

# Package: MSA2dist (via r-universe)

September 29, 2026

**Type** Package

**Title** MSA2dist calculates pairwise distances for DNAStringSet/AAStringSet using custom scoring, conducts codon analyses (including dNdS), and estimates Dxy/FST

**Version** 1.17.8

**Description** MSA2dist calculates pairwise distances between all sequences of a DNAStringSet 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. DNAStringSet 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. In addition, MSA2dist provides population genetic analyses, including the calculation of nucleotide divergence between populations (Dxy) and genetic differentiation statistics (FST) from aligned sequence data.

**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, MultipleAlignment, 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

**URL** <https://github.com/kullrich/MSA2dist>

**BugReports** <https://github.com/kullrich/MSA2dist/issues>

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libicu-dev zlib1g-dev

**Repository** <https://bioc.r-universe.dev>

**Date/Publication** 2026-06-26 21:52:06 UTC

**RemoteUrl** <https://github.com/bioc/MSA2dist>

**RemoteRef** HEAD

**RemoteSha** bf677287ca98daddec2381a95149e25cf5f9d93

## Contents

|                                |    |
|--------------------------------|----|
| aa2selfscore . . . . .         | 3  |
| aabin2astring . . . . .        | 4  |
| AAMatrix-data . . . . .        | 5  |
| astring2aabin . . . . .        | 6  |
| astring2aln . . . . .          | 7  |
| astring2dist . . . . .         | 8  |
| addmask2string . . . . .       | 9  |
| addpop2string . . . . .        | 10 |
| addpos2string . . . . .        | 12 |
| addregion2string . . . . .     | 13 |
| aln2astring . . . . .          | 14 |
| aln2dnastring . . . . .        | 15 |
| cds2aa . . . . .               | 16 |
| cds2codonaln . . . . .         | 17 |
| cdsstring2codonaln . . . . .   | 18 |
| codon2numberAMBIG . . . . .    | 20 |
| codon2numberTCAG . . . . .     | 20 |
| codonmat2pnps . . . . .        | 21 |
| codonmat2xy . . . . .          | 22 |
| collapseChar . . . . .         | 24 |
| collapseXStringSet . . . . .   | 24 |
| compareCodons . . . . .        | 25 |
| dist2dxy . . . . .             | 26 |
| dnabin2dnastring . . . . .     | 29 |
| dnastring2aln . . . . .        | 30 |
| dnastring2codonmat . . . . .   | 31 |
| dnastring2dist . . . . .       | 32 |
| dnastring2dnabin . . . . .     | 34 |
| dnastring2dxy . . . . .        | 35 |
| dnastring2kaks . . . . .       | 38 |
| dnastring2weightedPi . . . . . | 40 |

|                                    |           |
|------------------------------------|-----------|
| GENETIC_CODE_TCAG . . . . .        | 42        |
| getmask . . . . .                  | 43        |
| getpos . . . . .                   | 44        |
| globalDeletion . . . . .           | 45        |
| globalDeletionAA . . . . .         | 46        |
| granthamMatrix . . . . .           | 46        |
| hiv-data . . . . .                 | 47        |
| indices2kaks . . . . .             | 48        |
| iupac-data . . . . .               | 50        |
| iupacMatrix . . . . .              | 50        |
| makePostalignedSeqs . . . . .      | 51        |
| normpop . . . . .                  | 52        |
| pal2nal . . . . .                  | 54        |
| popinteger . . . . .               | 55        |
| popnames . . . . .                 | 56        |
| rcpp_distSTRING . . . . .          | 57        |
| rcpp_dxyfst_pop . . . . .          | 58        |
| rcpp_KaKs . . . . .                | 59        |
| rcpp_pairwiseDeletionAA . . . . .  | 60        |
| rcpp_pairwiseDeletionDNA . . . . . | 60        |
| rcpp_weightedPi . . . . .          | 61        |
| rcpp_weightedPi_pop . . . . .      | 62        |
| region . . . . .                   | 63        |
| regionused . . . . .               | 64        |
| string2region . . . . .            | 65        |
| subString . . . . .                | 66        |
| uptriidx . . . . .                 | 67        |
| <b>Index</b>                       | <b>68</b> |

---

|              |                     |
|--------------|---------------------|
| aa2selfscore | <i>aa2selfscore</i> |
|--------------|---------------------|

---

**Description**

This function return the selfscore from an AAStringSet.

**Usage**

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

**Arguments**

- aa                   AAStringSet [mandatory]
- scorematrix       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 AAStringSet.

**Usage**

```
aabin2aastring(aabin)
```

**Arguments**

aabin                      ape AAbin [mandatory]

**Value**

An object of class AAStringSet

**Author(s)**

Kristian K Ullrich

**See Also**

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

## Examples

```
data(woodmouse, package="ape")
## convert into AAStringSet
#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

[dnasttring2dist](#)

**Examples**

```
## load example sequence data
data("hiv", package="MSA2dist")
#aastree2dist(cds2aa(hiv), score=granthamMatrix())
hiv |> cds2aa() |> aastree2dist(score=granthamMatrix())
## create mask
mask1 <- IRanges::IRanges(start=c(11,41,71), end=c(20,50,80))
## use mask
hiv |> cds2aa() |> aastree2dist(score=granthamMatrix(), mask=mask1)
## use region
region1 <- IRanges::IRanges(start=c(1,75), end=c(45,85))
hiv |> cds2aa() |> aastree2dist(score=granthamMatrix(), region=region1)
## use mask and region
hiv |> cds2aa() |> aastree2dist(score=granthamMatrix(),
  mask=mask1, region=region1)
## use asymmetric score matrix
myscore <- granthamMatrix()
myscore[5, 6] <- 0
h <- hiv |> cds2aa() |> aastree2dist(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 | <i>addpos2string</i> |
|---------------|----------------------|

---

### Description

This function adds GenomicRanges information, CHROM, START and END to a DNASTringSet 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   | DNASTringSet or AAStringSet [mandatory] |
| chrom | chromosome name [mandatory]             |
| start | start position [mandatory]              |
| end   | end position [mandatory]                |

### Value

An object of class DNASTringSet 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 | <i>aln2dnastring</i> |
|---------------|----------------------|

---

## 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**

|              |                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------|
| 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]                                        |
| genetic.code | The genetic code to use for the translation of codons into Amino Acid letters [default: NULL] |
| return.cds   | 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 <a href="#">pairwiseAlignment</a> ) [default: global]                                                                 |
| substitutionMatrix | substitution matrix representing the fixed substitution scores for an alignment (see <a href="#">pairwiseAlignment</a> ) [default: BLOSUM62] |
| gapOpening         | the cost for opening a gap in the alignment (see <a href="#">pairwiseAlignment</a> ) [default: 10]                                           |
| gapExtension       | the incremental cost incurred along the length of the gap in the alignment (see <a href="#">pairwiseAlignment</a> ) [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**

|                                 |                                                                                                                                              |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cds</code>                | two sequences DNASTringSet [mandatory]                                                                                                       |
| <code>aa</code>                 | two sequences AAStringSet [mandatory]                                                                                                        |
| <code>type</code>               | type of alignment (see <a href="#">pairwiseAlignment</a> ) [default: global]                                                                 |
| <code>substitutionMatrix</code> | substitution matrix representing the fixed substitution scores for an alignment (see <a href="#">pairwiseAlignment</a> ) [default: BLOSUM62] |
| <code>gapOpening</code>         | the cost for opening a gap in the alignment (see <a href="#">pairwiseAlignment</a> ) [default: 10]                                           |
| <code>gapExtension</code>       | the incremental cost incurred along the length of the gap in the alignment (see <a href="#">pairwiseAlignment</a> ) [default: 0.5]           |
| <code>remove.gaps</code>        | 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 | <i>codon2numberAMBIG</i> |
|-------------------|--------------------------|

---

**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`

*codonmat2pnps*

---

**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  
 dn ds 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 <a href="#">dnastring2codonmat</a> [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**

[dnasttring2codonmat](#) [codonmat2pnps](#) [dnasttring2kaks](#) [kaks](#)

**Examples**

```
## load example sequence data
data("hiv", package="MSA2dist")
#codonmat2xy(dnasttring2codonmat(hiv))
hiv |> dnasttring2codonmat() |> codonmat2xy()
#codonmat2xy(dnasttring2codonmat(hiv), threads=2)
hiv |> dnasttring2codonmat() |> codonmat2xy(threads=2)
```

---

|              |                     |
|--------------|---------------------|
| collapseChar | <i>collapseChar</i> |
|--------------|---------------------|

---

**Description**

This function collapse identical character sequences into unique patterns with a mapping back to the original order.

**Usage**

```
collapseChar(xchar)
```

**Arguments**

|       |                              |
|-------|------------------------------|
| xchar | character vector [mandatory] |
|-------|------------------------------|

**Value**

list containing:

- unique
- xmap

**Author(s)**

Kristian K Ullrich

**Examples**

```
data(woodmouse, package="ape")
#collapseChar(as.character(dnabin2dnastring(woodmouse)))
woodmouse |> dnabin2dnastring() |> as.character() |>
collapseChar()
```

---

|                    |                           |
|--------------------|---------------------------|
| collapseXStringSet | <i>collapseXStringSet</i> |
|--------------------|---------------------------|

---

**Description**

This function collapse identical sequences into unique patterns with a mapping back to original order from a BStringSet, DNASTringSet, RNASTringSet or AAStringSet.

**Usage**

```
collapseXStringSet(xstringset)
```

**Arguments**

xstringset      XStringSet [mandatory]

**Value**

list containing:

- unique
- xmap

**Author(s)**

Kristian K Ullrich

**See Also**

[XStringSet-class](#)

**Examples**

```
data(woodmouse, package="ape")
#collapseXStringSet(dnabin2dnastring(woodmouse))
woodmouse |> dnabin2dnastring() |> collapseXStringSet()
```

---

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])
```

---

|          |                 |
|----------|-----------------|
| dist2dxy | <i>dist2dxy</i> |
|----------|-----------------|

---

Description

This function takes the results of the `dnastring2dist` function to calculate dxy and fst for all population combinations and optional RND, Gmin, RNDmin and RNDmax if one specifies popx, popy and popout.

Usage

```
dist2dxy(d, pop = NULL, popx = NULL, popy = NULL, popout = NULL)
```

Arguments

|      |                                                                                                                                                                                                                                                                                                                                                             |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| d    | list returned by <code>dnastring2dist()</code>                                                                                                                                                                                                                                                                                                              |
| pop  | [default: NULL] optional population assignment: <ul style="list-style-type: none"><li>• character vector (same length as dna)</li><li>• named character vector (mapping seqnames to pop)</li><li>• integer vector (same length as dna)</li><li>• named integer vector (mapping index to pop)</li><li>• list (see <code>addpop2dnastring()</code>)</li></ul> |
| popx | [default: NULL] Character string specifying population x in the rooted population topology ((x,y),o)                                                                                                                                                                                                                                                        |

|        |                                                                                                             |
|--------|-------------------------------------------------------------------------------------------------------------|
| popy   | [default: NULL] Character string specifying population y in the rooted population topology ((x,y),o)        |
| popout | [default: NULL] Character string specifying outgroup population in the rooted population topology ((x,y),o) |

### Details

The function computes pairwise genetic differentiation statistics from a precomputed sequence distance matrix.

Let  $d_{ij}$  denote the per-site sequence distance between sequences  $i$  and  $j$ , and let  $S_{ij}$  be the number of aligned sites used.

#### Within-population diversity

For population  $x$  with sequences  $inx$ :

$$D_x = \frac{1}{\binom{n_x}{2}} \sum_{i < j \text{ in } X} d_{ij}$$

#### Between-population divergence ( $D_{xy}$ )

For populations  $x$  and  $y$ :

$$D_{xy} = \frac{1}{n_x n_y} \sum_{i \text{ in } x} \sum_{j \text{ in } y} d_{ij}$$

#### Fixation index ( $F_{ST}$ )

Using the distance-based estimator:

$$F_{ST} = \frac{D_{xy} - \frac{D_x + D_y}{2}}{D_{xy}}$$

#### Minimum divergence ( $D_{min}$ )

$$D_{min}(x, y) = \min_{i \text{ in } x, j \text{ in } y} d_{ij}$$

#### Outgroup-corrected divergence

Let  $o$  be an outgroup population:

$$D(x, o) = \frac{1}{n_x n_o} \sum_{i \text{ in } x} \sum_{k \text{ in } o} d_{ik}$$

$$D(y, o) = \frac{1}{n_y n_o} \sum_{j \text{ in } y} \sum_{k \text{ in } o} d_{jk}$$

$$D_{out} = \frac{D(x, o) + D(y, o)}{2}$$

#### RND statistics

$$RND = \frac{D_{xy}}{D_{out}}$$

$$G_{min} = \frac{D_{min}(x, y)}{D_{xy}}$$

$$RND_{min} = \frac{D_{min}(x, y)}{D_{out}}$$

### Interpretation

- Dxy: average sequence divergence between populations
- FST: relative differentiation scaled by within-population diversity
- Dmin: closest haplotype distance between populations
- RND: divergence normalized by outgroup distance
- Gmin: proportion of minimum divergence relative to mean divergence
- RNDmin: minimum divergence scaled by outgroup distance
- RNDmax: max divergence scaled by outgroup distance

### Value

list containing:

- dxy pairwise matrix
- fst pairwise matrix
- pi\_within pop matrix
- pi\_between 3d matrix
- valid\_sites
- populations
- popx
- popy
- popout
- Dx
- Dy
- Dxy
- FSTxy
- Dmin
- Dmax
- Dx\_out
- Dy\_out
- Dout
- Gmin

- RND
- RNDmin
- RNDmax
- regionUsed
- pop.info

**Author(s)**

Kristian K Ullrich

**See Also**

[dnastring2dist](#), [addpop2string](#)

**Examples**

```
## load example sequence data
data("iupac", package="MSA2dist")
#d <- dnastring2dist(iupac)
#dist2dxy(d)
iupac |> dnastring2dist() |> dist2dxy()
## 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 <- iupac |> addpop2string(poplist)
#d <- dnastring2dist(iupac)
#dist2dxy(d, pop=popinteger(iupac))
iupac |> dnastring2dist() |> dist2dxy(pop=popinteger(iupac))
#d <- dnastring2dist(iupac)
#dist2dxy(d, pop=popinteger(iupac),
#popx="FRA", popy="IRA", popout="AFG")
iupac |> dnastring2dist() |> dist2dxy(pop=popinteger(iupac),
popx="FRA", popy="IRA", popout="AFG")
```

---

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 DNASTringSet

**Author(s)**

Kristian K Ullrich

**See Also**

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

**Examples**

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

---

dnasttring2aln

*dnasttring2aln*

---

**Description**

This function converts a DNASTringSet into an seqinr alignment.

**Usage**

```
dnasttring2aln(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 | <i>dnastring2codonmat</i> |
|--------------------|---------------------------|

---

**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 DNAStringSet.

## Usage

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

## Arguments

|           |                                                                                            |
|-----------|--------------------------------------------------------------------------------------------|
| dna       | DNAStringSet [mandatory]                                                                   |
| model     | specify model either "IUPAC" or any model from <code>ape::dist.dna</code> [default: IUPAC] |
| threads   | number of parallel threads [default: 1]                                                    |
| symmetric | symmetric score matrix [default: TRUE]                                                     |
| score     | score matrix use score matrix to calculate distances [default: NULL]                       |
| mask      | IRanges object indicating masked sites [default: NULL]                                     |

|          |                                                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------|
| region   | IRanges object indicating region to use for dist calculation. Default is null, meaning all sites are used [default: NULL] |
| collapse | collapse sequence to patterns first [default: TRUE]                                                                       |
| ...      | other ape::dist.dna parameters (see <a href="#">dist.dna</a> )                                                            |

**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]
```

---

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>dnastring2dnabin</code> | <i><code>dnastring2dnabin</code></i> |
|-------------------------------|--------------------------------------|

---

## Description

This function converts a `DNAStringSet` into an ape `DNABin`.

## Usage

```
dnastring2dnabin(dna)
```

## Arguments

|                  |                                       |
|------------------|---------------------------------------|
| <code>dna</code> | <code>DNAStringSet</code> [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()
```

---

|               |                      |
|---------------|----------------------|
| dnastring2dxy | <i>dnastring2dxy</i> |
|---------------|----------------------|

---

## Description

This function calculates dxy and fst for all population combinations of a DNAStrngSet.

## Usage

```
dnastring2dxy(
  dna,
  model = "IUPAC",
  estimator = "prob",
  mask = NULL,
  region = NULL,
  pop = NULL
)
```

## Arguments

|           |                                                                                                                                                                                                                                                                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dna       | DNAStrngSet [mandatory]                                                                                                                                                                                                                                                                                                                              |
| model     | specify model either "IUPAC" (diploid; decode IUPAC) or "sequence" (haploid; only A,C,T,G counted as discrete alleles) [default: IUPAC]                                                                                                                                                                                                              |
| estimator | specify estimator either "count" (count-based; computes nucleotide diversity as the observed proportion of pairwise allele differences, nDiff / nComp) or "prob" (allele-frequency based; computes expected heterozygosity as $1 - \sum p_i^2$ ) [default: prob]                                                                                     |
| mask      | IRanges object indicating masked sites [default: NULL]                                                                                                                                                                                                                                                                                               |
| region    | IRanges object indicating region to use for dist calculation. Default is null, meaning all sites are used [default: NULL]                                                                                                                                                                                                                            |
| pop       | [default: NULL] optional population assignment: <ul style="list-style-type: none"> <li>• character vector (same length as dna)</li> <li>• named character vector (mapping seqnames to pop)</li> <li>• integer vector (same length as dna)</li> <li>• named integer vector (mapping index to pop)</li> <li>• list (see addpop2dnastring())</li> </ul> |

## Details

Four estimator/model combinations are available:

### 1. model = "IUPAC", estimator = "prob"

IUPAC ambiguity codes are decoded into allele counts. For example, R contributes one A and one G allele, while N contributes one A, one C, one G and one T allele. Allele frequencies are estimated as

$$p_i = \frac{n_i}{\sum_j n_j}$$

Within-population diversity is calculated as expected heterozygosity:

$$\pi = 1 - \sum_i p_i^2$$

Between-population divergence is calculated from allele frequencies:

$$d_{xy} = 1 - \sum_i p_{x,i} p_{y,i}$$

Mean values are obtained by averaging across all valid sites.

### 2. model = "IUPAC", estimator = "count"

IUPAC ambiguity codes are decoded into allele counts as above. Nucleotide diversity is estimated directly from pairwise allele counts:

$$\pi = \frac{AC + AG + AT + CG + CT + GT}{\binom{n}{2}}$$

where  $A, C, G, T$  denote allele counts at a site and  $n$  is the total number of decoded alleles.

Between-population divergence is estimated from observed pairwise differences between populations:

$$d_{xy} = \frac{A_x(C_y + G_y + T_y) + C_x(A_y + G_y + T_y) + G_x(A_y + C_y + T_y) + T_x(A_y + C_y + G_y)}{n_x n_y}$$

### 3. model = "sequence", estimator = "prob"

Only unambiguous nucleotides A, C, G and T are used. Ambiguous symbols are ignored. Allele frequencies are estimated from the observed nucleotides:

$$p_i = \frac{n_i}{n}$$

Within-population diversity and between-population divergence are:

$$\pi = 1 - \sum_i p_i^2$$

$$d_{xy} = 1 - \sum_i p_{x,i} p_{y,i}$$

### 4. model = "sequence", estimator = "count"

Only unambiguous nucleotides A, C, G and T are counted as alleles. Ambiguous symbols are ignored. Nucleotide diversity is estimated from pairwise differences:

$$\pi = \frac{AC + AG + AT + CG + CT + GT}{\binom{n}{2}}$$

Between-population divergence is estimated from pairwise differences between populations:

$$d_{xy} = \frac{A_x(C_y + G_y + T_y) + C_x(A_y + G_y + T_y) + G_x(A_y + C_y + T_y) + T_x(A_y + C_y + G_y)}{n_x n_y}$$

For all estimator/model combinations,

$$F_{ST} = \frac{d_{xy} - \frac{\pi_x + \pi_y}{2}}{d_{xy}}$$

where  $\pi_x$  and  $\pi_y$  are the mean within-population nucleotide diversities and  $d_{xy}$  is the mean between-population nucleotide divergence.

### Value

list containing:

- dxy
- fst
- pi\_within
- pi\_between
- valid\_sites
- populations
- model
- estimator
- regionUsed
- pop.info

### Author(s)

Kristian K Ullrich

### See Also

[dnasttring2dist](#), [addpop2string](#)

## Examples

```
## load example sequence data
data("iupac", package="MSA2dist")
#dnastring2dxy(iupac, model="IUPAC")
iupac |> dnastring2dxy(model="IUPAC")
#dnastring2dxy(iupac, model="IUPAC", estimator="count")
iupac |> dnastring2dxy(model="IUPAC", estimator="count")
#dnastring2dxy(iupac, model="sequence")
iupac |> dnastring2dxy(model="sequence")
#dnastring2dxy(iupac, model="sequence", estimator="count")
iupac |> dnastring2dxy(model="sequence", estimator="count")
## 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 <- iupac |> addpop2string(poplist)
#dnastring2dxy(iupac, pop=popinteger(iupac), model="IUPAC")
iupac |> dnastring2dxy(pop=popinteger(iupac), model="IUPAC")
#dnastring2dxy(iupac, pop=popnames(iupac), model="IUPAC")
iupac |> dnastring2dxy(pop=popnames(iupac), model="IUPAC")
#dnastring2dxy(iupac, pop=popinteger(iupac), model="IUPAC",
#estimator="count")
iupac |> dnastring2dxy(pop=popinteger(iupac), model="IUPAC",
estimator="count")
#dnastring2dxy(iupac, pop=popinteger(iupac), model="sequence")
iupac |> dnastring2dxy(pop=popinteger(iupac), model="sequence")
#dnastring2dxy(iupac, pop=popinteger(iupac), model="sequence",
#estimator="count")
iupac |> dnastring2dxy(pop=popinteger(iupac), model="sequence",
estimator="count")
## create mask
mask1 <- IRanges::IRanges(start=c(1,61,121), end=c(30,90,150))
## use mask
iupac |> dnastring2dxy(model="IUPAC", mask=mask1)
## use region
region1 <- IRanges::IRanges(start=c(1,139), end=c(75,225))
iupac |> dnastring2dxy(model="IUPAC", region=region1)
## use mask and region
iupac |> dnastring2dxy(model="IUPAC", mask=mask1, region=region1)
```

---

dnastring2kaks

*dnastring2kaks*

---

## Description

This function calculates Ka/Ks (pN/pS) for all combinations of a DNAStrngSet. If the sequences in the DNAStrngSet 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)
```

---

dnastring2weightedPi    *dnastring2weightedPi*

---

## Description

This function calculates weightedPi for all population combinations of a DNAStringSet.

**Usage**

```

dnastring2weightedPi(
  dna,
  model = "IUPAC",
  estimator = "count",
  mask = NULL,
  region = NULL,
  pop = NULL
)

```

**Arguments**

|                        |                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>dna</code>       | DNAStringSet [mandatory]                                                                                                                                                                                                                                                                                                                                          |
| <code>model</code>     | specify model either "IUPAC" (diploid; decode IUPAC) or "sequence" (haploid; only A,C,T,G counted as discrete alleles) [default: IUPAC]                                                                                                                                                                                                                           |
| <code>estimator</code> | specify estimator either "count" (count-based; computes nucleotide diversity as the observed proportion of pairwise allele differences, $n_{\text{Diff}} / n_{\text{Comp}}$ ) or "prob" (allele-frequency based; computes expected heterozygosity as $1 - \sum p_i^2$ ) [default: count]                                                                          |
| <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>pop</code>       | [default: NULL] optional population assignment: <ul style="list-style-type: none"> <li>• character vector (same length as dna)</li> <li>• named character vector (mapping seqnames to pop)</li> <li>• integer vector (same length as dna)</li> <li>• named integer vector (mapping index to pop)</li> <li>• list (see <code>addpop2dnastring()</code>)</li> </ul> |

**Value**

list containing:

- `weightedPi`
- `populations`
- `model`
- `estimator`
- `regionUsed`
- `pop.info`

**Author(s)**

Kristian K Ullrich

**See Also**

[dnastring2dist](#), [addpop2string](#)

**Examples**

```
## load example sequence data
data("iupac", package="MSA2dist")
#dnastring2weightedPi(iupac, model="IUPAC")
iupac |> dnastring2weightedPi(model="IUPAC")
#dnastring2weightedPi(iupac, model="IUPAC", estimator="prob")
iupac |> dnastring2weightedPi(model="IUPAC", estimator="prob")
#dnastring2weightedPi(iupac, model="sequence")
iupac |> dnastring2weightedPi(model="sequence")
#dnastring2weightedPi(iupac, model="sequence", estimator="prob")
iupac |> dnastring2weightedPi(model="sequence", estimator="prob")
## 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 <- iupac |> addpop2string(poplist)
#dnastring2weightedPi(iupac, pop=popinteger(iupac), model="IUPAC")
iupac |> dnastring2weightedPi(pop=popinteger(iupac), model="IUPAC")
#dnastring2weightedPi(iupac, pop=popnames(iupac), model="IUPAC")
iupac |> dnastring2weightedPi(pop=popnames(iupac), model="IUPAC")
#dnastring2weightedPi(iupac, pop=popinteger(iupac), model="IUPAC",
#estimator="prob")
iupac |> dnastring2weightedPi(pop=popinteger(iupac), model="IUPAC",
estimator="prob")
#dnastring2weightedPi(iupac, pop=popinteger(iupac), model="sequence")
iupac |> dnastring2weightedPi(pop=popinteger(iupac), model="sequence")
#dnastring2weightedPi(iupac, pop=popinteger(iupac), model="sequence",
#estimator="prob")
iupac |> dnastring2weightedPi(pop=popinteger(iupac), model="sequence",
estimator="prob")
## create mask
mask1 <- IRanges::IRanges(start=c(1,61,121), end=c(30,90,150))
## use mask
iupac |> dnastring2weightedPi(model="IUPAC", mask=mask1)
## use region
region1 <- IRanges::IRanges(start=c(1,139), end=c(75,225))
iupac |> dnastring2weightedPi(model="IUPAC", region=region1)
## use mask and region
iupac |> dnastring2weightedPi(model="IUPAC", mask=mask1, region=region1)
```

---

GENETIC\_CODE\_TCAG

*GENETIC\_CODE\_TCAG*

---

**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

*getmask*

---

**Description**

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

**Usage**

```
getmask(seq)
```

**Arguments**

seq                      `DNAStringSet` 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

*globalDeletion*

---

**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 | <i>globalDeletionAA</i> |
|------------------|-------------------------|

---

### 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 | <i>granthamMatrix</i> |
|----------------|-----------------------|

---

### 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) `DNASTringSet`.

### Usage

```
data(iupac)
```

### Format

an object of class `DNASTringSet` 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 | <i>makePostalignedSeqs</i> |
|---------------------|----------------------------|

---

**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))))
```

---

normpop

*normpop*


---

**Description**

This function normalizes population representation

**Usage**

```
normpop(pop, seqnames)
```

**Arguments**

|          |                                                                                                                                                                                                                                                                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pop      | [mandatory] population assignment: <ul style="list-style-type: none"> <li>• character vector (same length as seqnames)</li> <li>• named character vector (mapping seqnames to pop)</li> <li>• integer vector (same length as seqnames)</li> <li>• named integer vector (mapping index to pop)</li> <li>• list (see <code>addpop2dnastring()</code>)</li> </ul> |
| seqnames | sequence names [mandatory]                                                                                                                                                                                                                                                                                                                                     |

**Value**

list containing:

- pop\_idx
- pop\_name
- pop\_levels
- pop\_lookup
- poplist.names
- poplist.integer

**Author(s)**

Kristian K Ullrich

**See Also**[addpop2string](#)**Examples**

```
## load example sequence data
data("iupac", package="MSA2dist")
## 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)))
normpop(pop=poplist, seqnames=names(iupac))
## 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)))
normpop(pop=poplist, seqnames=names(iupac))
## 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))])
normpop(pop=poplist, seqnames=names(iupac))
## create character vector
popchar <- c(rep("FRA", 8),
             rep("GER", 8),
             rep("IRA", 8),
             rep("AFG", 6))
normpop(pop=popchar, seqnames=names(iupac))
## create named character vector
popcharn <- c(rep("FRA", 8),
              rep("GER", 8),
              rep("IRA", 8),
              rep("AFG", 6))
names(popcharn) <- names(iupac)
normpop(pop=popcharn, seqnames=names(iupac))
## create integer vector
popint <- c(rep(1L, 8),
            rep(2L, 8),
            rep(3L, 8),
            rep(4L, 6))
normpop(pop=popint, seqnames=names(iupac))
## create named integer vector
popintn <- c(rep(1L, 8),
             rep(2L, 8),
             rep(3L, 8),
             rep(4L, 6))
names(popintn) <- names(iupac)
normpop(pop=popintn, seqnames=names(iupac))
```

---

pal2nal

*pal2nal*


---

## Description

This function takes an `AAStringSet` alignment and its corresponding coding sequences `DNASTringSet` 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>DNASTringSet</code> [mandatory]                                     |
| <code>remove.gaps</code> | 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)
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

*popinteger*

---

## Description

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

## Usage

```
popinteger(seq)
```

## Arguments

seq                      DNASTringSet 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

*popnames*

---

## Description

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

## Usage

```
popnames(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(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 containing:

- distSTRING
- sitesUsed

## Author(s)

Kristian K Ullrich

## Examples

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

---

|                  |                         |
|------------------|-------------------------|
| rcpp_dxy_fst_pop | <i>rcpp_dxy_fst_pop</i> |
|------------------|-------------------------|

---

**Description**

calculates dxy and fst per population combination

**Usage**

```
rcpp_dxy_fst_pop(dnavevector, pop_idx, model = "IUPAC", estimator = "prob")
```

**Arguments**

|             |                           |
|-------------|---------------------------|
| dnavevector | StringVector [mandatory]  |
| pop_idx     | IntegerVector [mandatory] |
| model       | string [default: IUPAC]   |
| estimator   | string [default: count]   |

**Value**

list containing:

- dxy
- fst
- pi\_within
- pi\_between
- populations
- model
- estimator

**Author(s)**

Kristian K Ullrich

**Examples**

```
## load example sequence data
data("iupac", package="MSA2dist")
rcpp_dxy_fst_pop(dnavevector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="IUPAC",
  estimator="count")
rcpp_dxy_fst_pop(dnavevector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="IUPAC",
  estimator="prob")
```

```
rcpp_dxy_fst_pop(dnavector=as.character(iupac),
pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
model="sequence",
estimator="count")
rcpp_dxy_fst_pop(dnavector=as.character(iupac),
pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
model="sequence",
estimator="prob")
```

rcpp\_KaKs

*rcpp\_KaKs*

## 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 containing:

- sitesUsed

**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 containing:

- sitesUsed

**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)
```

---

|                 |                        |
|-----------------|------------------------|
| rcpp_weightedPi | <i>rcpp_weightedPi</i> |
|-----------------|------------------------|

---

**Description**

calculates weighted pi

**Usage**

```
rcpp_weightedPi(dnavector, model = "IUPAC", estimator = "count")
```

**Arguments**

|           |                          |
|-----------|--------------------------|
| dnavector | StringVector [mandatory] |
| model     | string [default: IUPAC]  |
| estimator | string [default: count]  |

**Value**

list containing:

- weightedPi
- model
- estimator

**Author(s)**

Kristian K Ullrich

**Examples**

```
## load example sequence data
data("hiv", package="MSA2dist")
rcpp_weightedPi(dnavector=as.character(hiv), model="IUPAC",
  estimator="count")
rcpp_weightedPi(dnavector=as.character(hiv), model="sequence",
  estimator="count")
rcpp_weightedPi(dnavector=as.character(hiv), model="IUPAC",
  estimator="prob")
rcpp_weightedPi(dnavector=as.character(hiv), model="sequence",
  estimator="prob")
```

---

|                     |                            |
|---------------------|----------------------------|
| rcpp_weightedPi_pop | <i>rcpp_weightedPi_pop</i> |
|---------------------|----------------------------|

---

**Description**

calculates weighted pi per population

**Usage**

```
rcpp_weightedPi_pop(dnavector, pop_idx, model = "IUPAC", estimator = "count")
```

**Arguments**

|           |                           |
|-----------|---------------------------|
| dnavector | StringVector [mandatory]  |
| pop_idx   | IntegerVector [mandatory] |
| model     | string [default: IUPAC]   |
| estimator | string [default: count]   |

**Value**

list containing:

- weightedPi
- populations
- model
- estimator

**Author(s)**

Kristian K Ullrich

## Examples

```
## load example sequence data
data("iupac", package="MSA2dist")
rcpp_weightedPi_pop(dnavector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="IUPAC",
  estimator="count")
rcpp_weightedPi_pop(dnavector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="IUPAC",
  estimator="prob")
rcpp_weightedPi_pop(dnavector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="sequence",
  estimator="count")
rcpp_weightedPi_pop(dnavector=as.character(iupac),
  pop_idx=c(rep(1,8),rep(2,8),rep(3,8),rep(4,6)),
  model="sequence",
  estimator="prob")
```

---

|        |
|--------|
| region |
|--------|

---

*region*

## 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, [5](#)
- hiv-data, [47](#)
- iupac-data, [50](#)
  
- aa2selfscore, [3](#)
- aabin2aastring, [4](#)
- AAMatrix (AAMatrix-data), [5](#)
- AAMatrix-data, [5](#)
- aastring2aabin, [6](#)
- aastring2aln, [7](#)
- aastring2dist, [8](#), [47](#)
- AAStringSet, [4](#), [14](#)
- addmask2string, [9](#), [11–13](#)
- addpop2string, [10](#), [10](#), [12](#), [13](#), [29](#), [37](#), [42](#), [44](#), [53](#), [55](#), [56](#), [63](#), [64](#)
- addpos2string, [10](#), [11](#), [12](#), [13](#)
- addregion2string, [10–12](#), [13](#)
- aln2aastring, [14](#)
- aln2dnastring, [15](#)
- as.alignment, [4](#), [6](#), [7](#), [14](#), [15](#), [30](#), [31](#), [34](#)
- as.DNABin.alignment, [4](#), [6](#), [30](#), [34](#)
  
- cds2aa, [16](#)
- cds2codonaln, [17](#), [51](#)
- cdsstring2codonaln, [18](#)
- codon2number (GENETIC\_CODE\_TCAG), [42](#)
- codon2numberAMBIG, [20](#)
- codon2numberTCAG, [20](#)
- codonmat2pnps, [21](#), [23](#)
- codonmat2xy, [22](#)
- collapseChar, [24](#)
- collapseXStringSet, [24](#)
- compareCodons, [25](#)
  
- dist.dna, [33](#), [47](#), [51](#)
- dist2dxy, [26](#)
- dnabin2dnastring, [29](#)
- dnastring2aln, [30](#)
- dnastring2codonmat, [22](#), [23](#), [31](#)
- dnastring2dist, [8](#), [29](#), [32](#), [37](#), [42](#), [51](#), [65](#)
- dnastring2dnabin, [34](#)
- dnastring2dxy, [35](#)
- dnastring2kaks, [23](#), [38](#)
- dnastring2weightedPi, [40](#)
- DNAStrngSet, [15](#), [30](#)
  
- GENETIC\_CODE, [20](#), [21](#), [43](#)
- GENETIC\_CODE\_TCAG, [42](#)
- getmask, [43](#)
- getpos, [44](#)
- globalDeletion, [45](#)
- globalDeletionAA, [46](#)
- granthamMatrix, [46](#)
  
- hiv (hiv-data), [47](#)
- hiv-data, [47](#)
  
- indices2kaks, [48](#)
- iupac (iupac-data), [50](#)
- iupac-data, [50](#)
- iupacMatrix, [50](#)
  
- kaks, [22](#), [23](#), [26](#), [40](#), [49](#)
  
- makePostalignedSeqs, [51](#)
  
- normpop, [52](#)
  
- pairwiseAlignment, [17–19](#), [51](#), [54](#)
- pal2nal, [54](#)
- popinteger, [55](#)
- popnames, [56](#)
  
- rcpp\_distSTRING, [57](#)
- rcpp\_dxyfst\_pop, [58](#)
- rcpp\_KaKs, [59](#)
- rcpp\_pairwiseDeletionAA, [60](#)
- rcpp\_pairwiseDeletionDNA, [60](#)
- rcpp\_weightedPi, [61](#)
- rcpp\_weightedPi\_pop, [62](#)

region, [63](#)  
regionused, [64](#)  
  
string2region, [65](#)  
subseq, [66](#)  
substitution\_matrices, [4](#)  
subString, [66](#)  
  
translate, [16](#)  
  
uptriidx, [67](#)