

# Package ‘crisprScore’

April 7, 2026

**Version** 1.14.0

**Date** 2025-06-20

**Title** On-Target and Off-Target Scoring Algorithms for CRISPR gRNAs

**Depends** R (>= 4.1), crisprScoreData (>= 1.1.3)

**Imports** basilisk (>= 1.9.2), BiocGenerics, Biostrings, IRanges, methods, randomForest, reticulate, stringr, utils, XVector

**Suggests** BiocStyle, knitr, rmarkdown, testthat

**biocViews** CRISPR, FunctionalGenomics, FunctionalPrediction

**Description** Provides R wrappers of several on-target and off-target scoring methods for CRISPR guide RNAs (gRNAs).

The following nucleases are supported: SpCas9, AsCas12a, enAsCas12a, and Rfx-Cas13d (CasRx).

The available on-

target cutting efficiency scoring methods are RuleSet1, Azimuth, DeepHF, DeepCpf1, enPAM+GB, and CRISPRscan. Both the CFD and MIT scoring methods are available for off-target

specificity prediction. The package also provides a Lindel-derived score to predict the probability of a gRNA to produce indels inducing a frameshift for the Cas9 nuclease.

Note that DeepHF, DeepCpf1 and enPAM+GB are not available on Windows machines.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**StagedInstall** no

**BugReports** <https://github.com/crisprVerse/crisprScore>

**URL** <https://github.com/crisprVerse/crisprScore/issues>

**LazyData** true

**git\_url** <https://git.bioconductor.org/packages/crisprScore>

**git\_branch** RELEASE\_3\_22

**git\_last\_commit** 86c5f68

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2026-04-07

**Author** Jean-Philippe Fortin [aut, cre, cph],  
 Aaron Lun [aut],  
 Luke Hoberecht [ctb],  
 Pirunthan Perampalam [ctb]

**Maintainer** Jean-Philippe Fortin <fortin946@gmail.com>

## Contents

getAzimuthScores . . . . .	2
getCasRxRFScores . . . . .	3
getCFDScores . . . . .	5
getCrispraiScores . . . . .	6
getCRISPRaterScores . . . . .	7
getCRISPRscanScores . . . . .	8
getDeepCpf1Scores . . . . .	9
getDeepHFScores . . . . .	10
getDeepSpCas9Scores . . . . .	11
getEnPAMGBScores . . . . .	12
getLindelScores . . . . .	13
getMITScores . . . . .	14
getRuleSet1Scores . . . . .	15
getRuleSet3Scores . . . . .	16
scoringMethodsInfo . . . . .	17
sgnaExampleCrispra . . . . .	18
sgnaExampleCrispri . . . . .	18
tssExampleCrispra . . . . .	19
tssExampleCrispri . . . . .	19
<b>Index</b>	<b>20</b>

---

getAzimuthScores	<i>Calculate on-target sgRNA activity scores for Cas9 using Azimuth</i>
------------------	---

---

## Description

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the Azimuth scoring method. The Azimuth algorithm is an improvement upon the commonly-used 'Rule Set 2', also developed by the Doench lab.

## Usage

```
getAzimuthScores(sequences, fork = FALSE)
```

## Arguments

sequences	Character vector of 30bp sequences needed for Azimuth scoring, see details below.
fork	Set to TRUE to preserve changes to the R configuration within the session.

## Details

The input sequences for Azimuth scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (23 nucleotides) and 3 nucleotides downstream of the protospacer sequence, for a total of 30 nucleotides: [4nt][20nt-spacer][NGG][3nt]. Note that a canonical PAM sequence (NGG) is required for Azimuth.

## Value

**getAzimuthScores** returns a data.frame with sequence and score columns. The Azimuth score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

## Author(s)

Jean-Philippe Fortin

## References

Doench, J., Fusi, N., Sullender, M. et al. Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. Nat Biotechnol 34, 184–191 (2016). <https://doi.org/10.1038/nbt.3437>.

## Examples

```
if (interactive()){
  flank5 <- "ACCT" #4bp
  spacer <- "ATCGATGCTGATGCTAGATA" #20bp
  pam <- "AGG" #3bp
  flank3 <- "TTG" #3bp
  input <- paste0(flank5, spacer, pam, flank3)
  results <- getAzimuthScores(input)
}
```

---

getCasRxRFScores

*Calculate on-target sgRNA activity scores for CasRx using CasRx-RF*

---

## Description

Calculate on-target sgRNA activity scores for CasRx (RfxCas13d) using the CasRx-RF algorithm.

## Usage

```
getCasRxRFScores(
  mrnaSequence,
  directRepeat = "aaccctaccaactggctcggggtttgaac",
  binaries = NULL,
  sort = FALSE,
  verbose = TRUE
)
```

**Arguments**

<code>mrnaSequence</code>	A DNASTringSet representing the mRNA sequence for which to extract spacer sequences and calculate scores.
<code>directRepeat</code>	String specifying the direct repeat used in the CasRx construct.
<code>binaries</code>	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.
<code>sort</code>	Should spacers be sorted by score? FALSE by default.
<code>verbose</code>	Should messages be printed to console? TRUE by default.

**Details**

The function first extracts all 23mer spacer sequences targeting the mRNA sequence, and scores them for on-target activity.

**Value**

A data.frame with the following columns:

- `ID` Character vector specifying spacer ID.
- `spacer` 23-mer spacer sequence.
- `pfs_site` coordinate of the protospacer flanking sequence (PFS).
- `protospacer` 23-mer protospacer sequence (reverse complement of the spacer sequence).
- `PFS` PFS nucleotide.
- `score` Raw score (not standardized).
- `standardizedScore` Score standardized between 0 and 1.
- `quartile` Quartile score (1 to 4, with 4 being the best quartile).

A scores closer to 1 indicates higher predicted on-target activity.

**Author(s)**

Jean-Philippe Fortin

**References**

Wessels HH, Méndez-Mancilla A, Guo X, et al. Massively parallel Cas13 screens reveal principles for guide RNA design. *Nat biotechnol.* 2020 Jun;38(6):722-7.

**Examples**

```
if (interactive()){
  fasta <- file.path(system.file(package="crisprScore"),
    "casrxrf/test.fa")
  mrnaSequence <- Biostrings::readDNASTringSet(filepath=fasta,
    format="fasta",
    use.names=TRUE)
  results <- getCasRxRFScores(mrnaSequence)
}
```

---

getCFDScores	<i>Calculate CFD off-target specificity scores</i>
--------------	--

---

### Description

Calculate cutting frequency determination (CFD) off-target specificity scores for CRISPR/Cas9 or CRISPR/CasRX.

### Usage

```
getCFDScores(spacers, protospacers, pams, nuclease = c("SpCas9", "CasRx"))
```

### Arguments

spacers	Character vector of 20bp spacer sequences. Must be in 5' to 3' direction. For SpCas9, must be of length 20bp. For CasRx, must be at most of length 27bp.
protospacers	Character vector of 20bp protospacer sequences (target sequences). Must be in 5' to 3' direction.
pams	Character vector of PAM sequences.
nuclease	String specifying the nuclease. Either "SpCas9" (default) or "CasRx".

### Value

**getCFDScores** returns a data.frame with spacer, protospacer, and score columns. The CFD score takes on a value between 0 and 1. For a given pair (on-target, off-target), a higher CFD score indicates a higher likelihood for the nuclease to cut at the off-target. Non-canonical PAM sequences are taken into account by the CFD algorithm.

### Author(s)

Jean-Philippe Fortin

### References

Doench, J., Fusi, N., Sullender, M. et al. Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. Nat Biotechnol 34, 184–191 (2016). <https://doi.org/10.1038/nbt.3437>.

### Examples

```
# Calculating MIT scores for two off-targets with respect to
# one spacer sequence:
spacer <- "AGGTGTAGTGTGTGATAA"
protospacer1 <- "CGGTGTAGTGTGTGATAA"
protospacer2 <- "CGGTGTCGTGTGTGATAA"
results <- getCFDScores(spacers=spacer,
  protospacers=c(protospacer1, protospacer2),
  pams=c("AGG", "CGG")
)
```

---

getCrispraiScores      *Calculate on-target sgRNA activity scores for CRISPRa and CRISPRi*

---

### Description

Use the Weissman lab scoring method (library design v2) to calculate on-target sgRNA activity scores for Cas9-based CRISPR activation (CRISPRa) and CRISPR inactivation (CRISPRi) gene perturbation studies. The algorithm incorporates chromatin features, transcription start site, and sequence to predict gRNA activity scores. Only sgRNAs designed for the human genome (hg38 build) using Cas9 are supported at the moment, and only spacers of length 19 are supported at the moment.

### Usage

```
getCrispraiScores(
  tss_df,
  sgrna_df,
  verbose = FALSE,
  modality = c("CRISPRa", "CRISPRi"),
  fastaFile = NULL,
  chromatinFiles = NULL
)
```

### Arguments

tss_df	A data.frame specifying coordinates of transcription start site (TSS) of the targeted promoter regions. Must have the following columns: gene_symbol, promoter, transcripts, position, strand, and chr. See details section below for more information.
sgrna_df	A A data.frame specifying coordinates and spacer sequences of the sgRNAs to score. Must have the following columns: grna_id, tss_id, pam_site, strand, and spacer_19mer. See details section below for more information.
verbose	Should messages be printed to the console? TRUE by default.
modality	Which mode of perturbation is being used? Must be a string specifying either CRISPRa or CRISPRi.
fastaFile	String specifying fasta file of the hg38 genome.
chromatinFiles	Named character vector of length 3 specifying BigWig files containing chromatin accessibility data.

### Details

tss\_df details: This must be a data.frame that contains the following columns: \* tss\_id: string specifying name of the TSS. \* gene\_symbol: string specifying sHGNC/HUGO gene identifier. \* promoter: string specifying promoter ID (e.g. "P1" or "P2"). \* transcripts: Ensembl transcript identifier. \* position: start position of TSS in hg38 coordinates. \* strand: strand of the gene/TSS. Must be either + or -. \* chr: string specifying chromosome (e.g. "chr1").

sgrna\_df details: This must be a data.frame that contains the following columns: \* grna\_id: string specifying a unique sgRNA identifier. \* tss\_id: string specifying name of the TSS. \* pam\_site: genomic ccoordinate of the N in the NGG PAM sequence. \* strand: strand fo the sgRNA. Must be either + or -. \* spacer\_19mer: string specifying sgRNA 19mer spacer sequence.

**Value**

**getCrispraiScores** returns a data.frame with grna\_id and score columns. The Weissman score takes on a value between 0 and 1. A higher score indicates higher sgRNA efficiency.

**Author(s)**

Pirunthan Perampalam, Jean-Philippe Fortin

**References**

Horlbeck et al. Compact and highly active next-generation libraries for CRISPR-mediated gene repression and activation eLife 2016;5:e19760. <https://doi.org/10.7554/eLife.19760>.

**Examples**

```
## Not run:
results <- getCrispraiScores(tss_df=tssExampleCrispra,
                             sgrna_df=sgrnaExampleCrispra,
                             modality="CRISPRa")

results <- getCrispraiScores(tss_df=tssExampleCrispri,
                             sgrna_df=sgrnaExampleCrispri,
                             modality="CRISPRi")

## End(Not run)
```

---

getCRISPRaterScores     *Calculate on-target sgRNA activity scores for Cas9 using CRISPRater*

---

**Description**

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the DeepHF scoring method. Both U6 and T7 promoters are supported. Three different versions of the SpCas9 nuclease are supported: wildtype (WT-SpCas9), high-fidelity Cas9 (SpCas9-HF1) and enhanced Cas9 (eSpCas9). Currently not supported on Windows machines.

**Usage**

```
getCRISPRaterScores(sequences)
```

**Arguments**

sequences     Character vector of 20bp protospacer sequences.

**Details**

Input sequences for CRISPRater scoring must be 20 spacer sequences.

**Value**

**getCrisprRaterScores** returns a data.frame with sequence and score columns. The CRISPRater score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

**Author(s)**

Jean-Philippe Fortin

**References**

Labuhn M, Adams FF, Ng M, et al. Refined sgRNA efficacy prediction improves large-and small-scale CRISPR–Cas9 applications. *Nucleic acids research*. 2018 Feb 16;46(3):1375-85.

**Examples**

```
spacer <- "ATCGATGCTGATGCTAGATA" #20bp
results <- getCRISPRaterScores(spacer)
```

---

getCRISPRscanScores     *Calculate on-target sgRNA activity scores for Cas9 using CRISPRscan*

---

**Description**

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the CRISPRscan scoring method. The method is also known as the Moreno-Mateos score. The CRISPRscan algorithm was trained using in vitro transcription of sgRNAs using a T7 promoter, and might therefore be suboptimal to predict sgRNA activity when expressed from U6 promoter.

**Usage**

```
getCRISPRscanScores(sequences)
```

**Arguments**

sequences     Character vector of 35bp sequences needed for CRISPRscan scoring, see details below.

**Details**

The input sequences for Rule Set 1 scoring require 6 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (23 nucleotides) and 6 nucleotides downstream of the protospacer sequence, for a total of 35 nucleotides: [6nt][20nt-spacer][NGG][6nt]. Note that a canonical PAM sequence (NGG) is required for CRISPRscan.

**Value**

**getCRISPRscanScores** returns a data.frame with sequence and score columns. The CRISPRscan score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

**Author(s)**

Jean-Philippe Fortin

**References**

Moreno-Mateos MA, et al. CRISPRscan: designing highly efficient sgRNAs for CRISPR-Cas9 targeting in vivo. *Nature methods*. 2015 Oct;12(10):982-8.

## Examples

```
flank5 <- "ACCTGG" #6bp
spacer <- "ATCGATGCTGATGCTAGATA" #20bp
pam <- "AGG" #3bp
flank3 <- "TTGAGC" #6bp
input <- paste0(flank5, spacer, pam, flank3)
results <- getCRISPRscanScores(input)
```

---

getDeepCpf1Scores	<i>Calculate on-target sgRNA activity scores for Cas12a using DeepCpf1</i>
-------------------	--

---

## Description

Calculate on-target sgRNA activity scores for CRISPR/Cas12a-induced knockout using the DeepCpf1 scoring method. Currently not supported on Windows machines.

## Usage

```
getDeepCpf1Scores(sequences, convertPAM = TRUE, fork = FALSE)
```

## Arguments

sequences	Character vector of 34bp sequences needed for DeepCpf1 scoring, see details below.
convertPAM	Should non-canonical PAM sequences be converted to TTTC? TRUE by default.
fork	Set to TRUE to preserve changes to the R configuration within the session.

## Details

The input sequences for DeepCpf1 scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (4bp PAM sequence + 23bp spacer sequence) and 3 nucleotides downstream of the protospacer sequence, for a total of 34 nucleotides. If convertPAM is set to TRUE, any non-canonical PAM sequence will be converted to TTTC for scoring purposes.

## Value

**getDeepCpf1Scores** returns a data.frame with sequence and score columns. The DeepCpf1 score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

## Author(s)

Jean-Philippe Fortin

## References

Kim, H., Min, S., Song, M. et al. Deep learning improves prediction of CRISPR-Cpf1 guide RNA activity. Nat Biotechnol 36, 239–241 (2018). <https://doi.org/10.1038/nbt.4061>.

**Examples**

```

if (interactive()){
flank5 <- "ACCG" #4bp
pam    <- "TTTT" #4bp
spacer <- "AATCGATGCTGATGCTAGATATT" #23bp
flank3 <- "AAG" #4bp
input  <- paste0(flank5, pam, spacer, flank3)
results <- getDeepCpf1Scores(input)
}

```

---

getDeepHFScores

*Calculate on-target sgRNA activity scores for Cas9 using DeepHF*


---

**Description**

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the DeepHF scoring method. Both U6 and T7 promoters are supported. Three different versions of the SpCas9 nuclease are supported: wildtype (WT-SpCas9), high-fidelity Cas9 (SpCas9-HF1) and enhanced Cas9 (eSpCas9). Currently not supported on Windows machines.

**Usage**

```

getDeepHFScores(
  sequences,
  enzyme = c("WT", "ESP", "HF"),
  promoter = c("U6", "T7"),
  fork = FALSE
)

```

**Arguments**

sequences	Character vector of 23bp protospacer sequences.
enzyme	Character string specifying the Cas9 variant. 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"). "U6" by default.
fork	Set to TRUE to preserve changes to the R configuration within the session.

**Details**

Input sequences for DeepHF scoring must be 23bp protospacer sequences (20bp spacer sequences + 3bp PAM sequences). Only canonical PAM sequences (NGG) are allowed. Users can specify for which Cas9 they wish to score sgRNAs by using the argument enzyme: "WT" for Wildtype Cas9 (WT-SpCas9), "HF" for high-fidelity Cas9 (SpCas9-HF), or "ESP" for enhanced Cas9 (eSpCas9). For wildtype Cas9, users can also specify the promoter used for expressing sgRNAs using the argument promoter ("U6" by default).

**Value**

**getDeepHFScores** returns a data.frame with sequence and score columns. The DeepHF score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

**Author(s)**

Jean-Philippe Fortin

**References**

Wang, D., Zhang, C., Wang, B. et al. Optimized CRISPR guide RNA design for two high-fidelity Cas9 variants by deep learning. Nat Commun 10, 4284 (2019). <https://doi.org/10.1038/s41467-019-12281-8>

**Examples**

```
if (interactive()){
  spacer <- "ATCGATGCTGATGCTAGATA" #20bp
  pam <- "AGG" #3bp
  input <- paste0(spacer, pam)

  # Wiltype Cas9 using U6 promoter:
  results <- getDeepHFScores(input)

  # Wiltype Cas9 using T7 promoter:
  results <- getDeepHFScores(input, promoter="T7")

  #' High-fidelity Cas9:
  results <- getDeepHFScores(input, enzyme="HF")

  #' Enhanced Cas9:
  results <- getDeepHFScores(input, enzyme="ESP")
}
```

---

getDeepSpCas9Scores	<i>Calculate on-target sgRNA activity scores for SpCas9 using DeepSp-Cas9</i>
---------------------	---

---

**Description**

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the DeepSp-Cas9 scoring method.

**Usage**

```
getDeepSpCas9Scores(sequences, fork = FALSE)
```

**Arguments**

sequences	Character vector of 30bp sequences needed for DeepSpCas9 scoring, see details below.
fork	Set to TRUE to preserve changes to the R configuration within the session.

**Details**

The input sequences for DeepSpCas9 scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (20bp spacer sequence + 3bp PAM sequence ) and 3 nucleotides downstream of the protospacer sequence, for a total of 30 nucleotides.

**Value**

**getDeepSpCas9Scores** returns a data.frame with sequence and score columns. The getDeepSpCas9Scores score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

**Author(s)**

Jean-Philippe Fortin

**References**

Kim HK, Kim Y, Lee S, et al. SpCas9 activity prediction by DeepSpCas9, a deep learning–base model with high generalization performance. Science advances. 2019 Nov 6;5(11):eaax9249.

**Examples**

```
if (interactive()){
  flank5 <- "ACCG" #4bp
  spacer <- "AATCGATGCTGATGCTAGAT" #20bp
  pam <- "AGG" #3bp
  flank3 <- "AAT" #3bp
  input <- paste0(flank5, spacer, pam, flank3)
  results <- getDeepSpCas9Scores(input)
}
```

---

getEnPAMGBScores	<i>Calculate on-target sgRNA activity scores for enCas12a using en-PAM+GB</i>
------------------	---

---

**Description**

Calculate on-target sgRNA activity scores for CRISPR/Cas12a-induced knockout using the en-PAM+GB scoring method. Currently not supported on Windows machines.

**Usage**

```
getEnPAMGBScores(sequences, fork = FALSE)
```

**Arguments**

sequences	Character vector of 34bp sequences needed for enPAM+GB scoring, see details below.
fork	Set to TRUE to preserve changes to the R configuration within the session.

**Details**

The input sequences for enPAM+GB scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (4bp PAM sequence + 23bp spacer sequence) and 3 nucleotides downstream of the protospacer sequence, for a total of 34 nucleotides. Both canonical and non-canonical PAM sequences can be provided.

**Value**

`getEnPAMGBScores` returns a data.frame with sequence and score columns.

**Author(s)**

Jean-Philippe Fortin

**References**

DeWeirdt, P.C., Sanson, K.R., Sangree, A.K. et al. Optimization of AsCas12a for combinatorial genetic screens in human cells. *Nat Biotechnol* 39, 94–104 (2021). <https://doi.org/10.1038/s41587-020-0600-6>.

**Examples**

```
if (interactive()){
  flank5 <- "CATG" #4bp
  pam    <- "TTTT" #4bp
  spacer <- "TTTGGGAACCAATCGATAATCAC" #23bp
  flank3 <- "ATT" #3bp
  input  <- paste0(flank5, pam, spacer, flank3)
  results <- getEnPAMGBScores(input)
}
```

---

getLindelScores	<i>Predict frameshift ratios from CRISPR/Cas9 indel prediction using Lindel</i>
-----------------	---

---

**Description**

Predict frameshift ratios from CRISPR/Cas9 indel prediction using the Lindel prediction algorithm.

**Usage**

```
getLindelScores(sequences, fork = FALSE)
```

**Arguments**

sequences	Character vector of 65bp sequences needed for Lindel scoring, see details below.
fork	Set to TRUE to preserve changes to the R configuration within the session.

**Details**

The input sequences for Lindel scoring require 13 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (23 nucleotides) and 29 nucleotides downstream of the protospacer sequence, for a total of 65 nucleotides. Note that only canonical PAM sequences (NGG) are accepted by Lindel.

**Value**

A data.frame with predicted frameshift ratio (between 0 and 1). A higher ratio indicates a greater chance of a frameshift indel introduced by CRISPR/Cas9-induced double-strand breaks.

**Author(s)**

Jean-Philippe Fortin

**References**

Wei Chen, Aaron McKenna, Jacob Schreiber, Maximilian Haeussler, Yi Yin, Vikram Agarwal, William Stafford Noble, Jay Shendure, Massively parallel profiling and predictive modeling of the outcomes of CRISPR/Cas9-mediated double-strand break repair, *Nucleic Acids Research*, Volume 47, Issue 15, 05 September 2019, Pages 7989–8003, <https://doi.org/10.1093/nar/gkz487>.

**Examples**

```
if (interactive()){
  flank5 <- "ACCTTTTAATCGA" #13bp
  spacer <- "TGCTGATGCTAGATATTAAG" #20bp
  pam <- "TGG" #3bp
  flank3 <- "CTTTTAATCGATGCTGATGCTAGATATTA" #29bp
  input <- paste0(flank5, spacer, pam, flank3)
  results <- getLindelScores(input)
}
```

---

`getMITScores`*Calculate MIT off-target specificity scores for CRISPR/Cas9*

---

**Description**

Calculate MIT off-target specificity scores for CRISPR/Cas9.

**Usage**

```
getMITScores(spacers, protospacers, pams, includeDistance = TRUE)
```

**Arguments**

<code>spacers</code>	Character vector of 20bp spacer sequences.
<code>protospacers</code>	Character vector of 20bp protospacer sequences for off-targets.
<code>pams</code>	Character vector of 3nt PAM sequences.
<code>includeDistance</code>	Should distance between mismatches be considered during scoring? TRUE by default.

**Value**

`getMITScores` returns a data.frame with `spacer`, `protospacer`, and `score` columns. The MIT score takes on a value between 0 and 1. For a given pair (on-target, off-target), a higher MIT score indicates a higher likelihood for the Cas9 nuclease to cut at the off-target. Non-canonical PAM sequences are taken into account by the MIT algorithm.

**Author(s)**

Jean-Philippe Fortin

## References

Hsu, P., Scott, D., Weinstein, J. et al. DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat Biotechnol* 31, 827–832 (2013). <https://doi.org/10.1038/nbt.2647>.

## Examples

```
# Calculating MIT scores for two off-targets with respect to
# one spacer sequence:
spacer <- "AGGTGTAGTGTGTGATAA"
protospacer1 <- "CGGTGTAGTGTGTGATAA"
protospacer2 <- "CGGTGTCGTGTGTGATAA"
pams <- c("AGG", "CGG")
results <- getMITScores(spacers=spacer,
  protospacers=c(protospacer1, protospacer2),
  pams=pams
)
```

---

getRuleSet1Scores	<i>Calculate on-target sgRNA activity scores for Cas9 using Rule Set 1</i>
-------------------	--

---

## Description

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the Rule Set 1 scoring method. The Rule Set 1 algorithm was an early on-target efficiency method developed by the Doench lab.

## Usage

```
getRuleSet1Scores(sequences)
```

## Arguments

sequences	Character vector of 30bp sequences needed for Rule Set 1 scoring, see details below.
-----------	--

## Details

The input sequences for Rule Set 1 scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (23 nucleotides) and 3 nucleotides downstream of the protospacer sequence, for a total of 30 nucleotides: [4nt][20nt-spacer][NGG][3nt]. Note that a canonical PAM sequence (NGG) is required for Rule Set 1.

## Value

**getRuleSet1Scores** returns a data.frame with sequence and score columns. The Rule Set 1 score takes on a value between 0 and 1. A higher score indicates higher knockout efficiency.

## Author(s)

Jean-Philippe Fortin

## References

Doench, John G., et al. Rational design of highly active sgRNAs for CRISPR-Cas9-mediated gene inactivation. *Nat Biotech* 32, 1262-1267 (2014). <https://doi.org/10.1038/nbt.3026>.

## Examples

```
flank5 <- "ACCT" #4bp
spacer <- "ATCGATGCTGATGCTAGATA" #20bp
pam <- "AGG" #3bp
flank3 <- "TTG" #3bp
input <- paste0(flank5, spacer, pam, flank3)
results <- getRuleSet1Scores(input)
```

---

getRuleSet3Scores	<i>Calculate on-target sgRNA activity scores for SpCas9 using Rule Set 3</i>
-------------------	--

---

## Description

Calculate on-target sgRNA activity scores for CRISPR/Cas9-induced knockout using the Rule Set 3 scoring method.

## Usage

```
getRuleSet3Scores(
  sequences,
  tracrRNA = c("Hsu2013", "Chen2013"),
  mode = c("sequence", "target")
)
```

## Arguments

sequences	Character vector of 30bp sequences needed for Rule Set 3 scoring, see details below.
tracrRNA	String specifying which tracrRNA is used. Must be either "Hsu2013" (default) or "Chen2013".
mode	String specifying which prediction mode is used. Must be either "sequence" (default) or "target".

## Details

The input sequences for Rule Set 3 scoring require 4 nucleotides upstream of the protospacer sequence, the protospacer sequence itself (20bp spacer sequence + 3bp PAM sequence) and 3 nucleotides downstream of the protospacer sequence, for a total of 30 nucleotides.

## Value

**getRuleSet3Scores** returns a data.frame with sequence and score columns. The getRuleSet3Scores score is similar to a Z-score. A higher score indicates higher knockout efficiency.

**Author(s)**

Jean-Philippe Fortin

**References**

doi: <https://doi.org/10.1101/2022.06.27.497780>

**Examples**

```
if (interactive()){
  flank5 <- "ACCG" #4bp
  spacer <- "AATCGATGCTGATGCTAGAT" #20bp
  pam <- "AGG" #3bp
  flank3 <- "AAT" #3bp
  input <- paste0(flank5, spacer, pam, flank3)
  results <- getRuleSet3Scores(input)
}
```

---

scoringMethodsInfo     *data.frame detailing available scoring methods*

---

**Description**

data.frame detailing available scoring methods with information needed to extract nucleotide sequences needed by each scoring algorithm.

**Usage**

```
data(scoringMethodsInfo)
```

**Format**

A data frame with 6 columns:

**method** name of the scoring method

**nuclease** nuclease compatible with the scoring method

**left** upstream offset (relative to PAM site) to extract nucleotide sequence needed for scoring

**right** downstream offset (relative to PAM site) to extract nucleotide sequence needed for scoring

**type** type of the scoring algorithm (on-target or off-target)

**label** proper case-sensitive method name for labeling

**len** length of the nucleotide sequence needed for scoring

---

sgrnaExampleCrispra    *Example CRISPRa gRNAs data.frame for the getCrispraiScores function*

---

### Description

Example CRISPRa gRNAs data.frame for the getCrispraiScores function. The targeted TSSs are described in the object tssExampleCrispra.

### Usage

```
data(sgrnaExampleCrispra)
```

### Format

A data.frame with 5 columns:

**grna\_id** String specifying gRNA unique identifier.

**tss\_id** String specifying the targeted TSS id.

**pam\_site** Genomic coordinate specifying the first nucleotide of PAM sequence.

**strand** Strand of the gRNA strand. Must be "+" or "-".

**spacer\_19mer** String specifying the nucleotide sequence of the 19mer spacer sequence.

---

sgrnaExampleCrispri    *Example CRISPRi gRNAs data.frame for the getCrispraiScores function*

---

### Description

Example CRISPRi gRNAs data.frame for the getCrispraiScores function. The targeted TSSs are described in the object tssExampleCrispri.

### Usage

```
data(sgrnaExampleCrispri)
```

### Format

A data.frame with 5 columns:

**grna\_id** String specifying gRNA unique identifier.

**tss\_id** String specifying the targeted TSS id.

**pam\_site** Genomic coordinate specifying the first nucleotide of PAM sequence.

**strand** Strand of the gRNA strand. Must be "+" or "-".

**spacer\_19mer** String specifying the nucleotide sequence of the 19mer spacer sequence.

---

tssExampleCrispra      *Example TSS data.frame for the getCrispraiScores function*

---

**Description**

Example TSS data for gRNAs stored in sgrnaExampleCrispra.

**Usage**

```
data(tssExampleCrispra)
```

**Format**

A data.frame with 7 columns:

**tss\_id** String specifying the targeted TSS id.

**gene\_symbol** String specifying gene symbol.

**promoter** String specifying the promoter suffix to add to the gene symbol columns to obtain the unique TSS id.

**transcripts** Ensembl IDs of the targeted transcript.

**position** Integer specifying genomic coordinate of the TSS.

**strand** Strand of TSS. Must be "+" or "-".

**chr** String specifying the chromosome name.

---

tssExampleCrispri      *Example TSS data.frame for the getCrispraiScores function*

---

**Description**

Example TSS data for gRNAs stored in sgrnaExampleCrispri.

**Usage**

```
data(tssExampleCrispri)
```

**Format**

A data.frame with 7 columns:

**tss\_id** String specifying the targeted TSS id.

**gene\_symbol** String specifying gene symbol.

**promoter** String specifying the promoter suffix to add to the gene symbol columns to obtain the unique TSS id.

**transcripts** Ensembl IDs of the targeted transcript.

**position** Integer specifying genomic coordinate of the TSS.

**strand** Strand of TSS. Must be "+" or "-".

**chr** String specifying the chromosome name.

# Index

## \* datasets

- scoringMethodsInfo, [17](#)
- sgrnaExampleCrispra, [18](#)
- sgrnaExampleCrispri, [18](#)
- tssExampleCrispra, [19](#)
- tssExampleCrispri, [19](#)

- getAzimuthScores, [2](#)
- getCasRxRFScores, [3](#)
- getCFDScores, [5](#)
- getCrispraiScores, [6](#)
- getCRISPRaterScores, [7](#)
- getCRISPRscanScores, [8](#)
- getDeepCpf1Scores, [9](#)
- getDeepHFScores, [10](#)
- getDeepSpCas9Scores, [11](#)
- getEnPAMGBScores, [12](#)
- getLindelScores, [13](#)
- getMITScores, [14](#)
- getRuleSet1Scores, [15](#)
- getRuleSet3Scores, [16](#)

- scoringMethodsInfo, [17](#)
- sgrnaExampleCrispra, [18](#)
- sgrnaExampleCrispri, [18](#)

- tssExampleCrispra, [19](#)
- tssExampleCrispri, [19](#)