Package ‘SynMut’

April 1, 2025

Type Package

Title SynMut: Designing Synonymously Mutated Sequences with Different Genomic Signatures

Version 1.22.0

Description There are increasing demands on designing virus mutants with specific dinucleotide or codon composition.
This tool can take both dinucleotide preference and/or codon usage bias into account while designing mutants.
It is a powerful tool for in silico designs of DNA sequence mutants.

License GPL-2

Encoding UTF-8

Suggests BiocManager, knitr, rmarkdown, testthat, devtools, prettydoc, glue

VignetteBuilder knitr

Imports seqinr, methods, Biostrings, stringr, BiocGenerics

RoxygenNote 7.1.0

Collate 'regioned_dna_Class.R' 'codon_mimic.R' 'input_seq.R' 'codon_random.R' 'codon_to.R' 'dinu_to.R' 'distance_analysis.R' 'region_related.R' 'seq_random.R' 'zzz.R'

biocViews SequenceMatching, ExperimentalDesign, Preprocessing

BugReports `https://github.com/Koohoko/SynMut/issues`

URL `https://github.com/Koohoko/SynMut`

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

git_branch RELEASE_3_20

git_last_commit 30b8575

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-03-31

Author Haogao Gu [aut, cre],
Leo L.M. Poon [led]

Maintainer Haogao Gu <`hggu@connect.hku.hk`>

Contents

`codon_dist` *Calculating the codon usage difference between sequences*

Description

We use a least squares approach to estimate the codon usage difference between DNA sequences.

Usage

```
codon_dist(seq, ref)

## S4 method for signature 'ANY'
codon_dist(seq, ref)
```

Arguments

`seq`	the input DNA sequnece of `DNAStringSet` or `regioned_dna` class.
`ref`	the reference DNA sequnece of `DNAStringSet` or `regioned_dna` class.

Details

idea inspired by "Daniel Macedo de Melo Jorge, Ryan E. Mills, Adam S. Lauring, CodonShuffle: a tool for generating and analyzing synonymously mutated sequences, Virus Evolution, Volume 1, Issue 1, March 2015, vev012, https://doi.org/10.1093/ve/vev012"

Value

vector

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_cu(rgd.seq)

mut.seq <- codon_random(rgd.seq)
codon_dist(mut.seq, rgd.seq)
mut.seq2 <- codon_random(rgd.seq, keep = TRUE)
codon_dist(mut.seq2, rgd.seq)
```

---

codon_mimic *Mimic a target codon usage bias*

Description

Mutating the current DNA sequences in the regioned_dna object to mimic a target codon usage pattern.

Usage

```
codon_mimic(object, alt, numcode = 1, ...)

## S4 method for signature 'regioned_dna,vector'
codon_mimic(object, alt, numcode)

## S4 method for signature 'regioned_dna,DNAStringSet'
codon_mimic(object, alt, numcode)

## S4 method for signature 'DNAStringSet,DNAStringSet'
codon_mimic(object, alt, numcode)
```

Arguments

`object`	regioned_dna object
`alt`	target codon usage vector or DNAStringSet object representing target codon usage
`numcode`	The ncbi genetic code number for translation. Default value: `1`. Details please refer to `?seqinr::translate` ("https://rdrr.io/cran/seqinr/man/translate.html").
`...`	...

Details

The ideas for `codon_mimic` is similar to `codon_to`: first extract the mutable regions and then do the mutation. However the codons in the fixed (not mutable) regions will also alter the final codon usage, thus we have to adjust for the fixed codons when introducing sysnonymous codons. The ideal deisgn for `codon_mimic` is not unique as the swap between positions of the synonymous codons will not change the codon usage bias.

Details pleas refer to: https://koohoko.github.io/SynMut/algorithm.html

Value

regioned_dna

See Also

`input_seq`, `codon_to`, `codon_random`, `dinu_to`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
target <- get_cu(rgd.seq)[2,]
new <- codon_mimic(rgd.seq, alt = target)
get_cu(new) - get_cu(rgd.seq)

target <- Biostrings::DNAStringSet("TTGAAAA-CTC-N--AAG")
new <- codon_mimic(rgd.seq, alt = target)
get_cu(new) - get_cu(rgd.seq)
get_freq(new) - get_freq(rgd.seq)
get_rscu(new) - get_rscu(rgd.seq)
```

codon_random *Generate random synonymous mutations*

Description

Generating random synonymous mutations (in user-defined region), with optionally keeping/not keeping the original codon usage bias.

Usage

```
codon_random(object, n = 1, keep = FALSE, numcode = 1, ...)

## S4 method for signature 'regioned_dna'
codon_random(object, n, keep, numcode)

## S4 method for signature 'DNAStringSet'
codon_random(object, n, keep, numcode)
```

Arguments

`object`	A regioned_dna object.
`n`	Optional n parameter specifying what proportion of the codons to be mutated. Default value: `1`.
`keep`	Logical parameter controling whether keeping the codon usage bias
`numcode`	The ncbi genetic code number for translation. Default value: `1`. Details please refer to `?seqinr::translate` ("https://rdrr.io/cran/seqinr/man/translate.html").
`...`	...

Details

This method randomly sample synonymous codons for n propotion of every mutable codons in the sequences. This process will be likely to alter the codon usage bias of the original sequences. However the keep = TRUE argument help to preserve the codon usage bias. It is done via the synsequence function in seqinr package. The synsequence function essentially swaps the position of the synonymous codons without introducing new codons into the original sequences.

Value

A regioned_dna object containing the mutants; Or a DNAStringSet object if the input is a DNAStringSet object.

See Also

input_seq, dinu_to, codon_to, codon_mimic

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
set.seed(2019)
get_cu(codon_random(rgd.seq, n = 0.5))
get_cu(codon_random(rgd.seq))
```

---

codon_to *Maximize or minimize the usage of certain codon.*

Description

Input string of a codon to either the "max.codon = " or "min.codon = " parameter to maximize or minimize the usage of certain codon in the sequence.

Usage

```
codon_to(object, max.codon = NA, min.codon = NA, ...)

## S4 method for signature 'regioned_dna'
codon_to(object, max.codon, min.codon)
```

Arguments

object	A regioned_dna object.
max.codon	A string of a codon.
min.codon	A string of a codon.
...	...

Details

The ideas for this function is simple. We first extract the mutable regions for every sequences, then mutated the synonymous codons of the input to the desired. There will be only one ideal design for the maximization problem, however there may be numerous comparable designs having the same minimal usage of certain codon, as we randomly sample synonymous codon for substitution when solving the minimization problem.

Value

A regioned_dna object.

See Also

input_seq, dinu_to, codon_random, codon_mimic

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_cu(codon_to(rgd.seq, max.codon = "AAC")) - get_cu(rgd.seq)
get_cu(codon_to(rgd.seq, min.codon = "AAC")) - get_cu(rgd.seq)
```

---

dinu_dist *Calculating the dinucleotide usage difference between sequences*

Description

We use a least squares approach to estimate the dinucleotide usage difference between DNA sequences

Usage

```
dinu_dist(seq, ref)

## S4 method for signature 'ANY'
dinu_dist(seq, ref)
```

Arguments

`seq`	the input DNA sequnece of `DNAStringSet` or `regioned_dna` class.
`ref`	the reference DNA sequnece of `DNAStringSet` or `regioned_dna` class.

Details

similar method that applied in "Daniel Macedo de Melo Jorge, Ryan E. Mills, Adam S. Lauring, CodonShuffle: a tool for generating and analyzing synonymously mutated sequences, Virus Evolution, Volume 1, Issue 1, March 2015, vev012, https://doi.org/10.1093/ve/vev012"

Value

vector

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_cu(rgd.seq)

mut.seq <- codon_random(rgd.seq)
dinu_dist(mut.seq, rgd.seq)
```

dinu_to *Maximize or minimize the usage of certain dinucleotide.*

Description

Input string of a dinucleotide to either the "max.dinu = " or "min.codon = " parameter to maximize or minimize the usage of certain codon in the sequence. Using a greedy algorithm with priority given to dinucleotide12 or dinucleotide23.

Usage

```
dinu_to(object, max.dinu = NA, min.dinu = NA, keep = FALSE, numcode = 1, ...)

## S4 method for signature 'regioned_dna'
dinu_to(object, max.dinu, min.dinu, keep, numcode)
```

Arguments

`object`	A regioned_dna object.
`max.dinu`	A string of a dinucleotide.
`min.dinu`	A string of a dinucleotide.
`keep`	A logical varibale stating if the codon usage of the original sequences should be keep. Default: False.
`numcode`	The ncbi genetic code number for translation. Default value: `1`. Details please refer to `?seqinr::translate` ("https://rdrr.io/cran/seqinr/man/translate.html").
`...`	...

Details

The detail strategy for this function please refer to: https://koohoko.github.io/SynMut/algorithm.html

Value

regioned_dna

See Also

`input_seq`, `codon_to`, `codon_random`, `codon_mimic`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_du(dinu_to(rgd.seq, max.dinu = "cg")) - get_du(rgd.seq)
get_du(dinu_to(rgd.seq, min.dinu = "AA")) - get_du(rgd.seq)
get_du(dinu_to(rgd.seq, max.dinu = "cg", keep = TRUE)) - get_du(rgd.seq)
get_cu(dinu_to(rgd.seq, max.dinu = "CG", keep = TRUE)) - get_cu(rgd.seq)
```

get_cu *Get codon usage matrix*

Description

Access the codon usage matrix

Usage

```
get_cu(object, ...)

## S4 method for signature 'regioned_dna'
get_cu(object)

## S4 method for signature 'DNAStringSet'
get_cu(object)
```

Arguments

`object`	regioned_dna / DNAStringSet
`...`	...

Value

matrix

See Also

`input_seq`, `get_region`, `get_nu`, `get_du`, `get_freq`, `get_rscu`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_cu(rgd.seq)
```

get_dna *Get the DNAStringSet data*

Description

Access the DNA sequence data in DNAStringSet.

Usage

```
get_dna(object, ...)

## S4 method for signature 'regioned_dna'
get_dna(object)
```

Arguments

`object`	A regioned_dna object.
`...`	...

Value

DNAStringSet

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_dna(rgd.seq)
```

`get_du` *Get dinucleotide usage matrix*

Description

Access the dinucleotide usage matrix

Usage

```
get_du(object, ...)

## S4 method for signature 'regioned_dna'
get_du(object)

## S4 method for signature 'DNAStringSet'
get_du(object)
```

Arguments

`object`	regioned_dna / DNAStringSet
`...`	...

Value

matrix

See Also

`input_seq`, `get_region`, `get_nu`, `get_cu`, `get_freq`, `get_rscu`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_du(rgd.seq)
```

get_freq — *Get codon usage frequency of synonymous codons*

Description

Access the synonymous codon usage frequency

Usage

```
get_freq(object, numcode = 1, ...)

## S4 method for signature 'regioned_dna'
get_freq(object, numcode)

## S4 method for signature 'DNAStringSet'
get_freq(object, numcode)

## S4 method for signature 'matrix'
get_freq(object, numcode)

## S4 method for signature 'vector'
get_freq(object, numcode)
```

Arguments

`object`	regioned_dna / DNAStringSet / codon usage matrix (vector)
`numcode`	The ncbi genetic code number for translation. Default value: `1`. Details please refer to `?seqinr::translate` ("https://rdrr.io/cran/seqinr/man/translate.html").
`...`	...

Value

matrix

See Also

`input_seq`, `get_region`, `get_cu`, `get_du`, `get_rscu`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_freq(rgd.seq)
```

get_nu *Get nucleotide usage matrix*

Description

Access the nucleotide usage matrix

Usage

```
get_nu(object, ...)

## S4 method for signature 'regioned_dna'
get_nu(object)

## S4 method for signature 'DNAStringSet'
get_nu(object)
```

Arguments

`object`	regioned_dna / DNAStringSet
`...`	...

Value

matrix

See Also

`input_seq`, `get_region`, `get_cu`, `get_du`, `get_rscu`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_nu(rgd.seq)
```

get_region *Get the variable region*

Description

Access the variable regions

Usage

```
get_region(object, ...)

## S4 method for signature 'regioned_dna'
get_region(object)
```

Arguments

`object`	regioned_dna
`...`	...

Value

list

See Also

`input_seq`, `get_cu`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_region(rgd.seq)
```

`get_rscu` *Get Relative Synonymous Codon Usage (rscu) of synonymous codons*

Description

Access the Relative Synonymous Codon Usage rscu

Usage

```
get_rscu(object, numcode = 1, ...)

## S4 method for signature 'regioned_dna'
get_rscu(object, numcode)

## S4 method for signature 'DNAStringSet'
get_rscu(object, numcode)
```

Arguments

`object`	regioned_dna / DNAStringSet / codon usage matrix (vector)
`numcode`	The ncbi genetic code number for translation. Default value: `1`. Details please refer to ?`seqinr::translate` ("https://rdrr.io/cran/seqinr/man/translate.html").
`...`	...

Value

matrix

See Also

`input_seq`, `get_region`, `get_cu`, `get_du`, `get_freq`

Examples

```
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)
get_rscu(rgd.seq)
```

`input_seq` *Import region / constructing regioned_dna object*

Description

Constructing `regioned_dna` from DNAStringSet. Optionally input a `region` data.frame to define restricted amino-acid region for mutation.

Usage

```
input_seq(object, region = NA, ...)

## S4 method for signature 'character'
input_seq(object, region)

## S4 method for signature 'DNAStringSet'
input_seq(object, region)

## S4 method for signature 'DNAString'
input_seq(object, region)
```

Arguments

`object`	Filepath or DNAstringSet. The input sequences is suggested to be in open reading frame(ORF).
`region`	`NA`. A data.frame specifying paticular regions (positions in amino acid sequence) that is allowed to be mutated in the sequences. Both `1` / `0` or `TRUE` / `FALSE` encoding is OK. Please refer to Examples below for reference.
`...`	...

Value

A regioned_dna-class object

See Also

`get_cu`, `get_du`, `get_region`, `get_dna`

Examples

```
# Creating a input_seq class directly from system file
filepath <- system.file("extdata", "example.fasta", package = "SynMut")
rgd.seq <- input_seq(filepath)

# Optionally input with region dataframe
filepath.fasta <- system.file("extdata", "example.fasta", package = "SynMut")
```

```
fp.csv <- system.file("extdata", "target_regions.csv", package = "SynMut")
region <- read.csv(fp.csv)
rgd.seq <- input_seq(filepath.fasta, region)

# Creating from exsisting DNAStringSet object
seq <- Biostrings::DNAStringSet("ATCGATCGA")
rgd.seq <- input_seq(seq)
```

regioned_dna-class *An S4 class to record DNA sequences and variable regions for mutations*

Description

Recording codon DNA sequences and region.

Slots

`dnaseq` a DNAStingSet object recording the sequence(s)

`region` a list specifying paticular regions in the sequences allowed to be mutated

Author(s)

Haogao Gu

See Also

`input_seq`, `get_cu`, `get_region`

seq_random *Generate* `n` *random DNA sequnces of length* `m`

Description

Generate `n` random DNA sequnces of length `m`, optional exclude stop codons.

Usage

```
seq_random(n = 1, m, no.stop.codon = FALSE, ...)

## S4 method for signature 'numeric,numeric'
seq_random(n, m, no.stop.codon)
```

Arguments

`n`	the number of the output sequence(s).
`m`	the length of the ouput sequence(s). Eighter a fixed number or a vector of different numbers.
`no.stop.codon`	Default FALSE. If TRUE, the stop codons in the frame 1 would be substituted to another random codon.
`...`	...

Value

a DNAStringSet object

Examples

```
seq_random(n = 1, m = 99)
seq_random(n = 10, m = 30)
seq_random(n = 10, m = 1:10)
seq.nsc <- seq_random(n = 10, m = 100, no.stop.codon = TRUE)
get_cu(seq.nsc)
```

Index