Package ‘eRNAkit’

April 22, 2025

Type Package

Title A tool kit for characterising eRNA reguratory functions beyond the nucleus.

Version 0.2.1

Description eRNAkit, is a comprehensive and accessible resource designed to address the gap in non-canonical eRNA functions in humans.
It includes an array of analtyical workflows, db integrating eRNA subcellular localisation, RNA–RNA interaction and expression.
It also includes a ui app "eRNAkitApp to interactively explore the db.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports mgcv,
Biostrings,
GenomicRanges,
tidyr

Contents

byRange *Find Overlaps Between emi$core and a Genomic Coordinate String*

Description

Parses a coordinate string and finds overlapping entries in a emi$core. Returns a data.frame including all metadata columns.

Usage

```
byRange(erob, s)
```

Arguments

erob	A ‘GRanges‘ object to search within.
s	A character string in the format "chr1:100-200" or "1:100-200".

Value

A data.frame of overlapping GRanges entries with metadata columns. Returns an empty data.frame if no overlaps are found.

Examples

```
find_overlap_df(my_gr, "chr1:2000-3000")
```

check_null — *Check if all elements of a db "emi" is NULL.*

Description

This is specific for Dr. Anene's DBs. It is likely useful after subsetting the DB.

Usage

```
check_null(db)
```

Arguments

db	The DB object to check for NULL

Value

TRUE if all elements are NULL or FALSE.

chimericPRO — *Process chimeric junction file from star-alignment.*

Description

Function to process chimeric junction files into a bedpe format. It splits the pairs to make it easy to find overlaps.

Usage

```
chimericPRO(file)
```

Arguments

file	The file path for the chimeric.junction from star.

Value

A list with n1 and n2 pairs. The name column match the ID.

compute_mad *Compute median absolute deviation with constant.*

Description

Compute median absolute deviation with constant.

Usage

```
compute_mad(x)
```

Arguments

`x`	A vector of values to compute.
`constant`	The constant value to add mad equal to 0.
`na.rm`	Option to remove "TRUE" or not "FALSE" NA values.

Value

MAD for each value in the vector

Examples

```
compute_mad(c(2, 4, 7, 8,9))
```

eeCigar *Extract end coordinates from cigar string.*

Description

Function to extract end position from cigar string. The function only adds M, D and S to the count.

Usage

```
eeCigar(start = 2, strand = "+", cigar = "10M5D20M2I")
```

Arguments

`start`	Start position.
`strand`	The strand. Defualt is + standard.
`cigar`	The cigar string.

Value

The end coordinate.

`entropyTS` *Compute Tissue/Cell Type Specificity Scores*

Description

This function calculates specificity scores for eRNA or genes based on their expression profiles across multiple tissues or cell types. The specificity score is computed using Shannon entropy and a logarithmic correction.

Usage

```
entropyTS(x)
```

Arguments

`x`	A data frame or matrix where rows represent eRNAs and columns represent different tissues or cell types. Only numeric columns (expression values) are used for calculations.

Details

The function follows these steps:

1. Compute expression ratios by normalizing each eRNA's expression across tissues/cell types.
2. Calculate Shannon entropy for each eRNA.
3. Compute the specificity score as $log_2(N) - H$, where \(N\) is the number of cell types and \(H\) is Shannon entropy.
4. Define specificity by checking if the highest expression ratio is at least 2× the second highest and if the specificity score is greater than 1.

Value

A data frame with three columns:

ID The identifier (row names of input data).

Specificity_Score The calculated specificity score for each row.

Is_Specific The tissue/cell type where the transcript is specific, or FALSE if not specific.

`eRNASeq` *Extract eRNA sequences from a FASTA file using BED coordinates*

Description

This function extracts DNA sequences for eRNAs from a reference FASTA file based on coordinates provided in a BED file. It optionally supports strand-specific extraction and writes the result to a new FASTA file.

Usage

```
eRNASeq(b, fasta)
```

Arguments

`b`	Data frame with intervals. It must "chr", "start", "end", "strand" as names. With any other mcols.
`fasta`	Character. Path to the reference genome in FASTA format.

Value

A `DNAStringSet` object containing the extracted sequences.

`eRNkitApp` *Lunch eRNAkit Web application.*

Description

This lunches the eRNAkit UI for interactive exploration of the emi db.

Usage

```
eRNkitApp()
```

Value

The Web UI

`getGRange` *Get genomic range object for annGE or chimericPRO.*

Description

Make GRanges object from dataframe. This function expects a specific input so make sure you have that. It must have columns named "chr", "start", "end", "strand"

Usage

```
getGRange(tab = exmp)
```

Arguments

`tab`

Value

Returns GRanges

`listDB` *List all database from the eRNAkit.*

Description

Function to list DB .rds included with the eRNAkit.

Usage

```
listDB()
```

Value

A character list of all the dbs included

Examples

```
listDB()
```

`loadDB` *Load a database from the eRNAkit.*

Description

Function to load DB .rds DB into environment.

Usage

```
loadDB(name)
```

Arguments

`name`	Name of the db set to load (without .rds)

Value

The loaded R object

Examples

```
ribo <- loadDB("ribo")
```

`log2_count2cpm` *Count to log2 count per million "CPM"*

Description

Convert count to log2 CPM on matrix

Usage

```
log2_count2cpm(df = "data", side = "r:1.c:2")
```

Arguments

`df:`	Data frame
`side:`	Operate on row-1 or column-2

Value

Data frame of converted counts

Examples

```
log2_count2cpm(df, 2)
```

`make_windows` *Create n.bp Genomic Windows or Retain Original Interval*

Description

Given a genomic interval, this function splits it into non-overlapping windows of n base pairs if the interval length is greater than n. If the interval is shorter than or equal to n, it is returned as-is.

Usage

```
make_windows(chrom, start, end, name, size = 100)
```

Arguments

`chrom`	A character or numeric value representing the chromosome name or number.
`start`	An integer representing the start coordinate of the genomic interval (1-based).
`end`	An integer representing the end coordinate of the genomic interval (inclusive).
`name`	A character string providing an identifier for the interval.
`size`	An integer indicating the size of the windows "n".

Value

A `data.frame` with columns `chrom`, `start`, `end`, and `name`. If the interval is longer than size, the result contains multiple rows representing the windows. For shorter intervals, a single row is returned.

`max_cols` *Get id of max column by rows.*

Description

Function to get ids of max columns in a vector. Second max column is include for future tooling. 0 values are set to NA to avoid return 0 as max.

Usage

```
max_cols(x)
```

Arguments

`x`	Named vector to get max values.

Value

Returns the max and second max ids.

Examples

```
max_cols(c(A = 10, B = 20, C = 30))
```

`max_cols_matrix` *Get id of max column by rows.*

Description

Function to get ids of max columns in a data frame by row. Second max column is include for future tooling. 0 values are set to NA to avoid return 0 as max.

Usage

```
max_cols_matrix(df)
```

Arguments

`df`	Data frame or matrix to get max values by row.

Value

Returns a data frame of max and second max column ids.

Examples

```
max_cols(matrix(1:9, nrow = 3, ncol = 3))
```

outlier_matrix — *Detect outliers in a matrix.*

Description

Function to apply the two outlier function to a matrix. This is specific to eRNAkit usage

Usage

```
outlier_matrix(m, type = "pnorm", name = "eRNA")
```

Arguments

`m`	matrix of expression values gene/eRNA in column and sample in rows. It expects the row.names to set to gene IDs.
`type`	String indicating the p-value estimation method. "pnorm" or "permutation"
`name`	The name to use for the new gene column. This option is to differentiate eRNA/mRNA runs.

Value

Table of results based on the output of the outlier functions.

Examples

```
outlier_matrix(matrix(1:9, nrow = 3, ncol = 3), "pnorm")
```

parseGR — *Parse a Genomic Coordinate String into a GRanges Object*

Description

Takes a genomic coordinate string in the format ‘"chr1:100-200"‘ or ‘"1:100-200"‘ (case-insensitive) and returns a ‘GRanges‘ object representing that region.

Usage

```
parseGR(c)
```

Arguments

`c`	A character string specifying the genomic region. Valid formats include ‘"chr1:100-200"‘, ‘"CHR1:100-200"‘, or ‘"1:100-200"‘. Whitespace is ignored. Chromosome prefix is normalized to lowercase ‘"chr"‘ format.

Value

A ‘GRanges‘ object corresponding to the parsed genomic range.

Examples

```
parse_genomic_range("chr1:100-200")
parse_genomic_range("CHR2:1000-2000")
parse_genomic_range("3:500-1000")
```

`permutation_outliers` *Detect outliers in a named vector using permutations.*

Description

Function to detect outliers in a named vector of values. It uses permutations to get empirical p-values.

Usage

```
permutation_outliers(x, n = 1000)
```

Arguments

`x`	A named vector of values.
`n`	Number of permutations to run.

Value

A table of all the results including intermediate tables.

Examples

```
permutation_outliers(c(2, 4, 7, 8,9), n=100)
```

`plotLoc` *Plot the figures for localisation data.*

Description

This function expects tables pre-processed by the winloc function.

Usage

```
plotLoc(df, t = c("p1", "p2", "p3"))
```

Arguments

`df`	Data frame to plot from the winloc list of tables.
`t`	The type of plot p1, p2, or p3, matching the winloc table

Value

GGplot2 plot

`plotOC` *Plot Expression Across Tissue or Cells*

Description

This function generates a bar plot showing the expression levels of eRNAs across different organs or cells. It expects the data from emi db. The data is transformed into a long format using ‘pivot_longer‘, excluding certain columns (‘E‘, ‘Specificity_Score‘, ‘Is_Specific‘, ‘expressed‘). The plot is ordered by expression values in descending order, and a color scale is used based on the ‘E‘ column.

Usage

```
plotOC(x, t = "Organ")
```

Arguments

`x`	A data frame containing the expression data with columns representing organs or cells and an ‘E‘ column indicating the eRNA expression values.
`t`	A string specifying the column name to be used for the x-axis. Defaults to ‘"Organ"‘. It can be any column name in ‘x‘ (e.g., ‘"Cells"‘).

Value

A ‘ggplot‘ object showing the expression data as a bar plot.

Examples

```
# Example 2: Plot expression across cells
plotOC(x2, "Cells")
```

`pnorm_outliers` *Detect outliers in a named vector using normalized z-score.*

Description

Function to detect outliers in a named vector of values. It uses CDF of pnorm to get p-values

Usage

```
pnorm_outliers(x)
```

Arguments

`x`	A named vector of values.

Value

A table of all the results including intermediate tables.

Examples

```
pnorm_outliers(c(2, 4, 7, 8,9))
```

---

remove_zeros — *Filter matrix to remove rows with sum 0.*

Description

Filter matrix to remove rows with sum 0.

Usage

```
remove_zeros(matrix)
```

Arguments

`matrix`

Value

The filtered matrix

Examples

```
remove_zeros(matrix(1:9, nrow = 3, ncol = 3))
```

---

rna2rna — *Extract and count the eRNA:mRNA interactions.*

Description

Function to extract and count eRNA:mRNA interactions from GRanges object.

Usage

```
rna2rna(chimGR, geneGR, enhGR)
```

Arguments

`chimGR`	Chimeric junction GRanges object. It should be made from ChimPRO output.
`geneGR`	Gene annotation GRanges object. It should be made from annGE$gene.
`enhGR`	Enhancer annotation GRanges object. It should be made from annGE$erna.

Value

Interactions counts. Uses IDs for naming enhancer and genes.

`rna2rnaR` *Extract extract locations of eRNA:mRNA interactions.*

Description

Function to extract locations of eRNA:mRNA interactions from GRanges object.

Usage

```
rna2rnaR(chimGR, geneGR, enhGR)
```

Arguments

`chimGR`	Chimeric junction GRanges object. It should be made from ChimPRO output.
`geneGR`	Gene annotation GRanges object. It should be made from annGE$gene.
`enhGR`	Enhancer annotation GRanges object. It should be made from annGE$erna.

Value

bedpe type table. xx.x and xx.y are the target pairs

`stringS` *Process Gene String*

Description

Splits a comma-separated gene string into a character vector and trims any surrounding white spaces.

Usage

```
stringS(ss)
```

Arguments

`ss`	A single character string with gene names separated by commas (e.g. ‘"gene1, gene2 , gene3"‘).

Value

A character vector of cleaned gene names.

`subDB` *Filter a list of data frames by column and value*

Description

This function filters each data frame in a list by a specific value in a specified column. It skips any data frames that do not contain the column.

Usage

```
subDB(db, col, value)
```

Arguments

`db`	A list of data frames.
`col`	The name of the column to filter by (e.g., "E" or "G").
`value`	The value to filter for.

Value

A subset of the db

Examples

```
subDB(emi, "E", "en100035")
```

`wide2long` *Pivot a wide db table to long*

Description

Converts wide emi db table to long format for processing

Usage

```
wide2long(df, keep_cols)
```

Arguments

`df`	A data frame in wide format.
`keep_cols`	A character vector of column names to retain (not pivot).

Value

A data frame in long format with columns: the retained identifiers, a ‘variable‘ column for the former column names, and a ‘value‘ column for the values.

`winLoc` *Generate location-based subsets from emi loc table*

Description

This function processesthe emi db loc table.

Usage

```
winLoc(subloc)
```

Arguments

`subloc`	emi$loc subset prefered.

Details

It returns: - p1: CPM values split by RNA type (only cytosol and nucleus) - p2: REI values for ‘cytosol‘ polyA+ RNA split by Cell line - p3: REI values comparing subcellular cytoplasmic compartments to cytosol (total RNA), split by Enhancer (E) - d: The original filtered input dataframe

Value

A named list with components: ‘p1‘, ‘p2‘, ‘p3‘, and ‘d‘.

Index