Package 'edlibR'

November 21, 2022

Title R Integration for Edlib, the C/C++ Library for Exact Pairwise Sequence Alignment using Edit (Levenshtein) Distance

Version 1.0.1

Description
Bindings to edlib, a lightweight performant C/C++ library for exact pairwise sequence alignment using edit distance (Levenshtein distance). The algorithm computes the optimal alignment path, but also can be used to find only the start and/or end of the alignment path for convenience. Edlib was designed to be ultrafast and require little memory, with the capability to handle very large sequences. Three alignment methods are supported: global (Needleman-Wunsch), infix (Hybrid Wunsch), and prefix (Semi-Hybrid Wunsch). The original C/C++ library is described in ``Edlib: a C/C++ library for fast, exact sequence alignment using edit distance", M. Šošić, M. Šikić, <doi:10.1093/bioinformatics/btw753>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.5), stringr (>= 1.4.0)

Suggests testthat (>= 3.1.0), rmarkdown, knitr

RoxygenNote 7.1.2

VignetteBuilder knitr

URL https://github.com/evanbiederstedt/edlibR

BugReports https://github.com/evanbiederstedt/edlibR/issues

NeedsCompilation yes

LinkingTo Rcpp

SystemRequirements C++11

Author Martin Šošić [aut],
Evan Biederstedt [aut, cre]

Maintainer Evan Biederstedt <evan.biederstedt@gmail.com>

Repository CRAN

Date/Publication 2022-11-21 13:10:06 UTC

R topics documented:

`align` *Align query with target using edit distance*

Description

Align query with target using edit distance

Usage

```
align(
  query,
  target,
  mode = "NW",
  task = "distance",
  k = -1,
  cigarFormat = "extended",
  additionalEqualities = NULL
)
```

Arguments

`query`	character string Combined with target must have no more than 256 unique values
`target`	character string Combined with query must have no more than 256 unique values
`mode`	character string (default="NW") Alignment method to be used. Possible values are: - 'NW' for global (default). Note that 'NW' stands for 'Needleman-Wunsch'. - 'HW' for infix. Note that 'HW' stands for 'Hybrid Wunsch'. - 'SHW' for prefix. Note that 'SHW' stands for 'Semi-Hybrid Wunsch'.
`task`	character string (default="distance") Specifies what to calculate. The less there is to calculate, the faster it is. Possible options are (ranked from fastest to slowest): - 'distance': Find the edit distance and the end locations in the target (default). - 'locations': Find the edit distance, the end locations, and the start locations. - 'path': Find the edit distance, the start and end locations, and the alignment path.
`k`	integer (default=-1) Max edit distance to search for — the lower this value, the faster the calculation. Set to -1 (default) to have no limit on edit distance.
`cigarFormat`	character string (default="extended") Specifies which format to use for writing out the CIGAR string. The two possible values are 'standard' and 'extended' (Note: the function getNiceAlignment() only accepts 'cigarFormat="extended"'): - 'standard': Standard uses the following symbols to generate a CIGAR string:

Match: 'M', Insertion: 'I', Deletion: 'D', Mismatch: 'M'. Note that 'M' in this setting can denote either a sequence match or mismatch. - 'extended': Extended uses the following symbols to generate a CIGAR string: Match: '=', Insertion to target: 'I', Deletion from target: 'D', Mismatch: 'X'. e.g. CIGAR of "5=1X1=1I" means "5 matches, 1 mismatch, 1 match, 1 insertion (to target)". For more details on the CIGAR format, please check <http://samtools.github.io/hts-specs/SAMv1.pdf> and <http://drive5.com/usearch/manual/cigar.html>.

`additionalEqualities`
: List of vectors contains pairs of characters (default=NULL) Allows users to extend the definition of equality used in the alignment. The input 'additionalEqualities' must be a list of character vectors whereby each character vector contains a pair of character strings. (NOTE: the character vectors must contain exactly two strings, a pair.) Each pair defines two values as equal. This can be useful e.g. when you want edlib to be case insensitive, or if you want certain characters to act as wildcards. If NULL, there will be no additional extensions to edlib's default equality definition.

Value

List with the following fields: - editDistance: (integer) The edit distance. This is set to -1 if it is larger than k. - alphabetLength: (integer) Length of unique characters in 'query' and 'target' - locations: (list of vectors) List of R vectors of locations, in the format list(c(start, end)). Note: if the start or end positions are NULL, this is encoded as 'NA' to work correctly with R vectors. - cigar: (character string) CIGAR is a standard format for the alignment path. - cigarFormat: (character string) Format provided by the parameter 'cigarFormat' in the function align() which is returned here for the function getNiceAlignment(). (Note: the function getNiceAlignment() only accepts 'extended')

Examples

```
align("ACTG", "CACTRT", mode="HW", task="path")
align("elephant", "telephone")
align("ACTG", "CACTRT", mode="HW", task="path", additionalEqualities=list(c("R", "A"), c("R", "G")))
```

`getNiceAlignment` *Output alignments from align() in NICE format. This outputs the alignment from align() in a visually informative format for human inspection.*

Description

Output alignments from align() in NICE format. This outputs the alignment from align() in a visually informative format for human inspection.

Usage

```
getNiceAlignment(alignResult, query, target, gapSymbol = "-")
```

Arguments

alignResult	list Output of the method align() Note: align() requires the argument task="path" for 'alignResult' to output a CIGAR for getNiceAlignment() Note: Also, align() requires the argument cigarFormat="extended" in order for getNiceAlignment() to work
query	character string The exact query used for alignResult
target	character string The exact target used for alignResult
gapSymbol	character (default="-") Character used to represent gaps in the alignment between query and target. This must be a single character, i.e. a string of length 1.

Value

Alignment in NICE format, which is an informative visual representation of how the query and target align to each other. e.g., for "telephone" and "elephant", it would look like: telephone |||||.|. -elephant It is represented as an R list with the following fields: - query_aligned (character string) - matched_aligned (character string) ('|' for match, '.' for mismatch, ' ' for insertion/deletion) - target_aligned (character string) Normally you will want to print these three in order above with the function nice_print(), or another method to apply pretty-printing to R lists

Examples

```
query = "elephant"
target = "telephone"
result = align(query, target, task = "path")
nice_algn = getNiceAlignment(result, query, target)
```

---

nice_print *Prints the output of getNiceAlignment() in a visually informative format in order to inspect the alignment*

Description

Prints the output of getNiceAlignment() in a visually informative format in order to inspect the alignment

Usage

```
nice_print(niceAlignment)
```

Arguments

niceAlignment	list Output of the method getNiceAlignment()

Value

Pretty-prints the list returned by getNiceAlignment()

Examples

```
query = "elephant"
target = "telephone"
result = align(query, target, task = "path")
nice_algn = getNiceAlignment(result, query, target)
nice_print(nice_algn)
```

Index