

Package: loopQTL (via r-universe)

September 19, 2026

Title Anchor-Restricted Loop QTL Mapping from Hi-C

Version 0.99.0

Description Maps chromatin loop QTLs by testing whether genotype at a loop's anchors associates with the loop's contact strength across samples. Reads .hic files via strawr, builds an exact-coordinate consensus loop set, normalizes contact strength across samples via band-wise quantile normalization in the style of BNBC (implemented on top of preprocessCore), and runs anchor-restricted SNP-by-loop regressions with MatrixEQTL. Intra-chromosomal loops only; no FDR or lead-SNP selection (left to the user).

URL <https://github.com/rhielab/loopQTL>

BugReports <https://github.com/rhielab/loopQTL/issues>

License GPL-3 + file LICENSE

Encoding UTF-8

BiocType Software

biocViews Software, DNA3DStructure, Epigenetics, HiC, QuantitativeTraitLocus, SNP

Depends R (>= 4.4)

Imports methods, stats, utils, strawr, preprocessCore, MatrixEQTL, GenomicRanges, IRanges, InteractionSet, S4Vectors, VariantAnnotation, Rsamtools, SummarizedExperiment, BiocParallel, data.table, ggplot2, scales, rtracklayer, patchwork

Suggests BiocStyle, knitr, rmarkdown

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

RoxygenNote 7.3.3

Config/pak/sysreqs
make libbz2-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-09-19 01:09:43 UTC

RemoteUrl <https://github.com/BiocStaging/loopQTL>

RemoteRef HEAD

RemoteSha cc881e77e9259e5461fd949c1420c9e47216bcec

Contents

loopQTL-class	2
loopQTLannotate	4
loopQTLchooseK	6
loopQTLexampleData	7
loopQTLnormalize	7
loopQTLpairData	9
loopQTLplotLocus	10
loopQTLplotPair	11
loopQTLplotSnpWindow	12
loopQTLsetup	14
runQTL	16
Index	18

loopQTL-class	<i>Class to store loopQTL input data and analysis results</i>
---------------	---

Description

loopQTL objects should normally be generated by loopQTLsetup(). See its documentation for more information on the expected format of input data.

Usage

```
## S4 method for signature 'loopQTL'
samples(object)

## S4 method for signature 'loopQTL'
loops(object)

## S4 method for signature 'loopQTL'
cis(object)

## S4 method for signature 'loopQTL'
vcfPath(object)
```

```
## S4 method for signature 'loopQTL'
resolution(object)

## S4 method for signature 'loopQTL'
phenotype(object)

## S4 method for signature 'loopQTL'
results(object)

## S4 method for signature 'loopQTL'
metadata(object)

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

Arguments

object a loopQTL object

Details

All slots can also be accessed through an identically named function, i.e., to access `results(obj)`, use `loopQTL::results(obj)`.

Slots

samples List of unique sample IDs

loops Consensus intra-chromosomal loops as a [InteractionSet::GInteractions](#) object

cis Optional numeric vector of per-sample cis-contact totals. Must have names matching the IDs in **samples**, and be in the same order as **samples**.

vcfPath Path to the tabix-indexed VCF

resolution Analysis bin size (bp)

phenotype Matrix containing normalized numeric phenotype data for each loop (in the rows) and each sample (in the columns). Dimensions and order has not been run.

results QTL result table. Empty if `runQTL()` has not been run. Contains a row for each loop-SNP pair and columns `loopID`, `snpID`, `chrom`, `pos`, `ref`, `alt`, `anchor`, `beta`, `se`, `t`, `p`, `nUsed`, and `maf`. If `loopQTLannotate()` has been run, also contains `gene`, `peak` (optional), and `Type` columns.

metadata Function-specific metadata

See Also

`loopQTLsetup`

Examples

```
obj <- loopQTLExampleData()
obj
samples(obj)
loops(obj)
```

loopQTLannotate	<i>Annotate loopQTL results with target genes and regulatory peaks</i>
-----------------	--

Description

This function annotates SNP-loop QTL results with target genes, optional peak overlap, and a SNP-loop Type classification.

Usage

```
loopQTLannotate(
  obj,
  gtf,
  peak = NULL,
  TSSwindow = 2000,
  anchorWindow = 5000,
  peakAnnotation = "Peak",
  removeTSS = TRUE
)
```

Arguments

obj	A loopQTL object with results(obj) filled by runQTL().
gtf	Path to a GTF file with gene entries.
peak	Optional path to a peak bed file. If supplied, an extra peak column is added and the Type labels gain the "rs(<peakAnnotation>)-..." annotations.
TSSwindow	Base-pair window flank around each gene's TSS to define promoters. Default 2000.
anchorWindow	Base-pair window flank added around each anchor when looking for overlapping promoters and peaks. Default 5000.
peakAnnotation	Character label substituted into the Type values when peak is supplied. Default "Peak". (e.g "Enhancer" or "H3K27ac")
removeTSS	Logical; default TRUE. When TRUE, a TSS-based filtering on peaks will be done to get the distal peak elements.

Details

Adds three columns to `results(obj)`:

gene The name of the genes found at the target anchor. Empty if no gene is found.

peak TRUE if the SNP overlaps any peak from the supplied peak file, FALSE otherwise. When `removeTSS = TRUE`, peaks that overlap any TSS region are dropped before this overlap check (so promoter-like peaks don't count).

Type One of:

- "WithinTSS" - SNP overlaps any gene's TSS ($\pm$ TSSwindow). Short-circuits all other annotation when `removeTSS = TRUE`.
- "rs-gene" - SNP not in TSS, has a target gene at the partner anchor.
- "rs-<peakAnnotation>" - SNP not in TSS, no gene at partner, but partner anchor overlaps a peak (only when peak is supplied). e.g. "rs-Enhancer".
- "rs-0" - SNP not in TSS, no gene AND no peak at the partner anchor.
- "rs(<peakAnnotation>)-gene" - additionally overlaps a peak (only when peak is supplied), has a target gene. e.g. "rs(Enhancer)-gene".
- "rs(<peakAnnotation>)-<peakAnnotation>" - SNP overlaps peak, partner overlaps peak, no gene at partner. e.g. "rs(Enhancer)-Enhancer".
- "rs(<peakAnnotation>)-0" - overlaps a peak but no target gene and no peak at partner.

If both promoter and peaks are found at the target anchor, it gives priority on genes.

Value

The input `obj` with `results(obj)` augmented by `gene`, `peak` (when applicable), and `Type` columns.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of loopQTLsetup().
## Here we just use the example object.
obj <- loopQTLExampleData()
## get the path of example annotation files
extdataDir <- system.file("extdata", package = "loopQTL")
peakPath <- file.path(extdataDir, "example_annotation.bed")
gtfPath <- file.path(extdataDir, "example.gtf.gz")

## Annotate SNP-loop pairs with gene, peak overlap, and Type
obj <- loopQTLannotate(
  obj,
  gtf          = gtfPath,
  peak         = peakPath,
  peakAnnotation = "Enhancer",
  TSSwindow    = 2000,
  removeTSS    = TRUE)

## check the annotated results
head(results(obj))
```

loopQTLchooseK

Auto-select the number of PCs for covariates

Description

This function runs PCA on the normalized phenotype matrix and picks a number of principal components K to use as covariates in the downstream regression. This function returns a list with the chosen K and the sample $\times$ K matrix of PCs.

Usage

```
loopQTLchooseK(
  obj,
  Kmethod = c("elbow", "BE"),
  normal = TRUE,
  Kmax = NULL,
  ...
)
```

Arguments

<code>obj</code>	A loopQTL object with phenotype matrix filled by <code>loopQTLnormalize()</code> .
<code>Kmethod</code>	"elbow" (default) or "BE".
<code>normal</code>	Apply RINT per loop before PCA. Default TRUE.
<code>Kmax</code>	The maximum number of PCs returned. Default <code>length(samples(obj)) - 2</code> .
<code>...</code>	Forwarded to PCAForQTL's <code>runElbow</code> or <code>runBE</code> function.

Value

A list:

- `K` Chosen number of PCs (integer).
- `PCs` samples $\times$ K matrix of principal components.
- `method` Which K-selection method was used.

Examples

```
obj <- loopQTLExampleData()
Kres <- loopQTLchooseK(obj, Kmethod = "elbow")
Kres$K
```

loopQTLExampleData	<i>Load the bundled example loopQTL object</i>
--------------------	--

Description

Convenience helper for @examples blocks and vignette code that need a fully-populated loopQTL object without re-running the full pipeline. Loads `example_loopQTL_obj.rda` shipped in `inst/extdata/` and rewrites the file paths baked into the object so they resolve against the currently installed package location.

Usage

```
loopQTLExampleData()
```

Value

A [loopQTL](#) object with phenotype, results, and metadata populated.

Examples

```
obj <- loopQTLExampleData()
obj
```

loopQTLnormalize	<i>Cross-normalization on Hi-C signal</i>
------------------	---

Description

This function normalizes contact strength across samples so that per-loop phenotypes are directly comparable. For each chromosome it pulls the whole-genome matrix per sample, log-CPM adjusts, then does band-wise quantile normalization across samples per genomic-distance band. Normalized values at the consensus-loop bin-pairs are written to `phenotype(obj)`.

Usage

```
loopQTLnormalize(  
  obj,  
  BPPARAM = BiocParallel::SerialParam(),  
  saveWindows = FALSE,  
  windowsOut = NULL,  
  window = 1e+06,  
  snpWindow = 0  
)
```

Arguments

obj	A loopQTL object created from loopQTLsetup().
BPPARAM	A BiocParallel::BiocParallelParam for parallelization over chromosomes. Default BiocParallel::SerialParam() .
saveWindows	Logical. If TRUE, also save a local contact submatrix around each in-loop SNP for downstream functions usage.
windowsOut	Required when saveWindows = TRUE. Path to a .rds file the per-SNP windows are written to.
window	Minimum window extent in bp around each in-loop SNP to create .rds file when saveWindows = TRUE. Default 1000000.
snpWindow	Base-pair window flank added to each consensus anchor when selecting in-loop SNPs. Default 0.

Details

Please note that you need to set saveWindows to TRUE to additionally output a per-SNP contact-window matrix as .rds file. This file is needed by the loopQTLplotSnpWindow() and loopQTLplotLocus() downstream functions. saveWindows does not change phenotype(obj) or affect the runQTL() step.

Value

The loopQTL obj with phenotype(obj) filled.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of loopQTLsetup().
## Here we just use the example object.
obj <- loopQTLexampleData()
## Example code to run loopQTLnormalize(). By default it will NOT create
## the .rds contact submatrix file that downstream functions need.
obj <- loopQTLnormalize(
  obj,
  BPPARAM      = BiocParallel::MulticoreParam(workers = 2),
  snpWindow    = 10000)

## Example code to run loopQTLnormalize() and also create the `.rds`
## contact submatrix file needed by downstream visualization functions.

## First, get a temporary directory in which to save the output file.
tempDirectory <- tempdir()
obj <- loopQTLnormalize(
  obj,
  BPPARAM      = BiocParallel::MulticoreParam(workers = 2),
  saveWindows  = TRUE,
  windowsOut   = file.path(tempDirectory, "loop_snp_windows.rds"),
  window       = 2000000,
  snpWindow    = 10000)
```

loopQTLpairData	<i>Sample-level view of one loop $\times$ SNP pair</i>
-----------------	---

Description

This function returns the per-sample phenotype and genotype (dosage 0/1/2) for a single SNP-loop pair. Useful for inspecting a specific hit or exporting sample-level values for reporting.

Usage

```
loopQTLpairData(obj, loopID, snpID, normal = TRUE)
```

Arguments

obj	A loopQTL object with results(obj) filled.
loopID	Character vector of loopIDs to extract.
snpID	Character vector of snpIDs to extract.
normal	Apply RINT per loop before returning phenotype values. default TRUE.

Value

data.frame with columns sample, loopID, snpID, genotype (0 - Reference homozygous; 1 - heterozygous, 2 - Alternative homozygous), phenotype, cis.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of runQTL().
##Here we just use the example object.
obj <- loopQTLExampleData()
##Check results so that you can choose your interested loopID and snpID
head(results(obj))
##Then you can specify the specific SNP-loop pair data you want to check at samples level.
pair <- loopQTLpairData(
  obj,
  loopID = "L0000187_chr1_46030000_46220000",
  snpID = "rs11584814")
head(pair)
```

loopQTLplotLocus	<i>Locus-level track view of loops, SNPs, and genes</i>
------------------	---

Description

This function renders a locus-scale ggplot combining the consensus loops, the tested SNP, and (when a GTF is supplied) gene annotations, over a user-specified region. Please note that to use `values = "cpm"` or `values = "raw"` to show per-loop contact strength, you need to set `saveWindows = TRUE` to save the per-SNP contact-window matrices from `loopQTLnormalize()`.

Usage

```
loopQTLplotLocus(
  obj,
  snpID,
  chrom,
  start,
  end,
  gtf,
  pThreshold = 0.05,
  values = c("normalized", "cpm", "raw"),
  windows = NULL,
  strongColor = "#FF7917",
  weakColor = "grey90",
  limit = NULL,
  geneBiotypes = "protein_coding",
  axisUnit = c("Mb", "bp")
)
```

Arguments

<code>obj</code>	A loopQTL object with <code>results(obj)</code> filled by <code>runQTL()</code> .
<code>snpID</code>	Single snpID present in <code>results(obj)</code> .
<code>chrom</code>	Chromosome. e.g. "chr1".
<code>start</code>	The start genomic coordinate in bp for the x-axis.
<code>end</code>	The end genomic coordinate in bp for the x-axis.
<code>gtf</code>	Path to a GTF file.
<code>pThreshold</code>	Nominal p-value cutoff for "significant" loops. Default 0.05.
<code>values</code>	Which contact-strength matrix to color arcs by: "normalized" Default. The matrix from <code>phenotype(obj)</code> . Matches what <code>runQTL()</code> tested. "cpm" CPM matrix. Requires <code>saveWindows</code> output. "raw" raw count matrix. Requires <code>saveWindows</code> .

windows	Optional path/loaded list from loopQTLnormalize(saveWindows = TRUE). Required when values is "cpm" or "raw".
strongColor	Color for the strongest signal on the arc gradient. Default "#FF7917" (orange).
weakColor	Color for the weakest signal. Default "grey90".
limit	Optional numeric cap on the color scale (values above are limited to strongColor).
geneBiotypes	Character vector of gene_biotype values to keep from the GTF. Default "protein_coding"; pass NULL to keep all.
axisUnit	"Mb" (default) or "bp" for x-axis tick labels.

Value

A ggplot (gene / SNP / arc panels stacked) object. Save with ggplot2::ggsave().

Examples

```
## Load the example object. In real analysis, the input
## should be the output of runQTL().
## If you specify `values = "cpm"` or `values = "raw"`, the object must
## carry the contact matrix `.rds` (see `obj@metadata$windowsOut`);
## otherwise the normalized matrix stored in the object is used.
## Here we just use the example object.
obj <- loopQTLexampleData()
## You also need a gtf file input to give the genomic annotations. Here we provide an example gtf.
gtfPath <- system.file("extdata", "example.gtf.gz", package = "loopQTL")
## Return is a ggplot object so you can ggsave() to output.
p <- loopQTLplotLocus(
  obj,
  chrom = "chr1",
  start = 45900000,
  end = 46300000,
  snpID = "rs11584814",
  values = "cpm",
  gtf = gtfPath)
```

loopQTLplotPair

Per-sample boxplot for a single SNP-loop pair

Description

This function renders the per-sample boxplot for a chosen result SNP-loop pair. It is returned as a ggplot object that can be saved with ggplot2::ggsave().

Usage

```
loopQTLplotPair(obj, loopID, snpID, normal = TRUE, jitter = TRUE)
```

Arguments

obj	A loopQTL object with results(obj) filled by runQTL().
loopID	A single loopID to plot.
snpID	A single snpID to plot.
normal	Plot Rank-based Inverse Normal Transformed (RINT'd) phenotype values. Defaults to TRUE. Set to FALSE to plot pre-RINT normalized values.
jitter	Add a jittered point overlay (default TRUE).

Value

A ggplot object.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of runQTL().
##Here we just use the example object.
obj <- loopQTLExampleData()
##Check results so that you can choose your interested loopID and snpID
head(results(obj))
##Then you can specify the specifi SNP-loop pair data you want to plot as classical QTL box plot.
##Return is a ggplot object so you can save with ggsave().
p <- loopQTLplotPair(
  obj,
  loopID = "L0000187_chr1_46030000_46220000",
  snpID = "rs11584814")
```

loopQTLplotSnpWindow *Hi-C-style beta heatmap around a SNP*

Description

This function renders a Hi-C-style heatmap centered on a chosen SNP: for every bin-pair in the SNP's cached contact window (from loopQTLnormalize(saveWindows = TRUE)), the color encodes the per-sample regression coefficient (beta) of the bin-pair's contact strength on the SNP's dosage. Set normal = TRUE and covariates = "auto" to match the model used in runQTL().

Usage

```
loopQTLplotSnpWindow(
  obj,
  snpID,
  chrom = NULL,
  start = NULL,
  end = NULL,
  windows = NULL,
  mirror = TRUE,
```

```

minSamplesPerCell = 3,
normal = FALSE,
covariates = NULL,
Kmethod = c("elbow", "BE"),
Kmax = NULL,
limit = NULL,
palette = "RdBu",
metric = c("betaLogp", "beta"),
values = c("normalized", "cpm", "raw"),
axisUnit = c("Mb", "bp")
)

```

Arguments

obj	A loopQTL object with metadata\$windowsOut filled by loopQTLnormalize(saveWindows = TRUE).
snpID	A single snpID present in the saved windows file.
chrom	Optional chrom for verification. If supplied, must match the SNP's chrom in the windows file.
start	Optional Start genomic position in bp to crop the heatmap to.
end	Optional End genomic position in bp to crop the heatmap to.
windows	Either NULL (default, loads from metadata(obj)\$windowsOut), a path to a windows .rds, or the loaded list.
mirror	Logical; mirror the upper-triangle beta values to the lower triangle so the heatmap is symmetric. Default TRUE.
minSamplesPerCell	Minimum usable (non-NA dose + complete covariates) samples required to fit the regression for a given cell. Cells with fewer samples are returned NA. Default 3.
normal	Apply Rank-based Inverse Normal Transformation (RINT) per cell across samples before fitting. Matches the normal = TRUE default in runQTL() . Default FALSE.
covariates	Covariate handling for the regression: NULL Default. Fit cell ~ dosage per cell (no covariates). "auto" Auto-pick K phenotype PCs via PCAForQTL on phenotype(obj) (RINT'd if normal = TRUE), then fit cell ~ dosage + PCs per cell. Matches what runQTL() tested, so beta here equals the beta column in results(obj). matrix/data.frame User-supplied samples x k covariates; partial regression as above. Rownames or matched order required.
Kmethod	"elbow" (default) or "BE". Passed through to loopQTLchooseK() when covariates = "auto". Ignored otherwise.
Kmax	Hard cap on the auto-chosen K when covariates = "auto". Defaults to the number of samples minus 2 (same as runQTL()). Pass a smaller value (e.g. floor(nSamples / 4)) for a more conservative covariate set.

limit	Symmetric color-scale limit on beta. Values beyond +/-limit are limited to the deepest color. Default NULL) in the cropped view.
palette	RColorBrewer palette name for the color pattern in the heatmap. Default "RdBu".
metric	What to render per cell: "betaLogp" Default. Use raw * -log10(p) values to plot, this consider both beta and significane of the regression analysis. "beta" Use raw beta values to plot.
values	Which cell values to run the regression on: "normalized" Default. Use the phenotype(obj) to perform the regression "cpm" Use counts per million matrix to perform the regression. "raw" Use raw straw counts to perform the regression
axisUnit	Unit for the graph axes - "Mb" (default) or "bp".

Value

A ggplot object.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of runQTL().
##Also please note that this function requires the object to have the contact matrix `rda` file.
##Check `obj@windowsOut` to see if your object contains this file.
##Here we just use the example object.
obj <- loopQTLExampleData()
##Check the results to see if you have any interested SNPs that you want to plot on.
head(results(obj))
p <- loopQTLplotSnpWindow(
  obj,
  chrom      = "chr1",
  start      = 45900000,
  end        = 46300000,
  snpID      = "rs11584814",
  normal     = TRUE,
  covariates = "auto")
```

loopQTLsetup

Validate inputs and build loopQTL object

Description

- validates the input files header, including .vcf and .hic files;
- reads each per-sample BEDPE and builds a consensus loop set by exact-coordinate matching. Loops found fewer than minSamples samples will be dropped;
- Automatically computes per-sample cis-contact depth from the .hic files when the samplesheet lacks a cis column.

Usage

```
loopQTLsetup(
  samples,
  vcf,
  resolution,
  minSamples = 2,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

<code>samples</code>	a data.frame with columns <code>Sample</code> , <code>hic</code> , <code>loops</code> . Optional <code>cis</code> column. If the <code>cis</code> column is absent, it is calculated from the <code>.hic</code> files automatically.
<code>vcf</code>	Path to a VCF. The vcf should be bgzipped with tabix index.
<code>resolution</code>	a numeric bin size recognized in the <code>.hic</code> file (e.g. 10000).
<code>minSamples</code>	The minimum number of samples supporting a consensus loop. Default is 2 samples.
<code>BPPARAM</code>	A BiocParallel::BiocParallelParam that parallelizes the cis-computing straw pulls.

Details

The samplesheet is a tab-separated file with columns `Sample`, `hic`, `loops`, and optionally `cis`.

Value

A [loopQTL](#) object with `samples`, `loops`, `vcfPath`, `resolution`, `cis`, and `metadata` populated. `phenotype` is an empty placeholder until `loopQTLnormalize()` fills it.

Examples

```
## Get the necessary input example files path.
extdataDir <- system.file("extdata", package = "loopQTL")
vcfPath <- file.path(extdataDir, "example.vcf.gz")
samplesheetPath <- file.path(extdataDir, "example_samplesheet.tsv")
samples <- read.table(samplesheetPath, header = TRUE)

##modify the relative file path
samples$hic <- unlist(
  lapply(samples$hic, function(x) {
    file.path(extdataDir, x)
  })
)
samples$loops <- unlist(
  lapply(samples$loops, function(x) {
    file.path(extdataDir, x)
  })
)

## Run loopQTLsetup
obj <- loopQTLsetup(
  samples = samples,
```

```
vcf          = vcfPath,
resolution = 10000,
minSamples = 2)
```

runQTL

Anchor-restricted association testing

Description

This function first reads all anchor-window SNPs from the VCF, and runs MatrixEQTL twice - once treating anchor A as the gene position, once anchor B. The two result tables are outer-joined on SNP-loop level. Covariates are supplied via the `covariates` argument - typically the phenotype PCs chosen in the previous step. By default, rank-based inverse normal transformation (RINT) is applied per loop before regression.

Usage

```
runQTL(
  obj,
  window = 0,
  covariates = NULL,
  normal = TRUE,
  Kmethod = c("elbow", "BE"),
  Kmax = NULL,
  pvOutputThreshold = 1
)
```

Arguments

<code>obj</code>	A loopQTL object on which <code>loopQTLnormalize()</code> has been run.
<code>window</code>	Base-pair window flank added to each anchor bin when matching SNPs. Default 0.
<code>covariates</code>	Optional <code>samples x k</code> matrix/data.frame. If NULL, will be auto-chosen via <code>loopQTLchooseK()</code> .
<code>normal</code>	Apply Rank-based Inverse Normal Transformation (RINT) per loop before regression. Default TRUE.
<code>Kmethod</code>	"elbow" or "BE"; passed to <code>loopQTLchooseK()</code> when <code>covariates</code> is NULL. Default "elbow".
<code>Kmax</code>	The maximum number of PCs used. Defaults to the number of samples minus two.
<code>pvOutputThreshold</code>	MatrixEQTL p-value threshold. Default 1 (return every pair).

Value

The loopQTL object with `results(obj)` filled.

Examples

```
## Load the example object. In real analysis, the input
## should be the output of loopQTLnormalize().
## Here we just use the example object.
obj <- loopQTLexampleData()
## Run the function, please note that the covariates option is not specified.
obj <- runQTL(
  obj,
  window      = 10000)

## In real analysis we recommend running runQTL() with covariates
## specified for more accurate regression, typically the top-K PCs.
## The helper loopQTLchooseK() picks K following the PCAForQTL method
## (https://github.com/heatherjzhou/PCAForQTL).
Kres <- loopQTLchooseK(obj)
obj <- runQTL(
  obj,
  window      = 10000,
  covariates = Kres$PCs)
```

Index

BiocParallel::BiocParallelParam, [8](#), [15](#)
BiocParallel::SerialParam(), [8](#)

cis (loopQTL-class), [2](#)
cis, loopQTL-method (loopQTL-class), [2](#)

InteractionSet::GInteractions, [3](#)

loopQTL, [7](#), [15](#)
loopQTL-class, [2](#)
loopQTLannotate, [4](#)
loopQTLchooseK, [6](#)
loopQTLchooseK(), [13](#)
loopQTLexampleData, [7](#)
loopQTLnormalize, [7](#)
loopQTLnormalize(), [10](#)
loopQTLpairData, [9](#)
loopQTLplotLocus, [10](#)
loopQTLplotPair, [11](#)
loopQTLplotSnpWindow, [12](#)
loopQTLsetup, [14](#)
loops (loopQTL-class), [2](#)
loops, loopQTL-method (loopQTL-class), [2](#)

metadata (loopQTL-class), [2](#)
metadata, loopQTL-method
(loopQTL-class), [2](#)

phenotype (loopQTL-class), [2](#)
phenotype, loopQTL-method
(loopQTL-class), [2](#)

resolution (loopQTL-class), [2](#)
resolution, loopQTL-method
(loopQTL-class), [2](#)
results (loopQTL-class), [2](#)
results, loopQTL-method (loopQTL-class),
[2](#)
runQTL, [16](#)
runQTL(), [13](#)

samples (loopQTL-class), [2](#)
samples, loopQTL-method (loopQTL-class),
[2](#)
show, loopQTL-method (loopQTL-class), [2](#)
vcfPath (loopQTL-class), [2](#)
vcfPath, loopQTL-method (loopQTL-class),
[2](#)