Package: demystas (via r-universe)

August 31, 2024

Title Demystifying Automation on Simple Tasks

Version 1.3.11

Date 2018-11-14

Description There are a myriad of daily tasks which seem simple, but yet hide complex computational requirements. These are the kind of challenges that we encounter that are not entirely part of our main objective, but need to be overcome nonetheless. These include conducting complicated string searches, querying spatial information and sometimes just organizing messy data. This package aims to offer useful functions for data analysts which might help in making these ``on-the-fly'' tasks lighter and more automated.

Depends R(>= 2.10.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 6.1.0

Imports doParallel, snow, foreach, doSNOW, utils, sp, raster, geosphere, methods, parallel, stats

ValidationKey 23400039

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://pik-piam.r-universe.dev

RemoteUrl https://github.com/pik-piam/demystas

RemoteRef HEAD

RemoteSha 734c76db6231d55b2140d801186d3a547e9548e8

Contents

`demystas-package` *Demystifying Automation on Simple Tasks (demystas)*

Description

There are a myriad of daily tasks which seem simple, but yet hide complex computational requirements. These are the kind of challenges that we encounter that are not entirely part of our main objective, but need to be overcome nonetheless. These include conducting complicated string searches, querying spatial information and sometimes just organizing messy data.

Details

This package aims to offer useful functions for data analysts which might help in making these "on-the-fly" tasks lighter and more automated.

`coords2spi` *Convert coordinates to spatial information*

Description

Wrapper function for ‘sp::over‘ in overlaying points over polygons to retrieve relevant intersection information.

Usage

```
coords2spi(points, global)
```

Arguments

`points`	a data frame with two columns of coordinates, or a Spatial object listed here under "x" in showMethods(sp::over). These represent points to be mapped onto polygon(s). Coordinates in ‘points‘ should be in the same column order as those in ‘global‘
`global`	an object onto which ‘points‘ is mapped, possible classes corresponding to ‘points‘ listed under "y" in showMethods(sp::over). If ‘points‘ is a data frame, it will be converted into a SpatialPoints object.

Value

a data frame corresponding to the coordinates in ‘points‘ mapped onto ‘global‘ with the attributes of ‘global‘

Author(s)

Atreya Shankar

See Also

`over`

Examples

```
## Not run:

require(rworldmap)
test <- demystas::coords2spi(as.data.frame(cbind(60,50)), getMap(resolution="low"))

## End(Not run)
```

`greps` *Sequential pairwise grep-style matching*

Description

Sequentially performs a two-way grep-style analysis on two character vectors. Calculates pairwise matching scores based on a rigid customized routine and returns matching strings ranked from best to worst. The user is able to influence the algorithm by tweaking matching parameters.

Usage

```
greps(x, y, sepx = "\\.", sepy = "\\.", limitChar = 0,
  limitWord = 0, booster = 0.8, wordIgnore = NULL,
  checkBoth = TRUE, ignore.case = TRUE)
```

Arguments

`x`	a character vector containing elements to be considered in pairwise grep-analysis. Words are separated by ‘sepx‘.
`y`	a character vector containing elements to be considered in pairwise grep-analysis. Words are separated by ‘sepy‘.
`sepx`	a regex-style expression which indicates how words are separated in ‘x‘. If ‘x‘ is already a final vector and does not need to be segmented, input ‘sepx = NULL‘. Defaults to "\\."

`sepy`	a regex-style expression which indicates how words are separated in ‘y‘. If ‘y‘ is already a final vector and does not need to be segmented, input ‘sepy = NULL‘. Defaults to "\\."
`limitChar`	a numerical value from 0 to 1 which provides a lower proportional bound for a word-to-word match to be considered significant. If the user prioritizes loosely matched words, one can leave this value low such as 0.1. Alternatively, if the end-user prioritizes strongly matched individual words, ‘limitChar‘ can be increased to a value of say, 0.7. Defaults to 0.
`limitWord`	a numerical value greater than or equal to 0 which provides a proportional filter for significant overall characters matched. Defaults to 0.
`booster`	a numerical value between 0 to 1 which provides a boost to the matching score of exceptionally well-matched words. For meaningful results, its value should be greater than ‘limitWord‘. Defaults to 0.8.
`wordIgnore`	a character vector which should be ignored while searching for matches. Examples could be redundant characters such as "the" or "of". Defaults to NULL.
`checkBoth`	a logical which indicates whether both left and right grep analyses should be conducted (TRUE), or if only a left grep analysis is necessary (FALSE). Defaults to TRUE.
`ignore.case`	a logical which indicates if cases should be ignored when matching. Defaults to TRUE.

Value

a list containing two matrices. The first "result" matrix has a total number of rows equal to the length of vector x. The first column contains a repeat of vector ‘x‘ and the corresponding columns contain ranked ‘y‘ vector matches to the corresponding rows. The matches are ranked from best to worst as column number increases. The second "rank" matrix contains a matrix with equivalent dimension as the first matrix. Instead of containing the matches from ‘y‘, this matrix contains the matching scores of the respective components from the first matrix. A ranking score of 99 implies a perfect match. Perfect matches are isolated for each row.

Author(s)

Atreya Shankar

Examples

```
## Not run:

x <- c("foo.test.xyz", "baz.foosh", "bat")
y <- c("ba","foosba.asd", "bats.at", "foos", "gams.asd")
test <- demystas::greps(x, y)

## End(Not run)
```

`grepsAbb` *Sequential abbreviation mapping*

Description

Sequentially maps abbreviations onto complete words and returns a matrix of valid possibilities.

Usage

```
grepsAbb(x, y)
```

Arguments

`x`	a character vector containing abbreviations.
`y`	a character vector containing whole words that could correspond to abbreviations.

Value

a matrix with total number of rows equal to the length of vector ‘x‘. The first column is a repeat of vector ‘x‘. Further columns represent matched possible ‘y‘ vector components.

Author(s)

Atreya Shankar

Examples

```
## Not run:

x <- c("BLG", "BLD", "LAT", "EMM")
y <- c("Boulder","Latino", "Eminem", "Emmys", "Building")
test <- demystas::grepsAbb(x, y)

## End(Not run)
```

`grepsParallel` *Parallel pairwise grep-style matching*

Description

Performs a two-way grep-style analysis on two character vectors using parallel computation. Calculates pairwise matching scores based on a rigid customized routine and returns matching strings ranked from best to worst. The user is able to influence the algorithm by tweaking matching parameters.

Usage

```
grepsParallel(x, y, noCores, sepx = "\\.", sepy = "\\.",
  limitChar = 0, limitWord = 0, booster = 0.8, wordIgnore = NULL,
  checkBoth = TRUE, ignore.case = TRUE)
```

Arguments

`x`	a character vector containing elements to be considered in pairwise grep-analysis. Words are separated by ‘sepx‘.
`y`	a character vector containing elements to be considered in pairwise grep-analysis. Words are separated by ‘sepy‘.
`noCores`	is a numerical value specifying the number of cores to be used for parallel computation.
`sepx`	a regex-style expression which indicates how words are separated in ‘x‘. If ‘x‘ is already a final vector and does not need to be segmented, input ‘sepx = NULL‘. Defaults to "\\."
`sepy`	a regex-style expression which indicates how words are separated in ‘y‘. If ‘y‘ is already a final vector and does not need to be segmented, input ‘sepy = NULL‘. Defaults to "\\."
`limitChar`	a numerical value from 0 to 1 which provides a lower proportional bound for a word-to-word match to be considered significant. If the user prioritizes loosely matched words, one can leave this value low such as 0.1. Alternatively, if the end-user prioritizes strongly matched individual words, ‘limitChar‘ can be increased to a value of say, 0.7. Defaults to 0.
`limitWord`	a numerical value greater than or equal to 0 which provides a proportional filter for significant overall characters matched. Defaults to 0.
`booster`	a numerical value between 0 to 1 which provides a boost to the matching score of exceptionally well-matched words. For meaningful results, its value should be greater than ‘limitWord‘. Defaults to 0.8.
`wordIgnore`	a character vector which should be ignored while searching for matches. Examples could be redundant characters such as "the" or "of". Defaults to NULL.
`checkBoth`	a logical which indicates whether both left and right grep analyses should be conducted (TRUE), or if only a left grep analysis is necessary (FALSE). Defaults to TRUE.
`ignore.case`	a logical which indicates if cases should be ignored when matching. Defaults to TRUE.

Value

a list containing two matrices. The first "result" matrix has a total number of rows equal to the length of vector x. The first column contains a repeat of vector ‘x‘ and the corresponding columns contain ranked ‘y‘ vector matches to the corresponding rows. The matches are ranked from best to worst as column number increases. The second "rank" matrix contains a matrix with equivalent dimension as the first matrix. Instead of containing the matches from ‘y‘, this matrix contains the matching scores of the respective components from the first matrix. A ranking score of 99 implies a perfect match. Perfect matches are isolated for each row.

Author(s)

Atreya Shankar

Examples

```
## Not run:

x <- c("foo.test.xyz", "baz.foosh", "bat")
y <- c("ba","foosba.asd", "bats.at", "foos", "gams.asd")
test <- demystas::grepsParallel(x, y, 2)

## End(Not run)
```

`spNearest` *Detect nearest polygon (country) to point(s)*

Description

Finds nearest polygon (country) to point(s), useful for cases where the ‘sp::over‘ function yields NA results. Should be used with unprojected coordinates íe. lon/lat, ideally on the WGS84 ellipsoid

Usage

```
spNearest(points, global, inc = 100)
```

Arguments

`points`	a data frame with two columns of coordinates (first longitude, then latitude), or a SpatialPoints* object.
`global`	a data frame with two columns of coordinates (first longitude, then latitude), or a SpatialLines* object or a SpatialPolygons* object. This represents the object onto which ‘points‘ is mapped.
`inc`	a numerical value which indicates how much the entire bounding box of ‘global‘ shoud be segmented to find nearest countries. Defaults to 100.

Value

a data frame with attributes from ‘global‘ about nearest polygon (country) to ‘points‘

Author(s)

Atreya Shankar

See Also

`dist2Line`

Examples

```
## Not run:

require(rworldmap)
require(rworldxtra)
points <- rbind(c(-81.779,52.234), c(-80.873, 51.126))
test <- demystas::spNearest(points, getMap(resolution="high"))

## End(Not run)
```

`vectorEnum` *Enumerating vectors*

Description

Enumarates components of a vector in order to make each component unique. Useful for cases where duplicates of vector components must be avoided.

Usage

```
vectorEnum(x, sep = ".")
```

Arguments

`x`	a vector with possibly duplicated components
`sep`	a character indicating how the enumeration and original vector components should be separated. Defaults to "."

Value

a vector in the same order and dimension as ‘x‘ with each component uniquely enumerated

Author(s)

Atreya Shankar

Examples

```
## Not run:

x <- c(rep("foo", 10), rep("bat", 25), rep("baz", 10), rep("foo", 10))
test <- demystas::vectorEnum(x)

## End(Not run)
```

Index