Package 'TextForecast'

January 5, 2019

Type Package

Title Forecasting and Regression Analysis using textual data.

Version 0.1.0

Maintainer Lucas Godeiro <lucas.godeiro@hotmail.com>

Description This package carries out forecasting and regression analysis using textual analysis and supervised machine learning techniques as LASSO, Elastic Net and Ridge Regression to select the most predictive words/terms.

License GLP-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports SnowballC, forecast, pdftools, rpart, stats, text2vec, tidyr, tidytext, tm, tsDyn, tseries, vars, wordcloud, dplyr, plyr, udpipe, class, lars, lsa, quantreg, tau, RColorBrewer, forcats, ggplot2, glmnet, tibble

URL https://github.com/lucasgodeiro/TextForecast

BugReports https://github.com/lucasgodeiro/TextForecast/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Luiz Renato Lima [cre], Lucas Godeiro [aut]

R topics documented:

`get_collocations` *get_collocations function*

Description

get_collocations function

Usage

```
get_collocations(corpus_dates, path_name, ntrms, ngrams_number, min_freq)
```

Arguments

`corpus_dates`	a character vector indicating the subfolders where are located the texts.
`path_name`	the folders path where the subfolders with the dates are located.
`ntrms`	maximum numbers of collocations that will be filtered by tf-idf. We rank the collocations by tf-idf in a decreasing order. Then, after we select the words with the ntrms highest tf-idf.
`ngrams_number`	integer indicating the size of the collocations. Defaults to 2, indicating to compute bigrams. If set to 3, will find collocations of bigrams and trigrams.
`min_freq`	integer indicating the frequency of how many times a collocation should at least occur in the data in order to be returned.

Value

a list containing a matrix with the all collocations couting and another with a td-idf filtered collocations counting according to the ntrms.

Examples

```
## Not run:
st_year=2017
end_year=2018
path_name=system.file("news",package="TextForecast")
qt=paste0(sort(rep(seq(from=st_year,to=end_year,by=1),12)),c("m1","m2","m3","m4","m5","m6","m7","m8","m9","
z_coll=get_collocations(corpus_dates=qt[1:23],path_name=path_name,ntrms=500,ngrams_number=3,min_freq=10)

## End(Not run)
```

get_terms *Title*

Description

Title

Usage

```
get_terms(corpus_dates, ntrms_words, st, path.name, ntrms_collocation,
  ngrams_number, min_freq)
```

Arguments

corpus_dates	a character vector indicating the subfolders where are located the texts.
ntrms_words	maximum numbers of words that will be filtered by tf-idf. We rank the word by tf-idf in a decreasing order. Then, we select the words with the ntrms highest tf-idf.
st	set 0 to stem the words and 1 otherwise.
path.name	the folders path where the subfolders with the dates are located.
ntrms_collocation	maximum numbers of collocations that will be filtered by tf-idf. We rank the collocations by tf-idf in a decreasing order. Then, after we select the words with the ntrms highest tf-idf.
ngrams_number	integer indicating the size of the collocations. Defaults to 2, indicating to compute bigrams. If set to 3, will find collocations of bigrams and trigrams.
min_freq	integer indicating the frequency of how many times a collocation should at least occur in the data in order to be returned.

Value

a list containing a matrix with the all collocations and words couting and another with a td-idf filtered collocations and words counting according to the ntrms.

Examples

```
## Not run:
st_year=2017
end_year=2018
path_name=system.file("news",package="TextForecast")
qt=paste0(sort(rep(seq(from=st_year,to=end_year,by=1),12)),c("m1","m2","m3","m4","m5","m6","m7","m8","m9",'
z_terms=get_terms(corpus_dates=qt[1:23],path.name=path_name,ntrms_words=500,ngrams_number=3,st=0,ntrms_coll

## End(Not run)
```

get_words *get_words function*

Description

get_words function

Usage

```
get_words(corpus_dates, ntrms, st, path_name)
```

Arguments

corpus_dates	A vector of characters indicating the subfolders where are located the texts.
ntrms	maximum numbers of words that will be filtered by tf-idf. We rank the word by tf-idf in a decreasing order. Then, we select the words with the ntrms highest tf-idf.
st	set 0 to stem the words and 1 otherwise.
path_name	the folders path where the subfolders with the dates are located.

Value

a list containing a matrix with the all words couting and another with a td-idf filtered words counting according to the ntrms.

Examples

```
## Not run:
st_year=2017
end_year=2018
path_name=system.file("news",package="TextForecast")
qt=paste0(sort(rep(seq(from=st_year,to=end_year,by=1),12)),c("m1","m2","m3","m4","m5","m6","m7","m8","m9","
z_wrd=get_words(corpus_dates=qt[1:23],path_name=path_name,ntrms=500,st=0)

## End(Not run)
```

hard_thresholding *hard thresholding*

Description

hard thresholding

Usage

```
hard_thresholding(x, w, y, p_value, newx)
```

Arguments

x	the input matrix x.
w	the optional input matrix w, that cannot be selected.
y	the response variable.
p_value	the threshold p-value.
newx	matrix that selection will applied. Useful for time series, when we need the observation at time t.

Value

the variables less than p-value.

Examples

```
data("stock_data")
data("optimal_factors")
y=as.matrix(stock_data[,2])
y=as.vector(y)
w=as.matrix(stock_data[,3])
pc=as.matrix(optimal_factors)
t=length(y)
news_factor <- hard_thresholding(w=w[1:(t-1),],x=pc[1:(t-1),],y=y[2:t],p_value = 0.01,newx = pc)
```

---

news_data *News Data*

Description

A simple `tibble` containing the term counting of the financial news from the wall street journal and the news york times from 1992:01 through 2018:11.

Usage

```
news_data
```

Format

A tibble with 1631 components.

dates The vector of dates.

X The terms counting.

optimal_alphas *Title optimal alphas function*

Description

Title optimal alphas function

Usage

```
optimal_alphas(x, w, y, grid_alphas, cont_folds, family)
```

Arguments

x	A matrix of variables to be selected by shrinkrage methods.
w	A matrix or vector of variables that cannot be selected(no shrinkrage).
y	response variable.
grid_alphas	a grid of alphas between 0 and 1.
cont_folds	Set TRUE for contiguous folds used in time depedent data.
family	The glmnet family.

Value

lambdas_opt a vector with the optimal alpha and lambda.

Examples

```
## Not run:
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
X=news_data[,2:ncol(news_data)]
x=as.matrix(X)
grid_alphas=seq(by=0.05,to=0.95,from=0.05)
cont_folds=TRUE
t=length(y)
optimal_alphas=optimal_alphas(x[1:(t-1),],w[1:(t-1),],y[2:t],grid_alphas,TRUE,"gaussian")

## End(Not run)
```

optimal_number_factors *optimal number of factors function*

Description

optimal number of factors function

Usage

```
optimal_number_factors(x, kmax)
```

Arguments

x	a matrix x.
kmax	the maximum number of factors

Value

a list with the optimal factors.

Examples

```
data("optimal_x")
optimal_factor <- optimal_number_factors(x=optimal_x,kmax=8)
```

optimal_x *Optimal x*

Description

A simple matrix containing the optimal words selected by Elastic Net from 1992:01 through 2018:11.

Usage

```
optimal_x
```

Format

A matrix with the most predictive terms.

x The matrix with 4 components.

stock_data *Stock Data*

Description

A simple `tibble` containing the S&P 500 return and the VIX volatility index from 1992:01 through 2018:11.

Usage

```
stock_data
```

Format

A tibble with 3 components.

dates The vector of dates.

sp_return The S&P 500 returns.

vix The volatility index.

text_forecast *Text Forecast function*

Description

Text Forecast function

Usage

```
text_forecast(x, y, h, intercept)
```

Arguments

`x`	the input matrix x.
`y`	the response variable
`h`	the forecast horizon
`intercept`	TRUE for include intercept in the forecast equation.

Value

The h step ahead forecast

Examples

```
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
data("optimal_factors")
pc=optimal_factors
z=cbind(w,pc)
fcsts=text_forecast(z,y,1,TRUE)
```

`text_nowcast` *text nowcast*

Description

text nowcast

Usage

```
text_nowcast(x, y, intercept)
```

Arguments

`x`	the input matrix x. It should have 1 observation more that y.
`y`	the response variable
`intercept`	TRUE for include intercept in the forecast equation.

Value

the nowcast h=0 for the variable y.

Examples

```
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
data("optimal_factors")
pc=optimal_factors
z=cbind(w,pc)
t=length(y)
ncsts=text_nowcast(z,y[1:(t-1)],TRUE)
```

tf_idf *tf-idf function*

Description

tf-idf function

Usage

```
tf_idf(x)
```

Arguments

x	a input matrix x of terms counting.

Value

a list with the terms tf-idf and the terms tf-idf in descending order.

Examples

```
data("news_data")
X=as.matrix(news_data[,2:ncol(news_data)])
```

top_terms *Top Terms Function*

Description

Top Terms Function

Usage

```
top_terms(x, w, y, alpha, lambda, k, wordcloud, max.words, scale, rot.per,
  family)
```

Arguments

x	the input matrix of terms to be selected.
w	optional argument. the input matrix of structured data to not be selected.
y	the response variable
alpha	the glmnet alpha
lambda	the glmnet lambda
k	the k top terms
wordcloud	set TRUE to plot the wordcloud
max.words	the maximum number of words in the wordcloud
scale	the wordcloud size.
rot.per	wordcloud proportion 90 degree terms
family	glmnet family

Value

the top k terms and the corresponding wordcloud.

Examples

```
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
X=news_data[,2:ncol(news_data)]
x=as.matrix(X)
grid_alphas=seq(by=0.05,to=0.95,from=0.05)
cont_folds=TRUE
t=length(y)
optimal_alphas=optimal_alphas(x[1:(t-1),],w[1:(t-1),],y[2:t],grid_alphas,TRUE,"gaussian")
top_trms<- top_terms(x[1:(t-1),],w[1:(t-1),],y[2:t],optimal_alphas[[1]],optimal_alphas[[2]],10,TRUE,10,c(5,
```

`tv_dictionary` *tv dictionary function*

Description

tv dictionary function

Usage

```
tv_dictionary(x, w, y, alpha, lambda, newx, family)
```

Arguments

`x`	A matrix of variables to be selected by shrinkrage methods.
`w`	Optional Argument. A matrix of variables to be selected by shrinkrage methods.
`y`	the response variable.
`alpha`	the alpha required in glmnet.
`lambda`	the lambda required in glmnet.
`newx`	Matrix that selection will applied. Useful for time series, when we need the observation at time t.
`family`	the glmnet family.

Value

X_star: a list with the coefficients and a matrix with the most predictive terms.

Examples

```
## Not run:
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
X=news_data[,2:ncol(news_data)]
x=as.matrix(X)
grid_alphas=seq(by=0.05,to=0.95,from=0.05)
cont_folds=TRUE
t=length(y)
optimal_alphas=optimal_alphas(x[1:(t-1),],w[1:(t-1),],y[2:t],grid_alphas,TRUE,"gaussian")
x_star=tv_dictionary(x=x[1:(t-1),],w=w[1:(t-1),],y=y[2:t],alpha=optimal_alphas[1],lambda=optimal_alphas[2]

## End(Not run)
```

`tv_sentiment_index` *tv sentiment index function*

Description

tv sentiment index function

Usage

```
tv_sentiment_index(x, w, y, alpha, lambda, newx, family, k)
```

Arguments

`x`	A matrix of variables to be selected by shrinkrage methods.
`w`	Optional Argument. A matrix of variables to be selected by shrinkrage methods.
`y`	the response variable.
`alpha`	the alpha required in glmnet.
`lambda`	the lambda required in glmnet.
`newx`	Matrix that selection will applied. Useful for time series, when we need the observation at time t.
`family`	the glmnet family.
`k`	the highest positive and negative coefficients to be used.

Value

The time-varying sentiment index. The index is based on the word/term counting and is computed using: tv_index=(pos-neg)/(pos+neg).

Examples

```
set.seed(1)
data("stock_data")
data("news_data")
y=as.matrix(stock_data[,2])
w=as.matrix(stock_data[,3])
data("news_data")
X=news_data[,2:ncol(news_data)]
x=as.matrix(X)
grid_alphas=seq(by=0.05,to=0.95,from=0.05)
cont_folds=TRUE
t=length(y)
optimal_alphas=optimal_alphas(x[1:(t-1),],w[1:(t-1),],y[2:t],grid_alphas,TRUE,"gaussian")
tv_index <- tv_sentiment_index(x[1:(t-1),],w[1:(t-1),],y[2:t],optimal_alphas[[1]],optimal_alphas[[2]],x,"ga
```

Index