Package ‘XNAString’

February 20, 2025

Title Efficient Manipulation of Modified Oligonucleotide Sequences

Version 1.15.0

Date 31.05.2021

Description The XNAString package allows for description of base sequences and associated chemical modifications in a single object. XNAString is able to capture single stranded, as well as double stranded molecules. Chemical modifications are represented as independent strings associated with different features of the molecules (base sequence, sugar sequence, backbone sequence, modifications) and can be read or written to a HELM notation. It also enables secondary structure prediction using RNAfold from ViennaRNA. XNAString is designed to be efficient representation of nucleic-acid based therapeutics, therefore it stores information about target sequences and provides interface for matching and alignment functions from Biostrings and pwalign packages.

biocViews SequenceMatching, Alignment, Sequencing, Genetics

Depends R (>= 4.1)

Imports utils, Biostrings, pwalign, BSgenome, data.table, GenomicRanges, IRanges, methods, Rcpp, stringi, S4Vectors, future.apply, stringr, formattable, stats

Suggests BiocStyle, knitr, rmarkdown, markdown, testthat, BSgenome.Hsapiens.UCSC.hg38, pander

LinkingTo Rcpp

License GPL-2

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Collate 'RcppExports.R' 'utils.R' 'xnaStringSetClass.R' 'setterGetter.R' 'classUnion.R' 'xnaStringClass.R' 'XNAString2Helm.R' 'XNAStringFromHelm.R' 'alphabetFrequency.R' 'data.R' 'dictFromMimir.R' 'dinucleotideFrequency.R' 'globals.R' 'helm2String.R' 'matchPDict.R' 'matchPattern.R' 'pairwiseAlignment.R' 'predictDuplexStructure.R' 'predictMfeStructure.R' 'reverseComplement.R'

VignetteBuilder knitr

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

git_branch devel

git_last_commit 6cd7edb

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-02-19

Author Anna Górska [aut],
Marianna Plucinska [aut, cre],
Lykke Pedersen [aut],
Lukasz Kielpinski [aut],
Disa Tehler [aut],
Peter H. Hagedorn [aut]

Maintainer Marianna Plucinska <`marianna.plucinska@roche.com`>

Contents

`alphabetFrequency` *XNAAlphabetFrequency returns letters frequency for a given object in base, sugar or backbone slot*

Description

XNAAlphabetFrequency returns letters frequency for a given object in base, sugar or backbone slot

XNAAlphabetFrequency method returns alphabet frequency for a given object. It works for 3 slots: base, sugar and backbone. If matrix_nbr equals 1, alphabet frequency for the first elements in the slot is returned. Letters can be given as argument, otherwise unique letters in object's dictionary are in use.

Usage

```
XNAAlphabetFrequencyFun(
  obj,
  slot,
  letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
```

```
)

XNAAlphabetFrequency(
  obj,
  slot,
  letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE,
  ...
)

## S4 method for signature 'XNAString'
XNAAlphabetFrequency(
  obj,
  slot,
  letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
)

## S4 method for signature 'XNAStringSet'
XNAAlphabetFrequency(
  obj,
  slot,
  letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
)
```

Arguments

`obj`	XNAString or XNAStringSet class
`slot`	string (slot name: base, sugar or backbone)
`letters`	character (or character vector)
`matrix_nbr`	numeric (1 or 2, if 1 - first slot's element is use, if 2 - 2nd element in slot)
`as.prob`	logical - if TRUE frequency returned as probability of occurence
`base_only`	logical - if TRUE, frequency checked for 'A', 'C', 'G', 'T', other
`...`	optional arguments to generic function to support additional methods

Value

matrix (frequency matrix for a given slot)

Examples

```
xnastring_obj <- XNAString(
  name = "b",
  base = c("AACC", "GGEE"),
  sugar = c("FFOO", "OODD")
)
XNAAlphabetFrequency(obj = xnastring_obj,
                     slot = "base")
XNAAlphabetFrequency(obj = xnastring_obj,
                     slot = "base",
                     as.prob = TRUE)
XNAAlphabetFrequency(obj = xnastring_obj,
                     slot = "base",
                     base_only = TRUE)
XNAAlphabetFrequency(obj = xnastring_obj,
                     slot = "base",
                     letters = c("A", "C"))
XNAAlphabetFrequency(obj = xnastring_obj,
                     slot = "base",
                     matrix_nbr = 2)

xnastring_obj_2 <- XNAString(
  base = c("ATCG"),
  sugar = c("FODD"),
  backbone = c("SBB")
)
XNAStringSet_obj <- XNAStringSet(objects = list(
  xnastring_obj,
  xnastring_obj_2
))
XNAAlphabetFrequency(XNAStringSet_obj, "sugar")
```

`backbone` *Backbone setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
backbone(x, ...)

## S4 method for signature 'XNAString'
backbone(x)

## S4 method for signature 'XNAStringSet'
backbone(x, i = 1)
```

```
backbone(x, ...) <- value

## S4 replacement method for signature 'XNAString'
backbone(x) <- value

## S4 replacement method for signature 'XNAStringSet'
backbone(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
backbone(obj)
```

`base` *Base setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
base(x, ...)

## S4 method for signature 'XNAString'
base(x)

## S4 method for signature 'XNAStringSet'
base(x, i = 1)

base(x, ...) <- value

## S4 replacement method for signature 'XNAString'
base(x) <- value

## S4 replacement method for signature 'XNAStringSet'
base(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
base(obj)
```

`changeBase` *Translate base slot based on complementary bases dictionary. Base sequence in transformed using compl_target column.*

Description

Translate base slot based on complementary bases dictionary. Base sequence in transformed using compl_target column.

Usage

```
changeBase(compl_dict, bases)
```

Arguments

`compl_dict`	complementary bases dictionary
`bases`	string, one or two-elements vector

Value

string

`complementary_bases` *Default XNAString complementarity dictionary*

Description

A dataset containing default internal XNAString dictionary with base complemetary.

Usage

```
data(complementary_bases)
```

Format

A data.table with 6 rows and 3 variables:

base base symbol

target complementary base

compl_target complementary target

Source

RMR internal bioinformatics database (Mimir)

`compl_dictionary` *Compl_dictionary setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
compl_dictionary(x, ...)

## S4 method for signature 'XNAString'
compl_dictionary(x)

compl_dictionary(x, ...) <- value

## S4 replacement method for signature 'XNAString'
compl_dictionary(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
compl_dictionary(obj)
```

`concatDict` *Concatenate HELM-symbol custom dictionary with built-in HELM-symbol dictionary (xna_dictionary)*

Description

Concatenate HELM-symbol custom dictionary with built-in HELM-symbol dictionary (xna_dictionary)

Usage

```
concatDict(
  custom_dict,
  default_dict = xna_dictionary,
  helm_colname = "HELM",
  type_colname = "type",
  symbol_colname = "symbol"
)
```

Arguments

`custom_dict`	custom HELM-symbol dictionary
`default_dict`	built-in HELM-symbol dictionary (xna_dictionary)
`helm_colname`	helm column name in custom dictionary
`type_colname`	type column name in custom dictionary
`symbol_colname`	symbol column name in custom dictionary

Value

data.table

Examples

```
my_dict <- data.table::data.table(
  HELM = c("[[B]]"),
  type = c("base"),
  symbol = c("B")
)
concatDict(my_dict)
```

`conjugate3` *Conjugate3 setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
conjugate3(x, ...)

## S4 method for signature 'XNAString'
conjugate3(x)

## S4 method for signature 'XNAStringSet'
conjugate3(x, i = 1)

conjugate3(x, ...) <- value

## S4 replacement method for signature 'XNAString'
conjugate3(x) <- value

## S4 replacement method for signature 'XNAStringSet'
conjugate3(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
conjugate3(obj)
```

`conjugate5` *Conjugate5 setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
conjugate5(x, ...)

## S4 method for signature 'XNAString'
conjugate5(x)

## S4 method for signature 'XNAStringSet'
conjugate5(x, i = 1)

conjugate5(x, ...) <- value

## S4 replacement method for signature 'XNAString'
conjugate5(x) <- value

## S4 replacement method for signature 'XNAStringSet'
conjugate5(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
```

```
  dictionary = my_dic
)
conjugate5(obj)
```

`default_backbone` *Default_backbone setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
default_backbone(x, ...)

## S4 method for signature 'XNAString'
default_backbone(x)

## S4 method for signature 'XNAStringSet'
default_backbone(x)

default_backbone(x, ...) <- value

## S4 replacement method for signature 'XNAString'
default_backbone(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  default_sugar = 'F',
  default_backbone = 'X',
  dictionary = my_dic
)
default_backbone(obj)
```

`default_sugar` *Default_sugar setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
default_sugar(x, ...)

## S4 method for signature 'XNAString'
default_sugar(x)

## S4 method for signature 'XNAStringSet'
default_sugar(x)

default_sugar(x, ...) <- value

## S4 replacement method for signature 'XNAString'
default_sugar(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  default_sugar = 'F',
  default_backbone = 'X',
  dictionary = my_dic
)
default_sugar(obj)
```

`dictionary` *Dictionary setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
dictionary(x, ...)

## S4 method for signature 'XNAString'
dictionary(x)

dictionary(x, ...) <- value

## S4 replacement method for signature 'XNAString'
dictionary(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
dictionary(obj)
```

`dinucleotideFrequency` *XNADinucleotideFrequencyFun returns double letters frequency for a given object in base, sugar or backbone slot*

Description

XNADinucleotideFrequencyFun returns double letters frequency for a given object in base, sugar or backbone slot

XNADinucleotideFrequency method returns dinucleotide frequency for a given object. It works for 3 slots: base, sugar and backbone. If matrix_nbr equals 1, dinucleotide frequency for the first elements in the slot is returned. Double letters can be given as argument, otherwise unique double letters in object's dictionary are in use.

Usage

```
XNADinucleotideFrequencyFun(
  obj,
  slot,
  double_letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
)

XNADinucleotideFrequency(
  obj,
  slot,
  double_letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE,
  ...
)

## S4 method for signature 'XNAString'
XNADinucleotideFrequency(
  obj,
  slot,
  double_letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
)

## S4 method for signature 'XNAStringSet'
XNADinucleotideFrequency(
  obj,
  slot,
  double_letters = NA,
  matrix_nbr = 1,
  as.prob = FALSE,
  base_only = FALSE
)
```

Arguments

`obj`	XNAString or XNAStringSet class
`slot`	string (slot name: base, sugar or backbone)
`double_letters`	string (or string vector) - double letters
`matrix_nbr`	numeric (1 or 2, if 1 - first slot’s element is use, if 2 - 2nd element in slot)
`as.prob`	logical - if TRUE frequency returned as probability of occurence

`base_only`	logical - if TRUE, frequency checked for 'A', 'C', 'G', 'T', other
`...`	optional arguments to generic function to support additional methods

Value

matrix (frequency matrix for a given slot)

Examples

```
  my_dic <-
data.table::data.table(
 type = c(rep("base", 3), rep("sugar", 2), rep("backbone", 3)),
 symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
xnastring_obj <- XNAString(
 name = "b",
 base = c("GGEG"),
 sugar = c("FFOO"),
 dictionary = my_dic
)
XNAString::XNADinucleotideFrequency(
 obj = xnastring_obj,
 slot = "base",
 matrix_nbr = 1
)
```

`dt2Set` *Function which creates XNAstringSet object from table with base, sugar and backbone columns.*

Description

Function which creates XNAstringSet object from table with base, sugar and backbone columns.

Usage

```
dt2Set(
  table,
  col.base = "base",
  col.sugar = "sugar",
  col.backbone = "backbone",
  col.target = "target",
  default_sugar = NA,
  default_backbone = NA,
  compl_dict = complementary_bases
)
```

Arguments

`table`	data.table or data.frame (must incluse base, sugar and backbone columns)
`col.base`	character (name of base column)
`col.sugar`	character (name of sugar column)
`col.backbone`	character (name of backbone column)
`col.target`	character (name of target column)
`default_sugar`	character - only one letter. Will be replicated nchar(base) times
`default_backbone`	character - only one letter. Will be replicated nchar(base)-1 times
`compl_dict`	data.table with following columns: "base", "target". By default internal XNAString dictionary is used

Value

XNAStringSet object

Examples

```
dt <- data.table::data.table(
  base = c("TT", "GG"),
  sugar = c("FF", "FO"),
  backbone = c("S", "S")
)
dt2Set(dt)
```

`duplex_structure` *Duplex_structure setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
duplex_structure(x, ...)

## S4 method for signature 'XNAString'
duplex_structure(x)

## S4 method for signature 'XNAStringSet'
duplex_structure(x)

duplex_structure(x, ...) <- value

## S4 replacement method for signature 'XNAString'
duplex_structure(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
duplex_structure(obj)
```

`helm2String` *Translate RNA from HELM notation to multi-string notation*

Description

This function translates RNA molecules encoded in HELM notation into multi-string notation. It uses dictionary which links HELM code for base, sugar and backbone elements with symbols used in multi-string notation.

Usage

```
helm2String(helm, dictionary = xna_dictionary, remove_linker = TRUE)
```

Arguments

`helm`	string with HELM sequence, which contains one RNA polymer and optionally CHEM element
`dictionary`	data.table with following columns: "HELM", "type", "symbol". By default internal XNAString dictionary is used.
`remove_linker`	logical defines if linker should be clipped from RNA

Value

named list of strings with following elements: base, sugar, backbone, conjugate5, conjugate3

Author(s)

Marianna Plucinska

Examples

```
helm2String("RNA1{[dR](A)P.[dR](A)P.[dR](A)}$$$$V2.0")
```

`instanceOf` *Check on an object type*

Description

Check on an object type

Usage

```
instanceOf(object, type)
```

Arguments

`object`	an object of any class
`type`	class of an object

Value

logical information. TRUE if `object` class equals `type`

Examples

```
instanceOf(1, "numeric")
```

`listOflists2Dt` *Save list of lists as data.table*

Description

Save list of lists as data.table

Usage

```
listOflists2Dt(list_of_lists)
```

Arguments

`list_of_lists`	list of lists that will be saved as data.table.

Value

data.table

Examples

```
nested_list <- list(
  list(base = c("T"), sugar = c("G")),
  list(base = c("U"), sugar = c("G"))
)
listOflists2Dt(nested_list)
```

`mimir2XnaDict` *Reformat mimir table to XNA dictionary standards*

Description

Reformat mimir table to XNA dictionary standards

Usage

```
mimir2XnaDict(table, base.col, sugar.col, backbone.col)
```

Arguments

`table`	data.table or data.frame (must incluse "HELM", "TS_BASE_SEQ", "TS_SUGAR_SEQ" and "TS_BACKBONE_SEQ" columns)
`base.col`	character (base column name)
`sugar.col`	character (sugar column name)
`backbone.col`	character (backbone column name)

Value

data.table (written in the xna_dictionary format)

Examples

```
dt <- data.table::data.table(HELM = c("([PPG])", "[fR]", "[srP]"),
                TS_BASE_SEQ = c("F", NA, NA),
                TS_SUGAR_SEQ = c(NA, NA, 'F'),
                TS_BACKBONE_SEQ = c(NA, 'S', NA))
mimir2XnaDict(dt, 'TS_BASE_SEQ', 'TS_SUGAR_SEQ', 'TS_BACKBONE_SEQ')
```

`name` *Name setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
name(x, ...)

## S4 method for signature 'XNAString'
name(x)

## S4 method for signature 'XNAStringSet'
name(x, i = 1)

name(x, ...) <- value

## S4 replacement method for signature 'XNAString'
name(x) <- value

## S4 replacement method for signature 'XNAStringSet'
name(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
name(obj)
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj1 <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
obj2 <- XNAString(
  name = "b",
  base = c("GGE", "EEE"),
  sugar = c("FFO", "OOO"),
  dictionary = my_dic
)
XNAStringSetObj <- XNAStringSet(objects = list(obj1, obj2))
name(XNAStringSetObj)
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
```

```
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
name(obj) <- "new_name"
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj1 <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
obj2 <- XNAString(
  name = "b",
  base = c("GGE", "EEE"),
  sugar = c("FFO", "OOO"),
  dictionary = my_dic
)
XNAStringSetObj <- XNAStringSet(objects = list(obj1, obj2))
name(XNAStringSetObj, 1) <- c("new1", "new2")
```

`objects` *Objects getter method for XNAStringSet class*

Description

Getter methods enable extraction of single slots from XNAStringSet objects. E.g. objects method extracts objects slot from XNAStringSet object. It is a list of XNAString objects.

Usage

```
objects(x, ...)

## S4 method for signature 'XNAStringSet'
objects(x)
```

Arguments

`x`	XNAStringSet object
`...`	optional arguments to generic function to support additional methods

Value

list of XNAString objects

Examples

```
my_dic <- data.table::data.table(type = c(rep('base',3),
                                          rep('sugar',2),
                                          rep('backbone',3)),
                                 symbol = c('G', 'E', 'A', 'F',
                                            'O', 'S', 'B', 'X'))
obj2 <- XNAString(name = 'b',
                  base = 'GGE',
                  sugar = 'FFO',
                  dictionary = my_dic)
obj3 <- XNAString(name = 'b',
                  base = c('GGE','EEE'),
                  sugar = c('FFO', 'OOO'),
                  dictionary = my_dic)
XNAStringSetObj <- XNAStringSet(objects=list(obj2, obj3))
objects(XNAStringSetObj)
```

`parseRnaHelmComponent` *Parse monomers from HELM to multi-string notation*

Description

Parse monomers from HELM to multi-string notation

Usage

```
parseRnaHelmComponent(rna_component, dictionary = xna_dictionary)
```

Arguments

`rna_component`	list of monomers building RNA
`dictionary`	data.table with following columns: "HELM", "type", "symbol". By default internal XNAString dictionary is used.

Value

list of three strings: base, sugar, backbone

Author(s)

Marianna Plucinska

Examples

```
parseRnaHelmComponent(c("[dR](A)P", "[dR](A)P", "[dR](A)"))
```

`predictDuplexStructure` *Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.*

Description

This function is a wrapper for RNAcofold from ViennaRNA package.

Usage

```
predictDuplexStructureFun(obj)

predictDuplexStructure(obj, ...)

## S4 method for signature 'XNAString'
predictDuplexStructure(obj)
```

Arguments

`obj`	XNAString object
`...`	optional arguments to generic function to support additional methods

Value

list (structure and mfe)

Examples

```
obj1 <- XNAString(
  base = "ATCG",
  sugar = "FODD",
  conjugate3 = "TAG"
)
predictDuplexStructure(obj1)
```

`predictMfeStructure` *Prediction of MFE structure with ViennaRNA package*

Description

This function is a wrapper for RNAfold from ViennaRNA package.

Usage

```
predictMfeStructureFun(obj)

predictMfeStructure(obj, ...)

## S4 method for signature 'XNAString'
predictMfeStructure(obj)
```

Arguments

`obj`	XNAString object
`...`	optional arguments to generic function to support additional methods

Value

character, secondary structure in dot-bracket notation

Examples

```
obj1 <- XNAString(
  base = "ATCG",
  sugar = "FODD",
  conjugate3 = "TAG"
)
predictMfeStructure(obj1)
```

`reverseComplementFun` *Reverse complement sequence based on dictionary*

Description

Reverse complement sequence based on dictionary

Usage

```
reverseComplementFun(obj)
```

Arguments

`obj`	XNAString object

Value

string with reverse complement sequence

`secondary_structure` *Secondary_structure setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
secondary_structure(x, ...)

## S4 method for signature 'XNAString'
secondary_structure(x)

## S4 method for signature 'XNAStringSet'
secondary_structure(x)

secondary_structure(x, ...) <- value

## S4 replacement method for signature 'XNAString'
secondary_structure(x) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
secondary_structure(obj)
```

`seqAlphabetFrequency` *Create set of functions and methods to calculate alphabet frequency in base, sugar and backbone slots*

Description

Create set of functions and methods to calculate alphabet frequency in base, sugar and backbone slots

Usage

```
seqAlphabetFrequency(unique_letters, seq, as.prob)
```

Arguments

`unique_letters`	string (or character) - these letters pose column names
`seq`	string (or character) - frequency is calculated for this string
`as.prob`	logical - if TRUE frequency returned as probability of occurrence

Value

numeric - named numeric vector

Examples

```
seqAlphabetFrequency(c("A", "B", "C"), c("AABA"), as.prob = FALSE)
```

---

seqDinucleotideFrequency

Create set of functions and methods to calculate dinucleotide frequency in base, sugar and backbone slots

---

Description

Create set of functions and methods to calculate dinucleotide frequency in base, sugar and backbone slots

Usage

```
seqDinucleotideFrequency(unique_sets, seq, as.prob)
```

Arguments

`unique_sets`	string vector of double letters -these letters pose column names
`seq`	string (or character) - frequency is calculated for this string
`as.prob`	logical - if TRUE frequency returned as probability of occurrence

Value

numeric - named numeric vector

Examples

```
seqDinucleotideFrequency(c("AB", "BA", "CD"),
                                   "ABABAB",
                                    as.prob = FALSE)
seqDinucleotideFrequency(c("GC", "CG", "CC"),
                         "GCCG",
                         as.prob = FALSE)
```

---

seqVectorAlphabetFrequency

seqVectorAlphabetFrequency function calculates frequency for strings vector

---

Description

seqVectorAlphabetFrequency function calculates frequency for strings vector

Usage

```
seqVectorAlphabetFrequency(unique_letters, seq_vec, as.prob)
```

Arguments

`unique_letters`	string (or character) - these letters pose column names
`seq_vec`	vector of strings (or characters) - frequency will be calculated for this vector
`as.prob`	logical - if TRUE frequency returned as probability of occurence

Value

matrix - each row denotes frequency for a specific string of vector

Examples

```
seqVectorAlphabetFrequency(c("A", "B", "C"),
  c("AABA", "BBBCCC"),
  as.prob = FALSE
)
```

`seqVectorDinucleotideFrequency`

seqVectorDinucleotideFrequency function calculates frequency for strings vector

Description

seqVectorDinucleotideFrequency function calculates frequency for strings vector

Usage

```
seqVectorDinucleotideFrequency(unique_sets, seq_vec, as.prob)
```

Arguments

`unique_sets`	string vector of double letters -these letters pose column names
`seq_vec`	vector of strings (or characters) - frequency will be calculated for this vector
`as.prob`	logical - if TRUE frequency returned as probability of occurence

Value

matrix - each row denotes frequency for a specific string of vector

Examples

```
seqVectorDinucleotideFrequency(c("AB", "BA", "CD"),
                                          c("ABABAB", "ABABCD"),
                                          as.prob = FALSE)
```

`set2Dt` *set2Dt function - changes XNAStringSet object to data.table*

Description

set2Dt function - changes XNAStringSet object to data.table

Usage

```
set2Dt(obj, slots)
```

Arguments

`obj`	XNAStringSet object
`slots`	slots that are saved as column names (possibilities: "name", "base", "sugar", "backbone", "target", "conjugate5", "conjugate3" and "dictionary")

Value

data.table

Examples

```
my_dic <- data.table::data.table(type = c(rep('base',3),
                                         rep('sugar',2),
                                         rep('backbone',3)),
                                symbol = c('G', 'E', 'A', 'F',
                                           'O', 'S', 'B', 'X'))
obj2 <- XNAString(name = 'b',
                  base = 'GGE',
                  sugar = 'FFO',
                  dictionary = my_dic)
obj3 <- XNAString(name = 'b',
                  base = c('GGE','EEE'),
                  sugar = c('FFO', 'OOO'),
                  dictionary = my_dic)
XNAStringSetObj <- XNAStringSet(objects=list(obj2, obj3))
set2Dt(XNAStringSetObj, c('base', 'sugar'))

my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj2 <- XNAString(
  name = "b",
```

```
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
obj3 <- XNAString(
  name = "b",
  base = c("GGE", "EEE"),
  sugar = c("FFO", "OOO"),
  dictionary = my_dic
)
XNAStringSetObj <- XNAStringSet(objects = list(obj2, obj3))
set2Dt(XNAStringSetObj, c("base", "sugar"))
```

`set2List` *Define method to save XNAStringSet object as a list of XNAString objects*

Description

Define method to save XNAStringSet object as a list of XNAString objects

Usage

```
set2List(obj)

## S4 method for signature 'XNAStringSet'
set2List(obj)
```

Arguments

`obj`	XNAStringSet object

Value

list of XNAString objects

Examples

```
my_dic <- data.table::data.table(type = c(rep('base',3),
                                          rep('sugar',2),
                                          rep('backbone',3)),
                                 symbol = c('G', 'E', 'A', 'F',
                                            'O', 'S', 'B', 'X'))
obj2 <- XNAString(name = 'b',
                  base = 'GGE',
                  sugar = 'FFO',
                  dictionary = my_dic)
obj3 <- XNAString(name = 'b',
                  base = c('GGE','EEE'),
```

```
                  sugar = c('FFO', 'OOO'),
                  dictionary = my_dic)
XNAStringSetObj <- XNAStringSet(objects=list(obj2, obj3))
set2List(XNAStringSetObj)
```

---

siRNA_HELM — *siRNA_HELM function takes XNAString object and returns pairing information for base slot. Works only for double stranded molecules.*

Description

siRNA_HELM function takes XNAString object and returns pairing information for base slot. Works only for double stranded molecules.

Usage

```
siRNA_HELM(xnastring_obj)
```

Arguments

`xnastring_obj`	XNAString object

Value

string

Examples

```
obj1 <- XNAString(
  base = c("CCCCUGCCGUGGUUCAUAA", "UUAUGAACCACGGCAGGGGCG"),
  sugar = c("OOFOFOFOFOFOFOFOFOF", "FFOFOFOFOFOFOFOFOFOFO"),
  backbone = c("0000000000000000000", "000000000000000000000"),
  conjugate3 = c("")
)

siRNA_HELM(obj1)
```

---

sugar — *Sugar setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
sugar(x, ...)

## S4 method for signature 'XNAString'
sugar(x)

## S4 method for signature 'XNAStringSet'
sugar(x, i = 1)

sugar(x, ...) <- value

## S4 replacement method for signature 'XNAString'
sugar(x) <- value

## S4 replacement method for signature 'XNAStringSet'
sugar(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
```

```
  dictionary = my_dic
)
sugar(obj)
```

`target` *Target setter/getter method*

Description

Getter methods enable extraction of single slots from XNAString and XNAStringSet objects. E.g. name method extracts name slot from XNAString/XNAStringSet object.

Usage

```
target(x, ...)

## S4 method for signature 'XNAString'
target(x)

## S4 method for signature 'XNAStringSet'
target(x, i = 1)

target(x, ...) <- value

## S4 replacement method for signature 'XNAString'
target(x) <- value

## S4 replacement method for signature 'XNAStringSet'
target(x, i = 1) <- value
```

Arguments

`x`	XNAString/XNAStringSet object
`...`	optional arguments to generic function to support additional methods
`i`	numeric - possibilities: 1 or 2. If 1 - 1st slots elements printed out, 2nd otherwise. In case the second element is not in the object, empty char created. This parameter is only available for XNAStringSet objects.
`value`	character vector applied only for setter method

Details

Setter methods enable overwriting single slots from XNAString and XNAStringSet objects. E.g. name<- method overwrites existing name slot

Value

vector in getter method, XNAStringSet object (with replaced name slot) in setter method

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
target(obj)
```

`typedListCheck` *Check if all objects are of XNAString class and dictionaries are the same*

Description

Check if all objects are of XNAString class and dictionaries are the same

Usage

```
typedListCheck(object)
```

Arguments

`object`	an object of any class. An object must contain 'objects' (list type) slot

Value

logical information. Checks the whole list of objects, TRUE if class of all objects equals 'XNAString' and their dictionaries are the same.

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj2 <- XNAString(
```

```
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
obj3 <- XNAString(
  name = "b",
  base = c("GGE", "EEE"),
  sugar = c("FFO", "OOO"),
  dictionary = my_dic
)
XNAStringSetObj <- XNAStringSet(objects = list(obj2, obj3))
typedListCheck(XNAStringSetObj)
```

`uniqueChars` *Utility functions useful when programming and developing XNAString class*

Description

Utility functions useful when programming and developing XNAString class

Usage

```
uniqueChars(x)
```

Arguments

`x`	A string vector

Value

A list of vectors with unique characters found in `x` string

Examples

```
uniqueChars("TRGFFTR")
uniqueChars(c("TRGFFTR", "AATGRC"))
```

`XNAMatchPattern` *Finds pattern in reference sequence*

Description

This is function finding all the occurrences of a given pattern (typically short) in a (typically long) reference sequence

Usage

```
XNAMatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto"
)

## S4 method for signature 'XNAString,character'
XNAMatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto"
)

## S4 method for signature 'XNAString,XString'
XNAMatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto"
)
```

Arguments

`pattern`	XNAString object with non-empty target slot

`subject`	string or DNAString object
`target.number`	numeric - if target is a multi-element vector, then specify which element in use. 1 is the default
`max.mismatch`	The maximum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`min.mismatch`	The minimum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`with.indels`	If TRUE then indels are allowed. In that case, min.mismatch must be 0 and max.mismatch is interpreted as the maximum "edit distance" allowed between the pattern and a match. Note that in order to avoid pollution by redundant matches, only the "best local matches" are returned. Roughly speaking, a "best local match" is a match that is locally both the closest (to the pattern P) and the shortest.
`fixed`	If TRUE (the default), an IUPAC ambiguity code in the pattern can only match the same code in the subject, and vice versa. If FALSE, an IUPAC ambiguity code in the pattern can match any letter in the subject that is associated with the code, and vice versa.
`algorithm`	One of the following: "auto", "naive-exact", "naive-inexact", "boyer-moore", "shift-or" or "indels".

Value

an XStringViews object for `matchPattern`.

Examples

```
s1 <-
XNAString::XNAString(
 base = Biostrings::DNAString("GCGGAGAGAGCACAGATACA"),
 sugar = "FODDDDDDDDDDDDDDDDDD",
 target = Biostrings::DNAStringSet("GGCGGAGAGAGCACAGATACA")
)
XNAString::XNAMatchPattern(
s1,
"GGCGGAGAGAGCACAGATACAGGCGGAGAGAGCACAGATACA"
)
```

`XNAMatchPDict` *Find set of patterns in reference sequence*

Description

This is function finding all the occurrences of a given set of patterns (typically short) in a (typically long) reference sequence

Usage

```
XNAMatchPDict(
  pdict,
  subject,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto",
  verbose = FALSE
)

## S4 method for signature 'XNAString,character'
XNAMatchPDict(
  pdict,
  subject,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto",
  verbose = FALSE
)

## S4 method for signature 'XNAString,XString'
XNAMatchPDict(
  pdict,
  subject,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto",
  verbose = FALSE
)
```

Arguments

`pdict`	XNAString object, target slot taken as pdict object from Biostrings
`subject`	string containing sequence
`max.mismatch`	The maximum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`min.mismatch`	The minimum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`with.indels`	If TRUE then indels are allowed. In that case, min.mismatch must be 0 and max.mismatch is interpreted as the maximum "edit distance" allowed between the pattern and a match. Note that in order to avoid pollution by redundant

	matches, only the "best local matches" are returned. Roughly speaking, a "best local match" is a match that is locally both the closest (to the pattern P) and the shortest.
`fixed`	If TRUE (the default), an IUPAC ambiguity code in the pattern can only match the same code in the subject, and vice versa. If FALSE, an IUPAC ambiguity code in the pattern can match any letter in the subject that is associated with the code, and vice versa.
`algorithm`	One of the following: "auto", "naive-exact", "naive-inexact", "boyer-moore", "shift-or" or "indels".
`verbose`	TRUE or FALSE.

Value

an MIndex object of length `M`, and `countPDict` an integer vector of length `M`.

Examples

```
s2 <-
XNAString::XNAString(
 base = "GCGGAGAGAGCACAGATACA",
 sugar = "FODDDDDDDDDDDDDDDDDD",
 target = Biostrings::DNAStringSet(c(
   "GGCGGAGAGAGCACAGATACA", "GGCGGAGAGAGCACAGATACA"
 ))
)
o <- XNAString::XNAMatchPDict(
 s2,
 "GGCGGAGAGAGCACAGATACAGGGGCGGAGAGAGCACAGATACACGGAGAGAGCACAGATACA"
)
```

`xnaObj2Dt` *xnaObj2Dt function - changes XNAString object to data.table*

Description

xnaObj2Dt function - changes XNAString object to data.table

Usage

```
xnaObj2Dt(obj, slots)
```

Arguments

`obj`	XNAString object
`slots`	slots that are saved as column names (possibilities: "name", "base", "sugar", "backbone", "target", "conjugate5", "conjugate3" and "dictionary")

Value

data.table

`XNAPairwiseAlignment` *Pairwise alignment methods for XNAString object*

Description

This function performs pairwise alignment for sequences stored in target slot of XNAString object with subject

Usage

```
XNAPairwiseAlignment(pattern, subject, ...)

## S4 method for signature 'XNAString,character'
XNAPairwiseAlignment(
  pattern,
  subject,
  type = "global",
  substitutionMatrix = NULL,
  fuzzyMatrix = NULL,
  gapOpening = 10,
  gapExtension = 4,
  scoreOnly = FALSE
)
```

Arguments

`pattern`	XNAString object, pattern taken from target slot.
`subject`	a character vector of length 1, an XString, or an XStringSet object of length 1.
`...`	optional arguments to generic function to support additional methods
`type`	type of alignment. One of "global", "local", "overlap", "global-local", and "local-global" where "global" = align whole strings with end gap penalties, "local" = align string fragments, "overlap" = align whole strings without end gap penalties, "global-local" = align whole strings in pattern with consecutive subsequence of subject, "local-global" = align consecutive subsequence of pattern with whole strings in subject.
`substitutionMatrix`	substitution matrix representing the fixed substitution scores for an alignment. It cannot be used in conjunction with patternQuality and subjectQuality arguments.
`fuzzyMatrix`	fuzzy match matrix for quality-based alignments. It takes values between 0 and 1; where 0 is an unambiguous mismatch, 1 is an unambiguous match, and values in between represent a fraction of "matchiness".

`gapOpening`	the cost for opening a gap in the alignment.
`gapExtension`	the incremental cost incurred along the length of the gap in the alignment.
`scoreOnly`	logical to denote whether or not to return just the scores of the optimal pairwise alignment.

Value

an instance of class `PairwiseAlignments`

Examples

```
mat <-
 pwalign::nucleotideSubstitutionMatrix(
   match = 1,
   mismatch = -3,
   baseOnly = TRUE
)
s1 <-
  XNAString::XNAString(
    base = "GCGGAGAGAGCACAGATACA",
    sugar = "FODDDDDDDDDDDDDDDDDD",
    target = Biostrings::DNAStringSet("GGCGGAGAGAGCACAGATACA")
  )

  XNAString::XNAPairwiseAlignment(s1,
                                  "ACCCACACACACACACACACAC",
                                  "global",
                                  substitutionMatrix = mat
  )
```

`XNAReverseComplement` *Reverse complement sequence based on dictionary*

Description

Reverse complement sequence based on dictionary

Usage

```
XNAReverseComplement(obj, ...)

## S4 method for signature 'XNAString'
XNAReverseComplement(obj)
```

Arguments

`obj`	XNAString object
`...`	optional arguments to generic function to support additional methods

Value

string with reverse complement sequence

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
XNAReverseComplement(obj)
```

`XNAString-class` *Development of XNAString class aims at enabling efficient manipulation of modified oligonucleotide sequences. The class consists of the following slots: name, base, sugar, backbone, target, conjugate5, conjugate3, secondary_structure, duplex_structure, dictionary (HELM-string dictionary), compl_dictionary.*

Description

The package inherits some of the functionalities from Biostrings package. In contrary to Biostrings sequences, XNAString classes allow for description of base sequence, sugar and backbone in a single object. XNAString is able to capture single stranded oligonucleotides, siRNAs, PNAs, shRNAs, gRNAs and synthetic mRNAs, and enable users to apply sequence-manipulating Bioconductor packages to their analysis. XNAString can read and write a HELM notation, compute alphabet frequency, align and match targets.

Usage

```
XNAString(
  name,
  base,
  sugar,
  backbone,
  target,
  conjugate5,
  conjugate3,
  secondary_structure,
```

```
  duplex_structure,
  dictionary,
  compl_dictionary,
  default_sugar,
  default_backbone
)

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

## S4 method for signature 'XNAString'
initialize(
  .Object,
  name,
  base,
  sugar,
  backbone,
  target,
  conjugate5,
  conjugate3,
  secondary_structure,
  duplex_structure,
  dictionary,
  compl_dictionary,
  default_sugar,
  default_backbone
)

seqtype(x)

## S4 method for signature 'XNAString'
seqtype(x)
```

Arguments

`name`	string (or character)
`base`	string (or character), RNAString, RNAStringSet, DNAString or DNAStringSet
`sugar`	string (or character)
`backbone`	string (or character)
`target`	DNAStringSet, DNAString or character
`conjugate5`	string (or character)
`conjugate3`	string (or character)
`secondary_structure`	list
`duplex_structure`	list

`dictionary`	data.table with following columns: "HELM", "type", "symbol". By default internal XNAString dictionary is used.
`compl_dictionary`	data.table with following columns: "base", "target". By default internal XNAString dictionary is used
`default_sugar`	character, a single letter which will be replicated in sugar slot as default value
`default_backbone`	character, a single letter which will be replicated in backbone slot as default value
`object`	XNAString object
`.Object`	XNAString object
`x`	A single string specifying the type of sequences

Value

Object which consists of `name`, `base`, `sugar`, `backbone`, `target`, `conjugate5`, `conjugate3`, `secondary_structure`, `duplex_structure`, `dictionary`, `compl_dictionary`.

Author(s)

Anna Gorska

Examples

```
obj1 <- XNAString(
  base = "ATCG",
  sugar = "FODD",
  conjugate3 = "TAG"
)
obj2 <- XNAString(
  base = "ATCG",
  sugar = "FODD",
  backbone = "SBB"
)
str(obj2)
name(obj2) <- 'a'
base(obj2) <- 'ATTT'
sugar(obj2) <- 'LMFF'
backbone(obj2) <- 'BAB'
conjugate5(obj2) <- 'TFJSJG'
conjugate3(obj2) <- 'ARTSS'
my_dic <- data.table::data.table(type = c(rep('base',3),
                                          rep('sugar',2),
                                          rep('backbone',3)),
                                 symbol = c('G', 'E', 'A', 'F',
                                            'O', 'S', 'B', 'X'))
obj1 <- XNAString(base = 'AAE',
                  sugar = 'FFO',
                  backbone='SB',
                  dictionary = my_dic)
```

```
obj2 <- XNAString(base = c('EAA', 'AAAA'),
                  sugar = c('FFO', 'OOOO'),
                  name = c('a'),
                  conjugate5 = c('TTT'),
                  dictionary = my_dic)

my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj1 <- XNAString(
  base = "AAE",
  sugar = "FFO",
  backbone = "SB",
  dictionary = my_dic
)
obj2 <- XNAString(
  base = c("EAA", "AAAA"),
  sugar = c("FFO", "OOOO"),
  name = c("a"),
  conjugate5 = c("TTT"),
  dictionary = my_dic
)
```

`XNAString2XNAStringSet` *XNAString2XNAStringSet function - changes XNAString object to XNAStringSet*

Description

XNAString2XNAStringSet function - changes XNAString object to XNAStringSet

Usage

```
XNAString2XNAStringSet(XNAString_obj)
```

Arguments

`XNAString_obj`	XNAString object

Value

XNAStringSet object

`xnastringClassUnions` *setClassUnion definitions*

Description

setClassUnion definitions used in XNAString class. charOrDNAOrRNA consists of character, DNAString, RNAString, DNAStringSet, RNAStringSet. charOrDNA consists of character, DNAString, DNAStringSet

---

`xnastringElementsNumber` *Function which checks if XNAString object satisfies predefined slots length*

Description

Function which checks if XNAString object satisfies predefined slots length

Usage

```
xnastringElementsNumber(
  xnastring_obj,
  cond_name = "==1",
  cond_base,
  cond_sugar,
  cond_backbone,
  cond_target = ">0",
  cond_conj5 = "==1",
  cond_conj3 = "==1"
)
```

Arguments

`xnastring_obj`	XNAString object
`cond_name`	allowed name elements in object
`cond_base`	allowed base elements in object
`cond_sugar`	allowed sugar elements in object
`cond_backbone`	allowed backbone elements in object
`cond_target`	allowed target elements in object
`cond_conj5`	allowed conj5 elements in object
`cond_conj3`	allowed conj3 elements in object

Value

logical

Examples

```
obj <- XNAString(
  base = c("EAA", "AAA"),
  sugar = c("FFO", "OOO"),
  name = c("a"),
  conjugate5 = c("TTT")
)
xnastringElementsNumber(obj,
  cond_name = "==1",
  cond_base = "%in% c(1,2)",
  cond_sugar = "%in% c(1,2)",
  cond_backbone = "%in% c(1,2)",
  cond_target = ">0",
  cond_conj5 = "==1",
  cond_conj = "==1"
)
```

`XNAStringFromHelm` *Create XNAString object from HELM - user interface*

Description

Create XNAString object from HELM - user interface

Usage

```
XNAStringFromHelm(
  helm,
  name = NA_character_,
  dictionary = xna_dictionary,
  compl_dictionary = complementary_bases,
  remove_linker = TRUE
)
```

Arguments

`helm`	string (or strings vector) with HELM sequence, which contains one RNA polymer and optionally CHEM element
`name`	character (or character vector)
`dictionary`	data.table with following columns: "HELM", "type", "symbol". By default internal XNAString dictionary is used.
`compl_dictionary`	data.table with following columns: "base", "target". By default internal XNAString dictionary is used
`remove_linker`	logical defines if linker should be clipped from RNA

Value

XNAString object if single helm, XNAStringSet object otherwise

Author(s)

Marianna Plucinska

Examples

```
XNAStringFromHelm("RNA1{[dR](A)P.[dR](A)P.[dR](A)}$$$$V2.0")
XNAStringFromHelm("RNA1{[dR](A)P.[dR](A)P.[dR](A)}$$$$V2.0", 'name')
XNAStringFromHelm(c("RNA1{[dR](A)P.[dR](A)P.[dR](A)}$$$$V2.0",
                    "RNA1{[dR](T)P.[dR](T)P.[dR](A)}$$$$V2.0"),
                  c('name1', 'name2'))
```

`XNAStringSet-class` *Create class which consists of XNAString objects given as a list*

Description

Create class which consists of XNAString objects given as a list

Create XNAStringSet object

Define show method

Method to extract a row/rows (either by row index or by 'name' slot) XNAStringSet object is returned.

Method to extract a single row (either by row index or by 'name' slot) XNAString object is returned.

Usage

```
XNAStringSet(
  objects = NA,
  base = NA,
  sugar = NA,
  backbone = NA,
  target = NA,
  col.base = "base",
  col.sugar = "sugar",
  col.backbone = "backbone",
  col.target = "target",
  default_sugar = NA,
  default_backbone = NA,
  compl_dict = complementary_bases
)

## S4 method for signature 'XNAStringSet'
```

```
show(object)

## S4 method for signature 'XNAStringSet,ANY,ANY,ANY'
x[i]

## S4 method for signature 'XNAStringSet,ANY,ANY'
x[[i]]
```

Arguments

`objects`	list of XNAString objects
`base`	string (or character), RNAString, RNAStringSet, DNAString or DNAStringSet. In use only when objects argument is empty.
`sugar`	string (or character). In use only when objects argument is empty.
`backbone`	string (or character). In use only when objects argument is empty.
`target`	DNAStringSet, DNAString or character. In use only when objects argument is empty.
`col.base`	character (name of base column). In use only when objects argument is empty.
`col.sugar`	character (name of sugar column). In use only when objects argument is empty.
`col.backbone`	character (name of backbone column). In use only when o bjects argument is empty.
`col.target`	character (name of target column). In use only when objects argument is empty.
`default_sugar`	character - only one letter. Will be replicated nchar(base) times. In use only when objects argument is empty.
`default_backbone`	character - only one letter. Will be replicated nchar(base)-1 times. In use only when objects argument is empty.
`compl_dict`	data.table with following columns: "base", "target". By default internal XNAString dictionary is used. In use only when objects argument is empty.
`object`	XNAStringSet object
`x`	XNAStringSet object
`i`	numeric, integer, character, logical - filter needed for extraction method

Value

XNASTringSet object

Author(s)

Anna Gorska

Examples

```
my_dic <- data.table::data.table(
  type = c(
    rep("base", 3),
    rep("sugar", 2),
    rep("backbone", 3)
  ),
  symbol = c("G", "E", "A", "F", "O", "S", "B", "X")
)
obj1 <- XNAString(
  name = "a",
  base = "GGE",
  sugar = "FFO",
  backbone = "SB",
  dictionary = my_dic
)
obj2 <- XNAString(
  name = "b",
  base = "GGE",
  sugar = "FFO",
  dictionary = my_dic
)
obj3 <- XNAString(
  name = "b",
  base = c("GGE", "EEE"),
  sugar = c("FFO", "OOO"),
  dictionary = my_dic
)
XNAStringSetObj <- XNAStringSet(objects = list(obj1, obj2, obj3))
```

`XNAStringToHelm` *XNAStringToHelmFun function takes XNAString object and translates base, sugar and backbone to HELM notation*

Description

XNAStringToHelmFun function takes XNAString object and translates base, sugar and backbone to HELM notation

Usage

```
XNAStringToHelm(xnastring_obj, dictionary = xna_dictionary)
```

Arguments

`xnastring_obj`	XNAString object
`dictionary`	HELM-symbol dictionary

Value

string (HELM notation)

Examples

```
obj <- XNAString(
  base = "AAA",
  sugar = "DDD",
  backbone = "OO"
)
XNAStringToHelm(obj)
```

`XNAVmatchPattern` *This is function finding all the occurrences of a given pattern (typically short) in a (typically long) set of reference sequences.*

Description

This is function finding all the occurrences of a given pattern (typically short) in a (typically long) set of reference sequences.

Implementation of this method is based on vmatchPatterrm method from BSgenome

Usage

```
XNAVmatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto",
  exclude = "",
  maskList = logical(0),
  userMask = IRanges::IRangesList(),
  invertUserMask = FALSE
)

## S4 method for signature 'XNAString,character'
XNAVmatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
```

```
  fixed = TRUE,
  algorithm = "auto"
)

## S4 method for signature 'XNAString,XStringSet'
XNAVmatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto"
)

## S4 method for signature 'XNAString,BSgenome'
XNAVmatchPattern(
  pattern,
  subject,
  target.number = 1,
  max.mismatch = 0,
  min.mismatch = 0,
  with.indels = FALSE,
  fixed = TRUE,
  algorithm = "auto",
  exclude = "",
  maskList = logical(0),
  userMask = IRanges::IRangesList(),
  invertUserMask = FALSE
)
```

Arguments

`pattern`	XNAString object with non-empty target slot
`subject`	string, string vector or DNAString / DNAStringSet / chromosome from BSgenome object
`target.number`	numeric - if target is a multi-element vector, then specify which element in use. 1 is the default
`max.mismatch`	The maximum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`min.mismatch`	The minimum number of mismatching letters allowed. If non-zero, an algorithm that supports inexact matching is used.
`with.indels`	If TRUE then indels are allowed. In that case, min.mismatch must be 0 and max.mismatch is interpreted as the maximum "edit distance" allowed between the pattern and a match. Note that in order to avoid pollution by redundant matches, only the "best local matches" are returned. Roughly speaking, a "best

	local match" is a match that is locally both the closest (to the pattern P) and the shortest.
`fixed`	If TRUE (the default), an IUPAC ambiguity code in the pattern can only match the same code in the subject, and vice versa. If FALSE, an IUPAC ambiguity code in the pattern can match any letter in the subject that is associated with the code, and vice versa.
`algorithm`	One of the following: "auto", "naive-exact", "naive-inexact", "boyer-moore", "shift-or" or "indels".
`exclude`	A character vector with strings that will be used to filter out chromosomes whose names match these strings. Needed for BSParams object if subject is a chromosome object from BSgenome
`maskList`	A named logical vector of maskStates preferred when used with a BSGenome object. When using the bsapply function, the masks will be set to the states in this vector.
`userMask`	An IntegerRangesList, containing a mask to be applied to each chromosome.
`invertUserMask`	Whether the userMask should be inverted.

Value

An MIndex object for `vmatchPattern`.

Examples

```
s3 <-
XNAString::XNAString(
 base = "GCGGAGAGAGCACAGATACA",
 sugar = "FODDDDDDDDDDDDDDDDDD",
 target = Biostrings::DNAStringSet(
     c("AAAAGCTTTACAAAATCCAAGATC", "GGCGGAGAGAGCACAGATACA")
 )
)
chrom <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38$chr1
result <- XNAString::XNAMatchPattern(s3, chrom)
```

`xna_dictionary` *Default XNAString dictionary*

Description

A dataset containing default internal XNAString dictionary with HELM to string translation.

Usage

```
data(xna_dictionary)
```

Format

A data.table with 20 rows and 3 variables:

HELM	HELM sequence coding monomer
type	if element is coding base, sugar, backbone
symbol	single string translation of HELM

Source

RMR internal bioinformatics database (Mimir)

Index