Package 'multistateQTL'

November 12, 2026

Type Package

Date 2025-03-07

Title Toolkit for the analysis of multi-state QTL data

Version 2.4.1

Description A collection of tools for doing various analyses of multi-state QTL data, with a focus on visualization and interpretation. The package 'multistateQTL' contains functions which can remove or impute missing data, identify significant associations, as well as categorise features into global, multi-state or unique. The analysis results are stored in a 'QTLExperiment' object, which is based on the 'SummarisedExperiment' framework.

License GPL-3

URL https://github.com/dunstone-a/multistateQTL

BugReports https://github.com/dunstone-a/multistateQTL/issues

Encoding UTF-8

Depends QTLExperiment, SummarizedExperiment, ComplexHeatmap, collapse

Imports methods, S4Vectors, data.table, grid, dplyr, tidyr, matrixStats, stats, fitdistrplus, viridis, ggplot2, circlize, mashr, grDevices

Suggests testthat, BiocStyle, knitr, covr, rmarkdown

biocViews FunctionalGenomics, GeneExpression, Sequencing, Visualization, SNP, Software

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

git_url https://git.bioconductor.org/packages/multistateQTL

git_branch RELEASE_3_23

git_last_commit 1ab09a2

git_last_commit_date 2026-10-08

Repository Bioconductor 3.23

Date/Publication 2026-11-11

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>

Contents

multistateQTL-package 2
callSignificance 3
getComplete 4
getSignificant 5
getTopHits 6
plotCompareStates 7
plotPairwiseSharing 8
plotQTLClusters 10
plotSimulationParams 11
plotUpSet 12
qtleEstimate 13
qtleParams 14
qtleSimulate 15
replaceNAs 16
runPairwiseSharing 17
runSignificantFeatures 18
runTestMetrics 19
simPerformance 20

Index **21**

`multistateQTL-package` *multistateQTL: Toolkit for the analysis of multi-state QTL data*

Description

A collection of tools for doing various analyses of multi-state QTL data, with a focus on visualization and interpretation. The package 'multistateQTL' contains functions which can remove or impute missing data, identify significant associations, as well as categorise features into global, multi-state or unique. The analysis results are stored in a 'QTLExperiment' object, which is based on the 'SummarisedExperiment' framework.

Author(s)

Maintainer: Amelia Dunstone <amelia.dunstone@svi.edu.au> (ORCID)

Authors:

- Amelia Dunstone <amelia.dunstone@svi.edu.au> (ORCID)
- Christina Del Azodi <cazodi@svi.edu.au>

Other contributors:

- Davis McCarthy <dmccarthy@svi.edu.au> [contributor]

See Also

Useful links:

- https://github.com/dunstone-a/multistateQTL
- Report bugs at https://github.com/dunstone-a/multistateQTL/issues

---

`callSignificance` *Call associations as significant in each state.*

Description

Call associations as significant in each state.

Usage

```
callSignificance(object, ...)

callSignificance(object, ...) <- value

## S4 method for signature 'QTLExperiment'
callSignificance(
  object,
  thresh = 0.05,
  secondThresh = thresh,
  feature = .feature_id,
  assay = "pvalues",
  mode = "simple",
  p.adjust.method = "fdr",
  ...
)
```

Arguments

`object`	A QTLExperiment object.
`...`	arguments passed to `callSignificance`
`value`	Value to place in getSignificance
`thresh`	Significance threshold.
`secondThresh`	Significance threshold for associations with significance in one state.
`feature`	rowData column name with feature identifier.
`assay`	assay name with significance score.

mode — Method to determine significance threshold per state. Options are ‘simple‘, ‘feature-wise-FDR‘, and ‘global-FDR‘.

p.adjust.method — Method of multiple-test correction if mode != simple

Details

This function adds a new assay to multistateQTL object with TRUE/FALSE significance calls for each test for each state.

Value

A ‘QTLExperiment‘ object with a new assay called ‘significant‘ and with a column called nSignificant added to the colData.

Author(s)

Christina B Azodi

See Also

wilcox.test, on which this function is based.

Examples

```
qtle <- mockQTLE()

assays(qtle)
qtle <- callSignificance(qtle)

# There is now an assay called 'significant'
assays(qtle)

# Use feature-wise FDR correction ------------------------------------------
qtle_feat <- callSignificance(qtle, thresh=0.1, mode="feature-wise-FDR")
```

---

getComplete *Filter QTLExperiment based on missing data*

Description

Method to filter QTLExperiment objects to remove tests with greater than the permitted rate of missing values.

Usage

```
getComplete(qtle, n = 1, verbose = FALSE)
```

Arguments

qtle	A 'QTLExperiment' object
n	Number (or percent if n < 1) of states requiring non-null values
verbose	logical. Whether to print progress messages.

Value

a subset of the 'QTLExperiment' object, with only tests with fewer NAs than specified by n.

Examples

```
# Create a QTLExperiment object with NA values ------------------------------
sim <- qtleSimulate(
    nStates=10, nFeatures=100, nTests=1000,
    global=0.2, multi=0.4, unique=0.2, k=2)

# Randomly remove 1000 elements from the betas matrix.
na_pattern <- sample(seq(1, ncol(sim)*nrow(sim)), 1000)
sim_na <- sim
assay(sim_na, "betas")[na_pattern] <- NA

# Original object has more rows than the output of getComplete()
dim(sim_na)

sim_complete <- getComplete(sim_na)
dim(sim_complete)
```

---

getSignificant *Filter to only QTLs significant in at least one state*

Description

Filter to only QTLs significant in at least one state

Usage

```
getSignificant(qtle, n = 1, assaySig = "significant", verbose = FALSE)
```

Arguments

qtle	'QTLExperiment' object
n	Number (or percent if n < 1) of states with significant association
assaySig	The assay containing TRUE/FALSE significance calls for each QTL test.
verbose	logical. Whether to print progress messages.

Value

a subset of the 'QTLExperiment' object, where all rows are significant in at least one state.

Examples

```
qtle <- mockQTLE()

qtle <- callSignificance(qtle)
dim(qtle)
qtle_sig <- getSignificant(qtle)

# There are fewer rows because we have removed tests which are not significant
# in any state.
dim(qtle_sig)
```

---

getTopHits *Filter QTLExperiment to keep only top hits*

Description

Method to return a subset of a QTLExperiment object containing only the tests that are top hits. Top hits are defined as the test for each feature with the most significant test statistic. Returns an array of the top QTL for each feature across all states

Usage

```
getTopHits(
  qtle,
  mode = c("global", "state"),
  assay = "pvalues",
  assaySig = "significant",
  verbose = FALSE
)
```

Arguments

qtle	A 'QTLExperiment' object
mode	global/state to specify if the top hit per feature is desired from across all states or for each state.
assay	The assay containing the test statistic to minimize.
assaySig	The assay containing TRUE/FALSE significance calls for each QTL test.
verbose	logical. Whether to print progress messages.

Value

A subset of the 'QTLExperiment' object, with only tests that are the top hits for each feature ('mode=global') or for each feature for each state ('mode=state').

Examples

```
sumstats <- mockSummaryStats(nStates=10, nQTL=100, names=TRUE)
qtle <- QTLExperiment(
    assay=list(
    betas=sumstats$betas,
    errors=sumstats$errors,
    pvalues=sumstats$pvalues,
    lfsrs=sumstats$pvalues))

# Add 'significant' assay to object
qtle <- callSignificance(qtle)

# Filter to the top tests for each feature
qtle_glob <- getTopHits(qtle, assay="lfsrs", mode="global", verbose = TRUE)
# There are 3 rows corresponding to the three features.
table(feature_id(qtle_glob))

# At most one QTL is retained for each combination of feature_id and state_id
qtle_feat <- getTopHits(qtle, assay="lfsrs", mode="state", verbose = TRUE)
table(feature_id(qtle_feat))
```

---

plotCompareStates *Compare QTL between two states*

Description

Convenience method for comparing the assay value, specified by assay, between two states.

Usage

```
plotCompareStates(
  object,
  x,
  y,
  assay = "betas",
  FUN = identity,
  assaySig = "significant",
  alpha = 0.2,
  colBoth = "#4477AA",
  colDiverging = "#EE6677",
  colNeither = "gray50",
  colX = "#CCBB44",
  colY = "#AA3377"
)
```

Arguments

object	an QTLExperiment object.
x	Name of state for x-axis
y	Name of state for y-axis
assay	name of assay to plot.
FUN	Function to be applied to fillBy assay before plotting (e.g. identity, abs, log10).
assaySig	name of assay with TRUE/FALSE significance calls.
alpha	Transparency.
colBoth	Color for tests significant in both states.
colDiverging	Color for tests significant in both states, with diverging effect sizes.
colNeither	Color for null tests.
colX	Color for tests significant in the x-axis state only.
colY	Color for tests significant in the y-axis state only.

Value

Returns a list containing the counts for each color category and the plot object.

Examples

```
sim <- qtleSimulate(
    nStates=10, nFeatures=100, nTests=1000,
    global=0.2, multi=0.4, unique=0.2, k=2)
sim <- callSignificance(sim, mode="simple", assay="lfsrs",
    thresh=0.0001, secondThresh=0.0002)
sim_sig <- getSignificant(sim)
sim_top <- getTopHits(sim_sig, assay="lfsrs", mode="state")
sim_top <- runPairwiseSharing(sim_top)
sim_top <- runTestMetrics(sim_top)
plotCompareStates(sim_top, x="S01", y="S02")
```

---

plotPairwiseSharing *Heatmap of pairwise QTL sharing with state-level annotations*

Description

Methods to plot a heatmap of the pairwise sharing of QTL as calculated by ‘runPairwiseSharing‘.

Usage

```
plotPairwiseSharing(
  object,
  slot = "pairwiseSharing",
  annotateRowsBy = NULL,
  annotateColsBy = NULL,
  annotateCells = FALSE,
  colourRange = NULL,
  name = "colnames",
  distMethod = "pearson",
  size = 8,
  ...
)
```

Arguments

object	A QTLExperiment object
slot	Name of slot in metadata list with Pairwise Sharing matrix.
annotateRowsBy	character or array of characters specifying the column(s) in colData to be plotted as row annotations.
annotateColsBy	character or array of characters specifying the column(s) in colData to be plotted as column annotations.
annotateCells	Logical to annotate cells with their values.
colourRange	Optional range for the color legend
name	character specifying the column in colData to use to label rows and columns. Default is colnames(qtle).
distMethod	Distance method used for hierarchical clustering. Valid values are the supported methods in dist() function.
size	numeric scalar giving default font size for plotting theme.
...	Further arguments passed to Heatmap.

Value

Returns a ComplexHeatmap object.

Examples

```
sim <- qtleSimulate(
    nStates=10, nFeatures=100, nTests=1000,
    global=0.2, multi=0.4, unique=0.2, k=2)
sim <- callSignificance(sim, mode="simple", assay="lfsrs",
    thresh=0.0001, secondThresh=0.0002)
sim_sig <- getSignificant(sim)
sim_top <- getTopHits(sim_sig, assay="lfsrs", mode="state")
sim_top <- runPairwiseSharing(sim_top)

plotPairwiseSharing(sim_top)
```

```
# Plot with complex column annotations
plotPairwiseSharing(sim_top, annotateColsBy = c("nSignificant", "multistateGroup"))
```

`plotQTLClusters` *Heatmap of QTL across states*

Description

Convenience method for plotting values from any assay specified by fillBy across states and tests.

Usage

```
plotQTLClusters(
  object,
  fillBy = "betas",
  FUN = identity,
  minSig = 1,
  annotateColsBy = NULL,
  annotateRowsBy = NULL,
  show_row_names = FALSE,
  state_id = "state_id",
  columnOrder = NULL,
  rowOrder = NULL,
  row_km = 0,
  column_km = 0,
  ...
)
```

Arguments

`object`	an `QTLExperiment` object.
`fillBy`	name of assay to use for main heatmap object.
`FUN`	Function to be applied to fillBy assay before plotting (e.g. identity, abs, log10).
`minSig`	minimum number of significant states for QTL to be included.
`annotateColsBy`	character or array of characters specifying the column(s) in colData to be plotted to annotate states.
`annotateRowsBy`	character or array of characters specifying the column(s) in rowData to be plotted to annotate QTL tests.
`show_row_names`	Logical to plot row (i.e. test) names.
`state_id`	colData column to use to label states.
`columnOrder`	Null for clustering or array to overwrite state order
`rowOrder`	Null for clustering or array to overwrite test order

row_km	Set k for k-means clustering of tests.
column_km	Set k for k-means clustering of states
...	Further arguments passed to Heatmap.

Value

Returns a ComplexHeatmap object.

Examples

```
sim <- qtleSimulate(
    nStates=10, nFeatures=100, nTests=1000,
    global=0.2, multi=0.4, unique=0.2, k=2)
sim <- callSignificance(sim, mode="simple", assay="lfsrs",
    thresh=0.0001, secondThresh=0.0002)

sim_sig <- getSignificant(sim)
sim_top <- getTopHits(sim_sig, assay="lfsrs", mode="state")
sim_top <- runTestMetrics(sim_top)
sim_top <- runPairwiseSharing(sim_top)
sim_top_ms <- subset(sim_top, qtl_type_simple == "multistate")

plotQTLClusters(sim_top)

# Plot with annotations for multi state group
plotQTLClusters(sim_top_ms, annotateColsBy = c("multistateGroup"),
    annotateRowsBy = c("qtl_type", "mean_beta", "QTL"))
```

---

plotSimulationParams *Distribution of estimated simulation parameters*

Description

Distribution of estimated simulation parameters

Usage

```
plotSimulationParams(params, n = 1e+05)
```

Arguments

params	Estimated simulation parameters from qtleEstimate.
n	Number of random values to sample for plots.

Value

A ggplot2 object

Examples

```
qtle <- mockQTLE()
params <- qtleEstimate(qtle, threshSig = 0.05, threshNull = 0.5)
plotSimulationParams(params=params)
```

`plotUpSet` *Upset plot of QTL sharing between states with state-level annotations*

Description

Convenient method to plot an UpSet plot showing the number of QTL that are significant in sets of states.

Usage

```
plotUpSet(
  object,
  assay = "significant",
  name = "colnames",
  minShared = 10,
  minDegree = 2,
  maxDegree = NULL,
  annotateColsBy = NULL,
  comb_order = "comb_size",
  set_order = order(ss),
  ...
)
```

Arguments

`object`	an `QTLExperiment` object
`assay`	Name of assay to use to assess significance.
`name`	character specifying the column in colData to use to label rows and columns. Default is colnames(qtle).
`minShared`	minimum number of shared QTL for set to be included.
`minDegree`	minimum degree of sharing for set to be included.
`maxDegree`	maximum degree of sharing for set to be included.
`annotateColsBy`	character or array of characters specifying the column(s) in colData to be plotted as annotations.
`comb_order`	characters specifying how sets should be ordered. Options include the set size (set_size), combination size (comb_size), degree (deg).
`set_order`	Array specifying order of states.
`...`	Further arguments passed to `UpSet`

Value

Returns a ComplexHeatmap object.

Examples

```
sim <- qtleSimulate(
    nStates=10, nFeatures=100, nTests=1000,
    global=0.2, multi=0.4, unique=0.2, k=2)
sim <- callSignificance(sim, mode="simple", assay="lfsrs",
    thresh=0.0001, secondThresh=0.0002)
sim_sig <- getSignificant(sim)
sim_top <- getTopHits(sim_sig, assay="lfsrs", mode="state")
sim_top <- runPairwiseSharing(sim_top)

plotUpSet(sim_top)

# Upset plot with complex row annotations
plotUpSet(sim_top, annotateColsBy = c("nSignificant", "multistateGroup"))
```

---

qtleEstimate *Estimate parameters from real data for simulating multi-state QTL summary statistics*

Description

Estimate parameters from real data for simulating multi-state QTL summary statistics

Usage

```
qtleEstimate(
  data,
  assay = "pvalues",
  threshSig = 0.001,
  threshNull = 0.1,
  verbose = TRUE
)

qtleParams2()
```

Arguments

data	A ‘QTLExperiment‘ object or named list containing "betas" and "errors" matrices.
assay	Assay containing test statistic information to use.
threshSig	Max threshold (pval/lfsr) for calling tests as significant.
threshNull	Min threshold (pval/lfsr) for calling tests as null.
verbose	Logical.

Details

The simulation consists of user defined number of equal numbers of four different types of effects: null, equal among conditions, present only in first condition, independent across conditions

Value

A list with parameter estimates for the QTLExperiment object.

Examples

```
qtle <- mockQTLE()
qtleEstimate(qtle)
```

`qtleParams` *Default qtle simulation parameters*

Description

Returns a list of the default values used for parameters when simulating multistateQTL data. Parameters include:

- betas.sig.shape
- betas.sig.rate
- cv.sig.shape
- cv.sig.rate
- betas.null.shape
- betas.null.rate
- cv.null.shape
- cv.null.rate

Usage

```
qtleParams()
```

Details

The default parameters returned by this function were generated using expression QTL (eQTL) summary statistics from the Genotype-Tissue Expression (GTEx) Project (Version 8) for the ten tissues with the largest sample sizes for eQTL mapping. The eQTL tests were filtered to include only eQTLs on chromosome 1 that were available in all 10 tissues.

`qtleParams()` returns the parameters used for the plots in the vignette. `qtleParams2()` returns the parameter values from the 'Orchestrating multi-state QTL analysis with R' manuscript and accompanying reproducible analysis workflow (see WEEO_2022_multistateQTL).

Value

A list with the default parameter values which can be used when simulating multistateQTL data.

Examples

```
qtleParams()
```

`qtleSimulate` *Create simulated multistateQTL data for testing purposes*

Description

Create simulated multistateQTL data for testing purposes

Usage

```
qtleSimulate(
  params = qtleParams(),
  nTests = 100,
  nFeatures = NULL,
  nStates = 5,
  global = 0.5,
  multi = 0,
  unique = 0,
  k = 2,
  betaSd = 0.1,
  lfsr = TRUE,
  verbose = TRUE
)
```

Arguments

`params`	list of parameters required to simulate betas and beta errors. Generated by 'qtleEstimate()' or 'qtleParams()'.
`nTests`	number of QTL tests
`nFeatures`	number of QTL features to simulate tests for, NULL mean nFeatures = nTests.
`nStates`	number of states
`global`	percent of QTL tests with significant effects shared across all states
`multi`	percent of QTL tests with significant effects shared across a subset of states.
`unique`	percent of QTL tests with significant effects in only one state
`k`	number of multi-state clusters or an array with the cluster assignments.
`betaSd`	The desired standard deviation or an array of standard deviations equal to the length of states for sampling beta values for each state.
`lfsr`	Logical to calculate lfsr using mashr_1by1.
`verbose`	Logical.

Details

The simulation consists of user defined number of equal numbers of four different types of effects: null, equal among conditions, present only in first condition, independent across conditions

Value

A simulated ‘QTLExperiment‘ object.

Examples

```
qtleSimulate(nTests=100, nStates=5, global=0.1, multi=0.2, unique=0.05)
```

---

replaceNAs *Return multistateQTL object with NAs filled in*

Description

A convenience function for imputing or filling in NAs in a ‘QTLExperiment‘ object.

Usage

```
replaceNAs(
  object,
  methods = list(betas = 0, errors = "mean", pvalues = 1, lfsrs = 1),
  verbose = FALSE
)
```

Arguments

object	A ‘QTLExperiment‘ object
methods	A named list with the method for each assay. Available methods are to replace with a given integer or with the row mean or median.
verbose	logical. Whether to print progress messages.

Value

A ‘QTLExperiment‘ object with the same dimensions as the original object, but with the NA values replaced according to the input specifications.

Examples

```
#' # Create a QTLExperiment object with NA values ------------------------------
qtle <- mockQTLE()

# Randomly remove 1000 elements from the betas matrix.
na_pattern <- sample(seq(1, ncol(qtle)*nrow(qtle)), 1000)
qtle_na <- qtle
assay(qtle_na, "betas")[na_pattern] <- NA

# There are some NA values in the "betas" assay
any(is.na(betas(qtle_na)))

qtle_complete <- replaceNAs(qtle_na)

# Now we don't have any NA values
any(is.na(betas(qtle_complete)))

## Specify a specific method to impute NAs ---------------------------------

qtle_median <- replaceNAs(
    qtle_na,
    methods=list(betas = 0, errors = "median", pvalues = 1),
    verbose=TRUE)
```

`runPairwiseSharing` *Compute the proportion of (significant) QTL shared by pairs of conditions*

Description

Compute the proportion of (significant) QTL shared by pairs of conditions

Usage

```
runPairwiseSharing(
  qtle,
  assay = "betas",
  assaySig = "significant",
  factor = 0.5,
  FUN = identity,
  ...
)
```

Arguments

`qtle`	A 'QTLExperiment' object.
`assay`	The assay containing the metric used to determine sharing (i.e. the metric to be within a factor X to be considered shared).

assaySig	The assay containing significance information.
factor	a number in [0,1] the factor within which effects are considered to be shared
FUN	a function to be applied to the estimated effect sizes before assessing sharing. Default 'FUN=identity', 'FUN=abs' ignores the sign of the effects when assessing sharing.
...	Additional parameters to pass on to internal functions.

Details

For each pair of states, the effects that are significant (as determined by 'callSignificance') in at least one of the two states are identified. Then the fraction of those with an estimated effect size (i.e. betas) within a factor 'factor' of one another is computed and returned.

Value

The 'QTLExperiment' object with a matrix called pairwiseSharing added to the metadata.

Examples

```
m <- mockQTLE()
m <- callSignificance(m, assay="pvalues")
runPairwiseSharing(m) # sharing by magnitude (same sign)
runPairwiseSharing(m, factor=0) # sharing by sign
runPairwiseSharing(m, FUN=abs) # sharing by magnitude when sign is ignored
```

---

runSignificantFeatures

Function to summarize QTL summary statistics by feature ID

Description

This function adds a summary of the features with significant QTL in each state to the metadata of the QTLExperiment object.

Usage

```
runSignificantFeatures(object, assay = "significant")
```

Arguments

object	A QTLExperiment object with multiple QTL tests (i.e., rows) for at least one feature.
assay	Assay containing T/F significance calls for each test.

Value

The ‘QTLExperiment‘ object with a summary of significant features in the metadata and a new column ‘nSignificantFeatures‘ in the colData.

Examples

```
qtle <- mockQTLE()

qtle <- callSignificance(qtle)

# There is an assay called 'significant'
assays(qtle)

# Obtain summary of significant features for each state
qtle <- runSignificantFeatures(qtle)

# There is a summary added to the metadata of the object
metadata(qtle)
```

---

runTestMetrics *Classify multi-state QTL*

Description

Takes the results from ‘callSignificance()‘ and from the assay ‘betas‘ to categorize each QTL test using two classification strategies:

Strategy 1 (qtl_type): (1) global-shared, (2) global-diverging, (3) multi-state-shared, (4) multi-state-diverging, or (5) unique.

Strategy 2 (qtl_type_simple): (1) global, (2) multi-state, or (3) unique.

Usage

```
runTestMetrics(
  qtle,
  assay = "betas",
  assaySig = "significant",
  globalBuffer = 0,
  ...
)
```

Arguments

qtle	QTLExperiment qtle
assay	Name of assay containing QTL effect size estimate (e.g. betas)
assaySig	Name of assay with TRUE/FALSE significance calls

globalBuffer	Number of states that can be not-significant and the QTL will still be called as global, for example, if globalBuffer=1, then a QTL will be considered global if if is significant in all or all but 1 state.
...	arguments passed to runTestMetrics

Details

If a test is significant in more than one sign across different states, returns TRUE in rowData(qtle)$diverging

Value

The 'QTLExperiment' object with the following columns added to the rowData: nSignificant, effect_sd, qtl_type, qtl_type_simple

Examples

```
m <- mockQTLE()
m <- callSignificance(m)
m <- runTestMetrics(m)
```

---

simPerformance *Performance metrics for multistateQTL simulations*

Description

Performance metrics for multistateQTL simulations

Usage

```
simPerformance(qtle, assay = "significant")
```

Arguments

qtle	A 'multistateQTL' object.
assay	Name of the 'multistateQTL' assay containing the significance calls

Value

description

Examples

```
sim <- qtleSimulate()
sim <- callSignificance(sim, assay="lfsrs", thresh=0.1)
simPerformance(sim)
```

Index

∗ **internal**
multistateQTL-package, 2

callSignificance, 3
callSignificance,QTLExperiment-method *(*callSignificance*)*, 3
callSignificance<- *(*callSignificance*)*, 3

getComplete, 4
getSignificant, 5
getTopHits, 6

Heatmap, *9*, *11*

multistateQTL *(*multistateQTL-package*)*, 2
multistateQTL-package, 2

plotCompareStates, 7
plotPairwiseSharing, 8
plotQTLClusters, 10
plotSimulationParams, 11
plotUpSet, 12

qtleEstimate, 13
qtleParams, 14
qtleParams2 *(*qtleEstimate*)*, 13
qtleSimulate, 15
QTLExperiment, *3*, *4*, *6*, *18*

replaceNAs, 16
runPairwiseSharing, 17
runSignificantFeatures, 18
runTestMetrics, 19

simPerformance, 20

UpSet, *12*

wilcox.test, *4*