Package 'scanMiRApp'

July 19, 2023

Type Package

Title scanMiR shiny application

Version 1.6.0

Date 2022-10-11

Imports AnnotationDbi, AnnotationFilter, AnnotationHub, BiocParallel, Biostrings, data.table, digest, DT, ensembldb, fst, GenomeInfoDb, GenomicFeatures, GenomicRanges, ggplot2, htmlwidgets, IRanges, Matrix, methods, plotly, rintrojs, rtracklayer, S4Vectors, scanMiR, scanMiRData, shiny, shinycssloaders, shinydashboard, shinyjqui, stats, utils, waiter

Suggests knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0), shinytest, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, BSgenome.Rnorvegicus.UCSC.rn6

Description A shiny interface to the scanMiR package. The application enables the scanning of transcripts and custom sequences for miRNA binding sites, the visualization of KdModels and binding results, as well as browsing predicted repression data. In addition contains the IndexedFst class for fast indexed reading of large GenomicRanges or data.frames, and some utilities for facilitating scans and identifying enriched miRNA-target pairs.

Depends R (>= 4.0)

License GPL-3

VignetteBuilder knitr

RoxygenNote 7.1.2

biocViews miRNA, SequenceMatching, GUI, ShinyApps

Config/testthat/edition 3

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

git_branch RELEASE_3_17

git_last_commit 81b3eae

git_last_commit_date 2023-04-25

Date/Publication 2023-07-18

Author Pierre-Luc Germain [cre, aut] (<https://orcid.org/0000-0003-3418-4218>),
Michael Soutschek [aut],
Fridolin Gross [ctb]

Maintainer Pierre-Luc Germain <pierre-luc.germain@hest.ethz.ch>

R topics documented:

`enrichedMirTxPairs` *enrichedMirTxPairs*

Description

Identifies pairs of miRNA and target transcripts that have an unexpectedly high number of sites.

Usage

```
enrichedMirTxPairs(m, minSites = 5, max.binom.p = 0.001)
```

Arguments

`m`	A GRanges of matches, as produced by `findSeedMatches`. This will be filtered down to only 8mer and 7mer sites.
`minSites`	The minimum number of sites for a given miRNA-target pair to be considered.
`max.binom.p`	The maximum binomial p-value of miRNA-target pairs.

Value

A data.frame of top combinations, including number of sites and the log-transformed binomial p-value.

Examples

```
# we create a dummy scan (see `runFullScan`)
library(scanMiR)
seqs <- getRandomSeq(n=10)
mirs <- c("TTGTATAA","AGCATTAA")
m <- findSeedMatches(seqs,mirs,verbose=FALSE)
# we look for enriched pairs
res <- enrichedMirTxPairs(m, minSites=1, max.binom.p=1)
res
```

`getTranscriptSequence` *getTranscriptSequence*

Description

Utility wrapper to extracts the sequence of a given transcript (UTR or CDS+UTR).

Usage

```
getTranscriptSequence(
  tx = NULL,
  annotation,
  annoFilter = NULL,
  extract = c("UTRonly", "withORF", "exons"),
  ...
)
```

Arguments

`tx`	The ensembl ID of the transcript(s)
`annotation`	A `ScanMiRAnno` object.
`annoFilter`	An optional 'AnnotationFilter' or 'AnnotationFilterList' to further filter the set of transcripts to be extracted
`extract`	Which parts of the transcripts to extract. For 'UTRonly' (default) only the 3' UTR regions are extracted, 'withORF' additionally extracts the coding regions, and 'exons' extracts all exons
`...`	Passed to `AnnotationHub`

Value

A `DNAStringSet`.

Examples

```
anno <- ScanMiRAnno("fake")
seq <- getTranscriptSequence( tx="ENSTFAKE0000056456", annotation=anno )
```

IndexedFst-class *IndexedFst*

Description

Objects of the IndexedFst class enable fast named random access to FST files. This is particularly appropriate for large data.frames which often need to be accessed according to the (e.g. factor) value of a particular column.

Usage

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

## S4 method for signature 'IndexedFst'
summary(object)

## S4 method for signature 'IndexedFst'
names(x)

## S4 method for signature 'IndexedFst'
length(x)

## S4 method for signature 'IndexedFst'
lengths(x)

## S4 method for signature 'IndexedFst'
nrow(x)

## S4 method for signature 'IndexedFst'
ncol(x)

## S4 method for signature 'IndexedFst'
colnames(x)

## S4 method for signature 'IndexedFst,ANY,ANY'
x[[i, j = NULL, ...]]

## S4 method for signature 'IndexedFst,ANY,ANY,ANY'
x[i, j = NULL, ..., drop = TRUE]

## S4 method for signature 'IndexedFst'
x$name

## S4 method for signature 'IndexedFst'
head(x, n = 6L, ...)
```

```
## S4 method for signature 'IndexedFst'
as.data.frame(x, name)
```

Arguments

object	an IndexedFst object
x	an IndexedFst object
i	the desired index (either numeric or name)
j, drop	ignored
...	ignored
name	the indexed name to fetch
n	the desired number of rows

Value

Depends on the method

Author(s)

Pierre-Luc Germain, <pierre-luc.germain@hest.ethz.ch>

See Also

saveIndexedFst, loadIndexedFst

Examples

```
# we first create and save an indexed FST file
tmp <- tempdir()
f <- system.file(tmp, "test")
d <- data.frame( category=sample(LETTERS[1:4], 10000, replace=TRUE),
                 var2=sample(LETTERS, 10000, replace=TRUE),
                 var3=runif(10000) )
format(object.size(d),units="Kb")
saveIndexedFst(d, "category", f)
rm(d)
# we then load the index, and can use category names for random access:
d <- loadIndexedFst(f)
format(object.size(d),units="Kb")
nrow(d)
names(d)
head(d$A)
```

`loadIndexedFst` *Saving and loading IndexedFst*

Description

Functions to save or load and indexed `fst` file

Saves a data.frame (or GRanges object) into an indexed FST file.

Usage

```
loadIndexedFst(file, nthreads = 1)

saveIndexedFst(
  d,
  index.by,
  file.prefix,
  nthreads = 1,
  index.properties = NULL,
  add.info = list(),
  ...
)
```

Arguments

`file`	Path to the fst file, it's index (.idx), or their prefix.
`nthreads`	Number of threads to use for reading (default 1). This does not affect the loading of the index itself, but will affect all downstream reading operations performed on the object. If NULL, will use 'fst::threads_fst()'.
`d`	A data.frame or `GRanges` object
`index.by`	A column of 'd' by which it should be indexed.
`file.prefix`	Path and prefix of the output files.
`index.properties`	An optional data.frame of properties, with the levels of 'index.by' as row names.
`add.info`	An optional list of additional information to save.
`...`	Passed to 'write.fst'

Value

'loadIndexedFst' returns an object of class `IndexedFst-class`, and 'saveIndexedFst' returns nothing.

See Also

`IndexedFst-class`

`IndexedFst-class`

Examples

```
# we first create and save an indexed FST file
tmp <- tempdir()
f <- system.file(tmp, "test")
d <- data.frame( category=sample(LETTERS[1:4], 10000, replace=TRUE),
                 var2=sample(LETTERS, 10000, replace=TRUE),
                 var3=runif(10000) )
saveIndexedFst(d, "category", f)
# we then load the index, and can use category names for random access:
d <- loadIndexedFst(f)
```

`plotSitesOnUTR` *plotSitesOnUTR*

Description

Wrapper function with minimal arguments to plot scanMiR-Binding sites on 3'UTRs of specified transcripts. The red dashed line indicates the background threshhold is indicated, the lightblue dashed line shows the average 8mer dissociation rate of the given miRNA

Usage

```
plotSitesOnUTR(
  tx,
  annotation,
  miRNA = NULL,
  label_6mers = FALSE,
  label_notes = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`tx`	An ensembl TranscriptID
`annotation`	A `ScanMiRAnno` object.
`miRNA`	A miRNA name in the mirbase format (eg. "hsa-miR-485-5p"), a 'KdModel', or a miRNA sequence or target seed.
`label_6mers`	Logical whether to label 6mer sites in the plot
`label_notes`	Logical whether to label special sites in the plot (as TDMD or Slicing)
`verbose`	Logical; whether to print updates on the processing
`...`	Any further arguments passed to `findSeedMatches`

Value

Returns a ggplot.

Examples

```
anno <- ScanMiRAnno("fake")
plotSitesOnUTR( tx="ENSTFAKE0000056456", annotation=anno,
                miRNA="hsa-miR-155-5p" )
```

---

runFullScan *runFullScan*

Description

Runs a full miRNA scan on all protein-coding transcripts (or UTRs) of an annotation.

Usage

```
runFullScan(
  annotation,
  mods = NULL,
  annoFilter = NULL,
  extract = c("UTRonly", "withORF", "exons"),
  onlyCanonical = TRUE,
  shadow = 15,
  cores = 1,
  maxLogKd = c(-1, -1.5),
  save.path = NULL,
  ...
)
```

Arguments

annotation	A ScanMiRAnno object
mods	An optional 'KdModelList' (defaults to the one in 'annotation')
annoFilter	An optional 'AnnotationFilter' or 'AnnotationFilterList' to filter the set of transcripts to be extracted
extract	Which parts of the transcripts to extract. For 'UTRonly' (default) only the 3' UTR regions are extracted, 'withORF' additionally extracts the coding regions, and 'exons' extracts all exons
onlyCanonical	passed to findSeedMatches
shadow	The size of the ribosomal shadow at the UTR starts
cores	The number of threads to use. Alternatively accepts a BiocParallelParam-class, as for instance produced by MulticoreParam.
maxLogKd	The maximum log_kd of sites to report
save.path	Optional, the path to which to save the results
...	Arguments passed to findSeedMatches

Value

A ‘GRanges‘ object

Examples

```
anno <- ScanMiRAnno("fake")
m <- runFullScan( annotation=anno )
m
```

ScanMiRAnno-class *ScanMiRAnno*

Description

ScanMiRAnno

Usage

```
ScanMiRAnno(
  species = NULL,
  genome = NULL,
  ensdb = NULL,
  models = NULL,
  scan = NULL,
  aggregated = NULL,
  version = NULL,
  addDBs = list(),
  ...
)
```

Arguments

species	The species/build acronym for automatic construction; if omitted, ‘genome‘ and ‘ensdb‘ should be given. Current possible values are: GRCh38, GRCm38, GRCm39, Rnor_6.
genome	A BSgenome-class, or a TwoBitFile
ensdb	An EnsDb-class (or a TxDb-class) object
models	An optional KdModelList
scan	An optional full scan (IndexedFst or GRanges)
aggregated	An optional per-transcript aggregation (IndexedFst or data.frame)
version	optional ensembl version
addDBs	A named list of additional tx-miRNA databases, each of which should be a data.frame with the columns ’transcript’, ’miRNA’, and ’score’.
...	Arguments passed to ‘AnnotationHub‘

Value

A 'ScanMiRAnno' object

Examples

```
anno <- ScanMiRAnno(species="fake")
anno
```

`ScanMiRAnno-methods` *Methods for the* `ScanMiRAnno` *class*

Description

Methods for the `ScanMiRAnno` class

Usage

```
## S4 method for signature 'ScanMiRAnno'
summary(object)

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

Arguments

`object`	An object of class `ScanMiRAnno`

Value

Depends on the method.

See Also

`ScanMiRAnno`

scanMiRApp *scanMiRApp A wrapper for launching the scanMiRApp shiny app*

Description

scanMiRApp A wrapper for launching the scanMiRApp shiny app

Usage

```
scanMiRApp(annotations = NULL, ...)
```

Arguments

annotations	A named list of ScanMiRAnno objects. If omitted, will use the base ones.
...	Passed to scanMiRserver

Value

A shiny app

Examples

```
if(interactive()){
  anno <- ScanMiRAnno("fake")
  scanMiRApp(list(fakeAnno=anno))
}
```

scanMiRserver *scanMiRserver*

Description

Server function for the scanMiR shiny app. Most users are expected to use scanMiRApp instead.

Usage

```
scanMiRserver(
  annotations = list(),
  modlists = NULL,
  maxCacheSize = 10 * 10^6,
  BP = SerialParam()
)
```

Arguments

annotations	A named list of ScanMiRAnno object.
modlists	A named list of 'KdModelList' objects. If omitted, will fetch it from the annotation objects.
maxCacheSize	Maximum cache size in bytes.
BP	BPPARAM for multithreading

Value

A shiny server function

Examples

```
# we'd normally fetch a real annotation:
# anno <- ScanMiRAnno("Rnor_6")
# here we'll use a fake one:
anno <- ScanMiRAnno("fake")
srv <- scanMiRserver(list(fake=anno))
```

---

scanMiRui *scanMiRui*

Description

UI for the scanMiR app.

Usage

```
scanMiRui()
```

Value

A shiny ui

Examples

```
ui <- scanMiRui()
```

Index