Title Comprehensive design of CRISPR gRNAs for nucleases and base editors

Version 1.6.0

Description Provides a comprehensive suite of functions to design and annotate CRISPR guide RNA (gRNAs) sequences. This includes on- and off-target search, on-target efficiency scoring, off-target scoring, full gene and TSS contextual annotations, and SNP annotation (human only). It currently supports five types of CRISPR modalities (modes of perturbations): CRISPR knockout, CRISPR activation, CRISPR inhibition, CRISPR base editing, and CRISPR knockdown. All types of CRISPR nucleases are supported, including DNA- and RNA-target nucleases such as Cas9, Cas12a, and Cas13d. All types of base editors are also supported. gRNA design can be performed on reference genomes, transcriptomes, and custom DNA and RNA sequences. Both unpaired and paired gRNA designs are enabled.

Depends R (>= 4.2.0), crisprBase (>= 1.1.3)

Imports AnnotationDbi, BiocGenerics, Biostrings, BSgenome, crisprBowtie (>= 0.99.8), crisprScore (>= 1.1.6), GenomeInfoDb, GenomicFeatures, GenomicRanges (>= 1.38.0), IRanges, Matrix, MatrixGenerics, methods, rtracklayer, S4Vectors, stats, txdbmaker, utils, VariantAnnotation

Suggests biomaRt, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, BiocStyle, crisprBwa (>= 0.99.7), knitr, rmarkdown, Rbowtie, Rbwa, RCurl, testthat

biocViews CRISPR, FunctionalGenomics, GeneTarget

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

VignetteBuilder knitr

BugReports https://github.com/crisprVerse/crisprDesign/issues

URL https://github.com/crisprVerse/crisprDesign

LazyData true
Contents

addCompositeScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
addConservationScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
addCrispraiScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
addDistanceToTss . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
addEditedAlleles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
addEditingSites . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
addExonTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
addGeneAnnotation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
addIsoformAnnotation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
addNtcs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
addOffTargetScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
addOnTargetScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
addOpsBarcodes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
addPamScores . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
addPfamDomains . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
addReinitiationFlag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
addRepeats . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
addRestrictionEnzymes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
addSequenceFeatures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
addSNPAnnotation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
addSpacerAlignments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
addTssAnnotation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
addTxTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
completeSpacers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
convertToMinMaxGRanges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
convertToProtoscalerGRanges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
crisprNuclease . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
designCompleteAnnotation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
designOpsLibrary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
findSpacerPairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
findSpacers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
flattenGuideSet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
getBarcodeDistanceMatrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
addCompositeScores

Add on-target composite score to a GuideSet object.

Description
Add on-target composite score to a GuideSet object.

Usage
addCompositeScores(object, ...)

## S4 method for signature 'GuideSet'
addCompositeScores(
  object,
  methods = c("azimuth", "ruleset1", "ruleset3", "lindel", "deepcpf1", "deephf",
              "deepspcas9", "enpamgb", "casrxf", "crisprater", "crisprscan", "crispra", "crispri"),
  scoreName = "score_composite"
)

## S4 method for signature 'PairedGuideSet'
addCompositeScores(
addCompositeScores

object, 
methods = c("azimuth", "ruleset1", "ruleset3", "lindel", "deepcpf1", "deephf", 
  "deepspcas9", "enpamgb", "crisprater", "crisprscan", "casrxrf", "crispra", "crispri"), 
scoreName = "score_composite"
)

## S4 method for signature 'NULL'
addCompositeScores(object)

Arguments

object       A GuideSet object or a PairedGuideSet object.
...          Additional arguments, currently ignored.
methods      Character vector specifying method names for on-target efficiency prediction 
             algorithms to be used to create the composite score. Note that the specified 
             scores must be added first to the object using addOnTargetScores.
scoreName    String specifying the name of the composite score to be used as a column name. 
             Users can choose whatever they like. Default is "score_composite".

Details

The function creates a composite score across a specified list of on-target scores by first transforming 
each individual score into a rank, and then taking the average rank across all specified methods. 
This can improve on-target activity prediction robustness. A higher score indicates higher on-target 
activity.

Value

guideSet with column specified by scoreName appended in mcols(guideSet).

Author(s)

Jean-Philippe Fortin

See Also

addOnTargetScores to add on-target scores.

Examples

gs <- findSpacers("CCAACATAGTGAACACAGTCTCTATAAAAGATAAAAATTAGCCGGGTTA")
gs <- addOnTargetScores(gs, methods=c("ruleset1", "crisprater"))
gs <- addCompositeScores(gs, methods=c("ruleset1", "crisprater"))
addConservationScores  Add on-target composite score to a GuideSet object.

Description

Add on-target composite score to a GuideSet object.

Usage

addConservationScores(object, ...)

## S4 method for signature 'GuideSet'
addConservationScores(
  object,
  conservationFile,
  nucExtension = 9,
  fun = c("mean", "max"),
  scoreName = "score_conservation"
)

## S4 method for signature 'PairedGuideSet'
addConservationScores(
  object,
  conservationFile,
  nucExtension = 9,
  fun = c("mean", "max"),
  scoreName = "score_conservation"
)

## S4 method for signature 'NULL'
addConservationScores(object)

Arguments

**object**  A GuideSet object or a PairedGuideSet object.

**...**  Additional arguments, currently ignored.

**conservationFile**  String specifying the BigWig file containing conservation scores.

**nucExtension**  Number of nucleotides to include on each side of the cut site to calculate the conservation score. 9 by default. The region will have (2*nucExtension + 1) nucleotides in total.

**fun**  String specifying the function to use to calculate the final conservation score in the targeted region. Must be either "mean" (default) or "max".

**scoreName**  String specifying the name of the conservation score to be used as a column name. Users can choose whatever they like. Default is "score_conservation".
addCrispraiScores

Details
The function creates a conservation score for each gRNA by using the max, or average, conservation score in the genomic region where the cut occurs. A BigWig file storing conservation stores must be provided. Such files can be downloaded from the UCSC genome browser. See vignette for more information.

Value
guideSet with column specified by scoreName appended in mcols(guideSet).

Author(s)
Jean-Philippe Fortin

Usage
addCrispraiScores(object, ...)

## S4 method for signature 'GuideSet'
addCrispraiScores(
  object,
  gr,
  tssObject,
  geneCol = "gene_id",
  modality = c("CRISPRi", "CRISPRa"),
  chromatinFiles = NULL,
  fastaFile = NULL
)

## S4 method for signature 'PairedGuideSet'
addCrispraiScores(
  object,
  gr,
  tssObject,
  geneCol = "gene_id",
  modality = c("CRISPRi", "CRISPRa"),
  chromatinFiles = NULL,
  fastaFile = NULL
)

Description
Add CRISPRa/CRISPRi on-target scores to a GuideSet object. Only available for SpCas9, and for hg38 genome. Requires crisprScore package to be installed.
addDistanceToTss

## S4 method for signature 'NULL'
addCrisprAIScores(object)

### Arguments

- **object**
  A `GuideSet` object or a `PairedGuideSet` object.

- **...**
  Additional arguments, currently ignored.

- **gr**
  A `GRanges` object derived from `queryTss` used to produce the `guideSet` object.

- **tssObject**
  A `GRanges` object containing TSS coordinates and annotation. The following columns must be present: "ID", promoter", "tx_id" and "gene_symbol".

- **geneCol**
  String specifying which column of the `tssObject` should be used for a unique gene identified. "gene_id" by default.

- **modality**
  String specifying which modality is used. Must be either "CRISPRi" or "CRISPRa".

- **chromatinFiles**
  Named character vector of length 3 specifying BigWig files containing chromatin accessibility data. See `crisprScore` vignette for more information.

- **fastaFile**
  String specifying fasta file of the hg38 genome.

### Value

- `guideSet` with an added column for the CRISPRai score.

### Author(s)

Jean-Philippe Fortin

### See Also

- `addOnTargetScores` to add other on-target scores.

### Description

Add distance to TSS for a specified TSS id.

### Usage

```r
addDistanceToTss(object, ...)  # S4 method for signature 'GuideSet'
addDistanceToTss(object, tss_id)  # S4 method for signature 'PairedGuideSet'
```
addEditedAlleles

Arguments

object A GuideSet object or a PairedGuideSet object.
... Additional arguments, currently ignored.
tss_id String specifying TSS id to calculate the distance. The column tssAnnotation(object)$tss_id will be used to search for the TSS id.

Value

A GuideSet object or a PairedGuideSet object with an additional metadata column called distance_to_tss reporting the distance (in nucleotides) between the TSS position of the TSS specified by tss_id and the protospacer position. The pam_site coordinate is used as the representative position of protospacer sequences.

Note that a TSS annotation must be available in the object. A TSS annotation can be added using addTssAnnotation.

Author(s)

Jean-Philippe Fortin

See Also

addTssAnnotation to add TSS annotation.

Examples

data(guideSetExampleFullAnnotation)
tss_id <- "ENSG00000120645_P1"
gs <- guideSetExampleFullAnnotation
gs <- addDistanceToTss(gs, tss_id)

addEditedAlleles

To add edited alleles for a CRISPR base editing GuideSet

Description

To add edited alleles for a CRISPR base editing GuideSet.

Usage

addEditedAlleles(
  guideSet,  
  baseEditor,  
  editingWindow = NULL,  
  nMaxAlleles = 100,  
  addFunctionalConsequence = TRUE,  
  addSummary = TRUE,  
)
addEditedAlleles

taxTable = NULL,
verbose = TRUE
)

Arguments

guideSet A GuideSet object.
baseEditor A BaseEditor object.
editingWindow A numeric vector of length 2 specifying start and end positions of the editing window with respect to the PAM site. If NULL (default), the editing window of the BaseEditor object will be considered.
nMaxAlleles Maximum number of edited alleles to report for each gRNA. Alleles from high to low scores. 100 by default.
addFunctionalConsequence Should variant classification of the edited alleles be added? TRUE by default. If TRUE, txTable must be provided.
addSummary Should a summary of the variant classified be added to the metadata columns of the guideSet object? TRUE by default.
taxTable Table of transcript-level nucleotide and amino acid information needed for variant classification. Usually returned by getTxInfoDataFrame.
verbose Should messages be printed to console? TRUE by default.

Value

The original guideSet object with an additional metadata column (editedAlleles) storing the annotated edited alleles. The edited alleles are always reported from 5' to 3' direction on the strand corresponding to the gRNA strand.

Author(s)

Jean-Philippe Fortin

Examples

data(BE4max, package="crisprBase")
data(grListExample, package="crisprDesign")
library(BSgenome.Hsapiens.UCSC.hg38)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
gr <- queryTxObject(grListExample,
  featureType="cds",
  queryColumn="gene_symbol",
  queryValue="IQSEC3")
gs <- findSpacers(gr[1],
  crisprNuclease=BE4max,
  bsgenome=bsgenome)
gs <- unique(gs)
gs <- gs[1:2] # For the sake of time

# Getting transcript info:
txid="ENST00000538872"

```r
taxTable <- getTxInfoDataFrame(tx_id=txid,
   txObject=grListExample,
   bsgenome=bsgenome)

#Adding alleles:
editingWindow <- c(-20,-8)
gs <- addEditedAlleles(gs,
   baseEditor=BE4max,
   txTable=txTable,
   editingWindow=editingWindow)
```

---

**addEditingSites**

*Add optimal editing site for base editing gRNAs.*

**Description**

Add optimal editing site for base editing gRNAs.

**Usage**

```r
addEditingSites(object, ...)
```

---

**Arguments**

- `object` A `GuideSet` object or a `PairedGuideSet` object.
- `...` Additional arguments, currently ignored.
- `substitution` String indicating which substitution should be used to estimate the optimal editing position. E.g. "C2T" will return the optimal editing position for C to T editing.

**Value**

An updated object with a column `editing_site` added to `mcols(object)`.

**Author(s)**

Jean-Philippe Fortin
addExonTable

Add a gene-specific exon table to a GuideSet object.

Description

Add a gene-specific exon table to a GuideSet object.

Usage

addExonTable(
  guideSet, 
  gene_id, 
  txObject, 
  valueColumn = "percentCDS", 
  useConsensusIsoform = FALSE 
)

Arguments

guideSet A GuideSet object or a PairedGuideSet object.
gene_id String specifying gene ID.
tXObject A TxDb object or a GRangesList object obtained using TxDb2GRangesList to provide a gene model annotation.
valueColumn String specifying column in geneAnnotation(guideSet) to use as values in the output exon table.
useConsensusIsoform Should a consensus isoform be used to annotate exons? FALSE by default. If TRUE, the isoform constructed by getConsensusIsoform will be used.

Value

A GuideSet object with a "exonTable" DataFrame stored in mcols(guideSet). The entries in the DataFrame correspond to the values specified by valueColumn. Rows correspond to gRNAs in the GuideSet, columns correspond to all exons found in txObject for gene specified by gene_id.

Author(s)

Jean-Philippe Fortin

See Also

addGeneAnnotation to add gene annotation and addTxTable to add a transcript table.
Examples

if (interactive()){
  data(guideSetExample, package="crisprDesign")
  data(grListExample, package="crisprDesign")
  guideSet <- addGeneAnnotation(guideSetExample,
                              txObject=grListExample)
  guideSet <- addExonTable(guideSet,
                      gene_id="ENSG00000120645",
                      txObject=grListExample)

  guideSet$exonTable
}

addGeneAnnotation  Add gene context annotation to a GuideSet object

Description

Add gene context annotation to spacer sequence stored in a GuideSet object

Usage

addGeneAnnotation(object, ...)

## S4 method for signature 'GuideSet'
addGeneAnnotation(
  object,
  txObject,
  anchor = c("cut_site", "pam_site", "editing_site"),
  ignore_introns = TRUE,
  ignore.strand = TRUE,
  addPfam = FALSE,
  mart_dataset = NULL
)

## S4 method for signature 'PairedGuideSet'
addGeneAnnotation(
  object,
  txObject,
  anchor = c("cut_site", "pam_site", "editing_site"),
  ignore_introns = TRUE,
  ignore.strand = TRUE,
  addPfam = FALSE,
  mart_dataset = NULL
)
Arguments

- **object**: A `GuideSet` object or a `PairedGuideSet` object.
- **txObject**: A `TxDb` object or a `GRangesList` object obtained using `TxDb2GRangesList` to provide a gene model annotation.
- **anchor**: String specifying which relative coordinate of gRNAs should be used to locate gRNAs within gene. Must be either "cut_site", "pam_site" or "editing_site".
- **ignore_introns**: Should gene introns be ignored when annotating? TRUE by default.
- **ignore.strand**: Should gene strand be ignored when annotating? TRUE by default.
- **addPfam**: Should Pfam domains annotation be added? FALSE by default. If set to TRUE, `biomaRt` must be installed.
- **mart_dataset**: String specifying dataset to be used by `biomaRt` for Pfam domains annotation. E.g. "hsapiens_gene_ensembl".

Details

For DNA-targeting nucleases, the different columns stored in `mcols(guideSet)[["geneAnnotation"]]

- **tx_id**: Transcript ID.
- **gene_symbol**: Gene symbol.
- **gene_id**: Gene ID.
- **protein_id**: Protein ID.
- **id**: gRNA ID.
- **pam_site**: gRNA PAM site coordinate.
- **cut_site**: gRNA cut site coordinate.
- **chr**: gRNA chromosome name.
- **strand**: gRNA strand.
- **cut_cds**: Is the gRNA cut site located within the coding sequence (CDS) of the targeted isoform?
- **cut_fiveUTRs**: Is the gRNA cut site located within the 5’UTR of the targeted isoform?
- **cut_threeUTRs**: Is the gRNA cut site located within the 3’UTR of the targeted isoform?
- **cut_introns**: Is the gRNA cut site located within an intron of the targeted isoform?
- **percentCDS**: Numeric value to indicate the relative position of the cut site with respect to the start of the CDS sequence when cut_cds is TRUE. The relative position is expressed as a percentage from the total length of the CDS.
- **percentRx**: Numeric value to indicate the relative position of the cut site with respect to the start of the mRNA sequence (therefore including 5’ UTR). The relative position is expressed as a percentage from the total length of the mRNA sequence.
- **aminoAcidIndex**: If cut_cds is TRUE, integer value indicating the amino acid index with respect to the start of the protein.
• **downstreamATG** Number of potential reinitiation sites (ATG codons) downstream of the gRNA cut site, within 85 amino acids.

• **nIsoforms** Numeric value indicating the number of isoforms targeted by the gRNA.

• **totalIsoforms** Numeric value indicating the total number of isoforms existing for the gene targeted by the gRNA and specified in gene_id.

• **percentIsoforms** Numeric value indicating the percentage of isoforms for the gene specified in gene_id targeted by the gRNA. Equivalent to nIsoforms/totalIsoforms*100.

• **isCommonExon** Logical value to indicate whether or not the gRNA is targeting an exon common to all isoforms.

• **nCodingIsoforms** Numeric value indicating the number of coding isoforms targeted by the gRNA. 5’ UTRs and 3’ UTRs are excluded.

• **totalCodingIsoforms** Numeric value indicating the total number of coding isoforms existing for the gene targeted by the gRNA and specified in gene_id.

• **percentCodingIsoforms** Numeric value indicating the percentage of coding isoforms for the gene specified in gene_id targeted by the gRNA. Equivalent to nCodingIsoforms/totalCodingIsoforms*100. 5’ UTRs and 3’ UTRs are excluded.

• **isCommonCodingExon** Logical value to indicate whether or not the gRNA is targeting an exon common to all coding isoforms.

**Value**

A GuideSet object with a "geneAnnotation" list column stored in mcols(guideSet). See details section for a description of the different gene annotation columns.

**Author(s)**

Jean-Philippe Fortin, Luke Hoberecht

**See Also**

addTssAnnotation to add TSS annotation, and geneAnnotation to retrieve an existing gene annotation.

**Examples**

data(guideSetExample, package="crisprDesign")
data(grListExample, package="crisprDesign")
guideSet <- addGeneAnnotation(guideSetExample[1:6],
                               txObject=grListExample)

# To access a gene annotation already added:
ann <- geneAnnotation(guideSet)
addIsoformAnnotation

Add isoform-specific annotation to a GuideSet object

Description

Add isoform-specific annotation to a GuideSet object.

Usage

addIsoformAnnotation(object, ...)

## S4 method for signature 'NULL'
addDistanceToTss(object)

## S4 method for signature 'GuideSet'
addIsoformAnnotation(object, tx_id)

## S4 method for signature 'PairedGuideSet'
addIsoformAnnotation(object, tx_id)

## S4 method for signature 'NULL'
addIsoformAnnotation(object)

Arguments

object A GuideSet object or a PairedGuideSet object.
...
Additional arguments, currently ignored.

Arguments

tx_id String specifying Ensembl ID for the isoform transcript of interested. E.g.
"ENST00000311936".

Value

A A GuideSet object or a PairedGuideSet object with the following added columns:
percentCDS, percentCodingIsoforms, and isCommonCodingExon. The column values are specific to the transcript specified by tx_id. The percentCDS column indicates at what percentage of the coding sequence the gRNA is cutting. The column percentCodingIsoforms indicates the percentage of coding isoforms that are targeted by the gRNA. The column isCommonCodingExon indicates whether or not the exon targeted by the gRNA is common to all isoforms for the gene.

Author(s)

Jean-Philippe Fortin
Examples

```r
data(guideSetExampleFullAnnotation)
tax_id <- "ENST00000538872"
gs <- guideSetExampleFullAnnotation
gs <- addIsoformAnnotation(gs, tx_id)
```

### addNtcs

Add non-targeting control (NTC) sequences to GuideSet

#### Description

Add non-targeting control (NTC) sequences to a GuideSet object.

#### Usage

```r
addNtcs(object, ...)
```

```r
## S4 method for signature 'GuideSet'
addNtcs(object, ntcs)
```

```r
## S4 method for signature 'PairedGuideSet'
addNtcs(object, ntcs)
```

```r
## S4 method for signature 'NULL'
addNtcs(object, ...)
```

#### Arguments

- `...`: Additional arguments, currently ignored.
- `ntcs`: A named character vector of NTC sequences. Sequences must consist of appropriate DNA or RNA bases, and have the same spacer length as spacers in `object`. Vector names are assigned as IDs and seqlevels, and must be unique and distinct from IDs and seqnames present in `object`.

#### Details

NTC sequences are appended as spacers to the GuideSet object. Each NTC sequence is assigned to its own "chromosome" in the ntc genome, as reflected in the Seqinfo of the resulting GuideSet object. As placeholder values, NTC ranges are set to 0 and strands set to *.

All annotation for NTC spacers appended to object are set to NA or empty list elements. To annotate NTC spacers, you must call the appropriate function after adding NTCs to the GuideSet object.

#### Value

The original object with appended ntcs spacers. Pre-existing annotation in object will be set to NA or empty list elements for appended NTC spacers.
Examples

```r
set.seed(1000)
data(guideSetExample, package="crisprDesign")
ntcs <- vapply(1:4, function(x){
  seq <- sample(c("A", "C", "G", "T"), 20, replace=TRUE)
  paste0(seq, collapse="")
}, FUN.VALUE=character(1))
names(ntcs) <- paste0("ntc_", 1:4)
gs <- addNtcs(guideSetExample, ntcs)
gs
```

addOffTargetScores  
Add CFD and MIT scores to a GuideSet object.

Description

Add CFD and MIT off-target scores to a GuideSet object. Both the CFD and MIT methods are available for the SpCas9 nuclease. The CFD method is also available for the CasRx nuclease. Other nucleases are currently not supported.

Usage

```r
addOffTargetScores(object, ...)  
## S4 method for signature 'GuideSet'
addOffTargetScores(object, max_mm = 2, includeDistance = TRUE, offset = 0)  
## S4 method for signature 'PairedGuideSet'
addOffTargetScores(object, max_mm = 2, includeDistance = TRUE, offset = 0)  
## S4 method for signature 'NULL'
addOffTargetScores(object)
```

Arguments

- `object`: A GuideSet object or a PairedGuideSet object. `crisprNuclease(object)` must be either using SpCas9 or CasRx.
- `...`: Additional arguments, currently ignored.
- `max_mm`: The maximum number of mismatches between the spacer sequence and the protospacer off-target sequence to be considered in the off-target score calculations. Off-targets with a number of mismatches greater than `max_mm` will be excluded; this is useful if one wants to avoid the aggregated off-target scores to be driven by a large number of off-targets that have low probability of cutting.
- `includeDistance`: Should a distance penalty for the MIT score be included? TRUE by default.
offset Numeric value specifying an offset to add to the denominator when calculating the aggregated score (inverse summation formula). 0 by default.

Details

See the `crisprScore` package for a description of the different off-target scoring methods.

Value

A GuideSet or a PairedGuideSet object with added scores. The alignments annotation returned by `alignments(object)` will have additional column storing off-target scores. Those scores representing the off-target score for each gRNA and off-target pair. For SpCas9, a column containing an aggregated specificity off-target score for each scoring method is added to the metadata columns obtained by `mcols(object)`.

Author(s)

Jean-Philippe Fortin, Luke Hoberecht

See Also

`link{addOnTargetScores}` to add on-target scores.

Examples

```r
data(guideSetExampleWithAlignments, package="crisprDesign")
gs <- guideSetExampleWithAlignments
gs <- addOffTargetScores(gs)
```

---

addOnTargetScores Add on-target scores to a GuideSet object.

Description

Add on-target scores to a GuideSet object for all methods available in the `crisprScore` package for a given CRISPR nuclease. Requires `crisprScore` package to be installed.

Usage

```r
addOnTargetScores(object, ...)
```

## S4 method for signature 'GuideSet'
addOnTargetScores(
  object,
  enzyme = c("WT", "ESP", "HF"),
  promoter = c("U6", "T7"),
  tracrRNA = c("Hsu2013", "Chen2013"),
)
directRepeat = "aacccctaccaactggtcggggtttgaaac",
binaries = NULL,
methods = c("azimuth", "ruleset1", "ruleset3", "lindel", "deepcpf1", "deephf",
         "deepspcas9", "enpamgb", "crisprate", "crisprscan")
)

## S4 method for signature 'PairedGuideSet'
addOnTargetScores(
  object,
  enzyme = c("WT", "ESP", "HF"),
  promoter = c("U6", "T7"),
  tracrRNA = c("Hsu2013", "Chen2013"),
  directRepeat = "aacccctaccaactggtcggggtttgaaac",
  binaries = NULL,
  methods = c("azimuth", "ruleset1", "ruleset3", "lindel", "deepcpf1", "deephf",
             "deepspcas9", "enpamgb", "crisprate", "crisprscan", "casrxrf")
)

## S4 method for signature 'NULL'
addOnTargetScores(object)

Arguments

object       A GuideSet object or a PairedGuideSet object.
...
enzyme      Character string specifying the Cas9 variant to be used for DeepHF scoring.
            Wildtype Cas9 (WT) by default. See details below.
promoter    Character string specifying promoter used for expressing sgRNAs for wildtype
            Cas9 (must be either "U6" or "T7") for DeepHF scoring. "U6" by default.
tracrRNA    String specifying which tracrRNA is used for SpCas9 Must be either "Hsu2013"
            (default) or "Chen2013". Only used for the RuleSet3 method.
directRepeat String specifying the direct repeat used in the CasRx construct.
binaries    Named list of paths for binaries needed for CasRx-RF. Names of the list must
            be "RNAfold", "RNAhybrid", and "RNApfold". Each list element is a string
            specifying the path of the binary. If NULL (default), binaries must be available
            on the PATH.
methods     Character vector specifying method names for on-target efficiency prediction
            algorithms.

Details

See crisprScore package for a description of each score.

Value

guideSet with columns of on-target scores appended in mcols(guideSet).
addOpsBarcodes

Author(s)
Jean-Philippe Fortin, Luke Hoberecht

See Also
addOffTargetScores to add off-target scores.

Examples
if (interactive()){
  gs <- findSpacers("CCACATAGTGAACCGTCTCTATAAAGAATAAAAAATTAGCCGGGTTA")
  gs <- addOnTargetScores(gs)
}

addOpsBarcodes
Add optical pooled screening (OPS) barcodes

Description
Add optical pooled screening (OPS) barcodes.

Usage
addOpsBarcodes(guideSet, n_cycles = 9, rt_direction = c("5prime", "3prime"))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>guideSet</td>
<td>A GuideSet object.</td>
</tr>
<tr>
<td>n_cycles</td>
<td>Integer specifying the number of sequencing cycles used in the in situ sequenc-</td>
</tr>
<tr>
<td></td>
<td>ing. This effectively determines the length of the barcodes to be used for se-</td>
</tr>
<tr>
<td></td>
<td>quencing.</td>
</tr>
<tr>
<td>rt_direction</td>
<td>String specifying from which direction the reverse transcription of the gRNA</td>
</tr>
<tr>
<td></td>
<td>spacer sequence will occur. Must be either &quot;5prime&quot; or &quot;3prime&quot;. &quot;5prime&quot; by</td>
</tr>
<tr>
<td></td>
<td>default.</td>
</tr>
</tbody>
</table>

Value
The original guideSet object with an additional column opsBarcode stored in mcols(guideSet). The column is a DNAStringSet storing the OPS barcode.

Author(s)
Jean-Philippe Fortin
addPamScores

Examples

data(guideSetExample, package="crisprDesign")
guideSetExample <- addOpsBarcodes(guideSetExample)

Description

Add PAM scores to a GuideSet object based on the CrisprNuclease object stored in the GuideSet object. PAM scores indicate nuclease affinity (recognition) to different PAM sequences. A score of 1 indicates a PAM sequence that is fully recognized by the nuclease.

Usage

addPamScores(object, ...)

## S4 method for signature 'GuideSet'
addPamScores(object)

## S4 method for signature 'PairedGuideSet'
addPamScores(object)

## S4 method for signature 'NULL'
addPamScores(object)

Arguments

object A GuideSet or a PairedGuideSet object.
...
Additional arguments, currently ignored.

Value

guideSet with an appended score_pam column in mcols(guideSet).

Author(s)

Jean-Philippe Fortin

Examples

# Using character vector as input:
data(enAsCas12a, package="crisprBase")
gs <- findSpacers("CCACATAGTGAACACCACGTCTCTATAGAATACAAAAATTTAGCCGGGTGTTA",
   canonical=FALSE,
   crisprNuclease=enAsCas12a)
gs <- addPamScores(gs)
Description

Add Pfam domains annotation to \texttt{GuideSet} object.

Usage

\begin{verbatim}
addPfamDomains(object, ...)  
## S4 method for signature 'GuideSet'  
addPfamDomains(object, pfamTable)  
## S4 method for signature 'PairedGuideSet'  
addPfamDomains(object, pfamTable)  
## S4 method for signature 'NULL'  
addPfamDomains(object)
\end{verbatim}

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A \texttt{GuideSet} object or a \texttt{PairedGuideSet} object.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments, currently ignored.</td>
</tr>
<tr>
<td>pfamTable</td>
<td>A \texttt{Dataframe} obtained using \texttt{preparePfamTable}.</td>
</tr>
</tbody>
</table>

Details

In order to call this function, the object must contain a gene annotation by calling first \texttt{addGeneAnnotation}.

Value

An updated object with a column \texttt{pfam} added to \texttt{geneAnnotation(object)}.

Author(s)

Jean-Philippe Fortin

See Also

See \texttt{preparePfamTable} to prepare the Pfam domain DataFrame object, and see \texttt{addGeneAnnotation} to add a gene annotation to the object.
addReinitiationFlag  

Add a logical flag for gRNAs leading to potential reinitiation

Description

Add a logical flag for gRNAs leading to potential reinitiation.

Usage

```r
addReinitiationFlag(
  guideSet, 
  tx_id, 
  grnaLocationUpperLimit = 150, 
  cdsCutoff = 0.2
)
```

Arguments

- `guideSet`  
  A `GuideSet` object.

- `tx_id`  
  String specifying Ensembl ID for the isoform transcript of interested. E.g. “ENST00000311936”.

- `grnaLocationUpperLimit`  
  Integer value specifying the number of nucleotides upstream of the start of the CDS in which to search for problematic gRNAs. Default value is 150. gRNAs beyond this value will not be flagged.

- `cdsCutoff`  
  Numeric value between 0 and 1 to specify the percentage of the CDS in which to search for problematic gRNAs. Default is 0.20. gRNAs beyond this value will not be flagged.

Value

The original object with an appended column `reinitiationFlag` with logical values. A TRUE value indicates a gRNA in proximity of a potential reinitiation site, and therefore should be avoided.

Author(s)

Jean-Philippe Fortin
addRepeats

Annotate a GuideSet object with repeat elements

Description
Add an annotation column to a GuideSet object that identifies spacer sequences overlapping repeat elements.

Usage
addRepeats(object, ...)

## S4 method for signature 'GuideSet'
addRepeats(object, gr.repeats = NULL, ignore.strand = TRUE)

## S4 method for signature 'PairedGuideSet'
addRepeats(object, gr.repeats = NULL, ignore.strand = TRUE)

## S4 method for signature 'NULL'
addRepeats(object)

Arguments
- **object**: A GuideSet object or a PairedGuideSet object.
- **...**: Additional arguments, currently ignored.
- **gr.repeats**: A GRanges object containing repeat elements regions.
- **ignore.strand**: Should gene strand be ignored when annotating? TRUE by default.

Value
guideSet with an inRepeats column appended in mcols(guideSet) that signifies whether the spacer sequence overlaps a repeat element.

Author(s)
Jean-Philippe Fortin, Luke Hoberecht

See Also
link{removeRepeats}.

Examples
data(guideSetExample, package="crisprDesign")
data(grRepeatsExample, package="crisprDesign")
guideSet <- addRepeats(guideSetExample,
gr.repeats=grRepeatsExample)
addRestrictionEnzymes  
*Restriction enzyme recognition sites in spacer sequences*

**Description**

Add restriction site enzymes annotation.

**Usage**

```r
addRestrictionEnzymes(object, ...)  
## S4 method for signature 'GuideSet'
addRestrictionEnzymes(
  object,
  enzymeNames = NULL,
  patterns = NULL,
  includeDefault = TRUE,
  flanking5 = "ACCG",
  flanking3 = "GTTT"
)

## S4 method for signature 'PairedGuideSet'
addRestrictionEnzymes(
  object,
  enzymeNames = NULL,
  patterns = NULL,
  includeDefault = TRUE,
  flanking5 = "ACCG",
  flanking3 = "GTTT"
)

## S4 method for signature 'NULL'
addRestrictionEnzymes(object)
```

**Arguments**

- `object`  
  A `GuideSet` or a `PairedGuideSet` object.

- `...`  
  Additional arguments, currently ignored.

- `enzymeNames`  
  Character vector of enzyme names.

- `patterns`  
  Optional named character vector for custom restriction site patterns. Vector names are treated as enzymes names. See example.

- `includeDefault`  
  Should commonly-used enzymes be included? TRUE by default.

- `flanking5, flanking3`  
  Character string indicating the 5’ or 3’ flanking sequence, respectively, of the spacer sequence in the lentivial vector.
addRestrictionEnzymes

Details

Restriction enzymes are often used for cloning purpose during the oligonucleotide synthesis of gRNA lentiviral constructs. Consequently, it is often necessary to avoid restriction sites of the used restriction enzymes in and around the spacer sequences. addRestrictionEnzymes allows for flagging problematic spacer sequences by searching for restriction sites in the [flanking5][spacer][flanking3] sequence.

The following enzymes are included when includeDefault=TRUE: EcoRI, KpnI, BsmBI, BsaI, BbsI, PacI, and MluI.

Custom recognition sequences in patterns may use the IUPAC nucleotide code, excluding symbols indicating gaps. Avoid providing enzyme names in patterns that are already included by default (if includeDefault=TRUE) or given by enzymeNames. Patterns with duplicated enzyme names will be silently ignored, even if the recognition sequence differs. See example.

Value

Adds a DataFrame indicating whether cutting sites for the specified enzymes are found in the gRNA cassette (flanking sequences + spacer sequences).

Author(s)

Jean-Philippe Fortin, Luke Hoberecht

See Also

text|enzymeAnnotation to retrieve existing enzyme annotation from a GuideSet object.

Examples

data(SpCas9, package="crisprBase")
seq <- c("ATTTCCGGAGGCGAATTCGGCGGGAGGAGGAAGACCGG")
guideSet <- findSpacers(seq, crisprNuclease=SpCas9)

# Using default enzymes:
guideSet <- addRestrictionEnzymes(guideSet)

# Using custom enzymes:
guideSet <- addRestrictionEnzymes(guideSet,
  patterns=c("GGTCCAA",
  "GGTCG"))

# Avoid duplicate enzyme names
guideSet <- addRestrictionEnzymes(guideSet,
  patterns=c("EcoRI="GANTC")) # ignored
addSequenceFeatures

Add spacer sequence feature annotation columns to a GuideSet object

Description

Add spacer sequence feature annotation columns, such as GC content, homopolymers, and hairpin predictions, to a GuideSet object.

Usage

addSequenceFeatures(object, ...)

## S4 method for signature 'GuideSet'
addSequenceFeatures(
  object,
  addHairpin = FALSE,
  backbone = "AGGCTAGTCCGT",
  tp53 = TRUE,
  ...
)

## S4 method for signature 'PairedGuideSet'
addSequenceFeatures(
  object,
  addHairpin = FALSE,
  backbone = "AGGCTAGTCCGT",
  tp53 = TRUE,
  ...
)

## S4 method for signature 'NULL'
addSequenceFeatures(object, ...)

Arguments

object A GuideSet or a PairedGuideSet object.

... Additional arguments, currently ignored.

addHairpin Whether to include predicted hairpin formation via sequence complementarity. FALSE by default. See details.

backbone Backbone sequence in the guide RNA that is susceptible to hairpin formation with a complementary region in the spacer sequence.

tp53 Should TP53-related toxicity features be added? TRUE by default. See details.
Details

The `addHairpin` argument set to `TRUE` will indicate which spacers are predicted to form internal hairpins. Such hairpins can happen when there is a palindromic sequence within the spacer having arms of >=4nt and >=50% GC content, and are separated by a loop of >=4nt. Backbone hairpin formation is predicted when the spacer and backbone share a complementary sequence of >=5nt and >=50% GC content. The argument `backbone` allows users to specify the vector backbone sequence directly downstream of the spacer sequence.

The `tp53` argument set to `TRUE` will add sequence-based features that have been reported to make SpCas9 gRNAs toxic for cells with wildtype TP53 (see https://doi.org/10.1038/s41467-022-32285-1). Currently, only one feature is reported and consists of the extended NNGG PAM sequence (1 nucleotide + PAM sequence) for SpCas9. gRNAs with extended CNGG PAM sequences, and in particular CCGG, should be avoided.

Value

The original object with the following columns appended to `mcols(object)`:

- `percentGC` — percent GC content
- `polyA`, `polyC`, `polyG`, `polyT` — presence of homopolymers of 4nt or longer
- `selfHairpin` — prediction of hairpin formation within the spacer sequence via self-complementarity if `addHairpin` is `TRUE`.
- `backboneHairpin` — prediction of hairpin formation with the backbone sequence via complementarity if `addHairpin` is `TRUE`.
- `NNGG` — extended PAM sequence for SpCas9 if `tp53` is `TRUE` corresponding to one nucleotide upstream of the PAM sequence followed by the PAM sequence itself.

Examples

```r
custom_seq <- c("ATTTCCGGAGGCGGAGAGGCGGGAGGAGCG")
data(SpCas9, package="crisprBase")
guideSet <- findSpacers(custom_seq, crisprNuclease=SpCas9)
guideSet <- addSequenceFeatures(guideSet)
```

---

**addSNPAnnotation**

Add SNP annotation to a `GuideSet` object

Description

Add SNP annotation to a `GuideSet` object. Only available for sgRNAs designed for human genome.
Usage

```r
addSNPAnnotation(object, ...)  
## S4 method for signature 'NULL'
addGeneAnnotation(object)
```

```r
## S4 method for signature 'GuideSet'
addSNPAnnotation(object, vcf, maf = 0.01)
```

```r
## S4 method for signature 'PairedGuideSet'
addSNPAnnotation(object, vcf, maf = 0.01)
```

```r
## S4 method for signature 'NULL'
addSNPAnnotation(object)
```

Arguments

- `...`: Additional arguments, currently ignored.
- `vcf`: Either a character string specifying a path to a VCF file or connection, or a VCF object.
- `maf`: Minimum minor allele frequency to report (for at least one source among 1000 Genomes and TOPMED). Must be between 0 and 1 (exclusive).

Details

The different columns stored in `mcols(guideSet)["snps"]` are:

- `ID`: sgRNA ID.
- `rs`: Reference SNP cluster ID (e.g. rs17852242)
- `rs_site`: Genomic coordinate of the SNP.
- `rs_site_rel`: Position of SNP relative to the PAM site.
- `allele_ref`: DNAString specifying the SNP reference allele.
- `allele_minor`: DNAString specifying the SNP minor allele.
- `MAF_1000G`: Minor allele frequency in the 1000 Genomes project.
- `MAF_TOPMED`: Minor allele frequency in the TOPMed project.
- `type`: Type of SNP ("ins": insertion, "del": deletion).
- `length`: Length of SNP in nucleotides.

Value

`guideSet` appended with `hasSNP` column and `snps` list-column, both stored in `mcols(guideSet)`.

See Also

- `link<snpAnnotation>` to retrieve an existing SNP annotation stored in a `GuideSet` object. See details section for a description of the different columns.
Examples

```r
vcf <- system.file("extdata",
    file="common_snps_dbsnp151_example.vcf.gz",
    package="crisprDesign")
data(guideSetExample, package="crisprDesign")
guideSet <- addSNPAnnotation(guideSetExample, vcf=vcf)
```

addSpacerAlignments

*Functions for finding and characterizing on- and off-targets of spacer sequences.*

Description

Functions for finding and characterizing on- and off-targets of spacer sequences.

Usage

```r
addSpacerAlignments(object, ...)
addSpacerAlignmentsIterative(object, ...)
```

## S4 method for signature 'GuideSet'

```r
addSpacerAlignmentsIterative(
    object,
    aligner = c("bowtie", "bwa", "biostrings"),
    colname = "alignments",
    addSummary = TRUE,
    txObject = NULL,
    tssObject = NULL,
    custom_seq = NULL,
    aligner_index = NULL,
    bsgenome = NULL,
    n_mismatches = 0,
    all_alignments = FALSE,
    canonical = TRUE,
    standard_chr_only = FALSE,
    both_strands = TRUE,
    anchor = c("cut_site", "pam_site"),
    annotationType = c("gene_symbol", "gene_id"),
    tss_window = NULL,
    alignmentThresholds = c(n0 = 5, n1 = 100, n2 = 100, n3 = 1000, n4 = 1000)
)
```

## S4 method for signature 'PairedGuideSet'

```r
addSpacerAlignmentsIterative(
    object,
```
addSpacerAlignments

aligner = c("bowtie", "bwa", "biostrings"),
colname = "alignments",
addSummary = TRUE,
tXObject = NULL,
tssObject = NULL,
custom_seq = NULL,
aligner_index = NULL,
bsgenome = NULL,
n_mismatches = 0,
all_alignments = FALSE,
canonical = TRUE,
standard_chr_only = FALSE,
both_strands = TRUE,
anchor = c("cut_site", "pam_site"),
annotationType = c("gene_symbol", "gene_id"),
tss_window = NULL,
alignmentThresholds = c(n0 = 5, n1 = 100, n2 = 100, n3 = 1000, n4 = 1000)
)

## S4 method for signature 'NULL'
addSpacerAlignmentsIterative(object)

## S4 method for signature 'GuideSet'
addSpacerAlignments(
  object,
  aligner = c("bowtie", "bwa", "biostrings"),
colname = "alignments",
addSummary = TRUE,
tXObject = NULL,
tssObject = NULL,
custom_seq = NULL,
aligner_index = NULL,
bsgenome = NULL,
n_mismatches = 0,
n_max_alignments = 1000,
all_alignments = TRUE,
canonical = TRUE,
standard_chr_only = FALSE,
both_strands = TRUE,
anchor = c("cut_site", "pam_site"),
annotationType = c("gene_symbol", "gene_id"),
tss_window = NULL)

## S4 method for signature 'PairedGuideSet'
addSpacerAlignments(
  object,
  aligner = c("bowtie", "bwa", "biostrings"),

colname = "alignments",
addSummary = TRUE,
tXObject = NULL,
tssObject = NULL,
custom_seq = NULL,
aligner_index = NULL,
bsgenome = NULL,
n_mismatches = 0,
n_max_alignments = 1000,
all_alignments = FALSE,
canonical = TRUE,
standard_chr_only = FALSE,
both_strands = TRUE,
anchor = c("cut_site", "pam_site"),
annotationType = c("gene_symbol", "gene_id"),
tss_window = NULL
)

## S4 method for signature 'NULL'
addSpacerAlignments(object)

getSpacerAlignments(
  spacers,
  aligner = c("bowtie", "bwa", "biostrings"),
custom_seq = NULL,
aligner_index = NULL,
bsgenome = NULL,
n_mismatches = 0,
n_max_alignments = 1000,
all_alignments = TRUE,
crisprNuclease = NULL,
canonical = TRUE,
standard_chr_only = FALSE,
both_strands = TRUE
)

Arguments

object               A GuideSet object or a PairedGuideSet object.
...                  Additional arguments, currently ignored.
aligner              Which genomic alignment method should be used? Must be one of "bowtie", "bwa", and "biostrings". "bowtie" by default. Note that "bwa" is not available for Windows machines.
colname              String specifying the column name storing the alignments in mcols(guideSet). "alignments" by default.
addSummary           Should summary columns be added to guideSet? TRUE by default.
tXObject             A TxDb object or a GRangesList object obtained using TxDb2GRangesList for annotating on-target and off-target alignments using gene annotation.
addSpacerAlignments

tssObject A GRanges object specifying TSS coordinates.
custom_seq Optional string specifying the target DNA sequence for the search space. This will limit the off-target search to the specified custom sequence.
aligner_index String specifying bowtie or BWA index. Must be provided when aligner is either "bowtie" or "bwa".
bsgenome A BSgenome object from which to extract sequences if a GRanges object is provided as input.
n_mismatches Maximum number of mismatches permitted between guide RNA and genomic DNA.
all_alignments Should all all possible alignments be returned? FALSE by default.
canonical TRUE returns only those alignments having canonical PAM sequences; FALSE returns alignments having canonical or noncanonical PAM sequences; NA returns all alignments regardless of their PAM sequence.
standard_chr_only Should only standard chromosomes be considered? If TRUE, the function will attempt to remove scaffold sequences automatically. FALSE by default.
both_strands When custom_seq is specified, should both strands be considered? TRUE by default.
anchor The position within the protospacer as determined by CrisprNuclease to use when annotating with overlapping gene regions.
annotationType Gene identifier to return when annotating alignments with gene and/or promoter overlaps. Corresponding txObject or tssObject argument must have mcol column name for selected type.
tss_window Window size of promoters upstream of gene TSS to search for overlap with spacer sequence. Must be a numeric vector of length 2: upstream limit and downstream limit. Default is c(-500, 500), which includes 500bp upstream and downstream of the TSS.
alignmentThresholds Named numeric vector of the maximum on-target alignments tolerated for addSpacerAlignmentsIterative. Thresholds not provided will take default values.
n_max_alignments Maximum number of alignments to report by bowtie for each spacer. Effectively set to Inf when allPossible is TRUE.
spacers Character vector of gRNA spacer sequences. All sequences must be equal in length.
crisprNuclease A CrisprNuclease object.

Details

The columns stored in mcols(guideSet)[["alignments"]]) are:

- spacer Spacer sequence of the query gRNA.
- protospacer Protospacer sequence in the target DNA.
- pam PAM sequence.
• **pan_site** PAM site of the found protospacer.
• **n_mismatches** Integer value specifying the number of nucleotide mismatches between the gRNA spacer sequence and the protospacer sequence found in the genome or custom sequence.
• **canonical** Whether the PAM sequence of the found protospacer sequence is canonical.
• **cute_site** Cut site of the found protospacer.

The following columns are also stored when a txObject is provided:

• **cds** Character vector specifying gene names of CDS overlapping the found protospacer sequence.
• **fiveUTRs** Character vector specifying gene names of 5’UTRs overlapping the found protospacer sequence.
• **threeUTRs** Character vector specifying gene names of 3’UTRs overlapping the found protospacer sequence.
• **exons** Character vector specifying gene names of exons overlapping the found protospacer sequence.
• **introns** Character vector specifying gene names of introns overlapping the found protospacer sequence.
• **intergenic** Character vector specifying the nearest gene when the found protospacer sequence is not located in a gene.
• **intergenic_distance** Distance in base pairs from the nearest gene when the found protospacer sequence is not located in a gene.

The following columns are also stored when a tssObject is provided:

• **promoters** Character vector specifying gene names of promoters, as defined by tss_window relative to the gene TSS, overlapping the found protospacer sequence.

Value

`getSpacerAlignments` returns a `GRanges` object storing spacer alignment data, including genomic coordinates, spacer and PAM sequences, and position of mismatches relative to `pan_site`.

`addSpacerAlignments` is similar to `getSpacerAlignments`, with the addition of adding the alignment data to a list-column in `mcols(guideSet)` specified by `colname`.

`addSpacerAlignmentsIterative` is similar to `addSpacerAlignments`, except that it avoids finding alignments for spacer sequences that have a large number of on-targets and/or off-targets to speed up the off-target search. The parameters `n0_max`, `n1_max` and `n2_max` specify the maximum number of on-targets (`n0`) and off-targets (`n1` for 1-mismatch off-targets, and `n2` for 2-mismatch off-targets) tolerated before the algorithm stops finding additional off-targets for spacer sequences that exceed those quotas.

Author(s)

Jean-Philippe Fortin, Luke Hoberecht
addTssAnnotation

Add TSS context annotation to a GuideSet object

Description

Add transcription start site (TSS) context annotation to spacer sequences stored in a GuideSet object.

Usage

addTssAnnotation(object, ...)

## S4 method for signature 'GuideSet'
addTssAnnotation(
  object,
  tssObject,
  anchor = c("cut_site", "pam_site"),
  tss_window = NULL,
  ignore.strand = TRUE
)

## S4 method for signature 'PairedGuideSet'
addTssAnnotation(
  object,
  tssObject,
  anchor = c("cut_site", "pam_site"),
  tss_window = NULL,
  ignore.strand = TRUE
)

## S4 method for signature 'NULL'
addTssAnnotation(object)

Arguments

- **object**: A GuideSet object or a PairedGuideSet object.
- **...**: Additional arguments, currently ignored.
- **tssObject**: A GRanges object containing TSS coordinates and annotation.
- **anchor**: A character string specifying which gRNA-specific coordinate to use (cut_site or pam_site) when searching for overlapping TSS regions. "cut_site" by default.
- **tss_window**: A numeric vector of length 2 establishing the window size of the genomic region around the TSS to include as the "TSS region". The values set the upstream and downstream limits, respectively. The default is c(-500, 500), which includes 500bp upstream (note the negative value) and downstream of the TSS.
- **ignore.strand**: If TRUE (default), includes annotation for gRNAs irrespective of their target strand. Otherwise, only gRNAs targeting the gene strand will be annotated.

Details

mcols(guideSet)[["tssAnnotation"]] includes all columns from mcols(tssObject) in addition to the columns described below.

- **chr**: gRNA chromosome name.
- **anchor_site**: Genomic coordinate used to search for overlapping TSS regions.
- **strand**: Strand the gRNA is located on.
- **tss_id**: The ID for the TSS in tssObject, if present.
- **tss_strand**: Strand the TSS is located on, as provided in tssObject.
- **tss_pos**: Genomic coordinate of the TSS, as provided in tssObject.
- **dist_to_tss**: Distance (in nucleotides) between the gRNA anchor_site and tss_pos. Negative values indicate gRNA targets upstream of the TSS.

Value

A GuideSet object with a tssAnnotation list column stored in mcols(guideSet). See details section for descriptions of TSS annotation columns.
addTxTable

Add a gene-specific transcript table to a GuideSet object.

Description

Add a gene-specific transcript table to a GuideSet object.

Usage

```
addTxTable(guideSet, gene_id, txObject, valueColumn = "percentCDS")
```

Arguments

- `guideSet` A GuideSet object or a PairedGuideSet object.
- `gene_id` String specifying gene ID.
- `txObject` A TxDb object or a GRangesList object obtained using TxDb2GRangesList to provide a gene model annotation.
- `valueColumn` String specifying column in geneAnnotation(guideSet) to use as values in the output transcript table.

Value

A GuideSet object with a "txTable" DataFrame stored in mcols(guideSet). The entries in the DataFrame correspond to the values specified by valueColumn. Rows correspond to gRNAs in the GuideSet, columns correspond to all transcripts found in txObject for gene specified by gene_id.
Author(s)
Jean-Philippe Fortin

See Also
addGeneAnnotation to add gene annotation.

Examples
if (interactive()){
  data(guideSetExample, package="crisprDesign")
  data(grListExample, package="crisprDesign")
  guideSet <- addGeneAnnotation(guideSetExample,
    txObject=grListExample)
  guideSet <- addTxTable(guideSet,
    gene_id="ENSG00000120645",
    txObject=grListExample)

  guideSet$txTable
}

completeSpacers  Get complete spacer information

Description
These functions serve to "fill-in-the-blank" for spacers lacking information.

Usage
getPAMSequence(chr, pam_site, strand, crisprNuclease = NULL, bsgenome = NULL)

getSpacerSequence(
  chr,
  pam_site,
  strand,
  crisprNuclease = NULL,
  bsgenome = NULL,
  spacerLen = NULL
)

getPAMSiteFromStartAndEnd(
  start = NULL,
  end = NULL,
  strand,
  crisprNuclease = NULL,
  spacerLen = NULL
)
Arguments

- **chr**: The chromosome in which the protospacer sequence is located.
- **pam_site**: Coordinate of the first nucleotide of the PAM sequence.
- **strand**: Either "+" or "-".
- **crisprNuclease**: A `CrisprNuclease` object.
- **bsgenome**: A `BSgenome` object.
- **spacerLen**: Spacer sequence length. If NULL, the information is obtained from `crisprNuclease`.
- **start**: Coordinate of the first nucleotide of the spacer sequences. Must be always less than `end`.
- **end**: Coordinate of the last nucleotide of the spacer sequence. Must be always greater than `start`.

Details

Functions that return coordinates (getPAMSite, getCutSite, getSpacerRanges) do not check whether coordinates exceed chromosomal lengths.

The start and end coordinates of a genomic range is strand-independent, and always obeys `start <= end`.

Value

A numeric or character vector, depending on the function.

getPAMSequence returns a character vector of PAM sequences.

g SpacerSequence returns a character vector of spacer sequences.

Examples

```r
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg38")){
library(BSgenome.Hsapiens.UCSC.hg38)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
dat <- data.frame(chr='chr4', start=1642343, strand='+')
dat$pam_site <- getPAMSiteFromStartAndEnd(start=dat$start, strand=dat$strand)
dat$pam <- getPAMSequence(chr=dat$chr, 
    pam_site=dat$pam_site, 
    strand=dat$strand, 
    bsgenome=bsgenome)
dat$spacer <- getSpacerSequence(chr=dat$chr, 
    pam_site=dat$pam_site, 
    strand=dat$strand, 
    bsgenome=bsgenome)
}
```
**convertToMinMaxGRanges**

Convert a GuideSet object into a GRanges containing the range of all targeting gRNAs.

**Description**

Convert a GuideSet object into a GRanges object containing the minimum and maximum coordinates for all targeting gRNAs.

**Usage**

```r
cvtColoroMinMaxGRanges(guideSet, anchor = c("cut_site", "pam_site"))
```

**Arguments**

- `anchor`: A character string specifying which gRNA-specific coordinate to use (`cut_site` or `pam_site`) when defining the min and max coordinates of GuideSet object.

**Value**

A GRanges object with start and end coordinates corresponding to the minimum and maximum coordinates of the GuideSet object sites defined by anchor.

**Author(s)**

Jean-Philippe Fortin, Luke Hoberecht

**Examples**

```r
data(guideSetExample, package="crisprDesign")
gr <- convertToMinMaxGRanges(guideSetExample)
```

---

**convertToProtospacerGRanges**

Convert PAM site coordinates to protospacer start and end coordinates.

**Description**

Convert PAM site coordinates to protospacer start and end coordinates.

**Usage**

```r
cvtColoroProtospacerGRanges(guideSet)
```
**Arguments**

guideSet A GuideSet object.

**Value**

A GuideSet object with start and end coordinates corresponding to the start and end coordinates of the protospacer sequences.

**Author(s)**

Jean-Philippe Fortin

**Examples**

data(guideSetExample, package="crisprDesign")
gr <- convertToProtospacerGRanges(guideSetExample)

---

**Description**

An S4 class to store CRISPR gRNA sequences with modular annotations.

**Usage**

crisprNuclease(object, ...)
targetOrigin(object, ...)
customSequences(object, ...)
bsgenome(object, ...)
spacers(object, ...)
protospacers(object, ...)
pamSites(object, ...)
snps(object, ...)
algnments(object, ...)
onTargets(object, ...)

**crisprNuclease**

An S4 class to store CRISPR gRNA sequences with modular annotations.
offTargets(object, ...)
geneAnnotation(object, ...)
tssAnnotation(object, ...)
enzymeAnnotation(object, ...)
editedAlleles(object, ...)
txTable(object, ...)
exonTable(object, ...)
tssAnnotation(object) <- value
geneAnnotation(object) <- value
enzymeAnnotation(object) <- value
snps(object) <- value
alignments(object) <- value
addCutSites(object, ...)

GuideSet(
  ids = NA_character_,
  protospacers = NA_character_,
  pams = NULL,
  seqnames = NA_character_,
  pam_site = 0L,
  strand = "*",
  CrisprNuclease = NULL,
  targetOrigin = c("bsgenome", "customSequences"),
  bgencode = NULL,
  customSequences = NULL,
  ...
  seqinfo = NULL,
  seqlengths = NULL
)

## S4 method for signature 'GuideSet'
targetOrigin(object)

## S4 method for signature 'GuideSet'
customSequences(object)
### S4 method for signature 'GuideSet'
bsgenome(object)

### S4 method for signature 'GuideSet'
crisprNuclease(object)

### S4 method for signature 'GuideSet'
spacers(object, as.character = FALSE, returnAsRna = FALSE)

### S4 method for signature 'GuideSet'
pams(object, as.character = FALSE, returnAsRna = FALSE)

### S4 method for signature 'GuideSet'
pamSites(object)

### S4 method for signature 'GuideSet'
cutSites(object)

### S4 method for signature 'GuideSet'
addCutSites(object)

### S4 method for signature 'GuideSet'
protospacers(
    object,
    as.character = FALSE,
    include.pam = FALSE,
    returnAsRna = FALSE
)

### S4 method for signature 'GuideSet'
spacerLength(object)

### S4 method for signature 'GuideSet'
prototypeSequence(object)

### S4 method for signature 'GuideSet'
pamLength(object)

### S4 method for signature 'GuideSet'
pamSide(object)

### S4 method for signature 'GuideSet'
snps(object, unlist = TRUE, use.names = TRUE)

### S4 method for signature 'GuideSet'
alignments(object, columnName = "alignments", unlist = TRUE, use.names = TRUE)
## S4 method for signature 'GuideSet'
onTargets(object, columnName = "alignments", unlist = TRUE, use.names = TRUE)

## S4 method for signature 'GuideSet'
offTargets(
  object,
  columnName = "alignments",
  max.mismatches = Inf,
  unlist = TRUE,
  use.names = TRUE
)

## S4 replacement method for signature 'GuideSet'
alignments(object) <- value

## S4 replacement method for signature 'GuideSet'
geneAnnotation(object) <- value

## S4 replacement method for signature 'GuideSet'
tssAnnotation(object) <- value

## S4 replacement method for signature 'GuideSet'
enzymeAnnotation(object) <- value

## S4 replacement method for signature 'GuideSet'
snps(object) <- value

## S4 method for signature 'GuideSet'
geneAnnotation(
  object,
  unlist = TRUE,
  gene_id = NULL,
  tx_id = NULL,
  gene_symbol = NULL,
  use.names = TRUE
)

## S4 method for signature 'GuideSet'
editedAlleles(object, unlist = TRUE, use.names = TRUE)

## S4 method for signature 'GuideSet'
tssAnnotation(
  object,
  unlist = TRUE,
  gene_id = NULL,
  gene_symbol = NULL,
  use.names = TRUE
)
## S4 method for signature 'GuideSet'
enzymeAnnotation(object, unlist = TRUE, use.names = TRUE)

## S4 method for signature 'GuideSet'
txTable(object, unlist = TRUE, use.names = TRUE)

## S4 method for signature 'GuideSet'
exonTable(object, unlist = TRUE, use.names = TRUE)

### Arguments

- `object` | GuideSet object.
- `...` | Additional arguments for class-specific methods
- `value` | Object to replace with
- `ids` | Character vector of unique gRNA ids. The ids can be anything, as long as they are unique.
- `protospacers` | Character vector of protospacers sequences.
- `pams` | Character vector of PAM sequences.
- `seqnames` | Character vector of chromosome names.
- `pam_site` | Integer vector of PAM site coordinates.
- `strand` | Character vector of gRNA strand. Only accepted values are "+" and ".-".
- `CrisprNuclease` | CrisprNuclease object.
- `targetOrigin` | String specifying the origin of the DNA target. Must be either 'bsgenome' or 'customSequences'.
- `bsgenome` | BSgenome object or string specifying BSgenome package name. Must be specified when `targetOrigin` is set to "bsgenome".
- `customSequences` | DNAStringSet object. Must be specified when `targetOrigin` is set to "customSequences".
- `seqinfo` | A Seqinfo object containing information about the set of genomic sequences present in the target genome.
- `seqlengths` | NULL, or an integer vector named with levels(seqnames) and containing the lengths (or NA) for each level in levels(seqnames).
- `as.character` | Should sequences be returned as a character vector? FALSE by default, in which case sequences are returned as a DNAStringSet.
- `returnAsRna` | Should the sequences be returned as RNA instead of DNA? FALSE by default.
- `include.pam` | Should PAM sequences be included? FALSE by default.
- `unlist` | Should the annotation be returned as one table instead of a list? TRUE by default.
- `use.names` | Whether to include spacer IDs as (row)names (TRUE), or as a separate column (FALSE).
- `columnName` | Name of the column storing the alignments annotation to be retrieved.
max_mismatches  What should be the maximum number of mismatches considered for off-targets? Inf by default.
gene_id  Character vector of Ensembl gene IDs to subset gene annotation data by. If NULL (default), all genes are considered.
tx_id  Character vector of Ensembl transcript IDs to subset gene annotation data by. If NULL (default), all transcripts are considered.
gene_symbol  Character vector of gene symbols to subset gene annotation data by. If NULL (default), all genes are considered.

Value

A GuideSet object.

Functions

- GuideSet(): Create a GuideSet object

Constructors

Use the constructor link{GuideSet} to create a GuideSet object.

Accessors

- crisprNuclease: To get CrisprNuclease object used to design gRNAs.
- spacers: To get spacer sequences.
- protospacers: To get protospacer sequences.
- spacerLength: To get spacer length.
- pams: To get PAM sequences.
- pamSites: To get PAM site coordinates.
- pamLength: To get PAM length.
- pamSide: To return the side of the PAM sequence with respect to the protospacer sequence.
- prototypeSequence: To get a prototype protospacer sequence.
- cutSites: To get cut sites.
- alignments: To get genomic alignments annotation.
- onTargets: To get on-target alignments annotation
- offTargets: To get off-target alignments annotation
- snps: To get SNP annotation.
- geneAnnotation: To get gene annotation.
- tssAnnotation: To get TSS annotation.
- enzymeAnnotation: To get restriction enzymes annotation.
- editedAlleles: To get edited alleles annotation.
Examples

protospacers <- c("AGGTCGTGTGCTGGGGGGG", 
                  "AGGTCGTGTGCTGGGGGGGG")
pams <- c("AGG", "CGG")
pam_site=c(10,11)
seqnames="chr7"
data(SpCas9, package="crisprBase")
CrisprNuclease <- SpCas9
strand=c("+", "-")
ids <- paste0("grna_", seq_along(protospacers))
gr <- GuideSet(ids=ids,
               protospacers=protospacers,
               pams=pams,
               seqnames=seqnames,
               CrisprNuclease=CrisprNuclease,
               pam_site=pam_site,
               strand=strand,
               targetOrigin="customSequences",
               customSequences=protospacers)

designCompleteAnnotation

One-step gRNA design and annotation function

Description

One-step gRNA design and annotation function to facilitate the design and generation of genome-wide gRNA databases for a combination of parameters such as nuclease, organism, and CRISPR modality.

Usage

designCompleteAnnotation(
    queryValue = NULL,
    queryColumn = "gene_id",
    featureType = "cds",
    modality = c("CRISPRko", "CRISPRa", "CRISPRi", "CRISPRkd"),
    bsgenome = NULL,
    bowtie_index = NULL,
    vcf = NULL,
    crisprNuclease = NULL,
    tssObject = NULL,
    txObject = NULL,
    grRepeats = NULL,
    scoring_methods = NULL,
    tss_window = NULL,
    n_mismatches = 3,
max_mm = 2,
canonical_ontarget = TRUE,
canonical_offtarget = FALSE,
all_alignments = TRUE,
fastaFile = NULL,
chromatinFiles = NULL,
geneCol = "gene_symbol",
conservationFile = NULL,
nucExtension = 9,
binaries = NULL,
canonicalIsoforms = NULL,
pfamTable = NULL,
verbose = TRUE
)

Arguments

queryValue  Vector specifying the value(s) to search for in txObject[[featureType]][[queryColumn]].
queryColumn  Character string specifying the column in txObject[[featureType]] to search for queryValue(s).
featureType  For CRISPRko, string specifying the type of genomic feature to use to design gRNAs. Must be one of the following: "transcripts", "exons", "cds", "fiveUTRs", "threeUTRs" or "introns". The default is "cds".
modality  String specifying the CRISPR modality. Must be one of the following: "CRISPRko", "CRISPRa", "CRISPRi" or "CRISPRkd". CRISPRkd is reserved for DNA-targeting nucleases only such as CasRx.
bsgenome  A BSgenome object from which to extract sequences if a GRanges object is provided as input.
bowtie_index  String specifying path to a bowtie index.
vcf  Either a character string specifying a path to a VCF file or connection, or a VCF object.
crisprNuclease  A CrisprNuclease object.
tssObject  A GRanges object specifying TSS coordinates.
txDBobject  A TxDb object or a GRangesList object obtained using TxDb2GRangesList for annotating on-target and off-target alignments using gene annotation.
grRepeats  A GRanges object containing repeat elements regions.
scoring_methods  Character vector to specify which on-target scoring methods should be calculated. See crisprScore package to obtain available methods.
tss_window  Vector of length 2 specifying the start and coordinates of the CRISPRa/CRISPRi target region with respect to the TSS position.
n_mismatches  Maximum number of mismatches permitted between guide RNA and genomic DNA.
max_mm  The maximimum number of mismatches between a spacer and an off-target to be accepted when calculating aggregate off-target scores. 2 by default.
canonical_on_target
Should only canonical PAM sequences be searched for designing gRNAs? TRUE by default.

canonical_off_target
Should only canonical PAM sequences be searched during the off-target search? TRUE by default.

all_alignments
Should all possible alignments be returned? TRUE by default.

fasta_file
String specifying fasta file of the hg38 genome. Only used for CRISPRa/i modality with hg38 genome and SpCas9 nuclease. This is needed to generate the CRISPRai scores. See the function addCrispraiScores for more details.

chromatin_files
Named character vector of length 3 specifying BigWig files containing chromatin accessibility data. Only used for CRISPRa/i modality with hg38 genome and SpCas9 nuclease. This is needed to generate the CRISPRai scores. See the function addCrispraiScores for more details.

gene_col
String specifying the column in the tssObject to be used to specify the gene name for the addCrispraiScores function. "gene_symbol" by default.

conservation_file
String specifying the BigWig file containing conservation scores.

nuc_extension
Number of nucleotides to include on each side of the cut site to calculate the conservation score. 9 by default. The region will have \((2\times\text{nucExtension} + 1)\) nucleotides in total.

binaries
Named list of paths for binaries needed for CasRx-RF. Names of the list must be "RNAfold", "RNAhybrid", and "RNAplfold". Each list element is a string specifying the path of the binary. If NULL (default), binaries must be available on the PATH.

canonicalIsoforms
Optional data.frame with 2 columns detailing Ensembl canonical isoforms. First column must be named "tx_id", and second column must be named "gene_id", corresponding to Ensembl transcript and gene ids, respectively.

pfam_table
A DataFrame obtained using preparePfamTable.

verbose
Should messages be printed?

Value
A GuideSet object.

Author(s)
Jean-Philippe Fortin
designOpsLibrary  Design gRNA library for optical pooled screening

Description

Design gRNA library for optical pooled screening

Usage

designOpsLibrary(
  df,
  n_guides = 4,
  gene_field = "gene",
  min_dist_edit = 2,
  dist_method = c("hamming", "levenshtein"),
  splitByChunks = FALSE
)

Arguments

df  data.frame containing information about candidate gRNAs from which to build the OPS library. See details.

n_guides  Integer specifying how many gRNAs per gene should be selected. 4 by default.
gene_field  String specifying the column in df specifying gene names.
min_dist_edit  Integer specifying the minimum distance edit required for barcodes to be considered dissimilar. Barcodes that have edit distances less than the min_dist_edit will not be included in the library. 2 by default.

dist_method  String specifying distance method. Must be either "hamming" (default) or "levenshtein".

splitByChunks  Should distances be calculated in a chunk-wise manner? FALSE by default. Highly recommended when the set of query barcodes is large to reduce memory footprint.

Value

A subset of the df containing the gRNAs selected for the OPS library.

Author(s)

Jean-Philippe Fortin
findSpacerPairs

Examples

data(guideSetExample, package="crisprDesign")
guideSet <- unique(guideSetExample)
guideSet <- addOpsBarcodes(guideSet)
guideSet <- guideSet[1:200]

df <- data.frame(ID=names(guideSet),
                 spacer=spacers(guideSet, as.character=TRUE),
                 opsBarcode=as.character(guideSet$opsBarcode))

# Creating mock gene:
df$gene <- rep(paste0("gene",1:10),each=20)
df$rank <- rep(1:20,10)
opsLib <- designOpsLibrary(df)

findSpacerPairs

Find pairs of CRISPR gRNA spacers from a pair of genomic regions.

Description

Returns all possible, valid gRNA sequences for a given CRISPR nuclease from either a GRanges object or a set of sequence(s) contained in either a DNAStringSet, DNAString or character vector of genomic sequences.

Usage

findSpacerPairs(
  x1,
  x2,
  sortWithinPair = TRUE,
  pamOrientation = c("all", "out", "in"),
  minCutLength = NULL,
  maxCutLength = NULL,
  crisprNuclease = NULL,
  bgene = NULL,
  canonical = TRUE,
  both_strands = TRUE,
  spacer_len = NULL,
  strict_overlap = TRUE,
  remove_ambiguities = TRUE
)

Arguments

x1 Either a GRanges, a DNAStringSet, or a DNAString object, or a character vector of genomic sequences. This specifies the sequence space from which gRNAs in position 1 of the pairs will be designed. Alternatively, a GuideSet object can be provided.
Either a GRanges, a DNAStringSet, or a DNAString object, or a character vector of genomic sequences. This specifies the sequence space from which gRNAs in position 2 of the pairs will be designed. Alternatively, a GuideSet object can be provided.

Should gRNAs be sorted by chr and position within a pair? TRUE by default.

String specifying a constraint on the PAM orientation of the pairs. Should be either "all" (default), "out" (for the so-called PAM-out orientation) or "in" (for PAM-in orientation).

Integer specifying the minimum cut length allowed (distance between the two cuts) induced by the gRNA pair. If NULL (default), the argument is ignored. Note that this parameter is only applicable for pairs of gRNAs targeting the same chromosome.

Integer specifying the maximum cut length allowed (distance between the two cuts) induced by the gRNA pair. If NULL (default), the argument is ignored. Note that this parameter is only applicable for pairs of gRNAs targeting the same chromosome.

A CrisprNuclease object.

A BSgenome object from which to extract sequences if x is a GRanges object.

Whether to return only guide sequences having canonical PAM sequences. If TRUE (default), only PAM sequences with the highest weights stored in the crisprNuclease object will be considered.

Whether to consider both strands in search for protospacer sequences. TRUE by default.

Length of spacers to return, if different from the default length specified by crisprNuclease.

Whether to only include gRNAs that cut in the input range, as given by cut_site (TRUE) or to include all gRNAs that share any overlap with the input range (FALSE). TRUE by default. Ignored when x is not a GRanges object.

Whether to remove spacer sequences that contain ambiguous nucleotides (not explicitly A, C, G, or T). TRUE by default.

This function returns a PairedGuideSet object that stores gRNA pairs targeting the two genomic regions provided as input. The gRNAs in position 1 target the first genomic region, and the gRNAs in position 2 target the second genomic region.

This function can be used for the following scenarios:

1. Designing pairs of gRNAs targeting different genes, for instance for dual-promoter Cas9 systems, or polycystronic Cas12a constructs. This can also be used to target a given gene with multiple gRNAs for improved efficacy (for instance CRISPRa and CRISPRi)
2. Designing pairs of gRNAs for double nicking systems such as Cas9 D10A.

See vignette for more examples.
findSpacers

Value

A PairedGuideSet object.

Author(s)

Jean-Philippe Fortin

See Also

findSpacers to find unpaired spacer sequences, and the PairedGuideSet object documentation to understand the output of findSpacerPairs.

Examples

```r
library(GenomicRanges)
library(ASgenome.Hsapiens.UCSC.hg38)
library(crisprBase)
bsgenome = BSgenome.Hsapiens.UCSC.hg38

# Region 1:
gr1 = GRanges(c("chr12"),
               IRanges(start=22224014, end=22225007))

# Region 2:
gr2 = GRanges(c("chr13"),
               IRanges(start=23224014, end=23225007))

# Pairs targeting the same region:
pairs <- findSpacerPairs(gr1, gr1, bsgenome=bsgenome)

# Pairs targeting two regions:
# The gRNA in position targets gr1
# and the gRNA in position 2 targets gr2
pairs <- findSpacerPairs(gr1, gr2, bsgenome=bsgenome)
```

---

**findSpacers**

Find CRISPR gRNA spacer sequences from a set of DNA sequences.

**Description**

Returns all possible, valid gRNA sequences for a given CRISPR nuclease from either a GRanges object or a set of sequence(s) contained in either a DNAStringSet, DNAString or character vector of genomic sequences.
Usage

findSpacers(
  x,
  crisprNuclease = NULL,
  bsgenome = NULL,
  canonical = TRUE,
  both_strands = TRUE,
  spacer_len = NULL,
  strict_overlap = TRUE,
  remove_ambiguities = TRUE,
  remove_duplicates = TRUE,
)

Arguments

x Either a GRanges, a DNAStringSet, or a DNAString object, or a character vector of genomic sequences. See details.
crisprNuclease A CrisprNuclease object.
bsgenome A BSgenome object from which to extract sequences if x is a GRanges object.
canonical Whether to return only guide sequences having canonical PAM sequences. If TRUE (default), only PAM sequences with the highest weights stored in the crisprNuclease object will be considered.
both_strands Whether to consider both strands in search for protospacer sequences. TRUE by default.
spacer_len Length of spacers to return, if different from the default length specified by crisprNuclease.
strict_overlap Whether to only include gRNAs that cut in the input range, as given by cut_site (TRUE) or to include all gRNAs that share any overlap with the input range (FALSE). TRUE by default. Ignored when x is not a GRanges object.
remove_ambiguities Whether to remove spacer sequences that contain ambiguous nucleotides (not explicitly A, C, G, or T). TRUE by default.
remove_duplicates Whether to remove duplicated protospacer sequences originating from overlapping genomic ranges. TRUE by default.

Details

If x is a GRanges object then a BSgenome must be supplied to bsgenome, from which the genomic sequence is obtained, unless the bsgenome can be inferred from genome(x), for example, "hg38". Otherwise, all supplied sequences are treated as the "+" strands of chromosomes in a "custom" genome.

Ranges or sequences in x may contain names where permitted. These names are stored in region in the mcols of the output, and as seqnames of the output if x is not a GRanges object. If not NULL, names(x) must be unique, otherwise ranges or sequences are enumerated with the "region_" prefix.
When x is a `GRanges`, the *strand is interpreted as both strands. Consequently, the both_strands argument has no effect on such ranges.

**Value**

A `GuideSet` object.

**Author(s)**

Jean-Philippe Fortin, Luke Hoberecht

**Examples**

```r
# Using custom sequence as input:
my_seq <- c(my_seq="CCANAGTGAAACCACGTCTCTCTATAAAGAATACAAAAAATTAGCCGGGTGTTA")
guides <- findSpacers(my_seq)

# Exon-intro region of human KRAS specified
# using a GRanges object:
library(GenomicRanges)
library(BSgenome.Hsapiens.UCSC.hg38)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
gr_input <- GRanges(c("chr12"),
                   IRanges(start=25224014, end=25227007))
guideSet <- findSpacers(gr_input, bsgenome=bsgenome)

data(enAsCas12a, package="crisprBase")
guideSet <- findSpacers(gr_input, canonical=FALSE,
                        bsgenome=bsgenome,
                        crisprNuclease=enAsCas12a)
```

---

**flattenGuideSet**  

Create a list of annotation tables from a `GuideSet` object

**Description**

Create a list of annotation tables from a `GuideSet` object

**Usage**

`
flattenGuideSet(guideSet, useSpacerCoordinates = TRUE, primaryOnly = FALSE)
`
getBarcodeDistanceMatrix

Get distance between query and target sets of barcodes

Description
Get distance between query and target sets of barcodes

Usage
getBarcodeDistanceMatrix(
  queryBarcodes,
  targetBarcodes = NULL,
  binnarize = TRUE,
  min_dist_edit = NULL,
  dist_method = c("hamming", "levenshtein"),
  ignore_diagonal = TRUE,
  splitByChunks = FALSE,
  n_chunks = NULL
)
getConsensusIsoform

Arguments

queryBarcodes  Character vector of DNA sequences or DNAStringSet.
targetBarcodes  Optional character vector of DNA sequences or DNAStringSet. If NULL, distances will be calculated between barcodes provided in queryBarcodes.
binnarize       Should the distance matrix be made binnary? TRUE by default. See details section.
min_dist_edit   Integer specifying the minimum distance edit required for barcodes to be considered dissimilar when binnarize=TRUE, ignored otherwise.
dist_method     String specifying distance method. Must be either "hamming" (default) or "levenshtein".
ignore_diagonal When targetBarcodes=NULL, should the diagonal distances be set to 0 to ignore self distances? TRUE by default.
splitByChunks   Should distances be calculated in a chunk-wise manner? FALSE by default. Highly recommended when the set of query barcodes is large to reduce memory footprint.
n_chunks       Integer specifying the number of chunks to be used when splitByChunks=TRUE. If NULL (default), number of chunks will be chosen automatically.

Value

A sparse matrix of class dgCMatrix or dsCMatrix in which rows correspond to queryBarcodes and columns correspond to targetBarcodes. If binnarize=TRUE, a value of 0 indicates that two barcodes have a distance greater of equal to min_dist_edit, otherwise the value is 1. If binnarize=FALSE, values represent the actual calculated distances between barcodes.

Author(s)

Jean-Philippe Fortin

Examples

data(guideSetExample, package="crisprDesign")
guideSetExample <- addOpsBarcodes(guideSetExample)
barcodes <- as.character(guideSetExample$opsBarcode)
dist <- getBarcodeDistanceMatrix(barcodes, min_dist_edit=2)

getConsensusIsoform  Get the genomic ranges of a consensus isoform

Description

Get the genomic ranges of a consensus isoform. The consensus isoform is taken as the union of exons across all isoforms where overlapping exons are merged to produce a simplified set through the reduce method of the GenomicRanges package.
Usage

getConsensusIsoform(gene_id, txObject)

Arguments

gene_id String specifying Ensembl ID for the gene of interest. E.g. "ENSG00000049618". ID must be present in txObject$exons$gene_id.

txObject A TxDb object or a GRangesList object obtained using TxDb2GRangesList to provide a gene model annotation.

Value

A GRanges object.

Author(s)

Jean-Philippe Fortin

Examples

data(grListExample)
gene_id <- "ENSG00000120645"
gr <- getConsensusIsoform(gene_id, grListExample)

getMrnaSequences Retrieve mRNA sequences

Description

A function for retrieving mRNA sequences of select transcripts.

Usage

getMrnaSequences(txids, txObject, bsgenome)

Arguments

txids A character vector of Ensembl transcript IDs. IDs not present in txObject will be silently ignored.

txObject A TxDb object or a GRangesList object obtained from TxDb2GRangesList. Defines genomic ranges for txids.

bsgenome A BSgenome object from which to extract mRNA sequences.

Value

A DNAStringSet object of mRNA sequences. Note that sequences are returned as DNA rather than RNA.
getPreMrnaSequences

Author(s)
Jean-Philippe Fortin

Examples
library(BSgenome.Hsapiens.UCSC.hg38)
data(grListExample)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
txids <- c("ENST00000538872", "ENST00000382841")
out <- getMrnaSequences(txids, grListExample, bsgenome)

getPreMrnaSequences Retrieve pre-mRNA sequences

Description
A function for retrieving pre-mRNA sequences of select transcripts.

Usage
getPreMrnaSequences(txids, txObject, bsgenome)

Arguments
  txids A character vector of Ensembl transcript IDs. IDs not present in txObject will be silently ignored.
  txObject A TxDb object or a GRangesList object obtained from TxDb2GRangesList. Defines genomic ranges for txids.
  bsgenome A BSgenome object from which to extract pre-mRNA sequences.

Value
A DNAStringSet object of mRNA sequences. Note that sequences are returned as DNA rather than RNA.

Author(s)
Jean-Philippe Fortin

Examples
library(BSgenome.Hsapiens.UCSC.hg38)
data(grListExample)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
txids <- c("ENST00000538872", "ENST00000382841")
out <- getPreMrnaSequences(txids, grListExample, bsgenome)
getTssObjectFromTxObject

Extract TSS coordinates from a gene model object

Description

Extract TSS coordinates from a gene model object.

Usage

getTssObjectFromTxObject(txObject)

Arguments

txObject A TxDb object or a GRangesList object obtained using TxDb2GRangesList for annotating on-target and off-target alignments using gene annotation.

Value

A GRanges object containing TSS coordinates

Author(s)

Jean-Philippe Fortin

Examples

data(grListExample, package="crisprDesign")
tss <- getTssObjectFromTxObject(grListExample)

getTxDb

Description

Convenience function for constructing a TxDb object.

Usage

getTxDb(file = NA, organism, release = NA, tx_attrib = "gencode_basic", ...)

getTxDb
Arguments

- **file**: File argument for `makeTxDbFromGFF` (see help page for `makeTxDbFromGFF`). If NA (default), function will return a TxDb object from Ensembl using `makeTxDbFromEnsembl`.
- **organism**: String specifying genus and species name (e.g. "Homo sapiens" for human). Required if file is not provided. If file is provided, this value can be set to NA to have organism information as unspecified.
- **release**: Ensembl release version; passed to `makeTxDbFromEnsembl` when file is not specified. See help page for `makeTxDbFromEnsembl`.
- **tx_attrib**: Argument passed to `makeTxDbFromEnsembl` when file is not specified. See help page for `makeTxDbFromEnsembl`.
- **...**: Additional arguments passed to either `makeTxDbFromGFF` (if file is specified) or `makeTxDbFromEnsembl` if file is NA.

Value

A TxDb object.

Author(s)

Jean-Philippe Fortin, Luke Hoberecht

Examples

```r
if (interactive()){
  # To obtain a TxDb for Homo sapiens from Ensembl:
  txdb <- getTxDb()

  # To obtain a TxDb from a GFF file:
  file='https://www.mirbase.org/ftp/CURRENT/genomes/hsa.gff3'
  txdb <- getTxDb(file=file)
}
```

---

**getTxInfoDataFrame**

*To obtain a DataFrame of transcript-specific CDS and mRNA coordinates*

Description

To obtain a DataFrame of transcript-specific CDS and mRNA coordinates.

Usage

```r
getTxInfoDataFrame(
  tx_id, 
  txObject, 
  bsgenome, 
  extend = 30, 
  checkCdsLength = TRUE 
)
```
getTxInfoDataFrame

Arguments

- **tx_id**: String specifying ENSEMBL Transcript id.
- **txObject**: A TxDb object or a GRangesList object obtained using TxDb2GRangesList.
- **bsgenome**: BSgenome object from which to extract sequences if a GRanges object is provided as input.
- **extend**: Integer value specifying how many nucleotides in intron regions should be included.
- **checkCdsLength**: Should the CDS nucleotide length be a multiple of 3? TRUE by default.

Value

A DataFrame containing nucleotide and amino acid information. The columns are:

- **chr**: Character specifying chromosome.
- **pos**: Integer value specifying coordinate in reference genome.
- **strand**: Character specifying strand of transcript.
- **nuc**: Character specifying nucleotide on the strand specified by `strand`.
- **aa**: Character specifying amino acid.
- **aa_number**: Integer specifying amino acid number from 5' end.
- **exon**: Integer specifying exon number.
- **pos_plot**: Integer specifying plot coordinate. Useful for plotting.
- **pos_mRNA**: Integer specifying relative mRNA coordinate from the start of the mRNA.
- **pos_CDS**: Integer specifying relative CDS coordinate from the start of the CDS.
- **region**: Character specifying gene region: 3UTR, 5UTR, CDS, Intron, Upstream (promoter) or downstream.

Author(s)

Jean-Philippe Fortin

Examples

```r
library(BSgenome.Hsapiens.UCSC.hg38)
bsgenome <- BSgenome.Hsapiens.UCSC.hg38
data("grListExample")
tx_id <- "ENST00000538872"
df <- getTxInfoDataFrame(tx_id=tx_id,
  txObject=grListExample,
  bsgenome=bsgenome)
```
grListExample

Example of a TxDb object converted to a GRangesList

Description

Example of a TxDb object converted to a GRangesList object for human gene IQSEC3 (ENSG00000120645).

Usage

data(grListExample, package="crisprDesign")

Format

Named GRangesList with 7 elements: transcripts, exons, cds, fiveUTRs, threeUTRs, introns and tss.

Details

The full human transcriptome TxDb object was obtained from the Ensembl 104 release using the getTxDb function and converted to a GRangesList object using the TxDb2GRangesList function and subsequently subsetted to only contain the IQSEC3 gene (ENSG00000120645) located at the start of chr12 in the human genome (hg38 build).

grRepeatsExample

Example of a GRanges object containing repeat elements

Description

Example of a GRanges object containing genomic coordinates of repeat elements found in the neighborhood of human gene IQSEC3 (ENSG00000120645).

Usage

data(grRepeatsExample, package="crisprDesign")

Format

A GRanges object.
GuideSet2DataFrames  
Create a list of annotation tables from a GuideSet object

Description
Create a list of annotation tables from a GuideSet object

Usage
GuideSet2DataFrames(guideSet, useSpacerCoordinates = TRUE, primaryOnly = FALSE)

Arguments
- **guideSet**: A GuideSet object
- **useSpacerCoordinates**: Should the spacer coordinates be used as start and end coordinates? TRUE by default. If FALSE, the PAM site coordinate is used for both start and end.
- **primaryOnly**: Should only the primary table (on-targets) be returned? FALSE by default.

Value
A simple list of tables containing annotations derived from a GuideSet object. The first table ("primary") is always available, while the other tables will be only available when the annotations were added to the GuideSet object.

- **primary**: Primary table containing genomic coordinates and sequence information of the gRNA sequences. Also contains on-target and off-target scores when available.
- **alignments**: Table of on- and off-target alignments.
- **geneAnnotation**: Gene context annotation table.
- **tssAnnotation**: TSS context annotation table.
- **enzymeAnnotation**: Boolean table indicating whether or not recognition motifs of restriction enzymes are found.
- **snps**: SNP annotation table (human only).

Author(s)
Jean-Philippe Fortin, Luke Hoberecht

Examples
```r
data(guideSetExampleFullAnnotation)
tables <- GuideSet2DataFrames(guideSetExampleFullAnnotation)
```
guideSetExample

Example of a GuideSet object storing gRNA sequences targeting the CDS of IQSEC3

Description

Example of a GuideSet object storing gRNA sequences targeting the coding sequence of human gene IQSEC3 (ENSG00000120645) for SpCas9 nuclease.

Usage

data(guideSetExample, package="crisprDesign")

Format

A GuideSet object.

Details

The object was obtained by calling findSpacers on the CDS region of human gene IQSEC3. See code in inst/scripts/generateGuideSet.R.

guideSetExampleFullAnnotation

Example of a fully-annotated GuideSet object storing gRNA sequences targeting the CDS of IQSEC3

Description

Example of a fully-annotated GuideSet object storing gRNA sequences targeting the coding sequence of human gene IQSEC3 (ENSG00000120645) for SpCas9 nuclease.

Usage

data(guideSetExampleFullAnnotation, package="crisprDesign")

Format

A GuideSet object.

Details

The object was obtained by applying all available add* annotation functions in crisprDesign (e.g. addSequenceFeatures) to a randomly selected 20-guide subset of guideSetExample. See code in inst/scripts/generateGuideSetFullAnnotation.R.
guideSetExampleWithAlignments

Example of a GuideSet object storing gRNA sequences targeting the CDS of IQSEC3 with off-target alignments.

Description

Example of a GuideSet object storing gRNA sequences targeting the coding sequence of human gene IQSEC3 (ENSG00000120645) for SpCas9 nuclease with off-target alignments.

Usage

data(guideSetExampleWithAlignments, package="crisprDesign")

Format

A GuideSet object.

Details

The object was obtained by adding off-target alignments to a randomly selected 20-guide subset of guideSetExample. See code in inst/scripts/generateGuideSetFullAnnotation.R.

pamOrientation

An S4 class to store pairs of CRISPR gRNA sequences.

Description

An S4 class to store pairs of CRISPR gRNA sequences.

Usage

pamOrientation(object, ...)

pamDistance(object, ...)

spacerDistance(object, ...)

cutLength(object, ...)

PairedGuideSet(GuideSet1 = NULL, GuideSet2 = NULL)

## S4 method for signature 'PairedGuideSet'
pamOrientation(object)
## S4 method for signature 'PairedGuideSet'
pamDistance(object)

## S4 method for signature 'PairedGuideSet'
spacerDistance(object)

## S4 method for signature 'PairedGuideSet'
cutLength(object)

## S4 method for signature 'PairedGuideSet'
crisprNuclease(object, index = NULL)

## S4 method for signature 'PairedGuideSet'
spacers(object, as.character = FALSE, returnAsRna = FALSE, index = NULL)

## S4 method for signature 'PairedGuideSet'
pams(object, as.character = FALSE, returnAsRna = FALSE, index = NULL)

## S4 method for signature 'PairedGuideSet'
pamSites(object, index = NULL)

## S4 method for signature 'PairedGuideSet'
cutSites(object, index = NULL)

## S4 method for signature 'PairedGuideSet'
protospacers(
    object,
    as.character = FALSE,
    include.pam = FALSE,
    returnAsRna = FALSE,
    index = NULL
)

## S4 method for signature 'PairedGuideSet'
spacerLength(object, index = NULL)

## S4 method for signature 'PairedGuideSet'
pamLength(object, index = NULL)

## S4 method for signature 'PairedGuideSet'
pamSide(object, index = NULL)

### Arguments

- **object**  
  A `PairedGuideSet` object.
- **...**  
  Additional arguments for class-specific methods
- **GuideSet1**  
  A `GuideSet` object containing gRNAs at the first position of the pairs.
- **GuideSet2**  
  A `GuideSet` object containing gRNAs at the second position of the pairs.
preparePfamTable

index
Integer value indicating gRNA position. Must be either 1, 2, or NULL (default). If NULL, both positions are returned.

as.character
Should sequences be returned as a character vector? FALSE by default, in which case sequences are returned as a DNAStringSet.

returnAsRna
Should the sequences be returned as RNA instead of DNA? FALSE by default.

include.pam
Should PAM sequences be included? FALSE by default.

Value
A PairedGuideSet object.

Functions
- PairedGuideSet(): Create a PairedGuideSet object

Constructors
Use the constructor link{PairedGuideSet} to create a PairedGuideSet object.

Examples
library(crisprDesign)
data(guideSetExample, package="crisprDesign")
gs <- guideSetExample
gs <- gs[order(BiocGenerics::start(gs))]
gs1 <- gs[1:10]
gs2 <- gs[1:10+10]
pgs <- PairedGuideSet(gs1, gs2)

preparePfamTable

Description
Obtain Pfam domains from biomaRt for all transcripts found in a gene model object.

Usage
preparePfamTable(txObject, mart_dataset)

Arguments
tXObject
A TxDb object or a GRangesList object obtained using TxDb2GRangesList to provide a gene model annotation.
mart_dataset
String specifying dataset to be used by biomaRt for Pfam domains annotation. E.g. "hsapiens_gene_ensembl".
Value

A DataFrame object with the following columns:

- `ensembl_transcript_id` Ensembl transcript ID.
- `pfam` Pfam domain name.
- `pfam_start` Start amino acid coordinate of the Pfam domain.
- `pfam_end` End amino acid coordinate of the Pfam domain.

Author(s)

Jean-Philippe Fortin, Luke Hoberecht

Examples

data(grListExample, package="crisprDesign")

if (interactive()){
  pfamTable <- preparePfamTable(grListExample,
                              mart_dataset="hsapiens_gene_ensembl")
}

queryTss                  Convenience function to search for TSS coordinates.

Description

Convenience function to search for TSS coordinates.

Usage

queryTss(tssObject, queryColumn, queryValue, tss_window = NULL)

Arguments

tssObject          A GRanges containing genomic positions of transcription starting sites (TSSs).
queryColumn        String specifying which column of mcols(tssObject) should be searched for.
queryValue         Character vector specifying the values to search for in tssObject[[queryColumn]].
tss_window         Numeric vector of length 2 establishing the genomic region to return. The value pair sets the 5 prime and 3 prime limits, respectively, of the genomic region with respect to the TSS. Use negative value(s) to set limit(s) upstream of the TSS. Default is `c(-500, 500)`, which includes 500bp upstream and downstream of the TSS.

Value

A GRanges object. Searches yielding no results will return an empty GRanges object.
queryTxObject

Convenience function to search for gene coordinates.

Description
Convenience function to search for gene coordinates.

Usage
queryTxObject(
  txObject,
  featureType = c("transcripts", "exons", "cds", "fiveUTRs", "threeUTRs", "introns"),
  queryColumn, 
  queryValue
)

Arguments

  txObject          A TxDb object or a GRangesList object obtained using TxDb2GRangesList.
  featureType      The genomic feature in txObject to base your query on. Must be one of the following: "transcripts", "exons", "cds", "fiveUTRs", "threeUTRs" or "introns".
  queryColumn      Character string specifying the column in txObject[[featureType]] to search for queryValue(s).
  queryValue       Vector specifying the value(s) to search for in txObject[[featureType]][[queryColumn]].

Value

  A GRanges object. Searches yielding no results will return an empty GRanges object.

Author(s)
Luke Hoberecht, Jean-Philippe Fortin
See Also

`queryTss` for querying TSS annotations.

Examples

data(grListExample, package="crisprDesign")
queryTxObject(grListExample,
  featureType="cds",
  queryColumn="gene_symbol",
  queryValue="IQSEC3")

---

**rankSpacers**

**Recommended gRNA ranking**

**Description**

Function for ranking spacers using recommended crisprDesign criteria. CRISPRko, CRISPRa and CRISPRi modalities are supported.

**Usage**

```r
rankSpacers(
  guideSet,
  tx_id = NULL,
  commonExon = FALSE,
  modality = c("CRISPRko", "CRISPRa", "CRISPRi"),
  useDistanceToTss = TRUE
)
```

**Arguments**

- **guideSet** A `GuideSet` object.
- **tx_id** Optional string specifying transcript ID to use isoform-specific information for gRNA ranking.
- **commonExon** Should gRNAs targeting common exons be prioritized? FALSE by default. If TRUE, tx_id must be provided.
- **modality** String specifying the CRISPR modality. Should be one of the following: "CRISPRko", "CRISPRa", or "CRISPRi".
- **useDistanceToTss** Should distance to TSS be used to rank gRNAs for CRISPRa and CRISPRi applications? TRUE by default. For SpCas9 and human targets, this should be set to FALSE if `addCrispraiScores` was used.
Details

For each nuclease, we rank gRNAs based on several rounds of priority. For SpCas9, gRNAs with unique target sequences and without 1-or 2-mismatch off-targets located in coding regions are placed into the first round. Then, gRNAs with a small number of one- or two-mismatch off-targets (less than 5) are placed into the second round. Remaining gRNAs are placed into the third round. Finally, any gRNAs overlapping a common SNP (human only), containing a polyT stretch, or with extreme GC content (below 20 are placed into the fourth round.

If tx_id is specified, within each round of selection, gRNAs targeting the first 85 percent of the specific transcript are prioritized first. If tx_id is specified, and commonExon is set to TRUE, gRNAs targeting common exons across isoforms are also prioritized. If a conservation score is available, gRNAs targeting conserved regions (phyloP conservation score greater than 0), are also prioritized.

Within each bin, gRNAs are ranked by a composite on-target activity rank to prioritize active gRNAs. The composite on-target activity rank is calculated by taking the average rank across the DeepHF and DeepSpCas9 scores for CRISPRko. For CRISPRa or CRISPRi, the CRISPRai scores are used if available.

The process is identical for enAsCas12a, with the exception that the enPAMGb method is used as the composite score.

For CasRx, gRNAs targeting all isoforms of a given gene, with no 1- or 2-mismatch off-targets, are placed into the first round. gRNAs targeting at least 50 percent of the isoforms of a given gene, with no 1- or 2-mismatch off-targets, are placed into the second round. Remaining gRNAs are placed into the third round. Within each round of selection, gRNAs are further ranked by the CasRxRF on-target score.

Value

A GuideSet object ranked from best to worst gRNAs, with a column rank stored in mcols(guideSet) indicating gRNA rank.

Author(s)

Luke Hoberecht, Jean-Philippe Fortin

Examples

data(guideSetExampleFullAnnotation, package="crisprDesign")
gs <- rankSpacers(guideSetExampleFullAnnotation,
                  tx_id = "ENST00000538872")
gs
Description
These objects are imported from other packages. Follow the links below to see their documentation.

Description
Remove GuideSet gRNAs that overlap repeat elements.

Usage
removeRepeats(object, ...)

## S4 method for signature 'GuideSet'
removeRepeats(object, gr.repeats = NULL, ignore.strand = TRUE)

## S4 method for signature 'PairedGuideSet'
removeRepeats(object, gr.repeats = NULL, ignore.strand = TRUE)

## S4 method for signature 'NULL'
removeRepeats(object)

Arguments

object  A GuideSet object or a PairedGuideSet object.
...  Additional arguments, currently ignored.
gr.repeats  A GRanges object containing repeat elements regions.
ignore.strand  Should gene strand be ignored when annotating? TRUE by default.

Value
object filtered for spacer sequences not overlapping any repeat elements. An inRepeats column is also appended in mcols(object).

Author(s)
Jean-Philippe Fortin, Luke Hoberecht

See Also
link(addRepeats).
Examples

```r
data(guideSetExample, package="crisprDesign")
data(grRepeatsExample, package="crisprDesign")
guideSet <- removeSpacers(guideSetExample,
  gr.repeats=grRepeatsExample)
```

removeSpacersWithSecondaryTargets

*Remove gRNAs targeting secondary targets*

Description

Remove gRNAs targeting secondary targets

Usage

```r
removeSpacersWithSecondaryTargets(
  guideSet,
  geneID,
  geneColumn = "gene_id",
  ignoreGenesWithoutSymbols = TRUE,
  ignoreReadthroughs = TRUE
)
```

Arguments

- **guideSet**: A `GuideSet` object.
- **geneID**: String specifying gene ID of the main gene target.
- **geneColumn**: Column in `geneAnnotation(guideSet)` specifying gene IDs.
- **ignoreGenesWithoutSymbols**: Should gene without gene symbols be ignored when removing co-targeting gRNAs?
- **ignoreReadthroughs**: Should readthrough genes be ignored when removing co-targeting gRNAs?

Details

The protospacer target sequence of gRNAs can be located in overlapping genes, and this function allows users to filter out such gRNAs. This ensures remaining gRNAs are targeting only one gene.

Value

A `GuideSet` object with gRNAs targeting multiple targets removed.

Author(s)

Jean-Philippe Fortin
tssObjectExample

Example of a GRanges object containing TSS coordinates

Description
Example of a GRanges containing transcription starting site (TSS) coordinates for human gene IQSEC3 (ENSG00000120645).

Usage
data(tssObjectExample, package="crisprDesign")

Format
GRanges object of length 2 corresponding to the 2 TSSs of gene IQSEC3.

Details
The TSS coordinates were obtained from the two transcript stored in the grListExample object for gene IQSEC3.

TxDb2GRangesList Convert a TxDb object into a GRangesList

Description
Convenience function to reformat a TxDb object into a GRangesList.

Usage

TxDb2GRangesList(
  txdb,
  standardChromOnly = TRUE,
  genome = NULL,
  seqlevelsStyle = c("UCSC", "NCBI")
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txdb</td>
<td>A TxDb object.</td>
</tr>
<tr>
<td>standardChromOnly</td>
<td>Should only standard chromosomes be kept? TRUE by default.</td>
</tr>
<tr>
<td>genome</td>
<td>Optional string specifying genome. e.g. &quot;hg38&quot;, to be added to genome(txdb).</td>
</tr>
<tr>
<td>seqlevelsStyle</td>
<td>String specifying which style should be used for sequence names. &quot;UCSC&quot; by default (including &quot;chr&quot;). &quot;NCBI&quot; will omit &quot;chr&quot; in the sequence names.</td>
</tr>
</tbody>
</table>
Value

A named RangesList of length 7 with the following elements: transcripts, exons, introns, cds, fiveUTRs, threeUTRs and tss.

Author(s)

Jean-Philippe Fortin, Luke Hoberecht

See Also

getTxDb to obtain a TxDb object.

Examples

```r
if (interactive()){
  # To obtain a TxDb for Homo sapiens from Ensembl:
  txdb <- getTxDb()

  # To convert to a GRanges list:
  txdb <- TxDb2GRangesList(txdb)
}
```

---

**updateOpsLibrary**  
*Update OPS library with additional gRNAs*

**Description**

Update OPS library with additional gRNAs

**Usage**

```r
updateOpsLibrary(
  opsLibrary,  
  df,  
  n_guides = 4,  
  gene_field = "gene",  
  min_dist_edit = 2,  
  dist_method = c("hamming", "levenshtein"),  
  splitByChunks = FALSE  
)
```
Arguments

- **opsLibrary** data.frame obtained from `designOpsLibrary`.
- **df** data.frame containing information about additional candidate gRNAs to add to the OPS library.
- **n_guides** Integer specifying how many gRNAs per gene should be selected. 4 by default.
- **gene_field** String specifying the column in df specifying gene names.
- **min_dist_edit** Integer specifying the minimum distance edit required for barcodes to be considered dissimilar. Barcodes that have edit distances less than the min_dist_edit will not be included in the library. 2 by default.
- **dist_method** String specifying distance method. Must be either "hamming" (default) or "levenshtein".
- **splitByChunks** Should distances be calculated in a chunk-wise manner? FALSE by default. Highly recommended when the set of query barcodes is large to reduce memory footprint.

Value

A data.frame containing the original gRNAs from the input `opsLibrary` data.frame as well as additional gRNAs selected from the input data.frame `df`.

Author(s)

Jean-Philippe Fortin

Examples

data(guideSetExample, package="crisprDesign")
guideSet <- unique(guideSetExample)
guideSet <- addOpsBarcodes(guideSet)
guideSet1 <- guideSet[1:200]
guideSet2 <- guideSet[201:400]

```
df1 <- data.frame(ID=names(guideSet1),
  spacer=spacers(guideSet1, as.character=TRUE),
  opsBarcode=as.character(guideSet1$opsBarcode))
df2 <- data.frame(ID=names(guideSet2),
  spacer=spacers(guideSet2, as.character=TRUE),
  opsBarcode=as.character(guideSet2$opsBarcode))
```

# Creating mock gene:
df1$gene <- rep(paste0("gene",1:10),each=20)
df2$gene <- rep(paste0("gene",1:10+10),each=20)
df1$rank <- rep(1:20,10)
df2$rank <- rep(1:20,10)
opsLib <- designOpsLibrary(df1)
opsLib <- updateOpsLibrary(opsLib, df2)
validateOpsLibrary Validate gRNA library for optical pooled screening

Description

Validate gRNA library for optical pooled screening

Usage

validateOpsLibrary(
  df,
  min_dist_edit = 2,
  dist_method = c("hamming", "levenshtein")
)

Arguments

df data.frame containing information about candidate gRNAs from which to build the OPS library. See details.

min_dist_edit Integer specifying the minimum distance edit required for barcodes to be considered dissimilar.

dist_method String specifying distance method. Must be either "hamming" (default) or "levenshtein".

Value

The original df is all checks pass. Otherwise, a stop error.

Author(s)

Jean-Philippe Fortin

Examples

data(guideSetExample, package="crisprDesign")
guideSet <- unique(guideSetExample)
guideSet <- addOpsBarcodes(guideSet)
df <- data.frame(ID=names(guideSet),
  spacer=spacers(guideSet, as.character=TRUE),
  opsBarcode=as.character(guideSet$opsBarcode))
df$gene <- rep(paste0("gene",1:40),each=20)
df$rank <- rep(1:20,40)
opsLib <- designOpsLibrary(df)
opsLib <- validateOpsLibrary(opsLib)
Index

* datasets
  grListExample, 63
  grRepeatsExample, 63
  guideSetExample, 65
  guideSetExampleFullAnnotation, 65
  guideSetExampleWithAlignments, 66
  tssObjectExample, 75

* internal
  reexports, 72

addCompositeScores, 3
addCompositeScores, GuideSet-method (addCompositeScores), 3
addCompositeScores, NULL-method (addCompositeScores), 3
addCompositeScores, PairedGuideSet-method (addCompositeScores), 3
addConservationScores, 5
addConservationScores, GuideSet-method (addConservationScores), 5
addConservationScores, NULL-method (addConservationScores), 5
addConservationScores, PairedGuideSet-method (addConservationScores), 5
addCrispraiScores, 6
addCrispraiScores, GuideSet-method (addCrispraiScores), 6
addCrispraiScores, NULL-method (addCrispraiScores), 6
addCrispraiScores, PairedGuideSet-method (addCrispraiScores), 6
addCutSites (crisprNuclease), 41
addCutSites, GuideSet-method (crisprNuclease), 41
addDistanceToTss, 7
addDistanceToTss, GuideSet-method (addDistanceToTss), 7
addDistanceToTss, NULL-method (addDistanceToTss), 7
addEditedAlleles, 8
addEditingSites, 10
addEditingSites, GuideSet-method (addEditingSites), 10
addEditingSites, NULL-method (addEditingSites), 10
addEditingSites, PairedGuideSet-method (addEditingSites), 10
addExonTable, 11
addGeneAnnotation, 11, 12, 22, 37, 38
addGeneAnnotation, GuideSet-method (addGeneAnnotation), 12
addGeneAnnotation, NULL-method (addSNPAnnotation), 28
addGeneAnnotation, PairedGuideSet-method (addGeneAnnotation), 12
addIsoformAnnotation, 15
addIsoformAnnotation, GuideSet-method (addIsoformAnnotation), 15
addIsoformAnnotation, NULL-method (addIsoformAnnotation), 15
addIsoformAnnotation, PairedGuideSet-method (addIsoformAnnotation), 15
addNtcs, 16
addNtcs, GuideSet-method (addNtcs), 16
addNtcs, NULL-method (addNtcs), 16
addNtcs, PairedGuideSet-method (addNtcs), 16
addOffTargetScores, 17, 20
addOffTargetScores, GuideSet-method (addOffTargetScores), 17
addOffTargetScores, NULL-method (addOffTargetScores), 17
addOffTargetScores, PairedGuideSet-method (addOffTargetScores), 17
addOnTargetScores, 4, 7, 18
addOnTargetScores, GuideSet-method
addOnTargetScores, 18
addOnTargetScores, NULL-method
addOnTargetScores, 18
addOnTargetScores, PairedGuideSet-method
addOnTargetScores, 18
addOpsBarcodes, 20
addPamScores, 21
addPamScores, GuideSet-method
addPamScores, 21
addPamScores, NULL-method
addPamScores, 21
addPamScores, PairedGuideSet-method
addPamScores, 21
addPfamDomains, 22
addPfamDomains, GuideSet-method
addPfamDomains, 22
addPfamDomains, NULL-method
addPfamDomains, 22
addPfamDomains, PairedGuideSet-method
addPfamDomains, 22
addRedeinitionFlag, 23
addRepeats, 24
addRepeats, GuideSet-method
addRepeats, 24
addRepeats, NULL-method
addRepeats, 24
addRepeats, PairedGuideSet-method
addRepeats, 24
addRestrictionEnzymes, 25
addRestrictionEnzymes, GuideSet-method
addRestrictionEnzymes, 25
addRestrictionEnzymes, NULL-method
addRestrictionEnzymes, 25
addRestrictionEnzymes, PairedGuideSet-method
addRestrictionEnzymes, 25
addSequenceFeatures, 27
addSequenceFeatures, GuideSet-method
addSequenceFeatures, 27
addSequenceFeatures, NULL-method
addSequenceFeatures, 27
addSequenceFeatures, PairedGuideSet-method
addSequenceFeatures, 27
addSNPAnnotation, 28
addSNPAnnotation, GuideSet-method
addSNPAnnotation, 28
addSNPAnnotation, NULL-method
addSNPAnnotation, 28
addSNPAnnotation, PairedGuideSet-method
addSNPAnnotation, 28
addSpacerAlignments, 30, 34
addSpacerAlignments, GuideSet-method
addSpacerAlignments, 30
addSpacerAlignments, NULL-method
addSpacerAlignments, 30
addSpacerAlignments, PairedGuideSet-method
addSpacerAlignments, 30
addSpacerAlignmentsIterative, 33, 34
addSpacerAlignmentsIterative
addSpacerAlignmentsIterative, 30
addSpacerAlignmentsIterative, GuideSet-method
addSpacerAlignmentsIterative, 30
addSpacerAlignmentsIterative, NULL-method
addSpacerAlignmentsIterative, 30
addSpacerAlignmentsIterative, PairedGuideSet-method
addSpacerAlignmentsIterative, 30
addTssAnnotation, 8, 14, 35
addTssAnnotation, GuideSet-method
addTssAnnotation, 35
addTssAnnotation, NULL-method
addTssAnnotation, 35
addTssAnnotation, PairedGuideSet-method
addTssAnnotation, 35
addTssAnnotation
addTssAnnotation
addTssAnnotation
addTxTable, 11, 37
alignments (crisprNuclease), 41
alignments, GuideSet-method
alignments, 41
alignments<-, GuideSet-method
alignments<-, 41
BaseEditor, 9
BSgenome, 33, 39, 45, 48, 52, 54, 58, 59, 62
bsgenome (crisprNuclease), 41
bsgenome, GuideSet-method
bsgenome, 41
customSequences, 38
customSequences (crisprNuclease), 41
customSequences, GuideSet-method
customSequences, 41
customSequences, PairedGuideSet-method
customSequences, 41
crisprNuclease, 21, 33, 39, 45, 46, 48, 52, 54
crisprNuclease, 41
crisprNuclease, GuideSet-method
crisprNuclease, 41
crisprNuclease, PairedGuideSet-method
crisprNuclease, 41
crisprNuclease
convertToMinMaxGRanges, 40
customSequences (crisprNuclease), 41
customSequences, GuideSet-method
customSequences, 41
customSequences, PairedGuideSet-method
customSequences, 41
crisprNuclease, 41
INDEX

cutLength (pamOrientation), 66
cutLength, PairedGuideSet-method (pamOrientation), 66
cutSites, GuideSet-method (crisprNuclease), 41
cutSites, PairedGuideSet-method (pamOrientation), 66

DataFrame, 22, 49, 69
designCompleteAnnotation, 47
designOpsLibrary, 50
DNAString, 51–54
DNAStringSet, 45, 51–54, 58, 59, 68

editedAlleles (crisprNuclease), 41
editedAlleles, GuideSet-method (crisprNuclease), 41
enzymeAnnotation, 26
enzymeAnnotation (crisprNuclease), 41
enzymeAnnotation, GuideSet-method (crisprNuclease), 41
enzymeAnnotation<- (crisprNuclease), 41
enzymeAnnotation<-, GuideSet-method (crisprNuclease), 41
exonTable (crisprNuclease), 41
exonTable, GuideSet-method (crisprNuclease), 41

findSpacerPairs, 51
findSpacers, 53, 53, 65
flattenGuideSet, 55

geneAnnotation, 14
geneAnnotation (crisprNuclease), 41
geneAnnotation, GuideSet-method (crisprNuclease), 41
geneAnnotation<- (crisprNuclease), 41
geneAnnotation<-, GuideSet-method (crisprNuclease), 41
getBarcodeDistanceMatrix, 56
getConsensusIsoform, 57
getMrnaSequences, 58
getPAMSequence (completeSpacers), 38
getPAMSiteFromStartAndEnd (completeSpacers), 38
getPreMrnaSequences, 59
getSpacerAlignments, 34
getSpacerAlignments (addSpacerAlignments), 30

getSpacerSequence (completeSpacers), 38
getTssObjectFromTxObject, 60
getTxDb, 60, 63, 76
getTxDbFromGFF, 61
GRanges, 7, 24, 33, 34, 36, 48, 51–55, 62, 63, 69, 70, 73, 75
GRangesList, 11, 13, 32, 37, 48, 58–60, 62, 63, 68, 70, 75, 76
grListExample, 63
grRepeatsExample, 63
GuideSet (crisprNuclease), 41
GuideSet-class (crisprNuclease), 41
GuideSet2DataFrames, 64
guideSetExample, 65
guideSetExampleFullAnnotation, 65
guideSetExampleWithAlignments, 66

makeTxDbFromEnsembl, 61
makeTxDbFromGFF, 61
mcols, 73
mcols (reexports), 72

offTargets (crisprNuclease), 41
offTargets, GuideSet-method (crisprNuclease), 41
onTargets (crisprNuclease), 41
onTargets, GuideSet-method (crisprNuclease), 41

PairedGuideSet, 4, 5, 7, 8, 10, 11, 13, 15–17, 19, 21, 22, 24, 25, 27, 29, 32, 36, 37, 52, 53, 67, 68, 73
PairedGuideSet (pamOrientation), 66
PairedGuideSet-class (pamOrientation), 66

pamDistance (pamOrientation), 66
pamDistance, PairedGuideSet-method (pamOrientation), 66
pamLength, GuideSet-method (crisprNuclease), 41
pamLength, PairedGuideSet-method (pamOrientation), 66

pamOrientation, 66

pamOrientation, PairedGuideSet-method (pamOrientation), 66
pams, GuideSet-method (crisprNuclease), 41
pams, PairedGuideSet-method
  (pamOrientation), 66
pamSide, GuideSet-method
  (crisprNuclease), 41
pamSide, PairedGuideSet-method
  (pamOrientation), 66
pamSites (crisprNuclease), 41
pamSites, GuideSet-method
  (crisprNuclease), 41
pamSites, PairedGuideSet-method
  (pamOrientation), 66
pamSites
  (crisprNuclease), 41
pamSites, GuideSet-method
  (crisprNuclease), 41
pamSites, PairedGuideSet-method
  (pamOrientation), 66
preparePfamTable, 22, 49, 68
protospacers (crisprNuclease), 41
protospacers, GuideSet-method
  (crisprNuclease), 41
protospacers, PairedGuideSet-method
  (pamOrientation), 66
prototypeSequence, GuideSet-method
  (crisprNuclease), 41
queryTss, 69, 71
queryTxObject, 70, 70
rankSpacers, 71
reexports, 72
removeRepeats, 73
removeRepeats, GuideSet-method
  (removeRepeats), 73
removeRepeats, NULL-method
  (removeRepeats), 73
removeRepeats, PairedGuideSet-method
  (removeRepeats), 73
removeSpacersWithSecondaryTargets, 74
Seqinfo, 16, 45
seqnames, 73
seqnames (reexports), 72
snps (crisprNuclease), 41
snps, GuideSet-method
  (crisprNuclease), 41
snps<-(crisprNuclease), 41
snps<-, GuideSet-method
  (crisprNuclease), 41
spacerDistance (pamOrientation), 66
spacerDistance, PairedGuideSet-method
  (pamOrientation), 66
spacerLength, GuideSet-method
  (crisprNuclease), 41
spacerLength, PairedGuideSet-method
  (pamOrientation), 66
spacers (crisprNuclease), 41
spacers, GuideSet-method
  (crisprNuclease), 41
spacers, PairedGuideSet-method
  (pamOrientation), 66
targetOrigin (crisprNuclease), 41
targetOrigin, GuideSet-method
  (crisprNuclease), 41
tssAnnotation, 37
tssAnnotation (crisprNuclease), 41
tssAnnotation, GuideSet-method
  (crisprNuclease), 41
tssAnnotation<-(crisprNuclease), 41
tssAnnotation<-, GuideSet-method
  (crisprNuclease), 41
tssObjectExample, 75
TxDb, 11, 13, 32, 37, 48, 58–63, 68, 70, 75, 76
TxDb2GRangesList, 11, 13, 32, 37, 48, 58–60, 62, 63, 68, 70, 75
txTable (crisprNuclease), 41
txTable, GuideSet-method
  (crisprNuclease), 41
updateOpsLibrary, 76
validateOpsLibrary, 78
VCF, 29, 48