	21
getShinyInputSVA . . . . .	21
getShinyInputSVAf . . . . .	22
getShinyInputSVAr . . . . .	22
gnormalize . . . . .	23
lmFitC . . . . .	23
log2CPM . . . . .	24
makeSVD . . . . .	25
pcRes . . . . .	25
plotPC . . . . .	26
plot_genewise_moments . . . . .	26
plot_samplewise_moments . . . . .	27
protein_example_data . . . . .	28
rnaseq_sim . . . . .	28
setShinyInput . . . . .	30
setShinyInputCombat . . . . .	30
setShinyInputOrig . . . . .	31
setShinyInputSVA . . . . .	31
setShinyInputSVAf . . . . .	32
setShinyInputSVAr . . . . .	32

**Index****33**


---

batchQC	<i>Checks for presence of batch effect and creates a html report with information including whether the batch needs to be adjusted</i>
---------	--

---

**Description**

Checks for presence of batch effect and creates a html report with information including whether the batch needs to be adjusted

**Usage**

```
batchQC(
  dat,
  batch,
  condition = NULL,
  report_file = "batchqc_report.html",
  report_dir = ".",
  report_option_binary = "11111111",
  view_report = FALSE,
```

```

    interactive = TRUE,
    batchqc_output = FALSE,
    log2cpm_transform = FALSE
  )

```

### Arguments

<code>dat</code>	Given data or simulated data from <code>rnaseq_sim()</code>
<code>batch</code>	Batch covariate
<code>condition</code>	Covariates or conditions of interest besides batch
<code>report_file</code>	Output report file name
<code>report_dir</code>	Output report directory path
<code>report_option_binary</code>	9 bits Binary String representing the plots to display and hide in the report
<code>view_report</code>	when TRUE, opens the report in a browser
<code>interactive</code>	when TRUE, opens the interactive shinyApp
<code>batchqc_output</code>	when TRUE, creates BatchQCout object in <code>batchqc_output.rda</code> R object file
<code>log2cpm_transform</code>	when TRUE, transforms the data using log2CPM - log2 Counts Per Million transformation function

### Value

outputfile Report file generated by batchQC

### Examples

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
batchQC(data.matrix, batch=batch, condition=condition, view_report=FALSE,
  interactive=FALSE)

```

---

BatchQCout-class

*The BatchQC output class to output BatchQC results*

---

### Description

Contains all currently-supported BatchQC output data classes:

## Details

slots:

**batchqc\_ev** a single object of class list

**pca** a single object of S3 class prcomp

---

batchQC_analyze	<i>Checks for presence of batch effect and reports whether the batch needs to be adjusted</i>
-----------------	---

---

## Description

Checks for presence of batch effect and reports whether the batch needs to be adjusted

## Usage

```
batchQC_analyze(data.matrix, batch, mod = NULL)
```

## Arguments

data.matrix	Given data or simulated data from <code>rnaseq_sim()</code>
batch	Batch covariate
mod	Model matrix for outcome of interest and other covariates besides batch

## Value

pca Principal Components Analysis object of the data

## Examples

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchQC_analyze(data.matrix, batch, mod=modmatrix)
```

batchqc\_circosplot      *Produce Circos plot*

---

**Description**

Produce Circos plot

**Usage**

```
batchqc_circosplot(dat, batch, AggMethod)
```

**Arguments**

dat	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
AggMethod	Aggregation Method

**Value**

Generates Circular Dendrogram plot for the given data

**Examples**

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
batchqc_circosplot(data.matrix, batch=batch, AggMethod='complete')
```

---

batchQC\_condition\_adjusted

*Returns adjusted data after remove the variation across conditions*

---

**Description**

Returns adjusted data after remove the variation across conditions

**Usage**

```
batchQC_condition_adjusted(data.matrix, batch, condition)
```

**Arguments**

data.matrix      Given data or simulated data from rnaseq\_sim()  
batch             Batch covariate  
condition         Condition covariate of interest

**Value**

Adjusted data after remove the variation across conditions

**Examples**

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
batchQC_condition_adjusted(data.matrix, batch, condition)
```

---

batchqc\_correlation      *Produce correlation heatmap plot*

---

**Description**

Produce correlation heatmap plot

**Usage**

```
batchqc_correlation(data.matrix, batch, mod = NULL)
```

**Arguments**

data.matrix      Given data or simulated data from rnaseq\_sim()  
batch             Batch covariate  
mod               Model matrix for outcome of interest and other covariates besides batch

**Value**

Correlation heatmap plot

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchqc_correlation(data.matrix, batch, mod=modmatrix)

```

---

batchqc\_corscatter      *Produce Median Correlation plot*

---

**Description**

Produce Median Correlation plot

**Usage**

```
batchqc_corscatter(data.matrix, batch, mod = NULL)
```

**Arguments**

data.matrix	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
mod	Model matrix for outcome of interest and other covariates besides batch

**Value**

Median Correlation plot

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchqc_corscatter(data.matrix, batch, mod=modmatrix)

```



---

batchqc\_explained\_variation

*Returns a list of explained variation by batch and condition combinations*

---

### Description

Returns a list of explained variation by batch and condition combinations

### Usage

```
batchqc_explained_variation(data.matrix, condition, batch)
```

### Arguments

data.matrix	Given data or simulated data from rnaseq_sim()
condition	Condition covariate of interest
batch	Batch covariate

### Value

List of explained variation by batch and condition

### Examples

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
batchqc_explained_variation(data.matrix, condition, batch)
```

---

batchQC\_filter\_genes *Returns a dataset after filtering genes of zero variance across batch and condition combinations*

---

### Description

Returns a dataset after filtering genes of zero variance across batch and condition combinations

### Usage

```
batchQC_filter_genes(data.matrix, batch, condition)
```

**Arguments**

data.matrix	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
condition	Condition covariate of interest

**Value**

Filtered dataset after filtering genes of zero variance across batch and condition combinations

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
filtered.data <- batchQC_filter_genes(data.matrix, batch, condition)

```

---

batchQC\_fsva\_adjusted *Use frozen surrogate variable analysis to remove the surrogate variables inferred from sva*

---

**Description**

Use frozen surrogate variable analysis to remove the surrogate variables inferred from sva

**Usage**

```
batchQC_fsva_adjusted(data.matrix, modmatrix, sva.object)
```

**Arguments**

data.matrix	Given data or simulated data from rnaseq_sim()
modmatrix	Model matrix for outcome of interest and other covariates besides batch
sva.object	SVA object

**Value**

Frozen Surrogate variables adjusted data

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
sva.object <- batchQC_sva(data.matrix, mod=modmatrix)
batchQC_fsva_adjusted(data.matrix, modmatrix, sva.object)

```

---

batchqc_heatmap	<i>Produce heatmap plots for the given data</i>
-----------------	---

---

**Description**

Produce heatmap plots for the given data

**Usage**

```
batchqc_heatmap(data.matrix, batch, mod = NULL, max_display = 50)
```

**Arguments**

data.matrix	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
mod	Model matrix for outcome of interest and other covariates besides batch
max_display	Maximum number of rows to display in heat map

**Value**

Heatmap plots for the given data

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchqc_heatmap(data.matrix, batch, mod=modmatrix)

```

---

batchQC_num.sv	<i>Returns the number of surrogate variables to estimate in the model using a permutation based procedure</i>
----------------	---

---

**Description**

Returns the number of surrogate variables to estimate in the model using a permutation based procedure

**Usage**

```
batchQC_num.sv(data.matrix, modmatrix)
```

**Arguments**

data.matrix	Given data or simulated data from <code>rnaseq_sim()</code>
modmatrix	Model matrix for outcome of interest and other covariates besides batch

**Value**

Number of Surrogate variables found

**Examples**

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchQC_num.sv(data.matrix, modmatrix)
```

---

batchqc_pca	<i>Performs principal component analysis and produces plot of the first two principal components</i>
-------------	--

---

**Description**

Performs principal component analysis and produces plot of the first two principal components

**Usage**

```
batchqc_pca(data.matrix, batch, mod = NULL)
```

**Arguments**

`data.matrix`      Given data or simulated data from `rnaseq_sim()`  
`batch`              Batch covariate  
`mod`                Model matrix for outcome of interest and other covariates besides batch

**Value**

PCA object from principal component analysis

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchqc_pca(data.matrix, batch, mod=modmatrix)

```

---

<code>batchqc_pca_svd</code>	<i>Performs PCA svd variance decomposition and produces plot of the first two principal components</i>
------------------------------	--

---

**Description**

Performs PCA svd variance decomposition and produces plot of the first two principal components

**Usage**

```
batchqc_pca_svd(data.matrix, batch, mod = NULL)
```

**Arguments**

`data.matrix`      Given data or simulated data from `rnaseq_sim()`  
`batch`              Batch covariate  
`mod`                Model matrix for outcome of interest and other covariates besides batch

**Value**

res PCA list with two components v and d.

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchqc_pca_svd(data.matrix, batch, mod=modmatrix)

```

---

batchqc\_pc\_explained\_variation

*Returns explained variation for each principal components*

---

**Description**

Returns explained variation for each principal components

**Usage**

```
batchqc_pc_explained_variation(pcs, vars, condition, batch)
```

**Arguments**

pcs	Principal components in the given data
vars	Variance of the Principal components in the given data
condition	Condition covariate of interest
batch	Batch covariate

**Value**

Explained variation table for each principal components

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)

```

```
pca <- batchqc_pca(data.matrix, batch, mod=modmatrix)
pcs <- t(data.frame(pca$x))
batchqc_pc_explained_variation(pcs, pca$sdev^2, condition, batch)
```

---

batchQC\_shapeVariation

*Perform Mean and Variance batch variation analysis*

---

## Description

Perform Mean and Variance batch variation analysis

## Usage

```
batchQC_shapeVariation(
  data,
  groups,
  plot = FALSE,
  groupCol = NULL,
  robustSample = FALSE,
  robustGene = FALSE
)
```

## Arguments

data	Given data
groups	a character vector indicating sample group membership
plot	Indicate whether to generate plot
groupCol	group color
robustSample	Indicate whether to use robust sample-wise test
robustGene	Indicate whether to use robust gene-wise test

## Value

Mean and Variance batch variation Overall and Pairwise p-values

## Examples

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
batchQC_shapeVariation(data.matrix, groups=batch)
```

---

batchQC_sva	<i>Estimate the surrogate variables using the 2 step approach proposed by Leek and Storey 2007</i>
-------------	--

---

**Description**

Estimate the surrogate variables using the 2 step approach proposed by Leek and Storey 2007

**Usage**

```
batchQC_sva(data.matrix, modmatrix)
```

**Arguments**

data.matrix	Given data or simulated data from rnaseq_sim()
modmatrix	Model matrix for outcome of interest and other covariates besides batch

**Value**

Surrogate variables analysis object

**Examples**

```
nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
batchQC_sva(data.matrix, modmatrix)
```

---

batchQC_svregress_adjusted	<i>Regress the surrogate variables out of the expression data</i>
----------------------------	---

---

**Description**

Regress the surrogate variables out of the expression data

**Usage**

```
batchQC_svregress_adjusted(data.matrix, modmatrix, sva.object)
```



**Arguments**

`data.matrix`      Given data or simulated data from `rnaseq_sim()`  
`modmatrix`        Model matrix for outcome of interest and other covariates besides batch  
`sva.object`        SVA object

**Value**

Surrogate variables regress adjusted data

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
sva.object <- batchQC_sva(data.matrix, mod=modmatrix)
batchQC_svregress_adjusted(data.matrix, modmatrix, sva.object)

```

---

batchtest

*Performs test to check whether batch needs to be adjusted*

---

**Description**

Performs test to check whether batch needs to be adjusted

**Usage**

```
batchtest(pca, batch, mod = NULL)
```

**Arguments**

`pca`                PCA object from principal component analysis  
`batch`             Batch covariate  
`mod`                Model matrix for outcome of interest and other covariates besides batch

**Value**

Summary of linear regression of first five principal components

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
pca <- batchqc_pca(data.matrix, batch, mod=modmatrix)
batchtest(pca, batch, mod=modmatrix)

```

---

combatPlot

*Adjust for batch effects using an empirical Bayes framework ComBat allows users to adjust for batch effects in datasets where the batch covariate is known, using methodology described in Johnson et al. 2007. It uses either parametric or non-parametric empirical Bayes frameworks for adjusting data for batch effects. Users are returned an expression matrix that has been corrected for batch effects. The input data are assumed to be cleaned and normalized before batch effect removal.*

---

**Description**

Adjust for batch effects using an empirical Bayes framework ComBat allows users to adjust for batch effects in datasets where the batch covariate is known, using methodology described in Johnson et al. 2007. It uses either parametric or non-parametric empirical Bayes frameworks for adjusting data for batch effects. Users are returned an expression matrix that has been corrected for batch effects. The input data are assumed to be cleaned and normalized before batch effect removal.

**Usage**

```
combatPlot(dat, batch, mod = NULL, par.prior = TRUE, prior.plots = TRUE)
```

**Arguments**

dat	Genomic measure matrix (dimensions probe x sample) - for example, expression matrix
batch	Batch covariate (only one batch allowed)
mod	Model matrix for outcome of interest and other covariates besides batch
par.prior	(Optional) TRUE indicates parametric adjustments will be used, FALSE indicates non-parametric adjustments will be used
prior.plots	(Optional)TRUE give prior plots with black as a kernel estimate of the empirical batch effect density and red as the parametric

**Value**

data A probe x sample genomic measure matrix, adjusted for batch effects.

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
pdata <- data.frame(batch, condition)
mod = model.matrix(~as.factor(condition), data = pdata)
combatPlot(data.matrix, batch, mod=mod)

```

---

example\_batchqc\_data *Batch and Condition indicator for signature data captured when activating different growth pathway genes in human mammary epithelial cells.*

---

**Description**

This data consists of three batches and ten different conditions corresponding to control and nine different pathways

This data consists of three batches and ten different conditions corresponding to control and nine different pathways

**Usage**

```

batch_indicator
signature_data

```

**Format**

A data frame with 89 rows and 2 variables:

**V1** Batch Indicator

**V2** Condition (Pathway) Indicator

A data frame with 18052 rows and 89 variables:

**Columns1-89** Control and Pathway activated samples

**rows1-18052** Genes 1-18052

**Value**

Batch indicator object  
Signature data

**Source**

GEO accession: GSE73628  
GEO accession: GSE73628

---

getShinyInput	<i>Getter function to get the shinyInput option</i>
---------------	---

---

**Description**

Getter function to get the shinyInput option

**Usage**

```
getShinyInput()
```

**Value**

shinyInput option

**Examples**

```
getShinyInput()
```

---

getShinyInputCombat	<i>Getter function to get the shinyInputCombat option</i>
---------------------	---

---

**Description**

Getter function to get the shinyInputCombat option

**Usage**

```
getShinyInputCombat()
```

**Value**

shinyInputCombat option

**Examples**

```
getShinyInputCombat()
```

---

`getShinyInputOrig`      *Getter function to get the shinyInputOrig option*

---

**Description**

Getter function to get the shinyInputOrig option

**Usage**

```
getShinyInputOrig()
```

**Value**

shinyInputOrig option

**Examples**

```
getShinyInputOrig()
```

---

`getShinyInputSVA`      *Getter function to get the shinyInputSVA option*

---

**Description**

Getter function to get the shinyInputSVA option

**Usage**

```
getShinyInputSVA()
```

**Value**

shinyInputSVA option

**Examples**

```
getShinyInputSVA()
```

---

getShinyInputSVAf      *Getter function to get the shinyInputSVAf option*

---

**Description**

Getter function to get the shinyInputSVAf option

**Usage**

```
getShinyInputSVAf()
```

**Value**

shinyInputSVAf option

**Examples**

```
getShinyInputSVAf()
```

---

getShinyInputSVAr      *Getter function to get the shinyInputSVAr option*

---

**Description**

Getter function to get the shinyInputSVAr option

**Usage**

```
getShinyInputSVAr()
```

**Value**

shinyInputSVAr option

**Examples**

```
getShinyInputSVAr()
```

---

gnormalize	<i>Perform Genewise Normalization of the given data matrix</i>
------------	--

---

**Description**

Perform Genewise Normalization of the given data matrix

**Usage**

```
gnormalize(dat)
```

**Arguments**

dat                    Given data matrix

**Value**

gnormdata Genewise Normalized data matrix

**Examples**

```
dat <- matrix(1:10, 2)
gnormdata <- gnormalize(dat)
```

---

lmFitC	<i>Fit linear model for each gene given a series of arrays. This is the standard lmFit function from limma package with the modification to accept an additional correlation matrix parameter option</i>
--------	--

---

**Description**

Fit linear model for each gene given a series of arrays. This is the standard lmFit function from limma package with the modification to accept an additional correlation matrix parameter option

**Usage**

```
lmFitC(
  object,
  design = NULL,
  ndups = 1,
  spacing = 1,
  block = NULL,
  correlation,
  cormatrix = NULL,
  weights = NULL,
  method = "ls",
  ...
)
```

**Arguments**

object	A matrix-like data object containing log-ratios or log-expression values for a series of arrays, with rows corresponding to genes and columns to samples. Any type of data object that can be processed by getEAWP is acceptable.
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates
ndups	positive integer giving the number of times each distinct probe is printed on each array.
spacing	positive integer giving the spacing between duplicate occurrences of the same probe, spacing=1 for consecutive rows.
block	vector or factor specifying a blocking variable on the arrays. Has length equal to the number of arrays. Must be NULL if ndups>2.
correlation	the inter-duplicate or inter-technical replicate correlation
cormatrix	the complete correlation matrix of the samples
weights	non-negative observation weights. Can be a numeric matrix of individual weights, of same size as the object expression matrix, or a numeric vector of array weights with length equal to ncol of the expression matrix, or a numeric vector of gene weights with length equal to nrow of the expression matrix.
method	fitting method; "ls" for least squares or "robust" for robust regression
...	other optional arguments to be passed to lm.series, gls.series or mrlm

**Value**

list containing log2(quantile counts per mil reads) and library sizes

---

log2CPM	<i>Compute log2(counts per mil reads) and library size for each sample</i>
---------	--

---

**Description**

Compute log2(counts per mil reads) and library size for each sample

**Usage**

```
log2CPM(qcounts, lib.size = NULL)
```

**Arguments**

qcounts	quantile normalized counts
lib.size	default is colsums(qcounts)

**Value**

list containing log2(quantile counts per mil reads) and library sizes



**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
data.matrix <- as.matrix(data.matrix)
log2CPM(data.matrix)

```

---

makeSVD	<i>Compute singular value decomposition</i>
---------	---

---

**Description**

Compute singular value decomposition

**Usage**

```
makeSVD(x)
```

**Arguments**

x                    matrix of genes by sample (ie. the usual data matrix)

**Value**

returns a list of svd components v and d

---

pcRes	<i>Compute variance of each principal component and how they correlate with batch and cond</i>
-------	--

---

**Description**

Compute variance of each principal component and how they correlate with batch and cond

**Usage**

```
pcRes(v, d, condition = NULL, batch = NULL)
```

**Arguments**

v                    from makeSVD  
d                    from makeSVD  
condition            factor describing experiment  
batch                factor describing batch

**Value**

A dataframe containig variance, cum. variance, cond.R-sqrd, batch.R-sqrd

---

plotPC	<i>Plot first 2 principal components</i>
--------	--

---

**Description**

Plot first 2 principal components

**Usage**

```
plotPC(v, d, ...)
```

**Arguments**

v	from makeSVD
d	from makeSVD
...	pass options to internal plot fct.

**Value**

a plot

---

plot_genewise_moments	<i>Visualize gene-wise moments</i>
-----------------------	------------------------------------

---

**Description**

Visualize gene-wise moments

**Usage**

```
plot_genewise_moments(data, batch, robust)
```

**Arguments**

data	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
robust	Boolean indicator of using robust (TRUE) or non-robust test (FALSE) in visualization

**Value**

Gene-wise moments

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
data_adjusted <- batchQC_condition_adjusted(data.matrix, batch, condition)
gene_moments <- plot_genewise_moments(data_adjusted, batch, robust=FALSE)

```

---

plot\_samplewise\_moments

*Visualize sample-wise moments*


---

**Description**

Visualize sample-wise moments

**Usage**

```
plot_samplewise_moments(data, batch, robust)
```

**Arguments**

data	Given data or simulated data from rnaseq_sim()
batch	Batch covariate
robust	Boolean indicator of using robust (TRUE) or non-robust test (FALSE) in visualization

**Value**

Sample-wise moments

**Examples**

```

nbatch <- 3
ncond <- 2
npercond <- 10
data.matrix <- rnaseq_sim(ngenes=50, nbatch=nbatch, ncond=ncond, npercond=
  npercond, basemean=10000, ggstep=50, bbstep=2000, ccstep=800,
  basedisp=100, bdispstep=-10, swvar=1000, seed=1234)
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
data_adjusted <- batchQC_condition_adjusted(data.matrix, batch, condition)
sample_moments <- plot_samplewise_moments(data_adjusted, batch, robust=FALSE)

```

---

protein\_example\_data    *Batch and Condition indicator for protein expression data*

---

### Description

This data consists of two batches and two conditions corresponding to case and control for the protein expression data

This data consists of two batches and two conditions corresponding to case and control

### Usage

protein\_sample\_info

protein\_data

### Format

A data frame with 24 rows and 4 variables:

**Arrayname** Array Name

**samplename** Sample Name

**Batch** Batch Indicator

**category** Condition (Case vs Control) Indicator

A data frame with 39 rows and 24 variables:

**Columns1-24** Control and Case samples

**rows1-39** Proteins 1-39

### Value

Protein data sample info

Protein data

---

rnaseq\_sim    *Generate simulated count data with batch effects for ngenes*

---

### Description

Generate simulated count data with batch effects for ngenes

**Usage**

```
rnaseq_sim(  
  ngenes = 50,  
  nbatch = 3,  
  ncond = 2,  
  npercond = 10,  
  basemean = 10000,  
  ggstep = 50,  
  bbstep = 2000,  
  ccstep = 800,  
  basedisp = 100,  
  bdispstep = 10,  
  swvar = 1000,  
  seed = 1000  
)
```

**Arguments**

ngenes	Number of genes to simulate
nbatch	Number of batches to simulate
ncond	Number of conditions to simulate
npercond	Number of samples per condition per batch to simulate
basemean	Base mean
ggstep	Gene to Gene step variation
bbstep	Batch to Batch step variation
ccstep	Condition to Condition step variation
basedisp	Base Dispersion
bdispstep	Batch to Batch Dispersion step variation
swvar	Sample-wise extra variation
seed	Random seed for reproducibility

**Value**

RNA Seq count data matrix

**Examples**

```
rnaseq_sim()  
rnaseq_sim(ngenes=100, nbatch=5, seed=1234)  
rnaseq_sim(ngenes=100, nbatch=3, ncond=2, npercond=10, basemean=10000,  
  ggstep=50, bbstep=20000, ccstep=8000, basedisp=100, bdispstep=10,  
  swvar=1000, seed=1234)
```

---

setShinyInput	<i>Setter function to set the shinyInput option</i>
---------------	---

---

**Description**

Setter function to set the shinyInput option

**Usage**

```
setShinyInput(x)
```

**Arguments**

x                    shinyInput option

**Value**

shinyInput option

**Examples**

```
setShinyInput(NULL)
```

---

setShinyInputCombat	<i>Setter function to set the shinyInputCombat option</i>
---------------------	---

---

**Description**

Setter function to set the shinyInputCombat option

**Usage**

```
setShinyInputCombat(x)
```

**Arguments**

x                    shinyInputCombat option

**Value**

shinyInputCombat option

**Examples**

```
setShinyInputCombat(NULL)
```

---

setShinyInputOrig      *Setter function to set the shinyInputOrig option*

---

**Description**

Setter function to set the shinyInputOrig option

**Usage**

```
setShinyInputOrig(x)
```

**Arguments**

x                      shinyInputOrig option

**Value**

shinyInputOrig option

**Examples**

```
setShinyInputOrig(NULL)
```

---

setShinyInputSVA      *Setter function to set the shinyInputSVA option*

---

**Description**

Setter function to set the shinyInputSVA option

**Usage**

```
setShinyInputSVA(x)
```

**Arguments**

x                      shinyInputSVA option

**Value**

shinyInputSVA option

**Examples**

```
setShinyInputSVA(NULL)
```

---

setShinyInputSVAF      *Setter function to set the shinyInputSVAF option*

---

**Description**

Setter function to set the shinyInputSVAF option

**Usage**

```
setShinyInputSVAF(x)
```

**Arguments**

x                      shinyInputSVAF option

**Value**

shinyInputSVAF option

**Examples**

```
setShinyInputSVAF(NULL)
```

---

setShinyInputSVAr      *Setter function to set the shinyInputSVAr option*

---

**Description**

Setter function to set the shinyInputSVAr option

**Usage**

```
setShinyInputSVAr(x)
```

**Arguments**

x                      shinyInputSVAr option

**Value**

shinyInputSVAr option

**Examples**

```
setShinyInputSVAr(NULL)
```



# Index

## \* datasets

- example\_batchqc\_data, [19](#)
- protein\_example\_data, [28](#)
  
- batch\_indicator (example\_batchqc\_data), [19](#)
- batchQC, [3](#)
- batchQC\_analyze, [5](#)
- batchqc\_circosplot, [6](#)
- batchQC\_condition\_adjusted, [6](#)
- batchqc\_correlation, [7](#)
- batchqc\_corscatter, [8](#)
- batchqc\_explained\_variation, [9](#)
- batchQC\_filter\_genes, [9](#)
- batchQC\_fsva\_adjusted, [10](#)
- batchqc\_heatmap, [11](#)
- batchQC\_num.sv, [12](#)
- batchqc\_pc\_explained\_variation, [14](#)
- batchqc\_pca, [12](#)
- batchqc\_pca\_svd, [13](#)
- batchQC\_shapeVariation, [15](#)
- batchQC\_sva, [16](#)
- batchQC\_svregress\_adjusted, [16](#)
- BatchQCout-class, [4](#)
- batchtest, [17](#)
  
- combatPlot, [18](#)
  
- example\_batchqc\_data, [19](#)
  
- getShinyInput, [20](#)
- getShinyInputCombat, [20](#)
- getShinyInputOrig, [21](#)
- getShinyInputSVA, [21](#)
- getShinyInputSVAf, [22](#)
- getShinyInputSVAr, [22](#)
- gnormalize, [23](#)
  
- lmFitC, [23](#)
- log2CPM, [24](#)
  
- makeSVD, [25](#)
  
- pcRes, [25](#)
- plot\_genewise\_moments, [26](#)
- plot\_samplewise\_moments, [27](#)
- plotPC, [26](#)
- protein\_data (protein\_example\_data), [28](#)
- protein\_example\_data, [28](#)
- protein\_sample\_info (protein\_example\_data), [28](#)
  
- rnaseq\_sim, [28](#)
  
- setShinyInput, [30](#)
- setShinyInputCombat, [30](#)
- setShinyInputOrig, [31](#)
- setShinyInputSVA, [31](#)
- setShinyInputSVAf, [32](#)
- setShinyInputSVAr, [32](#)
- signature\_data (example\_batchqc\_data), [19](#)