Package 'RLSeq'

October 9, 2022

Type Package

Title RLSeq: An analysis package for R-loop mapping data

Version 1.2.0

Description RLSeq is a toolkit for analyzing and evaluating R-loop mapping datasets. RLSeq serves two primary purposes: (1) to facilitate the evaluation of dataset quality, and (2) to enable R-loop analysis in the context of publicly-available data sets from RLBase. The package is intended to provide a simple pipeline, called with the `RLSeq()` function, which performs all main analyses. Individual functions are also accessible and provide custom analysis capabilities. Finally an HTML report is generated with `report()`.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Suggests AnnotationDbi, BiocStyle, covr, lintr, rcmdcheck, DT, httr, jsonlite, kableExtra, kernlab, knitr, magick, MASS, org.Hs.eg.db, R.utils, randomForest, readr, rmarkdown, rpart, testthat (>= 3.0.0), tibble, tidyr, TxDb.Hsapiens.UCSC.hg19.knownGene, futile.logger

Config/testthat/edition 3

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, RColorBrewer, grid, regioneR, valr, caretEnsemble, GenomicFeatures, rtracklayer, GenomicRanges, GenomeInfoDb, ComplexHeatmap, AnnotationHub, VennDiagram, callr, circlize, ggplotify, ggprism, methods, stats, RLHub, aws.s3, pheatmap

VignetteBuilder knitr

biocViews Sequencing, Coverage, Epigenetics, Transcriptomics, Classification

URL https://github.com/Bishop-Laboratory/RLSeq, https://bishop-laboratory.github.io/RLSeq/

BugReports https://github.com/Bishop-Laboratory/RLSeq/issues

BiocType Software

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

git_branch RELEASE_3_15

git_last_commit 7afa0ea

git_last_commit_date 2022-04-26

Date/Publication 2022-10-09

Author Henry Miller [aut, cre, cph] (<https://orcid.org/0000-0003-3756-3918>),
Daniel Montemayor [ctb] (<https://orcid.org/0000-0001-8702-5646>),
Simon Levy [ctb] (<https://orcid.org/0000-0002-4623-5716>),
Anna Vines [ctb] (<https://orcid.org/0000-0002-5149-7737>),
Alexander Bishop [ths, cph] (<https://orcid.org/0000-0002-5742-4387>)

Maintainer Henry Miller <millerh1@uthscsa.edu>

R topics documented:

`analyzeRLFS` *Analyze RLFS*

Description

Analyzes the enrichment of ranges within R-loop forming sequences (RLFS). See *details*.

Usage

```
analyzeRLFS(
  object,
  mask = NULL,
  quiet = FALSE,
  useMask = TRUE,
  noZ = FALSE,
  ntimes = 100,
  stepsize = 50,
  ...
)
```

Arguments

`object`	An RLRanges object.
`mask`	GRanges object containing masked genomic ranges. Not needed unless masked genome unavailable (see genomeMasks). Custom masks can be generated using regioneR::getMask.
`quiet`	If TRUE, messages are suppressed. Default: FALSE.
`useMask`	If FALSE, masked genome is not used. This is not recommended unless a mask is unavailable as it can lead to spurious results. Default: TRUE.
`noZ`	If TRUE, Z-score distribution is not calculated. Default: FALSE.
`ntimes`	Number of permutations to perform (default: 100).
`stepsize`	The step size for calculating the Z score distribution. Default: 50. See also regioneR::localZScore.
`...`	Arguments passed to regioneR::permTest.

Details

R-loop forming sequences are regions of the genome with sequences that are favorable for R-loop formation. They are computationally predicted with the QmRLFS-finder software program and serve as a data-independent test of whether a sample has mapped R-loops robustly or not.

Method:

Permutation testing is implemented via regioneR::permTest such that, for each permutation, R-loop peaks were randomized using regioneR::circularRandomizeRegions and then the number of overlaps with RLFS are counted. 100 permutations are used by default to build an empirical

distribution for peak/RLFS overlap. Then the true number of overlaps from non-randomized peaks and RLFS are compared to the null distribution to calculate Z-score and significance of enrichment. Finally, a Z-score distribution was calculated (using regioneR::localZScore) 5kb upstream and downstream of the average RLFS midpoint.

These results are subsequently used in the binary classification of the sample as "POS" (maps R-loops) or "NEG" (does not map R-loops). See also predictCondition.

Value

An RLRanges object with RLFS analysis results accessible via `RLSeq::rlresult(object, "rlfsRes")`. Contains the following structure:

- `perTestResults`
  - An object of the class `permTestResultsList` from `regioneR` with the results of permutation testing. See also regioneR::permTest for full description.
- `Z-scores`
  - An object of the class `localZScoreResultsList` from `regioneR`. Contains the results of local Z-score analysis +/-5kb around each RLFS. See also regioneR::localZScore.

Examples

```
# Example dataset
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Perform RLFS analysis (remove ntimes=2 and noZ=TRUE for a typical analysis)
rlr <- analyzeRLFS(rlr, ntimes = 2, noZ = TRUE)
```

---

auxdata *Auxiliary Data*

Description

A list containing data used by RLSeq functions. It can also be useful for checking the available modes and genomes in RLSeq. See also the `data-raw/auxdata.R` script that was used to create it.

Usage

```
auxdata
```

Format

An object of class `list` of length 10.

Details

Structure:
A named list containing the following entries:

- `db_cols`
  - A `tbl` with colors associated with each database in RLHub useful for plotting. See also RLHub::annotations.
- `annotypes`
  - A `tbl` containing the annotation databases and annotation types available from RLBase. See also RLHub::annotations.
- `ip_cols`
  - A `tbl` containing the colors associated with each "Immunoprecipitation type" (ip_type) in RLBase. See also RLHub::rlbase_samples.
- `mode_cols`
  - A `tbl` containing the colors associated with each R-loop mapping mode in RLHub::rlbase_samples.
- `heat_cols`
  - A `tbl` containing the colors associated with user-supplied data and RLBase data when running corrHeatmap.
- `label_cols`
  - A `tbl` containing the colors associated with the labels in RLBase. See also RLHub::rlbase_samples.
- `prediction_cols`
  - A `tbl` containing the colors associated with the predictions in RLBase. See also RLHub::rlbase_samples.
- `available_modes`
  - A `tbl` containing the modes available in RLBase and associated metadata. See also RLHub::rlbase_samples.
- `available_genomes`
  - A `character` showing all the official UCSC genomes available for use with RLSeq. See also available_genomes.
- `misc_modes`
  - A `character` showing the R-loop mapping modes that are lumped into the 'misc' category for simplification of plotting.

Examples

```
auxdata
```

`available_genomes` *Available Genomes*

Description

Contains metadata about all the genomes available in UCSC. It contains derived metadata, such as the effective genome sizes as well. See also the `data-raw/available_genomes.R` script to see processing steps.

Usage

```
available_genomes
```

Format

An object of class `data.frame` with 199 rows and 27 columns.

Details

Structure:

`available_genomes` is a `data.frame` with the following columns:

- `UCSC_orgID`
  - Official UCSC ID of the genome
- `description`
  - Verbose description of the assembly, source, and year/month of entry.
- `nibPath`
  - Endpoint of the genome in UCSC gbdb.
- `organism`
  - Name of the organism.
- `defaultPos`
  - Default location of genome browser view for this genome.
- `active`
  - Description not available.
- `orderKey`
  - Description not available.
- `genome`
  - The name of the genome.
- `scientificName`
  - The scientific name of the organism.
- `htmlPath`
  - Path in UCSC gbdb to the `description.html` file for the genome.
- `hgNearOk`
  - Description not available.
- `hgPbOk`
  - Description not available.
- `sourceName`
  - Name of organization providing the genome.
- `taxId`
  - The taxonomy ID of the organism.
- `genes_available`
  - If `TRUE`, the gene annotations are available in GTF format.
- `year`
  - The year the genome assembly was added.

- `eff_genome_size_XXbp`
  - The effective genome size of this genome. Calculated at various read lengths with khmer and used to improve the accuracy of analysis. See the `data-raw/available_genomes.R` script to see how this calculation was performed.
- `genome_length`
  - The total length of the genome.
- `rlfs_available`
  - If `TRUE`, R-loop forming sequences annotations are available in the RLBase AWS S3 repository.

Examples

```
available_genomes
```

`checkRLFSAnno` *Check RLFS*

Description

Helper function that checks whether a genome has RLFS available.

Usage

```
checkRLFSAnno(genome)
```

Arguments

`genome`	the UCSC genome name to check

Value

A logical, TRUE if available, FALSE if not

`corrAnalyze` *Analyze Correlations*

Description

Finds the pairwise correlation in signal around gold-standard R-Loop sites between the query sample and the coverage tracks in the RLBase database. See *details*.

Usage

```
corrAnalyze(object, force = FALSE)
```

Arguments

`object`	An RLRanges object.
`force`	Force `corrAnalyze` to run, even if on Windows. Default: FALSE.

Details

Currently, this does not work on windows.

Method:

The `corrAnalyze` function performs a correlation test that can be used to assess sample-sample similarity by calculating coverage signal (from genomic alignments) around "gold standard" R-loop sites (PMID: 33411340). The resulting correlation matrix is useful for determining how well a supplied sample correlates with previously-published datasets.

During the RLBase-data workflow, the signal for each R-loop mapping sample within "gold standard" R-loop sites was calculated see RLHub::gs_signal.

The `corrAnalyze` function loads RLHub::gs_signal and accepts an RLRanges object with a valid coverage slot. It then does the following:

1. The coverage is quantified within the "gold standard" sites and added as a column to the signal matrix from RLHub::gs_signal.
2. Then, the stats::cor function is used to calculate the Pearson correlation pairwise between all samples, yielding a correlation matrix
3. Finally, the correlation matrix is stashed in the in the `correlationMat` slot of the RLResults and returned.

Value

An RLRanges object with correlation results included as a `matrix`. The correlation matrix is accessed via `rlresults(object, "correlationMat")`.

Examples

```
# Example RLRanges object
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# corrAnalyze does not work on Windows OS
if (.Platform$OS.type != "windows") {

    # run corrAnalyze
    rlr <- corrAnalyze(rlr)
}
```

corrHeatmap *Plot Correlation Results*

Description

Plots a heatmap to visualize the pairwise Pearson correlation matrix generated via corrAnalyze.

Usage

```
corrHeatmap(object, returnData = FALSE, complex = TRUE, ...)
```

Arguments

object	An RLRanges with corrAnalyze already run.
returnData	If TRUE, plot data is returned instead of plotting. Default: FALSE
complex	If TRUE, ComplexHeatmap::Heatmap will be used for plotting. Otherwise, pheatmap::pheatmap is used. Default: TRUE
...	For internal use.

Value

A plot object or plotting data (if returnData is TRUE).

Examples

```
# Example RLRanges data with corrAnalyze() already run.
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Corr heatmap
corrHeatmap(rlr)
```

featureEnrich *Test Genomic Feature Enrichment*

Description

Tests the enrichment of genomic features in supplied peaks. See *details*.

Usage

```
featureEnrich(
  object,
  annotype = c("primary", "full"),
  annotations = NULL,
  downsample = 10000,
  quiet = FALSE
)
```

Arguments

`object`	An RLRanges object.
`annotype`	The type of annotations to use. Can be one of "primary" or "full". Default: "primary". See RLHub::annotations for greater detail.
`annotations`	A custom annotation list of the same structure described in RLHub::annotations.
`downsample`	If a numeric, data will be down sampled to the requested number of peaks. This improves the speed of genomic shuffling and helps prevent p-value inflation. If FALSE, then downsampling will not be performed. Default: 10000.
`quiet`	If TRUE, messages will be suppressed. Default: FALSE

Details

Method:

Annotations relevant to R-loops were curated as part of the RLBase-data workflow and are provided via RLHub::annotations.

In `featureEnrich`, each annotation "type" (e.g., "Exons", "Introns", etc) is compared to the supplied RLRanges, yielding enrichment statistics with the following procedure:

1. For each annotation type, the peaks are overlapped with the annotations.
2. Then, valr::bed_reldist is used to find the relative distance distribution between the peaks and the annotations for both the supplied RLRanges and shuffled RLRanges (via valr::bed_shuffle). Significance of the relative distance is calculated via stats::ks.test.
3. Then, Fisher's exact test is implemented via valr::bed_fisher to obtain the significance of the overlap and the odds ratio.

Value

An RLRanges object containing the results of the enrichment test accessed via `rlresult(object, "featureEnrichment")`. The results are in `tbl` format. For a full description of all columns in the output table see RLHub::feat_enrich_samples.

Examples

```
# Example RLRanges dataset
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# RL Region Test
featureEnrich(rlr)

# With custom annotations
small_anno <- list(
    "Centromeres" = readr::read_csv(
        system.file("extdata", "Centromeres.csv.gz", package="RLSeq"),
        show_col_types = FALSE
    )
)
featureEnrich(rlr, annotations = small_anno)
```

feature_ggplot *Feature ggplot*

Description

The core plotting component of `plotEnrichment`

Usage

```
feature_ggplot(x, usamp, limits, splitby)
```

Arguments

`x`	A `tbl` containing data for plotting.
`usamp`	The name of the user-supplied sample
`limits`	Specify limits on data. Useful for controlling infinite estimation of odds ratio resulting from fisher's exact test. To remove limits, set c(-Inf, Inf). Default: c(-10, 15).
`splitby`	Metadata by which to split plots. Can be "none", "prediction", or "label".

Value

A ggplot2 object.

geneAnnotation *Annotate R-Loops with Genes*

Description

Annotates RLRanges with entrez ids for overlapping genes. See *details*.

Usage

```
geneAnnotation(object, txdb = NULL)
```

Arguments

`object`	An RLRanges object.
`txdb`	The TxDb or EnsDb object containing gene annotations. If not supplied, annotations will be automatically downloaded from AnnotationHub. See also GenomicFeatures::TxDb.

Details

The geneAnnotation function provides a simple procedure for annotating RLRanges with gene IDs by overlap.

Annotations:

First. gene annotations are automatically downloaded using AnnotationHub::query with the following pattern:

```
AnnotationHub::query(
    x = ah,
    pattern = c("TxDb", "UCSC", "knownGene", genome)
)
```

Where genome is the UCSC genome id for the RLRanges object. If these annotations are unavailable, they should be provded using the txdb parameter. See also GenomicFeatures::TxDb.

Overlaps:

The annotations are subsequently overlapped with the ranges in the supplied RLRanges object using valr::bed_intersect and saved in the RLResults object as a tbl with a mapping of peak names to gene_id (entrez gene IDs).

Value

An RLRanges object with gene overlaps included. The results are available via rlresult(object, "geneAnnoRes"). The result object is a tbl with a mapping of peak_name (peak names from names(object)) to gene_id (entrez gene IDs).

Examples

```
# Example RLRanges data
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Perform gene annotation
rlr <- geneAnnotation(rlr)

# Supply custom TxDb if needed
if (GenomeInfoDb::genome(rlr)[1] == "hg19") {
    library(TxDb.Hsapiens.UCSC.hg19.knownGene)
    rlr <- geneAnnotation(rlr, txdb = TxDb.Hsapiens.UCSC.hg19.knownGene)
}
```

---

genomeMasks *Genome Masks*

Description

A collection of genome masks for use with analyzeRLFS. See the data-raw/genome_masks.R script for the processing steps.

Usage

```
genomeMasks
```

Format

An object of class `list` of length 8.

Details

Structure:

`genomeMasks` is a named list of `GRanges` objects. Each entry in the list follows the naming convention: `<genome>.masked`, where `<genome>` is an official UCSC genome ID. Each entry contains a `GRanges` object with the masked ranges from `<genome>`. The genomes provided correspond to the masked genomes available in BSgenome::available.genomes.

Examples

```
genomeMasks
```

---

getChromSizes *Get Chrom Sizes*

Description

Helper function which extracts chrom sizes from an RLRanges object.

Usage

```
getChromSizes(object)
```

Arguments

`object`	An RLRanges object.

Value

A tibble containing chrom sizes

getGSSignal *Get GS Signal*

Description

Extract signal around "gold-standard" R-loop sites

Usage

```
getGSSignal(coverage, gssignal)
```

Arguments

coverage	The path to a .bigWig file (can be a URL)
gssignal	The GS signal obtained from RLHub.

Value

A named list containing the results of correlation analysis.

getRLFSAnno *Get RLFS*

Description

Helper function that retrieves R-loop-forming sequences as GRanges

Usage

```
getRLFSAnno(object)
```

Arguments

object	An RLRanges object.

Value

A GRanges object with RLFS for that species.

peak_stats *Build peak statistics tibble*

Description

A helper function for building the peak statistics tibble

Usage

```
peak_stats(x, xshuff, y, chromSizeTbl, quiet = FALSE)
```

Arguments

x	The R-loop peaks to test.
xshuff	x, but shuffled around the genome to build a control peakset.
y	The annotations against which to test x.
chromSizeTbl	A tibble containing the sizes of each chromosome in x and y.
quiet	If TRUE, messages will be suppressed. Default: FALSE

Value

A tibble containing the test results.

plotEnrichment *Plot Enrichment Test Results*

Description

Creates a list of plots, one for each annotation database (see RLHub::annotations). These plots show the feature enrichment for the user-supplied sample in comparison to the samples in RLBase. This will only work if you did not use custom annotations with featureEnrich.

Usage

```
plotEnrichment(
  object,
  pred_POS_only = TRUE,
  label_POS_only = FALSE,
  splitby = c("none", "prediction", "label"),
  limits = c(-10, 15),
  returnData = FALSE,
  ...
)
```

Arguments

object	An RLRanges object with featureEnrich already run.
pred_POS_only	If TRUE, only "POS" predicted samples included (see also predictCondition). Default: TRUE.
label_POS_only	If TRUE, only "POS" labeled samples included (samples which are expected to robustly map R-loops, e.g., "D210N" condition R-ChIP data). Default: FALSE.
splitby	Metadata by which to split plots. Can be "none", "prediction", or "label".
limits	Specify limits on data range. This is used for controlling the infinite estimation of odds ratio resulting from fisher's exact test. To remove limits, set c(-Inf, Inf). Default: c(-10, 15).
returnData	If TRUE, plot data is returned instead of plot objects. Default: FALSE
...	For internal use.

Value

A named list of ggplot2::ggplot objects. Names correspond to the annotations provided. See also featureEnrich.

Examples

```
# Example dataset with featureEnrich() already run.
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Make plots, split by prediction
plotEnrichment(rlr, pred_POS_only = FALSE, splitby = "prediction")
```

---

plotRLFSRes *Plot RLFS analysis results*

Description

Plots the results of the R-loop-forming sequences (RLFS) analysis. The plot is a metaplot of the Z score distribution around RLFS with the p value from permutation testing annotated. See also analyzeRLFS.

Usage

```
plotRLFSRes(object, plotName = NULL, fft = FALSE, ...)
```

Arguments

object	An RLRanges object with analyzeRLFS already run. Alternatively, can be the RLFS results object from an RLRanges (from `rlresult(object, "rlfsRes")`).
plotName	A Sample name used for plotting. If blank, the RLRanges `sampleName` metadata entry is used (see RLRanges).
fft	If TRUE, the Fourier transform of the Z-score is plotted instead. Default: FALSE.
...	Additional parameters passed to ggplot2::ggplot.

Value

A ggplot object. See also ggplot2::ggplot.

Examples

```
# Example RLRanges dataset with analyzeRLFS() already run.
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Plot RLFS res
plotRLFSRes(rlr)

# Plot the Fourier transform instead
plotRLFSRes(rlr, fft = TRUE)
```

---

plotRLRegionOverlap *Plot RL-Region overlap with RLRanges*

Description

Convenience function for plotting the overlap between RLRanges and R-loop regions (RL regions) as calculated by rlRegionTest.

Usage

```
plotRLRegionOverlap(object, returnData = FALSE, rlregions_table = NULL, ...)
```

Arguments

object	An RLRanges object with rlRegionTest already run.
returnData	If TRUE, plot data is returned instead of plotting. Default: FALSE
rlregions_table	The table of RLRegions to overlap sample ranges with. Obtained from RLHUb using RLHub::rlregions_meta. Loaded from RLHub if not supplied. Default: NULL.
...	Additional arguments passed to VennDiagram::venn.diagram

Value

A ggplot2::ggplot object containing the venn diagram. Built using ggplotify::as.ggplot.

Examples

```
# Example dataset with rlRegionTest() already run.
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Plot RL-Region overlap
plotRLRegionOverlap(rlr)
```

```
# Return data only
plotRLRegionOverlap(rlr, returnData = TRUE)
```

`predictCondition` *Predict Condition*

Description

Uses the results of analyzeRLFS to predict whether a sample is "POS" (robust R-loop mapping) or "NEG" (poor R-loop mapping). See *details*.

Usage

```
predictCondition(object, rlfsRes = NULL, ...)
```

Arguments

`object`	An RLRanges object with analyzeRLFS already run. Ignored if `rlfsRes` is provided.
`rlfsRes`	If object not supplied, provide the rlfsRes list which is obtained from `rlresult(object, "rlfsRes")`.
`...`	Internal use only.

Details

Following R-loop forming sequences (RLFS) analysis, the quality model (see RLHub::models) is implemented for predicting the sample condition in coordination with other results from analyzeRLFS. A prediction of "POS" indicates robust R-loop mapping, whereas "NEG" indicates poor R-loop mapping. The succeeding sections describe this process in greater detail.

Application of binary classification model:

First, the binary classifier is applied, yielding a preliminary prediction of quality. This is accomplished via the following steps:

1. Calculate the Fourier transform of the Z-score distribution (see analyzeRLFS).
2. Reduce the dimensions to the engineered feature set (see table below).
3. Apply the preprocessing model (see RLHub::models) to normalize these features
4. Apply the classifier (see RLHub::models) to render a quality prediction.

Engineered feature set:
Abbreviations: Z, Z-score distribution; ACF, autocorrelation function; FT, Fourier Transform.

feature	description
Z1	mean of Z
Z2	variance of Z
Zacf1	mean of Z ACF
Zacf2	variance of Z ACF
ReW1	mean of FT of Z (real part)

ReW2	variance of FT of Z (real part)
ImW1	mean of FT of Z (imaginary part)
ImW2	variance of FT of Z (imaginary part)
ReWacf1	mean of FT of Z ACF (real part)
ReWacf2	variance of FT of Z ACF (real part)
ImWacf1	mean of FT of Z ACF (imaginary part)
ImWacf2	variance of FT of Z ACF (imaginary part)

Final quality prediction:

The results from the binary classifier are combined with other results from analyzeRLFS to yield a final prediction. To yield a prediction of "POS" all the following must be `TRUE`:

1. The RLFS Permutation test P value is significant ($p < .05$). Stored as `PVal Significant` in the results object.
2. The Z-score distribution at 0bp is > 0. Stored as `ZApex > 0` in the results object.
3. The Z-score distribution at 0bp is > the start and the end. Sored as `ZApex > ZEdges` in the results object.
4. binary The classifier predicts a label of "POS". Stored as `Predicted 'POS'` in the results object.

Value

An RLRanges object with predictions accessible via `rlresult(object, "predictRes")`.

Structure:

The results object is a named `list` of the structure:

- `Features`
  - A `tbl` with three columns that describe the engineered features used for prediction:
    - `feature`: the name of the feature (see *details*).
    - `raw_value`: The raw value of that feature in the supplied object.
    - `processed_value`: The normalized value of that feature after preprocessing (see *details*).
- Criteria
  - The four criteria which must all be `TRUE` to render a prediction of "POS" (see *details*).
- prediction
  - The final prediction. "POS" indicates robust R-loop mapping, "NEG" indicates poor R-loop mapping.

Examples

```
# Example data with analyzeRLFS already run
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# predict condition
rlr <- predictCondition(rlr)

# With rlfsRes
predRes <- predictCondition(rlfsRes = rlresult(rlr, "rlfsRes"))
```

report *RLSeq Report*

Description

Builds an HTML report to showcase the results available in the supplied RLRanges object (see also RLResults).

Usage

```
report(object, reportPath = "rlreport.html", quiet = FALSE, ...)
```

Arguments

object	An RLRanges object.
reportPath	A path indicating the report output HTML file. Default: "rlreport.html"
quiet	If TRUE, messages are suppressed. Default: FALSE.
...	Arguments passed to rmarkdown::render()

Value

TRUE

Examples

```
# Example data with RLSeq() already run.
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Get a TMP file (only for example usae)
tmp <- tempfile()

# Generate the report
report(rlr, reportPath = tmp)
```

RLRanges-class *Construct RLRanges Dataset*

Description

RLRanges is a subclass of GRanges, which stores R-loop peaks and metadata about the R-loop-mapping experiment, along with results from the analyses in RLSeq.

Usage

```
RLRanges(
  peaks = GenomicRanges::GRanges(),
  coverage = character(1),
  genome = character(1),
  mode = character(1),
  label = character(1),
  sampleName = "User-selected sample",
  qcol = NULL
)
```

Arguments

peaks	Path/URL to peak file or a GRanges object. This file should be in "broadPeak" format if possible. If not, then qcol should be specified.
coverage	Path/URL to the coresponding coverage file (in "bigWig" format). If not supplied, correlation tests will be skipped.
genome	UCSC genome ID. Acceptable types are listed in auxdata (available_genomes entry).
mode	Type of R-loop mapping from which peaks and coverage were derived. Acceptable types are listed in auxdata (available_modes entry). Can be unspecified.
label	"POS" (positive R-loop-mapping sample; e.g., DRIP-Seq S9.6 -RNH1) or "NEG" (negative control sample; e.g., DRIP-Seq S9.6 +RNH1 or Input). Can be unspecified.
sampleName	A unique name for identifying this sample. Can be unspecified.
qcol	The name of the metadata column which contains the score or significance of each peak. For broadPeak (preferred), this is the qvalue (column 11 after accounting for extra columns created during peakset building). If not specified, the last column will be chosen by default. **NOTE**: if supplying narrowPeak form peaks, the last column will NOT be appropriate and QCol should be specified as 11. If FALSE or if no metadata columns exist, it will be left blank and some operations in report() will not fully run.

Value

An object of class RLRanges. These objects are an extension of GRanges with the addition of sample metadata entries and RLResults.

Examples

```
# Example dataset
rlbase <- "https://rlbase-data.s3.amazonaws.com"
cvg <- file.path(rlbase, "coverage", "SRX7671349_hg38.bw")
pks <- system.file("extdata", "SRX7671349_hg38.broadPeak", package="RLSeq")

# Get RLRanges object
rlr <- RLRanges(pks,
```

```
    coverage = cvg, genome = "hg38", label = "NEG",
    mode = "RDIP", sampleName = "RDIP-Seq +RNH1", qcol = 9
)
```

`RLRangesFromRLBase` *Access RLBase samples as RLRanges*

Description

Accessor function which returns any sample in RLBase as an RLRanges object for use with RLSeq. For a full list of available samples, see RLHub::rlbase_samples.

Usage

```
RLRangesFromRLBase(acc, rlsamples = NULL)
```

Arguments

`acc`	The sample ID of the RLBase object. See the `rlsample` column in RLHub::rlbase_samples.
`rlsamples`	The tibble provided by RLHub::rlbase_samples. Providing these data ahead of time speeds up this operation. Default: NULL.

Value

An RLRanges object with all results available.

Examples

```
rlr <- RLRangesFromRLBase("SRX1070676")
```

`rlRegionTest` *R-Loop region test*

Description

Tests the overlap of user-supplied ranges with R-loop regions (RL regions).

Usage

```
rlRegionTest(object)
```

Arguments

`object`	An RLRanges object with genome "hg38".

Details

R-loop regions (RL regions) are consensus sites of R-loop formation. For more information, see RLHub::rlregions. The `rlRegionTest` is a simple function which finds the overlap of user-supplied samples with RL regions and calculates Fisher's exact test via valr::bed_fisher.

Value

An RLRanges object with test results accessible via `rlresult(object, "rlRegionRes")`.

Structure:

The structure of the results is a named `list` containing the following:

- `Overlap`
  - A `tbl` showing the overlap between RL regions and user-supplied ranges.
  - Column description:
    - `chrom` - The chromosome name
    - `start__peaks` - The starting position of the user-supplied peak in the overlap.
    - `end__peaks` - Same as above for end position.
    - `name__peaks` - The name of the user-supplied peak in the overlap (from `names(object)`).
    - `start/end/name__rlregion` - Same as above for RL regions.
    - `strand__rlregion` - The genomic strand of the RL region in the overlap.
    - `.overlap` - The size of the overlap.
- `Test_results`
  - A `tbl` showing the results of the Fisher's exact test. See valr::bed_fisher.

Examples

```
# Example RLRanges data
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# RL Region Test
rlRegionTest(rlr)
```

`rlresult` *RLSeq Results*

Description

Functions for creating and accessing the R-loop results (RL Results). These are a type of object used for holding the results of the tests implemented in RLSeq. They can be accessed using the `rlresult` function.

Usage

```
rlresult(object, resultName)
```

Arguments

`object`	RLRanges object.
`resultName`	Name of the result slot to access. See *details*.

Details

Slot descriptions:

- `featureEnrichment`
  - The `tbl` generated from running featureEnrich.
  - The structure and column descriptions are provided in detail within RLHub::feat_enrich_samples.
- `correlationMat`
  - The `matrix` generated from running corrAnalyze.
  - Contains pairwise pearson correlations between all samples in RLBase and the supplied RLRanges object.
- `rlfsRes`
  - The `list` generated from running analyzeRLFS.
  - See analyzeRLFS for description of structure.
- `geneAnnoRes`
  - The `tbl` generated from running geneAnnotation.
- `predictRes`
  - The `list` generated from running predictCondition.
- `rlRegionRes`
  - The `list` generated from running rlRegionTest.

Value

The contents of the requested slot.

Examples

```
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

rlresult(rlr, "predictRes")
```

RLSeq *RLSeq*

Description

Executes the RLSeq analysis workflow.

Usage

```
RLSeq(object, quiet = FALSE, skip = NULL, ...)
```

Arguments

`object`	An RLRanges object.
`quiet`	If `TRUE`, messages are suppressed. Default: `FALSE`.
`skip`	Analysis steps to skip. Default: `NULL`. See *details* for options.
`...`	Arguments passed to analyzeRLFS.

Details

The `RLSeq()` function does all of the following by default:

1. **RLFS Perm Test**. Runs the analyzeRLFS function to test the enrichment of user-supplied ranges within R-loop-forming sequences. *Cannot be skipped.*
2. **Predict Condition**. Runs the predictCondition function to predict whether the user-supplied sample robustly maps R-loops or not. *Cannot be skipped.*
3. **Feature enrichment test**. Runs the featureEnrich function to test the enrichment of user-supplied ranges within R-loop-relevant genomic features. Skip with `skip="featureEnrich"`.
4. **Correlation Analysis**. Runs the corrAnalyze function to test the correlation of user-supplied R-loop signal with other samples in RLBase around "gold-standard" R-loop regions. Skip with `skip="corrAnalyze"`.
5. **Gene annotation**. Runs the geneAnnotation function to find the overlap of genes with the user-supplied ranges. Skip with `skip="geneAnnotation"`.
6. **R-loop Region Analysis**. Runs the rlRegionTest function to find the overlap of user-supplied ranges with consensus R-loop sites (RL-Regions). Skip with `skip="rlRegionTest"`.

Value

An RLRanges object with results available (see rlresult).

Examples

```
# Example RLRanges
rlr <- readRDS(system.file("extdata", "rlrsmall.rds", package = "RLSeq"))

# Run RLSeq
# `useMask=FALSE`, `ntime=10`, and `skip=` for demonstration purposes here.
rlr <- RLSeq(
    rlr, useMask = FALSE, ntimes = 10,
    skip=c('featureEnrich', 'corrAnalyze', 'geneAnnotation', 'rlRegionTest')
)
```

tableToRegions *Table to Regions*

Description

Helper function to Convert "table" format to "regions" format.

Usage

```
tableToRegions(table)
```

Arguments

table	A tibble in "Table" format from RLHub.

Value

A tibble in "regions" format.

urlExists *Check if URL exists*

Description

Check if URL exists

Usage

```
urlExists(urlcon)
```

Arguments

urlcon	URL to check

Value

logical. TRUE if status code 200, FALSE if not

Index