Package 'GenomAutomorphism'

November 12, 2026

Title Compute the automorphisms between DNA's Abelian group representations

Version 1.14.1

URL https://github.com/genomaths/GenomAutomorphism

BugReports https://github.com/genomaths/GenomAutomorphism/issues

Description This is a R package to compute the automorphisms between pairwise aligned DNA sequences represented as elements from a Genomic Abelian group. In a general scenario, from genomic regions till the whole genomes from a given population (from any species or close related species) can be algebraically represented as a direct sum of cyclic groups or more specifically Abelian p-groups. Basically, we propose the representation of multiple sequence alignments of length N bp as element of a finite Abelian group created by the direct sum of homocyclic Abelian group of prime-power order.

Depends R (>= 4.6.0),

License Artistic-2.0

Encoding UTF-8

biocViews MathematicalBiology, ComparativeGenomics, FunctionalGenomics, MultipleSequenceAlignment, WholeGenome

Imports Biostrings, BiocGenerics, BiocParallel, Seqinfo, GenomicRanges, IRanges, matrixStats, XVector, dplyr, data.table, parallel, doParallel, foreach, methods, S4Vectors, stats, numbers, utils

Suggests spelling, rmarkdown, BiocStyle, testthat (>= 3.0.0), knitr

Roxygen list(markdown = TRUE)

Language en-US

LazyData false

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

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

git_branch RELEASE_3_23

git_last_commit 155e73d

git_last_commit_date 2026-08-01

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Robersy Sanchez [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5246-1453>)

Maintainer Robersy Sanchez <genomicmath@gmail.com>

Contents

aaindex1 4
aaindex2 5
aaindex3 6
aa_phychem_index 7
aln 8
aminoacid_dist 9
as.AutomorphismList 12
aut3D 13
autby_coef 15
autm 15
autm_3d 16
autm_z125 17
Automorphism-class 17
AutomorphismByCoef-class 18
AutomorphismByCoefList-class 19
automorphismByRanges 20
AutomorphismList-class 21
automorphisms 23
automorphism_bycoef 26
automorphism_prob 27
autZ125 30
autZ5 32
autZ64 33
base2codon 35
base2int 36
BaseGroup-class 39
BaseGroup_OR_CodonGroup-class 39
BaseSeq-class 39
BaseSeqMatrix-class 40
base_coord 40
base_repl 44
brca1_aln 44

brca1_aln2 . . . 45
brca1_autm . . . 46
brca1_autm2 . . . 46
cdm_z64 . . . 47
CodonGroup-class . . . 47
CodonMatrix-class . . . 48
CodonSeq-class . . . 49
codon_coord . . . 50
codon_dist . . . 52
codon_dist_matrix . . . 55
codon_matrix . . . 56
ConservedRegion-class . . . 58
conserved_regions . . . 59
covid_aln . . . 61
covid_autm . . . 61
cyc_aln . . . 62
cyc_autm . . . 63
dna_phyche . . . 64
dna_phychem . . . 65
GenomAutomorphism . . . 67
getAutomorphisms . . . 67
get_coord . . . 69
get_mutscore . . . 71
GRangesMatrixSeq-class . . . 74
GRanges_OR_NULL-class . . . 75
is.url . . . 76
ListCodonMatrix-class . . . 76
matrices . . . 77
MatrixList-class . . . 79
MatrixSeq-class . . . 80
mod . . . 81
modeq . . . 82
modlineq . . . 83
mut_type . . . 84
peptide_phychem_index . . . 85
reexports . . . 87
seqranges . . . 89
show,CodonSeq-method . . . 91
slapply . . . 91
sortByChromAndStart . . . 93
str2chr . . . 93
str2dig . . . 94
translation . . . 95
valid.Automorphism.mcols . . . 97
valid.AutomorphismByCoef . . . 97
valid.AutomorphismByCoefList . . . 98
valid.AutomorphismList . . . 98
valid.BaseGroup.elem . . . 99

valid.CodonGroup.mcols 99
valid.MatrixList 100
[,AutomorphismList,ANY-method 100

Index **103**

`aaindex1` *List of 566 Amino Acid Physicochemical Indexes from AAindex Database*

Description

The aminoacid indexes from Amino Acid Index Database https://www.genome.jp/aaindex/ are provided here. AAindex (ver.9.2) is a database of numerical indices representing various physicochemical and biochemical properties of amino acids and pairs of amino acids.

Usage

```
data("aaindex1", package = "GenomAutomorphism")
```

Format

A list carrying the description of the 566 Amino Acid Indices in AAindex ver.9.2 and the text file with the matrices imported from https://www.genome.jp/aaindex/.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

aaindex2 and aaindex3.

Examples

```
## Load the mutation matrices from database from the packages
data("aaindex1", package = "GenomAutomorphism", envir = environment())

## Get the available aminoacid indices.
mat <- aa_phychem_index(aaindex = "aaindex1", acc_list = TRUE)
mat[1:10]
```

aaindex2 *List of 94 Amino Acid Matrices from AAindex*

Description

The aminoacid similarity matrices from Amino Acid Index Database https://www.genome.jp/aaindex/ are provided here. AAindex (ver.9.2) is a database of numerical indices representing various physicochemical and biochemical properties of amino acids and pairs of amino acids.

Usage

```
data("aaindex2", package = "GenomAutomorphism")
```

Format

A list carrying the description of 94 Amino Acid Matrices in AAindex ver.9.2 and the text file of matrices imported from https://www.genome.jp/aaindex/.

Details

The similarity of amino acids can be represented numerically, expressed in terms of observed mutation rate or physicochemical properties. A similarity matrix, also called a mutation matrix, is a set of 210 numerical values, 20 diagonal and 20x19/2 off-diagonal elements, used for sequence alignments and similarity searches.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

aaindex2 and aa_mutmat, and get_mutscore.

Examples

```
## Load the mutation matrices from database from the packages
data("aaindex2", package = "GenomAutomorphism")

## Get the available matrices
mat <- aa_mutmat(aaindex = "aaindex2", acc_list = TRUE)
mat[1:10]
```

aaindex3 *Statistical protein contact potentials matrices from AAindex ver.9.2*

Description

A statistical potential (also knowledge-based potential, empirical potential, or residue contact potential) is an energy function derived from an analysis of known structures in the Protein Data Bank.

Usage

```
data("aaindex3", package = "GenomAutomorphism")
```

Format

A list carrying the description of the 47 Amino Acid Matrices in AAindex ver.9.2 and the text file of matrices imported from https://www.genome.jp/aaindex/.

Details

A list of 47 amino acid matrices from Amino Acid Index Database https://www.genome.jp/aaindex/ are provided here. AAindex is a database of numerical indices representing various physicochemical and biochemical properties of amino acids and pairs of amino acids.

The contact potential matrix of amino acids is a set of 210 numerical values, 20 diagonal and 20x19/2 off-diagonal elements, used for sequence alignments and similarity searches.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

aaindex1, aaindex2, and get_mutscore.

Examples

```
## Load the mutation matrices from database from the packages
data("aaindex3", package = "GenomAutomorphism")

## Get the available mutation matrices
mat <- aa_mutmat(aaindex = "aaindex3", acc_list = TRUE)
mat[1:10]
```

aa_phychem_index *Amino acid mutation matrix*

Description

The aminoacid similarity matrices from Amino Acid Index Database https://www.genome.jp/aaindex/ are provided here. AAindex (ver.9.2) is a database of numerical indices representing various physicochemical and biochemical properties of amino acids and pairs of amino acids.

The similarity of amino acids can be represented numerically, expressed in terms of observed mutation rate or physicochemical properties. A similarity matrix, also called a mutation matrix, is a set of 210 numerical values, 20 diagonal and 20x19/2 off-diagonal elements, used for sequence alignments and similarity searches.

Function ***aa_phychem_index*** is a wrapper function to call two other functions: ***aa_mutmat*** and ***aa_index***

Usage

```
aa_phychem_index(acc = NA, aaindex = NA, acc_list = FALSE, info = FALSE)

aa_mutmat(acc = NA, aaindex = c("aaindex2", "aaindex3"), acc_list = FALSE)

aa_index(acc = NA, acc_list = FALSE, info = FALSE)
```

Arguments

acc	Accession id for a specified mutation or contact potential matrix.
aaindex	Database where the requested accession id is located. The possible values are: "aaindex2" or "aaindex3".
acc_list	Logical. If TRUE, then the list of available matrices ids and index names is returned.
info	Logical. if TRUE, then whole information for the physicochemical index will be returned.

Value

Depending on the user specifications, a mutation or contact potential matrix, a list of available matrices (indices) ids or index names can be returned. More specifically:

aa_mutmat: Returns an aminoacid mutation matrix or a statistical protein contact potentials matrix.

aa_index: Returns the specified aminoacid physicochemical indices.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

aaindex1, aaindex2, aaindex3, and get_mutscore.

Examples

```
## Load the mutation matrices from database from the packages
data("aaindex1","aaindex2", package = "GenomAutomorphism" )

## Get the available mutation matrices
mat <- aa_mutmat(aaindex = "aaindex2", acc_list = TRUE)
mat[seq(10)]

## Return the 'Base-substitution-protein-stability matrix
## (Miyazawa-Jernigan, 1993)'
aa_mutmat(acc = "MIYS930101", aaindex = "aaindex2")

## Return the 'BLOSUM80 substitution matrix (Henikoff-Henikoff, 1992)'
aa_mutmat(acc = "HENS920103", aaindex = "aaindex2")

## Using wrapping function
aa_phychem_index(acc = "EISD840101", aaindex = "aaindex1")

## Just the info. The information provided after the reference
## corresponds to the correlation of 'EISD840101' with other indices.
aa_phychem_index(acc = "EISD840101", aaindex = "aaindex1", info = TRUE)
```

---

aln *Simulated* DNAStringSet *class object*

Description

This is a DNAStringSet carrying a small pairwise DNA sequence alignment to be used in the examples provided for the package functions.

Usage

```
data("aln", package = "GenomAutomorphism")
```

Format

DNAStringSet class object.

Examples

```
data("aln", package = "GenomAutomorphism")
aln
```

aminoacid_dist *Distance Between Aminoacids in Terms of Codon Distance*

Description

This function computes the distance between aminoacids in terms of a statistic of the corresponding codons. The possible statistics are: 'mean', 'median', or some user defined function.

Usage

```
aminoacid_dist(aa1, aa2, ...)

## S4 method for signature 'character,character'
aminoacid_dist(
  aa1,
  aa2,
  weight = NULL,
  stat = c("mean", "median", "user_def"),
  genetic_code = "1",
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'DNAStringSet,ANY'
aminoacid_dist(
  aa1,
  weight = NULL,
  stat = c("mean", "median", "user_def"),
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'AAStringSet,ANY'
aminoacid_dist(
  aa1,
  weight = NULL,
  stat = c("mean", "median", "user_def"),
```

```
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
    "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'CodonGroup_OR_Automorphisms,ANY'
aminoacid_dist(
  aa1,
  weight = NULL,
  stat = c("mean", "median", "user_def"),
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
    "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)
```

Arguments

aa1, aa2	A character string of codon sequences, i.e., sequences of DNA base-triplets. If only 'x' argument is given, then it must be a DNAStringSet-class object.
...	Not in use yet.
weight	A numerical vector of weights to compute weighted Manhattan distance between codons. If $weight = NULL$, then $weight = (1/4, 1, 1/16)$ for $group =$ "$Z4$" and $weight = (1/5, 1, 1/25)$ for $group =$ "$Z5$" (see codon_dist).
stat	The name of some statistical function summarizing data like 'mean', 'median', or some user defined function ('user_def'). If $stat =' user_def'$, then function must have a logical argument named 'na.rm' addressed to remove missing (NA) data (see e.g., mean).
genetic_code	A single string that uniquely identifies the genetic code to extract. Should be one of the values in the id or name2 columns of GENETIC_CODE_TABLE.
group	A character string denoting the group representation for the given codon sequence as shown in reference (2-3).
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the progress bar.

Details

Only aminoacids sequences given in the following alphabet are accepted: "A","R","N","D","C","Q","E","G","H","I","L","K" "M","F","P", "S","T","W","Y","V", "*", "-", and "X"; where symbols "*" and "-" denote the presence of a stop codon and of a gap, respectively, and letter "X" missing information, which are then taken as a gap.

The distance between any aminoacid and any of the non-aminoacid symbols is the ceiling of the greater distance found in the corresponding aminoacid distance matrix.

Value

A numerical vector with the pairwise distances between codons in sequences 'x' and 'y'.

References

1. Sanchez R. Evolutionary Analysis of DNA-Protein-Coding Regions Based on a Genetic Code Cube Metric. Curr. Top. Med. Chem. 2014;14: 407–417. https://doi.org/10.2174/1568026613666131204110022.
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF.

See Also

automorphisms and codon_coord

codon_dist

Examples

```
## Write down to aminoacid sequences
x <- "A*LTHMC"
y <- "AAMTDM-"

aminoacid_dist(aa1 = x, aa2 = y)

## Let's create an AAStringSet-class object
aa <- AAStringSet(c(x, y))

aminoacid_dist(aa1 = aa)

## Let's select cube "GCAT" and group "Z5"
aminoacid_dist(aa1 = aa, group = "Z5", cube = "TCGA")
```

as.AutomorphismList *Methods for AutomorphismList-class Objects*

Description

Several methods are available to be applied on Automorphism-class and AutomorphismList-class objects.

Usage

```
as.AutomorphismList(x, grs = GRanges(), ...)

## S4 method for signature 'GRangesList,GRanges_OR_NULL'
as.AutomorphismList(x, grs = GRanges(), ...)

## S4 method for signature 'list,GRanges_OR_NULL'
as.AutomorphismList(x, grs = GRanges(), ...)
```

Arguments

x	A DataFrame or a automorphisms class object.
grs	A GRanges-class object.
...	Not in use yet.

Value

An AutomorphismList-class object.

See Also

automorphism_bycoef, automorphisms

Examples

```
## Load a dataset
data("brca1_autm", package = "GenomAutomorphism")

## Let's transform it into a list of Automorphisms-class objects
x1 <- as.list(brca1_autm[seq(2)])

## Now, object 'x1' is transformed into an AutomorphismList-class object
as.AutomorphismList(x1)

## Alternatively, let's transform the list 'x1' into a GRangesList-class
## object.
x1 <- GRangesList(x1)

## Next, object 'x1' is transformed into an AutomorphismList-class object
as.AutomorphismList(x1)
```

aut3D *Compute the Automorphisms of Mutational Events Between two Codon Sequences Represented in Z5^3.*

Description

Given two codon sequences represented in the Z5^3 Abelian group, this function computes the automorphisms describing codon mutational events.

Usage

```
aut3D(
  seq = NULL,
  filepath = NULL,
  cube = c("ACGT", "TGCA"),
  cube_alt = c("CATG", "GTAC"),
  field = "GF5",
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+",
  genetic_code = getGeneticCode("1"),
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

seq	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences. The pairwise alignment provided in argument **seq** or the 'fasta' file **filepath** must correspond to codon sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube, cube_alt	A character string denoting pairs of the 24 Genetic-code cubes, as given in references (2-3). That is, the base pairs from the given cubes must be complementary to each other. Such a pair of cubes is called a dual cube pair and, as shown in reference (3), each pair integrates a group.
field	A character string denoting the Galois field where the 3D automorphisms are estimated. This can be 'GF(4)' or 'GF(5)', but only 'GF(5)' is implemented so far.
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.

genetic_code — The named character vector returned by getGeneticCode or similar. The translation of codon into aminoacids is valuable information useful for downstream statistical analysis. The standard genetic code is the default argument value applied in the translation of codons into aminoacids (see GENETIC_CODE_TABLE).

num.cores, tasks — Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).

verbose — If TRUE, prints the progress bar.

Details

Automorphisms in Z5^3 are described as functions $f(x) = AxmodZ5$, where A is a diagonal matrix, as noticed in reference (4).

Value

An object Automorphism-class with four columns on its metacolumn named: *seq1*, *seq2*, *autm*, and *cube*.

Author(s)

Robersy Sanchez (https://genomaths.com).

References

1. Sanchez R, Morgado E, Grau R. Gene algebra from a genetic code algebraic structure. J Math Biol. 2005 Oct;51(4):431-57. doi: 10.1007/s00285-005-0332-8. Epub 2005 Jul 13. PMID: 16012800. (PDF).
2. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. https://doi.org/10.1101/2021.06.01.446543.
3. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
4. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Automorphism on Z5^3
autms <- aut3D(seq = aln)
autms
```

autby_coef *Automorphisms between DNA Primate BRCA1 Genes Grouped by Coefficients*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of primate BRCA1 DNA repair genes grouped by automorphism's coefficients. The grouping derives from the dataset brca1_autm after applying function automorphism_bycoef.

Usage

```
data("autby_coef", package = "GenomAutomorphism")
```

Format

AutomorphismByCoefList class object.

Examples

```
## Load the data set
data("autby_coef", package = "GenomAutomorphism")
autby_coef

## Mutation type found in the data
unique(autby_coef$human_1.human_2$mut_type)
```

autm *Automorphisms between DNA Sequences from two COVID-19 genomes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the SARS coronavirus GZ02 (GenBank: AY390556.1: 265-13398_13398-21485) and Bat SARS-like coronavirus isolate bat-SL-CoVZC45 (GenBank: MG772933.1:265-13455_13455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named covid_aln and the automorphisms were estimated with function autZ64.

Usage

```
data("autm", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Details

The alignment of these DNA sequences is available at: https://github.com/genomaths/seqalignments/raw/master/COVID-19 in the fasta file 'AY390556.1_265-13398_13398-21485_RNA-POL_SARS_COVI_GZ02.fas'

Examples

```
data("autm", package = "GenomAutomorphism")
autm
```

---

autm_3d *Automorphisms between DNA Sequences from two COVID-19 genomes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the SARS coronavirus GZ02 (GenBank: AY390556.1: 265-13398_13398-21485) and Bat SARS-like coronavirus isolate bat-SL-CoVZC45 (GenBank: MG772933.1:265-13455_13455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named covid_aln and the automorphisms were estimated with function aut3D.

Usage

```
data("autm_3d", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Examples

```
data("autm_3d", package = "GenomAutomorphism")
autm_3d
```

autm_z125 — *Automorphisms between DNA Sequences from two COVID-19 genomes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the SARS coronavirus GZ02 (GenBank: AY390556.1: 265-13398_13398-21485) and Bat SARS-like coronavirus isolate bat-SL-CoVZC45 (GenBank: MG772933.1:265-13455_13455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named covid_aln and the automorphisms were estimated with function autZ125.

Usage

```
data("autm_z125", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Examples

```
data("autm_z125", package = "GenomAutomorphism")
autm_z125
```

Automorphism-class — *A class definition to store codon automorphisms in a given Abelian group representation.*

Description

Two classes are involved in to storing codon automorphisms: **Automorphism-class** and **AutomorphismList-class**.

Details

An **Automorphism-class** object has six columns: "seq1", "seq2","coord1", "coord2", "autm", and "cube". See the examples for function `automorphisms`. Observe that as the **Automorphism-class** inherits from `GRanges-class` the transformation starting from a `GRanges-class` object into an **Automorphism-class** is straightforward.

However, the transformation starting from a `data.frame` or a `DataFrame-class` object "x" requires for the creation of an additional `GRanges-class` object, which by default will have the argument seqnames = "1", strand = "+", start/end = seq(row(x)), length = nrow(x). These details must be keep in mind to prevent fundamental errors in the downstream analyses.

Value

Given the slot values, defines an Automorphism-class object.

Automorphism-class methods

as(from, "Automorphism")::

Permits the transformation of a data.frame or a DataFrame-class object into **Automorphism-class** object if the proper columns are provided.

Methods from GRanges-class can also be applied.

See Also

AutomorphismByCoef-class and AutomorphismList-class

AutomorphismByCoef-class

A class definition to store conserved gene/genomic regions found in an MSA.

Description

Objects from this class are generated by function automorphism_bycoef.

Value

Given the slot values, defines an AutomorphismByCoef-class object.

AutomorphismByCoefList-class methods

unlist(x)::

It transforms an AutomorphismByCoefList-class object into an AutomorphismByCoef-class object.

as(x, "AutomorphismByCoefList"):

It transforms a 'list' of AutomorphismByCoef-class object into an AutomorphismByCoefList-class object.

See Also

automorphism_bycoef

AutomorphismByCoefList-class and Automorphism-class

Examples

```
## Let's transform an AutomorphismByCoefList-class object into an
## AutomorphismByCoef-class object
data("autby_coef")
unlist(autby_coef[1:2])

## Herein a 'list' object of AutomorphismByCoef-class objects
lista <- list(human = autby_coef[[1]], gorilla = autby_coef[[2]])

## Let's transform the last list 'lista' into an
## AutomorphismByCoefList-class object
aut <- as(lista, "AutomorphismByCoefList")
aut

## Let's get the element names from object 'aut'
names(aut)

## Let's assign new names
names(aut) <- c("human_1", "gorilla_1")
names(aut)
```

AutomorphismByCoefList-class

A class definition for a list of AutomorphismByCoef class objects.

Description

A class definition for a list of AutomorphismByCoef class objects.

Details

AutomorphismByCoefList-class has the following methods:

as('from', "AutomorphismByCoefList"):
Where 'from' is a list of **AutomorphismByCoef-class**.

unlist(x):
Where 'x' is an **AutomorphismByCoefList-class** object.

Value

Given the slot values, defines an `AutomorphismByCoefList-class` object.

See Also

`AutomorphismByCoef-class` and `AutomorphismList-class`

`automorphismByRanges` *Get the automorphisms by ranges.*

Description

Automorphisms estimated on a pairwise or an MSA alignment can be grouped by ranges which inherits from `GRanges-class` or a `GRanges-class`.

Usage

```
automorphismByRanges(x, ...)

## S4 method for signature 'Automorphism'
automorphismByRanges(x)

## S4 method for signature 'AutomorphismList'
automorphismByRanges(
  x,
  min.len = 0L,
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

`x`	An AutomorphismList-class object returned by function `automorphisms`.
`...`	Not in use.
`min.len`	Minimum length of a range to be reported.
`num.cores, tasks`	Integers. Argument *num.cores* denotes the number of cores to use, i.e. at most how many child processes will be run simultaneously (see `bplapply` function from BiocParallel package). Argument *tasks* denotes the number of tasks per job. Value must be a scalar integer >= 0L. In this documentation a job is defined as a single call to a function, such as `bplapply`. A task is the division of the X argument into chunks. When tasks == 0 (default), X is divided as evenly as possible over the number of workers (see `MulticoreParam` from BiocParallel package).
`verbose`	logic(1). If TRUE, enable progress bar.

Value

A `GRanges-class` or a `GRangesList-class`. Each `GRanges-class` object with a column named *cube*, which carries the type of *cube* automorphisms.

Examples

```
## Load dataset
data("autm", package = "GenomAutomorphism")

automorphismByRanges(x = autm[c(1, 4)])
```

AutomorphismList-class *A class definition to store a list of Automorphism class objects.*

Description

A class definition to store a list of Automorphism class objects derived from the pairwise automorphism estimation from pairwise alignments. Objects from this class are created by function automorphisms and as.AutomorphismList.

Usage

```
## S4 method for signature 'AutomorphismList'
names(x)

## S4 replacement method for signature 'AutomorphismList'
names(x) <- value

## S4 method for signature 'AutomorphismList'
as.list(x)

## S4 method for signature 'AutomorphismList'
show(object)
```

Arguments

x	An AutomorphismList-class object.
value	A character vector naming the elements of the AutomorphismList-class object 'x'.
object	An object from AutomorphismList-class.

Value

An object from AutomorphismList-class.

AutomorphismList-class methods

as.AutomorphismList(x)::

as.AutomorphismList function transform a list of GRanges-class, a GRangesList-class, a list of data.frame or a DataFrame-class objects into an **AutomorphismList-class** object.

unlist(x):

It transforms an AutomorphismList-class object into an Automorphism-class object.

as.list(x):

It transforms a list of Automorphism-class objects into an AutomorphismList-class object.

as(x, "GRangesList"):

It transforms a GRangesList of Automorphism-class objects into an 'AutomorphismList-class' object.

names(x):

To get the element's names from an 'AutomorphismList-class' object.

names(x) <- value:

To assign names to the element from an 'AutomorphismList-class' object.

See Also

Automorphism-class and AutomorphismByCoefList-class.

Examples

```
## Load datasets
data("autm", "brca1_autm")

## Transforming a list of Automorphisms into an AutomorphismList object
lista <- list(human = brca1_autm[[1]], gorilla = brca1_autm[[2]])
as.AutomorphismList(lista)

## Alternatively we can set
aut <- as.list(brca1_autm[seq(2)])
class(aut)

## And reverse it
aut <- as.AutomorphismList(aut)
aut

## Let's get the element names from object 'aut'
names(aut)

## Let's assign new names
names(aut) <- c("human_1", "gorilla_1")
names(aut)

## Transforming a GRangesList of Automorphisms into an AutomorphismList
```

```
## object
lista <- as(lista, "GRangesList")
as.AutomorphismList(lista)

## Transform an AutomorphismList-class object into an Automorphism-class
## object
unlist(brca1_autm[seq(2)])
## Load a DNA sequence alignment
data("brca1_autm", package = "GenomAutomorphism")
names(brca1_autm)
## Load a DNA sequence alignment
data("brca1_autm", package = "GenomAutomorphism")
x1 <- brca1_autm[seq(2)]
names(x1)

## Let's assign a new names
names(x1) <- c("human_1.human_2.0", "human_1.gorilla_0")
names(x1)
## Load a DNA sequence alignment
data("brca1_autm", package = "GenomAutomorphism")

## The list of the first three elements
autm_list <- as.list(brca1_autm[seq(3)])
autm_list
```

`automorphisms` *Compute the Automorphisms of Mutational Events Between two Codon Sequences Represented in a Given Abelian group.*

Description

Given two codon sequences represented in a given Abelian group, this function computes the automorphisms describing codon mutational events. Basically, this function is a wrapping to call the corresponding function for a specified Abelian group.

Usage

```
automorphisms(seqs = NULL, filepath = NULL, group = "Z4", ...)

## S4 method for signature 'DNAStringSet_OR_NULL'
automorphisms(
  seqs = NULL,
  filepath = NULL,
  group = c("Z5", "Z64", "Z125", "Z5^3"),
  cube = c("ACGT", "TGCA"),
  cube_alt = c("CATG", "GTAC"),
  nms = NULL,
  start = NA,
  end = NA,
```

```
  chr = 1L,
  strand = "+",
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

seqs	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences. The pairwise alignment provided in argument **seq** or the 'fasta' file **filepath** must correspond to codon sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
group	A character string denoting the group representation for the given base or codon as shown in reference (1).
...	Not in use.
cube, cube_alt	A character string denoting pairs of the 24 Genetic-code cubes, as given in references (2-3). That is, the base pairs from the given cubes must be complementary to each other. Such a pair of cubes is called a dual cube pair and, as shown in reference (3), each pair integrates a group.
nms	Optional. Only used if the DNA sequence alignment provided carries more than two sequences. A character string giving short names for the alignments to be compared. If not given then the automorphisms between pairwise alignment are named as: 'aln_1', 'aln_2', and so on.
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the progress bar.

Details

Herein, automorphisms are algebraic descriptions of mutational event observed in codon sequences represented on different Abelian groups. In particular, as described in references (3-4), for each representation of the codon set on a defined Abelian group there are 24 possible isomorphic Abelian groups. These Abelian groups can be labeled based on the DNA base-order used to generate them. The set of 24 Abelian groups can be described as a group isomorphic to the symmetric group of degree four (S_4, see reference (4)). Function automorphismByRanges permits the classification of the pairwise alignment of protein-coding sub-regions based on the mutational events observed on it and on the genetic-code cubes that describe them.

Automorphisms in Z5, Z64 and Z125 are described as functions $f(x) = kxmod5$, $f(x) = kxmod64$, and $f(x) = kxmod125$, respectively, where k and x are elements from the set of integers modulo 5,

64, or 125, respectively. If an automorphisms cannot be found on any of the cubes provided in the argument $cube$, then function automorphisms will search for automorphisms in the cubes provided in the argument $cube_alt$.

Automorphisms in Z5^3 are described as functions $f(x) = AxmodZ5$, where A is a diagonal matrix.

Arguments **cube** and **cube_alt** must be pairs of dual cubes (see section 2.4 from reference 4).

Value

This function returns a Automorphism-class object with four columns on its metacolumn named: *seq1*, *seq2*, *autm*, and *cube*.

Methods

automorphismByRanges**::**

This function returns a GRanges-class object. Consecutive mutational events (on the codon sequence) described by automorphisms on a same cube are grouped in a range.

automorphism_bycoef**:**

This function returns a GRanges-class object. Consecutive mutational events (on the codon sequence) described by the same automorphisms coefficients are grouped in a range.

getAutomorphisms**:**

This function returns an AutomorphismList-class object as a list of Automorphism-class objects, which inherit from GRanges-class objects.

conserved_regions**:**

Returns a AutomorphismByCoef class object containing the requested regions.

Author(s)

Robersy Sanchez (https://genomaths.com).

References

1. Sanchez R, Morgado E, Grau R. Gene algebra from a genetic code algebraic structure. J Math Biol. 2005 Oct;51(4):431-57. doi: 10.1007/s00285-005-0332-8. Epub 2005 Jul 13. PMID: 16012800. (PDF).
2. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
3. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
4. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF

See Also

autZ64.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Automorphism on "Z5^3"
autms <- automorphisms(seqs = aln, group = "Z5^3", verbose = FALSE)
autms

## Automorphism on "Z64"
autms <- automorphisms(seqs = aln, group = "Z64", verbose = FALSE)
autms

## Automorphism on "Z64" from position 1 to 33
autms <- automorphisms(
    seqs = aln,
    group = "Z64",
    start = 1,
    end = 33,
    verbose = FALSE
)
autms
```

---

automorphism_bycoef *Automorphism Grouping by Coefficient*

Description

Automorphisms with the same automorphism's coefficients are grouped.

Usage

```
automorphism_bycoef(x, ...)

## S4 method for signature 'Automorphism'
automorphism_bycoef(x, mut.type = TRUE)

## S4 method for signature 'AutomorphismList'
automorphism_bycoef(
  x,
  min.len = 1L,
  mut.type = TRUE,
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

x	An Automorphism-class object returned by function automorphisms.
...	Not in use.
mut.type	Logical. Whether to include the mutation type as given by function mut_type.
min.len	Minimum length of a range to be reported.
num.cores, tasks	Integers. Argument *num.cores* denotes the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply function from BiocParallel package). Argument *tasks* denotes the number of tasks per job. Value must be a scalar integer >= 0L. In this documentation a job is defined as a single call to a function, such as bplapply. A task is the division of the X argument into chunks. When tasks == 0 (default), X is divided as evenly as possible over the number of workers (see MulticoreParam from BiocParallel package).
verbose	logic(1). If TRUE, enable progress bar.

Value

An AutomorphismByCoef-class object when *x* is an Automorphism-class object, or an AutomorphismByCoefList-class object when *x* is an AutomorphismList-class object. A coefficient with 0 value is assigned to mutational events that are not automorphisms, e.g., indel mutations.

See Also

automorphisms

Examples

```
## Load dataset
data("autm", package = "GenomAutomorphism")

automorphism_bycoef(x = autm[1:2])
```

---

automorphism_prob *Automorphism Probability*

Description

This function applies a Dirichlet-Multinomial Modelling (in Bayesian framework) to compute the posterior probability of each *type of mutational event*. DNA bases are classified based on the physicochemical criteria used to ordering the set of codons: number of hydrogen bonds (strong-weak, S-W), chemical type (purine-pyrimidine, Y-R), and chemical groups (amino versus keto, M-K) (see reference 4). Preserved codon positions are labeled with letter "H".

As a result, mutational events are grouped by type of mutation covering all the possible combinations of symbols: "Y", "R", "M", "W", "K", "S", and "H", for example: "YSH", "RSH", "MSH",

"WSH", "KSH", "SSH", "HSH", "YHH", "RHH", and so on. Insertion/deletion mutations are not considered.

Maximum Likelihood Estimation (MLE) for Dirichlet Parameters:

Given the observed data $n = (n_1, ..., n_k)$, where n_i is the frequency for the mutation type i, we want to estimate the parameters of the Dirichlet distribution, $\alpha = (\alpha_1, ..., \alpha_k)$, that maximize the marginal likelihood.

Marginal Likelihood:

The marginal likelihood of the data under the Dirichlet-multinomial model is given by

$$P(n|\alpha) = \frac{N!}{\prod_{i=1}^{k} n_i} \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\Gamma\left(N + \sum_{i=1}^{k} \alpha_i\right)} \prod_{i=1}^{k} \frac{\Gamma(n_i + \alpha_i)}{\Gamma(\alpha_i)}$$

where $N = \sum_{i=1}^{k} n_i$.

Optimization:

To perform MLE, we maximize the log of this likelihood: $log(P(n|\alpha))$. That is, we aim to maximize this log-likelihood with respect to α. This is done numerically because there's no closed-form solution. Here, we use:

$$Arg \max_{\alpha} \{log(P(n|\alpha)\}$$

with initial guess set as $\alpha_i = 1/(n_i + 1)$.

Posterior Distribution:

Let θ be the vector of probabilities for each mutation type. It's the parameter we're estimating, which represents the probabilities of observing each mutation type in the multinomial distribution. The conjugate distribution in this context refers to the Dirichlet distribution, which is the prior distribution for θ. When we have observed data, the posterior distribution of θ given this data is also a Dirichlet distribution due to the conjugate property.

The prior distribution, before observing any data, our belief about the distribution of θ is given by:

$$\theta \sim \ Dirichlet(\alpha_1, \alpha_k, \ldots, \alpha_k)$$

where the α_i are known as initial concentration parameters, which represents our initial belief or assumption about the distribution of the mutation types. These parameters control how the probability mass is distributed among the mutation types. We might set these initial parameters based on some prior knowledge or simply to provide a non-informative prior.

Once we have the MLE for α, the posterior distribution of θ given the data updates these parameters to incorporate the observed data:

$$\theta \mid \ n \sim \ Dirichlet(\alpha_1^* + n1, \alpha_k^* + n_2, \ldots, \alpha_k^* + n_k)$$

where α_i^* are the MLE estimates, i.e., $\alpha_i^* + n_i$ are our updated belief about the frequencies of mutation types in the population sample.

The expected values of θ_i, given the data under the posterior Dirichlet distribution is:

$$E\left[\theta_i\right] = \frac{\alpha_i^* + n_i}{\sum_{j=1}^{k}\left(\alpha_j^* + n_j\right)}$$

These expected values directly corresponds to the "posterior probability" of observing mutation type i.

This approach provides a rigorous estimation of the Dirichlet parameters under the Dirichlet-multinomial model using MLE.

Usage

```
automorphism_prob(x, ...)

## S4 method for signature 'AutomorphismByCoef'
automorphism_prob(
  x,
  initial_alpha = NULL,
  method = c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN", "Brent"),
  maxit = 500,
  abstol = 10^-8,
  ...
)

## S4 method for signature 'AutomorphismByCoefList'
automorphism_prob(
  x,
  initial_alpha = NULL,
  method = c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN", "Brent"),
  maxit = 500,
  abstol = 10^-8
)
```

Arguments

`x`	An AutomorphismByCoefList-class object returned by function automorphism_bycoef.
`...`	Not in use yet.
`initial_alpha`	A vector of initial guess values for pseudo counts α_i (see Description). Default is: $\alpha_i = 1/(n_i + 1)$, which corresponds to $initial_alpha = NULL$.
`method`, `maxit`, `abstol`	Parameter values to pass into `optim`.

Value

A data frame with the posterior probabilities.

See Also

automorphism_bycoef

Examples

```
## Load the data set
data("autby_coef", package = "GenomAutomorphism")
post_prob <- automorphism_prob(autby_coef[1:10])
head(post_prob,10)
```

---

autZ125 *Compute the Automorphisms of Mutational Events Between two Codon Sequences Represented in Z125.*

Description

Given two codon sequences represented in the Z125 Abelian group, this function computes the automorphisms describing codon mutational events.

Usage

```
autZ125(
  seq = NULL,
  filepath = NULL,
  cube = c("ACGT", "TGCA"),
  cube_alt = c("CATG", "GTAC"),
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+",
  genetic_code = getGeneticCode("1"),
  num.cores = multicoreWorkers() - 1,
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

seq	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences. The pairwise alignment provided in argument **seq** or the 'fasta' file **filepath** must correspond to codon sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube, cube_alt	A character string denoting pairs of the 24 Genetic-code cubes, as given in references (2-3). That is, the base pairs from the given cubes must be complementary to each other. Such a pair of cubes is called a dual cube pair and, as shown in reference (3), each pair integrates a group.
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.

genetic_code — The named character vector returned by getGeneticCode or similar. The translation of codon into aminoacids is valuable information useful for downstream statistical analysis. The standard genetic code is the default argument value applied in the translation of codons into aminoacids (see GENETIC_CODE_TABLE).

num.cores, tasks — Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).

verbose — If TRUE, prints the progress bar.

Details

Automorphisms in Z125 are described as functions $f(x) = kxmod125$, where k and x are elements from the set of integers modulo 125. As noticed in reference (1).

Value

An object Automorphism-class with four columns on its metacolumn named: *seq1*, *seq2*, *autm*, and *cube*.

References

1. Sanchez R, Morgado E, Grau R. Gene algebra from a genetic code algebraic structure. J Math Biol. 2005 Oct;51(4):431-57. doi: 10.1007/s00285-005-0332-8. Epub 2005 Jul 13. PMID: 16012800. (PDF).
2. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
3. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
4. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Automorphism on Z125
autms <- autZ125(seq = aln)
autms
```

autZ5 *Compute the Automorphisms of Mutational Events Between two Codon Sequences Represented in Z5.*

Description

Given two codon sequences represented in the Z5 Abelian group, this function computes the automorphisms describing codon mutational events.

Usage

```
autZ5(
  seq = NULL,
  filepath = NULL,
  cube = c("ACGT", "TGCA"),
  cube_alt = c("CATG", "GTAC"),
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+",
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

seq	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube, cube_alt	A character string denoting pairs of the 24 Genetic-code cubes, as given in references (2-3). That is, the base pairs from the given cubes must be complementary to each other. Such a pair of cubes is called a dual cube pair and, as shown in reference (3), each pair integrates a group.
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the progress bar.

Details

Automorphisms in Z5 are described as functions $f(x) = kxmod5$, where k and x are elements from the set of integers modulo 5. As noticed in reference (1). The pairwise alignment provided in argument **seq** or the 'fasta' file **filepath** must correspond to DNA base sequences.

Value

An object `Automorphism-class` with four columns on its metacolumn named: *seq1*, *seq2*, *autm*, and *cube*.

References

1. Sanchez R, Morgado E, Grau R. Gene algebra from a genetic code algebraic structure. J Math Biol. 2005 Oct;51(4):431-57. doi: 10.1007/s00285-005-0332-8. Epub 2005 Jul 13. PMID: 16012800. (PDF).
2. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
3. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
4. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF

See Also

`automorphisms`

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Automorphism on Z5
autms <- autZ5(seq = aln, verbose = FALSE)
autms
```

`autZ64` *Compute the Automorphisms of Mutational Events Between two Codon Sequences Represented in Z64.*

Description

Given two codon sequences represented in the Z64 Abelian group, this function computes the automorphisms describing codon mutational events.

Usage

```
autZ64(
  seq = NULL,
  filepath = NULL,
  cube = c("ACGT", "TGCA"),
  cube_alt = c("CATG", "GTAC"),
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+",
  genetic_code = getGeneticCode("1"),
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

seq	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences. The pairwise alignment provided in argument **seq** or the 'fasta' file **filepath** must correspond to codon sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube, cube_alt	A character string denoting pairs of the 24 Genetic-code cubes, as given in references (2-3). That is, the base pairs from the given cubes must be complementary to each other. Such a pair of cubes is called a dual cube pair and, as shown in reference (3), each pair integrates a group.
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.
genetic_code	The named character vector returned by getGeneticCode or similar. The translation of codon into aminoacids is valuable information useful for downstream statistical analysis. The standard genetic code is the default argument value applied in the translation of codons into aminoacids (see GENETIC_CODE_TABLE).
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the progress bar.

Details

Automorphisms in Z64 are described as functions $f(x) = k * x \ mod\ 64$, where k and x are elements from the set of integers modulo 64.

Value

An object Automorphism-class with four columns on its metacolumn named: *seq1*, *seq2*, *autm*, and *cube*.

Author(s)

Robersy Sanchez (https://genomaths.com).

References

1. Sanchez R, Morgado E, Grau R. Gene algebra from a genetic code algebraic structure. J Math Biol. 2005 Oct;51(4):431-57. doi: 10.1007/s00285-005-0332-8. Epub 2005 Jul 13. PMID: 16012800. (PDF).
2. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
3. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
4. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Automorphism on Z64
autms <- autZ64(seq = aln, verbose = FALSE)
autms
```

---

base2codon *Split a DNA sequence into codons*

Description

This function splits a DNA sequence into a codon sequence.

Usage

```
base2codon(x, ...)

## S4 method for signature 'character'
base2codon(x)

## S4 method for signature 'DNAStringSet'
```

```
base2codon(x)

## S4 method for signature 'DNAMultipleAlignment'
base2codon(x)
```

Arguments

x	A character string, DNAStringSet-class or DNAMultipleAlignment-class object carrying a DNA sequence.
...	Not in use.

Details

It is expected that the provided DNA sequence is a multiple of 3, otherwise gaps are added to the end of the sequence.

Value

If the argument of 'x' is a character string, then a character vector of codons will be returned. If the argument of 'x' is DNAStringSet-class or DNAMultipleAlignment-class object, then a matrix of codons is returned.

Author(s)

Robersy Sanchez https://genomaths.com. 01/15/2022

Examples

```
## Gaps are added at the sequence end.
seq <- c("ACCT")
base2codon(x = seq)

## This DNA sequence is a multiple of 3
seq <- c("ACCTCA")
base2codon(x = seq)

## Load a DNAStringSet. A matrix of codons is returned
data("aln", package = "GenomAutomorphism")
base2codon(x = aln)
```

---

base2int *Replace bases with integers from Z4 and Z5*

Description

A simple function to represent DNA bases as elements from the Abelian group of integers modulo 4 (Z4), 5 (Z5), or 2 (Z2).

Usage

```
base2int(base, ...)

## S4 method for signature 'character'
base2int(
  base,
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3", "Z2"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  phychem = list(A = NULL, T = NULL, C = NULL, G = NULL, N = NULL)
)

## S4 method for signature 'data.frame'
base2int(
  base,
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3", "Z2"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  phychem = list(A = NULL, T = NULL, C = NULL, G = NULL, N = NULL)
)
```

Arguments

base	A character vector, string , or a dataframe of letters from the DNA/RNA alphabet.
...	Not in use.
group	A character string denoting the group representation for the given base or codon as shown in reference (2-3).
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
phychem	Optional. Eventually, it could be useful to represent DNA bases by numerical values of measured physicochemical properties. If provided, then this argument must be a named numerical list. For example, the scale values of deoxyribonucleic acids proton affinity (available at https://www.wolframalpha.com/ and in cell phone app: Wolfram Alpha): $list('A' = 0.87,' C' = 0.88,' T' = 0.82,' G' = 0.89,' N' = NA)$ where symbol 'N' provides the value for any letter out of the DNA base alphabet. In this example, we could write NA or 0 (see example section).

Details

For Z2 (binary representation of DNA bases), the cube bases are represented in their order by: '00', '01', '10', and '11' (examples section).

Value

A numerical vector.

Author(s)

Robersy Sanchez https://genomaths.com

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi: 10.1101/2021.06.01.446543
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

base_coord, codon_coord, and dna_phychem.

Examples

```
## A triplet with a letter not from DNA/RNA alphabet
## 'NA' is introduced by coercion!
base2int("UDG")

## The base replacement in cube "ACGT and group "Z4"
base2int("ACGT")

## The base replacement in cube "ACGT and group "Z5"
base2int("ACGT", group = "Z5")

## A vector of DNA base triplets
base2int(c("UTG", "GTA"))

##  A vector of DNA base triplets with different number of triplets.
##  Codon 'GTA' is recycled!
base2int(base = c("UTGGTA", "CGA"), group = "Z5")

## Data frames

base2int(data.frame(x1 = c("UTG", "GTA"), x2 = c("UTG", "GTA")))


## Cube bases are represented in their order by: '00', '01', '10', and '11',
## For example for cube = "ACGT" we have mapping: A -> '00', C -> '01',
## G -> '10', and T -> '11'.

base2int("ACGT", group = "Z2", cube = "ACGT")
```

BaseGroup-class *A class definition to store codon automorphisms in a given Abelian group representation.*

Description

A class definition to store codon automorphisms in a given Abelian group representation.

Value

Given the slot values, defines a BaseGroup-class object.

See Also

automorphisms

BaseGroup_OR_CodonGroup-class *A definition for the union of classes 'BaseGroup' and 'CodonGroup'*

Description

A definition for the union of classes 'BaseGroup' and 'CodonGroup'

See Also

BaseGroup and CodonGroup.

BaseSeq-class *A class definition to store DNA base sequence.*

Description

A class definition to store DNA base sequence.

Value

Given the slot values, defines a BaseSeq-class object.

See Also

automorphisms

`BaseSeqMatrix-class` *A class definition to store DNA base sequence coordinates in a given Genetic Code Cube.*

Description

A class definition to store DNA base sequence coordinates in a given Genetic Code Cube.

Value

Given the slot values, defines a `BaseSeqMatrix-class` object.

See Also

`automorphisms`

`base_coord` *DNA Sequences Methods*

Description

Base coordinates on a given Abelian group representation:
Given a string denoting a codon or base from the DNA (or RNA) alphabet, function ***base_coord*** returns the base coordinates in the specified genetic-code Abelian group, as given in reference (1).

DNA sequences to `GRanges` of bases.:
Function ***seq2granges*** transform an object from `DNAStringSet`, `DNAMultipleAlignment-class` or a character into an object from BaseSeq.

BaseSeq-class object to DNAStringSet-class object.:
Function ***base_seq2string_set*** transforms an object from BaseSeq into an object from `DNAStringSet-class`.

Usage

```
base_coord(base = NULL, filepath = NULL, cube = "ACGT", group = "Z4", ...)

## S4 method for signature 'DNAStringSet_OR_NULL'
base_coord(
  base = NULL,
  filepath = NULL,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
      "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z4", "Z5"),
  start = NA,
```

```
  end = NA,
  chr = 1L,
  strand = "+"
)

seq2granges(base = NULL, filepath = NULL, ...)

## S4 method for signature 'DNAStringSet_OR_NULL'
seq2granges(
  base = NULL,
  filepath = NULL,
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+",
  seq_alias = NULL,
  ...
)

base_seq2string_set(x, ...)

## S4 method for signature 'BaseSeq'
base_seq2string_set(x)

base_matrix(base, ...)

## S4 method for signature 'DNAStringSet_OR_NULL'
base_matrix(
  base,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
      "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z4", "Z5"),
  seq_alias = NULL
)
```

Arguments

base	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
group	A character string denoting the group representation for the given base or codon as shown in reference (1).
...	Not in use yet.

`start`, `end`, `chr`, `strand`
: Optional parameters required to build a `GRanges-class`. If not provided the default values given for the function definition will be used.

`seq_alias`
: DNA sequence alias/ID and description.

`x`
: A 'BaseSeq' class object.

Details

Function 'base_coord':

Function ***base_coord*** is defined only for pairwise aligned sequences. Symbols "-" and "N" usually found in DNA sequence alignments to denote gaps and missing/unknown bases are represented by the number: '-1' on Z4 and '0' on Z5. In Z64 the symbol 'NA' will be returned for codons including symbols "-" and "N".

Functions 'seq2granges' and 'base_seq2string_set':

For the sake of brevity the metacolumns from the object returned by function 'seq2granges' are named as 'S1', 'S2', 'S3', and so on. The original DNA sequence alias are stored in the slot named 'seq_alias'. (see examples).

Value

Depending on the function called, different object will be returned:

Function 'base_coord':

This function returns a `BaseGroup-class` object carrying the DNA sequence(s) and their respective coordinates in the requested Abelian group of base representation (one-dimension, "Z4" or "Z5"). Observe that to get coordinates in the set of integer numbers ("Z") is also possible but they are not defined to integrate an Abelian group. These are just used for the further insertion the codon set in the 3D space (R^3).

Function 'seq2granges':

This function returns a BaseGroup object carrying the DNA sequence(s), one base per ranges. A BaseGroup object inherits from `GRanges-class`.

Function 'base_seq2string_set':

This function returns a `DNAStringSet-class`.

A `BaseGroup-class` object.

Author(s)

Robersy Sanchez `https://genomaths.com`

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.

3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

Symmetric Group of the Genetic-Code Cubes.

codon_coord and base2int.

base_coord and codon_coord.

Examples

```
## Example 1. Let's get the base coordinates for codons "ACG"
## and "TGC":
x0 <- c("ACG", "TGC")
x1 <- DNAStringSet(x0)
x1

## Get the base coordinates on cube = "ACGT" on the Abelian group = "Z4"
base_coord(x1, cube = "ACGT", group = "Z4")

## Example 2. Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## DNA base representation in the Abelian group Z4
bs_cor <- base_coord(
    base = aln,
    cube = "ACGT"
)
bs_cor

## Example 3. DNA base representation in the Abelian group Z5
bs_cor <- base_coord(
    base = aln,
    cube = "ACGT",
    group = "Z5"
)
bs_cor

## Example 4. Load a multiple sequence alignment (MSA) of primate BRCA1 DNA
## repair genes
data("brca1_aln2", package = "GenomAutomorphism")
brca1_aln2

## Get BaseSeq-class object
gr <- seq2granges(brca1_aln2)
gr

## Transform the BaseSeq-class object into a DNAStringSet-class object
str_set <- base_seq2string_set(gr)
str_set
```

```
## Recovering the original MSA
DNAMultipleAlignment(as.character(str_set))

## Example 5.
base_matrix(base = aln, cube = "CGTA", group = "Z5")

## Example 5.
```

`base_repl` *Replace bases with integers*

Description

Replace bases with integers

Usage

```
base_repl(base, cube, group, phychem)
```

Details

Internal use only.

Value

A numerical vector.

`brca1_aln` *Multiple Sequence Alignment (MSA) of Primate BRCA1 DNA repair genes.*

Description

This is a `DNAMultipleAlignment` carrying an MSA of BRCA1 DNA repair genes to be used in the examples provided for the package functions. The original file can be downloaded from GitHub at: https://bit.ly/3DimROD

Usage

```
data("brca1_aln", package = "GenomAutomorphism")
```

Format

`DNAMultipleAlignment` class object.

See Also

brca1_aln2, brca1_autm, and covid_aln.

Examples

```
data("brca1_aln", package = "GenomAutomorphism")
brca1_aln
```

`brca1_aln2` *Multiple Sequence Alignment (MSA) of Primate BRCA1 DNA repair genes.*

Description

This is a `DNAMultipleAlignment` carrying an MSA of BRCA1 DNA repair genes to be used in the examples provided for the package functions. The original file can be downloaded from GitHub at: https://bit.ly/3DimROD. This data set has 41 DNA sequences and it contains the previous 20 primate variants found in 'brca1_aln' data set plus 21 single mutation variants (SMV) from the human sequence NM_007298 transcript variant 4. The location of each SMV is given in the heading from each sequence.

Usage

```
data("brca1_aln2", package = "GenomAutomorphism")
```

Format

`DNAMultipleAlignment` class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_aln, brca1_autm2, cyc_aln, and covid_autm.

Examples

```
data("brca1_aln2", package = "GenomAutomorphism")
brca1_aln2
```

brca1_autm *Automorphisms between DNA Sequences from Primate BRCA1 Genes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of primate BRCA1 DNA repair genes. The automorphisms were estimated from the brca1_aln MSA with function autZ64.

Usage

```
data("brca1_autm", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_autm2, brca1_aln, brca1_aln2, and covid_autm.

Examples

```
data("brca1_autm", package = "GenomAutomorphism")
brca1_autm
```

brca1_autm2 *Automorphisms between DNA Sequences from Primate BRCA1 Genes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of primate BRCA1 DNA repair genes. The data set brca1_aln2 has 41 DNA sequences and it contains the previous 20 primate variants found in 'brca1_aln' data set plus 21 single mutation variants (SMV) from the human sequence NM_007298 transcript variant 4. The location of each SMV is given in the heading from each sequence. The automorphisms were estimated from the brca1_aln2 MSA with function autZ64.

Usage

```
data("brca1_autm2", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

---

cdm_z64 *Codon Distance Matrices for the Standard Genetic Code on Z4*

Description

This is a list of 24 codon distance matrices created with function codon_dist_matrix in the set of 24 genetic-code cubes on Z4 (using the default weights and assuming the standard genetic code (SGC)). The data set is created to speed up the computation when working with DNA sequences from superior organisms. Since distance matrices are symmetric, it is enough to provide the lower matrix. Each matrix is given as named/labeled vector (see the example).

Usage

```
data("cdm_z64", package = "GenomAutomorphism")
```

Format

A list object.

Examples

```
## Load the data set
data("cdm_z64", package = "GenomAutomorphism")
cdm_z64

## The lower matrix (given as vector) for cube "TCGA" (picking out the 20
## first values). Observe that this vector is labeled. Each numerical value
## corresponds to the distance between the codons specified by the
## name/label on it. For example, the distance between codons TTT and TCT
## is: 1.

head(cdm_z64[[ "TCGA" ]], 20)
```

CodonGroup-class *A class definition to store codon automorphisms in a given Abelian group representation.*

Description

A class definition to store codon automorphisms in a given Abelian group representation.

Value

Given the slot values, defines a CodonGroup-class object.

See Also

automorphisms

CodonMatrix-class *A Convenient Class to Store a Codon Coordinate in given Genetic Code cube.*

Description

A CodonMatrix is the object created by function codon_matrix

Usage

```
CodonMatrix(object, group, cube, seq_alias = NULL)
```

Arguments

object	A GRanges-class object.
group	The name of the base group, 'Z4' or 'Z5'.
cube	The name of the genetic-code cube applied to get the codon coordinates.
seq_alias	The 'alias' given to the codon sequence.

Value

A CodonMatrix-class object.

See Also

base_coord and codon_coord.

Examples

```
## CodonMatrix is generated by function 'codon_matrix' (inside a
## ListCodonMatrix-class object)
## Let's create DNAStringSet-class object
base <- DNAStringSet(x = c(S1 = 'ACGTGATCAAGT'))

x1 <- codon_matrix(base)
x1

## Extract the first element
x1[1]
x1$codon.1
x1[[1]]
```

CodonSeq-class *A class definition to store codon coordinates given in the Abelian group and the codon sequence.*

Description

An object from 'CodonSeq' or 'MatrixList' class is returned by function get_coord. This object will store the coordinate of each sequence in a list of 3D-vectors or a list of vectors located in the slot named 'CoordList'. The original codon sequence (if provided) will be stored in the slot named 'SeqRanges'.

Usage

```
coordList(x)

## S4 method for signature 'CodonSeq'
coordList(x)

seqRanges(x)

## S4 method for signature 'CodonSeq'
seqRanges(x)
```

Arguments

x	An object from CodonSeq-class.

Value

Given the slot values, defines a CodonSeq-class object.

Examples

```
## Load a DNA sequence alignment
data("aln", package = "GenomAutomorphism")

## Get base coordinates on 'Z5'
coord <- get_coord(
    x = aln,
    cube = "ACGT",
    group = "Z5"
)
coordList(coord)
## Load a DNA sequence alignment
data("aln", package = "GenomAutomorphism")

## Get base coordinates on 'Z5'
coord <- get_coord(
    x = aln,
```

```
    cube = "ACGT",
    group = "Z5"
)

seqRanges(coord)
```

---

codon_coord *Codon coordinates on a given Abelian group representation.*

Description

Given a string denoting a codon or base from the DNA (or RNA) alphabet and a genetic-code Abelian group as given in reference (1).

Usage

```
codon_coord(codon = NULL, ...)

## S4 method for signature 'BaseGroup'
codon_coord(codon, group = NULL)

## S4 method for signature 'DNAStringSet_OR_NULL'
codon_coord(
  codon = NULL,
  filepath = NULL,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
   "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
    "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3"),
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+"
)

## S4 method for signature 'matrix_OR_data_frame'
codon_coord(
  codon,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
   "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
    "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z64", "Z125", "Z4^3", "Z5^3")
)
```

Arguments

codon	An object from BaseGroup-class (generated with function base_coord), DNAStringSet or from DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences.

...	Not in use.
group	A character string denoting the group representation for the given base or codon as shown in reference (2-3).
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.

Details

Symbols "-" and "N" usually found in DNA sequence alignments to denote gaps and missing/unknown bases are represented by the number: '-1' on Z4 and '0' on Z5. In Z64 the symbol 'NA' will be returned for codons including symbols "-" and "N".

This function returns a GRanges-class object carrying the codon sequence(s) and their respective coordinates in the requested Abelian group or simply, when *group = 'Z5^3'* 3D-coordinates, which are derive from Z5 as indicated in reference (3). Notice that the coordinates can be 3D or just one-dimension ("Z64" or "Z125"). Hence, the pairwise alignment provided in argument **codon** must correspond to codon sequences.

Value

A CodonGroup-class object.

Author(s)

Robersy Sanchez https://genomaths.com

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi: 10.1101/2021.06.01.446543
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

Symmetric Group of the Genetic-Code Cubes.

codon_matrix, base_coord and base2int.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## DNA base representation in the Abelian group Z5
bs_cor <- codon_coord(
    codon = aln,
    cube = "ACGT",
    group = "Z5"
)
bs_cor ## 3-D coordinates


## DNA base representation in the Abelian group Z64
bs_cor <- codon_coord(
    codon = aln,
    cube = "ACGT",
    group = "Z64"
)
bs_cor

## Giving a matrix of codons
codon_coord(base2codon(x = aln))
```

---

codon_dist *Weighted Manhattan Distance Between Codons*

Description

This function computes the weighted Manhattan distance between codons from two sequences as given in reference (1). That is, given two codons x and y with coordinates on the set of integers modulo 5 ("Z5"): $x = (x_1, x_2, x_3)$ and $y = (y_1, y_2, y_3)$ (see (1)), the Weighted Manhattan distance between these two codons is defined as:

$$d_w(x, y) = |x_1 - y_1|/5 + |x_2 - y_2| + |x_3 - y_3|/25$$

If the codon coordinates are given on "Z4", then the Weighted Manhattan distance is defined as:

$$d_w(x, y) = |x_1 - y_1|/4 + |x_2 - y_2| + |x_3 - y_3|/16$$

Herein, we move to the generalized version given in reference (3), for which:

$$d_w(x, y) = |x_1 - y_1|w_1 + |x_2 - y_2|w_2 + |x_3 - y_3|w_3$$

where we use the vector of $weight = (w_1, w_2, w_3)$.

Usage

```
codon_dist(x, y, ...)

## S4 method for signature 'DNAStringSet'
codon_dist(
  x,
  weight = NULL,
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'character'
codon_dist(
  x,
  y,
  weight = NULL,
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'CodonGroup_OR_Automorphisms'
codon_dist(
  x,
  weight = NULL,
  group = c("Z4", "Z5"),
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
     "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE
)
```

Arguments

x, y	A character string of codon sequences, i.e., sequences of DNA base-triplets. If only 'x' argument is given, then it must be a DNAStringSet-class object.
...	Not in use yet.

`weight` A numerical vector of weights to compute weighted Manhattan distance between codons. If $weight = NULL$, then $weight = (1/4, 1, 1/16)$ for $group =$ "$Z4$" and $weight = (1/5, 1, 1/25)$ for $group =$ "$Z5$".

`group` A character string denoting the group representation for the given codon sequence as shown in reference (2-3).

`cube` A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).

`num.cores, tasks` Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).

`verbose` If TRUE, prints the progress bar.

Value

A numerical vector with the pairwise distances between codons in sequences 'x' and 'y'.

References

1. Sanchez R. Evolutionary Analysis of DNA-Protein-Coding Regions Based on a Genetic Code Cube Metric. Curr Top Med Chem. 2014;14: 407–417. https://doi.org/10.2174/1568026613666131204110022.
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560. PDF.

See Also

codon_dist_matrix, automorphisms, codon_coord, and aminoacid_dist.

Examples

```
## Let's write two small DNA sequences
x = "ACGCGTGTACCGTGACTG"
y = "TGCGCCCGTGACGCGTGA"

codon_dist(x, y, group = "Z5")

## Alternatively, data can be vectors of codons, i.e., vectors of DNA
## base-triplets (including gap symbol "-").
x = c("ACG","CGT","GTA","CCG","TGA","CTG","ACG")
y = c("TGC","GCC","CGT","GAC","---","TGA","A-G")

## Gaps are not defined on "Z4"
codon_dist(x, y, group = "Z4")
```

```
## Gaps are considered on "Z5"
codon_dist(x, y, group = "Z5")

## Load an Automorphism-class object
data("autm", package = "GenomAutomorphism")
codon_dist(x = head(autm,20), group = "Z4")

## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

codon_dist(x = aln, group = "Z5")
```

`codon_dist_matrix` *Compute Codon Distance Matrix*

Description

This function computes the codon distance matrix based on the weighted Manhattan distance between codons estimated with function `codon_dist`.

Usage

```
codon_dist_matrix(
  genetic_code = "1",
  group = c("Z4", "Z5"),
  weight = NULL,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
   "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
    "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  output = c("list", "vector", "dist"),
  num.cores = 1L
)
```

Arguments

`genetic_code`	A single string that uniquely identifies the genetic code to extract. Should be one of the values in the id or name2 columns of `GENETIC_CODE_TABLE`.
`group`	A character string denoting the group representation for the given codon sequence as shown in reference (2-3).
`weight`	A numerical vector of weights to compute weighted Manhattan distance between codons. If $weight = NULL$, then $weight = (1/4, 1, 1/16)$ for $group =$ "Z4" and $weight = (1/5, 1, 1/25)$ for $group =$ "Z5" (see `codon_dist`).
`cube`	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
`output`	Format of the returned lower triangular matrix: as a list of 63 elements (labeled) or as a labeled vector using codons as labels.
`num.cores`	An integer to setup the number of parallel workers via `makeCluster`.

Details

By construction, a distance matrix is a symmetric matrix. Hence, the knowledge of lower triangular matrix is enough for its application to any downstream analysis.

Value

A lower triangular matrix excluding the diagonal.

See Also

codon_dist.

Examples

```
## The distance matrix for codons for the Invertebrate Mitochondrial,
## cube "TGCA" with base-triplet represented on the group "Z5". Each
## coordinate from each returned numerical vector corresponds to the
## distance between codons given in the coordinate name.
x <- codon_dist_matrix(genetic_code = "5", cube = "TGCA", group = "Z5",
                       output = "vector")
x[seq(61, 63)]
```

---

codon_matrix *Codon Coordinate Matrix*

Description

This function builds the coordinate matrix for each sequence from an aligned set of DNA codon sequences.

Usage

```
codon_matrix(base, ...)

## S4 method for signature 'BaseSeqMatrix'
codon_matrix(base, num.cores = 1L, tasks = 0L, verbose = TRUE, ...)

## S4 method for signature 'DNAStringSet'
codon_matrix(
  base,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
      "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z4", "Z5"),
  num.cores = 1L,
  tasks = 0L,
  verbose = TRUE
```

```
)

## S4 method for signature 'DNAMultipleAlignment'
codon_matrix(
  base,
  cube = c("ACGT", "AGCT", "TCGA", "TGCA", "CATG", "GTAC", "CTAG", "GATC", "ACTG",
    "ATCG", "GTCA", "GCTA", "CAGT", "TAGC", "TGAC", "CGAT", "AGTC", "ATGC", "CGTA",
      "CTGA", "GACT", "GCAT", "TACG", "TCAG"),
  group = c("Z4", "Z5"),
  num.cores = 1L,
  tasks = 0L,
  verbose = TRUE
)
```

Arguments

base	A DNAMultipleAlignment, a DNAStringSet, or a BaseSeqMatrix.
...	Not in use yet.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the function log to stdout
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (3-4).
group	A character string denoting the group representation for the given base or codon as shown in reference (3-4).

Details

The purpose of this function is making the codon coordinates from multiple sequence alignments (MSA) available for further downstream statistical analyses, like those reported in references (1) and (2).

Value

A ListCodonMatrix object with the codon coordinate on its metacolumns.

Author(s)

Robersy Sanchez https://genomaths.com

References

1. Lorenzo-Ginori, Juan V., Aníbal Rodríguez-Fuentes, Ricardo Grau Ábalo, and Robersy Sánchez Rodríguez. "Digital signal processing in the analysis of genomic sequences." Current Bioinformatics 4, no. 1 (2009): 28-40.

2. Sanchez, Robersy. "Evolutionary analysis of DNA-protein-coding regions based on a genetic code cube metric." Current Topics in Medicinal Chemistry 14, no. 3 (2014): 407-417.
3. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi: 10.1101/2021.06.01.446543
4. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
5. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

codon_coord, base_coord and base2int.

Examples

```
## Load the MSA of Primate BRCA1 DNA repair genes
data("brca1_aln")

## Get the DNAStringSet for the first 33 codons and apply 'codon_matrix'
brca1 <- unmasked(brca1_aln)
brca1 <- subseq(brca1, start = 1, end = 33)
codon_matrix(brca1)

## Get back the alignment object and apply 'codon_matrix' gives us the
## same result.
brca1 <- DNAMultipleAlignment(as.character(brca1))
codon_matrix(brca1)
```

ConservedRegion-class *A class definition to store conserved gene/genomic regions found in an MSA.*

Description

A class definition to store conserved gene/genomic regions found in an MSA.

Valid ConservedRegion mcols

A class definition for a list of ConservedRegion class objects.

Valid ConservedRegionList mcols

Usage

```
valid.ConservedRegion(x)

valid.ConservedRegionList(x)
```

Arguments

x	A 'ConservedRegionList' object

Details

ConservedRegionList-class has the following method:

as('from', "ConservedRegionList"):
Where 'from' is a list of **ConservedRegion-class**.

Value

Given the slot values, defines a `ConservedRegion-class` or `ConservedRegionList-class` object.

---

conserved_regions *Conserved and Non-conserved Regions from an MSA*

Description

Returns the Conserved or the Non-conserved Regions from an MSA.

Usage

```
conserved_regions(x, ...)

## S4 method for signature 'Automorphism'
conserved_regions(
  x,
  conserved = TRUE,
  output = c("all_pairs", "unique_pairs", "unique")
)

## S4 method for signature 'AutomorphismList'
conserved_regions(
  x,
  conserved = TRUE,
  output = c("all_pairs", "unique_pairs", "unique"),
  num.cores = multicoreWorkers(),
  tasks = 0L,
  verbose = FALSE
)

## S4 method for signature 'AutomorphismByCoef'
conserved_regions(
  x,
  conserved = TRUE,
  output = c("all_pairs", "unique_pairs", "unique")
```

```
)

## S4 method for signature 'AutomorphismByCoefList'
conserved_regions(
  x,
  conserved = TRUE,
  output = c("all_pairs", "unique_pairs", "unique")
)
```

Arguments

x	An Automorphism-class, an AutomorphismList-class, an AutomorphismByCoef-class, or an AutomorphismByCoefList-class object.
...	Not in use.
conserved	Logical, Whether to return the *conserved* or the *non-conserved regions*.
output	A character string. Type of output.
num.cores, tasks	Integers. Argument *num.cores* denotes the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply function from BiocParallel package). Argument *tasks* denotes the number of tasks per job. Value must be a scalar integer >= 0L. In this documentation a job is defined as a single call to a function, such as bplapply. A task is the division of the X argument into chunks. When tasks == 0 (default), X is divided as evenly as possible over the number of workers (see MulticoreParam from BiocParallel package).
verbose	logic(1). If TRUE, enable progress bar.

Value

An AutomorphismByCoef-class object containing the requested regions.

Examples

```
## Load dataset
data("autm", package = "GenomAutomorphism")
conserved_regions(autm[1:3])
## Load automorphism found COVID datatset
data("covid_autm", package = "GenomAutomorphism")

## Conserved regions in the first 100 codons
conserv <- conserved_regions(covid_autm[1:100], output = "unique")
conserv
```

covid_aln *Pairwise Sequence Alignment (MSA) of COVID-19 genomes.*

Description

This is a DNAMultipleAlignment carrying the pairwise sequence alignment of SARS coronavirus GZ02 (GenBank: AY390556.1: 265-13398_13398-21485) and Bat SARS-like coronavirus isolate bat-SL-CoVZC45 (GenBank: MG772933.1:265-13455_13455-21542), complete genomes. The alignment is available at GitHub: https://github.com/genomaths/seqalignments/tree/master/COVID-19

Usage

```
data("covid_aln", package = "GenomAutomorphism")
```

Format

DNAMultipleAlignment class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_aln, brca1_autm2, cyc_aln and covid_autm.

Examples

```
data("covid_aln", package = "GenomAutomorphism")
covid_aln
```

covid_autm *Automorphisms between DNA Sequences from two COVID-19 genomes*

Description

This is an AutomorphismList object carrying a list of pairwise automorphisms between the SARS coronavirus GZ02 (GenBank: AY390556.1: 265-13398_13398-21485) and Bat SARS-like coronavirus isolate bat-SL-CoVZC45 (GenBank: KY417151.1: protein-coding regions). The pairwise DNA sequence alignment is available in the dataset named covid_aln and the automorphisms were estimated with function autZ64.

Usage

```
data("covid_autm", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_autm, brca1_autm2, cyc_autm, and covid_aln.

Examples

```
data("covid_autm", package = "GenomAutomorphism",
    envir = environment())
covid_autm
```

---

cyc_aln *Multiple Sequence Alignment (MSA) of Primate Somatic Cytochrome C*

Description

This is a DNAMultipleAlignment carrying an MSA of Primate Somatic Cytochrome C to be used in the examples provided for the package functions. The original file can be downloaded from GitHub at: https://bit.ly/3kdEAzs

Usage

```
data("cyc_aln", package = "GenomAutomorphism")
```

Format

DNAMultipleAlignment class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_aln, brca1_aln2, covid_aln, and cyc_autm.

Examples

```
data("cyc_aln", package = "GenomAutomorphism")
cyc_aln
```

---

cyc_autm *Automorphisms between DNA Sequences from Primate Cytochrome C Genes*

Description

This is a AutomorphismList object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of Primate Somatic Cytochrome C to be used in the examples provided for the package functions. The automorphisms were estimated from the cyc_aln MSA with function autZ64.

Usage

```
data("cyc_autm", package = "GenomAutomorphism")
```

Format

AutomorphismList class object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

brca1_autm, brca1_autm2, covid_autm, and covid_aln.

Examples

```
data("cyc_autm", package = "GenomAutomorphism")
cyc_autm
```

dna_phyche *Some Physicochemical Properties of DNA bases*

Description

This data set carries some relevant physicochemical properties of the DNA bases. Available properties are:

"proton_affinity: " It is an indication of the thermodynamic gradient between a molecule and the anionic form of that molecule upon removal of a proton from it (Wikipedia). The proton affinity values, given in kJ/mol, were taken from reference (1), also available in Wolfram Alpha at https://www.wolframalpha.com/ and in the cell phone App 'Wolfram Alpha'. Reference (2) provides several measurements accomplished by several computational and experimental approaches.

"partition_coef: " 1-octanol/water partition coefficients, logP. In the physical sciences, a partition coefficient (P) or distribution coefficient (D) is the ratio of concentrations of a compound in a mixture of two immiscible solvents at equilibrium (3). The partition coefficient measures how hydrophilic ("water-loving") or hydrophobic ("water-fearing") a chemical substance is. Partition coefficients are useful in estimating the distribution of drugs within the body. Hydrophobic drugs with high octanol-water partition coefficients are mainly distributed to hydrophobic areas such as lipid bilayers of cells. Conversely, hydrophilic drugs (low octanol/water partition coefficients) are found primarily in aqueous regions such as blood serum. The partition coefficient values included here were taken from reference (1), also available in Wolfram Alpha at https://www.wolframalpha.com/ and in the cell phone App 'Wolfram Alpha'.

"dipole_moment: " Dipole-dipole, dipole-induced-dipole and London force interactions among the bases in the helix are large, and make the free energy of the helix depend on the base composition and sequence. The dipole moment values were taken from reference (4). The dipole moment of DNA bases refers to the measure of polarity in the chemical bonds between atoms within the nucleobases. Dipole moments arise due to differences in electronegativity between the bonded atoms. In DNA bases, these dipole moments can influence the orientation of the bases when interacting with other molecules or surfaces, such as graphene/h-BN interfaces. The concept of dipole moments has been applied to analyze the electric moments of RNA-binding proteins, which can help identify DNA-binding proteins and provide insights into their mechanisms and prediction.

"tautomerization_energy: " The term "tautomerism" is usually defined as structural isomerism with a low barrier to interconversion between the isomers, for example, the enol/imino forms for cytosine and guanine. Tautomerization is a process where the chemical structure of a molecule, such as DNA bases, undergoes a rearrangement of its atoms. This rearrangement results in the formation of different isomers, called tautomers, which can exist in solution or in a cell. The DNA bases can undergo tautomeric shifts, which can potentially contribute to mutagenic mispairings during DNA replication. The energy required for tautomerization of DNA bases is known as tautomerization energy. These values were taken from reference (2) and the value for each base corresponds to the average of the values estimated from different measurement approaches.

Usage

```
data("dna_phyche", package = "GenomAutomorphism")
```

Format

A data frame object.

References

1. Wolfram Research (2007), ChemicalData, Wolfram Language function, https://reference.wolfram.com/language/ref/Ch (updated 2016).
2. Moser A, Range K, York DM. Accurate proton affinity and gas-phase basicity values for molecules important in biocatalysis. J Phys Chem B. 2010;114: 13911–13921. doi:10.1021/jp107450n.
3. Leo A, Hansch C, Elkins D. Partition coefficients and their uses. Chem Rev. 1971;71: 525–616. doi:10.1021/cr60274a001.
4. Vovusha H, Amorim RG, Scheicher RH, Sanyal B. Controlling the orientation of nucleobases by dipole moment interaction with graphene/h-BN interfaces. RSC Adv. Royal Society of Chemistry; 2018;8: 6527–6531. doi:10.1039/c7ra11664k.

Examples

```
data("dna_phyche", package = "GenomAutomorphism")
dna_phyche

## Select DNA base tautomerization energy
te <- as.list(dna_phyche$tautomerization_energy)
names(te) <- rownames(dna_phyche)

## Let's create DNAStringSet-class object
base <- DNAStringSet(x = c( seq1 ='ACGTGATCAAGT',
                            seq2 = 'GTGTGATCCAGT'))

dna_phychem(seqs = base, phychem = te,
            index_name = "Tautomerization-Energy")
```

---

dna_phychem *DNA numerical matrix*

Description

This function applies the numerical indices representing various physicochemical and biochemical properties of DNA bases. As results, DNA sequences are represented as numerical vectors which can be subject of further downstream statistical analysis and digital signal processing.

Usage

```
dna_phychem(seqs, ...)

## S4 method for signature 'character'
dna_phychem(
  seqs,
  phychem = list(A = NULL, T = NULL, C = NULL, G = NULL, N = NULL)
)

## S4 method for signature 'DNAStringSet_OR_DNAMultipleAlignment'
dna_phychem(
  seqs,
  phychem = list(A = NULL, T = NULL, C = NULL, G = NULL, N = NULL),
  index_name = NULL,
  ...
)
```

Arguments

seqs	A character string, a DNAStringSet or a DNAMultipleAlignment class object carrying the DNA pairwise alignment of two sequences.
...	Not in use.
phychem	A list of DNA bases physicochemical properties, e.g., like those provided in dna_phyche.
index_name	Optional. Name or brief description of the base physicochemical property applied to represent the DNA sequence.

Value

A MatrixSeq object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

peptide_phychem_index

Examples

```
## Let's create DNAStringSet-class object
base <- DNAStringSet(x = c( seq1 ='ACGTGATCAAGT',
                            seq2 = 'GTGTGATCCAGT',
                            seq3 = 'TCCTGATCAGGT'))


dna_phychem(seqs = base,
            phychem = list('A' = 0.87, 'C' = 0.88, 'T' = 0.82,
```

```
                    'G' = 0.89, 'N' = NA),
index_name = "Proton-Affinity")
```

`GenomAutomorphism` *GenomAutomorphism: An R package to compute the automorphisms between DNA sequences represented as elements from an Abelian group.*

Description

This is an R package to compute the automorphisms between pairwise aligned DNA sequences represented as elements from a Genomic Abelian group as described in reference (1). In a general scenario, whole chromosomes or genomic regions from a population (from any species or close related species) can be algebraically represented as a direct sum of cyclic groups or more specifically Abelian p-groups. Basically, we propose the representation of multiple sequence alignments (MSA) of length N as a finite Abelian group created by the direct sum of homocyclic Abelian group of *prime-power order*.

Author(s)

Maintainer: Robersy Sanchez <genomicmath@gmail.com> (ORCID)

Authors:

- Robersy Sanchez <genomicmath@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/genomaths/GenomAutomorphism
- Report bugs at https://github.com/genomaths/GenomAutomorphism/issues

---

`getAutomorphisms` *Get Automorphisms*

Description

For the sake of saving memory, each `Automorphism-class` objects is stored in an `AutomorphismList-class`, which does not inherit from a `GRanges-class`.

Usage

```
getAutomorphisms(x, ...)

## S4 method for signature 'AutomorphismList'
getAutomorphisms(x)

## S4 method for signature 'list'
getAutomorphisms(x)

## S4 method for signature 'DataFrame_OR_data.frame'
getAutomorphisms(x)
```

Arguments

x	An AutomorphismList-class.
...	Not in use.

Details

This function just transforms each Automorphism-class object into an object from the same class but now inheriting from a GRanges-class.

Value

This function returns an AutomorphismList-class object as a list of Automorphism-class objects, which inherit from GRanges-class objects.

An AutomorphismList-class object.

An Automorphism-class object.

Examples

```
## Load a dataset
data("autm", package = "GenomAutomorphism")
aut <- mcols(autm)
aut ## This a DataFrame object

## The natural ranges for the sequence (from 1 to length(aut)) are added
getAutomorphisms(aut)

## A list of automorphisms
aut <- list(aut, aut)
getAutomorphisms(aut)

## Automorphism-class inherits from 'GRanges-class'
aut <- as(autm, "GRanges")
as(aut, "Automorphism")
```

get_coord *DNA base/codon sequence and coordinates represented on a given Abelian group.*

Description

Given a string denoting a codon or base from the DNA (or RNA) alphabet and a genetic-code Abelian group as given in reference (1), this function returns an object from CodonGroup-class carrying the DNA base/codon sequence and coordinates represented on the given Abelian group.

Usage

```
get_coord(x, ...)

## S4 method for signature 'BaseGroup_OR_CodonGroup'
get_coord(x, output = c("all", "matrix.list"))

## S4 method for signature 'DNAStringSet_OR_NULL'
get_coord(
  x,
  output = c("all", "matrix.list"),
  base_seq = TRUE,
  filepath = NULL,
  cube = "ACGT",
  group = "Z4",
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+"
)
```

Arguments

x	An object from a BaseGroup-class, CodonGroup-class, DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences. Objects from BaseGroup-class and CodonGroup-class are generated with functions: base_coord and codon_coord, respectively.
...	Not in use.
output	See 'Value' section.
base_seq	Logical. Whether to return the base or codon coordinates on the selected Abelian group. If codon coordinates are requested, then the number of the DNA bases in the given sequences must be a multiple of 3.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).

group — A character string denoting the group representation for the given base or codon as shown in reference (1).

start, end, chr, strand — Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.

Details

Symbols '-' and 'N' usually found in DNA sequence alignments to denote gaps and missing/unknown bases are represented by the number: '-1' on Z4 and '0' in Z5. In Z64 the symbol 'NA' will be returned for codons including symbols '-' and 'N'.

Although the CodonGroup-class object returned by functions codon_coord and base_coord are useful to store genomic information, the base and codon coordinates are not given on them as numeric magnitudes. Function get_coord provides a way to get the coordinates as a numeric object while still preserving the base/codon sequence information.

Value

An object from CodonSeq-class class is returned when *output = 'all'*. This has two slots, the first one carrying a list of matrices and the second one carrying the codon/base sequence information. That is, if *x* is an object from CodonSeq-class class, then a list of matrices of codon coordinate can be retrieved as x@CoordList and the information on the codon sequence as x@SeqRanges.

if *output = 'matrix.list'*, then an object from MatrixList class is returned.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## DNA base representation in the Abelian group Z5
coord <- get_coord(
    x = aln,
    cube = "ACGT",
    group = "Z5"
)
coord ## A list of vectors

## Extract the coordinate list
coordList(coord)

## Extract the sequence list
seqRanges(coord)

## DNA codon representation in the Abelian group Z64
coord <- get_coord(
    x = aln,
    base_seq = FALSE,
    cube = "ACGT",
    group = "Z64"
```

```
)
coord

## Extract the coordinate list
coordList(coord)

## Extract the sequence list
seqRanges(coord)
```

---

get_mutscore *Get Mutation Score from an AAindex or a Mutation/Distance Matrix*

Description

This function is applied to get the mutation or contact potential scores representing the similarity/distance between amino acids corresponding to substitution mutations. The scores are retrieved from a mutation matrix or a statistical protein contact potentials matrix from AAindex (ver.9.2).

Alternatively, the mutation scores can be estimated based on an user mutation matrix, for example, see aminoacid_dist and codon_dist_matrix.

Usage

```
get_mutscore(aa1, aa2, ...)

## S4 method for signature 'character,character'
get_mutscore(
  aa1,
  aa2,
  acc = NULL,
  aaindex = NULL,
  mutmat = NULL,
  alphabet = c("AA", "DNA"),
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE,
  ...
)

## S4 method for signature 'BaseSeq,missing'
get_mutscore(
  aa1,
  aa2,
  acc = NULL,
  aaindex = NULL,
  mutmat = NULL,
  alphabet = c("AA", "DNA"),
```

```
  stat = mean,
  numcores = 1L,
  num.cores = 1L,
  tasks = 0L,
  output = c("dist", "matrix", "vector"),
  na.rm = TRUE,
  verbose = TRUE,
  ...
)

## S4 method for signature 'DNAStringSet,missing'
get_mutscore(
  aa1,
  aa2,
  acc = NULL,
  aaindex = NULL,
  mutmat = NULL,
  alphabet = c("AA", "DNA"),
  stat = mean,
  num.cores = 1L,
  tasks = 0L,
  verbose = TRUE,
  output = c("dist", "matrix", "vector"),
  na.rm = TRUE,
  ...
)

## S4 method for signature 'DNAMultipleAlignment,missing'
get_mutscore(
  aa1,
  aa2,
  acc = NULL,
  aaindex = NULL,
  mutmat = NULL,
  alphabet = c("AA", "DNA"),
  stat = mean,
  num.cores = 1L,
  tasks = 0L,
  verbose = TRUE,
  output = c("dist", "matrix", "vector"),
  na.rm = TRUE,
  ...
)
```

Arguments

aa1, aa2	A simple character representing an amino acid or a character string of letter from the amino acid alphabet or base-triplets from the DNA/RNA alphabet. If ***aa1*** is an object from any of the classes: BaseSeq, DNAStringSet, or DNAMultipleAlignment,

	then argument ***aa2*** is not required.
`...`	Not in use.
`acc`	Accession id for a specified mutation or contact potential matrix.
`aaindex`	Database where the requested accession id is located. The possible values are: "aaindex2" or "aaindex3".
`mutmat`	A mutation or any score matrix provided by the user.
`alphabet`	Whether the alphabet is from the 20 amino acid (AA) or four (DNA)/RNA base alphabet. This would prevent mistakes, i.e., the strings "ACG" would be a base-triplet on the DNA alphabet or simply the amino acid sequence of alanine, cysteine, and glutamic acid.
`num.cores, tasks`	Parameters for parallel computation using package `BiocParallel-package`: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see `bplapply` and the number of tasks per job (only for Linux OS).
`verbose`	Optional. Only if num.cores > 1. If TRUE, prints the function log to stdout.
`stat`	Statistic that will be used to summarize the scores of the DNA sequences provided. Only if ***aa1*** is an object from any of the classes: BaseSeq, `DNAStringSet`, or `DNAMultipleAlignment`.
`numcores`	An integer to setup the number of parallel workers via `makeCluster`.
`output`	Optional. Class of the returned object. Only if ***aa1*** is an object from any of the classes: BaseSeq, `DNAStringSet`, or `DNAMultipleAlignment`.
`na.rm`	a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds.

Details

If a score matrix is provided by the user, then it must be a symmetric matrix 20x20.

Value

A single numeric score or a numerical vector, or if ***aa1*** is an object from any of the classes: BaseSeq, `DNAStringSet`, or `DNAMultipleAlignment`, then depending on the user selection the returned object will be:

1. A lower diagonal numerical vector of the sequence pairwise scores.
2. A `dist`-class object.
3. A whole score matrix.

Author(s)

Robersy Sanchez `https://genomaths.com`

See Also

aa_mutmat, aaindex2 and aaindex3.

Examples

```
## A single amino acids substitution mutation
get_mutscore("A", "C", acc = "MIYS930101", aaindex = "aaindex2")

## A tri-peptide mutation
get_mutscore(aa1 = "ACG", aa2 = "ATG", acc = "MIYS930101",
            aaindex = "aaindex2", alphabet = "AA")

## A single base-triple mutation, i.e., a single amino acid substitution
## mutation
get_mutscore(aa1 = "ACG", aa2 = "CTA", acc = "MIYS930101",
            aaindex = "aaindex2", alphabet = "DNA")

## Peptides can be also written as:
get_mutscore(aa1 = c("A","C","G"), aa2 = c("C","T","A"),
            acc = "MIYS930101", aaindex = "aaindex2", alphabet = "AA")
```

GRangesMatrixSeq-class

Definition of GRangesMatrixSeq-class

Description

This is a very simple flexible class to store DNA and aminoacid aligned sequences together with their physicochemical properties. That is, a place where each aminoacid or codon from the sequence is represented by numerical value from a physicochemical index.

Constructor for 'GRangesMatrixSeq-class' object.

Usage

```
GRangesMatrixSeq(
  object = NULL,
  seqnames = Rle(factor()),
  start = integer(0),
  end = integer(0),
  ranges = IRanges(),
  strands = Rle(strand()),
  elementMetadata = DataFrame(),
  seqinfo = NULL,
  seqs = character(),
  names = character(),
  aaindex = character(),
  phychem = character(),
  accession = character()
)
```

Arguments

`object`	If provided, it must be a GRangesMatrixSeq-class object and in this case
`seqnames`, `start`, `end`, `ranges`, `strand`, `elementMetadata`, `seqinfo`	The same as in `GRanges`
`seqs`, `names`, `aaindex`, `phychem`, `accession`	The same as in MatrixSeq.

Details

This is a convenient function to transform a MatrixSeq-class object returned by function aa_phychem_index into a 'GRangesMatrixSeq-class' object. Since a 'GRangesMatrixSeq-class' inherits from `GRanges-class`, this transformation permits the application of several methods from GenomicRanges package in the downstream analysis.

Value

Given the slot values, defines a `GRangesMatrixSeq-class` object.

Only used to specify signature in the S4 setMethod.

Examples

```
aln <- c(S1 = "ATGCGGATTAGA", S2 = "ATGACGATCACA",
        S3 = "ATGAGATCACAG")
cd <- DNAMultipleAlignment(aln)
r1 <- peptide_phychem_index(unmasked(cd), acc = "EISD840101")

r2 <- GRangesMatrixSeq(r1)
r2

slot(r2, "phychem")
```

`GRanges_OR_NULL-class` *A definition for the union of 'GRanges' and 'NULL' class.*

Description

A definition for the union of 'GRanges' and 'NULL' class.

is.url *Check URLs*

Description

Check URLs

Usage

```
is.url(x)
```

Details

Internal use only.

Value

Logical values

ListCodonMatrix-class *A Convenient Class to Store Codon Coordinates in given Genetic Code cube.*

Description

ListCodonMatrix-class objects are generated by function codon_matrix.

Usage

```
ListCodonMatrix(object, cube, group, seq_alias = NULL, names = NULL)

valid.ListCodonMatrix(x)
```

Arguments

object	A list of CodonMatrix-class objects
x	A 'ListCodonMatrix-class' object

Value

A ListCodonMatrix-class object.

Examples

```
## ListCodonMatrix-class objects are generated by function 'codon_matrix'.
## Let's create DNAStringSet-class object
base <- DNAStringSet(x = c( seq1 ='ACGTGATCAAGT',
                            seq2 = 'GTGTGATCCAGT',
                            seq3 = 'TCCTGATCAGGT'))

x1 <- codon_matrix(base)
x1

## Extract the first element
x1[1]
x1$codon.1
x1[[1]]
```

`matrices` *Get the Coordinate Representation from DNA Sequences on Specified Abelian Group*

Description

Extract the Coordinate Representation from DNA Sequences on Specified Abelian Group.

Usage

```
matrices(x, ...)

## S4 method for signature 'MatrixList'
matrices(x)

## S4 method for signature 'CodonSeq'
matrices(x)

## S4 method for signature 'DNAStringSet_OR_NULL'
matrices(
  x,
  base_seq = TRUE,
  filepath = NULL,
  cube = "ACGT",
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3"),
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+"
)
```

Arguments

x	An object from a DNAStringSet or DNAMultipleAlignment class carrying the DNA pairwise alignment of two sequences.
...	Not in use.
base_seq	Logical. Whether to return the base or codon coordinates on the selected Abelian group. If codon coordinates are requested, then the number of the DNA bases in the given sequences must be a multiple of 3.
filepath	A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.
cube	A character string denoting one of the 24 Genetic-code cubes, as given in references (2-3).
group	A character string denoting the group representation for the given base or codon as shown in reference (1).
start, end, chr, strand	Optional parameters required to build a GRanges-class. If not provided the default values given for the function definition will be used.

Details

These are alternative ways to get the list of matrices of base/codon coordinate and the information on the codon sequence from CodonSeq and MatrixList class objects. These functions can either take the output from functions base_coord and matrices or to operate directly on a DNAStringSet or to retrieve a DNA sequence alignment from a file.

base_seq parameter will determine whether to return the matrices of coordinate for a DNA or codon sequence. While in function seqranges, **granges** parameter will determine whether to return a GRanges-class object or a DataFrame.

Value

A list of vectors (group = c("Z4", "Z5", "Z64", "Z125")) or a list of matrices (group = ("Z4^3", "Z5^3")) carrying the coordinate representation on the specified Abelian group.

Author(s)

Robersy Sanchez https://genomaths.com

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi: 10.1101/2021.06.01.446543
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

Symmetric Group of the Genetic-Code Cubes.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## Coordinate representation of the aligned sequences on "Z4".
## A list of vectors
matrices(
    x = aln,
    base_seq = TRUE,
    filepath = NULL,
    cube = "ACGT",
    group = "Z4",
)

## Coordinate representation of the aligned sequences on "Z4".
## A list of matrices
matrices(
    x = aln,
    base_seq = FALSE,
    filepath = NULL,
    cube = "ACGT",
    group = "Z5^3",
)
```

MatrixList-class *Definition of MatrixList-class*

Description

A class denoting a list of matrices.

Value

Given the slot values, defines a MatrixList-class object.

`MatrixSeq-class` *Definition of MatrixSeq-class*

Description

This is a very simple flexible class to store DNA and aminoacid aligned sequences together with their physicochemical properties. That is, a place where each aminoacid or codon from the sequence is represented by numerical value from a physicochemical index.

Usage

```
MatrixSeq(seqs, matrix, names, aaindex, phychem, accession)

## S4 method for signature 'MatrixSeq'
show(object)
```

Arguments

`seqs`, `matrix`, `names`, `aaindex`, `phychem`, `accession`	See detail section
`object`	An object from 'MatrixSeq' class

Details

seqs: A string character vector of DNA or aminoacid sequences.

matrix: A numerical matrix or a numerical vector (in the constructor) carrying the specified aminoacid physicochemical indices for aminoacid in the DNA or aminoacid sequence(s).

names: Alias/names/IDs DNA or aminoacid sequences.

aaindex: Aminoacid index database where the physicochemical index can be found.

phychem: Description of the physicochemical index applied to represent the DNA or aminoacid sequences.

accession: Accession number or ID of the applied physicochemical index in the database.

Value

Given the slot values, defines a `MatrixSeq-class` object.

A `MatrixSeq-class` object.

Prints a summary of the `MatrixSeq-class` object and returns it invisibly.

Author(s)

Robersy Sanchez https://genomaths.com

Examples

```
aln <- c(S1 = "ATGCGGATTAGA", S2 = "ATGACGATCACA", S3 = "ATGAGATCACAG")
cd <- DNAMultipleAlignment(aln)
r1 <- peptide_phychem_index(unmasked(cd), acc = "EISD840101")
r1

## Extract the second aminoacid sequence
r1[2]

## Using the sequence given name
r1$S1

## Extract the second aminoacid value from the first sequence
r1[1,2]

## Change the name of the second sequence
names(r1) <- c('S1', 'Seq1', 'S1')
r1

## Extract the amino acid sequences
slot(r1, 'seqs')
```

---

mod *Modulo Operation*

Description

Integer remainder of the division of the integer n by m: n mod m.

Usage

```
mod(n, m, ...)

## S4 method for signature 'matrix,numeric'
mod(n, m)
```

Arguments

n	A numeric vector (preferably of integers), a matrix where each element can be reduced to integers.
m	An integer vector (positive, zero, or negative).
...	Not in use.

Value

A numeric vector or matrix carrying the remainder of the division of 'n' by 'm'.

Author(s)

Robersy Sanchez (https://genomaths.com).

Examples

```
## Example 1
## Build a matrix 'n' and set a vector of integers 'm'
n <- diag(x=1, nrow = 4, ncol = 4) * c(43,125,2,112)
m <- c(64,4,4,64)

## Operation n mod m
mod(n = n, m = m)

## Or simply:
n %% m

## Example 2
m <- matrix(c(8,2,3, 11,12,13), nrow = 2)
m

m %% 4
```

---

modeq *A Wrapper Calling Modular Linear Equation Solver (MLE)*

Description

It is just a wrapper function to call modlin. This function is intended to be used internally. The MLE ($a * x = bmodn$) does not always have a solution. If the MLE has no solution the function will return the value -1. Also, if $a * x = bmodn$ has solution x = 0, then function *'modeq'* will return -1.

Usage

```
modeq(a, b, n)
```

Value

A number. If the equation has no solution in its domain of definition, it will return -1.

Examples

```
## The MLE 10 * x = 3 mod 64 has no solution
modeq(10, 3, 64)

## The result is the same as calling modlin(10, 4, 64)
modeq(10, 4, 64)
```

modlineq *Modular System of Linear Equation Solver (MLE)*

Description

If a, b, and n are integer vectors, this function tries to find, at each coordinate, the solution of the MLE $ax = b \bmod n$. If the MLE $ax = bmodn$ has no solution for some coordinate (see modlin), the value reported for that coordinate will be *no.sol*, and a corresponding translation vector will also be returned.

Usage

```
modlineq(a, b, n, no.sol = 0L)
```

Arguments

a	An integer or a vector of integers.
b	An integer or a vector of integers.
n	An integer or a vector of integers.
no.sol	Values to return when the equation is not solvable or yield the value 0. Default is 0.

Details

For a, b, and n integer scalars, it is just a wrapper function to call modlin.

Value

If the solution is exact, then a numerical vector will be returned, otherwise, if there is no exact solution for some coordinate, a list carrying the element on the diagonal matrix and a translation vector will be returned.

Examples

```
## Set the vector x, y, and m.
x <- c(9,32,24,56,60,27,28,5)
y <-  c(8,1,0,56,60,0,28,2)
modulo <- c(64,125,64,64,64,64,64,64)

## Try to solve the modular equation a x = b mod n
m <- modlineq(a = x, b = y, n = modulo)
m

## Or in matrix form
diag(m)

## The reverse mapping is an affine transformation
mt <- modlineq(a = y, b = x, n = modulo, no.sol = 1L)
```

```
mt

## That is, vector 'x' is recovered with the transformation
(y %*% diag(mt$diag) + mt$translation) %% modulo

# Or
cat("\n---- \n")

(y %*% diag(mt$diag) + mt$translation) %% modulo == x
```

`mut_type` *Classification of DNA base mutations*

Description

Each DNA/RNA base can be classified into three main classes according to three criteria (1): number of hydrogen bonds (strong-weak), chemical type (purine-pyrimidine), and chemical groups (amino versus keto). Each criterion produces a partition of the set of bases: 1. According to the number of hydrogen bonds (on DNA/RNA double helix): strong $S = (C, G)$ (three hydrogen bonds) and weak $W = (A, U)$ (two hydrogen bonds); 2. According to the chemical type: purines $R = (A, G)$ and pyrimidines $Y = (C, U)$. 3. According to the presence of amino or keto groups on the base rings: amino $M = (C, A)$ and keto $K = (G, U)$. So, each mutational event can be classified according to the type of mutation involved (2).

Usage

```
mut_type(x, y)
```

Arguments

`x, y`	Character strings denoting DNA bases

Value

A character string of same length of 'x' and 'y'.

References

1. A. Cornish-Bowden, Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984, Nucleic Acids Res. 13 (1985) 3021-3030.

2. MA.A. Jimenez-Montano, C.R. de la Mora-Basanez, T. Poschel, The hypercube structure of the genetic code explains conservative and non-conservative aminoacid substitutions in vivo and in vitro, Biosystems. 39 (1996) 117-125.

Examples

```
## Mutation type 'R'
mut_type("A", "G")

## Mutation type 'M'
mut_type("A", "C")

## Mutation type 'W'
mut_type("A", "T")

## Mutation type 'S'
mut_type("G", "C")
```

`peptide_phychem_index` *Amino acid numerical matrix*

Description

This function applies numerical indices representing various physicochemical and biochemical properties of amino acids and pairs of amino acids to DNA protein-coding or to aminoacid sequences. As results, DNA protein-coding or the aminoacid sequences are represented as numerical vectors which can be subject of further downstream statistical analysis and digital signal processing.

Usage

```
peptide_phychem_index(aa, ...)

## S4 method for signature 'character'
peptide_phychem_index(
  aa,
  acc = NULL,
  aaindex = NA,
  userindex = NULL,
  alphabet = c("AA", "DNA"),
  genetic.code = getGeneticCode("1"),
  no.init.codon = FALSE,
  if.fuzzy.codon = "error",
  ...
)

## S4 method for signature 'DNAStringSet_OR_DNAMultipleAlignment'
peptide_phychem_index(
  aa,
  acc = NULL,
  aaindex = NA,
  userindex = NULL,
  alphabet = c("AA", "DNA"),
```

```
  genetic.code = getGeneticCode("1"),
  no.init.codon = FALSE,
  if.fuzzy.codon = "error",
  num.cores = 1L,
  tasks = 0L,
  verbose = FALSE,
  ...
)
```

Arguments

aa	A character string, a DNAStringSet or a DNAMultipleAlignment class object carrying the DNA pairwise alignment of two sequences.
...	Not in use.
acc	Accession id for a specified mutation or contact potential matrix.
aaindex	Database where the requested accession id is located and from where the aminoacid indices can be obtained. The possible values are: "aaindex2" or "aaindex3".
userindex	User provided aminoacid indices. This can be a numerical vector or a matrix (20 x 20). If a numerical matrix is provided, then the aminoacid indices are computed as column averages.
alphabet	Whether the alphabet is from the 20 aminoacid (AA) or four (DNA)/RNA base alphabet. This would prevent mistakes, i.e., the strings "ACG" would be a base-triplet on the DNA alphabet or simply the amino acid sequence of alanine, cysteine, and glutamic acid.
genetic.code, no.init.codon, if.fuzzy.codon	The same as given in function translation.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the function log to stdout.

Details

If a DNA sequence is given, then it is assumed that it is a DNA base-triplet sequence, i.e., the base sequence must be a multiple of 3.

Errors can be originated if the given sequences carry letter which are not from the DNA or aminoacid alphabet.

Value

If argument ***aa*** is a character string, a numerical vector carrying the physicochemical index values for the given amino acid or DNA protein-coding sequence is returned. If ***aa*** is a DNAStringSet or DNAMultipleAlignment object, then a MatrixSeq object carrying the numerical matrix for all the given sequences is returned.

Author(s)

Robersy Sanchez https://genomaths.com

Examples

```
## Let's create DNAStringSet-class object
base <- DNAStringSet(x = c( seq1 ='ACGTCATAAAGT',
                            seq2 = 'GTGTAATACAGT',
                            seq3 = 'TCCTCATAAGGT'))

## The stop codon 'TAA' yields NA
aa <- peptide_phychem_index(base, acc = "EISD840101")
aa

## Description of the physicochemical index
slot(aa, 'phychem')

## Get the aminoacid sequences. The stop codon 'TAA' is replaced by '*'.
slot(aa, 'seqs')


aa <- peptide_phychem_index(base, acc = "MIYS850103", aaindex = "aaindex3")
aa

## Description of the physicochemical index
slot(aa, 'phychem')
```

---

reexports *Reexport useful functions to be available to users*

Description

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

BiocGenerics end(), end<-(), start(), start<-(), strand(), strand<-(), width()

Biostrings AAMultipleAlignment(), AAStringSet(), DNAMultipleAlignment(), DNAStringSet(), GENETIC_CODE_TABLE, getGeneticCode(), readDNAMultipleAlignment(), translate(), unmasked()

GenomicRanges GRangesList(), makeGRangesFromDataFrame()

matrixStats colMeans2(), colSds(), colSums2(), colVars(), rowMeans2(), rowSds(), rowSums2(), rowVars()

numbers modlin(), modq()

S4Vectors mcols(), mcols<-(), setValidity2()

XVector subseq()

Value

The same as in mcols.

The same as in mcols.

The same as in setValidity2.

The same as in DNAStringSet.

The same as in AAStringSet.

The same as in readDNAMultipleAlignment.

The same as in DNAMultipleAlignment.

The same as in AAMultipleAlignment.

The same as in subseq.

The same as in translate.

The same as in GENETIC_CODE_TABLE.

The same as in getGeneticCode.

The same as in unmasked.

The same as in width.

The same as in start.

The same as in start.

The same as in end.

The same as in end.

The same as in strand.

The same as in strand.

The same as in GRangesList.

The same as in makeGRangesFromDataFrame.

The same as in modq.

The same as in modlin.

The same as in rowSums2.

The same as in colSums2.

The same as in colMeans2.

The same as in rowMeans2.

The same as in rowVars.

The same as in colVars.

The same as in colSds.

The same as in rowSds.

Examples

```
## Load an Automorphism object and take its metacolumns
data("autm", package = "GenomAutomorphism")
mcols(autm)
## See \code{\link[BiocGenerics]{start}}.

## Load an Automorphism object and get some 'end' coordinates
data("autm", package = "GenomAutomorphism")
end(autm[20:50])
```

`seqranges` *Get DNA sequence Ranges and Coordinates representation on a given Abelian Group*

Description

Extract the gene ranges and coordinates from a pairwise alignment of codon/base sequences represented on a given Abelian group.

Usage

```
seqranges(x, ...)

## S4 method for signature 'CodonSeq'
seqranges(x, granges = TRUE)

## S4 method for signature 'DNAStringSet_OR_NULL'
seqranges(
  x,
  granges = TRUE,
  base_seq = TRUE,
  filepath = NULL,
  start = NA,
  end = NA,
  chr = 1L,
  strand = "+"
)
```

Arguments

`x`	An object from a `DNAStringSet` or `DNAMultipleAlignment` class carrying the DNA pairwise alignment of two sequences.
`...`	Not in use.
`granges`	Logical. Whether to return a `GRanges-class` object or a `DataFrame`.
`base_seq`	Logical. Whether to return the base or codon coordinates on the selected Abelian group. If codon coordinates are requested, then the number of the DNA bases in the given sequences must be a multiple of 3.

`filepath` A character vector containing the path to a file in **fasta** format to be read. This argument must be given if *codon & base* arguments are not provided.

`start`, `end`, `chr`, `strand`
Optional parameters required to build a `GRanges-class`. If not provided the default values given for the function definition will be used.

Details

This function provides an alternative way to get the codon coordinate and the information on the codon sequence from a `CodonSeq` class objects. The function can either take the output from functions `codon_coord` or to operate directly on a `DNAStringSet` or to retrieve a DNA sequence alignment from a file.

Value

A `GRanges-class` object if *granges* = *TRUE* (default), or a `DataFrame` object otherwise.

Author(s)

Robersy Sanchez https://genomaths.com

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. doi:10.1101/2021.06.01.446543
2. M. V Jose, E.R. Morgado, R. Sanchez, T. Govezensky, The 24 possible algebraic representations of the standard genetic code in six or in three dimensions, Adv. Stud. Biol. 4 (2012) 119-152.PDF.
3. R. Sanchez. Symmetric Group of the Genetic-Code Cubes. Effect of the Genetic-Code Architecture on the Evolutionary Process MATCH Commun. Math. Comput. Chem. 79 (2018) 527-560.

See Also

`matrices`, `codon_coord`, and `base_coord`.

Examples

```
## Load a pairwise alignment
data("aln", package = "GenomAutomorphism")
aln

## A GRanges object carrying the aligned DNA sequence.
seqranges(
    x = aln,
    base_seq = TRUE,
    filepath = NULL,
)

## A GRanges object carrying the aligned codon sequence.
seqranges(
```

```
    x = aln,
    base_seq = FALSE,
    filepath = NULL,
)
```

`show,CodonSeq-method` *Show method for 'CodonSeq' class object*

Description

Show method for 'CodonSeq' class object

Show method for 'ListCodonMatrix' class object

Show method for 'MatrixList' class object

Usage

```
## S4 method for signature 'CodonSeq'
show(object)

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

## S4 method for signature 'MatrixList'
show(object)
```

Arguments

`object`	An object from 'MatrixList' class

Value

Prints a summary of the `ListCodonMatrix-class` object and returns it invisibly.

Prints a summary of the `MatrixList-class` object and returns it invisibly.

`slapply` *Apply a function over a list-like object preserving its attributes*

Description

This function applies a function over a list-like object preserving its attributes and simplify (if requested) the list as `sapply` function does. **slapply** returns a list of the same length as 'x', each element of which is the result of applying FUN to the corresponding element of 'x'.

Usage

```
slapply(
  x,
  FUN,
  keep.attr = FALSE,
  class = NULL,
  simplify = TRUE,
  USE.NAMES = TRUE,
  ...
)
```

Arguments

x	A list-like or vector-like object.
FUN, ...	The same as described in lapply.
keep.attr	Logic. If TRUE, then the original attributes from 'x' are preserved in the returned list. Default is FALSE.
class	Name of the class to which the returned list belongs to. Default is NULL.
simplify, USE.NAMES	The same as described in sapply.

Value

Same as in ?base::lapply if keep.attr = FALSE. Otherwise same values preserving original attributes from 'x'.

Author(s)

Robersy Sanchez (https://genomaths.com).

See Also

lapply and sapply

Examples

```
## Create a list
x <- list(a = seq(10), beta = exp(seq(-3, 3)),
          logic = c(TRUE, FALSE, FALSE, TRUE))
class(x) <- "nice"

## To compute the list mean for each list element using 'base::lapply'
class(slapply(x, mean, simplify = FALSE))

## Simply 'base::lapply' preserving attributes
slapply(x, mean, keep.attr = TRUE, simplify = FALSE)

## To preserve attributes and simplify
slapply(x, mean, keep.attr = TRUE, simplify = TRUE)
```

sortByChromAndStart *Sorting* GRanges-class *objects*

Description

Sorts a GRanges-class objects by seqname (chromosome), start, and position.

Usage

```
sortByChromAndStart(x)

sortByChromAndEnd(x)
```

Arguments

x	GRanges object

Details

Objects that inherits from a GRanges-class can be sorted as well.

Value

GRanges-class object or from the original object class.

Examples

```
GR <- as(c("chr2:1-1", "chr1:1-1"), "GRanges")
GR <- sortByChromAndStart(GR)
```

str2chr *String to Character*

Description

A simple function to transform a string into character vector.

Usage

```
str2chr(x, split = "", ...)

## S4 method for signature 'character'
str2chr(x, split = "", ...)

## S4 method for signature 'list'
str2chr(x, split = "", num.cores = 1L, tasks = 0L, verbose = FALSE, ...)
```

Arguments

x	A character string or a list/vector of character strings.
split	The same as in strsplit
...	Further parameters for strsplit.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the function log to stdout.

Value

A character vector, or a list of character vectors if 'x' is a list.

Author(s)

Robersy Sanchez https://genomaths.com

Examples

```
## A character string
str2chr("ATCAGCGGGATCTT")

## A list of character strings
str2chr(list(str1 = "ATCAGCGGGATCTT", str2 = "CTTCTTCGTCAGGC"))
```

---

str2dig *String to Digits*

Description

A simple function to transform a string of digits into a numeric vector.

Usage

```
str2dig(x, split = "", ...)

## S4 method for signature 'character'
str2dig(x, split = "", ...)

## S4 method for signature 'list'
str2dig(x, split = "", num.cores = 1L, tasks = 0L, verbose = FALSE, ...)
```

Arguments

x	A character string or a list of character strings of numeric/digit symbols.
split	The same as in strsplit
...	Further parameters for strsplit.
num.cores, tasks	Parameters for parallel computation using package BiocParallel-package: the number of cores to use, i.e. at most how many child processes will be run simultaneously (see bplapply and the number of tasks per job (only for Linux OS).
verbose	If TRUE, prints the function log to stdout.

Value

An integer vector or a list of integer vectors.

Author(s)

Robersy Sanchez https://genomaths.com

Examples

```
## An integer vector
str2dig("12231456247")

## A list of integer vectors
str2dig(list(num1 = "12231456247", num2 = "521436897"))
```

---

translation *Translation of DNA/RNA sequences*

Description

This function extends translate function to include letters that are frequently found in the DNA sequence databases to indicate missing information and are not part of the DNA/RNA alphabet. Also, it is able to process sequences as just simple 'character' objects.

Usage

```
translation(x, ...)

## S4 method for signature 'character'
translation(
  x,
  genetic.code = getGeneticCode("1"),
  no.init.codon = FALSE,
  if.fuzzy.codon = "error"
```

```
)

## S4 method for signature 'BioString'
translation(
  x,
  genetic.code = getGeneticCode("1"),
  no.init.codon = FALSE,
  if.fuzzy.codon = "error"
)
```

Arguments

x	A character string or the same arguments given to function translate.
...	Not in use yet.
genetic.code	The same as in translate
no.init.codon, if.fuzzy.codon	Used only if 'x' is not a 'character' object. The same as in translate.

Details

If argument 'x' belongs to any of the classes admitted by function translate, then this function is called to make the translation.

Value

If argument *x* is a character string, a character string or vector with the translated amino acid sequence(s) is returned. Otherwise, the object type returned by translate is returned, typically an AAStringSet object.

Author(s)

Robersy Sanchez https://genomaths.com

See Also

translate

Examples

```
## Load a small DNA sequence alignment
data("aln", package = "GenomAutomorphism")

translation(aln)

## Load a pairwise DNA sequence alignment of COVID-19 genomes
data("covid_aln", package = "GenomAutomorphism")

translation(covid_aln)
```

`valid.Automorphism.mcols` *Valid Automorphism mcols*

Description

Valid Automorphism mcols

Valid Automorphism

Usage

```
valid.Automorphism.mcols(x)

valid.Automorphism(x)
```

Arguments

x	An 'Automorphism' object

Value

An Error if the metacolumn does not have a valid format

An Error if the Automorphism-class object is not valid.

`valid.AutomorphismByCoef` *Valid AutomorphismByCoef mcols*

Description

Valid AutomorphismByCoef mcols

Usage

```
valid.AutomorphismByCoef(x)
```

Arguments

x	An 'AutomorphismByCoef' object

Value

An error if 'x' is not a valid AutomorphismByCoef.

valid.AutomorphismByCoefList

Valid AutomorphismByCoefList mcols

Description

Valid AutomorphismByCoefList mcols

Usage

```
valid.AutomorphismByCoefList(x)
```

Arguments

x	An 'AutomorphismByCoefList' object

Value

An error if 'x' is not a valid AutomorphismByCoefList.

valid.AutomorphismList

Valid AutomorphismList mcols

Description

Valid AutomorphismList mcols

Usage

```
valid.AutomorphismList(x)
```

Arguments

x	An 'AutomorphismList' object

Value

An error if 'x' is not a valid AutomorphismList class object.

`valid.BaseGroup.elem` *Valid BaseGroup mcols*

Description

Valid BaseGroup mcols

Valid 'BaseGroup' inheritance from 'GRanges' class

Valid BaseGroup

Usage

```
valid.BaseGroup.elem(x)

valid.GRanges(x)

valid.BaseGroup(x)
```

Arguments

x	A 'BaseGroup' object

Value

If valid return NULL

If valid return NULL

If valid return NULL

`valid.CodonGroup.mcols`
Valid CodonGroup mcols

Description

Valid CodonGroup mcols

Valid CodonGroup

Usage

```
valid.CodonGroup.mcols(x)

valid.CodonGroup(x)
```

Arguments

x	A 'CodonGroup' object

Value

If valid return NULL

If valid return NULL

`valid.MatrixList` *Valid MatrixList*

Description

Valid MatrixList

Usage

```
valid.MatrixList(x)
```

Arguments

x	A 'MatrixList' object

Value

If valid return NULL

Only used to specify signature in the S4 setMethod.

`[,AutomorphismList,ANY-method` *An S4 class to extract elements for objects created with GenomAutomorphism package*

Description

First and second level subsetting of 'x'. Extraction using names can be done as x$name.

Usage

```
## S4 method for signature 'AutomorphismList,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ListCodonMatrix,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'MatrixSeq,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'AutomorphismList'
```

```
x[[i, j, ...]]

## S4 method for signature 'ListCodonMatrix'
x[[i, j, ...]]

## S4 method for signature 'AutomorphismList'
x$name

## S4 method for signature 'ListCodonMatrix'
names(x)

## S4 method for signature 'ListCodonMatrix'
x$name

## S4 method for signature 'MatrixSeq'
x$name

## S4 replacement method for signature 'MatrixSeq'
names(x) <- value
```

Arguments

x	An object from AutomorphismList, ListCodonMatrix, or MatrixSeq.
i, j, ...	As in Extract.
name	Element name in the list 'x'.
value	A character vector of up to the same length as x, or NULL.

Value

For the [method, an object of the same class as *x* (an AutomorphismList, ListCodonMatrix, or MatrixSeq object) carrying the requested subset. For the [[and $ methods applied on an AutomorphismList object, a single unwrapped GRanges-class element. For the [[and $ methods applied on a ListCodonMatrix object, a single CodonMatrix element.

Author(s)

Robersy Sanchez https://genomaths.com

Robersy Sanchez (https://genomaths.com).

Examples

```
## Load automorphisms found BRCA1 primate genes
data("brca1_autm", package = "GenomAutomorphism")

## Extract AutomorphismList object with only one element
brca1_autm[1]

## Extract Automorphism object with only one element
brca1_autm[[3]]
```

```
## Extract Automorphism object using element name.
brca1_autm[["human_1.gorilla_1"]]
```

Index

∗ **datasets**
aaindex1, 4
aaindex2, 5
aaindex3, 6
aln, 8
autby_coef, 15
autm, 15
autm_3d, 16
autm_z125, 17
brca1_aln, 44
brca1_aln2, 45
brca1_autm, 46
brca1_autm2, 46
cdm_z64, 47
covid_aln, 61
covid_autm, 61
cyc_aln, 62
cyc_autm, 63
dna_phyche, 64
∗ **internal**
[,AutomorphismList,ANY-method, 100
Automorphism-class, 17
AutomorphismByCoef-class, 18
AutomorphismByCoefList-class, 19
AutomorphismList-class, 21
base_repl, 44
BaseGroup-class, 39
BaseGroup_OR_CodonGroup-class, 39
BaseSeq-class, 39
BaseSeqMatrix-class, 40
CodonGroup-class, 47
CodonMatrix-class, 48
CodonSeq-class, 49
ConservedRegion-class, 58
GenomAutomorphism, 67
GRanges_OR_NULL-class, 75
GRangesMatrixSeq-class, 74
is.url, 76
ListCodonMatrix-class, 76
MatrixList-class, 79
MatrixSeq-class, 80
modeq, 82
reexports, 87
show,CodonSeq-method, 91
valid.Automorphism.mcols, 97
valid.AutomorphismByCoef, 97
valid.AutomorphismByCoefList, 98
valid.AutomorphismList, 98
valid.BaseGroup.elem, 99
valid.CodonGroup.mcols, 99
valid.MatrixList, 100
'[' *([,AutomorphismList,ANY-method)*, 100
'[[' *([,AutomorphismList,ANY-method)*, 100
'%%' *(mod)*, 81
'names<-' *(AutomorphismList-class)*, 21
[,AutomorphismList,ANY-method, 100
[,ListCodonMatrix,ANY-method *([,AutomorphismList,ANY-method)*, 100
[,MatrixSeq,ANY-method *([,AutomorphismList,ANY-method)*, 100
[[,AutomorphismList-method *([,AutomorphismList,ANY-method)*, 100
[[,ListCodonMatrix-method *([,AutomorphismList,ANY-method)*, 100
$ *([,AutomorphismList,ANY-method)*, 100
$,AutomorphismList-method *([,AutomorphismList,ANY-method)*, 100
$,ListCodonMatrix-method *([,AutomorphismList,ANY-method)*, 100
$,MatrixSeq-method *([,AutomorphismList,ANY-method)*,

100

aa_index *(aa_phychem_index)*, 7
aa_mutmat, *5*, *73*
aa_mutmat *(aa_phychem_index)*, 7
aa_phychem_index, 7, *75*
aaindex1, 4, *6*, *8*
aaindex2, *4, 5*, 5, *6*, *8*, *73*
aaindex3, *4*, 6, *8*, *73*
AAMultipleAlignment, *88*
AAMultipleAlignment *(reexports)*, 87
AAMultipleAlignment(), *87*
AAStringSet, *88*, *96*
AAStringSet *(reexports)*, 87
AAStringSet(), *87*
aln, 8
aminoacid_dist, 9, *54*, *71*
aminoacid_dist,AAStringSet,ANY-method *(aminoacid_dist)*, 9
aminoacid_dist,character,character-method *(aminoacid_dist)*, 9
aminoacid_dist,CodonGroup_OR_Automorphisms,ANY-method *(aminoacid_dist)*, 9
aminoacid_dist,DNAStringSet,ANY-method *(aminoacid_dist)*, 9
as.AutomorphismList, 12, *21*
as.AutomorphismList,GRangesList,GRanges_OR_NULL-method *(as.AutomorphismList)*, 12
as.AutomorphismList,list,GRanges_OR_NULL-method *(as.AutomorphismList)*, 12
as.list,AutomorphismList-method *(AutomorphismList-class)*, 21
aut3D, 13, *16*
autby_coef, 15
autm, 15
autm_3d, 16
autm_z125, 17
Automorphism *(Automorphism-class)*, 17
Automorphism-class, 17
automorphism_bycoef, *12*, *15*, *18*, *25*, 26, *29*
automorphism_bycoef,Automorphism-method *(automorphism_bycoef)*, 26
automorphism_bycoef,AutomorphismList-method *(automorphism_bycoef)*, 26
automorphism_prob, 27
automorphism_prob,AutomorphismByCoef-method *(automorphism_prob)*, 27
automorphism_prob,AutomorphismByCoefList-method *(automorphism_prob)*, 27
AutomorphismByCoef, *25*
AutomorphismByCoef *(AutomorphismByCoef-class)*, 18
AutomorphismByCoef-class, 18
AutomorphismByCoefList, *15*
AutomorphismByCoefList *(AutomorphismByCoefList-class)*, 19
AutomorphismByCoefList-class, 19
automorphismByRanges, 20, *24, 25*
automorphismByRanges,Automorphism-method *(automorphismByRanges)*, 20
automorphismByRanges,AutomorphismList-method *(automorphismByRanges)*, 20
AutomorphismList, *15–17*, *46, 47*, *61–63*, *101*
AutomorphismList *(AutomorphismList-class)*, 21
AutomorphismList-class, 21
automorphisms, *11, 12*, *17*, *20, 21*, 23, *25*, *27*, *33*, *39, 40*, *48*, *54*
automorphisms,DNAStringSet_OR_NULL-method *(automorphisms)*, 23
autZ125, *17*, 30
autZ5, 32
autZ64, *15*, *25*, 33, *46*, *61*, *63*

base2codon, 35
base2codon,character-method *(base2codon)*, 35
base2codon,DNAMultipleAlignment-method *(base2codon)*, 35
base2codon,DNAStringSet-method *(base2codon)*, 35
base2int, 36, *43*, *51*, *58*
base2int,character-method *(base2int)*, 36
base2int,data.frame-method *(base2int)*, 36
base_coord, *38*, 40, *43*, *48*, *50, 51*, *58*, *69, 70*, *78*, *90*
base_coord,DNAStringSet_OR_NULL-method *(base_coord)*, 40
base_matrix *(base_coord)*, 40
base_matrix,DNAStringSet_OR_NULL-method *(base_coord)*, 40
base_methods *(base_coord)*, 40
base_repl, 44
base_seq2string_set *(base_coord)*, 40

base_seq2string_set,BaseSeq-method (base_coord), 40
BaseGroup, *39*, *42*
BaseGroup (BaseGroup-class), 39
BaseGroup-class, 39
BaseGroup_OR_CodonGroup (BaseGroup_OR_CodonGroup-class), 39
BaseGroup_OR_CodonGroup-class, 39
BaseSeq, *40*, *72, 73*
BaseSeq (BaseSeq-class), 39
BaseSeq-class, 39
BaseSeqMatrix, *57*
BaseSeqMatrix (BaseSeqMatrix-class), 40
BaseSeqMatrix-class, 40
bplapply, *10*, *14*, *20*, *24*, *27*, *31, 32*, *34*, *54*, *57*, *60*, *73*, *86*, *94, 95*
brca1_aln, 44, *45, 46*, *61, 62*
brca1_aln2, *45*, 45, *46*, *62*
brca1_autm, *15*, *45*, 46, *62, 63*
brca1_autm2, *45, 46*, 46, *61–63*

cdm_z64, 47
codon_coord, *11*, *38*, *43*, *48*, 50, *54*, *58*, *69, 70*, *90*
codon_coord,BaseGroup-method (codon_coord), 50
codon_coord,DNAStringSet_OR_NULL-method (codon_coord), 50
codon_coord,matrix_OR_data_frame-method (codon_coord), 50
codon_dist, *10, 11*, 52, *55, 56*
codon_dist,character-method (codon_dist), 52
codon_dist,CodonGroup_OR_Automorphisms-method (codon_dist), 52
codon_dist,DNAStringSet-method (codon_dist), 52
codon_dist_matrix, *47*, *54*, 55, *71*
codon_matrix, *48*, *51*, 56, *76*
codon_matrix,BaseSeqMatrix-method (codon_matrix), 56
codon_matrix,DNAMultipleAlignment-method (codon_matrix), 56
codon_matrix,DNAStringSet-method (codon_matrix), 56
CodonGroup, *39*
CodonGroup (CodonGroup-class), 47
CodonGroup-class, 47
CodonMatrix, *101*
CodonMatrix (CodonMatrix-class), 48
CodonMatrix-class, 48
CodonSeq, *78*, *90*
CodonSeq (CodonSeq-class), 49
CodonSeq-class, 49
colMeans2, *88*
colMeans2 (reexports), 87
colMeans2(), *87*
colSds, *88*
colSds (reexports), 87
colSds(), *87*
colSums2, *88*
colSums2 (reexports), 87
colSums2(), *87*
colVars, *88*
colVars (reexports), 87
colVars(), *87*
conserved_regions, *25*, 59
conserved_regions,Automorphism-method (conserved_regions), 59
conserved_regions,AutomorphismByCoef-method (conserved_regions), 59
conserved_regions,AutomorphismByCoefList-method (conserved_regions), 59
conserved_regions,AutomorphismList-method (conserved_regions), 59
ConservedRegion (ConservedRegion-class), 58
ConservedRegion-class, 58
ConservedRegionList (ConservedRegion-class), 58
ConservedRegionList-class (ConservedRegion-class), 58
coordList (CodonSeq-class), 49
coordList,CodonSeq-method (CodonSeq-class), 49
covid_aln, *15–17*, *45*, *61*, 61, *62, 63*
covid_autm, *45, 46*, *61*, 61, *63*
cyc_aln, *45*, *61*, 62, *63*
cyc_autm, *62*, 63

data.frame, *17, 18*, *22*
DataFrame, *12*, *78*, *89, 90*
DataFrame_OR_data.frame-class (Automorphism-class), 17
dist, *73*
dna_phyche, 64, *66*
dna_phychem, *38*, 65

dna_phychem,character-method
(dna_phychem), 65
dna_phychem,DNAStringSet_OR_DNAMultipleAlignment-method
(dna_phychem), 65
DNAMultipleAlignment, *13*, *24*, *30*, *32*, *34*, *41*, *44*, *45*, *50*, *57*, *61*, *62*, *66*, *69*, *72*, *73*, *78*, *86*, *88*, *89*
DNAMultipleAlignment (reexports), 87
DNAMultipleAlignment(), *87*
DNAStringSet, *8*, *13*, *24*, *30*, *32*, *34*, *40*, *41*, *50*, *57*, *66*, *69*, *72*, *73*, *78*, *86*, *88–90*
DNAStringSet (reexports), 87
DNAStringSet(), *87*
DNAStringSet_OR_DNAMultipleAlignment-class
(GRangesMatrixSeq-class), 74
DNAStringSet_OR_NULL-class
(valid.MatrixList), 100

end, *88*
end (reexports), 87
end(), *87*
end<- (reexports), 87
Extract, *101*
extract
([,AutomorphismList,ANY-method), 100
extract-methods
([,AutomorphismList,ANY-method), 100

GENETIC_CODE_TABLE, *10*, *14*, *31*, *34*, *55*, *87*, *88*
GENETIC_CODE_TABLE (reexports), 87
GenomAutomorphism, 67
GenomAutomorphism-package
(GenomAutomorphism), 67
get_coord, *49*, 69, *70*
get_coord,BaseGroup_OR_CodonGroup-method
(get_coord), 69
get_coord,DNAStringSet_OR_NULL-method
(get_coord), 69
get_mutscore, *5*, *6*, *8*, 71
get_mutscore,BaseSeq,missing-method
(get_mutscore), 71
get_mutscore,character,character-method
(get_mutscore), 71
get_mutscore,DNAMultipleAlignment,missing-method
(get_mutscore), 71
get_mutscore,DNAStringSet,missing-method
(get_mutscore), 71
getAutomorphisms, *25*, 67
getAutomorphisms,AutomorphismList-method
(getAutomorphisms), 67
getAutomorphisms,DataFrame_OR_data.frame-method
(getAutomorphisms), 67
getAutomorphisms,list-method
(getAutomorphisms), 67
getGeneticCode, *14*, *31*, *34*, *88*
getGeneticCode (reexports), 87
getGeneticCode(), *87*
GRanges, *40*, *75*
GRanges-class, *93*
GRanges_OR_NULL-class, 75
GRangesList, *22*, *88*
GRangesList (reexports), 87
GRangesList(), *87*
GRangesMatrixSeq
(GRangesMatrixSeq-class), 74
GRangesMatrixSeq-class, 74

is.url, 76

lapply, *92*
ListCodonMatrix, *57*, *101*
ListCodonMatrix
(ListCodonMatrix-class), 76
ListCodonMatrix-class, 76

makeCluster, *55*, *73*
makeGRangesFromDataFrame, *88*
makeGRangesFromDataFrame (reexports), 87
makeGRangesFromDataFrame(), *87*
matrices, 77, *78*, *90*
matrices,CodonSeq-method (matrices), 77
matrices,DNAStringSet_OR_NULL-method
(matrices), 77
matrices,MatrixList-method (matrices), 77
MatrixList, *70*, *78*
MatrixList (MatrixList-class), 79
MatrixList-class, 79
MatrixSeq, *66*, *75*, *86*, *101*
MatrixSeq (MatrixSeq-class), 80
MatrixSeq-class, 80
mcols, *88*
mcols (reexports), 87
mcols(), *87*

mcols<- (reexports), 87
mean, *10*
mod, 81
mod,matrix,numeric-method (mod), 81
modeq, 82
modlin, *82*, *83*, *88*
modlin (reexports), 87
modlin(), *87*
modlineq, 83
modq, *88*
modq (reexports), 87
modq(), *87*
modulo (mod), 81
MulticoreParam, *20*, *27*, *60*
mut_type, *27*, 84

names,AutomorphismList-method
(AutomorphismList-class), 21
names,ListCodonMatrix-method
([,AutomorphismList,ANY-method), 100
names<-,AutomorphismList-method
(AutomorphismList-class), 21
names<-,MatrixSeq-method
([,AutomorphismList,ANY-method), 100

optim, *29*

peptide_phychem_index, *66*, 85
peptide_phychem_index,character-method
(peptide_phychem_index), 85
peptide_phychem_index,DNAStringSet_OR_DNAMultipleAlignment-method
(peptide_phychem_index), 85

readDNAMultipleAlignment, *88*
readDNAMultipleAlignment (reexports), 87
readDNAMultipleAlignment(), *87*
reexports, 87
rowMeans2, *88*
rowMeans2 (reexports), 87
rowMeans2(), *87*
rowSds, *88*
rowSds (reexports), 87
rowSds(), *87*
rowSums2, *88*
rowSums2 (reexports), 87
rowSums2(), *87*
rowVars, *88*
rowVars (reexports), 87
rowVars(), *87*

sapply, *91*, *92*
scale, *37*
seq2granges (base_coord), 40
seq2granges,DNAStringSet_OR_NULL-method
(base_coord), 40
seqRanges (CodonSeq-class), 49
seqranges, *78*, 89
seqRanges,CodonSeq-method
(CodonSeq-class), 49
seqranges,CodonSeq-method (seqranges), 89
seqranges,DNAStringSet_OR_NULL-method
(seqranges), 89
setValidity2, *88*
setValidity2 (reexports), 87
setValidity2(), *87*
show,AutomorphismList-method
(AutomorphismList-class), 21
show,CodonSeq-method, 91
show,ListCodonMatrix-method
(show,CodonSeq-method), 91
show,MatrixList-method
(show,CodonSeq-method), 91
show,MatrixSeq-method
(MatrixSeq-class), 80
show-AutomorphismList
(AutomorphismList-class), 21
show-CodonSeq (show,CodonSeq-method), 91
show-ListCodonMatrix
(show,CodonSeq-method), 91
show-MatrixList (show,CodonSeq-method), 91
show-MatrixSeq (MatrixSeq-class), 80
slapply, 91
sortByChromAndEnd
(sortByChromAndStart), 93
sortByChromAndStart, 93
start, *88*
start (reexports), 87
start(), *87*
start<- (reexports), 87
str2chr, 93
str2chr,character-method (str2chr), 93
str2chr,list-method (str2chr), 93
str2dig, 94
str2dig,character-method (str2dig), 94

str2dig,list-method *(str2dig)*, 94
strand, *88*
strand *(reexports)*, 87
strand(), *87*
strand<- *(reexports)*, 87
strsplit, *94, 95*
subseq, *88*
subseq *(reexports)*, 87
subseq(), *87*

translate, *88*, *95, 96*
translate *(reexports)*, 87
translate(), *87*
translation, *86*, 95
translation,BioString-method
(translation), 95
translation,character-method
(translation), 95

unmasked, *88*
unmasked *(reexports)*, 87
unmasked(), *87*

valid.Automorphism
(valid.Automorphism.mcols), 97
valid.Automorphism.mcols, 97
valid.AutomorphismByCoef, 97
valid.AutomorphismByCoefList, 98
valid.AutomorphismList, 98
valid.BaseGroup *(valid.BaseGroup.elem)*,
99
valid.BaseGroup.elem, 99
valid.CodonGroup
(valid.CodonGroup.mcols), 99
valid.CodonGroup.mcols, 99
valid.ConservedRegion
(ConservedRegion-class), 58
valid.ConservedRegionList
(ConservedRegion-class), 58
valid.GRanges *(valid.BaseGroup.elem)*, 99
valid.ListCodonMatrix
(ListCodonMatrix-class), 76
valid.MatrixList, 100

width, *88*
width *(reexports)*, 87
width(), *87*