Package ‘bioseq’

October 12, 2022

Type Package
Title A Toolbox for Manipulating Biological Sequences
Version 0.1.4
Maintainer Francois Keck <francois.keck@gmail.com>
Description Classes and functions to work with biological sequences (DNA, RNA and amino acid sequences). Implements S3 infrastructure to work with biological sequences as described in Keck (2020) <doi:10.1111/2041-210X.13490>. Provides a collection of functions to perform biological conversion among classes (transcription, translation) and basic operations on sequences (detection, selection and replacement based on positions or patterns). The package also provides functions to import and export sequences from and to other package formats.
License GPL-3
URL https://fkeck.github.io/bioseq/
BugReports https://github.com/fkeck/bioseq/issues
Encoding UTF-8
LazyData true
Depends R (>= 3.1.0)
Imports methods, vctrs, tibble, ape, crayon, dplyr, pillar, stringi, stringr, stringdist, readr, rlang
Suggests knitr, rmarkdown, testthat (>= 2.1.0), covr
VignetteBuilder knitr
RoxygenNote 7.2.1
NeedsCompilation no
Author Francois Keck [aut, cre, cph] (https://orcid.org/0000-0002-3323-4167)
Repository CRAN
Date/Publication 2022-09-06 11:40:18 UTC
### R topics documented:

- `aa` .................................................. 3
- `aliview` ............................................. 4
- `alphabets` .......................................... 4
- `as-tibble-ape` ....................................... 5
- `as-tibble-bioseq` ................................... 6
- `as_aa` ................................................ 7
- `as_AAbin` ............................................ 7
- `as_AAbin.tbl_df` ..................................... 8
- `as_dna` ................................................ 8
- `as_DNAbin` .......................................... 9
- `as_DNAbin.tbl_df` ................................. 9
- `as_rna` ............................................... 10
- `as_seqinr_alignment` .............................. 10
- `dic_genetic_codes` .................................. 11
- `dna` .................................................. 11
- `fragilaria` .......................................... 12
- `genetic-codes` ...................................... 12
- `is_aa` ................................................. 13
- `is_dna` .............................................. 14
- `is_rna` ............................................... 15
- `new_aa` ............................................. 15
- `new_dna` ............................................ 16
- `new_rna` ............................................ 16
- `read_fasta` ......................................... 16
- `rev_complement` .................................... 17
- `rna` .................................................. 18
- `seaview` ............................................. 18
- `seq-replace` ........................................ 19
- `seq_cluster` ....................................... 20
- `seq_combine` ....................................... 21
- `seq_consensus` ..................................... 22
- `seq_count_pattern` ................................ 23
- `seq_crop_pattern` ................................ 25
- `seq_crop_position` ............................... 26
- `seq_detect_pattern` ............................... 27
- `seq_disambiguate_IUPAC` ....................... 29
- `seq_extract_pattern` ......................... 30
- `seq_extract_position` ......................... 31
- `seq_nchar` .......................................... 32
- `seq_nseq` .......................................... 32
- `seq_remove_pattern` ............................ 33
- `seq_remove_position` ........................... 34
- `seq_replace_position` ........................... 35
- `seq_rev_translate` .............................. 36
- `seq_spellout` ...................................... 37
- `seq_split_kmer` ................................... 38
Build an amino acid (AA) vector

Description

`aa()` build a AA vector from a character vector.

Usage

```r
aa(...)  
```

Arguments

`...` character to turn into AA. Can be a set of name-value pairs.

Value

vector of class `bioseq_aa`

See Also

Other classes: `dna()`, `rna()`

Examples

```r
aa("AGGTGC", "TTCGA")
aa(Seq_1 = "AGGTGC", Seq_2 = "TTCGA")
x <- c("AGGTGC", "TTCGA")
aa(x)
```
Description

This function uses AliView (Larsson, 2014) to visualize DNA sequences. The software must be installed on the computer.

Usage

```r
aliview(
  x,
  aliview_exec = getOption("bioseq.aliview.exec", default = "aliview")
)
```

Arguments

- `x` a DNA, RNA or AA vector. Alternatively a DNAbin or AAbin object.
- `aliview_exec` a character string giving the path of the program.

Details

By default, the function assumes that the executable is installed in a directory located on the PATH. Alternatively the user can provide an absolute path to the executable (i.e. the location where the software was installed/uncompressed). This information can be stored in the global options settings using `options(bioseq.aliview.exec = "my_path_to_aliview")`.

References


See Also

Other GUI wrappers: `seaview()`

Alphabets

**Biological alphabets**

**Description**

List of the allowed characters for each type of sequences.

**DNA**

```
ACGTWSMKRYBDHVN-
```
**RNA**

\[
ACGUWSMKRYBDHVN-
\]

**AA**

\[
ACDEFGHIKLMNPQRSTVWXYZUO*-
\]

**References**


---

**as-tibble-ape**  
*Convert DNAbin/AAbin to tibble*

**Description**

These methods convert sequences from **ape** formats DNAbin and AAbin to tibbles.

**Usage**

```r
as_tibble.DNAbin(x, label = "label", sequence = "sequence", ...)

as_tibble.AAbin(x, label = "label", sequence = "sequence", ...)
```

**Arguments**

- `x` a DNAbin or AAbin object.
- `label` Name of the column that stores the sequence labels in the returned tibble.
- `sequence` Name of the column that stores the sequences in the returned tibble.
- `...` Not used.

**Value**

A tibble with two columns (if name is not NULL, the default) or one column (otherwise).

**See Also**

Other conversions: `as-tibble-bioseq`, `as_AAbin()`, `as_DNAbin()`, `as_aa()`, `as_dna()`, `as_rna()`, `as_seqinr_alignment()`
as-tibble-bioseq

Convert bioseq DNA, RNA and AA to tibble

Description
Convert bioseq DNA, RNA and AA to tibble

Usage
as_tibble.bioseq_dna(x, label = "label", sequence = "sequence", ...)

as_tibble.bioseq_rna(x, label = "label", sequence = "sequence", ...)

as_tibble.bioseq_aa(x, label = "label", sequence = "sequence", ...)

Arguments
  x        a DNA, RNA or AA vector.
  label    Name of the column that stores the sequence labels in the returned tibble.
  sequence Name of the column that stores the sequences in the returned tibble.
  ...      Not used.

Value
A tibble with two columns (if name is not NULL, the default) or one column (otherwise).

See Also
Other conversions: as-tibble-ape, as_AAbin(), as_DNAbin(), as_aa(), as_dna(), as_rna(), as_seqinr_alignment()

Examples
require(tibble)
x <- dna(A = "ACGTTAGTGTACGT", B = "CTCGAAATGA", C = NA)
as_tibble(x)
Coercion to an amino acid (AA) vector

Description
Coercion to an amino acid (AA) vector

Usage
as_aa(x)

Arguments
x An object to coerce.

Value
An amino acid vector of class bioseq_aa

See Also
Other conversions: as-tibble-ape, as-tibble-bioseq, as_AAbin(), as_DNAbin(), as_dna(), as_rna(), as_seqinr_alignment()

Coerce to AAbin

Description
Coerce to AAbin

Usage
as_AAbin(x, ...)

Arguments
x An object.
... Other parameters.

Value
An AAbin object.

See Also
Other conversions: as-tibble-ape, as-tibble-bioseq, as_DNAbin(), as_aa(), as_dna(), as_rna(), as_seqinr_alignment()
as_tbl_df

Coerce tibble to AAbin

Description
Coerce tibble to AAbin

Usage
```r
## S3 method for class 'tbl_df'
as_AAbin(x, sequences, labels = NULL, ...)
```

Arguments
- `x` a tibble.
- `sequences` Name of the tibble column that stores the sequences.
- `labels` Name of the tibble column that stores the sequence labels.
- `...` Other params.

Value
An AAbin object.

as_dna

Coercion to DNA vector

Description
Coercion to DNA vector

Usage
```r
as_dna(x)
```

Arguments
- `x` An object to coerce.

Value
A DNA vector of class bioseq_dna

See Also
Other conversions: `as-tibble-ape`, `as-tibble-bioseq`, `as_AAbin()`, `as_DNAbin()`, `as_aa()`, `as_rna()`, `as_seqinr_alignment()`
as_DNAbin

Coerce to DNAbin

Description

Coerce to DNAbin

Usage

as_DNAbin(x, ...)

Arguments

x  An object.
... Other parameters.

Value

A DNAbin object.

See Also

Other conversions: as-tibble-ape, as-tibble-bioseq, as_AAbin(), as_aa(), as_dna(), as_rna(), as_seqinr_alignment()

as_DNAbin.tbl_df

Coerce tibble to DNAbin

Description

Coerce tibble to DNAbin

Usage

## S3 method for class 'tbl_df'
as_DNAbin(x, sequences, labels = NULL, ...)

Arguments

x  a tibble.
sequences Name of the tibble column that stores the sequences.
labels   Name of the tibble column that stores the sequence labels.
...      Other params.

Value

A DNAbin object.
### as_rna

**Coercion to RNA vector**

**Description**

Coercion to RNA vector

**Usage**

```r
as_rna(x)
```

**Arguments**

- `x`  
  An object to coerce.

**Value**

A RNA vector of class `bioseq_rna`

**See Also**

Other conversions: `as-tibble-ape`, `as-tibble-bioseq`, `as_AAbin()`, `as_DNAbin()`, `as_aa()`, `as_dna()`, `as_seqinr_alignment()`

### as_seqinr_alignment

**Coerce to seqinr alignment**

**Description**

Coerce to seqinr alignment

**Usage**

```r
as_seqinr_alignment(x, ...)
```

**Arguments**

- `x`  
  An object.

- `...`  
  Other parameters.

**Value**

An alignment object.

**See Also**

Other conversions: `as-tibble-ape`, `as-tibble-bioseq`, `as_AAbin()`, `as_DNAbin()`, `as_aa()`, `as_dna()`, `as_rna()`
**dic_genetic_codes**

*Genetic code tables*

**Description**

The function returns a list of named vectors with Start, Stop and Full_name attributes.

**Usage**

```
dic_genetic_codes()
```

**Value**

A list of genetic code tables for DNA/RNA translation.

---

**dna**

*Build a DNA vector*

**Description**

`dna()` build a DNA vector from a character vector.

**Usage**

```
dna(...)  
```

**Arguments**

```
...       
```

characters to turn into DNA. Can be a set of name-value pairs.

**Value**

a vector of class `bioseq_dna`

**See Also**

Other classes: `aa()`, `rna()`

**Examples**

```
dna("AGGTGC", "TTCGA")

dna(Seq_1 = "AGGTGC", Seq_2 = "TTCGA")

x <- c("AGGTGC", "TTCGA")
dna(x)
```
**Description**

An unparsed FASTA of DNA sequences (rbcL) for various strains of Fragilaria retrieved from NCBI.

**Usage**

fragilaria

**Format**

A long character vector (unparsed FASTA).

**Source**

GenBank [https://www.ncbi.nlm.nih.gov/genbank/] using the following search term: "(rbcl) AND Fragilaria"

**See Also**

read fasta to parse these data.

---

**Available genetic codes**

1. Standard
2. Vertebrate Mitochondrial
3. Yeast Mitochondrial
4. Mold Mitochondrial; Protozoan Mitochondrial; Coelenterate Mitochondrial; Mycoplasma; Spiroplasma
5. Invertebrate Mitochondrial
6. Ciliate Nuclear; Dasycladacean Nuclear; Hexamita Nuclear
9. Echinoderm Mitochondrial; Flatworm Mitochondrial
10. Euplotid Nuclear

---

**Description**

List of all genetic code tables available in bioseq. The number in bold can be used to select a table in appropriate functions.
is_aa

11. Bacterial, Archaeal and Plant Plastid
12. Alternative Yeast Nuclear
13. Ascidian Mitochondrial
14. Alternative Flatworm Mitochondrial
15. Blepharisma Macronuclear
16. Chlorophycean Mitochondrial
21. Trematode Mitochondrial
22. Scenedesmus obliquus Mitochondrial
23. Thraustochytrium Mitochondrial
24. Pterobranchia Mitochondrial
25. Candidate Division SR1 and Gracilibacteria
26. Pachysolen tannophilus Nuclear
27. Karyorelict Nuclear
28. Condyllostoma Nuclear
29. Mesodinium Nuclear
30. Peritrich Nuclear
31. Blastocrithidia Nuclear
32. Balanophoraceae Plastid
33. Cephalodiscidae Mitochondrial

References

---

is_aa

Test if the object is an amino acid vector

Description
This function returns TRUE for objects of class bioseq_aa

Usage
is_aa(x)

Arguments
x An object.
is_dna

Value

Logical.

Examples

```r
x <- c("AGGTGC", "TTCGA")
is_aa(x)
y <- aa(x)
is_aa(x)
```

Description

This function returns TRUE for objects of class bioseq_dna

Usage

```r
is_dna(x)
```

Arguments

x

An object.

Value

Logical.

Examples

```r
x <- c("AGGTGC", "TTCGA")
is_dna(x)
y <- dna(x)
is_dna(y)
```
is_rna

Test if the object is a RNA vector

Description

This function returns TRUE for objects of class bioseq_rna

Usage

is_rna(x)

Arguments

x  An object.

Value

Logical.

Examples

x <- c("AGGTGC", "TTCGA")
is_rna(x)
y <- rna(x)
is_rna(x)

new_aa

Amino acid (AA) vector constructor

Description

Amino acid (AA) vector constructor

Usage

new_aa(x = character())

Arguments

x  a character vector.
**new_dna**

*DNA vector constructor*

**Description**

DNA vector constructor

**Usage**

```r
new_dna(x = character())
```

**Arguments**

- `x` : a character vector.

---

**new_rna**

*RNA vector constructor*

**Description**

RNA vector constructor

**Usage**

```r
new_rna(x = character())
```

**Arguments**

- `x` : a character vector.

---

**read_fasta**

*Read sequences in FASTA format*

**Description**

Read sequences in FASTA format

**Usage**

```r
read_fasta(file, type = "DNA")
```

**Arguments**

- `file` : A path to a file, a connection or a character string.
- `type` : Type of data. Can be "DNA" (the default), "RNA" or "AA".
**rev_complement**

**Value**

A DNA, RNA or AA vector (depending on type argument).

**See Also**

Other input/output operations: `write_fasta()`

---

### Description

Reverse and complement sequences

### Usage

```r
seq_complement(x)
seq_reverse(x)
```

### Arguments

- `x` a DNA or RNA vector. Function `seq_reverse` also accepts AA vectors.

### Value

A reverse or complement sequence (same class as the input).

### See Also

Other biological operations: `seq_rev_translate()`, `seq_translate()`, `transcription`

### Examples

```r
x <- dna("ACTTTGGCTAAG")
seq_reverse(x)
seq_complement(x)
```
**Description**

`rna()` build a RNA vector from a character vector.

**Usage**

```r
rna(...)  # Build a RNA vector from a character vector.
```

**Arguments**

+ `...`: characters to turn into RNA. Can be a set of name-value pairs.

**Value**

A vector of class `bioseq_rna`.

**See Also**

Other classes: `aa()`, `dna()`.

**Examples**

```r
rna("AGGUGC", "UUCGA")
rna(Seq_1 = "AGGUGC", Seq_2 = "UUCGA")
x <- c("AGGTGC", "TTCGA")
rna(x)
```

---

**Description**

This function opens SeaView (Gouy, Guindon & Gascuel, 2010) to visualize biological sequences and phylogenetic trees. The software must be installed on the computer.

**Usage**

```r
seaview(
  x,
  seaview_exec =getOption("bioseq.seaview.exec", default = "seaview")
)
```
Arguments

- **x**: a DNA, RNA or AA vector. Alternatively a DNAbin or AAbin object or a phylogenetic tree (class phylo).

- **seaview_exec**: a character string giving the path of the program.

Details

By default, the function assumes that the executable is installed in a directory located on the PATH. Alternatively the user can provide an absolute path to the executable (i.e. the location where the software was installed/uncompressed). This can be stored in the global options settings using `options(bioseq.seaview.exec = "my_path_to_seaview")`.

References


See Also

Other GUI wrappers: aliview()

Description

Replace matched patterns in sequences

Usage

`seq_replace_pattern(x, pattern, replacement)`

Arguments

- **x**: a DNA, RNA or AA vector.

- **pattern**: a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

- **replacement**: a vector of replacements.

Value

A vector of same class as x.
Patterns

It is important to understand how patterns are treated in bioseq. Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning. Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG. Alternatively, patterns can be a simple character vector containing regular expressions. Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC", "GG") will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

stri_replace from stringi and str_replace from stringr for the underlying implementation.

Other string operations: seq_combine(), seq_count_pattern(), seq_crop_pattern(), seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

x <- dna("ACGTTAGTGAGCGGT", "CTCGAATGA")
seq_replace_pattern(x, dna("AAA"), dna("GGGGGG"))
seq_replace_pattern(x, "^A.{2}T", "TTTTTT")

Description

Cluster sequences by similarity

Usage

seq_cluster(x, threshold = 0.05, method = "complete")

Arguments

x a DNA, RNA or AA vector of sequences to clustered.
threshold Threshold value (range in [0, 1]).
method the clustering method (see details).
The function uses **ape** `dist.dna` and `dist.aa` functions to compute pairwise distances among sequences and **hclust** for clustering.

Computing a full pairwise distance matrix can be computationally expensive. It is recommended to use this function for moderate size dataset.

Supported methods are:

- "single" (= Nearest Neighbour Clustering)
- "complete" (= Farthest Neighbour Clustering)
- "average" (= UPGMA)
- "mcquitty" (= WPGMA)

**Value**

An integer vector with group memberships.

**See Also**

Function **seq_consensus** to compute consensus and representative sequences for clusters.

Other aggregation operations: **seq_consensus()**

**Examples**

```r
x <- c("-----TACGCAGTAAAAGCTACTGATG",
       "CGTCATACGCAGTAAAAACTACTGATG",
       "CTTCATACGCAGTAAAAACTACTGATG",
       "CTTCATATGCAGTAAAAACTACTGATG",
       "CTTCATACGCAGTAAAAACTACTGATG",
       "CGTCATACGCAGTAAAAGCTACTGATG",
       "CTTCATATGCAGTAAAAGCTACTGACG")

x <- dna(x)
seq_cluster(x)
```

**seq_combine**

Combine multiple sequences

**Description**

Combine multiple sequences

**Usage**

```r
seq_combine(..., sep = "", collapse = NULL)
```
Arguments

... One or more vectors of sequences (DNA, RNA, AA). They must all be of the same type. Short vectors are recycled.

sep String to insert between input vectors.

collapse If not NULL, combine everything with this string as separator.

Details

The strings sep and collapse will be coerced to the type of input vectors with a warning if some character have to replaced.

Value

A vector of sequences (if collapse is NULL). A vector with a single sequence, otherwise.

See Also

stri_join from stringi and str_c from stringr for the underlying implementation.

Other string operations: seq-replace, seq_count_pattern(), seq_crop_pattern(), seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

```r
x <- dna("ACGTTAGTGCGCGT", "CTCGAATGA")
y <- dna("TTTTTTTT", "AAAAAAATAA")
seq_combine(x, y)
seq_combine(y, x, sep = "CCCCCC")
seq_combine(y, x, sep = "CCCCCC", collapse = "GGGG")
```

---

### seq_consensus

Find a consensus sequence for a set of sequences.

**Description**

Find a consensus sequence for a set of sequences.

**Usage**

```r
seq_consensus(x, method = "chr_majority", weights = NULL, gaps = TRUE)
```
seq_count_pattern

**Arguments**

- **x**: a DNA, RNA or AA vector.
- **method**: the consensus method (see Details).
- **weights**: an optional numeric vector of same length as `x` giving a weight for each input sequence.
- **gaps**: logical. Should the gaps ("." ) taken into account.

**Details**

"chr_majority", "chr_ambiguity", "seq_centrality", "seq_majority"

For chr_ambiguity gap character always override other characters. Use gaps = FALSE to ignore gaps.

**Value**

A consensus sequence

**See Also**

Other aggregation operations: seq_cluster()

**Examples**

```r
x <- c("-----TACGCAGTAAAAGCTACTGATG",
    "CGTCATACGCAGTAAAAACTACTGATG",
    "CTTCATACGCAGTAAAAACTACTGATG",
    "CTTCATATGCAGTAAAAACTACTGATG",
    "CTTCATACGCAGTAAAAACTACTGATG",
    "CGTCATACGCAGTAAAAGCTACTGATG",
    "CTTCATATGCAGTAAAAGCTACTGACG")

x <- dna(x)
seq_consensus(x)
```

---

seq_count_pattern  Count the number of matches in sequences

**Description**

Count the number of matches in sequences

**Usage**

```r
seq_count_pattern(x, pattern)
```
Arguments

x  a DNA, RNA or AA vector.

pattern  a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

Value

An integer vector.

Patterns

It is important to understand how patterns are treated in `bioseq`.

Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.

Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern `dna("ARG")` will match AAG or AGG.

Alternatively, patterns can be a simple character vector containing regular expressions.

Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern `list(c("AAA", "CCC"), "GG")` will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When `max_error` is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

`stri_count` from `stringi` and `str_count` from `stringr` for the underlying implementation.

Other string operations: `seq-replace`, `seq_combine()`, `seq_crop_pattern()`, `seq_crop_position()`, `seq_detect_pattern()`, `seq_extract_pattern()`, `seq_extract_position()`, `seq_remove_pattern()`, `seq_remove_position()`, `seq_replace_position()`, `seq_split_kmer()`, `seq_split_pattern()`

Examples

```r
x <- dna("ACGTTAGTGTAGCCGT", "CTCGAAATGA")
seq_count_pattern(x, dna("AAA"))
seq_count_pattern(x, "T.G")
```
seq_crop_pattern

Crop sequences using delimiting patterns

Description

Crop sequences using delimiting patterns

Usage

seq_crop_pattern(
  x,
  pattern_in,
  pattern_out,
  max_error_in = 0,
  max_error_out = 0,
  include_patterns = TRUE
)

Arguments

x a DNA, RNA or AA vector to be cropped.
pattern_in patterns defining the beginning (left-side).
pattern_out patterns defining the end (right-side).
max_error_in, max_error_out numeric values ranging from 0 to 1 and giving the maximum error rate allowed between the target sequence and pattern_in/pattern_out. Error rate is relative to the length of the pattern.
include_patterns logical. Should the matched pattern sequence included in the returned sequences?

Value

A cropped DNA, RNA or AA vector. Sequences where patterns are not detected returns NA.

Fuzzy matching

When max_error_in or max_error_out are greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

Patterns

It is important to understand how patterns are treated in bioseq.

Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.
Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG.

Alternatively, patterns can be a simple character vector containing regular expressions.

Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC", "GG")) will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

stri_extract from stringi, str_extract from stringr and afind from stringdist for the underlying implementation.

Other string operations: seq-replace, seq_combine(), seq_count_pattern(), seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

```r
x <- dna("ACGTTAAAAGTGAGCCCCGT", "CTCGAAATGA")
seq_crop_pattern(x, pattern_in = "AAAA", pattern_out = "CCCC")
```

---

**seq_crop_position**

_Crop sequences between two positions_

**Description**

Crop sequences between two positions

**Usage**

```r
seq_crop_position(x, position_in = 1, position_out = -1)
```

**Arguments**

- `x` a DNA, RNA or AA vector.
- `position_in` an integer giving the position where to start cropping.
- `position_out` an integer giving the position where to stop cropping.

**Value**

A cropped DNA, RNA or AA vector.
seq_detect_pattern

See Also

stri_sub from stringi and str_sub from stringr for the underlying implementation.

Other string operations: seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

x <- dna("ACGTTAGTGAGCCGT")

# Drop the first 3 nucleotides (ACG)
seq_crop_position(x, position_in = 4)

# Crop codon between position 4 and 6
seq_crop_position(x, position_in = 4, position_out = 6)

seq_detect_pattern

Detect the presence of patterns in sequences

Description

Detect the presence of patterns in sequences

Usage

seq_detect_pattern(x, pattern, max_error = 0)

Arguments

x
  a DNA, RNA or AA vector.

pattern
  a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

max_error
  numeric value ranging from 0 to 1 and giving the maximum error rate allowed between the target sequence and the pattern. Error rate is relative to the length of the pattern.

Value

A logical vector.
Patterns

It is important to understand how patterns are treated in bioseq. Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.

Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG.

Alternatively, patterns can be a simple character vector containing regular expressions.

Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC"), "GG") will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

stri_detect from stringi, str_detect from stringr and afind from stringdist for the underlying implementation.

Other string operations: seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(), seq_crop_position(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

x <- dna(c("AGTTAGTGTAGCGT", "CTCGAAATGA"))
seq_detect_pattern(x, dna(c("CCG", "AAA")))

# Regular expression
seq_detect_pattern(x, "^A.{2}T")

# Fuzzy matching
seq_detect_pattern(x, dna("AGG"), max_error = 0.2)
# No match. The pattern has three character, the max_error has to be > 1/3 to allow one character difference.
seq_detect_pattern(x, dna("AGG"), max_error = 0.4)
# Match
seq_disambiguate_IUPAC

Disambiguate biological sequences

Description

This function finds all the combinations of sequences corresponding to a given vector of sequences with ambiguities (IUPAC codes).

Usage

seq_disambiguate_IUPAC(x)

Arguments

x

a DNA, RNA or AA vector

Value

A list of DNA, RNA or AA vectors (depending on the input) giving all possible combinations.

See Also

Other op-misc: seq_nchar(), seq_nseq(), seq_spellout(), seq_stat_gc(), seq_stat_prop()

Examples

x <- dna(c("AYCTGW", "CTTN"))
seq_disambiguate_IUPAC(x)

y <- seq_transcribe(x)
seq_disambiguate_IUPAC(y)

z <- aa("YJSNAALNX")
z <- seq_translate(y)
seq_disambiguate_IUPAC(z)
seq_extract_pattern  

Extract matching patterns from sequences

Description

Extract matching patterns from sequences

Usage

seq_extract_pattern(x, pattern)

Arguments

x  
a DNA, RNA or AA vector.
pattern  
a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

Value

A list of vectors of same class as x.

Patterns

It is important to understand how patterns are treated in bioseq. Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.

Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG.

Alternatively, patterns can be a simple character vector containing regular expressions.

Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC"), "GG") will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

stri_extract from stringi and str_extract from stringr for the underlying implementation.

Other string operations: seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(), seq_crop_position(), seq_detect_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()
**seq_extract_position**

**Examples**

```r
x <- dna("ACGTTAGTGTAGCGT", "CTCGAAATGA")
seq_extract_pattern(x, dna("AAA"))
seq_extract_pattern(x, "T.G")
```

---

**seq_extract_position**  
*Extract a region between two positions in sequences*

**Description**

Extract a region between two positions in sequences

**Usage**

```r
seq_extract_position(x, position_in, position_out)
```

**Arguments**

- **x**: a DNA, RNA or AA vector.
- **position_in**: an integer giving the position where to start to extract.
- **position_out**: an integer giving the position where to stop to extract.

**Value**

A vector of same class as x.

**See Also**

`stri_extract` from `stringi` and `str_extract` from `stringr` for the underlying implementation.

Other string operations: `seq-replace`, `seq_combine()`, `seq_count_pattern()`, `seq_crop_pattern()`, `seq_crop_position()`, `seq_detect_pattern()`, `seq_extract_pattern()`, `seq_remove_pattern()`, `seq_remove_position()`, `seq_replace_position()`, `seq_split_kmer()`, `seq_split_pattern()`

**Examples**

```r
x <- dna("ACGTTAGTGTAGCGT", "CTCGAAATGA")
seq_extract_position(x, 3, 8)
```
### seq_nchar

**Count the number of character in sequences**

**Description**

Count the number of character in sequences

**Usage**

```r
seq_nchar(x, gaps = TRUE)
```

**Arguments**

- `x`: a DNA, RNA or AA vector.
- `gaps`: if `FALSE` gaps are ignored.

**Value**

An integer vector giving the size of each sequence of `x`.

**See Also**

Other op-misc: `seq_disambiguate_IUPAC()`, `seq_nseq()`, `seq_spellout()`, `seq_stat_gc()`, `seq_stat_prop()`

**Examples**

```r
x <- dna(c("ATGCAGA", "GGR-----","TTGCCTAGKTGAACC"))
seq_nchar(x)
seq_nchar(x, gaps = FALSE)
```

### seq_nseq

**Number of sequences in a vector**

**Description**

This is an alias for `length`.

**Usage**

```r
seq_nseq(x)
```

**Arguments**

- `x`: a DNA, RNA or AA vector.
seq_remove_pattern

Value
an integer.

See Also
Other op-misc: seq_disambiguate_IUPAC(), seq_nchar(), seq_spellout(), seq_stat_gc(), seq_stat_prop()

Description
Remove matched patterns in sequences

Usage
seq_remove_pattern(x, pattern)

Arguments
x
a DNA, RNA or AA vector.

pattern
a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

Value
A vector of same class as x.

Patterns
It is important to understand how patterns are treated in bioseq.
Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.
Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG.
Alternatively, patterns can be a simple character vector containing regular expressions.
Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC"), "GG") will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.
@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.
seq_remove_position

See Also

- `str_remove` from `stringr` for the underlying implementation.

Other string operations: `seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(),
seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(),
seq_remove_pattern(), seq_replace_position(), seq_split_kmer(), seq_split_pattern()

Examples

```r
x <- dna("ACGTTAGTGACCGTG", "CTCGAAATGA")
seq_remove_pattern(x, dna("AAA"))
seq_remove_pattern(x, "^A.2T")
```

Description

Remove a region between two positions in sequences.

Usage

```r
seq_remove_position(x, position_in, position_out)
```

Arguments

- `x`:
  
a DNA, RNA or AA vector.
- `position_in`:
  
an integer giving the position where to start to remove.
- `position_out`:
  
an integer giving the position where to stop to remove.

Value

A vector of same class as `x`.

See Also

- `str_remove` from `stringr` for the underlying implementation.

Other string operations: `seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(),
seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(),
seq_remove_pattern(), seq_replace_position(), seq_split_kmer(), seq_split_pattern`

Examples

```r
x <- dna("ACGTTAGTGACCGTG", "CTCGAAATGA")
seq_remove_position(x, 2, 6)
seq_remove_position(x, 1:2, 3:4)
```
seq_replace_position

Replace a region between two positions in sequences

Description

Replace a region between two positions in sequences

Usage

seq_replace_position(x, position_in, position_out, replacement)

Arguments

x a DNA, RNA or AA vector.
position_in an integer giving the position where to start to replace.
position_out an integer giving the position where to stop to replace.
replacement a vector of replacements.

Value

A vector of same class as x.

See Also

stri_replace from stringi and str_replace from stringr for the underlying implementation.

Other string operations: seq_replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(),
seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(),
seq_remove_pattern(), seq_remove_position(), seq_split_kmer(), seq_split_pattern()

Examples

x <- dna("ACGTTAGTGTAGCCGT", "CTCGAAATGA")
seq_replace_position(x, c(5, 2), 6, "-------")
seq_rev_translate

Reverse translate amino acid sequences

Description

The function perform reverse translation of amino acid sequences. Such operation does not exist in nature but is provided for completeness. Because of codon degeneracy it is expected to produce many ambiguous nucleotides.

Usage

seq_rev_translate(x, code = 1)

Arguments

- **x**: an amino acid sequence (bioseq_aa)
- **code**: an integer indicating the genetic code to use for reverse translation (default 1 uses the Standard genetic code). See Details.

Details

Gaps (-) are interpreted as unknown amino acids (X) but can be removed prior to the translation with the function seq_remove_gap.

Value

a vector of DNA sequences.

See Also

Other biological operations: rev_complement, seq_translate(), transcription

Examples

```r
x <- dna("ACTTTGGCTAAG")
y <- seq_translate(x)
z <- seq_rev_translate(y)
z
# There is a loss of information during the reverse translation
all.equal(x, z)
```
seq_spellout

seq_spellout  

**Spell out sequences**

Description

This function spells out nucleotides and amino acids in sequences.

Usage

```r
seq_spellout(x, short = FALSE, collapse = " - ")
```

Arguments

- `x` 
  a DNA, RNA or AA vector
- `short` 
  logical. If TRUE, the function will return 3-letters short names for amino acids (ignored for DNA and RNA).
- `collapse` 
  a character vector to separate the results. Set to NULL to avoid collapsing the results.

Value

A character vector if collapse is not NULL. A list of character vectors otherwise.

See Also

Other op-misc: `seq_disambiguate_IUPAC()`, `seq_nchar()`, `seq_nseq()`, `seq_stat_gc()`, `seq_stat_prop()`

Examples

```r
x <- dna("ACGT")
seq_spellout(x)

x <- rna("ACGU")
seq_spellout(x)

x <- aa("ACGBTX")
seq_spellout(x)
```
seq_split_kmer  

Split sequences into k-mers

Description
Split sequences into k-mers

Usage
seq_split_kmer(x, k)

Arguments
- x: A DNA, RNA or AA vector.
- k: an integer giving the size of the k-mer.

Value
a list of k-mer vectors of same class as x.

See Also
- seq_split_pattern.
- Other string operations: seq-replace, seq_combine(), seq_count_pattern(), seq_crop_pattern(), seq_crop_position(), seq_detect_pattern(), seq_extract_pattern(), seq_extract_position(), seq_remove_pattern(), seq_remove_position(), seq_replace_position(), seq_split_pattern()

Examples
x <- dna(a = "ACGTTAGTGACCGT", b = "CTCGAAATGA")
seq_split_kmer(x, k = 5)

seq_split_pattern  

Split sequences

Description
Split sequences

Usage
seq_split_pattern(x, pattern)
Arguments

x  a DNA, RNA or AA vector.

pattern  a DNA, RNA or AA vectors (but same as x) or a character vector of regular expressions, or a list. See section Patterns.

Value

A list of vectors of same class as x.

Patterns

It is important to understand how patterns are treated in `bioseq`.

Patterns are recycled along the sequences (usually the x argument). This means that if a pattern (vector or list) is of length > 1, it will be replicated until it is the same length as x. The reverse is not true and a vector of patterns longer than a vector of sequences will raise a warning.

Patterns can be DNA, RNA or AA vectors (but they must be from the same class as the sequences they are matched against). If patterns are DNA, RNA or AA vectors, they are disambiguated prior to matching. For example pattern dna("ARG") will match AAG or AGG.

Alternatively, patterns can be a simple character vector containing regular expressions.

Vectors of patterns (DNA, RNA, AA or regex) can also be provided in a list. In that case, each vector of the list will be collapsed prior matching, which means that each vector element will be used as an alternative pattern. For example pattern list(c("AAA", "CCC"), "GG") will match AAA or CCC in the first sequence, GG in the second sequence, AAA or CCC in the third, and so on following the recycling rule.

@section Fuzzy matching: When max_error is greater than zero, the function perform fuzzy matching. Fuzzy matching does not support regular expression.

See Also

`stri_split` from `stringi` and `str_split` from `stringr` for the underlying implementation.

Other string operations: `seq-replace`, `seq_combine()`, `seq_count_pattern()`, `seq_crop_pattern()`, `seq_crop_position()`, `seq_detect_pattern()`, `seq_extract_pattern()`, `seq_extract_position()`, `seq_remove_pattern()`, `seq_remove_position()`, `seq_replace_position()`, `seq_split_kmer()`

Examples

```r
x <- dna(a = "ACGTTAGTGGCCGT", b = "CTCGAAATGA")
seq_split_pattern(x, dna("AAA"))
seq_split_pattern(x, "T.G")
```
seq_stat_gc

**Compute G+C content**

**Description**

Compute G+C content

**Usage**

`seq_stat_gc(x)`

**Arguments**

- `x` a DNA or RNA

**Details**

Ambiguous characters (other than S and W) are ignored.

**Value**

A numeric vector of G+C proportions.

**See Also**

Other op-misc: `seq_disambiguate_IUPAC()`, `seq_nchar()`, `seq_nseq()`, `seq_spellout()`, `seq_stat_prop()`

**Examples**

```r
x <- dna(c("ATGCAGA", "GGR-----","TTGCCTAGKTGAACC"))
seq_stat_gc(x)
```

seq_stat_prop

**Compute proportions for characters**

**Description**

Compute proportions for characters

**Usage**

`seq_stat_prop(x, gaps = FALSE)`
seq_translate

Arguments

x  a DNA, RNA or AA vector.
gaps  if FALSE gaps are ignored.

Value

A list of vectors indicating the proportion of characters in each sequence.

See Also

Other op-misc: seq_disambiguate_IUPAC(), seq_nchar(), seq_nseq(), seq_spellout(), seq_stat_gc()

Examples

```r
x <- dna(c("ATGCAGA", "GGR-----", "TTGCCTAGKTGAACC"))
seq_stat_prop(x)
seq_stat_prop(x, gaps = TRUE)
```

---

seq_translate  Translate DNA/RNA sequences into amino acids

Description

Translate DNA/RNA sequences into amino acids

Usage

seq_translate(x, code = 1, codon_frame = 1, codon_init = FALSE)

Arguments

x  a vector of DNA (bioseq_dna) or RNA (bioseq_rna).
code  an integer indicating the genetic code to use for translation (default 1 uses the Standard genetic code). See Details.
codon_frame  an integer giving the nucleotide position where to start translation.
codon_init  a logical indicating whether the first codon is evaluated as a possible codon start and translated to methionine.
Details

Several genetic codes can be used for translation. See genetic-codes to get the list of available genetic codes and their ID number.

Gaps (-) are interpreted as unknown nucleotides (N) but can be removed prior to the translation with the function seq_remove_gap.

The function deals with ambiguities on both sides. This means that if ambiguous codons cannot be translated to amino acid, they are translated to the most specific ambiguous amino acids (X in the most extreme case).

Value

An amino acid vector (bioseq_aa).

See Also

Other biological operations: rev_complement, seq_rev_translate(), transcription

Examples

```r
x <- dna(c("ATGCAGA", "GGR","TTGCTAGKTGAACC", "AGGNGC", "NNN"))
seq_translate(x)
```

---

### transcription

**Transcribe DNA, reverse-transcribe RNA**

**Description**

Transcribe DNA, reverse-transcribe RNA

**Usage**

```r
seq_transcribe(x)
seq_rev_transcribe(x)
```

**Arguments**

- `x` A vector of DNA for seq_transcribe, a vector of RNA for seq_rev_transcribe

**Value**

A vector of RNA for seq_transcribe, a vector of DNA for seq_rev_transcribe

**See Also**

Other biological operations: rev_complement, seq_rev_translate(), seq_translate()
write_fasta

Write sequences in FASTA format

Description
Write sequences in FASTA format

Usage
write_fasta(x, file, append = FALSE, line_length = 80, block_length = 10)

Arguments
- **x**: a DNA, RNA or AA vector.
- **file**: a path to a file or a connection.
- **append**: a logical. If TRUE append the data to the file. If FALSE (default), overwrite the file.
- **line_length**: length (in number of character) of one line (excluding spaces separating blocks). Use Inf to avoid line breaks.
- **block_length**: length (in number of character) of one block. Use the same value as line_length or Inf to avoid block separation.

See Also
Other input/output operations: read_fasta()
Index

* GUI wrappers
  aliview, 4
  seaview, 18

* aggregation operations
  seq_cluster, 20
  seq_consensus, 22

* biological operations
  rev_complement, 17
  seq_rev_translate, 36
  seq_translate, 41
  transcription, 42

* classes
  aa, 3
  dna, 11
  rna, 18

* conversions
  as-tibble-ape, 5
  as-tibble-bioseq, 6
  as_aa, 7
  as_AAbin, 7
  as_dna, 8
  as_DNAbin, 9
  as_rna, 10
  as_seqinr_alignment, 10

* datasets
  fragilaria, 12

* input/output operations
  readfasta, 16
  writefasta, 43

* op-misc
  seq_disambiguate_IUPAC, 29
  seq_cchar, 32
  seq_nseq, 32
  seq_spellout, 37
  seq_stat_gc, 40
  seq_stat_prop, 40

* string operations
  seq_replace, 19
  seq_combine, 21
  seq_count_pattern, 23
  seq_crop_pattern, 25
  seq_crop_position, 26
  seq_detect_pattern, 27
  seq_extract_pattern, 30
  seq_extract_position, 31
  seq_remove_pattern, 33
  seq_remove_position, 34
  seq_replace_position, 35
  seq_split_kmer, 38
  seq_split_pattern, 38

aa, 3, 11, 18
afind, 26, 28
aliview, 4, 19
alphabets, 4
as-tibble-ape, 5
as-tibble-bioseq, 6
as_aa, 5–7, 8–10
as_AAbin, 5–7, 7, 8–10
as_AAbin.tbl_df, 8
as_dna, 5–7, 8, 9, 10
as_DNAbin, 5–8, 9, 10
as_DNAbin.tbl_df, 9
as_rna, 5–10, 10
as_seqinr_alignment, 5–10, 10
as_tibble.AAbin (as-tibble-ape), 5
as_tibble.bioseq_aa (as-tibble-bioseq), 6
as_tibble.bioseq_dna (as-tibble-bioseq), 6
as_tibble.bioseq_rna (as-tibble-bioseq), 6
as_tibble.DNAbin (as-tibble-ape), 5
dic_genetic_codes, 11
dist(aa), 21
dist dna, 21
dna, 3, 11, 18