Package: calACS (via r-universe)

August 22, 2024

Type Package

Title Calculations for All Common Subsequences

Version 2.2.2

Date 2016-3-31

Author Alan Gu

Maintainer Alan Gu <alan.on.ca@gmail.com>

Description Implements several string comparison algorithms, including
calACS (count all common subsequences), lenACS (calculate the
lengths of all common subsequences), and lenLCS (calculate the
length of the longest common subsequence). Some algorithms
differentiate between the more strict definition of
subsequence, where a common subsequence cannot be separated by
any other items, from its looser counterpart, where a common
subsequence can be interrupted by other items. This difference
is shown in the suffix of the algorithm (-Strict vs -Loose).
For example, q-w is a common subsequence of q-w-e-r and q-e-w-r
on the looser definition, but not on the more strict
definition. calACSLoose Algorithm from Wang, H. All common
subsequences (2007) IJCAI International Joint Conference on
Artificial Intelligence, pp. 635-640.

License GPL

LazyData TRUE

RoxygenNote 5.0.1

NeedsCompilation no

Date/Publication 2016-03-31 19:35:54

Repository https://agu-1.r-universe.dev

RemoteUrl https://github.com/cran/calACS

RemoteRef HEAD

RemoteSha c152cf80f7c20c141e6ec5b88a22e4ca8ceb252b

Contents

---

calACSLoose   *Calculate the total number of all common subsequences between a*
                       string and a vector/list of strings. Subsequences can be interrupted by
                       items, i.e. q-w is considered a subsequence of q-e-w-r

---

Description

Calculate the total number of all common subsequences between a string and a vector/list of strings.
Subsequences can be interrupted by items, i.e. q-w is considered a subsequence of q-e-w-r

Usage

```
calACSLoose(vecA, listB, sep = "-", dropFirstItem = FALSE)
```

Arguments

vecA	The single string
listB	The vector/list of 1 or more strings
sep	Delimiter separating each items in a sequence
dropFirstItem	Boolean. If true, the first item in each sequence is excluded from counting all subsequences

Value

The total number of all common subsequences as an integer in a vector

Examples

```
calACSLoose("q-w-e-r", c("q-e-w-r","q-r-e-w"), "-")
calACSLoose("itemToBeDropped-q-w-e-r", "itemToBeDroped-q-e-w-r", "-", dropFirstItem=TRUE)
```

calACSStrict	*Count the total number of all common subsequences between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'*

Description

Count the total number of all common subsequences between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'

Usage

```
calACSStrict(vecA, listB, sep = "-", dropFirstItem = FALSE,
  ignoreLenOneSubseq = FALSE, ignoreLenZeroSubseq = FALSE)
```

Arguments

vecA	The single string
listB	The vector/list of 1 or more strings
sep	Delimiter separating each items in a sequence
dropFirstItem	Boolean. If true, the first item in each sequence is excluded from counting all subsequences
ignoreLenOneSubseq	
	Boolean. If true, all length one subsequences are not counted as common subsequences
ignoreLenZeroSubseq	
	Boolean. If true, the length zero subsequence (empty set) is not counted as a common subsequence

Value

The total number of all common subsequences as an integer in a vector

Examples

```
calACSStrict("q-w-e-r", c("q-e-w-r","q-r-e-w"), "-")
calACSStrict("itemToBeDropped-q-w-e-r", "itemToBeDroped-q-e-w-r", "-", dropFirstItem=TRUE)
```

`lenACSStrict`	*Calculate the length of each common subsequences between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'*

Description

Calculate the length of each common subsequences between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'

Usage

```
lenACSStrict(vecA, listB, sep = "-", dropFirstItem = FALSE,
  ignoreLenOneSubseq = FALSE)
```

Arguments

`vecA`	The single string
`listB`	The vector/list of 1 or more strings
`sep`	Delimiter separating each items in a sequence
`dropFirstItem`	Boolean. If true, the first item in each sequence is excluded from counting all subsequences
`ignoreLenOneSubseq`	
	Boolean. If true, all length one subsequences are not counted as common subsequences

Value

A list of vectors of the length of each common subsequence

Examples

```
lenACSStrict("q-w-e-r", c("q-e-w-r","q-r-e-w","q-w-r-e"), "-")
lenACSStrict("itemToBeDropped-q-w-e-r", "itemToBeDroped-q-e-w-r", "-", dropFirstItem=TRUE)
```

lenLCSStrict	*Calculate the length of the longest common subsequence (KCS) between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'*

Description

Calculate the length of the longest common subsequence (KCS) between a string and a vector/list of strings. Subsequences cannot be interrupted by any item, i.e. q-w is not considered a subsequence of q-e-w-r due to the interrupting 'e'

Usage

```
lenLCSStrict(vecA, listB, sep = "-", dropFirstItem = FALSE)
```

Arguments

vecA	The single string
listB	The vector/list of 1 or more strings
sep	Delimiter separating each items in a sequence
dropFirstItem	Boolean. If true, the first item in each sequence is excluded from counting all subsequences

Value

A list of vectors of the length of each common subsequence

Examples

```
lenACSStrict("q-w-e-r", c("q-e-w-r","q-r-e-w","q-w-r-e"), "-")
lenACSStrict("itemToBeDropped-q-w-e-r", "itemToBeDroped-q-e-w-r", "-", dropFirstItem=TRUE)
```

longestVec	*The function takes in multiple vectors of any length, and returns the one with the longest length. The tieBreaker variable controls if the first or the last of the longest vectors gets returned in case there are multiple*

Description

The function takes in multiple vectors of any length, and returns the one with the longest length. The tieBreaker variable controls if the first or the last of the longest vectors gets returned in case there are multiple

Usage

```
longestVec(..., tieBreaker = "last")
```

Arguments

`...`	vectors of any length
`tieBreaker`	decides if the first or the last longest vector gets returned if there are multiple longest vectors. Can be either 'first' or 'last'. Default to 'last'.

Examples

```
longestVec(1:5, c('a','b'))
```

Index