Package 'Ularcirc'

November 21, 2024

Type Package

Title Shiny app for canonical and back splicing analysis (i.e. circular and mRNA analysis)

Version 1.25.0

Description Ularcirc reads in STAR aligned splice junction files and provides visualisation and analysis tools for splicing analysis. Users can assess backsplice junctions and forward canonical junctions.

biocViews DataRepresentation,Visualization, Genetics, Sequencing, Annotation, Coverage, AlternativeSplicing, DifferentialSplicing

License file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports AnnotationHub, AnnotationDbi, BiocGenerics, Biostrings, BSgenome, data.table (>= 1.9.4), DT, GenomicFeatures, GenomeInfoDb, GenomeInfoDbData, GenomicAlignments, GenomicRanges, ggplot2, ggrepel, gsubfn, mirbase.db, moments, Organism.dplyr, plotgardener, R.utils, S4Vectors, shiny, shinydashboard, shinyFiles, shinyjs, yaml

RoxygenNote 7.1.2

Suggests BSgenome.Hsapiens.UCSC.hg38, BiocStyle, httpuv, knitr, org.Hs.eg.db, rmarkdown, TxDb.Hsapiens.UCSC.hg38.knownGene

VignetteBuilder knitr

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

git_branch devel

git_last_commit a6da876

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-20

Author David Humphreys [aut, cre]

Maintainer David Humphreys <d.humphreys@victorchang.edu.au>

Contents

`BSJ_details` *BSJ_details This function returns details of a BSJ string and returns a list of coordinates. Can accept two different formats, Ularcirc or generic.*

Description

BSJ_details This function returns details of a BSJ string and returns a list of coordinates. Can accept two different formats, Ularcirc or generic.

Usage

```
BSJ_details(BSJ)
```

Arguments

`BSJ`	: backsplice junction as a string. See details below for example formats

Examples

```
bsj <-  'chr14_99465814_chr14_99458278'   # Historic Ularcirc format

bsj <- c("chr14_99465814_chr14_99458278","chr22_20933778_chr22_20934245",
         "chr12_120155720_chr12_120154969", "chr4_143543508_chr4_143543973",
         "chr10_7285955_chr10_7276891")

BSJ_details(bsj)

bsj <- 'chr10:100923974-100926020:+'     # generic format
```

```
BSJ_details(bsj)
```

---

bsj_fastq_generate *bsj_fastq_generate*

Description

Takes a circRNA predicted sequence and generates synthetic short sequence reads

Usage

```
bsj_fastq_generate(
  circRNA_Sequence,
  fragmentLength = 300,
  readLength = 100,
  variations = 4,
  headerID = ""
)
```

Arguments

circRNA_Sequence	: Linear sequence of a circRNA. i.e. the backspice junction is the first and last base of this sequence
fragmentLength	: Is the length the library fragment
readLength	: The sequence read length
variations	: Number of sequences returned for each read type. Note each sequence variation will start at a unique location (where possible)
headerID	: Character identifier that will be incorporated into sequence header

Value

Returns a list of two DNAstring sets labelled "read1" and "read2" which correspond to forward and reverse read pairs.

Examples

```
library('Ularcirc')


# Generate a 500nt sequence containing A and which is flanked with GG and CC.
circRNA_Sequence <- paste(rep('A',500),collapse='')
circRNA_Sequence <- paste('GG',circRNA_Sequence, 'CC', sep='')
# The GG and CC ends of sequence represent ends of linear exons that are circularised.
# Therefore the backsplice junction (BSJ) is GGCC.
# Generate reads that alternate over this BSJ
```

```
fastqReads <- bsj_fastq_generate(circRNA_Sequence, fragmentLength=300, readLength=100,
              variations = 4,   # Four type I , II, III, and IV reads generated
           headerID='circRNA_example')  # Identifier incorporated in name of each sequence
# The following will indicate 12 sequences are present in each list entry
length(fastqReads$read1)
length(fastqReads$read2)

# Can create fastq file as follows
Biostrings::writeXStringSet( fastqReads$read1,"circRNA_Sample_R1.fastq.gz",
                             compress = TRUE, format="fastq")
Biostrings::writeXStringSet( fastqReads$read2,"circRNA_Sample_R2.fastq.gz",
                             compress = TRUE, format="fastq")
```

---

bsj_to_circRNA_sequence

bsj_to_circRNA_sequence

Description

Takes one BSJ coordinate and generates a predicted circular RNA sequence.

Usage

```
bsj_to_circRNA_sequence(
  BSJ,
  geneID = NULL,
  genome,
  TxDb,
  annotationLibrary,
  reduce_candidates = TRUE,
  shiny = FALSE
)
```

Arguments

BSJ	: BSJ coordinate in the format of chr_coordinate_chr_coorindate OR chr:coordinate-coorindate:strand.
geneID	: The gene ID that the BSJ aligns to. Not essential as this can be identified from the BSJ coordinate, however time performance of function improved if this information can be provided.
genome	: Is the length f the library fragment
TxDb	: The sequence read length
annotationLibrary	: annotation database. See details for example.

reduce_candidates
: IF multiple exon entries align to a single BSJ then either return longest entry (TRUE) or all entries (FALSE)

shiny
: If TRUE then will setup shiny progress bars. Default is FALSE where a standard text progress bar is used.

Details

Backsplice junction coordinates are typically reported as a character string. Two formats are recognised, ":" delimited (eg circExplorer, CIRI) or "_" delimited (Ularcirc). The BSJ genomic coordinates are compared against the supplied gene model and exonic sequences from matching splice junctions are concatenated. This means the BSJ is the first and last nucleotide of the returned sequence. The current implementation will automatically check 0 or 1 base coordinates and any match is returned.

In some cases one BSJ will match multiple exon combinations. The default setting is to return the longest sequence. Alternatively all possibilities can be returned by setting reduce_candidates to FALSE. BSJ candidates that align to multiple exon combinations are added to duplicated list.

BSJ that do not align to any canonical junctions are returned as failed.

Value

Returns a DNAstring object.

Examples

```
library('Ularcirc')
TxDb <- TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene
genome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
annotationLibrary <- org.Hs.eg.db::org.Hs.eg.db

# Define BSJ. Following two formats are accepted
BSJ <- 'chr2:40430305-40428472:-'       # SLC8A1
BSJ  <- 'chr2_40430305_chr2_40428472'   # SLC8A1

circRNA_sequence <- bsj_to_circRNA_sequence(BSJ, "SLC8A1", genome,TxDb, annotationLibrary)

# You can also retrieve sequence without passing gene annotation - but this is slower
# circRNA_sequence <- bsj_to_circRNA_sequence(BSJ, NULL, genome,TxDb, annotationLibrary)

TxDb <- TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene
genome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
# EXAMPLE1 (3 fail and 2 will produce sequences)
BSJ <- c("chr14_99465814_chr14_99458278","chr22_20933778_chr22_20934245",
         "chr12_120155720_chr12_120154969", "chr4_143543508_chr4_143543973",
         "chr10_7285955_chr10_7276891")
GeneIDs <- c("SMARCA5","MSLN","RNF138","KIAA0368","CRKL")
circRNA_sequence <- bsj_to_circRNA_sequence(BSJ, GeneIDs, genome,TxDb, annotationLibrary)

# Returns a list with three items:
# (1) "identified" is a list of DNA strings from BSJ that aligned to FSJ coordinates of the gene model
# (2) "failed" is a character object of BSJ that did not align to FSJ coordinates of gene model. Each entry is
```

```
# named with gene ID.
# (3) "duplicates" (not implemented yet) identifies which BSJ returned multiple sequences
```

`chimericFilters` *chimericFilters*

Description

A wrapper function that prepares a list of filters that can be passed

Usage

```
chimericFilters(
  BSjuncName = NULL,
  sortDir = "Descending",
  indexNumber = 1,
  displayNumber = 10,
  displayRADscore = FALSE,
  RADcountThreshold = 10,
  applyFSJfilter = FALSE
)
```

Arguments

`BSjuncName`	: A character string that represents a backsplice junction ID. Set when needing to extract a specific junction. Default NULL.
`sortDir`	: Specifies how data is sorted, either "Descending" (default) or "Ascending".
`indexNumber`	: Filter data according to this file index
`displayNumber`	: Number of records to display in an shiny app
`displayRADscore`	: Boolean. If TRUE then will apply/calculate RAD score
`RADcountThreshold`	: Integer. The minimum count threshold required to calculate RAD score. i.e. A default RAD score of -1 will be applied to any BSJ with a count less than this score
`applyFSJfilter`	: Boolean of whether to apply FSJ filter

chimericStats *chimericStats*

Description

Simple function that returns a list of basic stats obtained from a STAR chimeric file

Usage

```
chimericStats(chimericDT)
```

Arguments

chimericDT : Data table of chimeric junctions as provided by STAR aligner

See Also

FilterChimericJuncs

Examples

```
extdata_path <- system.file("extdata",package = "Ularcirc")
chimeric.file <- paste0(extdata_path,"/SRR444655_subset.Chimeric.out.junction.gz")
chimericDT <- Ularcirc::loadSTAR_chimeric(chimeric.file,returnColIdx = 1:14)
Ularcirc::chimericStats(chimericDT$data_set)
chimericDT$filtered <- Ularcirc::FilterChimericJuncs(chimericDT$data_set, canonicalJuncs = TRUE)
Ularcirc::chimericStats(chimericDT$filtered)
```

Compatible_Annotation_DBs *Compatible_Annotation_DBs*

Description

Interogates Bioconductor databases and identifies those that are compatible with Ularcirc. Builds a list of commands that the user can copy to install the required database on their local computer. Once installed the databases are immediately available to Ularcirc upon re-starting the shiny app. This function requires connection to the internet.

Usage

```
Compatible_Annotation_DBs(search_term = "")
```

Arguments

search_term : character string of a full or part name of a database. Will return only those entries that contain this search term. Not case sensitive.

Value

Returns a list of compatible annotation databases

Examples

```
# Get all Bioconductor annotation databases that are compatible with Ularcirc
library('BSgenome')
library('httpuv')
library('AnnotationHub')
# Prepare a dataframe of all compatible annotation databases
## Not run:  compatible_DBs_human <- Compatible_Annotation_DBs("Hsapiens")

# Example of how to find a relevant database and load the relevant databases:
# This example find hg38 databases
   idx <- grep(pattern="hg38", x= compatible_DBs_human[,"genome"])

   if (!requireNamespace("BiocManager", quietly=TRUE))
       install.packages("BiocManager")
   BiocManager::install(c(compatible_DBs[idx,]))

## End(Not run)
```

FilterChimericJuncs *FilterChimericJuncs*

Description

A generic function that filters STAR chimeric junction files on certain genomic criteria (eg strand, same chromosome etc). Useful filter to remove the most obvious false positives. The default filter settings are suitable for circRNA discovery in humans / mice data sets.

Usage

```
FilterChimericJuncs(
  All_junctions,
  chromFilter = TRUE,
  strandFilter = TRUE,
  genomicDistance = c(200, 1e+05),
  canonicalJuncs = TRUE,
  fileID = c(-1),
  chrM_Filter = TRUE,
  invertReads = FALSE
)
```

Arguments

All_junctions	: data.table of chimeric reads from STAR aligner
chromFilter	: when TRUE (default) both chimera parts have to align to same chromosome
strandFilter	: when TRUE (default) both chimera parts have to align to same strand
genomicDistance	: minimum and maximum distance filters of chimeric reads on chromosome. Only is applied if ChromFilter is TRUE and StrandFilter is TRUE
canonicalJuncs	: Will include any canonical junctions (default TRUE). Note STAR keeps canonical junctions that do not conform to aligner rules.
fileID	: Specify a file index. Useful if planing to concatenating all data sets into a single table.
chrM_Filter	: Filter out mitochondrial chimeric reads (default TRUE)
invertReads	: Boolean that specifies in read strand should be inverted (default FALSE).

See Also

SelectUniqueJunctions, loadSTAR_chimeric

Examples

```
extdata_path <- system.file("extdata",package = "Ularcirc")
chimeric.file <- paste0(extdata_path,"/SRR444655_subset.Chimeric.out.junction.gz")
chimericsDT <- Ularcirc::loadSTAR_chimeric(chimeric.file,returnColIdx = 1:14)
chimericsDT$filtered <- Ularcirc::FilterChimericJuncs(chimericsDT$data_set, canonicalJuncs = TRUE)
```

FilterChimeric_Ularcirc

Wrapper function for Ularcirc shiny app which expects a list of objects to be returned

---

Description

NEED to ensure that unstranded boolean value is passed to this function Not tested via shiny app yet.

Usage

```
FilterChimeric_Ularcirc(
  All_junctions,
  chromFilter = TRUE,
  strandFilter = TRUE,
  genomicDistance = c(200, 1e+05),
  canonicalJuncs = TRUE,
  fileID = c(-1),
```

```
  chrM_Filter = TRUE,
  invertReads = FALSE,
  unstranded = FALSE,
  summaryNumber = 50
)
```

Arguments

All_junctions	: data.table of chimeric reads from STAR aligner
chromFilter	: when TRUE (default) both chimera parts have to align to same chromosome
strandFilter	: when TRUE (default) both chimera parts have to align to same strand
genomicDistance	: minimum and maximum distance filters of chimeric reads on chromosome. Only is applied if ChromFilter is TRUE and StrandFilter is TRUE
canonicalJuncs	: Will include any canonical junctions (default TRUE). Note STAR keeps canonical junctions that do not conform to aligner rules.
fileID	: Specify a file index. Useful if planing to concatenating all data sets into a single table.
chrM_Filter	: Filter out mitochondrial chimeric reads (default TRUE)
invertReads	: Boolean that specifies in read strand should be inverted (default FALSE).
unstranded	: Boolen for if reads are unstranded
summaryNumber	: Number (Integer) of records to display in shiny app

---

Junction_Sequence_from_Genome
Grab_BS_Junc_Sequence

Description

This function extracts genomic sequence that is likely to capture BSJ. Function does not cross validate to gene models.

Usage

```
Junction_Sequence_from_Genome(SelectUniqueJunct_Value, GeneList)
```

Arguments

GeneList	: GeneList
SelectUniqueJunct_value	: a dataframe with columns names startDonor, strandDonor, startAcceptor

`loadSTAR_chimeric` *loadSTAR_chimeric*

Description

Loads chimeric output file from the STAR aligner and returns a list containing three items (a data table, alignment stats and command line).

Usage

```
loadSTAR_chimeric(filename = NULL, ID_index = 0, returnColIdx = 1:21)
```

Arguments

`filename`	: filename of the STAR chimeric output file. Can be gzipped
`ID_index`	: An index (single integer) which will be added as a separate column in the returned data table. Useful when collating multiple files into one large matrix like object.
`returnColIdx`	: Numeric index of columns to return. Default 1:15

Details

:

Reads in a text or gzipped chimeric output file generated by the STAR aligner. Function automatically detects if the last two lines contains meta-data (produced from STAR 2.7) onwards.

Returns a list of containing three items: (1) data_set (2) alignmentStats and (3) commandLine.

The column names of data_set are defined as c("chromDonor","startDonor","strandDonor", "chromAcceptor", "startAcceptor","strandAcceptor","JuncType", "RepeatLength_L", "RepeatLength_R", "ReadName","FirstBase_1stSeq","CIGAR_1stSeg", "FirstBase_2ndSeq","CIGAR_2ndSeg", "Multimapping")

If ID_index is set to a value greater than 0 then an additional column called "DataSet" is created.

Columns can be subsetted by defining returnColIdx with an integer value that correspond to order of column names listed above.

`plot_AllJunctions` *plot_AllJunctions*

Description

Plots a BSJ, FSJ and transcripts for a nominated gene. Output is combined onto a single page. This function effectively wraps plotting functions from plotgardener

Usage

```
plot_AllJunctions(
  assembly = "hg38",
  chrom,
  chromstart,
  chromend,
  BSJData,
  BSJ_colors = "black",
  FSJData,
  FSJ_colors = "black",
  geneSymbol
)
```

Arguments

assembly	: Genome assembly
chrom	: chromosome
chromstart	: Starting position of chromosome
chromend	: End position of chromosome
BSJData	: Backsplice junction data table
BSJ_colors	: Backsplice junction assigned colours
FSJData	: Forward junction data table
FSJ_colors	: Forward junction assigned colours
geneSymbol	: Gene symbol

Value

Returns a list of two DNAstring sets labelled "read1" and "read2" which correspond to forward and reverse read pairs.

Examples

```
library('Ularcirc')
 # BSJ data.table
 BSJ_data <- data.table::data.table(chrom1="chr2",
                start1=c(40139400, 40160764, 40428472, 40428472),
                end1=c(40139400, 40160764,40428472, 40428472),
                chrom2="chr2", start2=c(40178494,40178494,40430302,40430305),
                 end2=c(40178494,40178494,40430302,40430305),
               score=c(13,20,360,1751))

 # FSJ
 FSJstarts1 <- c(40115630,40139677,40160865,40164985,40170350,40174721,
                              40174843,40175282,40278771,40430302,40430305)
 FSJstarts2 <- c(40139400,40160764,40164853,40170280,40174705,40174824,
                               40175260,40178386,40428472,40453160,40512348)
 FSJ_data <- data.table::data.table(chrom1="chr2", start1=FSJstarts1, end1=FSJstarts1,
```

```
                    chrom2="chr2", start2=FSJstarts2, end2=FSJstarts2,
                    score=c(225,825,685,666,633,596,517,542,685,101,171))

plot_AllJunctions(assembly="hg38", chrom="chr2",
                    chromstart=40096769, chromend=40611554,
                    BSJData=BSJ_data, FSJData=FSJ_data, geneSymbol="SLC8A1")
```

`RAD_score` *RAD_score*

Description

Theoretically the position of backsplice junctions should be distributed randomly across a amplicon. This function calculates the read alignment distribution (RAD) of backsplice junctions between forward and reverse read pairs. The RAD score is calculated from CIGAR strings which can be used to identify type II and type III alignments.

Usage

```
RAD_score(
  CIGAR_1stSeg = NULL,
  CIGAR_2ndSeg = NULL,
  RADcountThreshold = 10,
  digits = 2
)
```

Arguments

`CIGAR_1stSeg`	: CIGAR string of the first segment.
`CIGAR_2ndSeg`	: CIGAR string of the second segment
`RADcountThreshold`	: Minimum count threshold required to apply RAD score. If there are less than this many entries in CIGAR list then -1 is returned.
`digits`	: rounding of the RAD score to this many digits (default 2)

`SelectUniqueJunctions` *SelectUniqueJunctions*

Description

Builds a summary table from chimeric data obtained from the STAR aligner. Assembles table with the requested number of top entries. Populates with RAD score and FSJ score.

Usage

```
SelectUniqueJunctions(
  BSJ_junctions,
  filterlist = chimericFilters(),
  unstranded = FALSE,
  FSJ_Junctions = NULL,
  shinyapp = FALSE
)
```

Arguments

BSJ_junctions	: Junction to display
filterlist	: filterlist
unstranded	: If TRUE will match reads from both strands.
FSJ_Junctions	: Junction to display.
shinyapp	: Boolean. If true used to setup control status bars in shiny app.

Details

This is the workhorse for collated BSJ junctions from the input data. It will return selected rows of data (annotated) that will enable enhanced browsing of raw data on the fly.

Filter options: Junction abundance. Sort

---

sequence_from_exon_coords

sequence_from_exon_coords

Description

sequence_from_exon_coords

Usage

```
sequence_from_exon_coords(genome, exon_df)
```

Arguments

genome	: genome object
exon_df	: data frame of exons. Must have column with names "chrom", "start", "stop", "strand"

Ularcirc *Ularcirc*

Description

When the function is invoked the Ularcirc shiny app is started. The starting screen has quickstart instructions on how to use the software. Please refer to the Ularcirc vignette for a more detailed workflow.

Usage

```
Ularcirc()
```

Value

Does not return anything

Examples

```
# The following commands will load the shiny app either through an RStudio session or
# through your internet browser

library("Ularcirc")
## Not run:  Ularcirc()
```

Index