# Package 'crisprBase'

November 11, 2026

**Version** 1.16.0

**Date** 2025-10-27

**Title** Base functions and classes for CRISPR gRNA design

**Depends** utils, methods, R (>= 4.1)

**Imports** BiocGenerics, Biostrings, GenomicRanges, graphics, IRanges, S4Vectors, stringr

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

**biocViews** CRISPR, FunctionalGenomics

**Description** Provides S4 classes for general nucleases, CRISPR nucleases, CRISPR nickases, and base editors.Several CRISPR-specific genome arithmetic functions are implemented to help extract genomic coordinates of spacer and protospacer sequences. Commonly-used CRISPR nuclease objects are provided that can be readily used in other packages. Both DNA- and RNA-targeting nucleases are supported.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**BugReports** https://github.com/crisprVerse/crisprBase/issues

**URL** https://github.com/crisprVerse/crisprBase

**Collate** 'AllGenerics.R' 'rebase.R' 'Nuclease-class.R' 'Nickase-class.R' 'CrisprNuclease-class.R' 'CrisprNickase-class.R' 'BaseEditor-class.R' 'arithmetics.R' 'data.R' 'utils.R' 'converters.R' 'annotateMismatches.R'

**LazyData** true

**git_url** https://git.bioconductor.org/packages/crisprBase

**git_branch** RELEASE_3_23

**git_last_commit** 00f5ba9

**git_last_commit_date** 2026-04-28

**Repository** Bioconductor 3.23

**Date/Publication** 2026-11-11

**Author** Jean-Philippe Fortin [aut, cre]

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

# Contents

annotateMismatches . . . 2
AsCas12a . . . 3
baseEditorName . . . 4
BE4max . . . 6
CasRx . . . 7
CrisprNickase-class . . . 7
Csm . . . 10
enAsCas12a . . . 11
extractPamFromTarget . . . 11
extractProtospacerFromTarget . . . 12
getAvailableCrisprNucleases . . . 13
getCutSiteFromPamSite . . . 14
getCutSiteRanges . . . 14
getEditingSiteFromPamSite . . . 15
getPamRanges . . . 16
getProtospacerRanges . . . 17
getTargetRanges . . . 18
MAD7 . . . 19
motifs . . . 20
nickaseName . . . 23
plotEditingWeights . . . 27
reexports . . . 27
restrictionEnzymes . . . 28
SaCas9 . . . 28
spacerLength . . . 29
SpCas9 . . . 32
SpG . . . 33

**Index** **34**

---

`annotateMismatches` *Annotate mismatches between spacer and protospacer sequences*

---

## Description

Annotate mismatches between spacer and protospacer sequences.

## Usage

```
annotateMismatches(spacers, protospacers, rnase = FALSE)
```

### Arguments

| | |
|---|---|
| spacers | A character vector specifying spacer sequences (gRNA). |
| protospacers | A character vector specifying protospacer sequences (target DNA). |
| rnase | Is it for an RNAse? FALSE by default. If TRUE, spacers and protospacers are expected to be the reverse complement of each other. |

### Value

A data.frame storing spacer and protospacer columns, as well as number of mismatches, and positions for the different mismatches, if any. Positions are relative to the 5' end of the spacer sequences. For RNAses (e.g. CasRx), this means that a mismatch at position 1 corresponds to the last nucleotide of the protospacer sequence.

### Author(s)

Jean-Philippe Fortin

### Examples

```
spacers <- c("CCGGAGCGAGTTGCAGTAAGCAG",
    "GCCGGAGCGAGTTGCAGTAAGCA",
    "GGCCGGAGCGAGTTGCAGTAAGC")

protospacers=c("CTGCTTACTGCAACTCGCTCTGG",
    "TGCTTAATGCAACCCGCTCCGGC",
    "GCTTACTGCAACTCGCTCCGGCC")

ann <- annotateMismatches(spacers,
    protospacers,
    rnase=TRUE)
```

---

## AsCas12a *AsCas12a CrisprNuclease object*

---

### Description

CrisprNuclease object for the Wildtype Acidaminococcus Cas12a (AsCas12a) nuclease.

### Usage

```
data(AsCas12a, package="crisprBase")
```

### Format

CrisprNuclease object.

## Details

The AsCas12a nuclease recognizes TTTV PAM sequences. Spacer sequences must be located downstream of PAM sequences.

---

`baseEditorName` *An S4 class to represent a base editor*

---

## Description

An S4 class to represent a base editor

## Usage

```
baseEditorName(object)

baseEditorName(object) <- value

editingWeights(object, ...)

editingStrand(object, ...)

editingStrand(object) <- value

BaseEditor(
  CrisprNuclease,
  baseEditorName = NA_character_,
  editingStrand = c("original", "opposite"),
  editingWeights = NULL,
  scale = TRUE
)

## S4 method for signature 'BaseEditor'
show(object)

## S4 method for signature 'BaseEditor'
baseEditorName(object)

## S4 replacement method for signature 'BaseEditor'
baseEditorName(object) <- value

## S4 method for signature 'BaseEditor'
editingWeights(object, substitutions = NULL)

setEditingWeights(object, value, scale = TRUE)

## S4 method for signature 'BaseEditor'
```

```
editingStrand(object)

## S4 replacement method for signature 'BaseEditor'
editingStrand(object) <- value
```

## Arguments

| | |
|---|---|
| object | BaseEditor object. |
| value | Value to replaced with. |
| ... | Additional arguments for class-specific methods |
| CrisprNuclease | A CrisprNuclease object. |
| baseEditorName | String specifying base editor name. |
| editingStrand | String indicating which strand with respect to the target protospacer sequence will be edited. Must be either "original" or "opposite". "original" by default. |
| editingWeights | Numeric matrix of editing weights. Column names must be indicating relative position to the PAM site. Row names must be of the form "X2Y" where "X" represents the origin base, and "Y" represents the subtituted base. For instance, "C2T" indicates the row corresponding to C to T editing. |
| scale | Logical indicating if weights should be scaled to have a maximum of 1. TRUE by default. Should be set to FALSE if editing weights are already representing editing probabilities. |
| substitutions | Character vector indicating which substitutions should be returned. |

## Value

A BaseEditor object

## Functions

- `BaseEditor()`: Create a BaseEditor object

## Slots

`baseEditorName` Name of the base editor.

`editingWeights` Matrix of editing weights.

`editingStrand` String indicating which strand with respect to the target protospacer sequence will be edited. Must be either "original" or "opposite". "original" by default.

## Constructors

Use the constructor `link{BaseEditor}` to create a BaseEditor object.

## Accessors

`baseEditorName`**:** To get the name of the base editor.

`editingWeights`**:** To return the matrix of editing weights.

`editingStrand`**:** To return the editing strand.

### Setters

`baseEditorName<-`: To change the name of the base editor.

`editingWeights<-`: To change the matrix of editing weights.

`editingStrand<-`: To change the editing strand.

### Examples

```
# Creating an object for BE4max (C to T editor)
# based on experimental weights

ws <- c(0.7, 0.7, 0.8, 1.8, 1, 2, 1.4, 1.2, 2.3, 1.3, 2.4, 2.2, 3.4,
      2.2, 2.1, 3.5, 5.8, 16.2, 31.8, 63.2, 90.3, 100, 87, 62, 31.4,
      16.3, 10, 5.6, 3.3, 1.9, 1.8, 2.4, 1.7, 0.5, 0.2, 0.1)
ws <- matrix(ws, nrow=1, ncol=length(ws))
rownames(ws) <- "C2T"
colnames(ws) <- -36:-1
data(SpCas9, package="crisprBase")
BE4max <- BaseEditor(SpCas9,
                     baseEditorName="BE4max",
                     editingStrand="original",
                     editingWeights=ws)
metadata(BE4max)$description_base_editor <- "BE4max cytosine base editor."
```

---

## BE4max *BE4max BaseEditor object*

---

### Description

BaseEditor for the cytosine base editor CRISPR/Cas9 system BE4max. Editing weights were obtained from https://doi.org/10.1016/j.cell.2020.05.037

### Usage

```
data(BE4max, package="crisprBase")
```

### Format

BaseEditor object.

### Details

BaseEditor for the cytosine base editor CRISPR/Cas9 system BE4max. Editing weights were obtained from https://doi.org/10.1016/j.cell.2020.05.037.

## CasRx *CasRx CrisprNuclease object*

### Description

CrisprNuclease object for the Cas13d-NLS from Ruminococcus flavefaciens strain XPD3002 nuclease (RNase).

### Usage

```
data(CasRx, package="crisprBase")
```

### Format

CrisprNuclease object.

### Details

The CasRx nuclease was derived from Cas13d Ruminococcus flavefaciens string XPD3002. See 10.1016/j.cell.2018.02.033.

## CrisprNickase-class *An S4 class to represent a CRISPR nickase.*

### Description

An S4 class to represent a CRISPR nickase.

### Usage

```
CrisprNickase(
  nickaseName,
  nickingStrand = c("original", "opposite"),
  pams = NA_character_,
  weights = rep(1, length(pams)),
  metadata = list(),
  pam_side = NA_character_,
  spacer_gap = 0L,
  spacer_length = NA_integer_
)

## S4 method for signature 'CrisprNickase'
show(object)

## S4 method for signature 'CrisprNickase'
pamLength(object)
```

```
## S4 method for signature 'CrisprNickase'
spacerLength(object)

## S4 replacement method for signature 'CrisprNickase'
spacerLength(object) <- value

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

## S4 replacement method for signature 'CrisprNickase'
pamSide(object) <- value

## S4 method for signature 'CrisprNickase'
spacerGap(object)

## S4 replacement method for signature 'CrisprNickase'
spacerGap(object) <- value

## S4 method for signature 'CrisprNickase'
hasSpacerGap(object)

## S4 method for signature 'CrisprNickase'
targetLength(object)

## S4 method for signature 'CrisprNickase'
pams(object, primary = TRUE, ignore_pam = FALSE, as.character = FALSE)

## S4 method for signature 'CrisprNickase'
pamIndices(object)

## S4 method for signature 'CrisprNickase'
spacerIndices(object)

## S4 method for signature 'CrisprNickase'
prototypeSequence(object, primary = TRUE)
```

## Arguments

| | |
|---|---|
| nickaseName | Name of the CRISPR nickase. |
| nickingStrand | String specifying with strand with respect to the motif sequence (5' to 3') is nicked. Must be either "original" (default) or "opposite". |
| pams | Character vector of PAM sequence motifs written from 5' to 3. If the point of cleavage has been determined, the precise site is marked with ^. Only letters in the IUPAC code are accepted. For nickases that cleave away from their recognition sequence, the cleavage sites are indicated in parentheses. See details for more information. |

| | |
|---|---|
| weights | Optional numeric vector specifying relative weights of the PAM sequences to specify cleavage probabilities. |
| metadata | Optional list providing global metadata information. |
| pam_side | String specifying the side of the PAM sequence sequence with respect to the protospacer sequence. Must be either '3prime' (e.g. Cas9) or '5prime' (e.g. Cas12a) |
| spacer_gap | Integer specifying the length (in nucleotides) between the spacer sequence and the PAM sequence (e.g. 0 for Cas9 and Cas12a). |
| spacer_length | Integer specifying the length of the spacer sequence |
| object | CrisprNickase object. |
| value | For spacerLength<- and gapLength<-, must be a non-negative integer. For pamSide, must be either '5prime' or '3prime'. |
| primary | Should only the PAM sequence with the heighest weight be returned? If no cleavage weights are stored in the CrisprNickase object, all sequences are returned. TRUE by default. |
| ignore_pam | Should all possible k-mer sequences for a given PAM length be returned, irrespetively of the PAM sequence motifs stored in the CrisprNickase object? FALSE by default. |
| as.character | Should the PAM sequences be returned as a character vector? FALSE by default. |

## Value

A CrisprNickase object

## Functions

- `CrisprNickase()`: Create a CrisprNickase object

## Slots

`pam_side` String specifying the side of the PAM sequence with respect to the protospacer sequence. Must be either '3prime' (e.g. SpCas9) or '5prime' (e.g. AsCas12a)

`spacer_length` Integer specifying the length of the spacer sequence

`spacer_gap` Integer specifying the length (in nucleotides) between the spacer sequence and the PAM sequence (e.g. 0 for SpCas9 and AsCas12a).

## Constructors

Use the constructor `link{CrisprNickase}` to create a CrisprNickase object.

## Accessors

`nickaseName`**:** To get the name of the CRISPR nickase.

`spacerLength`**:** To return the length of the spacer sequence.

`targetLength`**:** To return the length of the target sequence (protospacer + pam).

`pamLength`: To return the length of the PAM sequence.

`pamSide`: To return the side of the PAM sequence with respect to the spacer sequence.

`spacerGap`: To return the length of the gap between the PAM and spacer sequences.

`pams`: To return the list of PAM sequences.

### Setters

`spacerGap<-`: To change the length of the gap between the PAM and spacer sequences.

`pamSide<-`: To change the side of the PAM sequence with respect to the protospacer sequence.

`spacerLength<-`: To change the length of the spacer sequence.

### Utility functions for genomic arithmetics

`pamIndices`: To return the relative coordinates of the PAM sequence within the protospacer sequence.

`spacerIndices`: To return the relatiive coordinates of the spacer sequence within the protospacer sequence.

### Examples

```
Cas9D10A <- CrisprNickase("Cas9D10A",
                          nickingStrand="opposite",
                          pams=c("(3)NGG", "(3)NAG", "(3)NGA"),
                          weights=c(1, 0.2593, 0.0694),
                          metadata=list(description="D10A-mutated  Streptococcus
                                        pyogenes Cas9 (SpCas9) nickase"),
                          pam_side="3prime",
                          spacer_length=20)

Cas9H840A <- CrisprNickase("Cas9H840A",
                           nickingStrand="original",
                           pams=c("(3)NGG", "(3)NAG", "(3)NGA"),
                           weights=c(1, 0.2593, 0.0694),
                           metadata=list(description="H840A-mutated  Streptococcus
                                         pyogenes Cas9 (SpCas9) nickase"),
                           pam_side="3prime",
                           spacer_length=20)
```

---

## Csm *Csm CrisprNuclease object*

---

### Description

CrisprNuclease object for the RNA-targeting Csm complex from Streptococcus thermophilus

## Usage

```
data(Csm, package="crisprBase")
```

## Format

CrisprNuclease object.

## Details

The specific Csm complex is an RNA-targeting nuclease derived from Streptococcus thermophilus. There is no preferred PAM sequences, and the default (optimal) spacer length is 32nt. See https://doi.org/10.1038/s41587-022-01649-9.

---

# enAsCas12a *enAsCas12a CrisprNuclease object*

---

## Description

CrisprNuclease object for the Enhanced Acidaminococcus Cas12a (AsCas12a) nuclease.

## Usage

```
data(enAsCas12a, package="crisprBase")
```

## Format

CrisprNuclease object.

## Details

The enAsCas12a nuclease recognizes an extended set of PAM sequences beyong the canonical TTTV sequence for AsCas12a. Spacer sequences must be located downstream of PAM sequences.

---

# extractPamFromTarget *Extract PAM sequences from target sequences*

---

## Description

Extract PAM sequences from target sequences (protospacer + PAM) using information stored in a CrisprNuclease object.

## Usage

```
extractPamFromTarget(targets, object)
```

### Arguments

| | |
|---|---|
| targets | Character vector of target sequences. |
| object | CrisprNuclease corresponding to the target sequences. |

### Value

Character vector of PAM sequences of length equal to that of the targets character vector.

### Author(s)

Jean-Philippe Fortin

### Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
# Extracting PAM sequences from Cas9 protospacers:
targets <- c("AGGTGCTGATTGTAGTGCTGCGG",
             "AGGTGCTGATTGTAGTGCTGAGG")
extractPamFromTarget(targets, SpCas9)
# Extracting PAM sequences from Cas12a targets:
targets <- c("TTTAAGGTGCTGATTGTAGTGCTGTGT",
            "TTTCAGGTGCTGATTGTAGTGCTGAAA")
extractPamFromTarget(targets, AsCas12a)
```

---

## extractProtospacerFromTarget

*Extract protospacer sequences from target sequences*

---

### Description

Extract protospacer sequences from target sequences (protospacer + PAM) using information stored in a CrisprNuclease object.

### Usage

```
extractProtospacerFromTarget(targets, object)
```

### Arguments

| | |
|---|---|
| targets | Character vector of targets sequences. |
| object | CrisprNuclease corresponding to the targets sequences. |

### Value

Character vector of protospacer sequences of length equal to that of the targets character vector.

### Author(s)

Jean-Philippe Fortin

### Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
# Extracting protospacer sequences from Cas9 targets:
targets <- c("AGGTGCTGATTGTAGTGCTGCGG",
             "AGGTGCTGATTGTAGTGCTGAGG")
extractProtospacerFromTarget(targets, SpCas9)
# Extracting protospacer sequences from Cas12a targets:
targets <- c("TTTAAGGTGCTGATTGTAGTGCTGTGT",
             "TTTCAGGTGCTGATTGTAGTGCTGAAA")
extractProtospacerFromTarget(targets, AsCas12a)
```

---

## getAvailableCrisprNucleases

*Return list of available CrisprNuclease objects in crisprBase*

---

### Description

Return list of available CrisprNuclease objects in crisprBase.

### Usage

```
getAvailableCrisprNucleases()
```

### Value

Character vector of available `CrisprNuclease` objects found in `crisprBase`.

### Author(s)

Jean-Philippe Fortin

### Examples

```
getAvailableCrisprNucleases()
```

## getCutSiteFromPamSite *Return cut site coordinates from PAM site coordinates*

### Description

Return cut site coordinates from PAM site coordinates.

### Usage

```
getCutSiteFromPamSite(pam_site, strand, nuclease = NULL)
```

### Arguments

| | |
|---|---|
| pam_site | Coordinate of the first nucleotide of the PAM sequence. |
| strand | Either "+" or "-". |
| nuclease | A CrisprNuclease object. |

### Value

numeric vector of cut sites

### Author(s)

Jean-Philippe Fortin

### Examples

```
data(SpCas9, package="crisprBase")
getCutSiteFromPamSite(pam_site=100, strand="+", nuclease=SpCas9)
getCutSiteFromPamSite(pam_site=100, strand="-", nuclease=SpCas9)
```

## getCutSiteRanges *Construct a cut site GRanges from a list of PAM sites*

### Description

Construct a cut site GRanges from a list of PAM sites using information stored in a CrisprNuclease object.

## Usage

```
getCutSiteRanges(
  gr = NULL,
  seqnames = NULL,
  pam_site = NULL,
  strand = NULL,
  nuclease = NULL
)
```

## Arguments

| | |
|---|---|
| gr | GRanges object of width 1 specifying the coordinates of the first nucleotide of the PAM sequences. |
| seqnames | Character vector of genomic sequence names. Ignored if gr is not NULL. |
| pam_site | Numeric vector specifying the coordinates of the first nucleotide of the PAM sequences corresponding to the PAM sequences. Ignored if gr is not NULL. |
| strand | Character vector specifying the strand of the PAM. Ignored if gr is not NULL. |
| nuclease | CrisprNuclease object. |

## Value

GRanges object representing genomic coordinates of the cut sites.

## Author(s)

Jean-Philippe Fortin

## Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
library(GenomicRanges)
gr <- GRanges("chr10",
              IRanges(start=c(100,120), width=1),
              strand=c("+","-"))
getCutSiteRanges(gr, nuclease=SpCas9)
getCutSiteRanges(gr, nuclease=AsCas12a)
```

---

# getEditingSiteFromPamSite

*Return optimal editing site coordinates from PAM site coordinates*

---

## Description

Return optimal editing site coordinates from PAM site coordinates.

## Usage

```
getEditingSiteFromPamSite(
  pam_site,
  strand,
  baseEditor = NULL,
  substitution = NULL
)
```

## Arguments

| | |
|---|---|
| pam_site | Coordinate of the first nucleotide of the PAM sequence. |
| strand | Either "+" or "-". |
| baseEditor | A BaseEditor object. |
| 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

numeric vector of editing sites.

## Author(s)

Jean-Philippe Fortin

## Examples

```
data(BE4max, package="crisprBase")
getEditingSiteFromPamSite(pam_site=100, strand="+", baseEditor=BE4max, "C2T")
```

---

# getPamRanges *Construct a PAM GRanges from a list of PAM sites*

---

## Description

Construct a PAM GRanges from a list of PAM sites using information stored in a CrisprNuclease object.

## Usage

```
getPamRanges(
  gr = NULL,
  seqnames = NULL,
  pam_site = NULL,
  strand = NULL,
  nuclease = NULL
)
```

### Arguments

| | |
|---|---|
| gr | GRanges object of width 1 specifying the coordinates of the first nucleotide of the PAM sequences. |
| seqnames | Character vector of genomic sequence names. Ignored if gr is not NULL. |
| pam_site | Numeric vector specifying the coordinates of the first nucleotide of the PAM sequences corresponding to the PAM sequences. Ignored if gr is not NULL. |
| strand | Character vector specifying the strand of the PAM. Ignored if gr is not NULL. |
| nuclease | CrisprNuclease object. |

### Value

GRanges object representing genomic coordinates of PAM sequences.

### Author(s)

Jean-Philippe Fortin

### Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
library(GenomicRanges)
gr <- GRanges("chr10",
              IRanges(start=c(100,120), width=1),
              strand=c("+","-"))
getPamRanges(gr, nuclease=SpCas9)
getPamRanges(gr, nuclease=AsCas12a)
```

---

## getProtospacerRanges *Construct a protospacer GRanges from a list of PAM sites*

---

### Description

Construct a protospacer GRanges from a list of PAM sites using information stored in a CrisprNuclease object.

### Usage

```
getProtospacerRanges(
  gr = NULL,
  seqnames = NULL,
  pam_site = NULL,
  strand = NULL,
  nuclease = NULL,
  spacer_len = NULL
)
```

### Arguments

| | |
|---|---|
| gr | GRanges object of width 1 specifying the coordinates of the first nucleotide of the PAM sequences. |
| seqnames | Character vector of genomic sequence names. Ignored if gr is not NULL. |
| pam_site | Numeric vector specifying the coordinates of the first nucleotide of the PAM sequences corresponding to the protospacers. Ignored if gr is not NULL. |
| strand | Character vector specifying the strand of the protospacer. Ignored if gr is not NULL. |
| nuclease | CrisprNuclease object. |
| spacer_len | Non-negative integer to overwrite the default spacer length stored in the CrisprNuclease object. s |

### Value

GRanges object representing genomic coordinates of protospacer sequences.

### Author(s)

Jean-Philippe Fortin

### Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
library(GenomicRanges)
gr <- GRanges("chr10",
              IRanges(start=c(100,120), width=1),
              strand=c("+","-"))
getProtospacerRanges(gr, nuclease=SpCas9)
getProtospacerRanges(gr, nuclease=AsCas12a)
```

---

## getTargetRanges *Construct a target GRanges from a list of PAM sites*

---

### Description

Construct a target (protospacer + PAM) GRanges from a list of PAM sites using information stored in a CrisprNuclease object.

### Usage

```
getTargetRanges(
  gr = NULL,
  seqnames = NULL,
  pam_site = NULL,
  strand = NULL,
```

```
  nuclease = NULL,
  spacer_len = NULL
)
```

## Arguments

| | |
|---|---|
| gr | GRanges object of width 1 specifying the coordinates of the first nucleotide of the PAM sequences. |
| seqnames | Character vector of genomic sequence names. Ignored if gr is not NULL. |
| pam_site | Numeric vector specifying the coordinates of the first nucleotide of the PAM sequences corresponding to the targets. Ignored if gr is not NULL. |
| strand | Character vector specifying the strand of the target. Ignored if gr is not NULL. |
| nuclease | CrisprNuclease object. |
| spacer_len | Non-negative integer to overwrite the default spacer length stored in the CrisprNuclease object. |

## Value

GRanges object representing genomic coordinates of the target sequences.

## Author(s)

Jean-Philippe Fortin

## Examples

```
data(SpCas9, AsCas12a, package="crisprBase")
library(GenomicRanges)
gr <- GRanges("chr10",
              IRanges(start=c(100,120), width=1),
              strand=c("+","-"))
getTargetRanges(gr, nuclease=SpCas9)
getTargetRanges(gr, nuclease=AsCas12a)
```

---

# MAD7 *MAD7 CrisprNuclease object*

---

## Description

CrisprNuclease object for the MAD7 nuclease (Cas12a-like nuclease)

## Usage

```
data(MAD7, package="crisprBase")
```

## Format

CrisprNuclease object.

## Details

The MAD7 nuclease recognizes YTTV PAM sequences. Spacer sequences must be located downstream of PAM sequences.

---

`motifs` *An S4 class to represent a nuclease.*

---

## Description

Return motif string representations of recognition sites.

Return length of the recognition sites sequences.

## Usage

```
motifs(object, ...)

motifLength(object, ...)

nucleaseName(object)

targetType(object)

weights(object, ...)

nucleaseName(object) <- value

targetType(object) <- value

weights(object) <- value

cutSites(object, ...)

isCutting(object)

isRnase(object)

isDnase(object)

Nuclease(
  nucleaseName,
  targetType = c("DNA", "RNA"),
  motifs = NULL,
```

```
  cutSites = NULL,
  weights = rep(1, length(motifs)),
  metadata = list()
)

## S4 method for signature 'Nuclease'
show(object)

## S4 method for signature 'Nuclease'
nucleaseName(object)

## S4 replacement method for signature 'Nuclease'
nucleaseName(object) <- value

## S4 method for signature 'Nuclease'
targetType(object)

## S4 replacement method for signature 'Nuclease'
targetType(object) <- value

## S4 method for signature 'Nuclease'
weights(object, expand = FALSE)

## S4 replacement method for signature 'Nuclease'
weights(object) <- value

## S4 method for signature 'Nuclease'
isCutting(object)

## S4 method for signature 'Nuclease'
isRnase(object)

## S4 method for signature 'Nuclease'
isDnase(object)

## S4 method for signature 'Nuclease'
motifs(
  object,
  primary = FALSE,
  strand = c("+", "-"),
  expand = FALSE,
  as.character = FALSE
)

## S4 method for signature 'Nuclease'
motifLength(object)

## S4 method for signature 'Nuclease'
```

```
cutSites(object, strand = c("+", "-", "both"), combine = TRUE, middle = FALSE)
```

## Arguments

| | |
|---|---|
| object | Nuclease object. |
| ... | Additional arguments for class-specific methods |
| value | New value to pass to the setter functions. |
| nucleaseName | Name of the nuclease. |
| targetType | String specifying target type ("DNA" or "RNA"). |
| motifs | Character vector of recognition sequence motifs written from 5' to 3' written in Rebase convention. If the point of cleavage has been determined, the precise site is marked with ^. Only letters in the IUPAC code are accepted. For nucleases that cleave away from their recognition sequence, the cleavage sites are indicated in parentheses. See details for more information. |
| cutSites | Matrix with 2 rows (+ and - strand, respectively) specifying the cleavage coordinates relative to the first nucleotide of the motif sequence. Each column corresponds to a motif specified in the motifs slot. |
| weights | Optional numeric vector specifying relative weights for the recognition motifs to specify cleavage probabilities. |
| metadata | Optional list providing global metadata information. |
| expand | Should sequences be expanded to only contain ATCG nucleotides? FALSE by default. |
| primary | Should only the motif with the highest weight be returned? FALSE by default. Only relevant if weights are stored in the Nuclease object. |
| strand | Strand to allow reverse complementation of the motif. "+" by default. |
| as.character | Should the motif sequences be returned as a character vector? FALSE by default. |
| combine | Should only unique values be considered? TRUE by default. |
| middle | For staggered cuts, should the middle point between the cut on the forward strand and the cut on the reverse strand be considered as the cut site? FALSE by default. |

## Value

A Nuclease object

## Functions

- Nuclease(): Create a Nuclease object

## Slots

nucleaseName Name of the nuclease.

targetType Character string indicationg target type ("DNA" or "RNA").

`motifs` DNAStringSet of recognition sequence motifs written from 5' to 3'.

`cutSites` Matrix with 2 rows (+ and - strand, respectively) specifying the cleavage coordinates relative to the first nucleotide of the motif sequence. Each column corresponds to a motif specified in the `motifs` slot.

`weights` Optional numeric vector specifying relative weights for the motifs corresponding to cleavage probabilities.

`metadata` Optional string providing a description of the nuclease.

## Constructors

Use the constructor `link{Nuclease}` to create a Nuclease object.

## Accessors

`nucleaseName`**:** To get the name of the nuclease.

`targetType`**:** To get the target type ("DNA" or "RNA").

`metadata`**:** To get the metadata list of the nuclease.

`motifs`**:** To get the recognition mofif nucleotide sequences.

`weights`**:** To get nuclease weights.

`cutSites`**:** To get nuclease cut sites.

## See Also

See the CrisprNuclease for CRISPR-specific nucleases.

## Examples

```
EcoRI <- Nuclease("EcoRI",
                  motifs=c("G^AATTC"),
                  metadata=list(description="EcoRI restriction enzyme"))
```

---

# nickaseName *An S4 class to represent a nickase*

---

## Description

An S4 class to represent a nickase

## Usage

```
nickaseName(object)

nickaseName(object) <- value

nickingStrand(object)

nickingStrand(object) <- value

Nickase(
  nickaseName,
  nickingStrand = c("original", "opposite"),
  motifs = NULL,
  cutSites = NULL,
  weights = rep(1, length(motifs)),
  metadata = list()
)

## S4 method for signature 'Nickase'
show(object)

## S4 method for signature 'Nickase'
nickaseName(object)

## S4 replacement method for signature 'Nickase'
nickaseName(object) <- value

## S4 method for signature 'Nickase'
nickingStrand(object)

## S4 replacement method for signature 'Nickase'
nickingStrand(object) <- value

## S4 method for signature 'Nickase'
weights(object, expand = FALSE)

## S4 replacement method for signature 'Nickase'
weights(object) <- value

## S4 method for signature 'Nickase'
isCutting(object)

## S4 method for signature 'Nickase'
motifs(
  object,
  primary = FALSE,
  strand = c("+", "-"),
  expand = FALSE,
```

```
  as.character = FALSE
)

## S4 method for signature 'Nickase'
motifLength(object)

## S4 method for signature 'Nickase'
cutSites(object, combine = TRUE)
```

## Arguments

| | |
|---|---|
| object | Nickase object. |
| value | New value to pass to the setter functions. |
| nickaseName | Name of the nickase. |
| nickingStrand | String specifying with strand with respect to the motif sequence (5' to 3') is nicked. Must be either "original" (default) or "opposite". |
| motifs | Character vector of recognition sequence motifs written from 5' to 3' written in Rebase convention. If the point of cleavage has been determined, the precise site is marked with ^. Only letters in the IUPAC code are accepted. For nickases that cleave away from their recognition sequence, the cleavage sites are indicated in parentheses. See details for more information. |
| cutSites | Vector specifying the cleavage coordinates relative to the first nucleotide of the motif sequence. Each column corresponds to a motif specified in the motifs slot. |
| weights | Optional numeric vector specifying relative weights for the recognition motifs to specify cleavage probabilities. |
| metadata | Optional list providing global metadata information. |
| expand | Should sequences be expanded to only contain ATCG nucleotides? FALSE by default. |
| primary | Should only the motif with the highest weight be returned? FALSE by default. Only relevant if weights are stored in the Nickase object. |
| strand | Strand to allow reverse complementation of the motif. "+" by default. |
| as.character | Should the motif sequences be returned as a character vector? FALSE by default. |
| combine | Should only unique values be considered? TRUE by default. |

## Value

A Nickase object

## Functions

- Nickase(): Create a Nickase object

## Slots

`nickaseName` Name of the nickase

`motifs` DNAStringSet of recognition sequence motifs written from 5' to 3'.

`nickingStrand` String specifying with strand with respect to the motif sequence (5' to 3') is nicked. Must be either "original" (default) or "opposite".

`cutSites` Vector specifying the cleavage coordinates relative to the first nucleotide of the motif sequence. Each column corresponds to a motif specified in the `motifs` slot.

`weights` Optional numeric vector specifying relative weights for the motifs corresponding to cleavage probabilities.

`metadata` Optional string providing a description of the nickase.

## Constructors

Use the constructor `link{Nickase}` to create a Nickase object.

## Accessors

`nickaseName`**:** To get the name of the nickase.

`nickingStrand`**:** To get the nicking strand.

`metadata`**:** To get the metadata list of the nickase

`motifs`**:** To get the recognition mofif nucleotide sequences.

`weights`**:** To get nickase weights.

`cutSites`**:** To get nickase cut sites.

## See Also

See the CrisprNickase for CRISPR-specific nickases.

## Examples

```
Nb.BsmI <- Nickase("Nb.BsmI",
                   motifs=c("GAATG^C"),
                   nickingStrand="opposite",
                   metadata=list(description="Nb.BsmI nicking enzyme."))
```

## plotEditingWeights *Quick plot to visualize editing weights*

### Description

Quick plot to visualize editing weights from a BaseEditor object.

### Usage

```
plotEditingWeights(
  baseEditor,
  discardEmptyRows = TRUE,
  substitutions = NULL,
  ylim = c(0, 1),
  ...
)
```

### Arguments

| | |
|---|---|
| baseEditor | A BaseEditor object. |
| discardEmptyRows | Should rows that have all weight equal to 0 be discarded? TRUE by default. |
| substitutions | Character vector specifying substitutions to be plotted. If NULL (default), all substitutions are shown. |
| ylim | Vector of length 2 specifying the y plot limits. |
| ... | Additional arguments to be passed to plot |

### Value

Nothing. A plot is generated as a side effect.

### Examples

```
if (interactive()){
    data(BE4max, package="crisprBase")
    plotEditingWeights(BE4max)
}
```

## reexports *Objects exported from other packages*

### Description

These objects are imported from other packages. Follow the links below to see their documentation.

**S4Vectors** metadata, metadata<-

## restrictionEnzymes *List of Nuclease objects representing common restriction enzymes*

### Description

List of Nuclease objects representing common restriction enzymes from REBASE database.

### Usage

```
data(restrictionEnzymes, package="crisprBase")
```

### Format

List of Nuclease objects.

### Details

List of Nuclease objects representing common restriction enzymes from REBASE database.

## SaCas9 *SaCas9 CrisprNuclease object*

### Description

CrisprNuclease object for the wildtype Staphylococcus aureus Cas9 (SaCas9) nuclease.

### Usage

```
data(SaCas9, package="crisprBase")
```

### Format

CrisprNuclease object.

### Details

The AsCas9 nuclease recognizes NNGRRT PAM sequences. Spacer sequences must be located upstream of PAM sequences. Editing weights were obtained from doi:10.1038/nature14299.

spacerLength *An S4 class to represent a CRISPR nuclease.*

## Description

An S4 class to represent a CRISPR nuclease.

## Usage

```
spacerLength(object, ...)

targetLength(object, ...)

pamLength(object, ...)

spacerGap(object)

hasSpacerGap(object)

spacerGap(object) <- value

spacerLength(object) <- value

pamSide(object, ...)

pamSide(object) <- value

pams(object, ...)

pamIndices(object, ...)

spacerIndices(object, ...)

prototypeSequence(object, ...)

CrisprNuclease(
  nucleaseName,
  targetType = c("DNA", "RNA"),
  pams = NA_character_,
  weights = rep(1, length(pams)),
  metadata = list(),
  pam_side = NA_character_,
  spacer_gap = 0L,
  spacer_length = NA_integer_
)

## S4 method for signature 'CrisprNuclease'
```

```
show(object)

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

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

## S4 replacement method for signature 'CrisprNuclease'
spacerLength(object) <- value

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

## S4 replacement method for signature 'CrisprNuclease'
pamSide(object) <- value

## S4 method for signature 'CrisprNuclease'
spacerGap(object)

## S4 replacement method for signature 'CrisprNuclease'
spacerGap(object) <- value

## S4 method for signature 'CrisprNuclease'
hasSpacerGap(object)

## S4 method for signature 'CrisprNuclease'
targetLength(object)

## S4 method for signature 'CrisprNuclease'
pams(object, primary = TRUE, ignore_pam = FALSE, as.character = FALSE)

## S4 method for signature 'CrisprNuclease'
pamIndices(object)

## S4 method for signature 'CrisprNuclease'
spacerIndices(object)

## S4 method for signature 'CrisprNuclease'
prototypeSequence(object, primary = TRUE)
```

## Arguments

| | |
|---|---|
| object | CrisprNuclease object. |
| ... | Additional arguments for class-specific methods |
| value | For spacerLength<- and gapLength<-, must be a non-negative integer. For pamSide, must be either '5prime' or '3prime'. |
| nucleaseName | Name of the CRISPR nuclease. |

| | |
|---|---|
| targetType | String specifying target type ("DNA" or "RNA"). |
| pams | Character vector of PAM sequence motifs written from 5' to 3. If the point of cleavage has been determined, the precise site is marked with ^. Only letters in the IUPAC code are accepted. For nucleases that cleave away from their recognition sequence, the cleavage sites are indicated in parentheses. See details for more information. |
| weights | Optional numeric vector specifying relative weights of the PAM sequences to specify cleavage probabilities. |
| metadata | Optional list providing global metadata information. |
| pam_side | String specifying the side of the PAM sequence sequence with respect to the protospacer sequence. Must be either '3prime' (e.g. Cas9) or '5prime' (e.g. Cas12a) |
| spacer_gap | Integer specifying the length (in nucleotides) between the spacer sequence and the PAM sequence (e.g. 0 for Cas9 and Cas12a). |
| spacer_length | Integer specifying the length of the spacer sequence |
| primary | Should only the PAM sequence with the heighest weight be returned? If no cleavage weights are stored in the CrisprNuclease object, all sequences are returned. TRUE by default. |
| ignore_pam | Should all possible k-mer sequences for a given PAM length be returned, irrespetively of the PAM sequence motifs stored in the CrisprNuclease object? FALSE by default. |
| as.character | Should the PAM sequences be returned as a character vector? FALSE by default. |

## Value

A CrisprNuclease object

## Functions

- `CrisprNuclease()`: Create a CrisprNuclease object

## Slots

`pam_side` String specifying the side of the PAM sequence with respect to the protospacer sequence. Must be either '3prime' (e.g. SpCas9) or '5prime' (e.g. AsCas12a)

`spacer_length` Integer specifying the length of the spacer sequence

`spacer_gap` Integer specifying the length (in nucleotides) between the spacer sequence and the PAM sequence (e.g. 0 for SpCas9 and AsCas12a).

## Constructors

Use the constructor `link{CrisprNuclease}` to create a CrisprNuclease object.

### Accessors

`nucleaseName`: To get the name of the CRISPR nuclease.

`spacerLength`: To return the length of the spacer sequence.

`targetLength`: To return the length of the target sequence (protospacer + pam).

`pamLength`: To return the length of the PAM sequence.

`pamSide`: To return the side of the PAM sequence with respect to the spacer sequence.

`spacerGap`: To return the length of the gap between the PAM and spacer sequences.

`pams`: To return the list of PAM sequences.

### Setters

`spacerGap<-`: To change the length of the gap between the PAM and spacer sequences.

`pamSide<-`: To change the side of the PAM sequence with respect to the protospacer sequence.

`spacerLength<-`: To change the length of the spacer sequence.

### Utility functions for genomic arithmetics

`pamIndices`: To return the relative coordinates of the PAM sequence within the target sequence.

`spacerIndices`: To return the relatiive coordinates of the spacer sequence within the target sequence.

### Examples

```
SpCas9 <- CrisprNuclease("SpCas9",
                         pams=c("(3/3)NGG", "(3/3)NAG", "(3/3)NGA"),
                         weights=c(1, 0.2593, 0.0694),
                         metadata=list(description="Wildtype Streptococcus
                                       pyogenes Cas9 (SpCas9) nuclease"),
                         pam_side="3prime",
                         spacer_length=20)
```

---

## SpCas9 *SpCas9 CrisprNuclease object*

---

### Description

CrisprNuclease object for the wildtype Streptococcus pyogenes Cas9 (SpCas9) nuclease.

### Usage

```
data(SpCas9, package="crisprBase")
```

### Format

CrisprNuclease object.

## Details

The SpCas9 nuclease recognizes NGG PAM sequences. Spacer sequences must be located upstream of PAM sequences.

---

SpG *SpG CrisprNuclease object*

---

## Description

CrisprNuclease object for the engineered Streptococcus pyogenes Cas9 SpG nuclease.

## Usage

```
data(SpG, package="crisprBase")
```

## Format

CrisprNuclease object.

## Details

The SpG nuclease recognizes NGN PAM sequences. Spacer sequences must be located upstream of PAM sequences. Editing weights were obtained from doi:10.1038/s41467-022-28884-7.

# Index


∗ **datasets**
AsCas12a, 3
BE4max, 6
CasRx, 7
Csm, 10
enAsCas12a, 11
MAD7, 19
restrictionEnzymes, 28
SaCas9, 28
SpCas9, 32
SpG, 33
∗ **internal**
reexports, 27

annotateMismatches, 2
AsCas12a, 3

BaseEditor, *5*, *16*, *27*
BaseEditor *(*baseEditorName*)*, 4
BaseEditor-class *(*baseEditorName*)*, 4
baseEditorName, 4
baseEditorName,BaseEditor-method *(*baseEditorName*)*, 4
baseEditorName<- *(*baseEditorName*)*, 4
baseEditorName<-,BaseEditor-method *(*baseEditorName*)*, 4
BE4max, 6

CasRx, 7
CrisprNickase, *9*, *26*
CrisprNickase *(*CrisprNickase-class*)*, 7
CrisprNickase-class, 7
CrisprNuclease, *5*, *14*, *23*, *30*, *31*
CrisprNuclease *(*spacerLength*)*, 29
CrisprNuclease-class *(*spacerLength*)*, 29
Csm, 10
cutSites *(*motifs*)*, 20
cutSites,Nickase-method *(*nickaseName*)*, 23
cutSites,Nuclease-method *(*motifs*)*, 20

editingStrand *(*baseEditorName*)*, 4
editingStrand,BaseEditor-method *(*baseEditorName*)*, 4
editingStrand<- *(*baseEditorName*)*, 4
editingStrand<-,BaseEditor-method *(*baseEditorName*)*, 4
editingWeights *(*baseEditorName*)*, 4
editingWeights,BaseEditor-method *(*baseEditorName*)*, 4
enAsCas12a, 11
extractPamFromTarget, 11
extractProtospacerFromTarget, 12

getAvailableCrisprNucleases, 13
getCutSiteFromPamSite, 14
getCutSiteRanges, 14
getEditingSiteFromPamSite, 15
getPamRanges, 16
getProtospacerRanges, 17
getTargetRanges, 18

hasSpacerGap *(*spacerLength*)*, 29
hasSpacerGap,CrisprNickase-method *(*CrisprNickase-class*)*, 7
hasSpacerGap,CrisprNuclease-method *(*spacerLength*)*, 29

isCutting *(*motifs*)*, 20
isCutting,Nickase-method *(*nickaseName*)*, 23
isCutting,Nuclease-method *(*motifs*)*, 20
isDnase *(*motifs*)*, 20
isDnase,Nuclease-method *(*motifs*)*, 20
isRnase *(*motifs*)*, 20
isRnase,Nuclease-method *(*motifs*)*, 20

MAD7, 19
metadata, *27*
metadata *(*reexports*)*, 27
metadata<- *(*reexports*)*, 27

motifLength (motifs), 20
motifLength,Nickase-method (nickaseName), 23
motifLength,Nuclease-method (motifs), 20
motifs, 20
motifs,Nickase-method (nickaseName), 23
motifs,Nuclease-method (motifs), 20

Nickase, *25*
Nickase (nickaseName), 23
Nickase-class (nickaseName), 23
nickaseName, 23
nickaseName,Nickase-method (nickaseName), 23
nickaseName<- (nickaseName), 23
nickaseName<-,Nickase-method (nickaseName), 23
nickingStrand (nickaseName), 23
nickingStrand,Nickase-method (nickaseName), 23
nickingStrand<- (nickaseName), 23
nickingStrand<-,Nickase-method (nickaseName), 23
Nuclease, *22*
Nuclease (motifs), 20
Nuclease-class (motifs), 20
nucleaseName (motifs), 20
nucleaseName,Nuclease-method (motifs), 20
nucleaseName<- (motifs), 20
nucleaseName<-,Nuclease-method (motifs), 20

pamIndices (spacerLength), 29
pamIndices,CrisprNickase-method (CrisprNickase-class), 7
pamIndices,CrisprNuclease-method (spacerLength), 29
pamLength (spacerLength), 29
pamLength,CrisprNickase-method (CrisprNickase-class), 7
pamLength,CrisprNuclease-method (spacerLength), 29
pams (spacerLength), 29
pams,CrisprNickase-method (CrisprNickase-class), 7
pams,CrisprNuclease-method (spacerLength), 29
pamSide (spacerLength), 29
pamSide,CrisprNickase-method (CrisprNickase-class), 7
pamSide,CrisprNuclease-method (spacerLength), 29
pamSide<- (spacerLength), 29
pamSide<-,CrisprNickase-method (CrisprNickase-class), 7
pamSide<-,CrisprNuclease-method (spacerLength), 29
plotEditingWeights, 27
prototypeSequence (spacerLength), 29
prototypeSequence,CrisprNickase-method (CrisprNickase-class), 7
prototypeSequence,CrisprNuclease-method (spacerLength), 29

reexports, 27
restrictionEnzymes, 28

SaCas9, 28
setEditingWeights (baseEditorName), 4
show,BaseEditor-method (baseEditorName), 4
show,CrisprNickase-method (CrisprNickase-class), 7
show,CrisprNuclease-method (spacerLength), 29
show,Nickase-method (nickaseName), 23
show,Nuclease-method (motifs), 20
spacerGap (spacerLength), 29
spacerGap,CrisprNickase-method (CrisprNickase-class), 7
spacerGap,CrisprNuclease-method (spacerLength), 29
spacerGap<- (spacerLength), 29
spacerGap<-,CrisprNickase-method (CrisprNickase-class), 7
spacerGap<-,CrisprNuclease-method (spacerLength), 29
spacerIndices (spacerLength), 29
spacerIndices,CrisprNickase-method (CrisprNickase-class), 7
spacerIndices,CrisprNuclease-method (spacerLength), 29
spacerLength, 29
spacerLength,CrisprNickase-method (CrisprNickase-class), 7
spacerLength,CrisprNuclease-method (spacerLength), 29

spacerLength<- *(spacerLength)*, 29
spacerLength<-,CrisprNickase-method *(CrisprNickase-class)*, 7
spacerLength<-,CrisprNuclease-method *(spacerLength)*, 29
SpCas9, 32
SpG, 33

targetLength *(spacerLength)*, 29
targetLength,CrisprNickase-method *(CrisprNickase-class)*, 7
targetLength,CrisprNuclease-method *(spacerLength)*, 29
targetType *(motifs)*, 20
targetType,Nuclease-method *(motifs)*, 20
targetType<- *(motifs)*, 20
targetType<-,Nuclease-method *(motifs)*, 20

weights *(motifs)*, 20
weights,Nickase-method *(nickaseName)*, 23
weights,Nuclease-method *(motifs)*, 20
weights<- *(motifs)*, 20
weights<-,Nickase-method *(nickaseName)*, 23
weights<-,Nuclease-method *(motifs)*, 20