Package 'microseq'

February 12, 2018

Type Package

Title Basic Biological Sequence Handling

Version 1.2.2

Date 2018-02-12

Author Lars Snipen, Kristian Hovde Liland

Maintainer Lars Snipen <lars.snipen@nmbu.no>

Description Basic functions for microbial sequence data analysis. The idea is to use the basic R data structures as much as possible, without building complex data types.

License GPL-2

Depends R (>= 3.3.0)

Imports Rcpp (>= 0.12.0)

LinkingTo Rcpp (>= 0.12.0)

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-12 15:27:17 UTC

R topics documented:

`microseq-package` *Basic Biological Sequence Analysis*

Description

A collection of functions for basic analysis of microbial sequence data.

Usage

```
microseq()
```

Details

Package:	microseq
Type:	Package
Version:	1.0
Date:	2016-06-15
License:	GPL-2

Author(s)

Lars Snipen, Kristian Hovde Liland
Maintainer: Lars Snipen <lars.snipen@nmbu.no>

`cmalign` *Multiple alignment using Infernal*

Description

Computing a multiple sequence alignment using the Infernal software.

Usage

```
cmalign(in.file, out.file, CM.file, threads = 1)
```

Arguments

`in.file`	Name of FASTA-file with input sequences.
`out.file`	Name of file to store the result.
`CM.file`	Name of file with correlation model.
`threads`	Number of CPU's to use

Details

The software Infernal (Nawrocki&Eddy, 2013) must be installed and available on the system. Test this by typing `system("cmalign -h")` in the Console, and some sensible output should be produced. For more details on Infernal, see http://eddylab.org/infernal/.

This function is most typically used to align 16S rRNA sequences.

The `cmalign` function will produce a multiple alignment, like e.g. `muscle`, but makes use of a *correlation model* to do so. A correlation model means in this case a description of how various bases have a long-range relation, due to folding of the sequence. This means that you can only use this function to align sequences for which you have such correlation models. Such models are typically available for a number of RNA-families, see below.

The argument `CM.file` is the name of a file with a valid correlation model, e.g. one downloaded from the Rfam database. See examples below for the 16S model supplied with this package.

Value

The result is written to the file specified in `out.file`.

Author(s)

Lars Snipen.

References

E.P. Nawrocki and S.R. Eddy, Infernal 1.1: 100-fold faster RNA homology searches, Bioinformatics 29:2933-2935 (2013).

See Also

`msaTrim`.

Examples

```
## Not run:
in.file <- file.path(file.path(path.package("microseq"),"extdata"),"16S.fasta")
cm.file <- file.path(file.path(path.package("microseq"),"extdata"),"ssu_bacteria.cm")
cmalign(in.file,"msa_infernal.fasta",cm.file)

## End(Not run)
```

gregexpr *Extended* gregexpr *with substring retrieval*

Description

An extension of the function base::gregexpr enabling retrieval of the matching substrings.

Usage

```
gregexpr(pattern, text, ignore.case = FALSE, perl = FALSE, fixed = FALSE,
  useBytes = FALSE, extract = FALSE)
```

Arguments

pattern	Character string containing a regular expression (or character string for fixed = TRUE) to be matched in the given character vector. Coerced by as.character to a character string if possible. If a character vector of length 2 or more is supplied, the first element is used with a warning. Missing values are not allowed.
text	A character vector where matches are sought, or an object which can be coerced by as.character to a character vector.
ignore.case	If FALSE, the pattern matching is *case sensitive* and if TRUE, case is ignored during matching.
perl	Logical. Should perl-compatible regexps be used? Has priority over extended.
fixed	Logical. If TRUE, 'pattern' is a string to be matched as is. Overrides all conflicting arguments.
useBytes	Logical. If TRUE the matching is done byte-by-byte rather than character-by-character. See base::gregexpr for details.
extract	Logical indicating if matching substrings should be extracted and returned.

Details

Extended version of base:gregexpr that enables the return of the substrings matching the pattern. The last argument 'extract' is the only difference to base::gregexpr. The default behaviour is identical to base::gregexpr, but setting extract=TRUE means the matching substrings are returned.

Value

It will either return what the base::gregexpr would (extract=FALSE) or a 'list' of substrings matching the pattern (extract=TRUE). There is one 'list' element for each string in 'text', and each list element contains a character vector of all matching substrings in the corresponding entry of 'text'.

Author(s)

Lars Snipen and Kristian Liland.

See Also

gregexpr

Examples

```
sequences<-c("ACATGTCATGTCC","CTTGTATGCTG")
gregexpr("ATG",sequences,extract=TRUE)
```

---

iupac2regex *Ambiguity symbol conversion*

Description

Converting DNA ambiguity symbols to regular expressions, and vice versa.

Usage

```
iupac2regex( sequence )
regex2iupac( sequence )
```

Arguments

sequence	Character string containing a DNA sequence.

Details

The DNA alphabet may contain ambiguity symbols, e.g. a W means either A or T. When using a regular expression search, these letters must be replaced by the proper regular expression, e.g. W is replaced by [AT] in the string. The iupac2regex makes this translation, while regex2iupac cobverts the other way again (replace [AT] with W).

Value

A string where the ambiguity symbol has been replaced by a regular expression (iupac2regex) or a regular expression has been replaced by an ambiguity symbol (regex2iupac).

Author(s)

Lars Snipen.

Examples

```
iupac2regex( "ACWGT" )
regex2iupac( "AC[AG]GT" )
```

msalign *Multiple alignment*

Description

Quickly computing a smallish multiple sequence alignment.

Usage

```
msalign(fdata, machine = "muscle")
```

Arguments

fdata	A Fasta object with input sequences.
machine	Function that does the 'dirty work'.

Details

This function computes a multiple sequence alignment given a set of sequences in a Fasta object, see readFasta for more on Fasta objects.

It is merely a wrapper for the function named in machine to avoid explicit writing and reading of files. This function should only be used for small data sets, since no result files are stored. For heavier jobs, use the machine function directly.

At present, the only machine function implemented is muscle, but other third-party machines may be included later.

Note that this function will run muscle with default settings, which is fine for small data sets.

Value

Results are returned as a Fasta object.

Author(s)

Lars Snipen.

See Also

muscle, msaTrim.

Examples

```
## Not run:
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fasta")
fdta <- readFasta(ex.file)
msa <- msalign(fdta)

## End(Not run)
```

msaTrim *Trimming multiple sequence alignments*

Description

Trimming a multiple sequence alignment by discarding columns with too many gaps.

Usage

```
msaTrim(msa, gap.end = 0.5, gap.mid = 0.9)
```

Arguments

msa	A Fasta object containing a multiple alignment.
gap.end	Fraction of gaps tolerated at the ends of the alignment (0-1).
gap.mid	Fraction of gaps tolerated inside the alignment (0-1).

Details

A multiple alignment is trimmed by removing columns with too many indels (gap-symbols). Any columns containing a fraction of gaps larger than gap.mid are discarded. For this reason, gap.mid should always be farily close to 1.0 therwise too many columns may be discarded, destroying the alignment.

Due to the heuristics of multiple alignment methods, both ends of the alignment tend to be uncertain and most of the trimming should be done at the ends. Starting at each end, columns are discarded as long as their fraction of gaps surpasses gap.end. Typically gap.end can be much smaller than gap.mid, but if set too low you risk that all columns are discarded!

Value

The trimmed alignment is returned as a Fasta object.

Author(s)

Lars Snipen.

See Also

muscle, cmalign.

Examples

```
## Not run:
msa.file <- file.path(file.path(path.package("microseq"),"extdata"),"msa.fasta")
msa <- readFasta(msa.file)
print(nchar(msa$Sequence))
msa.trimmed <- msaTrim(msa)
print(nchar(msa.trimmed$Sequence))
```

```
## End(Not run)
```

`muscle` *Multiple alignment using MUSCLE*

Description

Computing a multiple sequence alignment using the MUSCLE software.

Usage

```
muscle(in.file, out.file, quiet = FALSE, diags = FALSE, maxiters = 16)
```

Arguments

`in.file`	Name of FASTA-file with input sequences.
`out.file`	Name of file to store the result.
`quiet`	Logical, `quiet=FALSE` produces screen output during computations.
`diags`	Logical, `diags=TRUE` gives faster but less reliable alignment.
`maxiters`	Maximum number of iterations.

Details

The software MUSCLE (Edgar, 2004) must be installed and available on the system. Test this by typing `system("muscle")` in the Console, and some sensible output should be produced. NOTE: The executable must be named `muscle` on your system, no version numbers etc. For more details on MUSCLE, see http://www.drive5.com/muscle.

By default `diags=FALSE` but can be set to `TRUE` to increase speed. This should be done only if sequences are highly similar.

By default `maxiters=16`. If you have a large number of sequences (a few thousand), or they are very long, then this may be too slow for practical use. A good compromise between speed and accuracy is to run just the first two iterations of the algorithm. On average, this gives accuracy equal to T-Coffee and speeds much faster than CLUSTALW. This is done by the option `maxiters=2`.

Value

The result is written to the file specified in `out.file`.

Author(s)

Lars Snipen.

References

Edgar, R.C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Res, 32, 1792-1797.

See Also

msaTrim.

Examples

```
## Not run:
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fasta")
muscle(in.file=ex.file,out.file="deleteMe.fasta")

## End(Not run)
```

---

plot.Fasta *Plotting and summary of* Fasta *objects*

Description

Generic functions for plotting and printing the content of a Fasta object.

Usage

```
## S3 method for class 'Fasta'
plot(x, y = NULL, col = "tan4", border = "tan4", ...)

## S3 method for class 'Fasta'
summary(object, ...)
```

Arguments

x	A Fasta object, see below.
y	not used.
col	Color of bar interiors.
border	Color of bar borders.
...	Optional graphical arguments.
object	A Fasta object, see below.

Details

A Fasta object contains biological sequences in the FASTA format. It is a small (S3) extension to a data.frame. It is actually a data.frame containing at least two text columns named 'Header' and 'Sequence'. The 'Header' column contains the headerlines for each sequence, and the 'Sequence' columns the sequences themselves. A Fasta object is typically created by reading a FASTA formatted file into R by readFasta.

A Fasta object can be treated as a data.frame, which makes it quick and easy to search both 'Header' and 'Sequence' for specific regular expressions, sort or re-arrange the ordering of the sequences, extract subsets or add new data to an existing Fasta object.

The plot.Fasta function will display the content of the Fasta object as a histogram over the lengths of the sequences.

The summary.Fasta function will display a text giving the number of sequences and the alphabet, i.e. listing all unique symbols found in the file.

Author(s)

Lars Snipen and Kristian Hovde Liland.

See Also

readFasta, writeFasta.

Examples

```
# See the examples in the Help-file for readFasta/writeFasta
```

---

plot.Fastq *Plotting and summary of* Fastq *objects*

Description

Generic functions for plotting and printing the content of a Fastq object.

Usage

```
## S3 method for class 'Fastq'
plot(x, y = NULL, col = "tan4", border = "tan4", ...)

## S3 method for class 'Fastq'
summary(object, ...)
```

Arguments

x	A Fastq object, see below.
y	not used.
col	Color of bar interiors.
border	Color of bar borders.
...	Optional graphical arguments.
object	A Fastq object, see below.

Details

A Fastq object contains biological sequences in the FASTQ format. It is a small (S3) extension to a data.frame. It is actually a data.frame containing at least three text columns named 'Header', 'Sequence' and 'Quality'. The 'Header' column contains the headerlines for each sequence, the 'Sequence' columns the sequences themselves and the 'Quality' the base-quality sequence. A Fastq object is typically created by reading a FASTQ formatted file into R by readFastq.

A Fastq object can be treated as a data.frame, which makes it quick and easy to search both for specific regular expressions, sort or re-arrange the ordering of the sequences, extract subsets or add new data to an existing Fastq object.

A Fastq object can also be treated as a Fasta object. Using writeFasta will write the Fastq to a file in Fasta format.

The plot.Fastq function will display the content of the Fastq object as a histogram over the lengths of the sequences.

The summary.Fastq function will display a text giving the number of sequences and the alphabet, i.e. listing all unique symbols found in the file.

Author(s)

Lars Snipen and Kristian Hovde Liland.

See Also

readFastq, writeFastq.

Examples

```
# See the examples in the Help-file for readFastq/writeFastq
```

readFasta *Read and write FASTA files*

Description

Reads and writes biological sequences (DNA, RNA, protein) in the FASTA format.

Usage

```
readFasta(in.file)
writeFasta(fdta, out.file, width = 80)
```

Arguments

in.file	url/directory/name of FASTA file to read.
fdta	A 'Fasta' object, see 'Details' below.
out.file	Name of FASTA-file to create.
width	Number of sequence characters per line.

Details

These functions handle input/output of sequences in the commonly used FASTA format. For every sequence it is presumed there is one Header-line starting with a '>'.

The sequences are stored in a Fasta object. This is an extension of a data.frame containing two text-columns named 'Header' and 'Sequence'. If other columns are present, these will be ignored by writeFasta.

The Fasta object can be treated as a data.frame, but the generic functions plot.Fasta and summary.Fasta are defined. The data.frame property makes it straightforward to manipulate all headers or all sequences, or to extract or delete entries (rows), or to merge several data sets using rbind.

Value

readFasta returns a Fasta object with the contents of the FASTA file. This is an extension to a data.frame and contains two columns of text. The first, named 'Header', contains the headerlines and the second, named 'Sequence', contains the sequences.

writeFasta produces a FASTA file.

Author(s)

Lars Snipen and Kristian Hovde Liland.

See Also

plot.Fasta, summary.Fasta, readFastq.

Examples

```
## Not run:
# We need a FASTA-file to read, here is one example file:
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fasta")
# Reading a file with name in ex.file
fdta <- readFasta(ex.file)
summary(fdta)
plot(fdta)

## End(Not run)
```

`readFastq` *Read and write FASTQ files*

Description

Reads and writes files in the FASTQ format.

Usage

```
readFastq(in.file, Sanger = FALSE)
writeFastq(fdta, out.file)
```

Arguments

`in.file`	url/directory/name of FASTQ file to read.
`Sanger`	logical indicating if old, multi-line Sanger format is used (default = FALSE).
`fdta`	FASTQ object to write.
`out.file`	url/directory/name of FASTQ file to write.

Details

These functions handle input/output of sequences in the commonly used FASTQ format, typically used for storing DNA sequences (reads) after sequencing.

The sequences are stored in a `Fastq` object. This is an extension of a `data.frame` containing three text-columns named '`Header`', '`Sequence`' and '`Quality`'. If other columns are present, these will be ignored by `writeFastq`.

The `Fastq` object can be treated as a `data.frame`, but the generic functions `plot.Fastq` and `summary.Fastq` are defined. The `data.frame` property makes it straightforward to manipulate all headers or all sequences, or to extract or delete entries (rows), or to merge several data sets using `rbind`.

A `Fastq` object can also be treated as a `Fasta` object. Using `writeFasta` will write the `Fastq` object to a file in `Fasta` format.

Value

readFastq returns a Fastq object with the contents of the FASTQ file. This is an extension to a data.frame and contains three columns of text. The first, named 'Header', contains the header-lines and the second, named 'Sequence', contains the sequences and the third, named 'Quality', contains the base quality scores.

readFastq returns a Fastq object with the header, sequence and quality part of the FASTQ file.

Author(s)

Lars Snipen and Kristian Hovde Liland.

See Also

codereadFasta, plot.Fastq, summary.Fastq.

Examples

```
## Not run:
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fastq")
fdta <- readFastq(ex.file)
summary(fdta)

## End(Not run)
```

---

reverseComplement *Reverse-complementation of DNA*

Description

The standard reverse-complement of nucleotide sequences.

Usage

```
reverseComplement(nuc.sequences, reverse = TRUE)
```

Arguments

nuc.sequences	Character vector containing the nucleotide sequences.
reverse	Logical indicating if complement should be reversed.

Details

This function uses the Biostrings::reverseComplement function.

Value

A character vector of reverse-complemented sequences.

Author(s)

Lars Snipen and Kristian Hovde Liland.

Examples

```
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fasta")
fdta <- readFasta(ex.file)
reverseComplement(fdta$Sequence)
```

---

translate *Translation according to the standard genetic code*

Description

The translation from DNA(RNA) to amino acid sequence according to the standard genetic code.

Usage

```
translate(nuc.sequences, M.start = TRUE)
```

Arguments

nuc.sequences	Character vector containing the nucleotide sequences.
M.start	A logical indicating if the amino acid sequence should start with M regardless of start codon (ATG, GTG or TTG).

Details

This function uses the Biostrings::translate function.

Value

A character vector of translated sequences.

Author(s)

Lars Snipen and Kristian Hovde Liland.

Examples

```
ex.file <- file.path(file.path(path.package("microseq"),"extdata"),"small.fasta")
fdta <- readFasta(ex.file)
translate(fdta$Sequence)
```

Index