# Package ‘scQTLtools’

February 28, 2026

**Type** Package

**Title** scQTLtools: an R/Bioconductor package for comprehensive identification and visualization of single-cell eQTLs

**Version** 1.2.4

**Description** scQTLtools is a comprehensive R/Bioconductor package that facilitates end-to-end single-cell eQTL analysis, from preprocessing to visualization

**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), yulab.utils (>= 0.2.3)

**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, Preprocessing

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**LazyData** false

**Config/testthat/edition** 3

**git_url** https://git.bioconductor.org/packages/scQTLtools

**git_branch** RELEASE_3_22

**git_last_commit** 16fb670

**git_last_commit_date** 2026-02-02

**Repository** Bioconductor 3.22

**Date/Publication** 2026-02-27

**Author** Xiaofeng Wu [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0003-6254-5575>),
Xin Huang [aut, cph] (ORCID: <https://orcid.org/0009-0005-2755-0357>),
Jingtong Kang [com] (ORCID: <https://orcid.org/0009-0008-8343-3456>),
Siwen Xu [aut, cph] (ORCID: <https://orcid.org/0000-0001-7936-0639>)

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

# Contents

---

`scQTLtools-package` *scQTLtools: scQTLtools: an R/Bioconductor package for comprehensive identification and visualization of single-cell eQTLs*

---

## Description

scQTLtools is a comprehensive R/Bioconductor package that facilitates end-to-end single-cell eQTL analysis, from preprocessing to visualization

## Author(s)

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

Authors:

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

Other contributors:

- Jingtong Kang `<1203178107@qq.com>` (ORCID) [compiler]

## See Also

Useful links:

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

---

`adjust_pvalues` *Adjust p-values and filter SNP–gene pairs based on the adjusted p-values.*

---

## Description

Adjust p-values and filter SNP–gene pairs based on the adjusted p-values.

## Usage

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

### Arguments

| | |
|---|---|
| `result` | A data frame that contains information of SNP–gene pairs and raw p values. |
| `pAdjustMethod` | Method for p-value adjustment. 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

A data frame with adjusted p-values, filtered by threshold, containing information on SNP–gene 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` *Build a ZINB model.*

---

### Description

Build a ZINB model.

### Usage

```
buildZINB(counts)
```

### Arguments

| | |
|---|---|
| `counts` | A numeric vector of gene expression values. |

### Value

A list containing four parameters estimated from the ZINB model.

### Examples

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

## `callQTL` *Identify single-cell eQTLs.*

### Description

This function detects eQTLs using scRNA-seq data and genotype 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` | Distance (in base pairs) downstream of the gene start site to search for associated SNPs. |
| `upstream` | Distance (in base pairs) upstream of the gene end site to search for associated SNPs. |
| `gene_mart` | A Mart object representing the BioMart gene database. If `NULL`, the Ensembl Gene BioMart will be used. |
| `snp_mart` | A Mart object representing the BioMart SNP database. If `NULL`, the Ensembl SNP BioMart will be used. |
| `pAdjustMethod` | Method used for multiple testing correction. One of `"bonferroni"`, `"holm"`, `"hochberg"`, `"hommel"`, or `"BH"`. Default is `"bonferroni"`. |
| `useModel` | Model used for fitting. One of "poisson", "zinb", or "linear." |
| `pAdjustThreshold` | Only SNP–gene pairs with adjusted p-values below the threshold will be retained. Default is 0.05. |
| `logfcThreshold` | The minimum beta coefficient required to report a SNP–gene pair as an eQTL. |

### Value

A data frame in which each row corresponds to a detected SNP–gene eQTL pair, including statistical and model fitting results.

## Examples

```
data(EQTL_obj)
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 = EQTL_obj,
  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 *Validate SNP IDs in the input genotype matrix.*

---

### Description

Validate SNP IDs in the input genotype matrix.

### Usage

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

### Arguments

| | |
|---|---|
| snpList | A list of SNPs IDs. |
| snp_mart | An object of class Mart representing the BioMart SNP database to connect to. 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. Default is `hsapiens_snp` for human SNPs. |

### Value

A data frame containing the genomic locations of the valid SNPs.

### Examples

```
data(SNPData2)
snpList <- rownames(SNPData2)
snpDataset <- 'hsapiens_snp'
snps_loc <- checkSNPList(snpList = snpList,
                        snpDataset = snpDataset)
```

---

## `CPM_normalize` *Normalize gene expression using CPM.*

---

### Description

`CPM_normalize()` applies Counts Per Million (CPM) normalization to a raw gene expression matrix.

### Usage

```
CPM_normalize(expressionMatrix)
```

### Arguments

| | |
|---|---|
| `expressionMatrix` | A numeric matrix of raw gene expression counts, with genes as rows and cells as columns. |

### Value

A normalized gene expression matrix after applying CPM normalization.

### Examples

```
data(GeneData)
CPM_normalize(GeneData)
```

---

## `createGeneLoc` *Create a gene location data frame.*

---

### Description

Create a gene location data frame.

### Usage

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

### Arguments

| | |
|---|---|
| geneList | A gene ID or a list of genes IDs. |
| gene_mart | An object of class Mart representing the BioMart gene database to connect to. 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. Default is `hsapiens_gene_ensembl` for human genes. |
| OrgDb | The name of the OrgDb package to use for gene annotation. Supported values include `org.Hs.eg.db` and `org.Mm.eg.db`. |

### Value

A `data.frame` containing gene location information.

### Examples

```
data(GeneData)
geneList <- rownames(GeneData)
library(yulab.utils)
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 *Create an eQTLObject for storing sc-eQTL analysis data.*

---

### Description

This function creates an S4 object to store genotype and expression data, along with sample grouping and metadata for eQTL analysis.

### Usage

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

## Arguments

| | |
|---|---|
| `snpMatrix` | A genotype matrix where each row is a snp and each column is a cell. Encoding should be 0, 1, 2, 3. |
| `genedata` | A gene expression matrix, or a Seurat object or SingleCellExperiment object. |
| `biClassify` | Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default). |
| `species` | The species that the user wants to select, human or mouse. |
| `group` | A column name in the metadata of the Seurat or SingleCellExperiment object indicating cell groups (e.g., celltype or condition). |
| `...` | other parameters |

## Value

An S4 object of class `eQTLObject`.

## Examples

```
data(SNPData)
data(GeneData)
eqtl <- createQTLObject(snpMatrix = SNPData,
                        genedata = GeneData,
                        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 IDs. |
| `snp_mart` | An object of class Mart representing the BioMart SNP database to connect to. 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. Default is `hsapiens_snp` for human SNPs. |

## Value

A data frame containing the SNP genomic locations.

## 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)
```

---

`DataSet` *The package includes five examplt datasets: a gene expression matrix, two SNP genotype matrices, a Seurat object, and an eQTL object.*

---

## Description

GeneData: A dataset containing example gene expression data. Each row represents a gene and each column represents a cell.

SNPData: A dataset containing single nucleotide (SNP) data. Each row represents a variant and each column represents a cell.

Seurat_obj: A Seurat object containing `GeneData` along with associated metadata stored in the `meta.data` slot.

SNPData2: A smaller SNP dataset containing fewer variants and cells. Each row represents a variant and each column represents a cell.

EQTL_obj: An `eQTLObject` created using `createQTLObject`, where the raw expression matrix is normalized using `normalizeGene`, and both the genotype matrix and normalized expression matrix were filtered using `filterGeneSNP`.

## Format

A numeric matrix with 100 rows (genes) and 2705 columns (cells).

A numeric matrix with 1000 rows (SNPs) and 2705 columns (cells).

An object of class `Seurat`.

A numeric matrix with 500 rows (SNPs) and 500 columns (cells).

## See Also

`createQTLObject`, `normalizeGene` and `filterGeneSNP`, for the underlying functions that do the work.

An object of class `eQTLObject`.

## `DESeq_normalize` *Normalize the gene expression matrix with DESeq2.*

### Description

`DESeq_normalize()` normalizes a raw gene expression matrix using the **DESeq2** package.

### Usage

```
DESeq_normalize(expressionMatrix)
```

### Arguments

| | |
|---|---|
| `expressionMatrix` | A numeric matrix of raw gene expression counts, with genes as rows and cells as columns. |

### Value

A normalized gene expression matrix after applying DESeq normalization.

### Examples

```
data(GeneData)
DESeq_normalize(GeneData)
```

## `draw_boxplot` *Generate a boxplot of gene expression by SNP genotype.*

### Description

`draw_boxplot()` visualizes gene expression levels across different SNP genotypes using a boxplot.

### Usage

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

### Arguments

| | |
|---|---|
| `df` | A data frame containing gene expression values, SNP genotypes, and group labels. |
| `unique_group` | A character string indicating the unique group name. |

### 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` *Generate a histogram of gene expression by SNP genotype.*

---

### Description

`draw_histplot()` shows the distribution of expression values for each SNP genotype using histograms.

### Usage

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

### Arguments

| | |
|---|---|
| `df` | A data frame containing gene expression values, SNP genotypes, and group labels. |
| `unique_group` | A character string indicating the unique group name. |

### 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 violin-box-scatter plot.*

---

## Description

`draw_QTLplot()` creates a composite plot that overlays violin plots, boxplots, and scatter points to illustrate the distribution and variability of gene expression across SNP groups.

## Usage

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

## Arguments

| | |
|---|---|
| `df` | A data frame containing gene expression values, SNP genotypes, and group labels. |
| `unique_group` | A character string indicating the unique group name. |

## 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 gene expression by SNP genotype.*

---

## Description

`draw_violinplot()` visualizes gene expression levels across different SNP genotypes using violin plots.

## Usage

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

## Arguments

| | |
|---|---|
| `df` | A data frame containing gene expression values, SNP genotypes, and group labels. |
| `unique_group` | A character string indicating the unique group name. |

## 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)
```

---

`eQTLObject-class` *Class* `eQTLObject`

---

## Description

The eQTLObject class is designed to store data related to eQTL analysis, including data lists, result data frames, and additional metadata such as classification, species, and grouping information.

## Value

An S4 object of class `eQTLObject`.

## 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` Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default).

`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 metadata.

`useModel` model for fitting dataframe.

## filterGeneSNP *Filter gene expression and genotype matrices by cell percentage thresholds.*

### Description

Filter gene expression and genotype matrices by cell percentage thresholds.

### Usage

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

### Arguments

| | |
|---|---|
| `eQTLObject` | An S4 object of class `eQTLObject`. |
| `snpNumOfCellsPercent` | Numeric. Minimum percentage of cells required for each SNP genotype (e.g., AA, AG, and GG). Only SNPs where each genotype occurs in at least this proportion of cells are retained. Default is 10. |
| `expressionMin` | Numeric. Expression threshold used in combination with `expressionNumOfCellsPercent` to filter lowly expressed genes. Default is 0. |
| `expressionNumOfCellsPercent` | Numeric. Minimum percentage of cells in which a gene's expression must exceed `expressionMin` for the gene to be retained. Default is 10. |

### Value

An updated `eQTLObject` with filtered gene expression and SNP matrices.

### Examples

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

## `filter_by_abs_b` *Filter a data frame of SNP–gene pairs by absolute beta.*

### Description

Filter a data frame of SNP–gene pairs by absolute beta.

### Usage

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

### Arguments

| | |
|---|---|
| `result` | A data frame that containing SNP–gene pairs and their beta values (b-value). |
| `logfcThreshold` | A numeric value specifying the minimum absolute b-value to retain. Default is 0.1. |

### Value

A data frame containing only rows with absolute b-values above the specified threshold.

### 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 genotype matrix where each row is a snp and each column is a cell. Encoding should be 0, 1, 2, 3. |
| `SNPid` | A character string or numeric index representing the specific SNP of interest in the SNP matrix. |
| `biClassify` | Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default). |

## Value

A list of character vectors. Each vector contains the names of cells (i.e., column names of `snpMatrix`) corresponding to a specific genotype value at the given SNP.

## Examples

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

---

`get_counts` *Extract Counts from an Expression Matrix*

---

## 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 numeric matrix of gene expression counts, with genes as rows and cells as columns. |
| `Geneid` | A character string or numeric index representing the specific gene of interest in `expressionMatrix`. |
| `cells` | A character vector of cell names (column names of `expressionMatrix`) from which to extract counts for the specified gene. |

## Value

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

## Examples

```
data(GeneData)
get_counts(GeneData, "CNN2",
         c("CGGCAGTGTAGCCCTG", "GGAGGATTCCCGTTCA"))
```

## `get_filter_data` *Access filtered data stored in an eQTLObject.*

### Description

Access filtered data stored in an eQTLObject.

Method to access eQTLObject filter data.

### Usage

```
get_filter_data(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

Filtered matrices.

Filtered matrices.

### Examples

```
data(EQTL_obj)
get_filter_data(EQTL_obj)
```

## `get_model_info` *Access model specification from an eQTLObject*

### Description

Access model specification from an eQTLObject

Method to access eQTLObject used model information.

### Usage

```
get_model_info(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

A character string indicating the model.

used model information of eQTLObject.

### Examples

```
data(EQTL_obj)
get_model_info(EQTL_obj)
```

---

## get_raw_data *Access raw data stored in an eQTLObject.*

---

### Description

Access raw data stored in an eQTLObject.

Method to access eQTLObject raw data.

### Usage

```
get_raw_data(x)

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

### Arguments

| | |
|---|---|
| x | An `eQTLObject`. |

### Value

raw data matrix.

raw data matrix.

### Examples

```
data(EQTL_obj)
get_raw_data(EQTL_obj)
```

## `get_result_info` *Access eQTLs results from an eQTLObject.*

### Description

Access eQTLs results from an eQTLObject.

Method to access the result of identifying eQTLs.

### Usage

```
get_result_info(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

A data frame where each row corresponds to an identified SNP–gene pair.

A data frame with eQTL results.

### Examples

```
data(EQTL_obj)
get_result_info(EQTL_obj)
```

## `initialize_progress_bar` *Progress Bar for Model Analysis.*

### Description

Initializes a progress bar to track the computation progress during model fitting procedures such as `linearModel`, `poissonModel`, and `zinbModel`. This helps users monitor the status of long-running analyses.

### Usage

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

### Arguments

| | |
|---|---|
| `total` | Integer. The total number of iterations or steps to be completed in the analysis. |
| `k` | Character. An identifier or label for the specific group or subset being analyzed, used to annotate progress messages. |

## Value

An object of class progress_bar from the **progress** package, which can be updated using the $tick() method.

## 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 | A numeric matrix of raw gene expression counts, with genes as rows and cells as columns. |

## Value

A normalized gene expression matrix after applying limma normalization.

## Examples

```
data(GeneData)
limma_normalize(GeneData)
```

`linearModel` *Fit Linear Model for eQTL Mapping*

## Description

This function performs linear regression to identify gene–SNP associations based on single-cell expression and genotype data stored in an `eQTLObject`.

## Usage

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

## Arguments

| | |
|---|---|
| `eQTLObject` | An S4 object of class `eQTLObject`. |
| `geneIDs` | Character vector of gene IDs to include in the model fitting. |
| `snpIDs` | Character vector of SNP IDs to include in the model fitting. |
| `biClassify` | Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default). |
| `pAdjustMethod` | Method used for multiple testing correction. One of `"bonferroni"`, `"holm"`, `"hochberg"`, `"hommel"`, or `"BH"`. Default is `"bonferroni"`. |
| `pAdjustThreshold` | Only SNP–gene pairs with adjusted p-values below the threshold will be retained. Default is 0.05. |
| `logfcThreshold` | The minimum beta coefficient (effect size) required to report a SNP–gene pair as an eQTL. |

## Value

A data frame of gene–SNP pairs that pass the filtering criteria, including beta coefficients, p-values, adjusted p-values, and group labels.

## Examples

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

---

## `load_biclassify_info` *Access biclassification information from an eQTLObject.*

---

### Description

Access biclassification information from an eQTLObject.

Method to access eQTLObject biclassify information.

### Usage

```
load_biclassify_info(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

A character or list containing biclassification information.

biclassify information of eQTLObject.

### Examples

```
  data(EQTL_obj)
  load_biclassify_info(EQTL_obj)
```

---

## `load_group_info` *Access cell grouping information from an eQTLObject*

---

### Description

Access cell grouping information from an eQTLObject

Method to access eQTLObject cell grouping information.

### Usage

```
load_group_info(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

A data frame with grouping information.

A data frame with cell group assignments.

### Examples

```
data(EQTL_obj)
load_group_info(EQTL_obj)
```

---

## `load_species_info` *Access species information from an eQTLObject.*

---

### Description

Access species information from an eQTLObject.

Method to access eQTLObject species information.

### Usage

```
load_species_info(x)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |

### Value

A character string indicating the species.

species information of eQTLObject.

### Examples

```
data(EQTL_obj)
load_species_info(EQTL_obj)
```

## `log_normalize` *Normalize the gene expression matrix with logNormalize method.*

### Description

`log_normalize()` transforms an expression matrix by applying logarithm and scaling operations to normalize data.

### Usage

```
log_normalize(expressionMatrix)
```

### Arguments

| | |
|---|---|
| `expressionMatrix` | A numeric matrix of raw gene expression counts, with genes as rows and cells as columns. |

### Value

A normalized gene expression matrix after applying logNormalize normalization.

### Examples

```
data(GeneData)
log_normalize(GeneData)
```

## `normalizeGene` *Normalize gene expression data.*

### Description

This function performs normalization of the gene expression matrix stored within an `eQTLObject`, aiming to remove technical biases and improve comparability across cells. Multiple normalization methods are supported, including "logNormalize", "CPM", "TPM", "DESeq", and "limma".

### Usage

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

### Arguments

| | |
|---|---|
| `eQTLObject` | An S4 object of class `eQTLObject` containing the raw gene expression matrix. |
| `method` | Character string specifying the normalization method to use. Must be one of `"logNormalize"`, `"CPM"`, `"TPM"`, `"DESeq"`, or `"limma"`. Default is `"logNormalize"`. |

### Value

An `eQTLObject` with the normalized gene expression matrix stored in the slot named `"normExpMat"`.

### Examples

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

---

## `plots_theme_opts` *Customized ggplot2 Theme for Plots*

---

### Description

Customized ggplot2 Theme for Plots

### Usage

```
plots_theme_opts()
```

### Value

A `ggplot2` theme object for styling plots.

### 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` *Poisson model fitting the gene expression matrix and genotype matrix.*

---

### 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` | Character vector of gene IDs to include in the model fitting. |
| `snpIDs` | Character vector of SNP IDs to include in the model fitting. |
| `biClassify` | Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default). |
| `pAdjustMethod` | Method used for multiple testing correction. One of `"bonferroni"`, `"holm"`, `"hochberg"`, `"hommel"`, or `"BH"`. Default is `"bonferroni"`. |
| `pAdjustThreshold` | Only SNP–gene pairs with adjusted p-values below the threshold will be retained. Default is 0.05. |
| `logfcThreshold` | The minimum beta coefficient (effect size) required to report a SNP–gene pair as an eQTL. |

### Value

A data frame of gene–SNP pairs that pass the filtering criteria, including beta coefficients, p-values, adjusted p-values, and group labels.

### Examples

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

---

## `process_matrix` *Process a matrix to extract a row and convert it to a data frame.*

---

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| `id` | Character string specifying the row name to extract from the matrix. |
| `matrix` | A matrix from which the row will be extracted. |
| `name` | Character string specifying the column name for the extracted values. |

### 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 *Remove outliers from gene expression data and update cell groups.*

---

### Description

This function identifies and removes outlier expression values for a specified gene based on the Median Absolute Deviation (MAD) method. It then filters provided genotype-based cell groups, returning only cells with non-outlier expression values.

### Usage

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

### Arguments

| | |
|---|---|
| `exprsMat` | Input gene expression matrix with genes as rows and cells as columns. |
| `Geneid` | Character string specifying the gene ID to examine. |
| `A_cells` | Character vector of cell names belonging to genotype group A. |
| `B_cells` | Character vector of cell names belonging to genotype group B. |
| `C_cells` | Optional character vector of cell names belonging to genotype group C; if `NULL`, function returns two genotype groups. |

### Value

A named list of filtered cell vectors.

### 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` *Set filtered data in an eQTLObject.*

### Description

Set filtered data in an eQTLObject.

Method to set eQTLObject filter data.

### Usage

```
set_filter_data(x, value, name)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |
| `value` | A matrix to be stored as filtered data. |
| `name` | A character string indicating the key under which the matrix is stored in `filterData`. |

### Value

An updated `eQTLObject`.

An updated `eQTLObject`.

### Examples

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

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

## `set_model_info` *Set model specification in an eQTLObject*

### Description

Set model specification in an eQTLObject

Method to set eQTLObject used model information.

### Usage

```
set_model_info(x, value)

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

### Arguments

| | |
|---|---|
| x | An `eQTLObject`. |
| value | A character string specifying the model (e.g., "zinb", "poisson", "linear"). |

### Value

An updated `eQTLObject` with the new model specification.

An updated `eQTLObject`.

### Examples

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

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

---

## `set_raw_data` *Set raw data in an eQTLObject.*

---

### Description

Set raw data in an eQTLObject.

Method to set eQTLObject raw data.

### Usage

```
set_raw_data(x, value, name)

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

### Arguments

| | |
|---|---|
| x | An `eQTLObject`. |
| value | The raw data. |
| name | A character string indicating the key under which the matrix is stored in `rawData`. |

### Value

eQTLObject.

An updated `eQTLObject`.

### Examples

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

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

---

## `set_result_info` *Set eQTL results in an eQTLObject.*

---

### Description

Set eQTL results in an eQTLObject.

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

### Usage

```
set_result_info(x, value)

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

### Arguments

| | |
|---|---|
| `x` | An `eQTLObject`. |
| `value` | A data frame where each row corresponds to a SNP–gene pair. |

### Value

An updated `eQTLObject`.

An updated `eQTLObject`.

### Examples

```
data(EQTL_obj)
result <- data.frame(0, nrow = 3, ncol = 3)
set_result_info(EQTL_obj, result)

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

`TPM_normalize` *Normalize the gene expression matrix with TPM*

## 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` | A numeric matrix of raw gene expression counts, with genes as rows and cells as columns. |

## Value

A normalized gene expression matrix after applying TPM normalization.

## Examples

```
data(GeneData)
TPM_normalize(GeneData)
```

`visualizeQTL` *Visualize eQTL results by SNP–gene pair across groups*

## Description

Visualize eQTL results by SNP–gene pair across groups

## Usage

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

### Arguments

| | |
|---|---|
| `eQTLObject` | An S4 object of class `eQTLObject`. |
| `SNPid` | ID of SNP. |
| `Geneid` | ID of Gene. |
| `groupName` | A `character` vector specifying one or more cell group names to include in the plot. |
| `plottype` | A `character` string specifying the type of plot to generate. Must be one of `"QTLplot"`, `"violin"`, `"boxplot"`, or `"histplot"`. |
| `removeoutlier` | Logical; whether to identify and remove outliers. Default is `FALSE`. |

### Value

list

### Examples

```
data(EQTL_obj)
## We have to call the eQTLs firstly using \code{\link{callQTL()}}.
eqtl <- callQTL(eQTLObject = EQTL_obj, 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` | Character vector of gene IDs to include in the model fitting. |
| `snpIDs` | Character vector of SNP IDs to include in the model fitting. |

| | |
|---|---|
| `biClassify` | Logical; whether to convert genotype encoding in snpMatrix to 0, 1, and 2. `TRUE` indicates conversion; `FALSE` indicates no conversion (default). |
| `pAdjustMethod` | Method used for multiple testing correction. One of `"bonferroni"`, `"holm"`, `"hochberg"`, `"hommel"`, or `"BH"`. Default is `"bonferroni"`. |
| `pAdjustThreshold` | Only gene-SNP pairs with adjusted p-values below the threshold will be retained. Default is 0.05. |

## Value

A data frame of SNP–gene pairs that pass the filtering criteria, including p-values, adjusted p-values, and group labels.

## Examples

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

# Index