Package 'RLHub'

May 19, 2022

Title An ExperimentHub package for accessing processed RLSuite data sets

Version 1.2.0

Description |
RLHub provides a convenient interface to the processed data provided within RLSuite, a tool-chain for analyzing R-loop-mapping data sets.
The primary purpose of RLHub is to serve the processed data sets required by the RLSeq R package and the RLBase web service. Additionally, RLHub provides a stand-alone R interface to these data, benefiting users who are addressing questions related to R-loop regions (RL-Regions), R-loop-binding proteins (RLBPs), R-loop co-localizing factors, and the differences between R-loop-mapping methods.
The full data-generating protocol is found here: https://github.com/Bishop-Laboratory/RLBase-data.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

VignetteBuilder knitr

Imports AnnotationHub, ExperimentHub, utils

biocViews ExperimentData, ExperimentHub, ExpressionData, ReproducibleResearch, SequencingData, Genome, ChIPSeqData, RNASeqData, GEO, PackageTypeData, DNASeqData, ENCODE

Suggests knitr, DT, rmarkdown, testthat, BiocStyle

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

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

BiocType ExperimentHub

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

git_branch RELEASE_3_15

git_last_commit 4b08b12

git_last_commit_date 2022-04-26

Date/Publication 2022-05-19

Author Henry Miller [aut, cre, cph] (<https://orcid.org/0000-0003-3756-3918>),
Alexander Bishop [ths, cph] (<https://orcid.org/0000-0002-5742-4387>)

Maintainer Henry Miller <millerh1@uthscsa.edu>

R topics documented:

annots_primary_hg38 2
feat_enrich_samples 6
fft_model 8
gene_exp 9
gs_signal 10
rlbase_samples 11
rlbps 14
rlfs_res 15
rlregions_annot 16

Index **21**

`annots_primary_hg38` *Annotations*

Description

Genomic features relevant to R-loops. Both mm10 and hg38 annotations are available.

Usage

```
annots_primary_hg38(quiet = FALSE)

annots_full_hg38(quiet = FALSE)

annots_primary_mm10(quiet = FALSE)

annots_full_mm10(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

The `list` contains `tbl` objects (tidyverse-style data frames) containing annotations as genomic ranges. The **primary** annotations (e.g., `annots_primary_hg38()`) are an abbreviated version of the **full** annotations (e.g., `annots_full_hg38()`). See the description below for further details:

Databases available:

This section details the annotation databases which are available in RLHub. See the succeeding section ("Objects available based on accessor") for a list of which databases are available within each function. All processing was performed using this script as part of the RLBase-data processing protocol.

- **Centromeres**
  - Description: Centromere locations within the genome.
  - Source: UCSC table centromeres.
- **CNA**
  - Description: Copy-number alterations found in inherited disorder cell lines. See source for full description. CNV states (0-4) are represented in the data as separate types. For example, Deep deletion (0) sites are accessed with `annots_full_hg38()$CNA__0`.
  - Source: UCSC table coriellDelDup.
- **Cohesin**
  - Description: This database contains manually-curated STAG2 and STAG1 ChIP-Seq data reprocessed by the RLHub authors to find consensus STAG1 and STAG2 sites between cell lines (using chip-r.)
  - Source: processed data source *Pan et al* and processed further as part of the RLBase-data pipeline here.
- **CpG_Islands**:
  - Description: CpG island predicted locations throughout the genome.
  - Source: UCSC table cpgIslandExt.
- **Encode_CREs**:
  - Description: The UCSC Encode_CREs table contains putative promoter-like ("prom"), promoter-enhancer-like ("enhP"), distal-enhancer-like ("enhD"), H3K4me3 ("K4me3"), and CTCF ("CTCF") chromatin states across the genome.
  - Source: UCSC table encodeCcreCombined.
- **Encode_Histone**:
  - Description: consensus peaks from histone ChIP experiments downloaded from Encode. Biological replicates were summarized with chip-r.
  - Source: Encode v121. Manifest of samples downloaded is here.
- **encodeTFBS**:
  - Description: The collection of curated transcription-factor binding profiles from encode, made available by UCSC table browser.
  - Source: UCSC table encRegTfbsClustered.
- **G4Qexp**:
  - Description: G4-Quadruplex ChIP-Seq data
  - Source: GEO accession GSE63874.
- **G4Qpred**:

  - Description: Re-processed and binned G4-Quadruplex Predictions. The type names for this database are the G4Q prediction classes and follow the pattern `tl:N_nl:N_gn:N`. `tl`: the length of guanine tracts in region; `nl`: number of locations for G4 formation; `gn`: the number of possible simultaneous G4 structures. For more information, see the source publication here. Due to the large number of possible configurations of `tl`, `nl`, and `gn`, they were binned based on frequency.
  - Source: Figshare Rouchka et al. and direct download link
- **knownGene_RNAs**:
  - Description: RNA species provided by UCSC KnownGene, split up by the "transcriptType" column from the source table.
  - Source: UCSC table knownGene.
- **Microsatellite**:
  - Description: Microsatellite DNA regions predicted based on motif.
  - Source: UCSC table microsat.
- **PolyA**:
  - Description: List of predicted poly-A sites, split by the "name2" column of the source table.
  - Source: UCSC table wgEncodeGencodePolyaV38.
- **RBP_ChIP**:
  - Description: ChIP-Seq data sets for RNA-binding proteins (RBPs) generated by *Nostrand et al* for Encode. Data are split by ChIP target.
  - Source: Encode v121. Manifest of samples here from source study *Nostrand et al*.
- **RBP_eCLiP**:
  - Description: eCLiP-Seq data sets for RNA-binding proteins (RBPs) generated by *Nostrand et al* for Encode. Data are split by eCLiP target.
  - Source: Encode v121. Manifest of samples here from source study *Nostrand et al*.
- **Repeat_Masker**:
  - Description: Repeat masker table from UCSC containing genomic annotations for predicted repetitive elements, split by class of repetitive element ("repClass").
  - Source: UCSC table rmsk.
- **skewr**:
  - Description: Regions of G or C-skew profiled using the `skewr` program. See the RLBasedata `README.md` for steps.
  - Source: From UCSC goldenPath, hg38 and mm10 genomes. hg38 and mm10 gene GTF. CpG islands for mm10 and hg38 provided as described in the **CpG_Islands** entry above. Processing proceeded using `skewr` with `stochHMM v0.38`.
- **snoRNA_miRNA_scaRNA**:
  - Description: snoRNA, miRNA, and scaRNA species provided by UCSC table browser and split by the "type" column.
  - Source: UCSC table wgRna.
- **Splice_Events**:
  - Description: UCSC table of alternative splice events predicted from transcriptome data sets. Split by "name" column.
  - Source: UCSC table knownAlt.

- **Transcript_Features**:
  - Description: Transcript features (e.g., "exon", "intron", etc) provided by Bioconductor TxDb packages. Split based on the following features: "Exon", "Intron", "fiveUTR", "threeUTR", "TSS", "TTS", "Intergenic".
  - Source: TxDb for hg38 and mm10.
- **tRNAs**:
  - Description: UCSC table containing predicted tRNA genes.
  - Source: UCSC table tRNAs.

Objects available based on accessor function:

Here, we show which objects are available with each accessor function:

DataBase name	annots_primary_hg38()	annots_primary_mm10()	annots_full_hg38()	annots_ful
Centromeres	FALSE	FALSE	TRUE	FALSE
CNA	FALSE	FALSE	TRUE	FALSE
Cohesin	FALSE	FALSE	TRUE	FALSE
CpG_Islands	TRUE	TRUE	TRUE	TRUE
Encode_CREs	TRUE	TRUE	TRUE	TRUE
Encode_Histone	FALSE	FALSE	TRUE	FALSE
encodeTFBS	FALSE	FALSE	TRUE	FALSE
G4Qexp	FALSE	FALSE	TRUE	FALSE
G4Qpred	TRUE	FALSE	TRUE	FALSE
knownGene_RNAs	TRUE	FALSE	TRUE	FALSE
Microsatellite	FALSE	FALSE	TRUE	TRUE
PolyA	TRUE	FALSE	TRUE	FALSE
RBP_ChIP	FALSE	FALSE	TRUE	FALSE
RBP_eCLiP	FALSE	FALSE	TRUE	FALSE
Repeat_Masker	TRUE	TRUE	TRUE	TRUE
skewr	TRUE	TRUE	TRUE	TRUE
snoRNA_miRNA_scaRNA	TRUE	FALSE	TRUE	FALSE
Splice_Events	FALSE	FALSE	TRUE	TRUE
Transcript_Features	TRUE	TRUE	TRUE	TRUE
tRNAs	TRUE	TRUE	TRUE	TRUE

Object structure:

Accessor functions (e.g., `annots_primary_hg38()`) return a named `list` of `tbl` objects that specify feature ranges. Below, we detail the naming and structure of each.

List names:

The names in the `list` objects provided by each accessor function (e.g., `annots_primary_hg38()`) follow this structure: `DataBase__Type`. `DataBase` is the database from which annotations were derived and `Type` indicates the specific annotations from the database which are included in the `tbl`. This is required as some databases produce > 1 type of annotation (e.g., **Transcript_Features** contains "Exon" (`Transcript_Features__Exon`) and "Intron" (`Transcript_Features__Intron`)).

`tbl` *structure:*

Each `tbl` returned has the following structure:

chrom	start	end	strand	id
chr1	10015	10498	+	1
chr1	10614	11380	+	2
...				

Columns:

- "chom" - the Chromosome of the feature range (UCSC style)
- "start" - the starting position of the feature range.
- "end" - the end position of the feature range.
- "strand" - the strand of the feature range.
- "id" - A unique ID for the feature range.

Value

A list of `tbl` objects. See details.

Examples

```
annos <- annots_primary_hg38()

annos <- annots_full_hg38()

annos <- annots_primary_mm10()

annos <- annots_full_mm10()
```

`feat_enrich_samples` *Feature Enrichment*

Description

Feature Enrichment per RL-Region (feat_enrich_rlregions) and per Sample (feat_enrich_samples).

Usage

```
feat_enrich_samples(quiet = FALSE)

feat_enrich_rlregions(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

For every R-loop-mapping sample in RLBase (feat_enrich_samples) or for each of the three RL-Regions databases (feat_enrich_rlregions), peaks were calculated and then tested via `RLSeq::featureEnrich()` to determine enrichment within all available genomic features.

The processing steps are described in the RLBase-data README.md.

Structure:

Example data:

db	type	num_total_peaks	num_tested_peaks	num_tested_anno_ranges	num_total_anno_ranges
Cohesin	STAG1	9892	10000	9935	9939
Cohesin	STAG2	10613	10000	9935	10696
CpG_Islands	CpG_Islands	28189	10000	9989	31144
...	...	...	...	...	...

Column description:

- `db` - the database from which annotations were derived. See annotations.
- `type` - the type from which annotations were derived. See annotations.
- `num_tested_peaks`: The number of peak ranges tested for enrichment.
- `num_total_peaks`: The total number of peaks in the sample.
- `num_tested_anno_ranges`: The number of annotation ranges which the peaks were tested against.
- `num_total_anno_ranges`: The total number of ranges in the annotation.
- `avg_reldist_rl`: The mean relative distances between peaks and annotation ranges as described here.
- `avg_reldist_shuf`: Same as `avg_reldist_rl` except with peaks shuffled randomly to provide a negative control.
- `pval_reldist`: The `ks.test` p value from comparing the distribution of the peak and shuffle relative distances as described here.
- `stat_fisher_rl`: The the odds ratio from the fisher's exact test as described here.
- `stat_fisher_shuf`: Same as above, but with the shuffled peaks to provide a negative control.
- `pval_fisher_rl`: The p value from Fisher's exact test as described here.
- `pval_fisher_shuf`: Same as above, but with the shuffled peaks to provide a negative control.
- `experiment` (only `feat_enrich_samples()`): The R-loop mapping sample for which the calculation was performed. See rlbase_samples.
- `opt` (only `feat_enrich_rlregions()`): The type of RL-Region (see rlregions) which was tested.

Value

A `tbl` object. See details.

Examples

```
fes <- feat_enrich_samples()

fes <- feat_enrich_rlregions()
```

`fft_model` *Models*

Description

Models used for predicting sample label ("POS": robustly maps R-loops; "NEG": poorly maps R-loops) based on R-loop-forming sequences analysis (`RLSeq::analyzeRLFS()`). These models are used with `RLSeq::predictCondition()`.

Usage

```
fft_model(quiet = FALSE)

prep_features(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

The models were developed as part of a semi-automated online learning scheme found in the RLBase-data protocol here. Briefly, R-loop-forming sequences (RLFS) analysis was performed using `RLSeq::analyzeRLFS()` for every sample peakset in RLBase (see rlfs_res for full results). The samples were then manually inspected and any which starkly differed from their label were removed. Out of 693 possible samples, 135 were excluded due to a mismatch with their label. The remaining steps were performed automatically.

- First,The non-discarded samples were partitioned 50:25:25 (train:test:discovery). Feature transformation was performed on the full data-set using the "YeoJohnson" transform along with typical standardization via `caret::preProcess()`.
- Then, feature selection was performed in the discovery set using `Boruta::Boruta()`.
- Then, the training set was then trained using a stacked ensemble model:
  - The ensemble model is a Random Forest and the 5 base models in the stack are:
    - Latent Dirichlet allocation
    - Recursive partitioning
    - Generalized linear model (logit)
    - K-nearest neighbors
    - Support vector machine (radial)
  - 10-fold 5-repeated cross-validation was implemented during training.

- Finally, The model was then evaluated in the testing set. It demonstrates an accuracy of 0.9043. For more details, see the HTML report here.

Structure:

- `prep_features()`
  - A feature-transform model which prepares the data for classification.
  - It is an object of class `preProcess` from the `caret::preProcess()` function call.
- `fft_model()`
  - A binary classifier which returns "POS" or "NEG".
  - It is an object of class `caretStack` from the `caretEnsemble::caretList()` function call.

Usage:

These models are used internally by `RLSeq::predictCondition()`.

Value

A model object from the `caret` package.

Examples

```
fftModel <- fft_model()

pfModel <- prep_features()
```

`gene_exp` *Gene Expression*

Description

A `SummarizedExperiment` containing the gene expression results from RNA-Seq data sets published in tandem with R-loop-mapping data sets.

Usage

```
gene_exp(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

For each R-loop-mapping data set in RLBase, any associated RNA-Sequencing data was also downloaded and quantified using `salmon v1.4.0`. This enables comparison of R-loop abundance and gene expression. The list of SRA accessions from which data was obtained are found here: `colData(gene_exp())$sample`.

The processing steps are described in the RLBase-data README.md.

Structure:

- colData
  - Contains metadata about every sample matching the metadata. See rlbase_samples for a full description.
- assays
  - A `SimpleAssays` object containing three matrices (genes X samples):
    - `cts` - Raw read counts
    - `tpm` - TPM-normalized read counts
    - `vst` - VST-transformed read counts. (See `DESeq2::vst()`)

Value

A `SummarizedExperiment` object. See details.

Examples

```
geneExp <- gene_exp()
```

`gs_signal` *GS-Signal*

Description

A `tbl` containing the coverage from every human sample in RLBase around "gold-standard" R-loop sites.

Usage

```
gs_signal(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

Gold-standard R-loop sites are those which were profiled with ultra-long-read R-loop sequencing (SMRF-Seq) as described in *Malig et al., 2021.*. A bed file containing these sites was derived and lifted to hg38. Sites were then extended by 100kb bi-directionally and then binned into 1kb bins.

For each sample in RLBase, coverage tracks were calculated from alignments using `RLPipes`. Then, coverage tracks were summed within each of the 1kb bins using `deepTools` and wrangled into a `tbl`.

The processing steps are described in the RLBase-data README.md.

These data are used by `RLSeq::corrAnalyze()` to determine how well a query sample correlates with the samples in RLBase.

Structure:

Sample of data:

location	ERX2277510	ERX2277511	ERX3974959	...
chr1_67636071_67637071	23	22	93	...
chr1_67637071_67638071	44	39	84	...
chr1_67638071_67639071	26	30	100	...
...	...	...	...	...

Column desc:

- `location` - 1kb bins within "gold-standard" R-loop regions profiled with ultra-long-read R-loop sequencing (SMRF-Seq).
- `ERX.../SRX...` - Columns corresponding to RLBase samples (see rlbase_samples). For each sample, the total coverage (bigWig file) within the range is shown.

Value

A `tbl` object. See details.

Examples

```
gss <- gs_signal()
```

`rlbase_samples` *RLBase Sample Manifest*

Description

A `tbl` containing metadata about each sample in RLBase.

Usage

```
rlbase_samples(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

RLBase samples were curated by hand in Excel from searching keywords such as "R-loops" and "RNA:DNA hybrids" in GEO, SRA, and PubMed. Where R-loop mapping data was publically available, entries were added to the Excel spreadsheet such that every sample (SRX.../ERX.../GSM...) had it's own line. Information was noted for each sample, such as the "mode" (the type of R-loop mapping it was) and the "Condition" (e.g., "RNaseH1", "WKKD", etc). When genomic input controls were available, they were manually matched to the experimental samples for which they could serve as a background control during peak calling.

The up-to-date excel sheet is found here.

Throughout the process of analyzing the data (see RLBase-data), additional metadata was added to the sample sheet (see structure for full account).

Structure:

`rlbase_samples` is a `tbl` with the structure:

rlsample	label	condition	mode	lab	tissue	genotype	other	PMID	group	family	ip_t
SRX1070676	POS	S96	DRIPc	Fred Chedin	NT2	WT	NT	27373332	rl	DRIP	S9.6
SRX1070677	POS	S96	DRIPc	Fred Chedin	NT2	WT	NT	27373332	rl	DRIP	S9.6
SRX1070678	POS	S96	DRIP	Fred Chedin	NT2	WT	NT	27373332	rl	DRIP	S9.6
...	...	...	...	...	...	...	...	...	...	...	...

Column description:

- `rlsample` - The unique ID of the sample, same as in the SRA.
- `label` - Label corresponding to the author-supplied condition of the sample. "POS" indicates the sample should robustly map R-loops, "NEG" indicates the opposite.
- `condition` - The specific condition for each sample.
- `mode` - The type of R-loop mapping for each sample.
- `lab` - The senior author on the publication from which the data was provided.
- `tissue` - The tissue condition for the sample.
- `genotype` - The sample's genotype.
- `other` - A column for other pertinent metadata provided by the authors.
- `PMID` - The PMID associated with the sample
- `group` - One of "rl" (R-loop mapping) or "exp" (Expression data/RNA-Seq).
- `family` - The family of the "mode" (e.g., "DRIP" includes "sDRIP", "DRIPc", "qDRIP", etc)
- `ip_type` - The IP type of the "mode" for each sample. One of "S9.6", "dRNH" (dead RNaseH1), or "None".
- `strand_specific` - Whether the sample is stranded.
- `moeity` - The moeity which was IP'd (if applicable)
- `bisulfite_seq` - Whether the data uses bisulfite conversion sequencing (e.g., "BisDRIP-Seq" samples)

- `file_type` - The type of data (always "public" for RLBase samples).
- `experiment_original` - The original name of this sample as entered by hand in the curated Excel spreadsheet (usually converted from GSM to SRX).
- `control_original` - Same as above for the accompanying control sample (if applicable.)
- `study` - The SRA study accession for this sample.
- `name` - The sample's name as entered in SRA.
- `paired_end` - A logical indicating whether the data is paired end.
- `read_length` - The read length for the sample.
- `control` - The RLBase ID of the genomic input control sample corresponding to this sample (if applicable)
- `eff_genome_size` - The effective genome size based on read length and genome (calculated with the `khmer` package) see relevant portion of RLBase-data protocol here.
- `genome` - The UCSC genome ID for this sample.
- `prediction` - The prediction from running `RLSeq::predictCondition()`.
- `discarded` - A logical indicating whether this sample was discarded during model building for a mismatch with its "label" (see models).
- `numPeaks` - The number of peaks called for this sample.
- `expsamples` - The IDs of any corresponding expression samples.
- `exp_matchCond` - The meta data used to match this sample to any corresponding expression samples (if applicable).
  - **Method**: Some R-loop mapping studies also had matched RNA-Seq data. In these cases, they were also recorded with the same metadata (where applicable) as R-loop mapping samples. To match expression and R-loop samples, the `study`, `tissue`, `genotype`, and `other` columns were compared iteratively for each R-loop sample. If all four were a match with at least one expression samples, then those four columns would be assigned as the `exp_matchCond`. If only three were available, then they would become the `exp_matchCond`. To see the order in which columns were checked for possible matches, view the `buildExpression.R` script in the RLBase-data repo. See also the section on `corr(R/PVal/PAdj)` column in rlregions.
- `coverage_s3` - The location of the coverage tracks (`.bw`) in the AWS S3 bucket for RLBase data ('s3://rlbase_data/').
- `peaks_s3` - Same as above for peak files (`.broadPeak`)
- `fastq_stats_s3` - Same as above for fastq QC statistics data (`.json`).
- `bam_stats_s3` - Same as above for BAM QC statistics data (`.txt`).
- `report_html_s3` - Same as above for reports from `RLSeq::report()` (`.html`).
- `rlranges_rds_s3` - Same as above for RLRanges R objects, as in `RLSeq::RLRanges()` (`.rds`)
- `rlfs_rda_s3` - Same as above for rlfs_res objects generated by `RLSeq::analyzeRLFS()` (`.rda`).

Value

A `tbl`.

Examples

```
rlsamples <- rlbase_samples()
```

`rlbps` *R-loop Binding Proteins*

Description

A list of annotations for genomic features relevant to R-loops. Both mm10 and hg38 annotations are available.

Usage

```
rlbps(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

R-Loop binding proteins (RLBPs) are proteins which were discovered by mass-spectrometry after S9.6 immuno-precipitation (S9.6 is an antibody that binds R-loops). The data were aggregated from three studies which analyzed RLBPs in this manner:

1. *Nadel et al., 2015* - article
2. *Wang et al., 2018* - article
3. *Cristini et al., 2018* - article

The data were downloaded from the supplementary materials of each study and processed using custom R scripts as part of the RLBase-data pipeline. This processing involved normalization of mass-spec abundance calculations and the derivation of a "combined score" for ranking all the proteins.

Structure:

geneName	NADEL.2015	WANG.2018	CRISTINI.2018	combinedScore
DDX21	0.9	0	0.968	1.87
DHX9	0.85	0.0635	0.831	1.74
...	...	...	...	...

Columns:

- geneName
  - The official HGNC symbol of the RLBP
- NADEL.2015
  - Normalized abundance of protein from *Nadel et al., 2015*
- WANG.2018
  - Normalized abundance of protein from *Wang et al., 2018*

- CRISTINI.2018
  - Normalized abundance of protein from *Cristini et al., 2018*
- combinedScore
  - Sum of normalized abundance across studies.

Value

A `tbl` object. See details.

Examples

```
rlbpsData <- rlbps()
```

`rlfs_res` *RLFS-Test Results*

Description

A list containing the results of R-loop-forming sequences analysis (`RLSeq::analyzeRLFS()`) and subsequent sample classification `RLSeq::predictCondition()` as "POS" (robustly maps R-loops) or "NEG" (poorly maps R-loops) with all samples in RLBase.

Usage

```
rlfs_res(quiet = FALSE)
```

Arguments

`quiet`	If TRUE, messages are suppressed. Default: FALSE.

Details

Source:

R-loop-forming sequences were computationally predicted using `QmRLFS-finder.py`. The relevant section of the RLBase data prep protocol can be found here. Briefly, genomes available from UCSC were downloaded and analyzed with `QmRLFS-finder.py` on default settings. The resulting RLFS ranges were converted to `.bed` format.

RLFS analysis (via `RLSeq::analyzeRLFS()`) implements permutation testing to calculate the enrichment of a query R-loop mapping peakset within the relevant RLFS ranges. This analysis produces a p value and a Z-score distribution which are subsequently analyzed with `RLSeq::predictCondition()` (see models), yielding a quality prediction of "POS" (robustly maps R-loops) or "NEG" (poorly maps R-loops).

This approach was applied to every sample in RLBase (see rlbase_samples) to yield rlfs_res.

Structure:

`rlfs_res` is a named list where each name is the ID of a sample in RLBase (see rlbase_samples).

Each `list` item is itself a `list` comprising the following:

- rlfsData - A list with this structure:
  - perTestResults
    - An object of the class permTestResultsList from regioneR.
    - Contains the results of permutation testing.
  - Z-scores
    - An object of the class localZScoreResultsList from regioneR.
    - Contains the results of local Z-score analysis +/-5kb around each RLFS.
- rlfsPred - A list with this structure:
  - Features
    - A tbl containing the raw and transformed values of features used in RLSeq::predictCondition().
  - Criteria
    - A list containing the results of analyzing the four criteria which must all be TRUE for a prediction of "POS":
      - PVal Significant - The permutation test p value is < .05
      - ZApex > 0 - The center of the Z-score distribution is > 0.
      - ZApex > ZEdge - The center of the Z-score distribution is > left edge and right edge.
      - Predicted 'Case' - The classifier predicted the "Case" label ("POS").
  - prediction
    - The final prediction: "POS" or "NEG"

Value

A named list.

Examples

```
rlfsRes <- rlfs_res()
```

---

rlregions_annot *R-loop Regions*

Description

R-loop regions (RL regions) are consensus sites of R-loop formation derived from a meta-analysis of the R-loop mapping experiments in RLBase. RLHub includes information about these regions (rlregions_meta), the overlap of genomic features with RL regions (rlregions_annot), and the read count matrices for each sample in RLBase within each RL region (rlregions_counts).

Usage

```
rlregions_annot(quiet = FALSE)

rlregions_meta(quiet = FALSE)

rlregions_counts(quiet = FALSE)
```

Arguments

quiet	If TRUE, messages are suppressed. Default: FALSE.

Details

R-loop regions (RL regions) are consensus sites of R-loop formation derived from a meta-analysis of the R-loop mapping experiments in RLBase. In RLHub, we provide access to:

- rlregions_meta - The metadata describing these RL regions.
- rlregions_annot - RL regions overlapped with genome features (see annotations).
- rlregions_counts - RL region read counts across RLBase samples.

Value

A list of tbl objects. See details.

Source

RL Regions - rlregions_meta:

RL Regions were derived during this step of the RLBase-data protocol.

Here is a brief summary of the key processing steps.

1. Human RLBase samples which were labeled as "POS", classified as "POS" (see rlfs_res), and which have at least 5000 peaks were selected.
2. Then, these peaks were then randomly downsampled to 5000 ranges each and aggregated.
3. Then, the hg38 genome was binned into 10bp bins and the number of aggregated peaks was overlapped with each bin and counted, producing a bedGraph.
4. Then, macs3 was implemented to call peaks on the bedGraph file.
5. Finally, the resulting ranges were combined to create summary ranges.

Note: steps 2-4 were performed separately on datasets from dRNH (catalytically-dead RNaseH1) R-loop mapping experiments and S9.6-based experiments due to the noted differences in R-loop mapping between these modalities. In step 5, they were combined to create summary ranges, and the original source (dRNH or S9.6) was noted in the resulting metadata.

RL Regions anno - rlregions_annot:

Once RL regions were generated, they were overlapped with the annotations found in `annots_full_hg38()`. This was done separately for dRNH-derived, S9.6-derived, and combined RL regions.

RL Regions Counts - rlregions_counts:

Once RL regions were generated, `Rsubread::featureCounts()` was used to count the number of overlapping reads from the `.bam` alignment files for each sample in RLBase (rlbase_samples) within each RL region (only "combined" regions used).

These data were then VST-transformed and TPM was also calculated.

Structure

RL Regions - rlregions_meta:

`rlregions_meta()` is a `tbl` with the following structure:

rlregion	location	is_rlfs	source	confidence_level	medSignalVal	medPVal	med
All_RL64926	chrM:0-16569:.	TRUE	dRNH S96	53.21	1.962881	20.92420	17.0
All_RL19159	chr5:49656610-49661950:.	FALSE	dRNH S96	36.55	2.826610	19.16160	15.2
All_RL50251	chr16:46386300-46391280:.	FALSE	dRNH S96	31.68	3.258560	25.21965	21.0
...	...	...	...	...	...	...	

Column description:

- `rlregion` - The ID of the RL Region.
- `location` - The location of the RL Region. This column follows the pattern "chrom:start-stop:strand"
- `is_rlfs` - Logical indicating whether the RL region overlaps with an R-loop forming sequence.
- `source` - Indicates whether the RL region was discovered from catalytically-dead RNaseH1-based mapping (`dRNH`), S9.6-based mapping (`S96`), or both (`dRNH S96`).
- `conservation_score` - The percent of samples in which this range was found (calculated separately for `dRNH` and `S96` sites and then averaged together for sites from `dRNH S96` source).
- `medSignalVal` - The median broadPeak "signalVal" (see UCSC specification) of sample peaks in the RL region.
- `medPVal` - The median broadPeak -log10 pval (see UCSC specification) of sample peaks in the RL region.
- `medQVal` - The median broadPeak -log10 qval (adjusted pval) (see UCSC specification) of sample peaks in the RL region.
- `avgSignalVal/avgPVal/avgQVal` - same as above, but with mean instead of median.
- `medNumPeaks` - The median number of peaks for samples which overlapped with this RL Region.
- `avgNumPeaks` - Same as above with mean instead of median.
- `nSamples` - The number of samples which overlap with this peak.
- `nModes` - The number of different R-loop mapping methods in which this RL region was found.
- `nTissues` - The number of different tissues in which this RL region was found.
- `ip_types` - The "IP types" (S9.6, dRNH, or None) in which this RL Region was found.
- `nIPType` - The number of different "IP types" in which this RL region was found.
- `pct_case` - The percentage of samples which overlapped with this RL region that were predicted "POS" by `RLSeq::predictCondition()`.
- `geneIDs` - The IDs of all genes overlapping with this RL Region.
- `allGenes` - The Gene Symbols of all genes overlapping with this RL Region.
- `mainGenes` - The Gene Symbols of all genes (which appear in pathway databases) overlapping with this RL Region.
- `is_repeat` - A logical indicating whether the RL Region occurs in a repetitive genomic region as determined by repeat masker (and also peri-centromeric regions were included.)
- `confidence_score` - A confidence score for ranking RL Regions.
  - **Method**:
    - **Summary**: `confidence_score` is the geometric mean of features relevant to RL Region robustness with a penalty for regions found only in `dRNH` and `S96` alone.

    * **Calculation**: The `confidence_score` is calculated as following: `C = m * (cons * ns * mq * ms)^(1/4)`; where `m` is a penalty (of `1.25` weight) for RL Regions not found in both `S96` and `dRNH` sets. `cons` is the standardized log2 transform of the `conservation score`. `ns` is the standardization of the `nSamples` column. `mq` is the standardized log2 transform of the `medQVal` column. `ms` is the standardized log2 transform of the `medSignalVal` column.
- `corr(R/PVal/PAdj)` - Results of correlation analysis between R-loop abundance and expression within each RL Region. (see also gene_exp)
  - **Method**: R-loop alignment files were summarized within RL Regions to make a count matrix, which was then `log2(tpm + 1)` normalized. Then, the expression `log2(tpm + 1)` matrix was summarized to the RL Regions level as the mean expression of genes overlapping with the R-loop. Then, R-loop and expression samples were paired based on the study of origin, sample condition, and other factors (73 sample pairs in total) (see the description of the `exp_matchCond` column in rlbase_samples). Within each pairing, there were some many-to-many mappings between R-loop and expression samples, due to multiple samples having the same `exp_matchCond`. In these cases, RL region expression and R-loop abundance was summarized as the mean across redundant samples. Finally, the Spearman correlation between R-loop abundance and expression was calculated within each R-loop region (see cor.test), yielding `Rho` and the `p.value`. See also the relevant processing script (`buildExpression.R`) from the RLBase-data repo.
  - `corrR` - Spearman's Rho for the correlation between expression and R-loop abundance within each RL Region (see gene_exp).
  - `corrPVal` - The pvalue from running `cor.test` as described above.
  - `corrPAdj` - The result of multiple testing correction on `corrPVal` using `p.adjust` with default arguments. (See also p.adjust)

RL Regions anno - rlregions_annot:

`rlregions_annot()` is a `tbl` with the following structure:

rlregion	annotation
All_RL4615	CNA__3__4
All_RL4615	CNA__1__8
All_RL4615	G4Qexp__G4Q_PDS_NaK_GSE63874__95848
...	...

Column description:

- `rlregion` - The RL Region ID corresponding to the rlregions_meta table.
- `annotation` - The annotation which overlaps with the query RL region. Annotation names follow the convention "DataBase__Type__ID" (see also annotations).

RL Regions Counts - rlregions_counts:

`rlregions_counts()` is a `SummarizedExperiment` with the following structure:

- `dim`: 64418 532
- `assays`
  - `cts` - Raw read counts of each RLBase sample within each RL Region.
  - `tpm` - The TPM normalization of `cts`

  - vst - The VST transform of cts (see also DESeq2::vst()).
- colData - A DataFrame with structure:

experiment	label	strand_specific	paired_end	mode	prediction	discarded	numPeaks	bam
SRX113814	POS	FALSE	FALSE	DRIVE	NEG	TRUE	337	../RLBase-data/rlbas
SRX113812	POS	FALSE	FALSE	DRIP	NEG	TRUE	78	../RLBase-data/rlbas
SRX2675003	POS	TRUE	FALSE	R-ChIP	POS	FALSE	10331	../RLBase-data/rlbas
...	...	...	...	...	...	...	...	...

Column description:

- experiment - The RLBase sample ID
- label - The label provided by the data submitters ("POS" or "NEG")
- strand_specific - A logical indicating whether the data is stranded or not.
- paired_end - A logical indicating whether the data is paired-end or not.
- prediction - The predicted label from RLSeq::predictCondition(). Can be "POS" or "NEG".
- discarded - A logical indicating whether the sample was discarded during model building due to a mismatch with its label.
- numPeaks - The number of peaks called from this sample.
- bam - The relative path of the bam file for this sample.
- bam_avail - A logical indicating whether the bam file was available.

Examples

```
rlregions <- rlregions_meta()

rlregionsAnno <- rlregions_annot()

rlregionsCounts <- rlregions_counts()
```

Index

annotations, *7*, *17*, *19*
annotations *(*annots_primary_hg38*)*, 2
annots_full_hg38 *(*annots_primary_hg38*)*, 2
annots_full_hg38(), *17*
annots_full_mm10 *(*annots_primary_hg38*)*, 2
annots_primary_hg38, 2
annots_primary_mm10 *(*annots_primary_hg38*)*, 2

Boruta::Boruta(), *8*

caret::preProcess(), *8*, *9*
caretEnsemble::caretList(), *9*
cor.test, *19*

DESeq2::vst(), *10*, *20*

feat_enrich_rlregions, *6*, *7*
feat_enrich_rlregions *(*feat_enrich_samples*)*, 6
feat_enrich_samples, *6*, 6, *7*
fft_model, 8

gene_exp, 9, *19*
gs_signal, 10

models, *13*, *15*
models *(*fft_model*)*, 8

p.adjust, *19*
prep_features *(*fft_model*)*, 8

rlbase_samples, *7*, *10*, *11*, 11, *15*, *17*, *19*
rlbps, 14
rlfs_res, *8*, *13*, *15*, 15, *17*
rlregions, *7*, *13*
rlregions *(*rlregions_annot*)*, 16
rlregions_annot, *16*, 16, *17*, *19*
rlregions_counts, *16*, *17*, *19*
rlregions_counts *(*rlregions_annot*)*, 16
rlregions_meta, *16*, *17*, *19*
rlregions_meta *(*rlregions_annot*)*, 16
RLSeq::analyzeRLFS(), *8*, *13*, *15*
RLSeq::corrAnalyze(), *11*
RLSeq::featureEnrich(), *7*
RLSeq::predictCondition(), *8*, *9*, *13*, *15*, *16*, *18*, *20*
RLSeq::report(), *13*
RLSeq::RLRanges(), *13*
Rsubread::featureCounts(), *17*