Package ‘QTLExperiment’

March 7, 2024

Type Package
Date 2023-09-28
Title S4 classes for QTL summary statistics and metadata
Version 1.0.0
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 7.2.3

git_url https://git.bioconductor.org/packages/QTLExperiment
git_branch RELEASE_3_18
git_last_commit eb387bf
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-06
mockQTLE

Mock data for the QTLExperiment object

Description

Functions to create fake input data for QTLExperiments.

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
qtle-assays

Value

an object containing simulated data.

Author(s)

Christina B Azodi, Amelia Dunstone

Examples

nStates <- 6
nQTL <- 40

# Mock QTLEperiment 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$8
mockr_sim$8hat
mockr_sim$8hat

qtle-assays Named assay getters and setters

Description

These are methods for getting or setting assay(qtle, i=X, ...) where qtle is a QTLEperiment 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 QTLEperiment is updated.

Available methods

Here x is a QTLEperiment 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

qtte <- mockQTLE()
new_betas <- matrix(rnorm(nrow(qtte)*ncol(qtte)), ncol=ncol(qtte))
row.names(new_betas) <- row.names(qtte)
colnames(new_betas) <- colnames(qtte)
betas(qtte) <- new_betas
dim(betas(qtte))

---

## QTLcoerce

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

data A mashr object output from mash_set_data() or mash() from mashr.

sep String separating the feature_id from the variant_id in the row.names of the mashr object.

rowData if feature_id and variant_id are not in the row.names, a rowData matrix can be provided with this information.

verbose Logical.
Value

A QTLExperiment object.

Author(s)

Christina B Azodi, Amelia Dunstone

Examples

```r
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-col_ids

**Named colData getters and setters**

**Description**

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

**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 and in the int_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, int_colData, and in the row.names of the QTLExperiment object.

**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 state_id (for the getter) or state_id<~ (for the setter).

```
state_id(x, ...), state_id(x, ...) <- value: Get or set the state (i.e. column) names.
```
Author(s)

Christina B Azodi

Examples

```r
qtle <- mockQTLE()
state_id(qtle) <- sample(LETTERS, ncol(qtle), replace=TRUE)
state_id(qtle)
```

<table>
<thead>
<tr>
<th>QTL_e-combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combining QTLExperiment objects</td>
</tr>
</tbody>
</table>

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

```r
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)
```
Description

Methods to get or set internal fields from the QTLExperiment class. These functions are intended for package developers who want to make changes or improvements to the object without breaking user code or to add protected fields to a QTLExperiment. They should not be used by general users.

Value

For assays, returns the value stored in the requested field of the internal rowData, colData or metaData.

For assays<-value, the relevant internal field of the QTLExperiment is updated.

Getters

Here x is a QTLExperiment.

int_rowData(x): Returns a DataFrame of internal row metadata, with number of rows equal to nrow(x) (analogous to the user-visible rowData).

int_colData(x): Returns a DataFrame of internal column metadata, with number of rows equal to ncol(x) (analogous to the user-visible colData).

int_metadata(x): Returns a list of internal metadata (analogous to the user-visible metadata).

The following methods can return visible and internal data in a single DataFrame.

rowData(x, ..., internal=TRUE): Returns a DataFrame of the user-visible row metadata with the internal row metadata added column-wise. A warning is emitted if the user-visible metadata column names overlap with the internal fields. Any arguments in ... are passed to rowData,SummarizedExperiment-method.

colData(x, ..., internal=TRUE): Returns a DataFrame of the user-visible column metadata with the internal column metadata added column-wise. A warning is emitted if the user-visible metadata column names overlap with the internal fields. Any arguments in ... are passed to colData,SummarizedExperiment-method.

Setters

Here x is a QTLExperiment.

int_rowData(x) <- value: Replaces the internal row metadata with value, a DataFrame with number of rows equal to nrow(x) (analogous to the user-visible rowData<-).

int_colData(x) <- value: Replaces the internal column metadata with value, a DataFrame with number of rows equal to ncol(x) (analogous to the user-visible colData<-).

int_metadata(x) <- value: Replaces the internal metadata with value (analogous to the user-visible metadata<-).
Comments
The internal metadata fields store additional elements that are parallel to the rows or columns of a QTLExperiment class. This avoids the need to specify new slots and adjust the subsetting/combining code for a new data element. These elements being internal is important as it ensures that the implementation details are abstracted away. User interaction with these internal fields should be done via the designated getter and setter methods (e.g., feature_id), providing developers with freedom to change the internal representation without breaking user code.

Author(s)
Christina B Azodi

See Also
colData, rowData and metadata for the user-visible equivalents.

Examples
```r
tle <- mockQTL() int_metadata(tle)$whee <- 1
```

Description
Return the name of a QTLExperiment object.

Arguments
- `x` A QTLExperiment object.
- `value` Any character-like object or NULL to remove existing labels.

Value
For mainExpName(x), returns the name associated to x.
For mainExpName(x) <- value, the name of the object x is updated.

Available methods
In the following code snippets, x is a QTLExperiment objects.
mainExpName(x): Return the name assigned to x.
mainExpName(x) <- value: Change the name assigned to x to value.
mainExpName(x) <- NULL: Remove the name associated to x.


**Description**

Function to recover protected rowData (feature_id, variant_id) and colData (state_id) from internal structure.

**Usage**

```r
recover_qtle_ids(object)
```

**Arguments**

- `object` QTLExperiment object

**Value**

A QTLExperiment object with recovered rowData or colData.

**Examples**

# Recover a column in colData

```r
qtle <- mockQTL()  
head(colData(qtle))

new_colData <- DataFrame(
  list(some_info1=LETTERS[1:ncol(qtle)],
       some_info2=c(1:ncol(qtle))))

# colData is overwritten
colData(qtle) <- new_colData
```
```

head(colData(qtle))

# colData is recovered
qtle <- recover_qtle_ids(qtle)
head(colData(qtle))

# Recover information from rowData

head(rowData(qtle))

# variant_id are shuffled accidentally
rowData(qtle)$variant_id <- sample(rowData(qtle)$variant_id, nrow(qtle))
head(rowData(qtle))

# Recover rowData
qtle <- recover_qtle_ids(qtle)
head(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).

#### 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 `QTLExperiment` object. The values are stored in the `rowData` and in the `int_rowData` compartments so they can be easily accessed but not accidentally removed or overwritten.

#### 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`, `int_rowData`, and in the row.names of the `QTLExperiment` 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`, `int_rowData`, and in the row.names of the `QTLExperiment` object.

#### 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 `feature_id` (for the getter) or `feature_id<->` (for the setter).

- `feature_id(x, ...)`, `feature_id(x, ...) <- value`: Get or set the feature (e.g. gene, metabolite) names.
- `variant_id(x, ...)`, `variant_id(x, ...) <- value`: Get or set the variant (i.e. SNP) names.
**Author(s)**

Christina B Azodi

**See Also**

`QTLExperiment`, for the underlying class definition.

**Examples**

```r
qtle <- mockQTLE()
feature_id(qtle) <- sample(LETTERS, nrow(qtle), replace=TRUE)
feature_id(qtle)
variant_id(qtle) <- sample(paste0("rsid", 1:100), nrow(qtle), replace=TRUE)
variant_id(qtle)
```

---

**Description**

Includes methods to subset a `QTLExperiment` 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.

**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
```r
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))
```

---

**QTLe-version**

Return the version of a *QTLExperiment* object

---

Description

Specifies the version of the *QTLExperiment* package that an object of class *QTLExperiment* was created with.

Arguments

- `x` A *QTLExperiment* object.

Value

A package version, of class *package_version*.

Available methods

In the following code snippets, `x` is a *QTLExperiment* objects.

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

Author(s)
Christina B. Azodi, Amelia Dunstone

See Also

*QTLExperiment*, for the underlying class definition and *updateObject* to update the object to the latest version.
QTLExperiment-class

Examples

```r
qtle <- mockQTLEx()
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 SummarizedExperiment constructor to fill the slots of the base class.
state_id An array of state IDs the length of ncol(QTLEx).
feature_id An array of feature IDs the length of nrow(QTLEx).
variant_id An array of variant IDs the length of nrow(QTLEx).

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. that are commonly encountered during single-cell data analysis. 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

int_rowData A DataFrame containing at minimum feature_id and variant_id information
int_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**

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

qtle <- QTLExperiment(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, "QTLExperiment")
```

**Description**

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

**Usage**

```r
sumstats2qtle(
    input, 
    feature_id = "gene_id", 
    variant_id = "variant_pos", 
    betas = "slope", 
    errors = "slope_se", 
    pvalues = NULL, 
    n_max = Inf, 
    verbose = TRUE
)
```

**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.

**n_max**  Max number of rows to read per file. This is primarily used for testing purposes.

**verbose**  logical. Whether to print progress messages.

**Value**

A QTLExperiment object.

**Author(s)**

Christina B Azodi, Amelia Dunstone

**Examples**

```r
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
qtle1 <- sumstats2qtle(input_list,
                       feature_id="molecular_trait_id",
                       variant_id="rsid",
                       betas="beta",
                       errors="se",
                       pvalues="pvalue",
                       verbose=TRUE)

qtle1
head(betas(qtle1))

# data.frame version
qtle2 <- sumstats2qtle(input_df,
                       feature_id="molecular_trait_id",
                       variant_id="rsid",
                       betas="beta",
                       errors="se",
                       pvalues="pvalue",
                       verbose=TRUE)
```
updateObject

qtle2
head(betas(qtle2))

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 QTLExperiment 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.
Examples

```r
qtle <- mockQTE()
objectVersion(qtle)

qtle_new <- QTLExperiment::updateObject(qtle)
```
INDEX

mainExpName<-,QTLExperiment,character_OR_NULL-method
(QTLe-name), 8
mash2qtle (QTLe-coerce), 4
metadata, 7, 8
mockMASHR (mockQTLE), 2
mockMASHR_FIT (mockQTLE), 2
mockQTLE, 2
mockSummaryStats (mockQTLE), 2

objectVersion, 16
objectVersion (QTLe-version), 12
objectVersion, QTLExperiment-method
(QTLe-version), 12

package_version, 12
parallel_slot_names, QTLExperiment-method
(QTLe-internals), 7
pvalues (qtle-assays), 3
pvalues, QTLExperiment-method
(qtle-assays), 3
pvalues<-(qtle-assays), 3
pvalues<-, QTLExperiment-method
(qtle-assays), 3

qtle-assays, 3
QTLe-coerce, 4
qtle-col_ids, 5
QTLe-combine, 6
QTLe-internals, 7
QTLe-name, 8
QTLe-recover, 9
qtle-row_ids, 10
QTLe-subset, 11
QTLe-version, 12
QTLExperiment, 3, 5–12, 15, 16
QTLExperiment (QTLExperiment-class), 13
QTLExperiment-class, 13

RangedSummarizedExperiment, 13
rbind, 6
rbind, QTLExperiment-method
(QTLe-combine), 6
recover_qtle_ids (QTLe-recover), 9
rowData, 7, 8, 10, 13
rowData, QTLExperiment-method
(QTLe-internals), 7

state_id, 5, 13
state_id (qtle-col_ids), 5