Package ‘SeedMatchR’

October 24, 2023

Title Find Matches to Canonical SiRNA Seeds in Genomic Features

Version 1.1.1

Description On-target gene knockdown using siRNA ideally results from binding fully complementary regions in mRNA transcripts to induce cleavage. Off-target siRNA gene knockdown can occur through several modes, one being a seed-mediated mechanism mimicking miRNA gene regulation. Seed-mediated off-target effects occur when the ~8 nucleotides at the 5’ end of the guide strand, called a seed region, bind the 3’ untranslated regions of mRNA, causing reduced translation. Experiments using siRNA knockdown paired with RNA-seq can be used to detect siRNA sequences with potential off-target effects driven by the seed region. 'SeedMatchR' provides tools for exploring and detecting potential seed-mediated off-target effects of siRNA in RNA-seq experiments. 'SeedMatchR' is designed to extend current differential expression analysis tools, such as 'DESeq2', by annotating results with predicted seed matches. Using publicly available data, we demonstrate the ability of 'SeedMatchR' to detect cumulative changes in differential gene expression attributed to siRNA seed regions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

biocViews

Suggests knitr, rmarkdown, testthat (>= 3.0.0), org.Rn.eg.db

Config/testthat/edition 3

Imports dplyr, Biostrings, ggmsa, msa, ggplot2, utils, AnnotationHub, GenomeInfoDb, GenomicFeatures, cowplot, testit, lifecycle, stringr, stats

VignetteBuilder knitr

Depends R (>= 4.1.0)

URL https://tacazares.github.io/SeedMatchR/

NeedsCompilation no

Author Tareian Cazares [aut, cre] (<https://orcid.org/0000-0003-4624-6156>),
Gulcin Ozer [aut] (<https://orcid.org/0000-0002-8315-8051>),
Jibo Wang [aut],
Rick Higgs [aut],
Eli Lilly and Company [cph]

Maintainer Tareian Cazares <tareian.cazares@lilly.com>

Repository CRAN

Date/Publication 2023-10-24 20:30:02 UTC

R topics documented:

`check_gene_list_overlap` *Check if input gene lists overlap*

Description

Check if input gene lists overlap

Usage

```
check_gene_list_overlap(gene.lists)
```

Arguments

`gene.lists`	A list of gene lists. example: list(c("gene1", "gene2"), c("gene1"))

Value

Warning if gene sets overlap

Examples

```
# Overlap
check_gene_list_overlap(list(c("gene1", "gene2"), c("gene1")))

#No overlap
check_gene_list_overlap(list(c("gene1", "gene2"), c("gene3")))
```

`deseq_fc_ecdf` *Plot the ECDF for DESeq2 log2(Fold Changes)*

Description

This functions will take DESeq2 results as a `data.frame` and plot the ecdf for the input `gene.lists`.

The gene sets to plot should be provided as a list of lists.

Example:

`gene.lists = list("Background" = c("gene1", "gene2"), "Target" = c("gene2", "gene3"), "Overlap" = c("gene2"))`

This function will also perform statistical testing if `plot.hist` is TRUE. The output will be saved to a PDF if an `output.filename` is provided.

Users can define the groups that are to be compared in the statistical test using the `null.name` and `target.name` arguments. The names must be found in `gene.lists`. The `factor.order` is used to order the groups in the analysis.

This functions returns:

- `$plot`: The ECDF plot
- `$stats`: The stats results object

Usage

```
deseq_fc_ecdf(
  res,
  gene.lists,
  title = "ECDF",
  output.filename = NULL,
  palette = SeedMatchR.palette,
  factor.order = NULL,
  x.lims = c(-1, 1),
  stats.test = NULL,
  alternative = "greater",
  null.name = 1,
  target.name = 2,
  height = 5,
  width = 5,
  dpi = 320
)
```

Arguments

`res`	The DESeq2 results dataframe
`gene.lists`	A nest list of gene names. Example: gene.lists = list("Background" = gene.list2, "Target" = gene.list1, "Overlap" = gene.list3)
`title`	The tile of the plot
`output.filename`	If the output filename is provided, then the plot is saved.
`palette`	The color palette to use for your curves
`factor.order`	The order to use for the legends
`x.lims`	The xlimits range
`stats.test`	The statistic test to use. Options: KS, Kuiper, DTS, CVM, AD, Wass
`alternative`	The alternative hypothesis to test. Options: greater, less, two.sided
`null.name`	The name in the gene.list to use as the null for ecdf plots
`target.name`	The name in the gene.list to use as the target for ecdf plots
`height`	Plot height in inches
`width`	Plot width in inches
`dpi`	The dpi resolution for the figure

Value

A ggplot object for the ECDF plot

Examples

```
library(dplyr)

guide.seq = "UUAUAGAGCAAGAACACUGUUUU"

anno.db = load_species_anno_db("human")

features = get_feature_seqs(anno.db$tx.db, anno.db$dna)

# Load test data
get_example_data("sirna")

sirna.data = load_example_data("sirna")

res <- sirna.data$Schlegel_2022_Ttr_D1_30mkg

# Filter DESeq2 results for SeedMatchR
res = filter_deseq(res, fdr.cutoff=1, fc.cutoff=0, rm.na.log2fc = TRUE)

res = SeedMatchR(res, anno.db$gtf, features$seqs, guide.seq, "mer7m8")

# Gene set 1
```

```
mer7m8.list = res$gene_id[res$mer7m8 >= 1]

# Gene set 2
background.list = res$gene_id[!(res$mer7m8 %in% mer7m8.list)]

ecdf.results = deseq_fc_ecdf(res,
list("Background" = background.list, "mer7m8" = mer7m8.list),
stats.test = "KS",
factor.order = c("Background", "mer7m8"),
null.name = "Background",
target.name = "mer7m8")
```

`download_parse_file` *Download and parse DESeq2 output from GSE184929*

Description

Download and parse DESeq2 output from GSE184929

Usage

```
download_parse_file(download.path, output.path)
```

Arguments

`download.path`	File url to be downloaded
`output.path`	Filename used for saving downloaded file

Value

DESeq2 results as a data.frame.

Examples

```
download_parse_file()
```

ecdf_stat_test *Test for differences in log2(Fold Change) ECDFs between two gene lists using the `stats` package*

Description

This function uses the `stats` package to test the ECDF of log2(Fold Changes) between two groups based on DESeq2 analysis.

The inputs of this function are a DESeq2 results `data.frame` and two sets of gene IDs called `gene.list1` and `gene.list2`. The functions will look for a column called `log2FoldChange` in the dataframe.

Usage

```
ecdf_stat_test(
  res,
  gene.list1,
  gene.list2,
  stats.test = "KS",
  alternative = "greater"
)
```

Arguments

`res`	Input results file data frame
`gene.list1`	Gene list 1: Usually null distribution
`gene.list2`	Gene list 2: Target set of genes
`stats.test`	Stats test to use. Options: KS or Wilcoxen
`alternative`	The alternative hypothesis to test. Options: greater, less, two.sided

Value

A vector containing the dstat and pvalue

Examples

```
library(dplyr)

guide.seq = "UUAUAGAGCAAGAACACUGUUUU"

anno.db = load_species_anno_db("human")

features = get_feature_seqs(anno.db$tx.db, anno.db$dna)

# Load test data
get_example_data("sirna")
```

```
sirna.data = load_example_data("sirna")

res <- sirna.data$Schlegel_2022_Ttr_D1_30mkg

# Filter DESeq2 results for SeedMatchR
res = filter_deseq(res, fdr.cutoff=1, fc.cutoff=0, rm.na.log2fc = TRUE)

res = SeedMatchR(res, anno.db$gtf, features$seqs, guide.seq, "mer7m8")

# Gene set 1
mer7m8.list = res$gene_id[res$mer7m8 >= 1]

# Gene set 2
background.list = res$gene_id[!(res$mer7m8 %in% mer7m8.list)]

ecdf.res = ecdf_stat_test(res, mer7m8.list, background.list)
```

`filter_deseq` *Filter DESEQ2 Results for SeedMatchR*

Description

Filter `DESeqDataSet` results for use with seed matching and counting functions.

The filtering criteria are:

Filter out genes that are not expressed or counted at all: baseMean = 0 & pvalue = NA & log2FoldChange = NA

Filter out genes that are expressed, but there is not difference across groups: log2FoldChange = 0

Filter out genes with extreme outliers: pvalue = NA and padj = NA

Filter out genes that have been excluded by independent filtering. padj = NA

Filter results by the `fdr.cutoff`

Filter the results by the `log2FoldChange`

Filter the results by the `baseMean`

Remove NA gene_ids and log2FoldChange values

Usage

```
filter_deseq(
  res,
  fdr.cutoff = 1,
  fc.cutoff = 0,
  rm.na.name = FALSE,
  rm.na.log2fc = FALSE,
  baseMean.cutoff = 0
)
```

Arguments

`res`	The DESEQ2 results as a data frame
`fdr.cutoff`	The false discovery rate cutoff to use.
`fc.cutoff`	The fold change cutoff to use. The absolute value will be used as the cutoff and values greater-than-or-equal-to will be kept.
`rm.na.name`	Remove na values from the gene_name column
`rm.na.log2fc`	Remove na values from the log2FoldChange column
`baseMean.cutoff`	The minimum baseMean expression cutoff

Value

A modified DESEQ2 results table that has been filtered

Examples

```
# Load test data
get_example_data("sirna")

sirna.data = load_example_data("sirna")

res <- sirna.data$Schlegel_2022_Ttr_D1_30mkg

# Filter DESeq2 results for SeedMatchR
res = filter_deseq(res, fdr.cutoff=1, fc.cutoff=0, rm.na.log2fc = TRUE)
```

`get_example_data` *Download example DESeq2 data from GEO*

Description

This function will download data that can be used for SeedMatchR. Choosing 'sirna' will download 3 DESeq2 results files from GSE184929. Choosing 'mirna' will download the miRDB database as a tsv.

Usage

```
get_example_data(example.type)
```

Arguments

`example.type`	Name of the example to load. Options: sirna, mirna

Value

None?

Examples

```
get_example_data()
```

`get_feature_seqs` *Get transcripts features and feature sequences*

Description

This function is used to get the genomic features of interest and the DNA sequences associated with them. This function takes advantage of the `GenomicFeatures` package functions `threeUTRsByTranscript`, `fiveUTRsByTranscript`, `exonsBy`, `intronsByTranscript`, and `cdsBy`. These functions are used to generate the features given an input `tx.db` object. A 2bit dna input is also required for extracting features sequences.

The output of the this function is:

- `$db`: the feature `GRanges` object
- `$seqs`: `DNAStringSet` of sequences associated to those features

Usage

```
get_feature_seqs(tx.db, dna, feature.type = "3UTR")
```

Arguments

`tx.db`	A tx.db object
`dna`	A 2bit dna sequence
`feature.type`	The type of feature to return. Options: 3UTR, 5UTR, exons, introns, cds

Value

list containing the feature db object and the feature sequences

Examples

```
anno.db = load_species_anno_db("human")

features = get_feature_seqs(anno.db$tx.db, anno.db$dna)
```

`get_seed`	*Get the target seed sequence given a canonical seed name and input sequence*

Description

Given a sequence greater than 8 bp oriented 5' -> 3' and a seed definition, this function will return an object containing seed-specific sequence information. Users can input a custom seed name, but must provide the start position (`start.pos`) and stop position (`stop.pos`) that define the range of the seed sequence.

Built-in options: `mer8`, `mer7A1`, `mer7m8`, `mer6`

Note: The seed definitions `mer8` and `mer7A1` force a U at position g1. This results in an A in the target sequence being searched.

Usage

```
get_seed(guide.seq, seed.name = "mer7m8", start.pos = 1, stop.pos = 8)
```

Arguments

`guide.seq`	A character string greater than 8 bp and oriented 5'-> 3'.
`seed.name`	The seed name of interest. Options: mer8, mer7A1, mer7m8, mer6. If not in the default list, the start.pos and stop.pos arguments will be used to define the seed.
`start.pos`	The start position for a custom seed definition
`stop.pos`	The stop position for a custom seed definition

Value

An object with the entries:

- `Guide`: Input guide sequence. Input is expected to be RNA.
- `Seed.Name`: The seed name.
- `Seed.Seq.RNA`: The seed sequence as a `RNAString`
- `Seed.Seq.DNA`: The seed sequence as a `DNAString`
- `Target.Seq`: The target DNA sequence based on the reverse complement of the seed as a `DNAString`

Examples

```
# Example Ttr from Schlegel et al. 2022
guide.seq = "UUAUAGAGCAAGAACACUGUUUU"

# Get seed match
seed.seq = get_seed(guide.seq, "mer7m8")
```

`load_example_data` *Load example DESeq2 data into the environment*

Description

Load example DESeq2 data into the environment

Usage

```
load_example_data(example.type)
```

Arguments

`example.type`	Name of the example to load. Options: sirna, mirna

Value

Loads either the Schlegel 2022 RNAseq data or miRDB into the environment.

Examples

```
load_example_data()
```

`load_species_anno_db` *Load species specific* `AnnotationDb`

Description

Use `AnnotationHub` to load species-specific GTF and 2bit DNA sequences. This function currently works for human, rat, and mouse.

The function will return:

- `$gtf`: A `GRanges` object containing the GTF information
- `$tx.db`: A `tx.db` object made from the GTF
- `$dna`: The 2bit DNA sequence as a `DNAStringSet`

Usage

```
load_species_anno_db(species.name, remove.na.rows = TRUE)
```

Arguments

`species.name`	Species name. Options: human, rat, mouse
`remove.na.rows`	Remove rows with NA in the gene_id column

Value

Species specific `AnnotationDb`

Examples

```
anno.db = load_species_anno_db("human")
```

`plot_seeds` *Plot the Guide Strand with different optional seeds*

Description

Plot the Guide Strand with different optional seeds

Usage

```
plot_seeds(guide.seq)
```

Arguments

`guide.seq`	Guide a.k.a anti-sense sequence oriented 5' > 3'. Sequence must be greater than 8 bp.

Value

A msaggplot of the guide sequence in addition to the available seed sequences

Examples

```
library(msa)

# Ttr siRNA sequence
guide.seq = "UUAUAGAGCAAGAACACUGUUUU"

# generate seed plot
plotted.seeds = plot_seeds(guide.seq)
```

SeedMatchR *Find seed matches in genomic features*

Description

Find seed matches in a `DNAStringSet` object of sequences. This function will use `get.seed` extract the seed sequence from the guide sequence. The seed is then searched across all rows of the `DNAStringSet` object using `vpatterncount`.

This function returns the input DESeq2 results `data.frame` with an additional column that contains the counts for the input `seed.name`.

Usage

```
SeedMatchR(
  res,
  gtf,
  seqs,
  sequence,
  seed.name = "mer7m8",
  col.name = NULL,
  mismatches = 0,
  indels = FALSE,
  tx.id.col = TRUE
)
```

Arguments

`res`	A DESeq2 results `data.frame`
`gtf`	GTF file used to map features to genes. The object must have columns transcript_id and gene_id
`seqs`	The `DNAStringSet` object with sequence information for features. The names of the sequences should be the transcript names.
`sequence`	The `DNAString` guide sequence oriented 5' > 3'.
`seed.name`	The name of specific seed to extract. Options are: mer8, mer7A1, mer7m8, mer6
`col.name`	The string to use for the column name. Defaults to seed name
`mismatches`	The number of mismatches to allow in search
`indels`	Whether to allow indels in search
`tx.id.col`	Use the transcript_id column instead of gene_id

Value

A modified DESeq2 results dataframe that has column named after the seed of choice representing the number of match counts.

Examples

```
library(dplyr)

seq = "UUAUAGAGCAAGAACACUGUUUU"

anno.db = load_species_anno_db("human")

features = get_feature_seqs(anno.db$tx.db, anno.db$dna)

# Load test data
res <- Schlegel_2022_Ttr_D1_30mkg

# Filter DESeq2 results for SeedMatchR
res = filter_deseq(res, fdr.cutoff=1, fc.cutoff=0, rm.na.log2fc = TRUE)

res = SeedMatchR(res, anno.db$gtf, features$seqs, seq, "mer7m8")
```

Index