

Package ‘MSA2dist’

April 6, 2026

Type Package

Title MSA2dist calculates pairwise distances between all sequences of a DNAStrngSet or a AAStringSet using a custom score matrix and conducts codon based analysis

Version 1.14.0

Description MSA2dist calculates pairwise distances between all sequences of a DNAStrngSet or a AAStringSet using a custom score matrix and conducts codon based analysis. It uses scoring matrices to be used in these pairwise distance calculations which can be adapted to any scoring for DNA or AA characters. E.g. by using literal distances MSA2dist calculates pairwise IUPAC distances. DNAStrngSet alignments can be analysed as codon alignments to look for synonymous and nonsynonymous substitutions (dN/dS) in a parallelised fashion using a variety of substitution models. Non-aligned coding sequences can be directly used to construct pairwise codon alignments (global/local) and calculate dN/dS without any external dependencies.

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData false

biocViews Alignment, Sequencing, Genetics, GO

Depends R (>= 4.4.0)

Imports Rcpp, Biostrings, GenomicRanges, IRanges, ape, doParallel, dplyr, foreach, methods, parallel, pwalgn, rlang, seqinr, stats, stringi, stringr, tibble, tidyr, utils

Suggests rmarkdown, knitr, devtools, testthat, ggplot2, BiocStyle

LinkingTo Rcpp, RcppThread

VignetteBuilder knitr

NeedsCompilation yes

SystemRequirements C++11

URL <https://gitlab.gwdg.de/mpievolbio-it/MSA2dist>,
<https://mpievolbio-it.pages.gwdg.de/MSA2dist/>

BugReports <https://gitlab.gwdg.de/mpievolbio-it/MSA2dist/issues>

RoxygenNote 7.3.1

git_url <https://git.bioconductor.org/packages/MSA2dist>

git_branch RELEASE_3_22
git_last_commit 010a9f3
git_last_commit_date 2025-10-29
Repository Bioconductor 3.22
Date/Publication 2026-04-05
Author Kristian K Ullrich [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4308-9626>)
Maintainer Kristian K Ullrich <ullrich@evolbio.mpg.de>

Contents

aa2selfscore	3
aabin2aastring	4
AAMatrix-data	4
aastring2aabin	5
aastring2aln	6
aastring2dist	7
addmask2string	8
addpop2string	9
addpos2string	10
addregion2string	11
aln2aastring	12
aln2dnastring	13
cds2aa	14
cds2codonaln	15
cdsstring2codonaln	16
codon2numberAMBIG	17
codon2numberTCAG	18
codonmat2pnps	18
codonmat2xy	20
compareCodons	21
dnabin2dnastring	22
dnastring2aln	23
dnastring2codonmat	24
dnastring2dist	25
dnastring2dnabin	26
dnastring2kaks	27
GENETIC_CODE_TCAG	29
getmask	29
getpos	30
globalDeletion	31
globalDeletionAA	32
granthamMatrix	32
hiv-data	33
indices2kaks	34
iupac-data	36
iupacMatrix	36
makePostalignedSeqs	37
pal2nal	38
popinteger	39

<i>aa2selfscore</i>	3
popnames	40
rpp_distSTRING	41
rpp_KaKs	41
rpp_pairwiseDeletionAA	42
rpp_pairwiseDeletionDNA	43
region	43
regionused	44
string2region	45
subString	46
upriidx	47
Index	48

<code>aa2selfscore</code>	<i>aa2selfscore</i>
---------------------------	---------------------

Description

This function return the selfscore from an AAStringSet.

Usage

```
aa2selfscore(aa, scorematrix = "BLOSUM62")
```

Arguments

<code>aa</code>	AAStringSet [mandatory]
<code>scorematrix</code>	score matrix to use [default: BLOSUM62]

Value

data.frame

Author(s)

Kristian K Ullrich

See Also

[XStringSet-class](#), [substitution_matrices](#)

Examples

```
data(woodmouse, package="ape")
#cds2aa(dnabin2dnastring(woodmouse), shorten=TRUE,
#genetic.code=Biostrings::getGeneticCode("2"))
woodmouse |> dnabin2dnastring() |> cds2aa(shorten=TRUE,
genetic.code=Biostrings::getGeneticCode("2")) |> aa2selfscore()
```

aabin2aastring *aabin2aastring*

Description

This function converts an ape AAbin into AAStrngSet.

Usage

```
aabin2aastring(aabin)
```

Arguments

aabin ape AAbin [mandatory]

Value

An object of class AAStrngSet

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#) [as.DNAbin.alignment](#) [AAStrngSet](#)

Examples

```
data(woodmouse, package="ape")
## convert into AAStrngSet
#aabin2aastring(ape::trans(woodmouse, 2))
ape::trans(woodmouse, 2) |> aabin2aastring()
```

AAMatrix-data *AAMatrix-data*

Description

getAAMatrix() from the alakazam package.

Usage

```
data(AAMatrix)
```

Format

an object of class matrix

Value

score matrix

References

Gupta N, Vander Heiden J, Uduman M, Gadala-Maria D, Yaari G, Kleinstein S (2015) Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. *Bioinformatics*. **31(20)**, 3356-3358.

Examples

```
data("AAMatrix", package="MSA2dist")
```

aastring2aabin	<i>aastring2aabin</i>
----------------	-----------------------

Description

This function converts an AAStringSet into an ape AAbin.

Usage

```
aastring2aabin(aa)
```

Arguments

aa AAStringSet [mandatory]

Value

An object of class AAbin

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#) [as.DNAbin.alignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
## convert into AAbin
#aastring2aabin(cds2aa(cds1.cds2.aln))
cds1.cds2.aln |> cds2aa() |> aastring2aabin()
```

aastring2aln	<i>aastring2aln</i>
--------------	---------------------

Description

This function converts a AAStringSet into an seqinr alignment.

Usage

```
aastring2aln(aa)
```

Arguments

aa AAStringSet [mandatory]

Value

An object of class alignment which is a list with the following components:
nb the number of aligned sequences
nam a vector of strings containing the names of the aligned sequences
seq a vector of strings containing the aligned sequences
com a vector of strings containing the commentaries for each sequence or NA if there are no comments

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
#aastring2aln(cds2aa(cds1.cds2.aln))
cds1.cds2.aln |> cds2aa() |> aastring2aln()
```

aastring2dist	<i>aastring2dist</i>
---------------	----------------------

Description

This function calculates pairwise distances for all combinations of a AAStringSet.

Usage

```
aastring2dist(  
  aa,  
  threads = 1,  
  symmetric = TRUE,  
  score = NULL,  
  mask = NULL,  
  region = NULL  
)
```

Arguments

aa	AAStringSet [mandatory]
threads	number of parallel threads [default: 1]
symmetric	symmetric score matrix [default: TRUE]
score	score matrix use a score matrix to calculate distances [mandatory]
mask	IRanges object indicating masked sites [default: NULL]
region	IRanges object indicating region to use for dist calculation (by default all sites are used) [default: NULL]

Value

A data.frame of pairwise distance values distSTRING, sites used sitesUsed and region used regionUsed

Author(s)

Kristian K Ullrich

See Also

[dnstring2dist](#)

Examples

```
## load example sequence data  
data("hiv", package="MSA2dist")  
#aastring2dist(cds2aa(hiv), score=granthamMatrix())  
hiv |> cds2aa() |> aastring2dist(score=granthamMatrix())  
## create mask  
mask1 <- IRanges::IRanges(start=c(11,41,71), end=c(20,50,80))  
## use mask  
hiv |> cds2aa() |> aastring2dist(score=granthamMatrix(), mask=mask1)
```

```
## use region
region1 <- IRanges::IRanges(start=c(1,75), end=c(45,85))
hiv |> cds2aa() |> aastring2dist(score=granthamMatrix(), region=region1)
## use mask and region
hiv |> cds2aa() |> aastring2dist(score=granthamMatrix(),
  mask=mask1, region=region1)
## use asymmetric score matrix
myscore <- granthamMatrix()
myscore[5, 6] <- 0
h <- hiv |> cds2aa() |> aastring2dist(score=myscore, symmetric=FALSE)
h$distSTRING[1:2, 1:2]
```

addmask2string

addmask2string

Description

This function adds mask information as an IRanges object, START and END information, to a DNASTringSet or an AAStringSet and puts them into the metadata information. This information can be used to restrict the distance calculation to specific regions of the DNASTringSet or the AAStringSet.

Usage

```
addmask2string(seq, mask = NULL, append = TRUE)
```

Arguments

seq	DNASTringSet or AAStringSet [mandatory]
mask	IRanges object [mandatory]
append	indicate if mask should be appended or overwritten [default: TRUE]

Value

An object of class DNASTringSet or AAStringSet

Author(s)

Kristian K Ullrich

See Also

[addregion2string](#), [addpop2string](#), [addpos2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create mask
mask1 <- IRanges::IRanges(start=c(1,41), end=c(20,50))
## add mask
iupac.aa <- iupac.aa |> addmask2string(mask=mask1)
```

```

#(iupac.aa |> slot("metadata"))$mask
iupac.aa |> getmask()
## append mask
mask2 <- IRanges::IRanges(start=c(21), end=c(30))
iupac.aa <- iupac.aa |> addmask2string(mask=mask2)
#(iupac.aa |> slot("metadata"))$mask
iupac.aa |> getmask()
## overwrite mask
iupac.aa <- iupac.aa |> addmask2string(mask=mask2, append=FALSE)
#(iupac.aa |> slot("metadata"))$mask
iupac.aa |> getmask()
## reduce by mask
#iupac.aa.region <- iupac.aa |> string2region(mask=
#   (iupac.aa |> slot("metadata"))$mask)
iupac.aa.region <- iupac.aa |> string2region(mask=
  getmask(iupac.aa))
#iupac.aa.region |> slot("metadata")
iupac.aa.region |> getmask()

```

addpop2string

addpop2string

Description

This function adds population information to a DNASTringSet or an AAStringSet and puts them into the metadata information.

__Note__: All unassigned sequences will be put into pop "unassigned"!

Do not use "unassigned" as a population name!

__Note__: Names in a population in the poplist must match sequence names!

__Note__: Duplicated assignments are allowed!

Usage

```
addpop2string(seq, poplist)
```

Arguments

seq	DNASTringSet or AAStringSet [mandatory]
poplist	named list of populations either as index or names per population (do not mix index and names in one population) [mandatory]

Value

An object of class DNASTringSet or AAStringSet

Author(s)

Kristian K Ullrich

See Also

[addmask2string](#), [addregion2string](#), [addpos2string](#)

Examples

```

## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create poplist
poplist <- list(FRA = grep("Mmd.FRA", names(iupac)),
  GER = grep("Mmd.GER", names(iupac)),
  IRA = grep("Mmd.IRA", names(iupac)),
  AFG = grep("Mmm.AFG", names(iupac)))
iupac.aa <- iupac.aa |> addpop2string(poplist)
#(iupac.aa |> slot("metadata"))$pop.integer
iupac.aa |> popinteger()
#(iupac.aa |> slot("metadata"))$pop.names
iupac.aa |> popnames()
## mxixing index and names
poplist <- list(FRA = names(iupac)[grep("Mmd.FRA", names(iupac))],
  GER = grep("Mmd.GER", names(iupac)),
  IRA = names(iupac)[grep("Mmd.IRA", names(iupac))],
  AFG = grep("Mmm.AFG", names(iupac)))
iupac.aa <- iupac.aa |> addpop2string(poplist)
iupac.aa |> popinteger()
iupac.aa |> popnames()
## leaving out some sequences which will be assigned as "unassigned"
poplist <- list(FRA = names(iupac)[grep("Mmd.FRA", names(iupac))],
  GER = grep("Mmd.GER", names(iupac)),
  IRA = names(iupac)[grep("Mmd.IRA", names(iupac))])
iupac.aa <- iupac.aa |> addpop2string(poplist)
iupac.aa |> popinteger()
iupac.aa |> popnames()

```

addpos2string

addpos2string

Description

This function adds GenomicRanges information, CHROM, START and END to a DNAStrngSet or an AAStringSet and puts them into the metadata information. This information can be used to find overlaps with a chromosome wide mask.

Usage

```
addpos2string(seq, chrom = NULL, start = NULL, end = NULL)
```

Arguments

seq	DNAStrngSet or AAStringSet [mandatory]
chrom	chromosome name [mandatory]
start	start position [mandatory]
end	end position [mandatory]

Value

An object of class DNAStrngSet or AAStringSet

Author(s)

Kristian K Ullrich

See Also[addmask2string](#), [addregion2string](#), [addpop2string](#)**Examples**

```
## load example sequence data
data(iupac, package="MSA2dist")
## add position
iupac <- iupac |> addpos2string(chrom="chr1", start=1, end=1000)
#(iupac |> slot("metadata"))$GRanges
iupac |> getpos()
```

addregion2string	<i>addregion2string</i>
------------------	-------------------------

Description

This function adds region information as an IRanges object, START and END information, to a DNASTringSet or an AAStringSet and puts them into the metadata information. This information can be used to restrict the distance calculation to specific regions of the DNASTringSet or the AAStringSet.

Usage

```
addregion2string(seq, region = NULL, append = TRUE)
```

Arguments

seq	DNASTringSet or AAStringSet [mandatory]
region	IRanges object [mandatory]
append	indicate if region should be appended or overwritten [default: TRUE]

Value

An object of class DNASTringSet or AAStringSet

Author(s)

Kristian K Ullrich

See Also[addmask2string](#), [addpop2string](#), [addpos2string](#)

Examples

```

## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create region
region1 <- IRanges::IRanges(start=c(1,41), end=c(20,50))
## add region
iupac.aa <- iupac.aa |> addregion2string(region=region1)
#(iupac.aa |> slot("metadata"))$region
iupac.aa |> region()
## append region
region2 <- IRanges::IRanges(start=c(21), end=c(30))
iupac.aa <- iupac.aa |> addregion2string(region=region2)
#(iupac.aa |> slot("metadata"))$region
iupac.aa |> region()
## overwrite region
iupac.aa <- iupac.aa |> addregion2string(region=region2, append=FALSE)
#(iupac.aa |> slot("metadata"))$region
iupac.aa |> region()
## reduce by region
#iupac.aa.region <- iupac.aa |> string2region(region=
#   (iupac.aa |> slot("metadata"))$region)
iupac.aa.region <- iupac.aa |> string2region(region=
  region(iupac.aa))
#iupac.aa.region |> slot("metadata")
iupac.aa.region |> region()

```

aln2aastring

aln2aastring

Description

This function converts a seqinr alignment into an AAStringSet.

Usage

```
aln2aastring(aln)
```

Arguments

aln seqinr alignment [mandatory]

Value

An object of class AAStringSet

Author(s)

Kristian K Ullrich

See Also

[as.alignment AAStringSet](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG--CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
#aastring2aln(cds2aa(cds1.cds2.aln))
cds1.cds2.aln |> cds2aa() |> aastring2aln() |> aln2aastring()
```

aln2dnastring

aln2dnastring

Description

This function converts a seqinr alignment into an DNASTringSet.

Usage

```
aln2dnastring(aln)
```

Arguments

aln seqinr alignment [mandatory]

Value

An object of class DNASTringSet

Author(s)

Kristian K Ullrich

See Also

[as.alignment DNASTringSet](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG--CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
## convert into alignment
#dnastring2aln(cds1.cds2.aln)
cds1.cds2.aln |> dnastring2aln()
## convert back into DNASTringSet
#aln2dnastring(dnastring2aln(cds1.cds2.aln))
cds1.cds2.aln |> dnastring2aln() |> aln2dnastring()
```

`cds2aa`*cds2aa*

Description

This function translates a DNASTringSet into an AAStringSet.

Usage

```
cds2aa(  
  cds,  
  shorten = FALSE,  
  frame = 1,  
  framelist = NULL,  
  genetic.code = NULL,  
  return.cds = FALSE  
)
```

Arguments

<code>cds</code>	DNASTringSet [mandatory]
<code>shorten</code>	shorten all sequences to multiple of three [default: FALSE]
<code>frame</code>	indicates the first base of a the first codon [default: 1]
<code>framelist</code>	supply vector of frames for each entry [default: NULL]
<code>genetic.code</code>	The genetic code to use for the translation of codons into Amino Acid letters [default: NULL]
<code>return.cds</code>	return shorten cds instead of aa [default: FALSE]

Value

AAStringSet

Author(s)

Kristian K Ullrich

See Also

[XStringSet-class](#), [translate](#)

Examples

```
## define two cds sequences  
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")  
cds2 <- Biostrings::DNASTring("ATG---CATTGC")  
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),  
  Biostrings::DNASTringSet(cds2))  
#cds2aa(cds1.cds2.aln)  
cds1.cds2.aln |> cds2aa()  
## alternative genetic code  
data(woodmouse, package="ape")
```

```
#cds2aa(dnabin2dnastring(woodmouse), shorten=TRUE)
woodmouse |> dnabin2dnastring() |> cds2aa(shorten=TRUE)
#cds2aa(dnabin2dnastring(woodmouse), shorten=TRUE,
#genetic.code=Biostrings::getGeneticCode("2"))
woodmouse |> dnabin2dnastring() |> cds2aa(shorten=TRUE,
genetic.code=Biostrings::getGeneticCode("2"))
woodmouse |> dnabin2dnastring() |> cds2aa(shorten=TRUE, return.cds=TRUE) |>
cds2aa(genetic.code=Biostrings::getGeneticCode("2"))
```

cds2codonaln

cds2codonaln

Description

This function takes two single sequence DNASTring's or two single sequence DNASTringSet's, converts them into aa, calculates a global alignment and converts this alignment back into a codon alignment.

Usage

```
cds2codonaln(
  cds1,
  cds2,
  type = "global",
  substitutionMatrix = "BLOSUM62",
  gapOpening = 10,
  gapExtension = 0.5,
  remove.gaps = FALSE,
  ...
)
```

Arguments

cds1	single sequence DNASTringSet or DNASTring [mandatory]
cds2	single sequence DNASTringSet or DNASTring [mandatory]
type	type of alignment (see pairwiseAlignment) [default: global]
substitutionMatrix	substitution matrix representing the fixed substitution scores for an alignment (see pairwiseAlignment) [default: BLOSUM62]
gapOpening	the cost for opening a gap in the alignment (see pairwiseAlignment) [default: 10]
gapExtension	the incremental cost incurred along the length of the gap in the alignment (see pairwiseAlignment) [default: 0.5]
remove.gaps	specify if gaps in the codon alignment should be removed [default: FALSE]
...	other cds2aa parameters

Value

codon alignment as DNASTringSet

Author(s)

Kristian K Ullrich

References

Pagès, H et al. (2014) Biostrings: Efficient manipulation of biological strings. *R package version, 2(0)*.

See Also

[pairwiseAlignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNAString("ATGCAACATTGC")
cds2 <- Biostrings::DNAString("ATGCATTGC")
cds2codonaln(cds1, cds2)
```

cdsstring2codonaln *cdsstring2codonaln*

Description

This function takes two sequences as DNAStringSet, and their corresponding AAStringSet, calculates a global alignment and converts this alignment back into a codon alignment.

Usage

```
cdsstring2codonaln(
  cds,
  aa,
  type = "global",
  substitutionMatrix = "BLOSUM62",
  gapOpening = 10,
  gapExtension = 0.5,
  remove.gaps = FALSE
)
```

Arguments

cds	two sequences DNAStringSet [mandatory]
aa	two sequences AAStringSet [mandatory]
type	type of alignment (see pairwiseAlignment) [default: global]
substitutionMatrix	substitution matrix representing the fixed substitution scores for an alignment (see pairwiseAlignment) [default: BLOSUM62]
gapOpening	the cost for opening a gap in the alignment (see pairwiseAlignment) [default: 10]
gapExtension	the incremental cost incurred along the length of the gap in the alignment (see pairwiseAlignment) [default: 0.5]
remove.gaps	specify if gaps in the codon alignment should be removed [default: FALSE]

Value

codon alignment as DNASTringSet

Author(s)

Kristian K Ullrich

References

Pagès, H et al. (2014) Biostrings: Efficient manipulation of biological strings. *R package version, 2(0)*.

See Also

[pairwiseAlignment](#)

Examples

```
## define two cds sequences
cds <- Biostrings::DNASTringSet(c("ATGCAACATTGC", "ATGCATTGC"))
names(cds) <- c("cds1", "cds2")
## get protein alignment
aa <- MSA2dist::cds2aa(cds)
cdsstring2codonaln(cds, aa)
```

codon2numberAMBIG *codon2numberAMBIG*

Description

This function converts a codon into a number, but accept N and -.

Usage

```
codon2numberAMBIG(codon)
```

Arguments

codon [mandatory]

Value

An object of class `numeric`

Author(s)

Kristian K Ullrich

See Also

[GENETIC_CODE](#)

Examples

```
#unlist(lapply(names(Biostrings::GENETIC_CODE), codon2numberAMBIG))
names(Biostrings::GENETIC_CODE) |> codon2numberAMBIG()
```

codon2numberTCAG	<i>codon2numberTCAG</i>
------------------	-------------------------

Description

This function converts a codon into a number.

Usage

```
codon2numberTCAG(codon)
```

Arguments

codon [mandatory]

Value

An object of class `numeric`

Author(s)

Kristian K Ullrich

See Also

[GENETIC_CODE](#)

Examples

```
#unlist(lapply(names(Biostrings::GENETIC_CODE), codon2numberTCAG))
names(Biostrings::GENETIC_CODE) |> codon2numberTCAG()
```

codonmat2pnps	<i>codonmat2pnps</i>
---------------	----------------------

Description

This function calculates pn/ps according to *Nei and Gojobori (1986)*.

Usage

```
codonmat2pnps(codonmat)
```

Arguments

codonmat codon matrix of two columns to be compared [mandatory]

Value

An object of class pnps which is a list with the following components:

seq1 sequence1 name

seq2 sequence2 name

Codons sequence2 name

Compared sequence2 name

Ambiguous sequence2 name

Indels sequence2 name

Ns sequence2 name

Sd sequence2 name

Sn sequence2 name

S sequence2 name

N sequence2 name

ps sequence2 name

pn sequence2 name

pnps sequence2 name

ds sequence2 name

dn sequence2 name

dnds sequence2 name

Author(s)

Kristian K Ullrich

References

Nei and Gojobori. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **3(5)**, 418-426.

Ganeshan et al. (1997) Human immunodeficiency virus type 1 genetic evolution in children with different rates of development of disease. *J. Virology*. **71(1)**, 663-677.

Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*. **155(1)**, 431-449.

See Also

[kaks](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
#codonmat2pnps(dnastring2codonmat(hiv)[,c(1, 2)])
(hiv |> dnastring2codonmat())[,c(1, 2)] |> codonmat2pnps()
```

 codonmat2xy

codonmat2xy

Description

This function calculates average behavior of each codon for all pairwise comparisons for indels, syn, and nonsyn mutations according to *Nei and Gojobori (1986)*.

Usage

```
codonmat2xy(codonmat, threads = 1)
```

Arguments

codonmat	codon matrix obtained via dnastring2codonmat [mandatory]
threads	number of parallel threads [default: 1]

Value

A data.frame object with the following components:

Codon	Codon index
n	number of comparison
SynSum	Sum of syn
NonSynSum	Sum of nonsyn
IndelSum	Sum of indels
SynMean	average syn per codon
NonSynMean	average nonsyn per codon
IndelMean	average indels per codon
CumSumSynMean	cumulative average syn per codon
CumSumNonSynMean	cumulative average nonsyn per codon
CumSumIndelMean	cumulative indels per codon

Author(s)

Kristian K Ullrich

References

Nei and Gojobori. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **3(5)**, 418-426.

Ganeshan et al. (1997) Human immunodeficiency virus type 1 genetic evolution in children with different rates of development of disease. *J. Virology.* **71(1)**, 663-677.

Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics.* **155(1)**, 431-449.

See Also

[dnastring2codonmat](#) [codonmat2pnps](#) [dnastring2kaks](#) [kaks](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
#codonmat2xy(dnastring2codonmat(hiv))
hiv |> dnastring2codonmat() |> codonmat2xy()
#codonmat2xy(dnastring2codonmat(hiv), threads=2)
hiv |> dnastring2codonmat() |> codonmat2xy(threads=2)
```

compareCodons

compareCodons

Description

This function compares two codons and returns the number of syn and non-syn sites according to *Nei and Gojobori (1986)*.

Usage

```
compareCodons(codA, codB)
```

Arguments

codA	codon A [mandatory]
codB	codon B [mandatory]

Value

vector of syn and non-syn sites

Author(s)

Kristian K Ullrich

References

- Nei and Gojobori. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **3(5)**, 418-426.
- Ganeshan et al. (1997) Human immunodeficiency virus type 1 genetic evolution in children with different rates of development of disease. *J. Virology*. **71(1)**, 663-677.
- Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*. **155(1)**, 431-449.

See Also

[kaks](#)

Examples

```
compareCodons("AAA", "TTA")
compareCodons("AAA", "TAT")
compareCodons("AAA", "ATT")
compareCodons("AAA", "TTT")
## load example sequence data
data("hiv", package="MSA2dist")
compareCodons(dnastring2codonmat(hiv)[1,1], dnastring2codonmat(hiv)[1,2])
```

dnabin2dnastring *dnabin2dnastring*

Description

This function converts an ape DNABin into a DNAStrngSet.

Usage

```
dnabin2dnastring(dnabin)
```

Arguments

dnabin ape DNABin [mandatory]

Value

An object of class DNAStrngSet

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#) [as.DNABin.alignment](#) DNAStrngSet

Examples

```
data(woodmouse, package="ape")
## convert into DNAStrngSet
#dnabin2dnastring(woodmouse)
woodmouse |> dnabin2dnastring()
```

dnastring2aln	<i>dnastring2aln</i>
---------------	----------------------

Description

This function converts a DNASTringSet into an seqinr alignment.

Usage

```
dnastring2aln(dna)
```

Arguments

dna DNASTringSet [mandatory]

Value

An object of class alignment which is a list with the following components:

nb the number of aligned sequences

nam a vector of strings containing the names of the aligned sequences

seq a vector of strings containing the aligned sequences

com a vector of strings containing the commentaries for each sequence or NA if there are no comments

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG--CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
## convert into alignment
#dnastring2aln(cds1.cds2.aln)
cds1.cds2.aln |> dnastring2aln()
```

dnastring2codonmat *dnastring2codonmat*

Description

This function converts a DNASTringSet into a codon matrix.

Usage

```
dnastring2codonmat(cds, shorten = FALSE, frame = 1, framelist = NULL)
```

Arguments

cds	DNASTringSet [mandatory]
shorten	shorten all sequences to multiple of three [default: FALSE]
frame	indicates the first base of a the first codon [default: 1]
framelist	supply vector of frames for each entry [default: NULL]

Value

An object of class alignment which is a list with the following components:

nb the number of aligned sequences

nam a vector of strings containing the names of the aligned sequences

seq a vector of strings containing the aligned sequences

com a vector of strings containing the commentaries for each sequence or NA if there are no comments

Author(s)

Kristian K Ullrich

See Also

[as.alignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
## convert into alignment
#dnastring2codonmat(cds1.cds2.aln)
cds1.cds2.aln |> dnastring2codonmat()
## use frame 2 and shorten to circumvent multiple of three error
cds1 <- Biostrings::DNASTring("-ATGCAACATTGC-")
cds2 <- Biostrings::DNASTring("-ATG---CATTGC-")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
cds1.cds2.aln |> dnastring2codonmat(frame=2, shorten=TRUE)
```

`dnastring2dist` *`dnastring2dist`*

Description

This function calculates pairwise distances for all combinations of a DNAStrngSet.

Usage

```

dnastring2dist(
  dna,
  model = "IUPAC",
  threads = 1,
  symmetric = TRUE,
  score = NULL,
  mask = NULL,
  region = NULL,
  ...
)

```

Arguments

<code>dna</code>	DNAStrngSet [mandatory]
<code>model</code>	specify model either "IUPAC" or any model from <code>ape::dist.dna</code> [default: IUPAC]
<code>threads</code>	number of parallel threads [default: 1]
<code>symmetric</code>	symmetric score matrix [default: TRUE]
<code>score</code>	score matrix use score matrix to calculate distances [default: NULL]
<code>mask</code>	IRanges object indicating masked sites [default: NULL]
<code>region</code>	IRanges object indicating region to use for dist calculation. Default is null, meaning all sites are used [default: NULL]
<code>...</code>	other <code>ape::dist.dna</code> parameters (see dist.dna)

Value

A data.frame of pairwise distance values `distSTRING` and sites used `sitesUsed`

Author(s)

Kristian K Ullrich

See Also

[dist.dna](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
#dnastring2dist(hiv, model="IUPAC")
hiv |> dnastring2dist(model="IUPAC")
#dnastring2dist(hiv, model="K80")
hiv |> dnastring2dist(model="K80")
data("woodmouse", package="ape")
#dnastring2dist(dnabin2dnastring(woodmouse), score=iupacMatrix())
woodmouse |> dnabin2dnastring() |> dnastring2dist()
#dnastring2dist(hiv, model = "IUPAC", threads = 2)
hiv |> dnastring2dist(model = "IUPAC", threads = 2)
## create mask
mask1 <- IRanges::IRanges(start=c(1,61,121), end=c(30,90,150))
## use mask
hiv |> dnastring2dist(model="IUPAC", mask=mask1)
## use region
region1 <- IRanges::IRanges(start=c(1,139), end=c(75,225))
hiv |> dnastring2dist(model="IUPAC", region=region1)
## use mask and region
hiv |> dnastring2dist(model="IUPAC", mask=mask1, region=region1)
## use asymmetric score matrix
myscore <- iupacMatrix()
myscore[1, 4] <- 0.5
(hiv |> dnastring2dist(score=myscore, symmetric=FALSE))$distSTRING[1:2, 1:2]
```

dnastring2dnabin

dnastring2dnabin

Description

This function converts a DNAStrngSet into an ape DNABin.

Usage

```
dnastring2dnabin(dna)
```

Arguments

```
dna          DNAStrngSet [mandatory]
```

Value

An object of class DNABin

Author(s)

Kristian K Ullrich

See Also

[as.alignment as.DNABin.alignment](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
## convert into DNABin
#dnastring2dnabin(cds1.cds2.aln)
cds1.cds2.aln |> dnastring2dnabin()
```

dnastring2kaks	<i>dnastring2kaks</i>
----------------	-----------------------

Description

This function calculates Ka/Ks (pN/pS) for all combinations of a DNASTringSet. If the sequences in the DNASTringSet are not a multiple-sequence alignment, pairwise codon alignments can be calculated on the fly. Models used and implemented according to *Li (1993)* (using seqinr) or *Nei and Gojobori (1986)* (own implementation) or models from KaKs_Calculator2 ported to MSA2dist with Rcpp.

Usage

```
dnastring2kaks(
  cds,
  model = "Li",
  threads = 1,
  isMSA = TRUE,
  sgc = "1",
  verbose = FALSE,
  ...
)
```

Arguments

cds	DNASTringSet coding sequence alignment [mandatory]
model	specify codon model either "Li" or "NG86" or one of KaKs_Calculator2 model "NG", "LWL", "LPB", "MLWL", "MLPB", "GY", "YN", "MYN", "MS", "MA", "GNG", "GLWL", "GLPB", "GMLWL", "GMLPB", "GYN", "GMYN" [default: Li]
threads	number of parallel threads [default: 1]
isMSA	cds DNASTringSet represents MSA [default: TRUE]
sgc	standard genetic code (for KaKs Calculator models) [default: 1]
verbose	verbosity (for KaKs Calculator models) [default: FALSE]
...	other codon alignment parameters

Value

A data.frame of KaKs values

Author(s)

Kristian K Ullrich

References

- "MS/MA/GNG/GLWL/GLPB/GMLWL/GMLPB/GYN:" Wang et al. (2010) KaKs_Calculator 2.0: a toolkit incorporating gamma-series methods and sliding window strategies. *Genomics, proteomics & bioinformatics*. **8(1)**, 77-80.
- "Li/LWL:" Li et al. (1985) A new method for estimating synonymous and nonsynonymous rates of nucleotide substitution considering the relative likelihood of nucleotide and codon changes. *Mol. Biol. Evol.*, **2(2)**, 150-174.
- "Li/LPB:" Li (1993). Unbiased estimation of the rates of synonymous and nonsynonymous substitution. *Journal of molecular evolution*, 36(1), pp.96-99.
- "NG86/NG:" Nei and Gojobori. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **3(5)**, 418-426.
- "LPB:" Pamilo and Bianchi. (1993) Evolution of the Zfx and Zfy genes: Rates and interdependence between genes. *Mol. Biol. Evol.*, **10**, 271-281.
- "MLWL/MLPB:" Tzeng et al. (2004). Comparison of three methods for estimating rates of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **21(12)**, 2290-2298.
- "GY:" Goldman and Yang (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.*, **11(5)** 725-736.
- "YN:" Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*. **155(1)**, 431-449.
- "MYN:" Zhang et al. (2006). Computing Ka and Ks with a consideration of unequal transitional substitutions. *BMC evolutionary biology*, **6(1)**, 1-10.
- "data(hiv):" Ganeshan et al. (1997) Human immunodeficiency virus type 1 genetic evolution in children with different rates of development of disease. *J. Virology*. **71(1)**, 663-677.
- Wang et al. (2009). gamma-MYN: a new algorithm for estimating Ka and Ks with consideration of variable substitution rates. *Biology Direct*, **4(1)**, 1-18.

See Also

[kaks](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
#dnastring2kaks(hiv, model="Li")
hiv |> dnastring2kaks(model="Li")
#dnastring2kaks(hiv, model="NG86")
hiv |> dnastring2kaks(model="NG86")
#dnastring2kaks(hiv, model="NG86", threads=2)
hiv |> dnastring2kaks(model="NG86", threads=2)

## define three unaligned cds sequences
cds1 <- Biostrings::DNAString("ATGCAACATTGC")
cds2 <- Biostrings::DNAString("ATGCATTGC")
cds3 <- Biostrings::DNAString("ATGCAATGC")
cds_sequences <- Biostrings::DNAStringSet(list(cds1, cds2, cds3))
names(cds_sequences) <- c("cds1", "cds2", "cds3")
```

```
## set isMSA to FALSE to automatically create pairwise codon alignments
#dnastring2kaks(cds_sequences, model="Li", isMSA=FALSE)
cds_sequences |> dnastring2kaks(model="Li", isMSA=FALSE)
```

GENETIC_CODE_TCAG	<i>GENETIC_CODE_TCAG</i>
-------------------	--------------------------

Description

GENETIC_CODE from Biostrings extended by codon number and number of syn sites.

Usage

```
codon2number(codon)
```

Arguments

codon codon [mandatory]

Value

An object of class `numeric`

Author(s)

Kristian K Ullrich

See Also

[GENETIC_CODE](#)

Examples

```
GENETIC_CODE_TCAG
```

getmask	<i>getmask</i>
---------	----------------

Description

This function shows the mask slot from a `DNAStrngSet` or an `AAStringSet` metadata information.

Usage

```
getmask(seq)
```

Arguments

seq `DNAStrngSet` or `AAStringSet` [mandatory]

Value

IRanges information from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create mask
mask1 <- IRanges::IRanges(start=c(1,41), end=c(20,50))
## add mask
iupac.aa <- iupac.aa |> addmask2string(mask=mask1)
#(iupac.aa |> slot("metadata"))$mask
iupac.aa |> getmask()
```

getpos

getpos

Description

This function shows the position slot from a DNASTringSet or an AAStringSet metadata information.

Usage

```
getpos(seq)
```

Arguments

seq DNASTringSet or AAStringSet [mandatory]

Value

GenomicRanges information from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
## add position
iupac <- iupac |> addpos2string(chrom="chr1", start=1, end=1000)
#(iupac |> slot("metadata"))$GRanges
iupac |> getpos()
```

globalDeletion	<i>globalDeletion</i>
----------------	-----------------------

Description

This function returns a DNASTringSet reduced by all sites containing any gaps ("-", "+", ".") or missing ("N") sites.

Usage

```
globalDeletion(dna)
```

Arguments

dna	DNASTringSet [mandatory]
-----	--------------------------

Value

DNASTringSet

Author(s)

Kristian K Ullrich

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG--CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
globalDeletion(cds1.cds2.aln)
```

globalDeletionAA *globalDeletionAA*

Description

This function returns an AAStringSet reduced by all sites containing any gaps ("-", "+", ".") or missing ("X") sites.

Usage

```
globalDeletionAA(aa)
```

Arguments

aa AAStringSet [mandatory]

Value

AAStringSet

Author(s)

Kristian K Ullrich

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
#globalDeletionAA(cds2aa(cds1.cds2.aln))
cds1.cds2.aln |> cds2aa() |> globalDeletionAA()
```

granthamMatrix *granthamMatrix*

Description

This function creates a granthamMatrix object to be used with the rcpp_distSTRING function. By default, the grantham matrix is defined as from Grantham 1974. (see <https://link.springer.com/article/10.1007/s00335-017-9704-9>)

Usage

```
granthamMatrix()
```

Value

matrix

Author(s)

Kristian K Ullrich

References

Grantham R. (1974). Amino Acid Difference Formula to Help Explain Protein Evolution. *Science*, **185**(4154), 862-864.

See Also

[aastring2dist](#), [dist.dna](#)

Examples

```
granthamMatrix()
```

hiv-data

hiv-data

Description

Example cds sequences from HIV-1 sample 136 patient 1 from Sweden envelope glycoprotein (env) gene, V3 region as DNASTringSet.

Usage

```
data(hiv)
```

Format

an object of class DNASTringSet see [XStringSet-class](#)

References

Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*. **155**(1), 431-449.

Examples

```
data("hiv", package="MSA2dist")
```

indices2kaks

indices2kaks

Description

This function calculates Ka/Ks (pN/pS) for all combinations given in an indices list of a DNASTringSet. If the sequences in the DNASTringSet are not a multiple-sequence alignment, pairwise codon alignments can be calculated on the fly. Models used and implemented according to *Li (1993)* (using seqinr) or *Nei and Gojobori (1986)* (own implementation) or models from KaKs_Calculator2 ported to MSA2dist with Rcpp.

Usage

```
indices2kaks(
  cds,
  indices,
  model = "Li",
  threads = 1,
  isMSA = TRUE,
  sgc = "1",
  verbose = FALSE,
  ...
)
```

Arguments

cds	DNASTringSet coding sequence alignment [mandatory]
indices	list list of indices to calculate Ks/Ks [mandatory]
model	specify codon model either "Li" or "NG86" or one of KaKs_Calculator2 model "NG", "LWL", "LPB", "MLWL", "MLPB", "GY", "YN", "MYN", "MS", "MA", "GNG", "GLWL", "GLPB", "GMLWL", "GMLPB", "GYN", "GMYN" [default: Li]
threads	number of parallel threads [default: 1]
isMSA	cds DNASTringSet represents MSA [default: TRUE]
sgc	standard genetic code (for KaKs Calculator models) [default: 1]
verbose	verbosity (for KaKs Calculator models) [default: FALSE]
...	other codon alignment parameters

Value

A data.frame of KaKs values

Author(s)

Kristian K Ullrich

References

- "MS/MA/GNG/GLWL/GLPB/GMLWL/GMLPB/GYN:" Wang et al. (2010) KaKs_Calculator 2.0: a toolkit incorporating gamma-series methods and sliding window strategies. *Genomics, proteomics & bioinformatics*. **8(1)**, 77-80.
- "Li/LWL:" Li et al. (1985) A new method for estimating synonymous and nonsynonymous rates of nucleotide substitution considering the relative likelihood of nucleotide and codon changes. *Mol. Biol. Evol.*, **2(2)**, 150-174.
- "Li/LPB:" Li (1993). Unbiased estimation of the rates of synonymous and nonsynonymous substitution. *Journal of molecular evolution*, 36(1), pp.96-99.
- "NG86/NG:" Nei and Gojobori. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **3(5)**, 418-426.
- "LPB:" Pamilo and Bianchi. (1993) Evolution of the Zfx and Zfy genes: Rates and interdependence between genes. *Mol. Biol. Evol.*, **10**, 271-281.
- "MLWL/MLPB:" Tzeng et al. (2004). Comparison of three methods for estimating rates of synonymous and nonsynonymous nucleotide substitutions. *Mol. Biol. Evol.*, **21(12)**, 2290-2298.
- "GY:" Goldman and Yang (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.*, **11(5)** 725-736.
- "YN:" Yang et al. (2000) Codon-substitution models for heterogeneous selection pressure at amino acid sites. *Genetics*. **155(1)**, 431-449.
- "MYN:" Zhang et al. (2006). Computing Ka and Ks with a consideration of unequal transitional substitutions. *BMC evolutionary biology*, **6(1)**, 1-10.
- "data(hiv):" Ganeshan et al. (1997) Human immunodeficiency virus type 1 genetic evolution in children with different rates of development of disease. *J. Virology*. **71(1)**, 663-677.
- Wang et al. (2009). gamma-MYN: a new algorithm for estimating Ka and Ks with consideration of variable substitution rates. *Biology Direct*, **4(1)**, 1-18.

See Also

[kaks](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
## create indices
idx <- list(c(2, 3), c(5,7,9))
#indices2kaks(hiv, idx, model="Li")
hiv |> indices2kaks(idx, model="Li")
#indices2kaks(hiv, idx, model="NG86")
hiv |> indices2kaks(idx, model="NG86")
#indices2kaks(hiv, idx, model="NG86", threads=2)
hiv |> indices2kaks(idx, model="NG86", threads=2)

## define three unaligned cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATGCATTGC")
cds3 <- Biostrings::DNASTring("ATGCAATGC")
cds_sequences <- Biostrings::DNASTringSet(list(cds1, cds2, cds3))
names(cds_sequences) <- c("cds1", "cds2", "cds3")
## create indices
idx <- list(c(1, 2), c(1,3))
```

```
## set isMSA to FALSE to automatically create pairwise codon alignments
#indices2kaks(cds_sequences, idx, model="Li", isMSA=FALSE)
cds_sequences |> indices2kaks(idx, model="Li", isMSA=FALSE)
```

iupac-data

iupac-data

Description

Example IUPAC sequences created with `angsd` from different house mouse (*Mus musculus*) sub-populations from Harr et al. (2016) `DNAStrngSet`.

Usage

```
data(iupac)
```

Format

an object of class `DNAStrngSet` see [XStringSet-class](#)

References

Harr et al. (2016) Genomic resources for wild populations of the house mouse, *Mus musculus* and its close relative *Mus spretus*. *Scientific data*. **3(1)**, 1-14.

Examples

```
data("iupac", package="MSA2dist")
```

iupacMatrix

iupacMatrix

Description

This function creates a `iupacMatrix` object to be used with the `rcpp_distSTRING` function. By default, the `iupac matrix` is defined as literal distance obtained from Chang et al. 2017. (see <https://link.springer.com/article/10.1007/s00335-017-9704-9>)

Usage

```
iupacMatrix()
```

Value

score matrix

Author(s)

Kristian K Ullrich

References

Chang, P. L., Kopania, E., Keeble, S., Sarver, B. A., Larson, E., Orth, A., ... & Dean, M. D. (2017). Whole exome sequencing of wild-derived inbred strains of mice improves power to link phenotype and genotype. *Mammalian genome*, **28**(9-10), 416-425.

See Also

[dnastring2dist](#), [dist.dna](#)

Examples

```
iupacMatrix()
```

makePostalignedSeqs *makePostalignedSeqs*

Description

This function is a fork from an internal function from Biostrings

Usage

```
makePostalignedSeqs(x)
```

Arguments

x x

Value

get internal function makePostalignedSeqs

Author(s)

Kristian K Ullrich

See Also

[pairwiseAlignment](#), [cds2codonaln](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATGCATTGC")
makePostalignedSeqs(pwalign::pairwiseAlignment(
  cds2aa(Biostrings::DNASTringSet(cds1)),
  cds2aa(Biostrings::DNASTringSet(cds2))))
```

`pal2nal`*pal2nal*

Description

This function takes an `AAStringSet` alignment and its corresponding coding sequences `DNAStrngSet` and converts the protein alignment into a codon alignment.

Usage

```
pal2nal(pal, nal, remove.gaps = FALSE)
```

Arguments

<code>pal</code>	<code>AAStringSet</code> [mandatory]
<code>nal</code>	<code>DNAStrngSet</code> [mandatory]
<code>remove.gaps</code>	specify if gaps in the codon alignment should be removed [default: FALSE]

Value

codon alignment as `DNAStrngSet`

Author(s)

Kristian K Ullrich

References

Pagès, H et al. (2014) Biostrings: Efficient manipulation of biological strings. *R package version, 2(0)*.

See Also

[pairwiseAlignment](#)

Examples

```
## define two cds sequences
cds <- Biostrings::DNAStrngSet(c("ATGCAACATTGC", "ATGCATTGC"))
names(cds) <- c("cds1", "cds2")
## get protein alignment
aa <- MSA2dist::cds2aa(cds)
msa <- makePostalignedSeqs(pwalign::pairwiseAlignment(aa[1], aa[2]))[[1L]]
names(msa) <- names(aa)
## get codon alignment
nal <- MSA2dist::pal2nal(pal=msa, nal=cds)
nal
```

popinteger	<i>popinteger</i>
------------	-------------------

Description

This function shows the population integer slot from a DNAStrngSet or an AAStringSet metadata information.

Usage

```
popinteger(seq)
```

Arguments

```
seq          DNAStrngSet or AAStringSet [mandatory]
```

Value

population integer from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create poplist
poplist <- list(FRA = grep("Mmd.FRA", names(iupac)),
               GER = grep("Mmd.GER", names(iupac)),
               IRA = grep("Mmd.IRA", names(iupac)),
               AFG = grep("Mmm.AFG", names(iupac)))
iupac.aa <- iupac.aa |> addpop2string(poplist)
popinteger(iupac.aa)
```

popnames	<i>popnames</i>
----------	-----------------

Description

This function shows the population names slot from a DNAStrngSet or an AAStringSet metadata information.

Usage

```
popnames(seq)
```

Arguments

```
seq          DNAStrngSet or AAStringSet [mandatory]
```

Value

population names from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create poplist
poplist <- list(FRA = grep("Mmd.FRA", names(iupac)),
               GER = grep("Mmd.GER", names(iupac)),
               IRA = grep("Mmd.IRA", names(iupac)),
               AFG = grep("Mmm.AFG", names(iupac)))
iupac.aa <- iupac.aa |> addpop2string(poplist)
popnames(iupac.aa)
```

rcpp_distSTRING	<i>rcpp_distSTRING</i>
-----------------	------------------------

Description

calculates pairwise distances using a score matrix

Usage

```
rcpp_distSTRING(dnavector, scoreMatrix, ncores = 1L, symmetric = 1L)
```

Arguments

dnavector	StringVector [mandatory]
scoreMatrix	NumericMatrix [mandatory]
ncores	number of cores [default: 1]
symmetric	symmetric score matrix [default: 1]

Value

list

Author(s)

Kristian K Ullrich

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
rcpp_distSTRING(dnavector=as.character(hiv), scoreMatrix=iupacMatrix())
```

rcpp_KaKs	<i>rcpp_KaKs</i>
-----------	------------------

Description

calculates KaKs as implemented in KaKs Calculator 2.0 MSA2dist with Rcpp.

Usage

```
rcpp_KaKs(cdsstr, sgc = "1", method = "YN", verbose = FALSE)
```

Arguments

cdsstr	StringVector [mandatory]
sgc	standard genetic code to use [default: 1]
method	KaKs Calculator 2.0 codon model [default: YN]
verbose	specify if verbose output [default: FALSE]

Value

list

Author(s)

Kristian K Ullrich

References

Wang et al. (2010) KaKs_Calculator 2.0: a toolkit incorporating gamma-series methods and sliding window strategies. *Genomics, proteomics & bioinformatics*. **8(1)**, 77-80.

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
rcpp_KaKs(cdsstr=as.character(hiv[1:3]))
```

```
rcpp_pairwiseDeletionAA
      rcpp_pairwiseDeletionAA
```

Description

returns number of AA sites used

Usage

```
rcpp_pairwiseDeletionAA(aavector, ncores = 1L, symmetric = 1L)
```

Arguments

aavector	StringVector [mandatory]
ncores	number of cores [default: 1]
symmetric	symmetric score matrix [default: 1]

Value

list

Author(s)

Kristian K Ullrich

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
h <- hiv |> cds2aa() |> as.character()
rcpp_pairwiseDeletionAA(aavector=h, ncores=1)
```

```
rcpp_pairwiseDeletionDNA
      rcpp_pairwiseDeletionDNA
```

Description

returns number of DNA sites used

Usage

```
rcpp_pairwiseDeletionDNA(dnavector, ncores = 1L, symmetric = 1L)
```

Arguments

dnavector	StringVector [mandatory]
ncores	number of cores [default: 1]
symmetric	symmetric score matrix [default: 1]

Value

list

Author(s)

Kristian K Ullrich

Examples

```
## load example sequence data
data("woodmouse", package="ape")
w <- woodmouse |> dnabin2dnastring() |> as.character()
rcpp_pairwiseDeletionDNA(dnavector=w, ncores=1)
```

region	<i>region</i>
--------	---------------

Description

This function shows the region slot from a DNASTringSet or an AAStringSet metadata information.

Usage

```
region(seq)
```

Arguments

seq	DNASTringSet or AAStringSet [mandatory]
-----	---

Value

region IRanges object from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data(iupac, package="MSA2dist")
iupac.aa <- iupac |> cds2aa(shorten = TRUE)
## create region
region1 <- IRanges::IRanges(start=c(1,41), end=c(20,50))
## add region
iupac.aa <- iupac.aa |> addregion2string(region=region1)
iupac.aa |> region()
```

regionused

regionused

Description

This function shows the region used slot from a DNASTringSet or an AAStringSet metadata information.

Usage

```
regionused(seq)
```

Arguments

seq DNASTringSet or AAStringSet [mandatory]

Value

population names from metadata

Author(s)

Kristian K Ullrich

See Also

[addpop2string](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
## create mask
mask1 <- IRanges::IRanges(start=c(11,41,71), end=c(20,50,80))
## use mask
hiv.region <- hiv |> cds2aa() |> string2region(mask=mask1)
#(hiv.region |> slot("metadata"))$regionUsed
hiv.region |> regionused()
```

string2region	<i>string2region</i>
---------------	----------------------

Description

This function subsets a DNASTringSet or an AAStringSet by a mask and region given one or both options as IRanges.

Usage

```
string2region(seq, mask = NULL, region = NULL, add = TRUE)
```

Arguments

seq	DNASTringSet or AAStringSet [mandatory]
mask	IRanges object indicating masked sites [default: NULL]
region	IRanges object indicating region to use for dist calculation (by default all sites are used) [default: NULL]
add	indicate if mask and region should be added to metadata [default: TRUE]

Value

A list object with the following components:
DNASTringSet or AAStringSet
regionUsed

Author(s)

Kristian K Ullrich

See Also

[dnastring2dist](#)

Examples

```
## load example sequence data
data("hiv", package="MSA2dist")
## create mask
mask1 <- IRanges::IRanges(start=c(11,41,71), end=c(20,50,80))
## use mask
hiv.region <- hiv |> cds2aa() |> string2region(mask=mask1)
#(hiv.region |> slot("metadata"))$regionUsed
hiv.region |> regionused()
## use region
region1 <- IRanges::IRanges(start=c(1,75), end=c(45,85))
hiv.region <- hiv |> cds2aa() |> string2region(region=region1)
#(hiv.region |> slot("metadata"))$regionUsed
hiv.region |> regionused()
## use mask and region
hiv.region <- hiv |> cds2aa() |> string2region(mask=mask1, region=region1)
#(hiv.region |> slot("metadata"))$regionUsed
hiv.region |> regionused()
```

subString

subString

Description

This function gets a subsequence from a DNAString, RNAString, AAString, BString, DNAStringSet, RNAStringSet, AAStringSet, BStringSet object from the Biostrings package.

Usage

```
subString(x, s, e)
```

Arguments

x	DNAStringSet, RNAString, AAString, BString, DNAStringSet, RNAStringSet, AAStringSet, BStringSet [mandatory]
s	start vector [mandatory]
e	end vector [mandatory]

Value

subsequence of an Biostrings object

Author(s)

Kristian K Ullrich

See Also

[subseq](#)

Examples

```
## define two cds sequences
cds1 <- Biostrings::DNASTring("ATGCAACATTGC")
cds2 <- Biostrings::DNASTring("ATG---CATTGC")
cds1.cds2.aln <- c(Biostrings::DNASTringSet(cds1),
  Biostrings::DNASTringSet(cds2))
subString(cds1.cds2.aln, c(1,7), c(3,12))
```

uptriidx	<i>uptriidx</i>
----------	-----------------

Description

This function returns upper tri index for usage with pivot_long reduction.

Usage

```
uptriidx(n, diag = FALSE)
```

Arguments

n	dimension of initial matrix [mandatory]
diag	indicate if diag should be retained [default: FALSE]

Value

list of positions

Author(s)

Kristian K Ullrich

Examples

```
uptriidx(10)
```

Index

* datasets

- AAMatrix-data, 4
 - hiv-data, 33
 - iupac-data, 36
- aa2selfscore, 3
- aabin2aastring, 4
- AAMatrix (AAMatrix-data), 4
- AAMatrix-data, 4
- aastring2aabin, 5
- aastring2aln, 6
- aastring2dist, 7, 33
- AAStringSet, 4, 12
- addmask2string, 8, 9, 11
- addpop2string, 8, 9, 11, 30, 39, 40, 44
- addpos2string, 8, 9, 10, 11
- addrregion2string, 8, 9, 11, 11
- aln2aastring, 12
- aln2dnastring, 13
- as.alignment, 4–6, 12, 13, 22–24, 26
- as.DNAbin.alignment, 4, 5, 22, 26
- cds2aa, 14
- cds2codonaln, 15, 37
- cdsstring2codonaln, 16
- codon2number (GENETIC_CODE_TCAG), 29
- codon2numberAMBIG, 17
- codon2numberTCAG, 18
- codonmat2pnps, 18, 20
- codonmat2xy, 20
- compareCodons, 21
- dist.dna, 25, 33, 37
- dnabin2dnastring, 22
- dnastring2aln, 23
- dnastring2codonmat, 20, 24
- dnastring2dist, 7, 25, 37, 45
- dnastring2dnabin, 26
- dnastring2kaks, 20, 27
- DNAStrngSet, 13, 22
- GENETIC_CODE, 17, 18, 29
- GENETIC_CODE_TCAG, 29
- getmask, 29
- getpos, 30
- globalDeletion, 31
- globalDeletionAA, 32
- granthamMatrix, 32
- hiv (hiv-data), 33
- hiv-data, 33
- indices2kaks, 34
- iupac (iupac-data), 36
- iupac-data, 36
- iupacMatrix, 36
- kaks, 19–21, 28, 35
- makePostalignedSeqs, 37
- pairwiseAlignment, 15–17, 37, 38
- pal2nal, 38
- popinteger, 39
- popnames, 40
- rccpp_distSTRING, 41
- rccpp_KaKs, 41
- rccpp_pairwiseDeletionAA, 42
- rccpp_pairwiseDeletionDNA, 43
- region, 43
- regionused, 44
- string2region, 45
- subseq, 46
- substitution_matrices, 3
- subString, 46
- translate, 14
- uptriidx, 47