

# Package ‘QTLExperiment’

May 2, 2026

**Type** Package

**Date** 2025-10-27

**Title** S4 classes for QTL summary statistics and metadata

**Version** 2.5.0

**License** GPL-3

**URL** <https://github.com/dunstone-a/QTLExperiment>

**BugReports** <https://github.com/dunstone-a/QTLExperiment/issues>

**Encoding** UTF-8

**Depends** SummarizedExperiment

**Imports** methods, rlang, checkmate, dplyr, collapse, vroom, tidyr,  
tibble, utils, stats, ashR, S4Vectors, BiocGenerics

**Suggests** testthat, BiocStyle, knitr, rmarkdown, covr

**Description** QTLExperiment defines an S4 class for storing and manipulating summary statistics from QTL mapping experiments in one or more states.

It is based on the 'SummarizedExperiment' class and contains functions for creating, merging, and subsetting objects.

'QTLExperiment' also stores experiment metadata and has checks in place to ensure that transformations apply correctly.

**biocViews** FunctionalGenomics, DataImport, DataRepresentation,  
Infrastructure, Sequencing, SNP, Software

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3

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

**git\_branch** devel

**git\_last\_commit** ceefea5

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-01

**Author** Christina Del Azodi [aut],

Davis McCarthy [ctb],

Amelia Dunstone [cre, aut] (ORCID:

<https://orcid.org/0009-0009-6426-1529>)

**Maintainer** Amelia Dunstone <[amelia.dunstone@svi.edu.au](mailto:amelia.dunstone@svi.edu.au)>

## Contents

mockQTLE . . . . .	2
qtLe-assays . . . . .	3
QTLe-coerce . . . . .	4
qtLe-colData . . . . .	5
QTLe-combine . . . . .	6
QTLe-name . . . . .	7
QTLe-recover . . . . .	8
qtLe-rowData . . . . .	8
qtLe-row_ids . . . . .	9
qtLe-state-id . . . . .	10
QTLe-version . . . . .	11
QTLExperiment-class . . . . .	12
subset . . . . .	13
sumstats2qtLe . . . . .	14
updateObject . . . . .	16
<b>Index</b>	<b>18</b>

---

mockQTLE	<i>Mock data for the QTLExperiment object</i>
----------	---

---

### Description

Functions to create fake input data for QTLExperiments. Note that this data is generated simply, and does not have consistency between betas, errors and p-values. It is helpful to populate the slots of a QTLExperiment object and has expected properties of the betas, errors and p-values assays. Namely, betas are symmetric (specifically normally distributed), errors are non-negative, and p-values consist of a null and significant distribution. The significant effects make up 10 across states and tests.

Feature IDs are simulated by randomly selecting a feature from the list `c("geneA", "geneB", "geneC")` with replacement. Variant IDs are created by concatenating the string "snp" with a random number in the range 1000:100000. Row names combine the feature and variant IDs using a vertical line as the separator.

### Usage

```
mockQTLE(nStates = 10, nQTL = 100, names = TRUE)
```

```
mockSummaryStats(nStates = 10, nQTL = 100, names = TRUE)
```

```
mockMASHR(nStates = 10, nQTL = 100)
```

```
mockMASHR_FIT(nStates = 10, nQTL = 100)
```

### Arguments

nStates	Number of states
nQTL	Number of QTL associations
names	Logical to include column and row names

**Value**

an object containing simulated data.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**Examples**

```
nStates <- 6
nQTL <- 40

# Mock QTLExperiment data

qtLe <- mockQTLE(nStates, nQTL)
dim(qtLe)

mock_summary_stats <- mockSummaryStats(nStates=nStates, nQTL=nQTL)
mock_summary_stats$betas
mock_summary_stats$errors
mock_summary_stats$pvalues

# Mock MASHR data

mockr_sim <- mockMASHR(nStates=nStates, nQTL=nQTL)
mockr_sim$B
mockr_sim$Bhat
mockr_sim$Shat
```

---

qtLe-assays

*Named assay getters and setters*


---

**Description**

These are methods for getting or setting `assay(qtLe, i=X, ...)` where `qtLe` is a [QTLExperiment](#) object and `X` is the name of the method. For example, `betas` will get or set `X="betas"`.

**Value**

For assays, returns the value stored in the requested [assay](#).

For `assays<-value`, the relevant slot of the [QTLExperiment](#) is updated.

**Available methods**

Here `x` is a [QTLExperiment](#) object, `value` is a matrix-like object with the same dimensions as `x`, and `...` are further arguments passed to [assay](#) (for the getter) or [assay<-](#) (for the setter).

`betas(x, ...)`, `betas(x, ...) <- value`: Get or set a matrix of raw betas, i.e., QTL effect sizes.

`errors(x, ...)`, `errors(x, ...) <- value`: Get or set a matrix of raw beta standard errors.

`pvalues(x, ...)`, `pvalues(x, ...) <- value`: Get or set a matrix of raw significance scores (e.g. `pvals`, `qvals`)

`lfsrs(x, ...)`, `lfsrs(x, ...) <- value`: Get or set a matrix of local false sign rates.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**See Also**

[assay](#) and [assay<-](#), for the wrapped methods.

**Examples**

```
qtLe <- mockQTLe()  
new_betas <- matrix(rnorm(nrow(qtLe)*ncol(qtLe)), ncol=ncol(qtLe))  
row.names(new_betas) <- row.names(qtLe)  
colnames(new_betas) <- colnames(qtLe)  
betas(qtLe) <- new_betas  
dim(betas(qtLe))
```

---

QTLe-coerce

*Coercing mash data objects into QTLe objects*

---

**Description**

Function to coerce a mashr object (class list or mashr) into a QTLe object.

**Usage**

```
mash2qtLe(data, sep = NULL, rowData = NULL, verbose = FALSE)
```

```
.mashData_2_qtLe(data)
```

```
.mashFit_2_qtLe(data)
```

**Arguments**

<code>data</code>	A mashr object output from <code>mash_set_data()</code> or <code>mash()</code> from mashr.
<code>sep</code>	String separating the <code>feature_id</code> from the <code>variant_id</code> in the <code>row.names</code> of the mashr object
<code>rowData</code>	if <code>feature_id</code> and <code>variant_id</code> are not in the <code>row.names</code> , a <code>rowData</code> matrix can be provided with this information.
<code>verbose</code>	Logical.

**Value**

A [QTLeExperiment](#) object.

**Author(s)**

Christina B Azodi, Amelia Dunstone

## Examples

```
nStates <- 6
nQTL <- 40
mashr_sim <- mockMASHR(nStates, nQTL)

qtLe2 <- mash2qtLe(
  mashr_sim,
  rowData=DataFrame(
    feature_id=row.names(mashr_sim$Bhat),
    variant_id=sample(seq_len(nQTL)))
)
dim(qtLe2)
```

---

qtLe-colData

*Modify and view the colData of a QTLExperiment*

---

## Description

Methods for changing the [colData](#) of a QTLExperiment.

## Usage

```
## S4 replacement method for signature 'QTLExperiment,DataFrame'
colData(x) <- value

## S4 replacement method for signature 'QTLExperiment,NULL'
colData(x) <- value
```

## Arguments

**x** is a [QTLExperiment](#) object

**value** is a matrix-like object with number of rows equal to the number of columns in **x**.

## Details

The `state_id` column in the `colData` is protected, and operations ensure that this column is not removed from the `colData`.

## Value

For `colData`, a DFrame is returned. For `colData<-`, a modified [QTLExperiment](#) object is returned with the updated [colData](#).

## Author(s)

Christina B Azodi, Amelia Dunstone

**Examples**

```

qtle <- mockQTLE()
colData(qtle)
dim(colData(qtle))

qtle$batch <- "batch1"
colData(qtle)

# The state_id column is protected
colData(qtle) <- NULL
colData(qtle)

```

---

QTLe-combine

*Combining QTLExperiment objects*


---

**Description**

An overview of methods to combine multiple [QTLExperiment](#) objects by row or column. These methods ensure that all data fields remain synchronized when states or associations are added or removed.

**Value**

A [QTLExperiment](#) object.

**Combining**

In the following examples, `...` contains one or more [QTLExperiment](#) object.

`rbind(..., deparse.level=1)`: Returns a [QTLExperiment](#) object where all objects are combined row-wise. Metadata is combined as in `?rbind, SummarizedExperiment-method`. The `deparse.level` specifies how row.names are generated as described in `?rbind`.

`cbind(..., deparse.level=1)`: Returns a [QTLExperiment](#) object where all objects are combined column-wise. Metadata is combined as in `?cbind, SummarizedExperiment-method`. The `deparse.level` specifies how colnames are generated as described in `?cbind`.

**Author(s)**

Christina B Azodi

**Examples**

```

qtle <- mockQTLE()
qtle2 <- qtle
feature_id(qtle2) <- paste0("x", feature_id(qtle2))
rbind(qtle, qtle2)

qtle2 <- qtle
state_id(qtle2) <- paste0("x", state_id(qtle2))
cbind(qtle, qtle2)

```

---

QTLe-name	<i>Return the name of a <a href="#">QTLEperiment</a> object.</i>
-----------	--

---

### Description

Returns the name of an object of class [QTLEperiment](#).

### Arguments

object	A <a href="#">QTLEperiment</a> object.
value	Any character-like object or NULL to remove existing labels.

### Value

For `mainExpName(object)`, returns the name associated to object.

For `mainExpName(object) <- value`, the name of the object object is updated.

### Available methods

In the following code snippets, object is a [QTLEperiment](#) objects.

`mainExpName(object)`: Return the name assigned to object.

`mainExpName(object) <- value`: Change the name assigned to object to value.

`mainExpName(object) <- NULL`: Remove the name associated to object.

### Author(s)

Christina B. Azodi

### See Also

[QTLEperiment](#), for the underlying class definition.

### Examples

```
qtle <- mockQTLE()
mainExpName(qtle)
mainExpName(qtle) <- "test_name"
mainExpName(qtle)
```

---

QTLe-recover	<i>Recover QTLExperiment IDs</i>
--------------	----------------------------------

---

**Description**

Function to recover protected rowData (feature\_id, variant\_id) and colData (state\_id) from internal structure.

**Usage**

```
.recover_qtLe_ids(object)
```

**Arguments**

object	QTLExperiment object
--------	----------------------

**Value**

A [QTLExperiment](#) object with recovered rowData or colData.

---

qtLe-rowData	<i>rowData method for QTLExperiment</i>
--------------	---

---

**Description**

Methods for changing the [rowData](#) of a QTLExperiment.

**Usage**

```
## S4 method for signature 'QTLExperiment'
rowData(x, use.names = TRUE)

## S4 replacement method for signature 'QTLExperiment,DataFrame'
rowData(x) <- value

## S4 replacement method for signature 'QTLExperiment,NULL'
rowData(x) <- value
```

**Arguments**

x	is a <a href="#">QTLExperiment</a> object
use.names	is a logical specifying whether or not to propagate the rownames of x to the returned DFrame object.
value	is a matrix-like object with number of rows equal to the number of rows in x.

**Details**

The feature\_id and variant\_id columns in the rowData are protected, and operations ensure that these columns are preserved in the rowData.

**Value**

For `rowData`, a `DFrame` is returned. For `rowData<-`, a modified `QTLEExperiment` object is returned with the updated `rowData`.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**Examples**

```
qtLe <- mockQTLE()
rowData(qtLe)
dim(rowData(qtLe))

rowData(qtLe)$chr <- ifelse(feature_id(qtLe) %in% c("geneA", "geneB"), "chr1", "chr2")
rowData(qtLe)

# The state_id column is protected
rowData(qtLe) <- NULL
rowData(qtLe)
```

---

qtLe-row\_ids

*Named rowData getters and setters*


---

**Description**

These are methods for getting or setting protected `rowData` columns (i.e. `feature_id` and `variant_id`).

**Arguments**

`object` is a `QTLEExperiment` object.  
`value` is a vector with length equal to the number of rows of `x`.

**Details**

QTL are associations between a genetic variants and a quantitative feature. The `feature_id` and `variant_id` methods can be used to get or set feature IDs and variant IDs, respectively, across a `QTLEExperiment` object. The values are stored in the `rowData` compartments and have additional protections to prevent them being removed or overwritten. The `feature_id` can store gene or metabolite names, while `variant_id` could be used to store variant information such as SNP names.

**Value**

For `feature_id`, a vector is returned containing the name of the feature tested in each association. For `feature_id<-`, a modified object is returned with the updated `feature_ids` in `rowData`, and in the `row.names` of the `QTLEExperiment` object. For `variant_id`, a vector is returned containing the name of the variant tested in each association. For `variant_id<-`, a modified object is returned with the updated `variant_ids` in `rowData`, and in the `row.names` of the `QTLEExperiment` object.

**Available methods**

Here object is a [QTLExperiment](#) object, value is a vector-like object with compatible dimensions to object.

`feature_id(object)`: Get the feature names.

`feature_id(object) <- value`: Set the feature names.

`variant_id(object)`: Get the variant names.

`variant_id(object) <- value`: Set the variant names.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**See Also**

[QTLExperiment](#), for the underlying class definition.

**Examples**

```
qtle <- mockQTLE()
feature_id(qtle) <- gsub("gene", "Gene", feature_id(qtle))
feature_id(qtle)
variant_id(qtle) <- paste0(variant_id(qtle), "000")
variant_id(qtle)
```

---

qtle-state-id

*Modify and view state ID*

---

**Description**

These are methods for getting or setting protected colData columns (i.e. `state_id`).

**Usage**

```
## S4 method for signature 'QTLExperiment'
state_id(object)

## S4 replacement method for signature 'QTLExperiment'
state_id(object) <- value
```

**Arguments**

`object` is a [QTLExperiment](#) object  
`value` is a character vector with length equal to the number of columns/states in object.

**Details**

QTL are associations between a genetic variant and a quantitative state. The `state_id` methods can be used to get or set state IDs for all tests in a [QTLExperiment](#) object. The values are stored in the `colData` as the `state_id` field so it can be easily accessed but not accidentally removed or overwritten.

**Value**

For `state_id`, a vector is returned containing the name of the state tested in each association. For `state_id<-`, a modified object is returned with the updated `state_ids` in `colData`, and in the `row.names` of the `QTLEExperiment` object.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**Examples**

```
qtle <- mockQTLE()
state_id(qtle) <- gsub("state", "State_", state_id(qtle))
state_id(qtle)
```

---

QTLe-version	<i>Return the version of a <a href="#">QTLEExperiment</a> object</i>
--------------	--

---

**Description**

Specifies the version of the `QTLEExperiment` package that an object of class `QTLEExperiment` was created with.

**Arguments**

`object`            A `QTLEExperiment` object.

**Value**

A package version, of class `package_version`.

**Available methods**

In the following code snippets, `object` is a `QTLEExperiment` objects.

`objectVersion(object)`: Return the version of the package with which `object` was constructed.

**Author(s)**

Christina B. Azodi, Amelia Dunstone

**See Also**

[QTLEExperiment](#), for the underlying class definition and [updateObject](#) to update the object to the latest version.

**Examples**

```
qtle <- mockQTLE()
objectVersion(qtle)
```

---

QTLExperiment-class     *An S4 class to represent QTL summary statistics.*

---

### Description

The QTLExperiment class is designed to represent multi-state QTL data. It inherits from the [RangedSummarizedExperiment](#) class. In addition, the class supports storage of multi-state adjusted beta and betaSE results (e.g., mash) and storage of summary results (e.g., pairwise sharing).

### Arguments

...	Arguments passed to the <a href="#">SummarizedExperiment</a> constructor to fill the slots of the base class.
state_id	An array of state IDs the length of ncol(qtle).
feature_id	An array of feature IDs the length of nrow(qtle).
variant_id	An array of variant IDs the length of nrow(qtle).

### Details

In this class, rows should represent associations (feature\_id:variant\_id pairs) while columns represent states (e.g. tissues). Assays include betas and error associated with the betas (e.g. standard errors). As with any [SummarizedExperiment](#) derivative, different information (e.g., test-statistics, significance calls) can be stored in user defined [assay](#) slots, and additional row and column metadata can be attached using [rowData](#) and [colData](#), respectively.

The extra arguments in the constructor ([feature\\_id](#), [variant\\_id](#), and [state\\_id](#)) represent the main extensions implemented in the QTLExperiment class. This enables a consistent, formalized representation of key aspects of multi-state QTL data that are universal to the data structure. Readers are referred to the specific documentation pages for more details.

A QTLE can also be coerced from a [SummarizedExperiment](#) or [RangedSummarizedExperiment](#) instance.

### Value

A QTLExperiment object.

### Slots

[elementMetadata](#) A [DataFrame](#) containing at minimum [feature\\_id](#) and [variant\\_id](#) information. This is accessed using [rowData](#).

[colData](#) A [DataFrame](#) containing at minimum [state\\_id](#) information.

[int\\_metadata](#) A list of additional metadata items to store.

### Author(s)

Christina B Azodi

**Examples**

```

nStates <- 10
nQTL <- 100
betas <- matrix(rnorm(nStates * nQTL), ncol=nStates)
error <- matrix(abs(rnorm(nStates * nQTL)), ncol=nStates)

qtle <- QTLEperiment(
  assays=list(betas=betas, errors=error),
  feature_id=sample(1:10, nQTL, replace=TRUE),
  variant_id=sample(seq(1e3,1e5), nQTL),
  state_id=LETTERS[1:nStates])
qtle

## coercion from SummarizedExperiment
mock_sumstats <- mockSummaryStats(nStates=10, nQTL=100)
se <- SummarizedExperiment(
  assays=list(
    betas=mock_sumstats$betas,
    errors=mock_sumstats$errors))
as(se, "QTLEperiment")

```

subset

*Subsetting and replacing data in QTLEperiment objects***Description**

Includes methods to subset a [QTLEperiment](#) object by row and/or column and methods to replace all data for the specified rows and/or columns with another value. These methods ensure that all data fields remain synchronized when states or associations are removed. The [QTLEperiment](#) object is compatible with subsetting using square brackets or with the subset function.

**Usage**

```

## S4 method for signature 'QTLEperiment,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'QTLEperiment,ANY,ANY,QTLEperiment'
x[i, j, ...] <- value

## S4 method for signature 'QTLEperiment'
subset(x, i, ...)

```

**Arguments**

x	is a <a href="#">QTLEperiment</a> object
i	is a vector of subscripts indicating the rows to retain.
j	is a vector of subscripts indicating the columns to retain.
...	Further arguments to the subset function are passed to <code>S4Vectors:::evalqForSubset</code> .
drop	is passed to <code>SummarizedExperiment:::SummarizedExperiment-class</code> but is not used by those methods.
value	is the contents to be used for replacement.

**Value**

A [QTLExperiment](#) object.

**Subsetting**

In the following, `x` is a [QTLExperiment](#) object.

`x[i, j, ..., drop=TRUE]`: Returns a [QTLExperiment](#) containing the specified rows `i` and columns `j`, where `i` and `j` can be a logical, integer or character vector of subscripts, indicating the rows and columns, respectively, to retain. If either `i` or `j` is missing, than subsetting is only performed in the specified dimension. Arguments in `...` and `drop` are passed to [\[, SummarizedExperiment-method\]](#).

**Replacing**

In the following, `x` is a [QTLExperiment](#) object.

`x[i, j, ...] <- value`: Replaces all data for rows `i` and columns `j` with the corresponding fields in a [QTLExperiment](#) value, where `i` and `j` can be a logical, integer, or character vector of subscripts, indicating the rows and columns, respectively, to retain. If either `i` or `j` is missing, than subsetting is only performed in the specified dimension. If both are missing, `x` is replaced entirely with `value`. Arguments in `...` are passed to the corresponding [SummarizedExperiment](#) method.

**Author(s)**

Christina B Azodi

**Examples**

```
qtle <- mockQTLE()

# Subsetting:
qtle[1:10,]
qtle[,1:5]

# Can also use subset()
qtle$WHEE <- sample(c("A", "B", "C"), ncol(qtle), replace=TRUE)
subset(qtle, , WHEE=="A")

# Can also use split()
split(qtle, sample(c("A", "B", "C"), nrow(qtle), replace=TRUE))
```

---

sumstats2qtle

*Coerce QTL summary statistics into a QTLExperiment object*

---

**Description**

A suite of methods to extract QTL mapping summary statistics from common QTL workflow output files.

**Usage**

```

sumstats2qtle(
  input,
  feature_id = "gene_id",
  variant_id = "variant_pos",
  betas = "slope",
  errors = "slope_se",
  pvalues = NULL,
  check_dupes = FALSE,
  gene_filter = NULL,
  col_types = NULL,
  delayed = FALSE,
  n_max = Inf,
  verbose = TRUE,
  otherFields = NULL
)

```

**Arguments**

input	Named array or data.frame with state name and the file to the QTL summary statistics for that state. If data.frame is provided, it must include columns 'state' and 'path'. Additional columns will be stored in the colData annotation.
feature_id	The name/index of the column with the feature_id info.
variant_id	The name/index of the column with the variant_id info.
betas	The name/index of the column with the effect size/beta value.
errors	The name/index of the column with the effect size/beta standard error value.
pvalues	The name/index of the column with the significance score.
check_dupes	logical. Whether to check for duplicate tests for some combinations of state, feature ID and variant ID. This is necessary for some data (e.g., xQTLatlas) where multiple genetic variants were tested for each feature and state. To avoid col_lists in the output object, this argument subsets to the first test for each combination. Note that checking that tests are unique can slow down run-times so should be avoided if not necessary.
gene_filter	A subset of the feature IDs to be retained in the object. Useful if you would like to load all variant ID tests for a particular feature, e.g., all tests in a particular region of a genome (e.g., for locus plots).
col_types	Column types e.g., a compact string "ccddd". For more information see <a href="#">vroom</a> .
delayed	Logical scalar indicating whether matrices should be wrapped in <a href="#">DelayedArray</a> . Not yet implemented.
n_max	Max number of rows to read per file. This is primarily used for testing purposes.
verbose	logical. Whether to print progress messages.
otherFields	The names of additional columns from the data that are to be passed into the QTLExperiment rowData. For example, this could include the positions of the variant IDs in terms of base pairs, or the name of the alternative allele.

**Value**

A [QTLExperiment](#) object.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**Examples**

```
input_path <- system.file("extdata", package = "QTLExperiment")
state <- c("lung", "thyroid", "spleen", "blood")

# Input as a named array
input_list <- list(lung = paste0(input_path, "/GTEx_tx_lung.tsv"),
                  spleen = paste0(input_path, "/GTEx_tx_spleen.tsv"))

# Input as a data.frame.
# Must include columns 'state' and 'path'.
input_df <- data.frame(state = c("lung", "spleen"),
                      path = c(paste0(input_path, "/GTEx_tx_lung.tsv"),
                               paste0(input_path, "/GTEx_tx_spleen.tsv")))

# List version
qt1e1 <- sumstats2qt1e(input_list,
                      feature_id="molecular_trait_id",
                      variant_id="rsid",
                      betas="beta",
                      errors="se",
                      pvalues="pvalue",
                      verbose=TRUE)

qt1e1
head(betas(qt1e1))

# data.frame version
qt1e2 <- sumstats2qt1e(input_df,
                      feature_id="molecular_trait_id",
                      variant_id="rsid",
                      betas="beta",
                      errors="se",
                      pvalues="pvalue",
                      verbose=TRUE)

qt1e2
head(betas(qt1e2))
```

---

updateObject

*Update a QTLExperiment object*

---

**Description**

Update [QTLExperiment](#) objects to the latest version of the class structure. This is usually called by internal methods rather than by users or downstream packages.

**Usage**

```
## S4 method for signature 'QTLExperiment'
updateObject(object, ..., verbose = FALSE)
```

**Arguments**

object	An old <a href="#">QTLExperiment</a> object.
...	Additional arguments that are ignored.
verbose	Logical scalar indicating whether a message should be emitted as the object is updated.

**Details**

This function updates the [QTLExperiment](#) to match changes in the internal class representation. Changes are as follows:

- No updates yet.

**Value**

An updated version of object.

**Author(s)**

Christina B Azodi

**See Also**

[objectVersion](#), which is used to determine if the object is up-to-date.

# Index

- `.mashData_2_qtLe` (QTLe-coerce), 4
- `.mashFit_2_qtLe` (QTLe-coerce), 4
- `.recover_qtLe_ids` (QTLe-recover), 8
- `[,QTLEExperiment,ANY,ANY,ANY-method`  
    (subset), 13
- `[,QTLEExperiment,ANY,ANY-method`  
    (subset), 13
- `[,QTLEExperiment,ANY-method` (subset), 13
- `[<- ,QTLEExperiment,ANY,ANY,QTLEExperiment-method`  
    (subset), 13
  
- assay, 3, 4, 12
  
- betas (qtLe-assays), 3
- betas,QTLEExperiment-method  
    (qtLe-assays), 3
- betas<- (qtLe-assays), 3
- betas<- ,QTLEExperiment-method  
    (qtLe-assays), 3
  
- cbind, 6
- cbind,QTLEExperiment-method  
    (QTLe-combine), 6
- coerce,RangedSummarizedExperiment,QTLEExperiment-method  
    (QTLEExperiment-class), 12
- coerce,SummarizedExperiment,QTLEExperiment-method  
    (QTLEExperiment-class), 12
- colData, 5, 10–12
- colData (qtLe-colData), 5
- colData,QTLEExperiment-method  
    (qtLe-colData), 5
- colData<- (qtLe-colData), 5
- colData<- ,QTLEExperiment,ANY-method  
    (qtLe-colData), 5
- colData<- ,QTLEExperiment,DataFrame-method  
    (qtLe-colData), 5
- colData<- ,QTLEExperiment,NULL-method  
    (qtLe-colData), 5
  
- DelayedArray, 15
  
- errors (qtLe-assays), 3
- errors,QTLEExperiment-method  
    (qtLe-assays), 3
- errors<- (qtLe-assays), 3
  
- errors<- ,QTLEExperiment-method  
    (qtLe-assays), 3
  
- feature\_id, 9, 12
- feature\_id (qtLe-row\_ids), 9
- feature\_id,QTLEExperiment-method  
    (qtLe-row\_ids), 9
- feature\_id<- (qtLe-row\_ids), 9
- feature\_id<- ,QTLEExperiment-method  
    (qtLe-row\_ids), 9
  
- lfsrs (qtLe-assays), 3
- lfsrs,QTLEExperiment-method  
    (qtLe-assays), 3
- lfsrs<- (qtLe-assays), 3
- lfsrs<- ,QTLEExperiment-method  
    (qtLe-assays), 3
  
- mainExpName (QTLe-name), 7
- mainExpName,QTLEExperiment-method  
    (QTLe-name), 7
- mainExpName<- (QTLe-name), 7
- mainExpName<- ,QTLEExperiment,character\_OR\_NULL-method  
    (QTLe-name), 7
- mash2qtLe (QTLe-coerce), 4
- mockMASHR (mockQTLE), 2
- mockMASHR\_FIT (mockQTLE), 2
- mockQTLE, 2
- mockSummaryStats (mockQTLE), 2
  
- objectVersion, 17
- objectVersion (QTLe-version), 11
- objectVersion,QTLEExperiment-method  
    (QTLe-version), 11
  
- package\_version, 11
- pvalues (qtLe-assays), 3
- pvalues,QTLEExperiment-method  
    (qtLe-assays), 3
- pvalues<- (qtLe-assays), 3
- pvalues<- ,QTLEExperiment-method  
    (qtLe-assays), 3
  
- qtLe-assays, 3
- QTLe-coerce, 4

qtLe-colData, 5  
QTLe-combine, 6  
QTLe-name, 7  
QTLe-recover, 8  
qtLe-row\_ids, 9  
qtLe-rowData, 8  
qtLe-state-id, 10  
QTLe-version, 11  
QTLExperiment, 3–11, 13–17  
QTLExperiment (QTLExperiment-class), 12  
QTLExperiment-class, 12

RangedSummarizedExperiment, 12  
rbind, 6  
rbind, QTLExperiment-method  
    (QTLe-combine), 6  
rowData, 8, 9, 12  
rowData (qtLe-rowData), 8  
rowData, QTLExperiment-method  
    (qtLe-rowData), 8  
rowData<- (qtLe-rowData), 8  
rowData<-, QTLExperiment, DataFrame-method  
    (qtLe-rowData), 8  
rowData<-, QTLExperiment, NULL-method  
    (qtLe-rowData), 8  
rowData<-, QTLExperiment-method  
    (qtLe-rowData), 8

state\_id, 10, 12  
state\_id (qtLe-state-id), 10  
state\_id, QTLExperiment-method  
    (qtLe-state-id), 10  
state\_id<- (qtLe-state-id), 10  
state\_id<-, QTLExperiment-method  
    (qtLe-state-id), 10  
subset, 13  
subset, QTLExperiment-method (subset), 13  
SummarizedExperiment, 12, 14  
sumstats2qtLe, 14

updateObject, 11, 16  
updateObject, QTLExperiment-method  
    (updateObject), 16

variant\_id, 9, 12  
variant\_id (qtLe-row\_ids), 9  
variant\_id, QTLExperiment-method  
    (qtLe-row\_ids), 9  
variant\_id<- (qtLe-row\_ids), 9  
variant\_id<-, QTLExperiment-method  
    (qtLe-row\_ids), 9  
vroom, 15