Package 'GrpString'

January 6, 2017

Type Package

Title Patterns and Statistical Differences Between Two Groups of
Strings

Version 0.1.0

Date 2016-12-28

Author Hui (Tom) Tang, Norbert J. Pienta

Maintainer Hui (Tom) Tang <htang2013@gmail.com>

Description Methods include converting series of event names to strings,
discovering common patterns in a group of strings, discovering ``unique'' patterns
when comparing two groups of strings as well as the number and starting position
of each ``unique'' pattern in each string, finding the transition information, and
statistically comparing the difference between two groups of strings.

License GPL-2

Depends R (>= 3.0.1)

Imports utils, plyr, graphics

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2017-01-06 10:29:14

R topics documented:

GrpString-package *Patterns and Statistical Differences between Two Groups of Strings*

Description

Methods include converting series of event names to strings, discovering commen patterns in a group of strings, discovering "unique" patterns when comparing two groups of strings as well as the number and starting position of each "unique" pattern in each string, finding the transition information, and statistically comparing the difference between two groups of strings.

Event names can be any action names such as events in log files or areas of insterst (AOIs) in eye tracking research.

Details

Package:	GrpString
Type:	Package
Version:	0.1.0
Date:	2016-12-28
License:	GPL-2

Some functions have two types, one returning a data frame or a vector and the other exporting one or more than one .txt file to the current directory. The former is a simple version of the functions, while the latter can be considered as a generalized or complex version of the former one. This is because some data sets are large (e.g., many rows or columns), or it helps the users to view and manage results when more than one data set is exported. These function pairs include EveStr - EveString, CommonPatt - CommonPattern, and PatternInfo - UniPatterns.

In addition, to save the users' effort, the function EveString uses an input file (which can be a .txt or .csv file) instead of a data frame. This is because the input data is more convenient to store in a .txt or .csv file than in a data frame. We suggest the users to copy the relevent input files (including eve1d.txt and eve1d.csv) to a different directory, because the function exports files to the same directory where the input files locate.

Author(s)

Hui (Tom) Tang, Norbert J. Pienta

Maintainer: Hui (Tom) Tang <htang2013@gmail.com>

Examples

```
# Discover common patterns in a group of strings
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
CommonPatt(strs.vec, low = 30)
```

CommonPatt	*Discover common patterns in a group of strings - simplified version*

Description

CommonPatt finds common patterns shared by a group of strings.

A common pattern is defined as a substring with the minimum length of three that occurs at least twice among a group of strings.

Usage

```
CommonPatt(strings.vec, low = 10)
```

Arguments

strings.vec   String Vector.

low   Cutoff, which is the minimum percentage of the occurance of patterns that the user specifies. The default value is 10.

Details

The argument 'low' ranges from 0 to 100 in percentage.

Value

Exports a data frame containing patterns, lengths and percentages of patterns.

row name - The initial order of substrings, which can be ignored.

Column 1 - Pattern: common pattern.

Column 2 - Freq_total: the overall frequecy (times of occurance) of each pattern.

Column 3 - Percent_total: the ratio of Freq_total to the number of original strings, in percent.

Column 4 - Length: the length (i.e., number of characters) of pattern.

Column 5 - Freq_str: similar to Freq_total; but each pattern is counted only once in a string even if the string contains that pattern mulitiple times.

Column 6 - Percent_str: similar to Percent; but each pattern is counted only once in a string if this string contains the pattern.

Data is sorted by Length, then Freq, in decreasing order.

References

1. H. Tang; E. Day; L. Kendhammer; J. N. Moore; S. A. Brown; N. J. Pienta. (2016). Eye movement patterns in solving science ordering problems. Journal of eye movement research, 9(3), 1-13.

2. J. J. Topczewski; A. M. Topczewski; H. Tang; L. Kendhammer; N. J. Pienta.(2016). NMR Spectra through the eyes of a student: eye tracking applied to NMR items. Journal of chemical education.

3. J. M. West; A. H. Haake; E. P. Rozanksi; K. S. Karn. (2006). EyePatterns: Software for identifying patterns and similarities across fixation sequences. In Proceedings of the Symposium on Eye-tracking Research & Applications, ACM Press, New York, 149-154.

See Also

CommonPattern

Examples

```
# Simple strings, non-default cutoff
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
CommonPatt(strs.vec, low = 30)
```

CommonPattern	*Discover common patterns in a group of strings - full version*

Description

CommonPattern finds common patterns shared by a group of strings.

It is the extended version of CommonPatt and the ursers have more options.

A common pattern is defined as a substring with the minimum length of three that occurs at least twice among a group of strings.

Usage

```
CommonPattern(strings.vec, low = 5, high = 25, interval = 5, eveChar.df)
```

Arguments

`strings.vec`	String vector.
`low`	The lowest cutoff, which is the minimum percentage of the occurance of patterns that the user specifies. The default value is 5.
`high`	The highest cutoff, which is the maxmum percentage of the occurance of patterns that the user specifies. The default value is 25.
`interval`	The increment percentage from the lowest to the highest cutoff. The default value is 5.
`eveChar.df`	Data frame that stores the event name-character conversion key (optional).

Details

The arguments 'low', 'high' and 'interval' range from 0 to 100 in percentage.

Value

Exports in the current directory a set of text files containing patterns, lengths, percentages of patterns, and converted event names if eveChar.fle is included. The names of these files are the name of strings.vec appended with the percentages. In addtion, a file with all patterns that occurred at least 2 times as well as a file with all possible substrings (including the original strings themselves) are exported.

Note

The time to run this function can be relativle long (from seconds to mimutes depending on the number and lengths of strings as well as the performance of computers).

References

1. H. Tang; E. Day; L. Kendhammer; J. N. Moore; S. A. Brown; N. J. Pienta. (2016). Eye movement patterns in solving science ordering problems. Journal of eye movement research, 9(3), 1-13.

2. J. J. Topczewski; A. M. Topczewski; H. Tang; L. Kendhammer; N. J. Pienta.(2016). NMR Spectra through the eyes of a student: eye tracking applied to NMR items. Journal of chemical education.

3. J. M. West; A. H. Haake; E. P. Rozanksi; K. S. Karn. (2006). EyePatterns: Software for identifying patterns and similarities across fixation sequences. In Proceedings of the Symposium on Eye-tracking Research & Applications, ACM Press, New York, 149-154.

See Also

CommonPatt

Examples

```
# simple strings
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
CommonPattern(strs.vec, low = 30, high = 50, interval = 20)

# None-default cutoff values, with conversion back
data(eventChar.df)
data(str1)
s0 <- str1[5:15]
CommonPattern(s0, low = 20, high = 30, interval = 10, eveChar.df = eventChar.df)
```

---

event1s.df   *Data frame containing event names*

Description

A data frame containing event names, which are from an eye tracking study. Thus, each event name is actually an area of interst (AOI). There are 45 rows. Each row has 26 event names.

Usage

```
data("event1s.df")
```

Format

A data frame with 45 observations or rows.

Examples

```
data(event1s.df)
```

---

eventChar.df   *Event name - character conversion key*

Description

A data frme where each element in column event (event name) corresponds to an element in column char (character), which can be a letter, digit, or a special character.

Usage

```
data("eventChar.df")
```

Format

A data frame with 16 observations on the following 2 variables.

event a character vector

char a character vector

Examples

```
data(eventChar.df)
```

EveStr   *Convert sequences of event names to strings - same length*

Description

EveStr converts event names in a data frame to a string vector. In the data frame, each row, which has the same number of event names, is converted to a string based on the conversion key. A string vector is exported. As a result, in the vector, each converted string has the same length.

Usage

```
EveStr(eveName.df, eveName.vec, char.vec)
```

Arguments

eveName.df	Data frame that stores event names to be converted.
eveName.vec	Event name vector in a conversion key.
char.vec	Character vector in a conversion key.

Details

The lengths of eveName.vec and char.vec are the same.

Each element (event name) in eveName.vec corresponds to an element (character) in char.vec.

An element in char.vec can be a letter, digit, or a special character.

Value

Returns a string vector.

See Also

EveString

Examples

```
# small number of event names
event.df <- data.frame(c("aoi_1", "aoi_2"),
                       c("aoi_1", "aoi_3"),
                       c("aoi_3", "aoi_5"))
event.name.vec <- c("aoi_1", "aoi_2", "aoi_3", "aoi_4", "aoi_5")
label.vec <- c("a", "b", "c", "d", "e")
EveStr(event.df, event.name.vec, label.vec)

# more event names
data(event1s.df)
data(eventChar.df)
EveStr(event1s.df, eventChar.df$event, eventChar.df$char)
```

EveString	*Convert sequences of event names to strings - generalized*

Description

EveString converts event names in a data frame to a string vector.

In the data frame, each row, which can have different number of tokens, is converted to a string based on the conversion key. As a result, in the vector, converted strings may have differente lengths.

Usage

```
EveString(eveName.file, eveName.vec, char.vec)
```

Arguments

eveName.file	File that stores event names to be converted.
eveName.vec	Vector of event names in a conversion key.
char.vec	Characters vector in a conversion key.

Details

In general, it is not convenient to deal with data frames where different rows have different numbers of elements. Thus, it is easier to use a text file than to use a data frame when storing different numbers of event names in rows. As a result, this function uses a .txt or .csv file (for eveName.file) and handle such task to save users' effort.

Value

Exports a vector containing converted strings that usually have different lengths.

If not all event names are not converted to characters, a warning message will be printed out.

Note

eveName.file is the name of a file. Thus quote signs are needed when a file name (and its directory) is directly used in the function.

If the example is used, the eveName.file will be eve1d.txt, which is located in your R library. The users may copy eve1d.txt to a directory that can be easily found.

See Also

EveStr

Examples

```
data(eventChar.df)
event1d <- paste(path.package("GrpString"), "/extdata/eve1d.txt", sep = "")
EveString(event1d, eventChar.df$event, eventChar.df$char)
```

| HistDif | *Customize the positions of legend and p value in a histogram* |

Description

The positions of legend and p value in the histogram generated from function StrDif may not be ideal for different (permutations on differences of normalized Levenstein distances) situations. HistDif customizes the positions of legend and p value in the histogram of the statistical difference of two groups of strings.

Usage

```
HistDif(dif.vec, obsDif, pvalue, o.x = 0.01, o.y = 0, p.x = 0.015, p.y = 0)
```

Arguments

dif.vec	Vector containing differences of normalized Levenstein differences (LD) from the permutation test.
obsDif	The "observed" or original difference between between-group and within-group normalized LD.
pvalue	p value of the permutation test.
o.x	x coordinate of the legend in the histogram, default is 0.01.
o.y	y coordinate of the legend in the histogram, default is 0.
p.x	x coordinate of the p value in the histogram, default is 0.015.
p.y	y coordinate of the p value in the histogram, default is 0.

Details

The default values of o.y and p.y are 0. They are actually related to num_perm: o.y is above 0.2 * num_perm, and p.y is below 0.2 * num_perm. If non-default values are used, the values become absolute y coordinates.

See Also

StrDif

Examples

```
# simple example, use the vectors of ld difference values obtained from StrDifStat
strs1.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
strs2.vec <- c("xYZdkfAxDa", "ef1563xy", "BC9Dzy35X", "AkeC1fxz", "65CyAdC", "Dfy3f69k")
ld.dif.vec <- StrDif(strs1.vec, strs2.vec, num_perm = 500, p.x = 0.025)
HistDif(dif.vec = ld.dif.vec, obsDif = 0.00751, pvalue = 0.35600,
        o.x = 0.025, p.x = 0.040, p.y = 75)
```

---

p1_20up   *Patterns from string group 1*

Description

Patterns that occur at least 20 percent compared to the number of strings in string group 1. It can be obtained from one of the exported files from CommonPattern(str1).

Usage

```
data("p1_20up")
```

Format

The format is: chr [1:32] "212" "202" "BAB" "D0D" "F0F" "020" "B0B" "010" "404" "C0C" ...

Examples

```
data(p1_20up)
```

---

p2_25up   *Patterns from string group 2*

Description

Patterns that occur at least 25 percent compared to the number of strings in string group 2. It can be obtained from one of the exported files from CommonPattern(str2).

Usage

```
data("p2_25up")
```

Format

The format is: chr [1:32] "0D0D" "0E0E" "E0E0" "D0D" "E0E" "F0F" "B0B" "0C0" "0D0" ...

Examples

```
data(p2_25up)
```

PatternInfo   *Discover pattern information in one group of strings*

Description

PatternInfo discovers the start position of each pattern that occurs first or last as well as the number of patterns in each string.

Usage

```
PatternInfo(patterns, strings, rev = FALSE)
```

Arguments

patterns	Pattern vector.
strings	String vector.
rev	Determine whether returning the start positions of patterns that occur first or last. Default is first.

Value

Returns a data frame, which contains the length of each string, and the starting position of each pattern in each string.

See Also

UniPatterns

Examples

```
# simple strings and patterns
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
patts <- c("ABC", "123")
PatternInfo(patts, strs.vec)

# simple strings and patterns, starting position of last pattern
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
patts <- c("ABC", "123")
PatternInfo(patts, strs.vec, rev = TRUE)
```

---

str1   *String group 1*

Description

A vector containing 45 strings that have different lengths. It also can be obtained in the export file from the example in function EveString.

Usage

```
data("str1")
```

Format

The format is: chr [1:45] "D02F0E20DEDC0C30BDC0E45G050A0B5050A06BG0BA5607BA" ...

Examples

```
data(str1)
```

---

str2   *String group 1*

Description

A vector containing 29 strings that have different lengths.

Usage

```
data("str2")
```

Format

The format is: chr [1:29] "G21A1C14C2D0D21D2123201D23D21234320431212412421AB3EGEGE0E4G4B5G6A" ...

Examples

```
data(str2)
```

StrDif   *Statistically compare the difference between two group of strings*

Description

StrDifStat tests whether the difference between two groups of strings is statistically significant or not. The difference is based on normalized Levenstein distances between strings. A permutation test is used as the statistical method.

Usage

```
StrDif(grp1_string, grp2_string, num_perm = 1000,
        o.x = 0.01, o.y = 0, p.x = 0.015, p.y = 0)
```

Arguments

grp1_string   String group (vector) 1.

grp2_string   String group (vector) 2.

num_perm   Number of permutations. The default is 1000.

o.x   x coordinate of the legend in the histogram, default is 0.01.

o.y   y coordinate of the legend in the histogram, default is 0.

p.x   x coordinate of the p value in the histogram, default is 0.015.

p.y   y coordinate of the legend in the histogram, default is 0.

Details

The default values of o.y and p.y are 0. They are actually related to num_perm: o.y is above 0.2 * num_perm, and p.y is below 0.2 * num_perm. If non-default values are used, the values become absolute y coordinates.

Value

The function generates a histogram that demonstrates the distribution of the differences of LDs, the original difference, and the p value.

The function also returns a vector containing differences of normalized Levenstein distances (LD). The total number of differences is num_perm (number of permuations).

Differences are calculated by subtracting within-group LD from between-group LD. They range from -1 to 1. The "observed" difference is the difference from the original data set.

Note

1. Because the number of permutations is usually large (default is 1000), and so is the number of elements in the vector returned from the function, it's better for the user to use a vector to store the returned results, instead of printing out directly. See the examples.

2. The positions of legend and p value in the histogram generated from function StrDif may not be ideal for different (permutations on differences of normalized Levenstein distances) situations. Thus, this package includes another function, HistDif, to customize the positions of legend and p value in the histogram.

3. The time to run this function can be relativle long (from seconds to mimutes depending on the number and lengths of strings as well as the computer performance).

4. The first version of this function was developed with significant help from Dr. Rhonda DeCook in the Department of Statistics and Actuarial Science at the University of Iowa.

References

1. H. Tang; J. J. Topczewski; A. M. Topczewski; N. J. Pienta. Permutation Test for Groups of Scan-paths Using Normalized Levenshtein Distances and Application in NMR Questions. In Proceedings of the Symposium on Eye Tracking Research and Applications, Santa Barbara, CA, March 28-30, 2012; ACM Press: New York; pp 169-172.

2. M. Feusner; B. Lukoff. (2008). Testing for statistically significant differences between groups of scan patterns. In Proceedings of the Symposium on Eye-tracking Research & Applications, ACM Press, New York, 43-46.

See Also

HistDif

Examples

```
# simple stings, non-default permutation number and p-value position
strs1.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
strs2.vec <- c("xYZdkfAxDa", "ef1563xy", "BC9Dzy35X", "AkeC1fxz", "65CyAdC", "Dfy3f69k")
ld.dif.vec <- StrDif(strs1.vec, strs2.vec, num_perm = 500, p.x = 0.025)

# longer strings
data(str1)
data(str2)
s1 <- str1[1:6]
s2 <- str2[1:6]
ld.dif12.vec <- StrDif(s1, s2, num_perm = 500)
```

TransInfo   *Transitions in one group of strings*

Description

TransInfo discovers transitions of of two succesive characters in strings.

A transition is defined as a substring (in the forward order) with length of 2 characters. It can be considered as a special common pattern (length of 2).

Usage

```
TransInfo(strings.vec, type1 = "letters", type2 = "digits")
```

Arguments

strings.vec	String Vector.
type1	The first type of transition. Default value is letter.
type2	The second type of transition. Default value is digit.

Value

Returns a data frame, which contains the numbers of type1 trantions, type2 transtion, and transtions belonging to neither type1 nor type2.

References

1. H. Tang; E. Day; L. Kendhammer; J. N. Moore; S. A. Brown; N. J. Pienta. (2016) Eye movement patterns in solving science ordering problems. Journal of eye movement research, 9(3), 1-13.

2. J. J. Topczewski; A. M. Topczewski; H. Tang; L. Kendhammer; N. J. Pienta.(2016) NMR Spectra through the eyes of a student: eye tracking applied to NMR items. Journal of chemical education.

Examples

```
# default values
strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
TransInfo(strs.vec)

# non-default values
str1.vec <- c("ABCABEF", "CDCDAB")
TransInfo(str1.vec, type1 = "AB", type2 = "CD")
```

UniPatterns   *Discover unique patterns in two groups of strings*

Description

UniPatterns discovers "unique" patterns that are in one group of strings but not the other.

Usage

```
UniPatterns(grp1_pattern, grp2_pattern, grp1_string, grp2_string)
```

Arguments

grp1_pattern   Patterns shared by a centain percent of strings in string group 1.

grp2_pattern   Patterns shared by a centain percent of strings in string group 2

grp1_string   String group 1.

grp2_string   String group 2.

Details

A (common) pattern is defined as a substring with the minimum length of three that occurs at least twice among a group of strings.

A strict definition of unique pattern is that a unique patternis is a pattern that appears in only one of the two groups of strings. However, in practice, a pattern usually is not shared by all the strings in a group. Thus, unique patterns may be obtained from two pattern vectors, each of which contains patterns that are shared by a centain percent of strings in a group. As a result, "unique patterns" can possibly appear in both groups of strings.

Value

Exports five text files:

File that lists unique patterns: column 1 for string group 1; colum 2 for string group 2.

Four files that contain information about each group of patterns in each group of strings.

The information includes the number of each of the patterns in each string and the starting

positions of the first occurring patterns, as well as the lengths of original strings.

If a pattern does not appear in a string, -1 is returned.

In the above four files: the first column contains original strings; the second column contains the length of strings; the third column contains the number of unique patterns each string has each of the column from the fourth is the starting position of a pattern that first appears in a string.

In addition, messages are printed out for the four situations of each pattern group in each string group. The messages include the number and the ratio of strings that have at least one unique pattern.

See Also

PatternInfo, CommonPatt, CommonPattern

Examples

```
data(str1)
data(str2)
data(p1_20up)
data(p2_25up)
UniPatterns(p1_20up, p2_25up, str1, str2)
```

Index