Package 'GenomAutomorphism'

April 10, 2023

Title Compute the automorphisms between DNA's Abelian group representations

Version 1.0.3

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 (>= 4.2),

License Artistic-2.0

Encoding UTF-8

biocViews MathematicalBiology, ComparativeGenomics, FunctionalGenomics, MultipleSequenceAlignment

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

RoxygenNote 7.2.3

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

Roxygen list(markdown = TRUE)

Language en-US

LazyData false

Config/testthat/edition 3

VignetteBuilder knitr

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

git_branch RELEASE_3_16

git_last_commit 4ed0e65

git_last_commit_date 2023-02-24

Date/Publication 2023-04-10

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

Maintainer Robersy Sanchez <genomicmath@gmail.com>

R topics documented:

`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

```
aaindex2
```

Format

`AutomorphismList` class object.

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 mutation 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

```
aaindex3
```

Format

`AutomorphismList` class object.

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

`aaindex3`, `aa_mutmat`, 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_mutmat` *Amino acid mutation matrix*

Description

This returns an amino acid mutation matrix or a statistical protein contact potentials matrix from AAindex (ver.9.2).

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.

Usage

```
aa_mutmat(acc = NA, aaindex = NA, acc_list = FALSE)
```

Arguments

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

Value

A mutation or contact potential matrix, or the list of available matrices ids and index names is returned.

Author(s)

Robersy Sanchez `https://genomaths.com`

See Also

`aaindex2`, `aaindex3`, and `get_mutscore`.

Examples

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

## Get the available mutation matrices
mat <- aa_mutmat(aaindex = aaindex2, acc_list = TRUE)
```

```
mat[1:10]

## 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)
```

`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

```
aln
```

Format

`DNAStringSet` class object.

`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 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

The returned an AutomorphismList-class object.

See Also

automorphism_bycoef, automorphisms

Examples

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

## Let's transforming into a list of Automorphisms-class objects
x1 <- as.list(brca1_autm[1:2])

## Now, object 'x1' is transformed into a 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 a 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 = detectCores() - 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 each other. Such a cube pair are call dual cubes and, as shown in reference (3), each pair integrates 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 a 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 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 a `AutomorphismList` object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of primate somatic cytochrome C grouped by automorphism's coefficients. The grouping derives from the dataset `brca1_autm` after applying function `automorphism_bycoef`.

Usage

```
autby_coef
```

Format

`AutomorphismByCoefList` class object.

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

Description

This is a `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: MG772933.1:265-1345513455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named `covid_aln` and the automorphisms were estimated with function `autZ64`.

Usage

```
autm
```

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 a `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: MG772933.1:265-1345513455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named `covid_aln` and the automorphisms were estimated with function `aut3D`.

Usage

```
autm_3d
```

Format

`AutomorphismList` class object.

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

Description

This is a `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: MG772933.1:265-1345513455-21542), nonstructural_polyprotein. The pairwise DNA sequence alignment is available in the dataset named `covid_aln` and the automorphisms were estimated with function `autZ125`.

Usage

```
autm_z125
```

Format

`AutomorphismList` class object.

`automorphismByRanges` *Get the automorphisms by ranges.*

Description

Automorphisms estimated on a pairwise or a 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 = detectCores() - 1,
  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* automorphims.

Examples

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

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

`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 = detectCores() - 1,
  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 each other. Such a cube pair are call $dualcubes$ and, as shown in reference (3), each pair integrates 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 $f(x) = kxmod64$ and $f(x) = kxmod125$, where k and x are elements from the set of integers modulo 64 or modulo 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 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 inherits 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) 110-152.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` *Autmorphism 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 = detectCores() - 1,
  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. 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])
```

`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 = detectCores() - 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 each other. Such a cube pair are call dual cubes and, as shown in reference (3), each pair integrates 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 a 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 $f(x) = kx mod 64$, where k and x are elements from the set of integers modulo 64. 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) 110-152.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 = detectCores() - 1,
  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 each other. Such a cube pair are call dual cubes and, as shown in reference (3), each pair integrates 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 $f(x) = kx mod 64$, where k and x are elements from the set of integers modulo 64. 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) 110-152.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 = detectCores() - 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 each other. Such a cube pair are call dual cubes and, as shown in reference (3), each pair integrates 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 a 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) 110-152.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 split 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 the a DNA sequence.
...	Not in use.

Details

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

Value

If the argument of 'x' is character string, then a character vector of codons will 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 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) or 5 (Z5).

Usage

```
base2int(base, ...)

## S4 method for signature 'character'
base2int(
  base,
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3"),
 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")
)

## S4 method for signature 'data.frame'
base2int(
  base,
  group = c("Z4", "Z5", "Z64", "Z125", "Z4^3", "Z5^3"),
 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")
)
```

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).

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` and `codon_coord`.

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 must carry only single letters

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

`base_coord` *Base coordinates on a given 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

```
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 = "+"
)
```

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 2 3).
`group`	A character string denoting the group representation for the given base or codon as shown in reference (1).
`...`	Not in use.
`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 `BaseGroup` 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 of integer numbers ("Z") is also possible but they are not defined to integrate a Abelian group. These are just used for the further insertion the codon set in the 3D space (R^3).

Value

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`.

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
```

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

Description

This is a `DNAMultipleAlignment` carrying a 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

```
brca1_aln
```

Format

`DNAMultipleAlignment` class object.

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

Description

This is a `DNAMultipleAlignment` carrying a 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

```
brca1_aln2
```

Format

`DNAMultipleAlignment` class object.

`brca1_autm` *Automorphisms between DNA Sequences from Primate BRCA1 Genes*

Description

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

Usage

```
brca1_autm
```

Format

`AutomorphismList` class object.

`brca1_autm2` *Automorphisms between DNA Sequences from Primate BRCA1 Genes*

Description

This is a `AutomorphismList` object carrying a list of pairwise automorphisms between the DNA sequences from the MSA of primate BRCA1 DNA repair gene. The data set brca1_aln2 has 41 DNA sequences and it contains the previous 20 primate variants found in 'braca1_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

```
brca1_autm2
```

Format

`AutomorphismList` class object.

Details

The automorphisms were estimated from the `brca1_aln` MSA with function `autZ64`.

`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

```
cdm_z64
```

Format

A list object.

Examples

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

## 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: 0.0625.
head(cdm_z64[[ "TCGA" ]], 20)
```

`codon_coord` *Codon coordinates on a given 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.

`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 $x = (y_1, y_2, y_3)$ (see (1)), the Weighted Manhattan distance between this 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 define 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 gaps simbol "-").
```

```
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"),
  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 dowstream 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[61:63]
```

`conserved_regions` *Conserved and Non-conserved Regions from a MSA*

Description

Returns the Conserved or the Non-conserved Regions from a 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 = detectCores() - 1,
  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`	A `Automorphism-class`, a `AutomorphismList-class`, a `AutomorphismByCoef` or a `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

A `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: MG772933.1:265-1345513455-21542), complete genomes. The alignment is available at GitHub: `https://github.com/genomaths/seqalignments/tree/master/COVID-19`

Usage

```
covid_aln
```

Format

`DNAMultipleAlignment` class object.

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

Description

This is a `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

```
covid_autm
```

Format

`AutomorphismList` class object.

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

Description

This is a `DNAMultipleAlignment` carrying a 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

```
cyc_aln
```

Format

`DNAMultipleAlignment` class object.

`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

```
cyc_autm
```

Format

`AutomorphismList` class object.

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

Description

This is a 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*.

References

1. Robersy Sanchez, Jesus Barreto (2021) Genomic Abelian Finite Groups. `https://doi.org/10.1101/2021.06.01.446543`.
2. 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.
3. Sanchez R, Grau R, Morgado E. A novel Lie algebra of the genetic code over the Galois field of four DNA bases. Math Biosci. 2006;202: 156-174. doi:10.1016/j.mbs.2006.03.017
4. Sanchez R, Grau R. An algebraic hypothesis about the primeval genetic code architecture. Math Biosci. 2009/07/18. 2009;221: 60-76. `https://doi.org/10.1016/j.mbs.2009.07.001`.
5. 5. 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.
6. . 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.

`getAutomorphisms` *Get Automorphisms*

Description

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

An `AutomorphismList-class`

An `Automorphism-class`

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 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 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 the way to get the coordinates in a numeric object in object from and still to preserve the base/codon sequence information.

Value

An object from `CodonGroup-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 `CodonGroup-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 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 score are retrieve from a mutation matrix or a statistical protein contact potentials matrix from AAindex (ver.9.2).

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,
  ...
)
```

Arguments

`aa1, aa2`	A simple character representing an amino acids or a character string of letter from the amino acid alphabet or base-triplets from the DNA/RNA alphabet.
`...`	Not in use.
`acc`	Accession id for a specified mutation or contact potential matrix.
`aaindex`	Database where the requested accession id is locate. 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`	If TRUE, prints the function log to stdout.

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.

Author(s)

Robersy Sanchez `https://genomaths.com`

See Also

`aa_mutmat`, `aaindex2` and `aaindex3`

Examples

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

## 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")
```

`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 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 the 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

The 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",
)
```

`mod` *Modulo Operation*

Description

Integer remainder of the division of the integer n by m: n mod m. This function extend the application of function `numbers` to matrices where the operation on each row is with is accomplish with a different values of m, i.e, where m is a vector.

Usage

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

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

Arguments

`n`	A matrix where each element can be reduced to integers or the same as in `numbers`.
`m`	As in `numbers`.
`...`	Not in use yet.

Value

An element of x, an `Automorphism-class` object.

Author(s)

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

Examples

```
## 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
```

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

Description

If a, b, and c are integer vectors, this function try to find, at each coordinate, the solution of the MLE $ax = b$ mod n. If the MLE $ax = bmodn$ has not solutions (see `modlin`), the value reported for the coordinate will be 0 and the corresponding translation.

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 c 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 not exact solution for some coordinate, the 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 revovered with the transformaiton
(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). 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 as according to the type of involved in it (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")
```

`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 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 provide 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 the a DNA sequence alignment from a file.

Value

A `GRanges-class`

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,
)
```

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

Description

This function apply 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 = 1:10, beta = exp(-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 string

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

A integer vector or a list of integer vectors.

Author(s)

Robersy Sanchez `https://genomaths.com`

Examples

```
## A 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 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' belong to any of the classes admitted by function `translate`, then this function is called to make the translation.

Value

The translated amino acid sequence.

Author(s)

Robersy Sanchez `https://genomaths.com`

See Also

`translate`

Examples

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

translation(aln)

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

translation(covid_aln)
```

Index