Package: circRNAprofiler (via r-universe)

October 19, 2026

Type Package

Title circRNAprofiler: An R-Based Computational Framework for the Downstream Analysis of Circular RNAs

Version 1.26.0

Author Simona Aufiero

Maintainer Simona Aufiero <simo.aufiero@gmail.com>

Description R-based computational framework for a comprehensive in silico analysis of circRNAs. This computational framework allows to combine and analyze circRNAs previously detected by multiple publicly available annotation-based circRNA detection tools. It covers different aspects of circRNAs analysis from differential expression analysis, evolutionary conservation, biogenesis to functional analysis.

License GPL-3

Encoding UTF-8

biocViews Annotation, StructuralPrediction, FunctionalPrediction, GenePrediction, GenomeAssembly, DifferentialExpression

Depends R(>= 4.5.0)

RoxygenNote 7.2.3

Suggests testthat, knitr, roxygen2, rmarkdown, devtools, gridExtra, ggpubr, VennDiagram, BSgenome.Mmusculus.UCSC.mm9, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, BiocManager,

Imports dplyr, magrittr, readr, rtracklayer, stringr, stringi, DESeq2, edgeR, GenomicRanges, IRanges, seqinr, R.utils, reshape2, ggplot2, utils, rlang, S4Vectors, stats, GenomeInfoDb, universalmotif, AnnotationHub, BSgenome.Hsapiens.UCSC.hg19, Biostrings, gwascat, BSgenome,

VignetteBuilder knitr

URL https://github.com/Aufiero/circRNAprofiler

BugReports https://github.com/Aufiero/circRNAprofiler/issues

Config/pak/sysreqs make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:50:32 UTC

RemoteUrl https://github.com/bioc/circRNAprofiler

RemoteRef RELEASE_3_23

RemoteSha ec74a823e2299788faeb0a3ed30bc28c51abdfc8

Contents

ahChainFiles . . . 3
ahRepeatMasker . . . 3
annotateBSJs . . . 4
annotateRepeats . . . 5
annotateSNPsGWAS . . . 6
attractSpecies . . . 8
backSplicedJunctions . . . 8
checkProjectFolder . . . 9
filterCirc . . . 11
formatGTF . . . 12
getBackSplicedJunctions . . . 13
getCircSeqs . . . 14
getDeseqRes . . . 15
getDetectionTools . . . 17
getEdgerRes . . . 17
getMiRsites . . . 19
getMotifs . . . 21
getRandomBSJunctions . . . 23
getRegexPattern . . . 24
getSeqsAcrossBSJs . . . 25
getSeqsFromGRs . . . 26
gtf . . . 27
gwasTraits . . . 27
initCircRNAprofiler . . . 28
iupac . . . 29
liftBSJcoords . . . 29
memeDB . . . 30
mergeBSJunctions . . . 31
mergedBSJunctions . . . 32
mergeMotifs . . . 33
miRspeciesCodes . . . 34
plotExBetweenBSEs . . . 35
plotExPosition . . . 36
plotHostGenes . . . 37
plotLenBSEs . . . 37
plotLenIntrons . . . 39

plotMiR 40
plotMotifs 42
plotTotExons 44
rearrangeMiRres 45
volcanoPlot 46

Index **49**

`ahChainFiles` *ahChainFile*

Description

A data frame containing the AnnotationHub id related to the *.chain.gz file, specific for each species and genome. The file name reflects the assembly conversion data contained within in the format <db1>To<Db2>.over.chain.gz. For example, a file named mm10ToHg19.over.chain.gz file contains the liftOver data needed to convert mm10 (Mouse GRCm38) coordinates to hg19 (Human GRCh37).

Usage

```
data(ahChainFiles)
```

Format

A data frame with 1113 rows and 2 columns

id is the AnnotationHub id

chainFile is the *.chain.gz file

Examples

```
data(ahChainFiles)
```

`ahRepeatMasker` *ahRepeatMasker*

Description

A data frame containing the AnnotationHub id related to the repeatMasker db, specific for each species and genome.

Usage

```
data(ahRepeatMasker)
```

Format

A data frame with 84 rows and 3 columns

id is the AnnotationHub id

species is the species

genome is the genome assembly

Examples

```
data(ahRepeatMasker)
```

`annotateBSJs` *Annotate circRNA features.*

Description

The function annotateBSJs() annotates the circRNA structure and the introns flanking the corresponding back-spliced junctions. The genomic features are extracted from the user provided gene annotation.

Usage

```
annotateBSJs(
  backSplicedJunctions,
  gtf,
  isRandom = FALSE,
  pathToTranscripts = NULL
)
```

Arguments

`backSplicedJunctions`
: A data frame containing the back-spliced junction coordinates (e.g. detected or randomly selected). For detected back-spliced junctions see `getBackSplicedJunctions` and `mergeBSJunctions` (to group circRNAs detected by multiple detection tools). For randomly selected back-spliced junctions see `getRandomBSJunctions`.

`gtf`
: A data frame containing genome annotation information. It can be generated with `formatGTF`.

`isRandom`
: A logical indicating whether the back-spliced junctions have been randomly generated with `getRandomBSJunctions`. Deafult value is FALSE.

`pathToTranscripts`
: A string containing the path to the transcripts.txt file. The file transcripts.txt contains the transcript ids of the circRNA host gene to analyze. It must have one column with header id. By default pathToTranscripts is set to NULL and the file it is searched in the working directory. If transcripts.txt is located in

a different directory then the path needs to be specified. If this file is empty or absent the longest transcript of the circRNA host gene containing the back-spliced junctions are considered in the annotation analysis.

Value

A data frame.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)
```

---

annotateRepeats *Annotate repetitive elements*

Description

The function annotateRepeats() annotates repetitive elements located in the region flanking the back-spliced junctions of each circRNA. Repetitive elements are provided by AnnotationHub storage which collected repeats from RepeatMasker database. See AnnotationHub and http://www.repeatmasker.org for more details. An empty list is returned if none overlapping repeats are found.

Usage

```
annotateRepeats(targets, annotationHubID = "AH5122", complementary = TRUE)
```

Arguments

targets	A list containing the target regions to analyze. It can be generated with getSeqsFromGRs.
annotationHubID	A string specifying the AnnotationHub id to use. Type data(ahRepeatMasker) to see all possible options. E.g. if AH5122 is specified, repetitive elements from Homo sapiens, genome hg19 will be downloaded and annotated. Default value is "AH5122".
complementary	A logical specifying whether to filter and report only back-spliced junctions of circRNAs which flanking introns contain complementary repeats, that is, repeats belonging to a same family but located on opposite strands.

Value

A list.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){

genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve targets
targets <- getSeqsFromGRs(
    annotatedBSJs,
    genome,
    lIntron = 200,
    lExon = 10,
    type = "ie"
    )

# Annotate repeats

# repeats <- annotateRepeats(targets, annotationHubID  = "AH5122", complementary = TRUE)

}
```

---

annotateSNPsGWAS *Annotate GWAS SNPs*

Description

The function annotateSNPsGWAS() annotates GWAS SNPs located in the region flanking the back-spliced junctions of each circRNA. SNPs information including the corresponding genomic coordinates are retrieved from the GWAs catalog database. The user can restric the analysis to specific traits/diseases. These must go in the file traits.txt. If this file is absent or empty, all traits in the GWAS catalog are considered in the analysis. An empty list is returned if none overlapping SNPs are found.

Usage

```
annotateSNPsGWAS(
  targets,
  assembly = "hg19",
  makeCurrent = TRUE,
  pathToTraits = NULL
)
```

Arguments

targets	A list containing the target regions to analyze. It can be generated with getSeqsFromGRs.
assembly	A string specifying the human genome assembly. Possible options are hg38 or hg19. Current image for GWAS SNPS coordinates is hg38. If hg19 is specified SNPs coordinates are realtime liftOver to hg19 coordinates.
makeCurrent	A logical specifying whether to download the current image of the GWAS catalog. If TRUE is specified, the function makeCurrentGwascat from the gwascat package is used to get the more recent image (slow). Default value is TRUE. If FALSE is specified the image data(ebicat37) or data(ebicat38) are used. NOTE: This second option is not available anymore since data(ebicat37) or data(ebicat38) from gwascat are deprecated.
pathToTraits	A string containing the path to the traits.txt file. The file traits.txt contains diseases/traits specified by the user. It must have one column with header id. By default pathToTraits is set to NULL and the file it is searched in the working directory. If traits.txt is located in a different directory then the path needs to be specified. If this file is absent or empty SNPs associated with all diseases/traits in the GWAS catalog are considered in the SNPs analysis.

Value

A list.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve targets
targets <- getSeqsFromGRs(
    annotatedBSJs,
```

```
    genome,
    lIntron = 200,
    lExon = 10,
    type = "ie"
    )

# Annotate GWAS SNPs - slow

#snpsGWAS <- annotateSNPsGWAS(targets, makeCurrent = TRUE, assembly = "hg19")

}
```

`attractSpecies` *attractSpecies*

Description

A data frame containing the species for which RBP motifs are available in ATtRACT db. See also http://attract.cnic.es.

Usage

```
data(attractSpecies)
```

Format

A data frame with 37 rows and 1 columns

species is the species

Examples

```
data(attractSpecies)
```

`backSplicedJunctions` *backSplicedJunctions*

Description

A data frame containing genomic coordinates (genome assembly hg19) of circRNAs detected by three detection tools (CircMarker(cm), MapSplice2 (m) and NCLscan (n)) in the human left ventricle tissues of 3 controls, 3 patients with dilated cardiomyopathies (DCM) and 3 patients with hypertrophic cardiomyopathies (HCM). This data frame was generated with `getBackSplicedJunctions`.

Usage

```
data(backSplicedJunctions)
```

Format

A data frame with 63521 rows and 16 columns

id Unique identifier

gene is the gene name whose exon coordinates overlap that of the given back-spliced junctions

strand is the strand where the gene is transcribed

chrom the chromosome from which the circRNA is derived

startUpBSE is the 5' coordinate of the upstream back-spliced exon in the transcript. This corresponds to the back-spliced junction / acceptor site

endDownBSE is the 3' coordinate of the downstream back-spliced exon in the transcript. This corresponds to the back-spliced junction / donor site

tool are the tools that identified the back-spliced junctions

C1 Number of occurences of each circRNA in control 1

C2 Number of occurences of each circRNA in control 2

C3 Number of occurences of each circRNA in control 3

D1 Number of occurences of each circRNA in DCM 1

D2 Number of occurences of each circRNA in DCM 2

D3 Number of occurences of each circRNA in DCM 3

H1 Number of occurences of each circRNA in HCM 1

H2 Number of occurences of each circRNA in HCM 2

H3 Number of occurences of each circRNA in HCM 3

Examples

```
data(backSplicedJunctions)
```

---

checkProjectFolder *Check project folder*

Description

The function checkProjectFolder() verifies that the project folder is set up correctly. It checks that the mandatory files (.gtf file, the folders with the circRNAs_X.txt files and experiemnt.txt) are present in the working directory.The function `initCircRNAprofiler` can be used to initialize the project folder.

Usage

```
checkProjectFolder(
  pathToExperiment = NULL,
  pathToGTF = NULL,
  pathToMotifs = NULL,
  pathToMiRs = NULL,
  pathToTranscripts = NULL,
  pathToTraits = NULL
)
```

Arguments

`pathToExperiment`
: A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers: - label (1st column): unique names of the samples (short but informative). - fileName (2nd column): name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc. - group (3rd column): biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions. By default pathToExperiment is set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

`pathToGTF`
: A string containing the path to the the GTF file. Use the same annotation file used during the RNA-seq mapping procedure. By default pathToGTF is set to NULL and the file it is searched in the working directory. If .gtf is located in a different directory then the path needs to be specified.

`pathToMotifs`
: A string containing the path to the motifs.txt file. The file motifs.txt contains motifs/regular expressions specified by the user. It must have 3 columns with headers: - id (1st column): name of the motif. - e.g. RBM20 or motif1. - motif (2nd column): motif/pattern to search. - length (3rd column): length of the motif. By default pathToMotifs is set to NULL and the file it is searched in the working directory. If motifs.txt is located in a different directory then the path needs to be specified. If this file is absent or empty only the motifs of RNA Binding Proteins in the ATtRACT or MEME database are considered in the motifs analysis.

`pathToMiRs`
: A string containing the path to the miRs.txt file. The file miRs.txt contains the microRNA ids from miRBase specified by the user. It must have one column with header id. The first row must contain the miR name starting with the ">", e.g >hsa-miR-1-3p. The sequences of the miRs will be automatically retrieved from the mirBase latest release or from the given mature.fa file, that should be present in the working directory. By default pathToMiRs is set to NULL and the file it is searched in the working directory. If miRs.txt is located in a different directory then the path needs to be specified. If this file is absent or empty, all miRs of the species specified in input are considered in the miRNA analysis.

`pathToTranscripts`
: A string containing the path to the transcripts.txt file. The file transcripts.txt contains the transcript ids of the circRNA host gene to analyze. It must have

one column with header id. By default pathToTranscripts is set to NULL and the file it is searched in the working directory. If transcripts.txt is located in a different directory then the path needs to be specified. If this file is empty or absent the longest transcript of the circRNA host gene containing the back-spliced junctions are considered in the annotation analysis.

`pathToTraits` A string containing the path to the traits.txt file. contains diseases/traits specified by the user. It must have one column with header id. By default pathToTraits is set to NULL and the file it is searched in the working directory. If traits.txt is located in a different directory then the path needs to be specified. If this file is absent or empty SNPs associated with all diseases/traits in the GWAS catalog are considered in the SNPs analysis.

Value

An integer. If equals to 0 the project folder is correctly set up.

Examples

```
checkProjectFolder()
```

`filterCirc` *Filter circRNAs*

Description

The functions filterCirc() filters circRNAs on different criteria: condition and read counts. The info reported in experiment.txt file are needed for filtering step.

Usage

```
filterCirc(
  backSplicedJunctions,
  allSamples = FALSE,
  min = 3,
  pathToExperiment = NULL
)
```

Arguments

`backSplicedJunctions`
A data frame containing back-spliced junction coordinates and counts. See `getBackSplicedJunctions` and `mergeBSJunctions` (to group circRNA detected by multiple detection tools) on how to generated this data frame.

`allSamples` A string specifying whether to apply the filter to all samples. Default valu is FALSE.

`min` An integer specifying the read counts cut-off. If allSamples = TRUE and min = 0 all circRNAs are kept. If allSamples = TRUE and min = 3, a circRNA is kept if all samples have at least 3 counts. If allSamples = FALSE and min = 2 the filter is applied to the samples of each condition separately meaning that a circRNA is kept if at least 2 counts are present in all sample of 1 of the conditions. Default value is 3.

`pathToExperiment`

A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers:

label: (1st column) - unique names of the samples (short but informative).

fileName: (2nd column) - name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc.

group: (3rd column) - biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions.

By default pathToExperiment is set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

Value

A data frame.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

pathToExperiment <- system.file("extdata", "experiment.txt",
    package ="circRNAprofiler")

# Filter circRNAs
filteredCirc <- filterCirc(
    mergedBSJunctions,
    allSamples = FALSE,
    min = 5,
    pathToExperiment)
```

---

formatGTF *Format annotation file*

Description

The function formatGTF() formats the given annotation file.

Usage

```
formatGTF(pathToGTF = NULL)
```

Arguments

pathToGTF	A string containing the path to the GTF file. Use the same annotation file used during the RNA-seq mapping procedure. If .gtf file is not present in the current working directory the full path should be specified.

Value

A data frame.

Examples

```
gtf <- formatGTF()
```

---

getBackSplicedJunctions

Import detected circRNAs

Description

The function getBackSplicedJunctions() reads the circRNAs_X.txt with the detected circRNAs, adapts the content and generates a unique data frame with all circRNAs identified by each circRNA detection tool and the occurrences found in each sample (named as reported in the column label in experiment.txt).

Usage

```
getBackSplicedJunctions(gtf, pathToExperiment = NULL)
```

Arguments

gtf	A data frame containing the annotation information. It can be generated with formatGTF.
pathToExperiment	A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers: **label:** (1st column) - unique names of the samples (short but informative). **fileName:** (2nd column) - name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc. **group:** (3rd column) - biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions. By default pathToExperiment is set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

Value

A data frame.

See Also

backSplicedJunctions for a description of the data frame containing back-spliced junctions coordinates.

Examples

```
check <- checkProjectFolder()

if(check == 0){
# Create gtf object
gtf <- formatGTF(pathToGTF)

# Read and adapt detected circRNAs
backSplicedJunctions<- getBackSplicedJunctions(gtf)}
```

---

getCircSeqs *Retrieve circRNA sequences*

Description

The function getCircSeqs() retrieves the circRNA sequences. The circRNA sequence is given by the sequences of the exons in between the back-spliced-junctions.The exon sequences are retrieved and then concatenated together to recreate the circRNA sequence. To recreate the back-spliced junction sequence 50 nucleotides are taken from the 5' head and attached at the 3' tail of each circRNA sequence.

Usage

```
getCircSeqs(annotatedBSJs, gtf, genome)
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. It can be generated with annotateBSJs.
gtf	A data frame containing genome annotation information. It can be generated with formatGTF.
genome	A BSgenome object containing the genome sequences. It can be generated with getBSgenome. See available.genomes to see the BSgenome package currently available

Value

A list.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){

genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences
targets <- getCircSeqs(
    annotatedBSJs,
    gtf,
    genome)

}
```

`getDeseqRes` *Differential circRNA expression analysis adapted from DESeq2*

Description

The helper functions getDeseqRes() identifies differentially expressed circRNAs. The latter uses respectively the R Bioconductor packages DESeq2 which implements a beta-binomial model to model changes in circRNA expression.

Usage

```
getDeseqRes(
  backSplicedJunctions,
  condition,
  pAdjustMethod = "BH",
  pathToExperiment = NULL,
  ...
)
```

Arguments

`backSplicedJunctions`
: A data frame containing the back-spliced junction coordinates and counts in each analyzed sample. See `getBackSplicedJunctions` and `mergeBSJunctions`

(to group circRNA detected by multiple detection tools) on how to generated this data frame.

`condition` A string specifying which conditions to compare. Only 2 conditions at the time can be analyzed. Separate the 2 conditions with a dash, e.g. A-B. Use the same name used in column condition in experiment.txt. log2FC calculation is perfomed by comparing the condition positioned forward against the condition positioned backward in the alphabet. E.g. if there are 2 conditions A and B then a negative log2FC means that in condition B there is a downregulation of the corresponding circRNA. If a positive log2FC is found means that there is an upregulation in condition B of that circRNA.

`pAdjustMethod` A character string stating the method used to adjust p-values for multiple testing. See `p.adjust`. Deafult value is "BH".

`pathToExperiment`

A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers:

label: (1st column) - unique names of the samples (short but informative).

fileName: (2nd column) - name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc.

group: (3rd column) - biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions.

By default pathToExperiment is set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

`...` Arguments to be passed to the DESeq function used internally from DESeq2 package. If nothing is specified the default values of the function `DESeq` are used.

Value

A data frame.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

pathToExperiment <- system.file("extdata", "experiment.txt",
    package ="circRNAprofiler")

# Filter circRNAs
filteredCirc <- filterCirc(
    mergedBSJunctions,
    allSamples = FALSE,
    min = 5,
    pathToExperiment)

# Find differentially expressed circRNAs
```

```
deseqResBvsA <- getDeseqRes(
    filteredCirc,
    condition = "A-B",
    pAdjustMethod = "BH",
    pathToExperiment)
```

`getDetectionTools` *Create data frame with circRNA detection codes*

Description

The function getDetectionTools() creates a data frame containing the codes corresponding to each circRNA detection tool for which a specific import function has been developed.

Usage

```
getDetectionTools()
```

Value

A data frame

Examples

```
getDetectionTools()
```

`getEdgerRes` *Differential circRNA expression analysis adapted from EdgeR*

Description

The helper functions edgerRes() identifies differentially expressed circRNAs. The latter uses respectively the R Bioconductor packages EdgeR which implements a beta-binomial model to model changes in circRNA expression. The info reported in experiment.txt file are needed for differential expression analysis.

Usage

```
getEdgerRes(
  backSplicedJunctions,
  condition,
  normMethod = "TMM",
  pAdjustMethod = "BH",
  pathToExperiment = NULL
)
```

Arguments

backSplicedJunctions

A data frame containing the back-spliced junction coordinates and counts in each analyzed sample. See `getBackSplicedJunctions` and `mergeBSJunctions` (to group circRNA detected by multiple detection tools) on how to generated this data frame.

condition

A string specifying which conditions to compare. Only 2 conditions at the time can be analyzed. Separate the 2 conditions with a dash, e.g. A-B. Use the same name used in column condition in experiment.txt. log2FC calculation is perfomed by comparing the condition positioned forward against the condition positioned backward in the alphabet. E.g. if there are 2 conditions A and B then a negative log2FC means that in condition B there is a downregulation of the corresponding circRNA. If a positive log2FC is found means that there is an upregulation in condition B of that circRNA.

normMethod

A character string specifying the normalization method to be used. It can be "TMM","RLE","upperquartile" or"none". The value given to the method argument is given to the `calcNormFactors` used internally. Deafult value is "TMM".

pAdjustMethod

A character string stating the method used to adjust p-values for multiple testing. See `p.adjust`. Deafult value is "BH".

pathToExperiment

A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers:

label: (1st column) - unique names of the samples (short but informative).

fileName: (2nd column) - name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc.

group: (3rd column) - biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions.

By default pathToExperiment is set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

Value

A data frame.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

pathToExperiment <- system.file("extdata", "experiment.txt",
    package ="circRNAprofiler")

# Filter circRNAs
filteredCirc <- filterCirc(
    mergedBSJunctions,
```

```
    allSamples = FALSE,
    min = 5,
    pathToExperiment)

# Find differentially expressed circRNAs
deseqResBvsA <- getEdgerRes(
    filteredCirc,
    condition = "A-B",
    normMethod = "TMM",
    pAdjustMethod = "BH",
    pathToExperiment)
```

`getMiRsites` *Screen target sequences for miR binding sites*

Description

The function getMirSites() searches miRNA binding sites within the circRNA sequences. The user can restrict the analisis only to a subset of miRs. In this case, miR ids must go in miR.txt file. If the latter is absent or empty, all miRs of the specified miRspeciesCode are considered in the analysis.

Usage

```
getMiRsites(
  targets,
  miRspeciesCode = "hsa",
  miRBaseLatestRelease = TRUE,
  totalMatches = 7,
  maxNonCanonicalMatches = 1,
  pathToMiRs = NULL
)
```

Arguments

`targets`	A list containing the target sequences to analyze. This data frame can be generated with `getCircSeqs`.
`miRspeciesCode`	A string specifying the species code (3 letters) as reported in miRBase db. E.g. to analyze the mouse microRNAs specify "mmu", to analyze the human microRNAs specify "hsa". Type data(miRspeciesCode) to see the available codes and the corresponding species reported in miRBase 22 release. Default value is "hsa".
`miRBaseLatestRelease`	A logical specifying whether to download the latest release of the mature sequences of the microRNAs from miRBase (http://www.mirbase.org/ftp.

shtml). If TRUE is specified then the latest release is automatically downloaded. If FALSE is specified then a file named mature.fa containing fasta format sequences of all mature miRNA sequences previously downloaded by the user from mirBase must be present in the working directory. Default value is TRUE.

`totalMatches` An integer specifying the total number of matches that have to be found between the seed region of the miR and the seed site of the target sequence. If the total number of matches found is less than the cut-off, the seed site is discarded. The maximun number of possible matches is 7. Default value is 7.

`maxNonCanonicalMatches`
An integer specifying the max number of non-canonical matches (G:U) allowed between the seed region of the miR and the seed site of the target sequence. If the max non-canonical matches found is greater than the cut-off, the seed site is discarded. Default value is 1.

`pathToMiRs` A string containing the path to the miRs.txt file. The file miRs.txt contains the microRNA ids from miRBase specified by the user. It must have one column with header id. The first row must contain the miR name starting with the ">", e.g >hsa-miR-1-3p. The sequences of the miRs will be automatically retrieved from the mirBase latest release or from the given mature.fa file, that should be present in the working directory. By default pathToMiRs is set to NULL and the file it is searched in the working directory. If miRs.txt is located in a different directory then the path needs to be specified. If this file is absent or empty, all miRs of the specified species are considered in the miRNA analysis.

Value

A list.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences.
targets <- getCircSeqs(
    annotatedBSJs,
    gtf,
    genome)

# Screen target sequence for miR binding sites.
```

```
pathToMiRs <- system.file("extdata", "miRs.txt", package="circRNAprofiler")

#miRsites <- getMiRsites(
#   targets,
#   miRspeciesCode = "hsa",
#   miRBaseLatestRelease = TRUE,
#   totalMatches = 6,
#   maxNonCanonicalMatches = 1,
#   pathToMiRs )
}
```

`getMotifs` *Screen target sequences for recurrent motifs*

Description

The function getMotifs() scans the target sequences for the presence of recurrent motifs of a specific length defined in input. By setting rbp equals to TRUE, the identified motifs are matched with motifs of known RNA Binding Proteins (RBPs) deposited in the ATtRACT (http://attract.cnic.es) or MEME database (http://meme-suite.org/) and with motifs specified by the user. The user motifs must go in the file motifs.txt. If this file is absent or empty, only motifs from the ATtRACT or MEME database are considered in the analysis. By setting rbp equals to FALSE, only motifs that do not match with any motifs deposited in the databases or user motifs are reported in the final output. Location of the selected motifs is also reported. This corresponds to the start position of the motif within the sequence (1-index based).

Usage

```
getMotifs(
  targets,
  width = 6,
  database = "ATtRACT",
  species = "Hsapiens",
  memeIndexFilePath = 18,
  rbp = TRUE,
  reverse = FALSE,
  pathToMotifs = NULL
)
```

Arguments

`targets`	A list containing the target sequences to analyze. It can be generated with `getCircSeqs`, `getSeqsAcrossBSJs` or `getSeqsFromGRs`.
`width`	An integer specifying the length of all possible motifs to extract from the target sequences. Default value is 6.

`database` A string specifying the RBP database to use. Possible options are ATtRACT or MEME. Default database is "ATtRACT".

`species` A string specifying the species of the ATtRACT RBP motifs to use. Type data(attractSpecies) to see the possible options. Default value is "Hsapiens".

`memeIndexFilePath`
An integer specifying the index of the file path of the meme file to use.Type data(memeDB) to see the possible options. Default value is 18 corresponding to the following file: motif_databases/RNA/Ray2013_rbp_Homo_sapiens.meme

`rbp` A logical specifying whether to report only motifs matching with known RBP motifs from ATtRACT database or user motifs specified in motifs.txt. If FALSE is specified only motifs that do not match with any of these motifs are reported. Default values is TRUE.

`reverse` A logical specifying whether to reverse the motifs collected from ATtRACT database and from motifs.txt. If TRUE is specified all the motifs are reversed and analyzed together with the direct motifs as they are reported in the ATtRACT db and motifs.txt. Default value is FALSE.

`pathToMotifs` A string containing the path to the motifs.txt file. The file motifs.txt contains motifs/regular expressions specified by the user. It must have 3 columns with headers:

id: (1st column) - name of the motif. - e.g. RBM20 or motif1).

motif: (2nd column) -motif/pattern to search.

length: (3rd column) - length of the motif.

By default pathToMotifs is set to NULL and the file it is searched in the working directory. If motifs.txt is located in a different directory then the path needs to be specified. If this file is absent or empty only the motifs of RNA Binding Proteins in the ATtRACT database are considered in the motifs analysis.

Value

A list.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Example with the first back-spliced junction
# Multiple back-spliced junctions can also be analyzed at the same time

# Annotate the first back-spliced junction
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){
```

```
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences
targets <- getSeqsFromGRs(
    annotatedBSJs,
    genome,
    lIntron = 200,
    lExon = 10,
    type = "ie"
    )

# Get motifs
motifs <- getMotifs(
    targets,
    width = 6,
    database = 'ATtRACT',
    species = "Hsapiens",
    rbp = TRUE,
    reverse = FALSE)

}
```

`getRandomBSJunctions` *Retrieve random back-spliced junctions*

Description

The function getRandomBSJunctions() retrieves random back-spliced junctions from the user genome annotation.

Usage

```
getRandomBSJunctions(gtf, n = 100, f = 10, setSeed = NULL)
```

Arguments

`gtf`	A dataframe containing genome annotation information This can be generated with `formatGTF`.
`n`	Integer specifying the number of randomly selected transcripts from which random back-spliced junctions are extracted. Default value = 100.
`f`	An integer specifying the fraction of single exon circRNAs that have to be present in the output data frame. Default value is 10.
`setSeed`	An integer which is used for selecting random back-spliced junctions. Default values is set to NULL.

Value

A data frame.

Examples

```
# Load short version of the gencode v19 annotation file
data("gtf")

# Get 10 random back-spliced junctions
randomBSJunctions <- getRandomBSJunctions(gtf, n = 10, f = 10)
```

---

getRegexPattern *Convert IUPAC sequence to an regular expression*

Description

The function getRegexPattern() converts a nucleotide sequence with IUPAC codes to an regular expression.

Usage

```
getRegexPattern(iupacSeq, isDNA = FALSE)
```

Arguments

iupacSeq	A character string with IUPAC codes.
isDNA	A logical specifying whether the iupacSeq is DNA or RNA. Deafult value is FALSE.

Value

A character string.

Examples

```
regextPattern <- getRegexPattern("CGUKMBVNN", isDNA = FALSE)
```

`getSeqsAcrossBSJs` *Retrieve back-spliced junction sequences*

Description

The function getSeqsAcrossBSJs() retrieves the sequences across the back-spliced junctions. A total of 11 nucleotides from each side of the back-spliced junction are taken and concatenated together.

Usage

```
getSeqsAcrossBSJs(annotatedBSJs, gtf, genome)
```

Arguments

`annotatedBSJs`	A data frame with the annotated back-spliced junctions. It can be generated with `annotateBSJs`.
`gtf`	A dataframe containing genome annotation information. It can be generated with `formatGTF`.
`genome`	A BSgenome object containing the genome sequences. It can be generated with `getBSgenome`. See `available.genomes` to see the BSgenome package currently available.

Value

A list.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){

genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences
targets <- getSeqsAcrossBSJs(
    annotatedBSJs,
    gtf,
    genome)
}
```

getSeqsFromGRs *Retrieve sequences flanking back-spliced junctions*

Description

The function getSeqsFromGRs() includes 3 modules to retrieve 3 types of sequences. Sequences of the introns flanking back-spliced junctions, sequences from a defined genomic window surrounding the back-spliced junctions and sequences of the back-spliced exons.

Usage

```
getSeqsFromGRs(annotatedBSJs, genome, lIntron = 100, lExon = 10, type = "ie")
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. This data frame can be generated with annotateBSJs.
genome	A BSgenome object containing the genome sequences. It can be generated with in getBSgenome. See available.genomes to see the BSgenome package currently available
lIntron	An integer indicating how many nucleotides are taken from the introns flanking the back-spliced junctions. This number must be positive. Default value is 100.
lExon	An integer indicating how many nucleotides are taken from the back-spliced exons starting from the back-spliced junctions. This number must be positive. Default value is 10.
type	A character string specifying the sequences to retrieve. If type = "ie" the sequences are retrieved from the the genomic ranges defined by using the lIntron and lExon given in input. If type = "bse" the sequences of the back-spliced exons are retrieved. If type = "fi" the sequences of the introns flanking the back-spliced exons are retrieved. Default value is "ie".

Value

A list.

Examples

```
# Load data frame containing predicted back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
```

```
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg19", quietly = TRUE)){
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences
targets <- getSeqsFromGRs(
    annotatedBSJs,
    genome,
    lIntron = 200,
    lExon = 10,
    type = "ie"
    )
}
```

`gtf` *gtf*

Description

A data frame containing a short version of the already formatted gencode v19 based-genome annotations. This data frame can be generated with `formatGTF`.

Usage

```
data(gtf)
```

Format

An object of class `data.frame` with 4401 rows and 9 columns.

Examples

```
data(gtf)
```

`gwasTraits` *gwasTraits*

Description

A data frame containing the traits/disease extracted on the 31th October 2018 from the GWAS catalog. See also https://www.ebi.ac.uk/gwas/ for more detail.

Usage

```
data(gwasTraits)
```

Format

A data frame with 653 rows and 1 columns

id is the trait/disease as reported in the GWAS catalog

Examples

```
data(gwasTraits)
```

`initCircRNAprofiler` *Initialize the project folder*

Description

The function initCircRNAprofiler() initializes the project forlder.

Usage

```
initCircRNAprofiler(projectFolderName, detectionTools)
```

Arguments

`projectFolderName`	A character string specifying the name of the project folder.
`detectionTools`	A character vector specifying the tools used to predict circRNAs. The following options are allowed: mapsplice, nclscan, knife, circexplorer2, circmarker and uroborus. If the tool is not mapsplice, nclscan, knife, circexplorer2, uroborus or circmarker then use the option other. The user can choose 1 or multiple tools. Subfolders named as the specified tools will be generated under the working directory.

Value

A NULL object

Examples

```
## Not run:
initCircRNAprofiler(projectFolderName = "circProject",
    detectionTools = "mapsplice")

## End(Not run)
```

iupac *iupac*

Description

A data frame containing IUPAC codes and the correspoding regular expression.

Usage

```
data(iupac)
```

Format

A data frame with 18 rows and 4 columns

code is the IUPAC nucleotide code

base is the base

regexDNA is the regular expression of the corresponding IUPAC code

regexRNA is the regular expression of the corresponding IUPAC code

Examples

```
data(iupac)
```

liftBSJcoords *LiftOver back-spliced junction coordinates*

Description

The function liftBSJcoords() maps back-spliced junction coordinates between species ad genome assemblies by using the liftOver utility from UCSC. Only back-spliced junction coordinates where the mapping was successful are reported.

Usage

```
liftBSJcoords(
  backSplicedJunctions,
  map = "hg19ToMm9",
  annotationHubID = "AH14155"
)
```

Arguments

backSplicedJunctions
: A data frame containing the back-spliced junction coordinates (predicted or randomly selected). See getRandomBSJunctions, getBackSplicedJunctions and mergeBSJunctions (to group circRNAs detected by multiple detection tools), on how to generated this data frame.

map
: A character string in the format <db1>To<Db2> (e.g."hg19ToMm9") specifying the reference genome mapping logic associated with a valid .chain file. Default value is "hg19ToMm9".

annotationHubID
: A string specifying the AnnotationHub id associated with a valid *.chain file. Type data(ahChainFiles) to see all possibile options. E.g. if AH14155 is specified, the hg19ToMm9.over.chain.gz will be used to convert the hg19 (Human GRCh37) coordinates to mm10 (Mouse GRCm38). Default value is "AH14155".

Value

a data frame.

Examples

```
# Load a data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# LiftOver the first 10 back-spliced junction coordinates
liftedBSJcoords <- liftBSJcoords(mergedBSJunctions[1:10,], map = "hg19ToMm9")
```

---

memeDB *memeDB*

Description

A dataframe containing the file paths of the files containing the RBP motifs in meme format available in MEME db. See also http://meme-suite.org/doc/download.html. File paths extracted from RNA folder of motif_databases.12.19.tgz (Motif Databases updated 28 Oct 2019)

Usage

```
data(memeDB)
```

Format

A character vector with 25 rows ans 2 columns

path is the file path

index index of the file path

Examples

```
data(memeDB)
```

`mergeBSJunctions` *Group circRNAs identified by multiple prediction tools*

Description

The function mergeBSJunctions() shrinks the data frame by grouping back-spliced junctions commonly identified by multiple detection tools. The read counts of the samples reported in the final data frame will be the ones of the tool that detected the highest total mean across all samples. All the tools that detected the back-spliced junctions are then listed in the column "tool" of the final data frame. See `getDetectionTools` for more detail about the code corresponding to each circRNA detection tool.

NOTE: Since different detection tools can report sligtly different coordinates before grouping the back-spliced junctions, it is possible to fix the latter using the gtf file. In this way the back-spliced junctions coordinates will correspond to the exon coordinates reported in the gtf file. A difference of maximum 2 nucleodites is allowed between the bsj and exon coordinates. See param fixBSJsWithGTF.

Usage

```
mergeBSJunctions(
  backSplicedJunctions,
  gtf,
  pathToExperiment = NULL,
  exportAntisense = FALSE,
  fixBSJsWithGTF = FALSE
)
```

Arguments

`backSplicedJunctions`
: A data frame containing back-spliced junction coordinates and counts generated with `getBackSplicedJunctions`.

`gtf`
: A data frame containing genome annotation information, generated with `formatGTF`.

`pathToExperiment`
: A string containing the path to the experiment.txt file. The file experiment.txt contains the experiment design information. It must have at least 3 columns with headers: - label (1st column): unique names of the samples (short but informative). - fileName (2nd column): name of the input files - e.g. circRNAs_X.txt, where x can be can be 001, 002 etc. - group (3rd column): biological conditions - e.g. A or B; healthy or diseased if you have only 2 conditions.

By default pathToExperiment i set to NULL and the file it is searched in the working directory. If experiment.txt is located in a different directory then the path needs to be specified.

exportAntisense
: A logical specifying whether to export the identified antisense circRNAs in a file named antisenseCircRNAs.txt. Default value is FALSE. A circRNA is defined antisense if the strand reported in the prediction results is different from the strand reported in the genome annotation file. The antisense circRNAs are removed from the returned data frame.

fixBSJsWithGTF
: A logical specifying whether to fix the back-spliced junctions coordinates using the GTF file. Default value is FALSE.

Value

A data frame.

Examples

```
# Load detected back-soliced junctions
data("backSplicedJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

pathToExperiment <- system.file("extdata", "experiment.txt",
    package ="circRNAprofiler")

# Merge commonly identified circRNAs
mergedBSJunctions <- mergeBSJunctions(backSplicedJunctions, gtf,
    pathToExperiment)
```

---

mergedBSJunctions *mergedBSJunctions*

Description

A data frame containing genomic coordinates (genome assembly hg19) detected in human LV tissues of controls and diseased hearts. This data frame was generated with mergedBSJunctions which grouped circRNAs commonly identified by the three tools (CircMarker(cm), MapSplice2 (m) and NCLscan (n)) used for circRNAs detection.

Usage

```
data(mergedBSJunctions)
```

Format

A data frame with 41558 rows and 16 columns

id Unique identifier

gene is the gene name whose exon coordinates overlap that of the given back-spliced junctions

strand is the strand where the gene is transcribed

chrom the chromosome from which the circRNA is derived

startUpBSE is the 5' coordinate of the upstream back-spliced exon in the transcript. This corresponds to the back-spliced junction / acceptor site

endDownBSE is the 3' coordinate of the downstream back-spliced exon in the transcript. This corresponds to the back-spliced junction / donor site

tool are the tools that identified the back-spliced junctions

C1 Number of occurences of each circRNA in control 1

C2 Number of occurences of each circRNA in control 2

C3 Number of occurences of each circRNA in control 3

D1 Number of occurences of each circRNA in DCM 1

D2 Number of occurences of each circRNA in DCM 2

D3 Number of occurences of each circRNA in DCM 3

H1 Number of occurences of each circRNA in HCM 1

H2 Number of occurences of each circRNA in HCM 2

H3 Number of occurences of each circRNA in HCM 3

Examples

```
data(mergedBSJunctions)
```

`mergeMotifs` *Group motifs shared by multiple RBPs*

Description

A same RBP can recognize multiple motifs, the function mergeMotifs() groups all the motifs found for each RBP and report the total counts.

Usage

```
mergeMotifs(motifs)
```

Arguments

`motifs`	A data frame generated with `getMotifs`.

Value

A data frame.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Example with the first back-spliced junctions.
# Multiple back-spliced junctions can also be analyzed at the same time.

# Annotate detected back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences
targets <- getSeqsFromGRs(
    annotatedBSJs,
    genome,
    lIntron = 200,
    lExon = 10,
    type = "ie"
    )

# Get motifs
motifs <-
getMotifs(
    targets,
    width = 6,
    species = "Hsapiens",
    rbp = TRUE,
   reverse = FALSE)

# Group motifs
mergedMotifs <- mergeMotifs(motifs)
```

`miRspeciesCodes` *miRspeciesCodes*

Description

A data frame containing the code of the species as reported in miRBase 22 release. See also http://www.mirbase.org

Usage

```
data(miRspeciesCodes)
```

Format

A data frame with 271 rows and 2 columns

code is the unique identifier of the species as reported in miRBase db

species is the species

Examples

```
data(miRspeciesCodes)
```

---

plotExBetweenBSEs *Plot exons between back-spliced junctions*

Description

The function plotExBetweenBSEs() generates a bar chart showing the no. of exons in between the back-spliced junctions.

Usage

```
plotExBetweenBSEs(annotatedBSJs, title = "")
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. This data frame can be generated with annotateBSJs.
title	A character string specifying the title of the plot.

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Plot
p <- plotExBetweenBSEs(annotatedBSJs, title = "")
p
```

plotExPosition *Plot back-spliced exon positions*

Description

The function plotExPosition() generates a bar chart showing the position of the back-spliced exons within the transcript.

Usage

```
plotExPosition(annotatedBSJs, title = "", n = 0, flip = FALSE)
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. This data frame can be generated with annotateBSJs.
title	A character string specifying the title of the plot.
n	An integer specyfing the position counts cut-off. If 0 is specified all position are plotted. Deafult value is 0.
flip	A logical specifying whether to flip the transcripts. If TRUE all transcripts are flipped and the last exons will correspond to the first ones, the second last exons to the second ones etc. Default value is FALSE.

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Plot
p <- plotExPosition(annotatedBSJs, title = "", n = 0, flip = FALSE)
p
```

plotHostGenes *Plot circRNA host genes*

Description

The function plotHostGenes() generates a bar chart showing the no. of circRNAs produced from each the circRNA host gene.

Usage

```
plotHostGenes(annotatedBSJs, title = "")
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. This data frame can be generated with annotateBSJs.
title	A character string specifying the title of the plot.

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Plot
p <- plotHostGenes(annotatedBSJs, title = "")
p
```

plotLenBSEs *Plot length back-spliced exons*

Description

The function plotLenBSEs() generates vertical boxplots for comparison of length of back-spliced exons (e.g. detected Vs randomly selected).

Usage

```
plotLenBSEs(
  annotatedFBSJs,
  annotatedBBSJs,
  df1Name = "foreground",
  df2Name = "background",
  title = "",
  setyLim = FALSE,
  ylim = c(0, 8)
)
```

Arguments

annotatedFBSJs	A data frame with the annotated back-spliced junctions (e.g. detected). It can be generated with annotateBSJs. These act as foreground back-spliced junctions.
annotatedBBSJs	A data frame with the annotated back-spliced junctions (e.g. randomly selected). It can generated with annotateBSJs. These act as background back-spliced junctions.
df1Name	A string specifying the name of the first data frame. This will be displayed in the legend of the plot.
df2Name	A string specifying the name of the first data frame. This will be displayed in the legend of the plot.
title	A character string specifying the title of the plot
setyLim	A logical specifying whether to set y scale limits. If TRUE the value in ylim will be used. Deafult value is FALSE.
ylim	An integer specifying the lower and upper y axis limits Deafult values are c(0, 8).

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedFBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Get random back-spliced junctions
randomBSJunctions <- getRandomBSJunctions(n = 10, f = 10, gtf)

# Annotate random back-spliced junctions
annotatedBBSJs <- annotateBSJs(randomBSJunctions, gtf, isRandom = TRUE)
```

```
# Plot
p <- plotLenBSEs(
    annotatedFBSJs,
    annotatedBBSJs,
    df1Name = "foreground",
    df2Name = "background",
    title = "")
p
```

---

plotLenIntrons *Plot length introns flanking back-spliced junctions*

Description

The function plotLenIntrons() generates vertical boxplots for comparison of length of introns flanking the back-spliced junctions (e.g. detected Vs randomly selected).

Usage

```
plotLenIntrons(
  annotatedFBSJs,
  annotatedBBSJs,
  df1Name = "foreground",
  df2Name = "background",
  title = "",
  setyLim = FALSE,
  ylim = c(0, 8)
)
```

Arguments

annotatedFBSJs	A data frame with the annotated back-spliced junctions (e.g. detected). It can be generated with annotateBSJs. These act as foreground back-spliced junctions.
annotatedBBSJs	A data frame with the annotated back-spliced junctions (e.g. randomly selected). It can generated with annotateBSJs. These act as background back-spliced junctions.
df1Name	A string specifying the name of the first data frame. This will be displayed in the legend of the plot. Deafult value is "foreground".
df2Name	A string specifying the name of the first data frame. This will be displayed in the legend of the plot. Deafult value is "background".
title	A character string specifying the title of the plot.
setyLim	A logical specifying whether to set y scale limits. If TRUE the value in ylim will be used. Deafult value is FALSE.
ylim	An integer specifying the lower and upper y axis limits Deafult values are c(0, 8).

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedFBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Get random back-spliced junctions
randomBSJunctions <- getRandomBSJunctions( gtf, n = 10, f = 10)

# Annotate random back-spliced junctions
annotatedBBSJs <- annotateBSJs(randomBSJunctions, gtf, isRandom = TRUE)

# Plot
p <- plotLenIntrons(
    annotatedFBSJs,
    annotatedBBSJs,
    df1Name = "foreground",
    df2Name = "background",
    title = "")
p
```

---

plotMiR *Plot miRNA analysis results*

Description

The function plotMiR() generates a scatter plot showing the number of miRNA binding sites for each miR found in the target sequence.

Usage

```
plotMiR(rearragedMiRres, n = 40, color = "blue", miRid = FALSE, id = 1)
```

Arguments

`rearragedMiRres`
: A list containing containing rearranged miRNA analysis results. See `getMiRsites` and then `rearrangeMiRres`.

n	An integer specifying the miRNA binding sites cut-off. The miRNA with a number of binding sites equals or higher to the cut-off will be colored. Deafaut value is 40.
color	A string specifying the color of the top n miRs. Default value is "blue".
miRid	A logical specifying whether or not to show the miR ids in the plot. default value is FALSE.
id	An integer specifying which element of the list rearragedMiRres to plot. Each element of the list contains the miR resutls relative to one circRNA. Deafult value is 1.

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences.
targets <- getCircSeqs(
    annotatedBSJs,
    gtf,
    genome)

# Screen target sequence for miR binding sites.
pathToMiRs <- system.file("extdata", "miRs.txt", package="circRNAprofiler")

# miRsites <- getMiRsites(
#     targets,
#     miRspeciesCode = "hsa",
#     miRBaseLatestRelease = TRUE,
#     totalMatches = 6,
#     maxNonCanonicalMatches = 1,
#     pathToMiRs)

# Rearrange miR results
# rearragedMiRres <- rearrangeMiRres(miRsites)

# Plot
# p <- plotMiR(
#     rearragedMiRres,
```

```
#     n = 20,
#     color = "blue",
#     miRid = TRUE,
#     id = 3)
# p
```

`plotMotifs` *Plot motifs analysis results*

Description

The function plotMotifs() generates 2 bar charts showing the log2FC and the number of occurences of each motif found in the target sequences (e.g detected Vs randomly selected).

Usage

```
plotMotifs(
  mergedMotifsFTS,
  mergedMotifsBTS,
  log2FC = 1,
  n = 0,
  removeNegLog2FC = FALSE,
  nf1 = 1,
  nf2 = 1,
  df1Name = "foreground",
  df2Name = "background",
  angle = 0
)
```

Arguments

`mergedMotifsFTS`
: A data frame containing the number of occurences of each motif found in foreground target sequences (e.g from detected back-spliced junctions). It can be generated with the `mergeMotifs`.

`mergedMotifsBTS`
: A data frame containing the number of occurences of each motif found in the background target sequences (e.g. from random back-spliced junctions). It can be generated with the `mergeMotifs`.

`log2FC`
: An integer specifying the log2FC cut-off. Default value is 1.

  NOTE: log2FC is calculated as follow: normalized number of occurences of each motif found in the foreground target sequences / normalized number of occurences of each motif found in the background target sequences.

  To avoid infinity as a value for fold change, 1 was added to number of occurences of each motif found in the foreground and background target sequences before the normalization.

n — An integer specifying the number of motifs cutoff. E.g. if 3 is specifiyed only motifs that are found at least 3 times in the foreground or background target sequences are retained. Deafaut value is 0.

removeNegLog2FC — A logical specifying whether to remove the RBPs having a negative log2FC. If TRUE then only positive log2FC will be visualized. Default value is FALSE.

nf1 — An integer specifying the normalization factor for the first data frame mergedMotifsFTS. The occurrences of each motif plus 1 are divided by nf1. The normalized values are then used for fold-change calculation. Set this to the number or length of target sequences (e.g from detected back-spliced junctions) where the motifs were extracted from. Default value is 1.

nf2 — An integer specifying the normalization factor for the second data frame mergedMotifsBTS. The occurrences of each motif plus 1 are divided by nf2. The The normalized values are then used for fold-change calculation. Set this to the number of target sequences (e.g from random back-spliced junctions) where the motifs were extracted from. Default value is 1.

NOTE: If nf1 and nf2 is set equals to 1, the number or length of target sequences (e.g detected Vs randomly selected) where the motifs were extrated from, is supposed to be the same.

df1Name — A string specifying the name of the first data frame. This will be displayed in the legend of the plot. Deafult value is "foreground".

df2Name — A string specifying the name of the first data frame. This will be displayed in the legend of the plot. Deafult value is "background".

angle — An integer specifying the rotation angle of the axis labels. Default value is 0.

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedFBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get random back-spliced junctions
randomBSJunctions <- getRandomBSJunctions(gtf, n = 1, f = 10)

# Annotate random back-spliced junctions
annotatedBBSJs <- annotateBSJs(randomBSJunctions, gtf, isRandom = TRUE)

# Get genome
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")
```

```
# Retrieve target sequences from detected back-spliced junctions
targetsFTS <- getSeqsFromGRs(
   annotatedFBSJs,
   genome,
   lIntron = 200,
   lExon = 10,
   type = "ie"
   )

# Retrieve target sequences from random back-spliced junctions
targetsBTS <- getSeqsFromGRs(
   annotatedBBSJs,
   genome,
   lIntron = 200,
   lExon = 10,
   type = "ie"
   )

# Get motifs
 motifsFTS <- getMotifs(
     targetsFTS,
     width = 6,
     database = 'ATtRACT',
     species = "Hsapiens",
     rbp = TRUE,
     reverse = FALSE)

motifsBTS <- getMotifs(
     targetsBTS,
     width = 6,
     database = 'ATtRACT',
     species = "Hsapiens",
     rbp = TRUE,
     reverse = FALSE)

# Merge motifs
 mergedMotifsFTS <- mergeMotifs(motifsFTS)
 mergedMotifsBTS <- mergeMotifs(motifsBTS)

# Plot
 p <- plotMotifs(
     mergedMotifsFTS,
     mergedMotifsBTS,
     log2FC = 2,
     nf1 = nrow(annotatedFBSJs),
     nf2 = nrow(annotatedBBSJs),
     df1Name = "foreground",
     df2Name = "background")
```

---

plotTotExons *Plot exons in the circRNA host transcript*

Description

The function plotTotExons() generates a bar chart showing the total number of exons (totExon column) in the transcripts selected for the downstream analysis.

Usage

```
plotTotExons(annotatedBSJs, title = "")
```

Arguments

annotatedBSJs	A data frame with the annotated back-spliced junctions. This data frame can be generated with annotateBSJs.
title	A character string specifying the title of the plot

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first 10 back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1:10, ], gtf)

# Plot
p <- plotTotExons(annotatedBSJs, title = "")
p
```

---

rearrangeMiRres *Rearrange miR results*

Description

The function rearrangeMiRres() rearranges the results of the getMiRsites() function. Each element of the list contains the miR results relative to one circRNA. For each circRNA only miRNAs for which at least 1 miRNA binding site is found are reported.

Usage

```
rearrangeMiRres(miRsites)
```

Arguments

miRsites	A list containing the miR sites found in the RNA target sequence. it can be generated with getMiRsites.

Value

A list.

Examples

```
# Load data frame containing predicted back-spliced junctions
data("mergedBSJunctions")

# Load short version of the gencode v19 annotation file
data("gtf")

# Annotate the first back-spliced junctions
annotatedBSJs <- annotateBSJs(mergedBSJunctions[1, ], gtf)

# Get genome
genome <- BSgenome::getBSgenome("BSgenome.Hsapiens.UCSC.hg19")

# Retrieve target sequences.
 targets <- getCircSeqs(
     annotatedBSJs,
     gtf,
     genome)

# Screen target sequence for miR binding sites.
pathToMiRs <- system.file("extdata", "miRs.txt",package="circRNAprofiler")

# miRsites <- getMiRsites(
#     targets,
#     miRspeciesCode = "hsa",
#     miRBaseLatestRelease = TRUE,
#     totalMatches = 6,
#     maxNonCanonicalMatches = 1,
#     pathToMiRs)

# Rearrange miR results
# rearragedMiRres <- rearrangeMiRres(miRsites)
```

---

volcanoPlot *Plot differential circRNA expression results*

Description

The function volcanoPlot() generates a volcano plot with the results of the differential expression analysis.

Usage

```
volcanoPlot(
  res,
  log2FC = 1,
  padj = 0.05,
  title = "",
  gene = FALSE,
  geneSet = c(""),
  setxLim = FALSE,
  xlim = c(-8, 8),
  setyLim = FALSE,
  ylim = c(0, 5),
  color = "blue"
)
```

Arguments

res	A data frame containing the the differential expression resuls. It can be generated with getDeseqRes or getEdgerRes.
log2FC	An integer specifying the log2FC cut-off. Deafult value is 1.
padj	An integer specifying the adjusted P value cut-off. Deafult value is 0.05.
title	A character string specifying the title of the plot.
gene	A logical specifying whether to show all the host gene names of the differentially expressed circRNAs to the plot. Deafult value is FALSE.
geneSet	A character vector specifying which host gene name of the differentially expressed circRNAs to show in the plot. Multiple host gene names can be specofied. E.g. c('TTN, RyR2')
setxLim	A logical specifying whether to set x scale limits. If TRUE the value in xlim will be used. Deafult value is FALSE.
xlim	A numeric vector specifying the lower and upper x axis limits. Deafult values are c(-8 , 8).
setyLim	A logical specifying whether to set y scale limits. If TRUE the value in ylim will be used. Deafult value is FALSE.
ylim	An integer specifying the lower and upper y axis limits Deafult values are c(0, 5).
color	A string specifying the color of the differentially expressed circRNAs. Default value is "blue".

Value

A ggplot object.

Examples

```
# Load data frame containing detected back-spliced junctions
data("mergedBSJunctions")

pathToExperiment <- system.file("extdata", "experiment.txt",
    package ="circRNAprofiler")

# Filter circRNAs
filterdCirc <- filterCirc(
    mergedBSJunctions,
    allSamples = FALSE,
    min = 5,
    pathToExperiment)

# Find differentially expressed circRNAs
 deseqResBvsA <- getDeseqRes(
    filterdCirc,
    condition = "A-B",
    pAdjustMethod = "BH",
    pathToExperiment)

# Plot
p <- volcanoPlot(
    deseqResBvsA,
    log2FC = 1,
    padj = 0.05,
    title = "",
    setxLim = TRUE,
    xlim = c(-8 , 7.5),
    setyLim = FALSE,
    ylim = c(0 , 4),
    gene = FALSE)
p
```

Index

∗ **datasets**
  ahChainFiles, 3
  ahRepeatMasker, 3
  attractSpecies, 8
  backSplicedJunctions, 8
  gtf, 27
  gwasTraits, 27
  iupac, 29
  memeDB, 30
  mergedBSJunctions, 32
  miRspeciesCodes, 34

ahChainFiles, 3
ahRepeatMasker, 3
annotateBSJs, 4, *14*, *25, 26*, *35–39*, *45*
annotateRepeats, 5
annotateSNPsGWAS, 6
AnnotationHub, *5*
attractSpecies, 8
available.genomes, *14*, *25, 26*

backSplicedJunctions, 8, *14*

calcNormFactors, *18*
checkProjectFolder, 9

DESeq, *16*

filterCirc, 11
formatGTF, *4*, 12, *13, 14*, *23*, *25*, *27*, *31*

getBackSplicedJunctions, *4*, *8*, *11*, 13, *15*, *18*, *30, 31*
getBSgenome, *14*, *25, 26*
getCircSeqs, 14, *19*, *21*
getDeseqRes, 15, *47*
getDetectionTools, 17, *31*
getEdgerRes, 17, *47*
getMiRsites, 19, *40*, *46*
getMotifs, 21, *33*
getRandomBSJunctions, *4*, 23, *30*
getRegexPattern, 24
getSeqsAcrossBSJs, *21*, 25
getSeqsFromGRs, *5*, *7*, *21*, 26
gtf, 27
gwasTraits, 27

initCircRNAprofiler, *9*, 28
iupac, 29

liftBSJcoords, 29

makeCurrentGwascat, *7*
memeDB, 30
mergeBSJunctions, *4*, *11*, *15*, *18*, *30*, 31
mergedBSJunctions, 32, *32*
mergeMotifs, 33, *42*
miRspeciesCodes, 34

p.adjust, *16*, *18*
plotExBetweenBSEs, 35
plotExPosition, 36
plotHostGenes, 37
plotLenBSEs, 37
plotLenIntrons, 39
plotMiR, 40
plotMotifs, 42
plotTotExons, 44

rearrangeMiRres, *40*, 45

volcanoPlot, 46