

# Package ‘scQTLtools’

April 7, 2025

**Type** Package

**Title** An R package for single-cell eQTL analysis and visualization

**Version** 0.99.12

**Description** This package specializes in analyzing and visualizing eQTL at the single-cell level. It can read gene expression matrices or Seurat data, or SingleCellExperiment object along with genotype data. It offers a function for cis-eQTL analysis to detect eQTL within a given range, and another function to fit models with three methods. Using this package, users can also generate single-cell level visualization result.

**Depends** R (>= 4.4.1.0)

**Imports** ggplot2(>= 3.5.1), Matrix (>= 1.7-0), stats (>= 4.4.1), progress(>= 1.2.3), stringr(>= 1.5.1), dplyr(>= 1.1.4), SeuratObject(>= 5.0.2), methods(>= 4.4.1), magrittr(>= 2.0.3), patchwork(>= 1.2.0), DESeq2 (>= 1.45.3), VGAM (>= 1.1-11), limma (>= 3.61.9), biomaRt(>= 2.61.3), gamlss (>= 5.4-22), SingleCellExperiment(>= 1.27.2), SummarizedExperiment(>= 1.32.0), GOSemSim(>= 2.31.2)

**Suggests** BiocStyle, knitr, rmarkdown, org.Hs.eg.db, org.Mm.eg.db, org.Ce.eg.db, org.At.tair.db, testthat (>= 3.2.1.1)

**License** MIT + file LICENSE

**URL** <https://github.com/XFWuCN/scQTLtools>

**VignetteBuilder** knitr

**BugReports** <https://github.com/XFWuCN/scQTLtools/issues>

**biocViews** Software, GeneExpression, GeneticVariability, SNP, DifferentialExpression, GenomicVariation, VariantDetection, Genetics, FunctionalGenomics, SystemsBiology, Regression, SingleCell, Normalization, Visualization

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**LazyData** false

**Config/testthat/edition** 3

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

**git\_branch** devel

**git\_last\_commit** 58f5b0a

**git\_last\_commit\_date** 2025-03-26

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-06

**Author** Xiaofeng Wu [aut, cre, cph] (ORCID:  
<https://orcid.org/0009-0003-6254-5575>),  
 Xin Huang [aut, cph],  
 Jingtong Kang [com],  
 Siwen Xu [aut, cph]

**Maintainer** Xiaofeng Wu <1427972815@qq.com>

## Contents

scQTLtools-package . . . . .	3
adjust_pvalues . . . . .	4
buildZINB . . . . .	5
callQTL . . . . .	5
checkSNPList . . . . .	7
CPM_normalize . . . . .	8
createGeneLoc . . . . .	8
createQTLObject . . . . .	9
createSNPsLoc . . . . .	10
DESeq_normalize . . . . .	11
draw_boxplot . . . . .	12
draw_histplot . . . . .	12
draw_QTLplot . . . . .	13
draw_violinplot . . . . .	14
eQTLObject-class . . . . .	15
filterGeneSNP . . . . .	15
filter_by_abs_b . . . . .	16
get_cell_groups . . . . .	17
get_counts . . . . .	18
get_filter_data . . . . .	18
get_filter_data,eQTLObject-method . . . . .	19
get_model_info . . . . .	19
get_model_info,eQTLObject-method . . . . .	20
get_raw_data . . . . .	20
get_raw_data,eQTLObject-method . . . . .	21
get_result_info . . . . .	21
get_result_info,eQTLObject-method . . . . .	22
initialize_progress_bar . . . . .	22
limma_normalize . . . . .	23
linearModel . . . . .	24
load_biclassify_info . . . . .	25

load_biclassify_info,eQTLObject-method . . . . .	25
load_group_info . . . . .	26
load_group_info,eQTLObject-method . . . . .	26
load_species_info . . . . .	27
load_species_info,eQTLObject-method . . . . .	27
log_normalize . . . . .	28
normalizeGene . . . . .	28
plots_theme_opts . . . . .	29
poissonModel . . . . .	29
process_matrix . . . . .	31
remove_outliers . . . . .	31
set_filter_data . . . . .	32
set_filter_data,eQTLObject-method . . . . .	33
set_model_info . . . . .	34
set_model_info,eQTLObject-method . . . . .	34
set_raw_data . . . . .	35
set_raw_data,eQTLObject-method . . . . .	36
set_result_info . . . . .	36
set_result_info,eQTLObject-method . . . . .	37
show,eQTLObject-method . . . . .	38
testEQTL . . . . .	38
testGene . . . . .	39
testSeurat . . . . .	39
testSNP . . . . .	40
testSNP2 . . . . .	40
TPM_normalize . . . . .	41
visualizeQTL . . . . .	41
zinbModel . . . . .	42
<b>Index</b>	<b>44</b>

---

scQTLtools-package	<i>scQTLtools: An R package for single-cell eQTL analysis and visualization</i>
--------------------	---

---

## Description

This package specializes in analyzing and visualizing eQTL at the single-cell level. It can read gene expression matrices or Seurat data, or SingleCellExperiment object along with genotype data. It offers a function for cis-eQTL analysis to detect eQTL within a given range, and another function to fit models with three methods. Using this package, users can also generate single-cell level visualization result.

**Author(s)**

**Maintainer:** Xiaofeng Wu <1427972815@qq.com> ([ORCID](#)) [copyright holder]

Authors:

- Xin Huang <1097567240@qq.com> [copyright holder]
- Siwen Xu <siwxu@gdpu.edu.cn> [copyright holder]

Other contributors:

- Jingtong Kang <1203178107@qq.com> [compiler]

**See Also**

Useful links:

- <https://github.com/XFWuCN/scQTLtools>
- Report bugs at <https://github.com/XFWuCN/scQTLtools/issues>

---

adjust_pvalues	<i>Adjust p-values and perform threshold filtering based on the adjusted p-values.</i>
----------------	--

---

**Description**

Adjust p-values and perform threshold filtering based on the adjusted p-values.

**Usage**

```
adjust_pvalues(result, pAdjustMethod = "bonferroni", pAdjustThreshold = 0.05)
```

**Arguments**

result	Dataframe that contains gene-SNP pairs' information.
pAdjustMethod	Methods for p-value adjusting, one of "bonferroni", "holm", "hochberg", "holmel" or "BH". The default option is "bonferroni".
pAdjustThreshold	Only SNP-Gene pairs with adjusted p-values meeting the threshold will be displayed. Default by 0.05.

**Value**

A dataframe that has been adjusted and filtered, containing information on gene-SNP pairs.

**Examples**

```
example_data <- data.frame(
  gene = c("Gene1", "Gene2", "Gene3", "Gene4"),
  SNP = c("SNP1", "SNP2", "SNP3", "SNP4"),
  pvalue = c(0.001, 0.04, 0.03, 0.0005))
pAdjustMethod <- "BH"
pAdjustThreshold <- 0.05
adjusted_result <- adjust_pvalues(example_data, pAdjustMethod,
pAdjustThreshold)
```

---

buildZINB	<i>Build zinb model.</i>
-----------	--------------------------

---

**Description**

Build zinb model.

**Usage**

```
buildZINB(counts)
```

**Arguments**

counts            a vector for gene expression.

**Value**

Four parameters

**Examples**

```
data(testGene)
gene <- unlist(testGene[1, ])
result <- buildZINB(gene)
```

---

callQTL	<i>callQTL: Uncover single-cell eQTLs exclusively using scRNA-seq data. A function designed to identify eQTLs from scRNA-seq data.</i>
---------	--

---

**Description**

callQTL: Uncover single-cell eQTLs exclusively using scRNA-seq data. A function designed to identify eQTLs from scRNA-seq data.

**Usage**

```
callQTL(
  eQTLObject,
  gene_ids = NULL,
  downstream = NULL,
  upstream = NULL,
  gene_mart = NULL,
  snp_mart = NULL,
  pAdjustMethod = "bonferroni",
  useModel = "zinb",
  pAdjustThreshold = 0.05,
  logfcThreshold = 0.1
)
```

**Arguments**

eQTLObject	An S4 object of class eQTLObject.
gene_ids	A gene ID or a list of gene IDS.
downstream	Being used to match SNPs within a base range defined by the start position of genes.
upstream	Being used to match SNPs within a base range defined by the end position of genes.
gene_mart	An object of class Mart representing the BioMart database to connect to. If NULL, the function will use the Ensembl Gene BioMart.
snp_mart	An object of class Mart representing the BioMart database to connect to. If NULL, the function will use the Ensembl SNP BioMart.
pAdjustMethod	Methods for p-value adjusting, one of 'bonferroni', 'holm', 'hochberg', 'hommel' or 'BH'. Default by 'bonferroni'.
useModel	Model for fitting dataframe, one of 'poission', 'zinb', or 'linear'.
pAdjustThreshold	Only SNP-Gene pairs with adjusted p-values meeting the threshold will be displayed. Default by 0.05.
logfcThreshold	Represents the minimum beta threshold for fitting SNP-Gene pairs.

**Value**

A dataframe, each row describes eQTL discovering result of a SNP-Gene pair.

**Examples**

```
data(testEQTL)
library(biomaRt)
gene_mart <- useEnsembl(biomart = "genes",
                       dataset = "hsapiens_gene_ensembl",
                       mirror = 'asia')
snp_mart <- useEnsembl(biomart = "snps",
```

```

                                dataset = "hsapiens_snp",
                                mirror = 'asia')
eqtl <- callQTL(
  eQTLObject = testEQTl,
  gene_ids = NULL,
  downstream = NULL,
  upstream = NULL,
  gene_mart = gene_mart,
  snp_mart = snp_mart,
  pAdjustMethod = 'bonferroni',
  useModel = 'linear',
  pAdjustThreshold = 0.05,
  logfcThreshold = 0.025
)

```

---

checkSNPList	<i>Check if the SNP ids in the input genotype matrix are valid.</i>
--------------	---

---

### Description

Check if the SNP ids in the input genotype matrix are valid.

### Usage

```
checkSNPList(snpList, snp_mart = NULL, snpDataset = "hsapiens_snp")
```

### Arguments

snpList	a list of SNPs id.
snp_mart	An object of class 'Mart' representing the BioMart database connect to for SNPs. If provided, this should be a 'Mart' object obtained by calling 'useEnsembl()', which allows specifying a mirror in case of connection issues. If 'NULL', the function will create and use a 'Mart' object pointing to the Ensembl SNP BioMart using the specified 'snpDataset' and a default mirror.
snpDataset	A character string specifying the SNP dataset to use from ENSEMBL. The default is 'hsapiens_snp' for human SNPs.

### Value

SNP location dataframe

### Examples

```

data(testSNP2)
snpList <- rownames(testSNP2)
snpDataset <- 'hsapiens_snp'
snps_loc <- checkSNPList(snpList = snpList,
                        snpDataset = snpDataset)

```

---

CPM_normalize	<i>Normalize the gene expression matrix with CPM.</i>
---------------	---

---

**Description**

'CPM\_normalize()' scales an expression matrix using Counts Per Million (CPM) normalization, applying logarithm and scaling operations to adjust data.

**Usage**

```
CPM_normalize(expressionMatrix)
```

**Arguments**

expressionMatrix  
Input raw gene expression matrix.

**Value**

A gene expression matrix after normalized.

**Examples**

```
data(testGene)  
CPM_normalize(testGene)
```

---

createGeneLoc	<i>Create gene location dataframe.</i>
---------------	--

---

**Description**

Create gene location dataframe.

**Usage**

```
createGeneLoc(  
  geneList,  
  gene_mart = NULL,  
  geneDataset = "hsapiens_gene_ensembl",  
  OrgDb  
)
```



**Arguments**

geneList	A gene id or a list of genes id.
gene_mart	An object of class 'Mart' representing the BioMart database connect to for gene. If provided, this should be a 'Mart' object obtained by calling 'useEnsembl()', which allows specifying a mirror in case of connection issues. If 'NULL', the function will create and use a 'Mart' object pointing to the Ensembl Gene BioMart using the specified 'geneDataset' and a default mirror.
geneDataset	A character string specifying the gene dataset to use from ENSEMBL. The default is "hsapiens_gene_ensembl" for human genes.
OrgDb	OrgDb name:"org.Hs.eg.db", "org.Mm.eg.db".

**Value**

data.frame

**Examples**

```
data(testGene)
geneList <- rownames(testGene)
library(GOSemSim)
library(biomaRt)
OrgDb <- load_OrgDb("org.Hs.eg.db")
gene_mart <- useEnsembl(biomart = "genes",
                      dataset = "hsapiens_gene_ensembl",
                      mirror = 'asia')
gene_loc <- createGeneLoc(geneList = geneList,
                        gene_mart = gene_mart,
                        OrgDb = OrgDb)
```

---

createQTLObject	<i>createObject: Create the eQTLObject. We next create a S4 object. The object serves as a container that contains both data (like the count matrix) and meta.data.</i>
-----------------	---

---

**Description**

createObject: Create the eQTLObject. We next create a S4 object. The object serves as a container that contains both data (like the count matrix) and meta.data.

**Usage**

```
createQTLObject(
  snpMatrix,
  genedata,
  biClassify = FALSE,
  species = NULL,
  group = NULL,
  ...
)
```

**Arguments**

snpMatrix	A genotype matrix where each row is one variant and each column is one sample, and the scoring method is 0/1/2/3.
genedata	A gene expression matrix or a Seurat object, or a SingleCellExperiment object.
biClassify	The user chooses whether to convert the counting method of the snpMatrix to 0/1/2, TRUE indicates conversion, and FALSE indicates no conversion, default is no conversion.
species	The species that the user wants to select, human or mouse.
group	Provided by Seurat's meta.data, such as celltypes, cellstatus and so on. By default, it is NULL.
...	other parameters

**Value**

eQTLObject

**Examples**

```
data(testSNP)
data(testGene)
eqtl <- createQTLObject(snpMatrix = testSNP,
                        genedata = testGene,
                        biClassify = FALSE,
                        species = 'human',
                        group = NULL)
```

---

createSNPsLoc

*Create SNP location dataframe.*

---

**Description**

Create SNP location dataframe.

**Usage**

```
createSNPsLoc(snpList, snp_mart = NULL, snpDataset = "hsapiens_snp")
```

**Arguments**

snpList	a list of SNPs id.
snp_mart	An object of class 'Mart' representing the BioMart database connect to for SNPs. If provided, this should be a 'Mart' object obtained by calling 'useEnsembl()', which allows specifying a mirror in case of connection issues. If 'NULL', the function will create and use a 'Mart' object pointing to the Ensembl SNP BioMart using the specified 'snpDataset' and a default mirror.
snpDataset	A character string specifying the SNP dataset to use from ENSEMBL. The default is 'hsapiens_snp' for human SNPs.

**Value**

data.frame

**Examples**

```
snpList <- c('rs546', 'rs549', 'rs568', 'rs665', 'rs672')
library(biomaRt)
snp_mart <- useEnsembl(biomart = "snps",
                      dataset = "hsapiens_snp",
                      mirror = 'asia')
snp_loc <- createSNPsLoc(snpList = snpList,
                        snp_mart = snp_mart)
```

---

DESeq_normalize	<i>Normalize the gene expression matrix with DESeq.</i>
-----------------	---

---

**Description**

'DESeq\_normalize()' normalizes an expression matrix using the DESeq2 package.

**Usage**

```
DESeq_normalize(expressionMatrix)
```

**Arguments**

expressionMatrix  
Input raw gene expression matrix.

**Value**

A gene expression matrix after normalized.

**Examples**

```
data(testGene)
DESeq_normalize(testGene)
```

---

draw_boxplot	<i>Generate a boxplot of expression levels by SNP factor</i>
--------------	--

---

### Description

'draw\_boxplot()' creates a boxplot visualizing expression levels across different SNP factors in the dataframe. It uses ggplot2 to produce a plot with customizable aesthetics for clarity and presentation.

### Usage

```
draw_boxplot(df, unique_group)
```

### Arguments

df	Data frames listed as gene expression data, genotype data, and groups
unique_group	name of unique group

### Value

ggplot

### Examples

```
set.seed(123)
counts_Ref <- rnorm(50, mean = 10, sd = 2)
counts_Alt <- rnorm(50, mean = 12, sd = 2)
i <- rep("GroupA", 100)
unique_group <- unique(i)
dataframe <- data.frame(
  expression = c(counts_Ref, counts_Alt),
  snp = c(rep("REF", length(counts_Ref)),
          rep("ALT", length(counts_Alt))), group = i)
dataframe$snp <- factor(dataframe$snp, levels = c("REF", "ALT"))
draw_boxplot(df = dataframe, unique_group = unique_group)
```

---

draw_histplot	<i>Generate a hist plot of expression levels by SNP factor.</i>
---------------	---

---

### Description

'draw\_histplot()' generates histograms using ggplot2, displaying the distribution of expression values categorized by SNP type.

### Usage

```
draw_histplot(df, unique_group)
```

**Arguments**

df                    Data frames listed as gene expression data, genotype data, and groups  
unique\_group        name of unique group

**Value**

ggplot

**Examples**

```
set.seed(123)
counts_Ref <- rnorm(50, mean = 10, sd = 2)
counts_Alt <- rnorm(50, mean = 12, sd = 2)
i <- rep("GroupA", 100); unique_group <- unique(i)
dataframe <- data.frame(expression = c(counts_Ref, counts_Alt),
                        snp = c(rep("REF", length(counts_Ref)),
                               rep("ALT", length(counts_Alt))), group = i)
dataframe$snp <- factor(dataframe$snp, levels = c("REF", "ALT"))
draw_histplot(df = dataframe, unique_group = unique_group)
```

---

draw\_QTLplot                    *Create a combined plot with violin, boxplot, and scatter point overlay.*

---

**Description**

‘draw\_QTLplot()’ generates a combined plot using ggplot2, showing the distribution of expression values across different SNPs. It combines a violin plot, boxplot, and scatter points for each SNP category.

**Usage**

```
draw_QTLplot(df, unique_group)
```

**Arguments**

df                    Data frames listed as gene expression data, genotype data, and groups  
unique\_group        name of unique group

**Value**

ggplot

**Examples**

```

set.seed(123)
counts_Ref <- rnorm(50, mean = 10, sd = 2)
counts_Alt <- rnorm(50, mean = 12, sd = 2)
i <- rep("GroupA", 100);unique_group <- unique(i)
dataframe <- data.frame(expression = c(counts_Ref, counts_Alt),
                          snp = c(rep("REF", length(counts_Ref)),
                                  rep("ALT", length(counts_Alt))), group = i)
dataframe$snp <- factor(dataframe$snp, levels = c("REF", "ALT"))
draw_QTLplot(df = dataframe, unique_group = unique_group)

```

---

draw\_violinplot

*Generate a violin plot of expression levels by SNP factor*


---

**Description**

‘draw\_violinplot()’ creates a violin plot visualizing expression levels across different SNP factors in the dataframe. It uses ggplot2 to produce a plot with customizable aesthetics for clarity and presentation.

**Usage**

```
draw_violinplot(df, unique_group)
```

**Arguments**

df	Data frames listed as gene expression data, genotype data, and groups
unique_group	name of unique group

**Value**

```
ggplot
```

**Examples**

```

set.seed(123)
counts_Ref <- rnorm(50, mean = 10, sd = 2)
counts_Alt <- rnorm(50, mean = 12, sd = 2)
i <- rep("GroupA", 100);unique_group <- unique(i)
dataframe <- data.frame(expression = c(counts_Ref, counts_Alt),
                          snp = c(rep("REF", length(counts_Ref)),
                                  rep("ALT", length(counts_Alt))), group = i)
dataframe$snp <- factor(dataframe$snp, levels = c("REF", "ALT"))
draw_violinplot(df = dataframe, unique_group = unique_group)

```

---

eQTLObjct-class	<i>Class 'eQTLObjct' The eQTLObjct class is an R object designed to store data related to eQTL analysis, encompassing data lists, result data frames, and layers for biClassify, species, and grouping information.</i>
-----------------	---

---

### Description

Class 'eQTLObjct' The eQTLObjct class is an R object designed to store data related to eQTL analysis, encompassing data lists, result data frames, and layers for biClassify, species, and grouping information.

### Value

eQTLObjct

### Slots

rawData A gene expression dataframe, the row names represent gene IDs and the column names represent cell IDs.

filterData Gene expression matrix after normalizing.

eQTLResult The result dataframe obtained the sc-eQTL results.

biClassify The user chooses whether to convert the counting method of the snpMatrix to 0, 1, 2, TRUE indicates conversion, and FALSE indicates no conversion, default is no conversion.

species The species that the user wants to select, human or mouse.

groupBy Options for cell grouping, users can choose celltype, cellstatus, etc., depending on meta-data.

useModel model for fitting dataframe.

---

filterGeneSNP	<i>filterGeneSNP: Filter gene expression matrix and genotype matrix.</i>
---------------	--

---

### Description

filterGeneSNP: Filter gene expression matrix and genotype matrix.

### Usage

```
filterGeneSNP(
  eQTLObjct,
  snpNumOfCellsPercent = 10,
  expressionMin = 0,
  expressionNumOfCellsPercent = 10
)
```

**Arguments**

`eQTLObject` An S4 object of class `eQTLObject`.

`snpNumOfCellsPercent` Only SNPs where cells with each of the different genotypes (REF and ALT, or AA, Aa, and aa) individually account for at least ‘`snpNumOfCellsPercent`’ Default by 10.

`expressionMin` threshold for valid gene expression levels, utilized alongside another parameter, `expression.number.of.cells`. Default by 0.

`expressionNumOfCellsPercent` Only genes with expression levels exceeding ‘`expressionMin`’ in at least ‘`expressionNumOfCellsPercent`’ of cells are considered. The default value is 10.

**Value**

filtered matrices.

**Examples**

```
data(testSNP)
data(testGene)
eqtl <- createQTLObject(snpMatrix = testSNP, genedata = testGene)
eqtl <- normalizeGene(eqtl)
eqtl <- filterGeneSNP(eqtl,
  snpNumOfCellsPercent = 2,
  expressionMin = 0,
  expressionNumOfCellsPercent = 2)
```

---

<code>filter_by_abs_b</code>	<i>Filters data frame by absolute b-values, returning rows meeting or exceeding a threshold.</i>
------------------------------	--

---

**Description**

Filters data frame by absolute b-values, returning rows meeting or exceeding a threshold.

**Usage**

```
filter_by_abs_b(result, logfcThreshold)
```

**Arguments**

`result` Dataframe that contains gene-SNP pairs’ information.

`logfcThreshold` Represents the minimum beta threshold for fitting SNP-Gene pairs. Default by 0.1.

**Value**

A dataframe filtered by absolute b-values.



## Examples

```
example_result <- data.frame(  
  gene = c("Gene1", "Gene2", "Gene3", "Gene4"),  
  SNP = c("SNP1", "SNP2", "SNP3", "SNP4"),  
  b = c(-2.5, 1.0, -0.5, 3.0))  
logfcThreshold <- 0.1  
filtered_result <- filter_by_abs_b(example_result, logfcThreshold)
```

---

get\_cell\_groups      *Retrieve Cells by SNP Value*

---

## Description

This function extracts the names of cells from a SNP matrix that correspond to a specified value for a given SNP.

## Usage

```
get_cell_groups(snpMatrix, SNPid, biClassify)
```

## Arguments

snpMatrix	A matrix containing SNP data where rows represent SNPs and columns represent cells.
SNPid	A character string or numeric index representing the specific SNP of interest in the SNP matrix.
biClassify	The user chooses whether to convert the counting method of the snpMatrix to 0/1/2, TRUE indicates conversion, and FALSE indicates no conversion, default is no conversion.

## Value

A list of cell names (column names of the SNP matrix) that correspond to the specified genotype value for the given SNP.

## Examples

```
data(testSNP)  
biClassify <- FALSE  
get_cell_groups(testSNP, "1:632445", biClassify)
```

---

get_counts	<i>Extract Counts from an Expression Matrix</i>
------------	---

---

**Description**

This function retrieves expression counts for a specified gene from an expression matrix, based on the provided list of cells.

**Usage**

```
get_counts(expressionMatrix, Geneid, cells)
```

**Arguments**

expressionMatrix	A matrix containing gene expression data where rows represent genes and columns represent cells.
Geneid	A character string or numeric index representing the specific gene of interest in the expression matrix.
cells	A character vector of cell names (column names of the expression matrix) from which to extract counts for the specified gene.

**Value**

A numeric vector of expression counts for the specified gene in the selected cells.

**Examples**

```
data(testGene)
get_counts(testGene, "CNN2",
           c("CGGCAGTGTAGCCCTG", "GGAGGATCCCGTTCA"))
```

---

get_filter_data	<i>Generic to access eQTLObjct filter data</i>
-----------------	--

---

**Description**

Generic to access eQTLObjct filter data

**Usage**

```
get_filter_data(x)
```

**Arguments**

x	A eQTLObjct object.
---	---------------------

**Value**

filtered matrices.

**Examples**

```
data(testEQTL)
get_filter_data(testEQTL)
```

---

*get\_filter\_data,eQTLObjct-method*  
*Method to access eQTLObjct filter data*

---

**Description**

Method to access eQTLObjct filter data

**Usage**

```
## S4 method for signature 'eQTLObjct'
get_filter_data(x)
```

**Arguments**

x                    A eQTLObjct object.

**Value**

filtered matrices.

---

*get\_model\_info*                    *Generic to access eQTLObjct used model information*

---

**Description**

Generic to access eQTLObjct used model information

**Usage**

```
get_model_info(x)
```

**Arguments**

x                    A eQTLObjct object.

**Value**

used model information of eQTLObject.

**Examples**

```
data(testEQTL)
get_model_info(testEQTL)
```

---

get\_model\_info, eQTLObject-method

*Method to access eQTLObject used model information*

---

**Description**

Method to access eQTLObject used model information

**Usage**

```
## S4 method for signature 'eQTLObject'
get_model_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

used model information of eQTLObject.

---

get\_raw\_data

*Generic to access eQTLObject raw data*

---

**Description**

Generic to access eQTLObject raw data

**Usage**

```
get_raw_data(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

raw data matrix.

**Examples**

```
data(testEQTL)
get_raw_data(testEQTL)
```

---

*get\_raw\_data,eQTLObject-method*  
*Method to access eQTLObject raw data*

---

**Description**

Method to access eQTLObject raw data

**Usage**

```
## S4 method for signature 'eQTLObject'
get_raw_data(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

raw data matrix.

---

*get\_result\_info*            *Generic to access the result of identifying eQTLs from scRNA-seq data*

---

**Description**

Generic to access the result of identifying eQTLs from scRNA-seq data

**Usage**

```
get_result_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

A dataframe.

**Examples**

```
data(testEQTL)
get_result_info(testEQTL)
```

---

get\_result\_info, eQTLObject-method

*Method to access the result of identifying eQTLs from scRNA-seq data*

---

**Description**

Method to access the result of identifying eQTLs from scRNA-seq data

**Usage**

```
## S4 method for signature 'eQTLObject'
get_result_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

A dataframe.

---

initialize\_progress\_bar

*Progress Bar for Model Analysis.*

---

**Description**

This function initializes a progress bar for use in the 'linearModel', 'poissonModel' and 'zinbModel' function. It is designed to provide feedback on the progress of the analysis by displaying the current step and a percentage completion.

**Usage**

```
initialize_progress_bar(total, k)
```

**Arguments**

total	The total number of steps or iterations for which the progress bar will be updated.
k	A label or identifier for the specific group or iteration for which the progress bar is being initialized.

**Value**

A 'progress\_bar' object from the 'progress' package, which is used to track and display the progress.

**Examples**

```
unique_group <- c("CMP", "GMP")
total_snp_count <- 10 # assume each group have 100 SNP.
pb_model <- lapply(unique_group, function(k) {
  pb <- initialize_progress_bar(total = total_snp_count, k)
  for (i in seq_len(total_snp_count)) {
    Sys.sleep(0.1) # assume progress time
    pb$tick() # update pb
  }
})
```

---

limma\_normalize

*Normalize the gene expression matrix with limma*

---

**Description**

'limma\_normalize()' normalizes an expression matrix using the quantile normalization method provided by the limma package.

**Usage**

```
limma_normalize(expressionMatrix)
```

**Arguments**

expressionMatrix  
Input raw gene expression matrix.

**Value**

A gene expression matrix after normalized.

**Examples**

```
data(testGene)
limma_normalize(testGene)
```

---

 linearModel

*Linear model fitting the gene expression matrix and genotype matrix.*


---

### Description

Linear model fitting the gene expression matrix and genotype matrix.

### Usage

```
linearModel(
  eQTLObject,
  geneIDs,
  snpIDs,
  biClassify = FALSE,
  pAdjustMethod = "bonferroni",
  pAdjustThreshold = 0.05,
  logfcThreshold = 0.1
)
```

### Arguments

eQTLObject	An S4 object of class eQTLObject.
geneIDs	Matching genes can be used to fit data.
snpIDs	Matching SNPs can be used to fit data.
biClassify	The user chooses whether to convert the counting method of the snpMatrix to 0/1/2, TRUE indicates conversion, and FALSE indicates no conversion, default is no conversion.
pAdjustMethod	Methods for p-value adjusting, one of "bonferroni", "holm", "hochberg", "hommel" or "BH". The default option is "bonferroni".
pAdjustThreshold	Only SNP-Gene pairs with adjusted p-values meeting the threshold will be displayed. Default by 0.05.
logfcThreshold	Represents the minimum beta threshold for fitting SNP-Gene pairs. Default by 0.1.

### Value

Dataframe that contains gene-SNP pairs' information.

### Examples

```
data(testEQTL)
Gene <- rownames(slot(testEQTL, "filterData")$expMat)
SNP <- rownames(slot(testEQTL, "filterData")$snpMat)
linearResult <- linearModel(
  eQTLObject = testEQTL,
```



```
geneIDs = Gene,  
snpIDs = SNP,  
biClassify = FALSE,  
pAdjustMethod = "bonferroni",  
pAdjustThreshold = 0.05,  
logfcThreshold = 0.025)
```

---

load\_biclassify\_info *Generic to access eQTLObject biclassify information*

---

**Description**

Generic to access eQTLObject biclassify information

**Usage**

```
load_biclassify_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

biclassify information of eQTLObject.

**Examples**

```
data(testEQTL)  
load_biclassify_info(testEQTL)
```

---

load\_biclassify\_info, eQTLObject-method  
*Method to access eQTLObject biclassify information*

---

**Description**

Method to access eQTLObject biclassify information

**Usage**

```
## S4 method for signature 'eQTLObject'  
load_biclassify_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

biclassify information of eQTLOBJECT.

---

load_group_info	<i>Generic to access eQTLOBJECT cell grouping information</i>
-----------------	---

---

**Description**

Generic to access eQTLOBJECT cell grouping information

**Usage**

```
load_group_info(x)
```

**Arguments**

x                    A eQTLOBJECT object.

**Value**

A dataframe.

**Examples**

```
data(testEQTL)
load_group_info(testEQTL)
```

---

load_group_info,eQTLOBJECT-method	<i>Method to access eQTLOBJECT cell grouping information</i>
-----------------------------------	--

---

**Description**

Method to access eQTLOBJECT cell grouping information

**Usage**

```
## S4 method for signature 'eQTLOBJECT'
load_group_info(x)
```

**Arguments**

x                    A eQTLOBJECT object.

**Value**

A dataframe.

---

load_species_info	<i>Generic to access eQTLObject species information</i>
-------------------	---

---

**Description**

Generic to access eQTLObject species information

**Usage**

```
load_species_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

species information of eQTLObject.

**Examples**

```
data(testEQTL)
load_species_info(testEQTL)
```

---

load_species_info, eQTLObject-method	<i>Method to access eQTLObject species information</i>
--------------------------------------	--

---

**Description**

Method to access eQTLObject species information

**Usage**

```
## S4 method for signature 'eQTLObject'
load_species_info(x)
```

**Arguments**

x                    A eQTLObject object.

**Value**

species information of eQTLObject.

---

log_normalize	<i>Normalize the gene expression matrix with logNormalize method.</i>
---------------	---

---

**Description**

'log\_normalize()' transforms an expression matrix by applying logarithm and scaling operations to normalize data.

**Usage**

```
log_normalize(expressionMatrix)
```

**Arguments**

expressionMatrix  
Input raw gene expression matrix.

**Value**

A gene expression matrix after normalized.

**Examples**

```
data(testGene)
log_normalize(testGene)
```

---

normalizeGene	<i>normalizeGene: Normalize the gene expression data.</i>
---------------	---

---

**Description**

Gene expression matrix normalization is necessary to eliminate technical biases and variabilities, ensuring accurate and comparable analysis of gene expression data. Here we provide 'normalizeGene()' to normalize the data.

**Usage**

```
normalizeGene(eQTLObject, method = "logNormalize")
```

**Arguments**

eQTLObject      An S4 object of class eQTLObject.  
method            Method for normalizing for gene expression dataframe, one of "logNormalize", "CPM", "TPM", "DESeq" or "limma"

**Value**

A normalized gene expression matrix.

**Examples**

```
data(testEQTL)
eqtl <- normalizeGene(testEQTL, method = "logNormalize")
```

---

plots_theme_opts	<i>Theme options for customized plots</i>
------------------	---

---

**Description**

Theme options for customized plots

**Usage**

```
plots_theme_opts()
```

**Value**

A ggplot2 theme object with customized settings.

**Examples**

```
library(ggplot2)
data <- data.frame(
  x = c("A", "B", "C", "D", "E"),
  y = c(10, 20, 30, 40, 50))
ggplot(data, aes(x, y)) +
  geom_point() +
  plots_theme_opts()
```

---

poissonModel	<i>Poisson model fitting the gene expression matrix and genotype matrix.</i>
--------------	--

---

**Description**

Poisson model fitting the gene expression matrix and genotype matrix.

**Usage**

```
poissonModel(
  eQTLObject,
  geneIDs,
  snpIDs,
  biClassify = FALSE,
  pAdjustMethod = "bonferroni",
  pAdjustThreshold = 0.05,
  logfcThreshold = 0.1
)
```

**Arguments**

eQTLObject	An S4 object of class eQTLObject.
geneIDs	Matching genes can be used to fit data.
snpIDs	Matching SNPs can be used to fit data.
biClassify	The user chooses whether to convert the counting method of the snpMatrix to 0/1/2, TRUE indicates conversion, and FALSE indicates no conversion, default is FALSE.
pAdjustMethod	Methods for p-value adjusting, one of "bonferroni", "holm", "hochberg", "hommel" or "BH". The default option is "bonferroni".
pAdjustThreshold	Only SNP-Gene pairs with adjusted p-values meeting the threshold will be displayed. The default value is 0.05.
logfcThreshold	Represents the minimum beta threshold for fitting SNP-Gene pairs.

**Value**

Dataframe that contains gene-SNP pairs' information.

**Examples**

```
data(testEQTL)
Gene <- rownames(slot(testEQTL, "filterData")$expMat)
SNP <- rownames(slot(testEQTL, "filterData")$snpMat)
poissonResult <- poissonModel(
  eQTLObject = testEQTL,
  geneIDs = Gene,
  snpIDs = SNP,
  biClassify = FALSE,
  pAdjustMethod = "bonferroni",
  pAdjustThreshold = 0.05,
  logfcThreshold = 0.025
)
```

---

process_matrix	<i>Process a matrix to extract a row and convert it to a data frame</i>
----------------	---

---

**Description**

Process a matrix to extract a row and convert it to a data frame

**Usage**

```
process_matrix(id, matrix, name)
```

**Arguments**

id	The identifier for the row to be extracted from the matrix.
matrix	The input matrix from which the row will be extracted.
name	The column names for the resulting data frame.

**Value**

A data frame containing the extracted row and a column with the row names.

**Examples**

```
rownames <- c("CNN2", "TIGD2", "DTD2")
colnames <- c("Col1", "Col2", "Col3", "Col4")
matrix_data <- matrix(1:12, nrow = 3, ncol = 4,
  dimnames = list(rownames, colnames))
geneid <- "CNN2"
gene_mat <- process_matrix(geneid, matrix_data, "gene_mat")
```

---

remove_outliers	<i>Remove outliers from gene expression data and update cell lists</i>
-----------------	--

---

**Description**

remove\_outliers() is a function designed to process gene expression data stored in an expression matrix. It identifies outliers within the data based on the MAD method and filters them out. The function updates specified cell lists by retaining only those cells that have non-outlier expression values for a specified gene.

**Usage**

```
remove_outliers(exprsMat, Geneid, A_cells, B_cells, C_cells = NULL)
```

**Arguments**

exprsMat	Input gene expression matrix
Geneid	Chosen gene id.
A_cells	A genotype cells
B_cells	B genotype cells
C_cells	C genotype cells

**Value**

a list of cells ids

**Examples**

```
## Mock expression matrix
set.seed(123)
exprsMat <- matrix(rnorm(200), nrow = 5)
rownames(exprsMat) <- paste0("Gene", 1:nrow(exprsMat))
colnames(exprsMat) <- paste0("cell", 1:ncol(exprsMat))
A_cells <- colnames(exprsMat)[1:13] # Example A cell list
B_cells <- colnames(exprsMat)[14:26] # Example B cell list
C_cells <- colnames(exprsMat)[27:40] # Example C cell list
remove_outliers(exprsMat, "Gene1", A_cells, B_cells, C_cells)
```

---

set_filter_data	<i>Generic to set eQTLObject filter data</i>
-----------------	--

---

**Description**

Generic to set eQTLObject filter data

**Usage**

```
set_filter_data(x, value, name)
```

**Arguments**

x	A eQTLObject object.
value	The filtered data.
name	The matrix named 'name' is stored under the 'filterData' slot as an element within its list.

**Value**

eQTLObject.



**Examples**

```
data(testEQTL)
data123 <- matrix(0, nrow = 3, ncol = 3)
set_filter_data(testEQTL, data123, "expMat")
```

---

set\_filter\_data,eQTLOBJECT-method

*Method to set eQTLOBJECT filter data*

---

**Description**

Method to set eQTLOBJECT filter data

**Usage**

```
## S4 method for signature 'eQTLOBJECT'
set_filter_data(x, value, name)
```

**Arguments**

x	A eQTLOBJECT object.
value	The filtered data.
name	The matrix named 'name' is stored under the 'filterData' slot as an element within its list.

**Value**

eQTLOBJECT.

**Examples**

```
data(testEQTL)
data123 <- matrix(0, nrow = 3, ncol = 3)
set_filter_data(testEQTL, data123, "expMat")
```

---

set\_model\_info                      *Generic to set eQTLObject used model information*

---

**Description**

Generic to set eQTLObject used model information

**Usage**

```
set_model_info(x, value)
```

**Arguments**

x                                      A eQTLObject object.  
value                                  The used model information to set to eQTLObject.

**Value**

eQTLObject.

**Examples**

```
data(testEQTL)
useModel <- "zinb"
set_model_info(testEQTL, useModel)
```

---

set\_model\_info,eQTLObject-method  
   *Method to set eQTLObject used model information*

---

**Description**

Method to set eQTLObject used model information

**Usage**

```
## S4 method for signature 'eQTLObject'
set_model_info(x, value)
```

**Arguments**

x                                      A eQTLObject object.  
value                                  The used model information to set to eQTLObject.

**Value**

eQTLObject.

**Examples**

```
data(testEQTL)
useModel <- "zinb"
set_model_info(testEQTL, useModel)
```

---

set_raw_data	<i>Generic to set eQTLObject raw data</i>
--------------	---

---

**Description**

Generic to set eQTLObject raw data

**Usage**

```
set_raw_data(x, value, name)
```

**Arguments**

x	A eQTLObject object.
value	The raw data.
name	The matrix named 'name' is stored under the 'rawData' slot as an element within its list.

**Value**

eQTLObject.

**Examples**

```
data(testEQTL)
data123 <- matrix(0, nrow = 3, ncol = 3)
set_raw_data(testEQTL, data123, "rawExpMat")
```

---

```
set_raw_data, eQTLObject-method
```

*Method to set eQTLObject raw data*

---

**Description**

Method to set eQTLObject raw data

**Usage**

```
## S4 method for signature 'eQTLObject'
set_raw_data(x, value, name)
```

**Arguments**

x	A eQTLObject object.
value	The raw data.
name	The matrix named 'name' is stored under the 'rawData' slot as an element within its list.

**Value**

eQTLObject.

**Examples**

```
data(testEQTL)
data123 <- matrix(0, nrow = 3, ncol = 3)
set_raw_data(testEQTL, data123, "rawExpMat")
```

---

```
set_result_info
```

*Generic to set the result of identifying eQTLs from scRNA-seq data*

---

**Description**

Generic to set the result of identifying eQTLs from scRNA-seq data

**Usage**

```
set_result_info(x, value)
```

**Arguments**

x	A eQTLObject object.
value	A dataframe, each row describes eQTL discovering result of a SNP-Gene pair.

**Value**

eQTLObject.

**Examples**

```
data(testEQTl)
result <- matrix(0, nrow = 3, ncol = 3)
set_result_info(testEQTl, result)
```

---

set\_result\_info,eQTLObject-method

*Method to set the result of identifying eQTLs from scRNA-seq data*

---

**Description**

Method to set the result of identifying eQTLs from scRNA-seq data

**Usage**

```
## S4 method for signature 'eQTLObject'
set_result_info(x, value)
```

**Arguments**

x	A eQTLObject object.
value	A dataframe, each row describes eQTL discovering result of a SNP-Gene pair.

**Value**

eQTLObject.

**Examples**

```
data(testEQTl)
result <- matrix(0, nrow = 3, ncol = 3)
set_result_info(testEQTl, result)
```

---

show, eQTLObjct-method

*Show Method for eQTLObjct Class*

---

### Description

This method is to display information about an object of class eQTLObjct. When called on an eQTLObjct, it prints a descriptive message to the console

### Usage

```
## S4 method for signature 'eQTLObjct'  
show(object)
```

### Arguments

object            An S4 object of class eQTLObjct.

### Value

information of eQTLObjct

### Examples

```
data(testEQTL)  
testEQTL
```

---

testEQTL

*Test eqtl object*

---

### Description

An 'eqtlObject' created by the 'createQTLObjct' function, where the raw expression matrix is normalized using 'normalizeGene()', and both the genotype matrix and the normalized gene expression matrix are filtered by 'filterGeneSNP()'.  
testEQTL.rds is the RDS format versions of the original testEQTL.rda, providing the same normalized eQTL object for easier loading and use in R.

### Usage

```
data(testEQTL)  
  
data(testEQTL)
```

**Format**

A simple object.

A eqtlObject read by the 'readRDS' function.

**Examples**

```
data(testEQTL)
data(testEQTL)
```

---

testGene	<i>Test Gene Expression Dataset</i>
----------	-------------------------------------

---

**Description**

A dataset containing example gene expression data for testing purposes. 100 rows and 2705 columns. The row names represent gene IDs or SYMBOL and the column names represent cell IDs.

**Usage**

```
data(testGene)
```

**Format**

A simple matrix.

**Examples**

```
data(testGene)
```

---

testSeurat	<i>Test SeuratObject</i>
------------	--------------------------

---

**Description**

A Seurat object for single-cell RNA-seq data.

testSeurat.rds datasets are the RDS format versions of the original testSeurat.rda files, providing the preprocessed Seurat object for easier loading and use in R.

**Usage**

```
data(testSeurat)
```

```
data(testSeurat)
```

**Format**

A object

A Seurat read by the 'readRDS' function.

**Examples**

```
data(testSeurat)
data(testSeurat)
```

---

testSNP	<i>Test Genotype Dataset</i>
---------	------------------------------

---

**Description**

A dataset containing single nucleotide variant data. 1000 rows and 2705 columns. Each row is one variant and each column is one cell.

**Usage**

```
data(testSNP)
```

**Format**

A simple matrix.

**Examples**

```
data(testSNP)
```

---

testSNP2	<i>Test Genotype Dataset</i>
----------	------------------------------

---

**Description**

A dataset containing single nucleotide variant data. 500 rows and 500 columns. Each row is one variant and each column is one cell.

**Usage**

```
data(testSNP2)
```

**Format**

A simple matrix.

**Examples**

```
data(testSNP2)
```



---

TPM_normalize	<i>Normalize the gene expression matrix with TPM</i>
---------------	--

---

**Description**

'TPM\_normalize()' scales an expression matrix using Transcripts Per Million (TPM) normalization, applying logarithm and scaling operations to adjust data based on library size.

**Usage**

```
TPM_normalize(expressionMatrix)
```

**Arguments**

```
expressionMatrix  
    Input raw gene expression matrix.
```

**Value**

A gene expression matrix after normalized.

**Examples**

```
data(testGene)  
TPM_normalize(testGene)
```

---

visualizeQTL	<i>visualizeQTL: Visualize the gene-snp pairs by group.</i>
--------------	---

---

**Description**

visualizeQTL: Visualize the gene-snp pairs by group.

**Usage**

```
visualizeQTL(  
  eQTLObject,  
  SNPid,  
  Geneid,  
  groupName = NULL,  
  plotype = "QTLplot",  
  removeoutlier = FALSE  
)
```

**Arguments**

eQTLObject	An S4 object of class eQTLObject.
SNPid	ID of SNP.
Geneid	ID of Gene.
groupName	Users can choose one or more than one single cell groups.
plottype	Types of plot, one of "QTLplot", "violin", "boxplot" or "histplot".
removeoutlier	Whether identify and remove the outliers. Default by FALSE.

**Value**

list

**Examples**

```
data(testEQL)
## We have to call the eQTLs firstly using `callQTL()`.
eqtl <- callQTL(eQTLObject = testEQL, useModel = "linear")
visualizeQTL(eQTLObject = eqtl,
SNPid = "1:632647",
Geneid = "RPS27",
groupName = NULL,
plottype = "QTLplot",
removeoutlier = FALSE)
```

---

zinbModel

*Zinb model fitting the gene expression matrix.*


---

**Description**

Zinb model fitting the gene expression matrix.

**Usage**

```
zinbModel(
  eQTLObject,
  geneIDs,
  snpIDs,
  biClassify = FALSE,
  pAdjustMethod = "bonferroni",
  pAdjustThreshold = 0.05
)
```

**Arguments**

eQTLObject	An S4 object of class eQTLObject.
geneIDs	Matching genes can be used to fit data.
snpIDs	Matching SNPs can be used to fit data.
biClassify	The user chooses whether to convert the counting method of the snpMatrix to 0/1/2, TRUE indicates conversion, and FALSE indicates no conversion, default is no conversion.
pAdjustMethod	Methods for p-value adjusting, one of 'bonferroni', 'holm', 'hochberg', 'hommel' or 'BH'. The default option is 'bonferroni'.
pAdjustThreshold	Only SNP-Gene pairs with adjusted p-values meeting the threshold will be displayed. Default by 0.05.

**Value**

Dataframe that contains gene-SNP pairs' information.

**Examples**

```
data(testEQTL)
Gene <- rownames(slot(testEQTL, 'filterData')$expMat)
SNP <- rownames(slot(testEQTL, 'filterData')$snpMat)
zinbResult <- zinbModel(
  eQTLObject = testEQTL,
  geneIDs = Gene,
  snpIDs = SNP,
  biClassify = FALSE,
  pAdjustMethod = 'bonferroni',
  pAdjustThreshold = 0.05)
```

# Index

- \* **datasets**
  - testEQTL, [38](#)
  - testGene, [39](#)
  - testSeurat, [39](#)
  - testSNP, [40](#)
  - testSNP2, [40](#)
- \* **internal**
  - scQTLtools-package, [3](#)
- adjust\_pvalues, [4](#)
- buildZINB, [5](#)
- callQTL, [5](#)
- checkSNPList, [7](#)
- CPM\_normalize, [8](#)
- createGeneLoc, [8](#)
- createQTLObject, [9](#)
- createSNPsLoc, [10](#)
- DESeq\_normalize, [11](#)
- draw\_boxplot, [12](#)
- draw\_histplot, [12](#)
- draw\_QTLplot, [13](#)
- draw\_violinplot, [14](#)
- eQTLObject-class, [15](#)
- filter\_by\_abs\_b, [16](#)
- filterGeneSNP, [15](#)
- get\_cell\_groups, [17](#)
- get\_counts, [18](#)
- get\_filter\_data, [18](#)
- get\_filter\_data, eQTLObject-method, [19](#)
- get\_model\_info, [19](#)
- get\_model\_info, eQTLObject-method, [20](#)
- get\_raw\_data, [20](#)
- get\_raw\_data, eQTLObject-method, [21](#)
- get\_result\_info, [21](#)
- get\_result\_info, eQTLObject-method, [22](#)
- initialize\_progress\_bar, [22](#)
- limma\_normalize, [23](#)
- linearModel, [24](#)
- load\_biclassify\_info, [25](#)
- load\_biclassify\_info, eQTLObject-method, [25](#)
- load\_group\_info, [26](#)
- load\_group\_info, eQTLObject-method, [26](#)
- load\_species\_info, [27](#)
- load\_species\_info, eQTLObject-method, [27](#)
- log\_normalize, [28](#)
- normalizeGene, [28](#)
- plots\_theme\_opts, [29](#)
- poissonModel, [29](#)
- process\_matrix, [31](#)
- remove\_outliers, [31](#)
- scQTLtools (scQTLtools-package), [3](#)
- scQTLtools-package, [3](#)
- set\_filter\_data, [32](#)
- set\_filter\_data, eQTLObject-method, [33](#)
- set\_model\_info, [34](#)
- set\_model\_info, eQTLObject-method, [34](#)
- set\_raw\_data, [35](#)
- set\_raw\_data, eQTLObject-method, [36](#)
- set\_result\_info, [36](#)
- set\_result\_info, eQTLObject-method, [37](#)
- show, eQTLObject-method, [38](#)
- testEQTL, [38](#)
- testGene, [39](#)
- testSeurat, [39](#)
- testSNP, [40](#)
- testSNP2, [40](#)
- TPM\_normalize, [41](#)

visualizeQTL, [41](#)

zinbModel, [42](#)