Package ‘hiR’

December 11, 2012

Title Harmony Institute’s toolkit for R

Description Various helper tools for R maintained by HI data lab, since December 2012

Version 0.1

Author Brian Abelson `<brian@harmony-institute.org>`

Maintainer Brian Abelson `<brian@harmony-institute.org>`

License MIT

Depends R (>= 2.15.1), tm, lattice, makeR, classInt, scales,RColorBrewer, plyr, Matrix, glmnet, sentiment, Rstem, lda,rjson, RCurl, stringr, chron, grid, tools

Suggests maps, XML

LazyLoad yes

Collate
’assign_colors.R’ ’calendar_heat_map.R’ ’cbind_fill.R’’classify_sentiment.R’ ’gen_var_names.R’ ’geocode.R’’get_kl

R topics documented:

assign_colors 2
calendar_heat_map 3
cbind_fill 3
classify_sentiment 4
gen_var_names 5
geocode 5
get_klout_scores 6
lda 9
leading_zeros 10
match_gender 11
regress_text 11
word_stemmer 12

Index **13**

assign_colors *Partition a numeric vector into a set of breaks and assign colors*

Description

This function takes an input numeric vector and partitions it into a set number of breaks. It then assigns a color to each break via RColorBrewer

Usage

```
assign_colors(var, n = 9, style = "jenks",
  pal = "Spectral", na_color = "#787878",
  na_omit = FALSE, alph = 1)
```

Arguments

var	Numeric vector to partition
n	Number of colors / breaks
style	Breaks algorithm from "classIntervals" in the "classInt" package. These include: "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", or "jenks"
pal	Palette from RColorBrewer
na_color	Hex code to assign NA values
na_omit	Logical; should the function remove NAs. 'na_color' will be irrelevant if this is TRUE.
alph	Opacity level (0=transparent, 1=opaque)

Value

A data.frame with the variable, break assignments, and color assignments

Examples

```
var <- rnorm(100)
library("hiR")
var_cols <- assign_colors(var)
par(family="HersheySans")
plot(var_cols$var,
     pch=20,
     col=var_cols$col,
     xlab="index",
     ylab="value",
     main="assign_colors example")
```

calendar_heat_map *Create a calendar heat map with a set number of breaks*

Description

This function creates a calendar heat map with custom break values, allowing for comparisions between multiple time series.

Usage

```
calendar_heat_map(dates, values, breaks, ncolors = 9,
  pal = "Spectral", varname = "Values",
  date_form = "%Y-%m-%d")
```

Arguments

`dates`	Vector of dates.
`values`	Numeric vector of values per day.
`breaks`	Vector specifying values to breaks colors at (optional).
`ncolors`	Number of colors to use.
`pal`	Palette from RColorBrewer
`varname`	Name of variable for plot title.
`date_form`	Date format. Defaults to "%Y-%m-%d"

Examples

```
date <- seq(from=as.Date("2010-01-01"),
           to=as.Date("2012-12-31"),
           by='day')
value <- rnorm(length(date), mean = 10, sd=1)
library("hiR")
calendar_heat_map(dates=date, values=value)
```

cbind_fill *Like rbind.fill in plyr but with cbind.*

Description

Like rbind.fill in plyr but with cbind.

Usage

```
cbind_fill(...)
```

Arguments

`...`	data.frames to combine.

`classify_sentiment` *Classify the sentiment of text documents*

Description

This function takes a character vector of documents as an input and returns probabilistic sentiment classification. This function is a slight adjustment to "classify_polarity" in the "sentiment" package. WARNING: This still needs to be tweaked to return meaningul classifications. Use the pos/neg ratio as a better metric for now.

Usage

```
classify_sentiment(text, algorithm = "bayes",
  pstrong = 1, pweak = 0.75, prior = 1,
  neutral_range = c(1, 1.5), verbose = FALSE, ...)
```

Arguments

`text`	A character vector of text blobs.
`algorithm`	A string indicating whether to use the naive bayes algorithm or a simple voter algorithm.
`pstrong`	A numeric specifying the probability that a strongly subjective term appears in the given text.
`pweak`	A numeric specifying the probability that a weakly subjective term appears in the given text.
`prior`	A numeric specifying the prior probability to use for the naive Bayes classifier.
`neutral_range`	# A numeric vector specifying the low and high value of pos/neg ratio to classify as "neutral."
`verbose`	A logical specifying whether to print detailed output regarding the classification process.
`...`	Additional arguments to pass to create_matrix in the sentiment package

Examples

```
documents <- c("I am very happy, excited, and optimistic.",
               "I am very scared, annoyed, and irritated.",
               "Iraq's political crisis entered its second
               week one step closer to the potential
               dissolution of the government, with a call
               for elections by a vital coalition partner
               and a suicide attack that extended the spate
               of violence that has followed the withdrawal
               of U.S. troops.",
               "With nightfall approaching, Los Angeles
               authorities are urging residents to keep their
               outdoor lights on as police and fire officials
               try to catch the person or people responsible
               for nearly 40 arson fires in the last three days.")
library("hiR")
classify_sentiment(documents,algorithm="bayes",verbose=TRUE)
```

gen_var_names *Automatically generate variable names for subsetted dataframes.*

Description

Say you were building a dataset and wanted to automatically generate variable names by some pattern. For instance, you might want to do this with population counts within 100 census tracts by race IE: tracts <- paste("c", rep(1:100), sep=""); race - c("black", "white", "hispanic"); In this case you would want to generate 300 unique variable names This function will generate these variable names automatically when provided with: 1. the "roots" - in the example above, the unique census tracts 2. the "vars" - in the example above, the unique races

Usage

```
gen_var_names(roots, vars, delim = "_")
```

Arguments

roots	A set of names that serve as the root variable
vars	A set of names that represent the subsets of each root variable
delim	Character to separate roots and vars by. Defeaults to "_"

Examples

```
tracts <- paste("ct", rep(1:100), sep="")
race <- c("black", "white", "hispanic")
library("hiR")
gen_var_names(roots=tracts, vars=race)
```

geocode *Geocode strings of text via the Google API*

Description

The function hits the google maps API and tries to geocode strings of text

Usage

```
geocode(uid_location)
```

Arguments

uid_location	A data.frame with one column named "uid" - a vector unique ids and another column named "location" - a vector of strings of text to geocode

Value

A data.frame with the uid, location, lat, lng, and type indicating the geocoding precision

Examples

```
# Generate the data
uid <- paste0("city", 1:5)
location <- c("Boston, MA",
              "New York, NY",
              "Washington D.C.",
              "Philadelphia, PA",
              "Baltimore, MD"
              )
uid_location <- data.frame(uid, location)

# Run geocoding funciton
library("plyr")
library("hiR")
geocoded_data <- ddply(uid_location, .(uid), geocode)
summary(geocoded_data)

# Plot results
#param
par(family="HersheySans")

#map
library("maps")
regions <- c("new hampshire",
               "massachusetts",
               "rhode island",
               "penn",
               "connecticut",
               "washington d.c",
               "new york",
               "new jersey",
               "delaware",
               "maryland"
               )
map("state", region=regions, col="grey80")

#points + labels
points(geocoded_data$lng,
       geocoded_data$lat,
       pch=20,
       cex=2,
       col="steelblue")
text(geocoded_data$lng-0.5,
     geocoded_data$lat+0.3,
     labels=geocoded_data$location,
     cex=1,
     col="darkred")
title("Major Cities on the Eastern Seaboard")
```

---

get_klout_scores *Retrieve klout scores for a vector of twitter handles*

Description

Retrieve klout scores for a vector of twitter handles

Usage

```
get_klout_scores(twitter_handles, api_key,
  na_omit = TRUE)
```

Arguments

`twitter_handles`
A charachter vector of twitter handles - with or without "@"

`api_key` Your api key from http://klout.com/s/developers/

`na_omit` Logical; should the function remove handles that don’t have klout scores

Value

A list data.frame of twitter handles, klout ids, and klout scores

Examples

```
# EXAMPLE ONE:
# simply get a scouple of klout scores
# you can use my apikey for now but it will eventually break
library("hiR")
get_klout_scores(twitter_handles = c("brianabelson", "mhkeller"), api_key="8yng356gnjg37cvn4esbtewy")

# EXAMPLE TWO:
library("hiR")
# Now we’re going to scrape data from twittercounter.com
# to compare klout scores of the 100 twitter users with:
# the highest number of followers
# the highest number of friends
# the highest number of tweets

# STEP ONE: generate data.frame of urls and types
subpages <- seq(0, 80, 20)
follower_base <- 'http://twittercounter.com/pages/100'
follower_urls <- paste0(follower_base, "/", subpages)
follower_df <- data.frame(url = follower_urls,
                          list = rep("followers",  length(follower_urls)),
                          subpage = subpages,
                          stringsAsFactors = FALSE)
friend_base <- 'http://twittercounter.com/pages/friends'
friend_urls <- paste0(friend_base, "/", subpages)
friend_df <- data.frame(url = friend_urls,
                        list = rep("friends", length(friend_urls)),
                        subpage = subpages,
                        stringsAsFactors = FALSE)
tweet_base <- 'http://twittercounter.com/pages/tweets'
tweet_urls <- paste0(tweet_base, "/", subpages)
tweet_df <- data.frame(url = tweet_urls,
                       list = rep("tweets", length(friend_urls)),
                       subpage = subpages,
                       stringsAsFactors = FALSE)
df <- rbind(follower_df, friend_df, tweet_df)

# STEP TWO: Scrape data
# create scraping function
```

```
getHandles <- function(df) {
   # download page
    library("RCurl")
    url <- as.character(df$url)
    page <- getURL(url)
    library("XML")
    tree <- htmlTreeParse(page, useInternalNodes = TRUE)

   # get handles
    handle_nodes <- getNodeSet(tree, '//*[@class="row100user"]/div/a')
    handle <- laply(handle_nodes, function(x) {
                    handles <- xmlGetAttr(x, "href")
                    gsub("/", "", handles)
        })

  # calculate rank
    start <- as.numeric(df$subpage)
    n_handle <- length(handle)
    n1 <- start + 1
    n2 <-  start + n_handle
    rank <- n1:n2

   # list
    list <- rep(df$list, n_handle)

   # return
    data.frame(handle, rank, list, stringsAsFactors=FALSE)
}

# Run scraping function
library("plyr")
twitter_counter <- ddply(df, .(url), getHandles, .progress="text")
twitter_counter <- twitter_counter[,-1]

# STEP THREE: Get Klout scores
library("hiR")
klout_data <- get_klout_scores(twitter_handles=twitter_counter$handle,
                               api_key="8yng356gnjg37cvn4esbtewy",
                               na_omit=FALSE)
df <- data.frame(twitter_counter, klout_data)

# STEP FOUR: plot comparative distributions
# subsets and colors
lists <- unique(df$list)
lists <- data.frame(l = lists, stringsAsFactors=FALSE)
library("scales")
cols <- c("#5f0000", "#005e5f", "#005f30")
lists$col <- alpha(cols, 0.3)

# the plot
#parameters
par(family="serif",
   xaxs="i",
   cex.axis=0.7,
   mai=c(0.8,0.8,0.5,0.3),
   col.axis="grey50",
   lend="round",
   bty="n")
#shell
dummy_x <- as.numeric(df$score[df$list=="followers"])
plot(density(dummy_x),
    type="n",
    xlim=c(0,105),
    xlab="Klout Score",
    main="Density of Klout Scores by Top 100 lists")
#data
for(i in 1:nrow(lists)) {
   to_plot <- df[df$list==lists$l[i],]
   polygon(density(na.omit(as.numeric(to_plot$score))), col=lists$col[i])
}
#legend
legend(x=0,
       y=0.08,
       legend=paste("most", lists$l),
       col=lists$col,
       pch=20,
       bty="n")
```

lda *An easy-to-use and comprehensive implementation of topic modeling in R*

Description

lda is a wrapper for lda.collapsed.gibbs.sampler in the "lda" package. It fits topic models using latent dirichlet allocation, It provides arguments for cleaning the input text and tuning the parameters of the model. it also returns alot of useful information about the topics/documents in a format that you can easily join back to your original data this allows you to easily model outcomes based on the distribution of topics within a collection of texts

Usage

```
lda(text, ids = NULL, lower_case = TRUE,
  remove_stop_words = TRUE, stop_words_to_add = NULL,
  remove_numbers = TRUE, remove_punctuation = TRUE,
  remove_non_ascii = TRUE, stem_words = FALSE,
  char_range = c(2, 50), min_word_count = 5,
  n_topics = 10, n_topic_words = 20, n_iter = 1000,
  burnin = 100, alpha = 0.1, eta = 0.1,
  n_assignments = 3)
```

Arguments

`text` A character vector of text documents

`ids` A vector of ids (to allow joining results to other variables). default is 1:length(text)

`lower_case` Logical; should the function make the text lower case?

`remove_stop_words` Logical; should the function remove stop words? NOTE: this will also make the text lower case

`stop_words_to_add`
A character vector of stopwords to add

`remove_numbers` Logical; should the function remove numbers?

`remove_punctuation`
Logical; should the function remove punctuation?

`remove_non_ascii`
Logical; should the function remove non-ASCII characters?

`stem_words` Logical; should the function stem the words?

`char_range` A numeric vector of length two with low and high value of characters per word (inclusive!) - e.g: c(3,50)

`min_word_count` The number of times a word/feature must occur in a text to be considered

`n_topics` The number of topics to fit

`n_topic_words` The number of top topic words to return

`n_iter` The number of iterations

`burnin` The number of initial iterations to ignore. the function adds burnin to n_iter

`alpha` The scalar value of the dirichlet hyperparameter for topic proportions

`eta` The scalar value of the dirichlet hyperparamater for topic multinomials

`n_assignments` The number of assignments to return (returned as ass_topic_a, ass_topic_b, ass_topic_c, etc.)

Value

A list of length three, including: [[1]] topic_words: A table of the top n words per topic, n = n_topic_words [[2]] document_stats: A data.frame of stats about topics in each document [[3]] topic_words: A table of top topic words in each document

---

`leading_zeros` *Automatically add leading zeros to id columns*

Description

This function quickly and painlessly adds leading zeros to id varibles

Usage

```
leading_zeros(id = NULL, n_digits = NULL)
```

Arguments

`id` A vector of ids

`n_digits` The desired length of each id.

Examples

```
ids <- c("1", "12470192401" , "30479103", "42u1p9241", "532", "3153")
library("hiR")
leading_zeros(id = ids)
```

match_gender *Retrieve gender given a vector of first names \n Warning: only about 60 match_gender(names, full = FALSE) \itemnamesA character vector of names \itemfullLogical; should the function try to extract the first name? WARNING: names like "sarah ann" will turn into "sarah" about 60-70 percent of firstnames from twitter accurately classified Retrieve gender given a vector of first names \n Warning: only about 60 names <- c("cindy", "sally", "bob", "joe") library("hiR") match_gender(names)*

regress_text *Automate lasso/ridge text regression*

Description

Adapted from: https://github.com/johnmyleswhite/TextRegression

Usage

```
regress_text(text, y, stop_words = TRUE,
  stem_words = TRUE, stop_words_to_add = NULL,
  sparse = 0.99, family = "gaussian", alpha = 0.1,
  n_splits = 10, size = 0.8)
```

Arguments

`text,`	A charachter vector of text blobs to use as predictors.
`y`	The outcome variable. Its class depends on the family of regression selected.
`stop_words`	Logical; should the function remove stop words?
`stem_words`	Logical; should the function stem words?
`stop_words_to_add`	A character vector of additional stopwords
`sparse`	Level of sparsity at which a given feature will not be considered (0-1)
`family`	Regression type in glmnet
`alpha`	Alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.
`n_splits`	Number of times to resample data
`size`	How much of the data should be used during resampling for model fitting?

Value

A list with a data.frame of terms and coefficients, and optimal lamda and rmse metrics for model comparison

Examples

```
# from https://github.com/johnmyleswhite/TextRegression

text <- c('saying text is good',
          'saying text once and saying text twice is better',
          'saying text text text is best',
          'saying text once is still ok',
          'not saying it at all is bad',
          'because text is a good thing',
          'we all like text',
          'even though sometimes it is missing')

y <- c(1, 2, 3, 1, 0, 1, 1, 0)

library("hiR")
res <- regress_text(text, y)

print(res[[1]])
```

---

word_stemmer *Stem each feature in a blob of text*

Description

A vecotrizable wrapper for wordStem in the Rstem package

Usage

```
word_stemmer(document)
```

Arguments

document	A blob of text

Examples

```
documents <- c("running runner run", "jumping jump jumped")
library(tm)
corpus <- Corpus(VectorSource(documents))
library("hiR")
as.character(tm_map(corpus, word_stemmer))
```

Index

assign_colors, 2

calendar_heat_map, 3
cbind_fill, 3
classify_sentiment, 4

gen_var_names, 5
geocode, 5
get_klout_scores, 6

lda, 9
leading_zeros, 10

match_gender, 11

regress_text, 11

word_stemmer, 12