# TextForecast Package

*Luiz Renato Lima*[*]
*Lucas Godeiro*[†]

*January 15, 2019*

## Contents



[*]Department of Economics, University of Tennessee, Knoxville, United States and Department of Economics, Federal University of Paraiba, Brazil. e-mail: llima@utk.edu

[†]Department of Applied Social Sciences, Federal Rural University of the Semi-arid Region – UFERSA, Brazil. e-mail: lucasgodeiro@ufersa.edu.br

# 1 Introduction

This vignettes shows the functions and examples of the **TextForecast** package. The package functions are based on the Lima, Godeiro, and Mohsin (2018) paper and L. Godeiro (2018) Ph.D. thesis.

# 2 Installation

You can install the released version of TextForecast from github with:

```
install.packages("devtools")
library(devtools)
install_github("lucasgodeiro/TextForecast")
```

# 3 get words function

This function counts the words of the texts in the PDF format.

## 3.1 Arguments

**corpus_dates:** A vector of characters indicating the subfolders wher the texts are located.

**ntrms:** maximum numbers of words that will be filtered by tf-idf. We rank the word by tf-idf in a decreasing order. Words are selected based on 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.

### 3.2 Value

a list containing matrices with words counting and a td-idf based word counting.

### 3.3 Example

This is a basic example which shows you how to do a word counting from PDF files. The PDF files contain monthly financial News from The Wall Street Journal and The New York Times between 2017 and 2018.

```
## Example from function get_words.
library(TextForecast)
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","m10","m11","m12"))
z_wrd=get_words(corpus_dates=qt[1:23],path_name=path_name,ntrms=10,st=0)
zz=z_wrd[[2]]
head(zz)
```

## 4 get collocations function

This function counts the collocations of the texts in the PDF format. The PDF files contain monthly financial News from The Wall Street Journal and The New York Times between 2017 and 2018.

### 4.1 Arguments

**corpus_dates:** a character vector indicating the subfolders where the texts are located.

**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.

**ngrams_number:** integer indicating the size of the collocations. Default is set to 2, which computes bigrams. If it is set to 3, then it will find collocations of bigrams and trigrams.

**min_freq:** integer indicating the minimum frequency a collocation should occur in the dataset in order to be returned.

**language:** the texts language. Default is english. The languages available can be found in the link: https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-annotation.html

### 4.2 Value

a list containing matrices with the counting of all collocations and a td-idf based collocation counting.

### 4.3 Example

```
library(TextForecast)
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","m10","m11","m12"))
z_coll=get_collocations(corpus_dates=qt[1:23],path_name=path_name,
ntrms=20,ngrams_number=3,min_freq=10)
zz=z_coll[[2]]
#head(zz)
knitr::kable(head(zz, 23))
```

## 5 get terms function

This function counts the terms of the texts in the PDF format.

### 5.1 Arguments

**corpus_dates:** a character vector indicating the subfolders where the texts are located.

**ntrms_words:** maximum numbers of words that will be filtered by tf-idf. We rank the word by tf-idf in a decreasing order. Words are selected based on 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. Collocations are selected based on the ntrms highest tf-idf.

**ngrams_number:** integer indicating the size of the collocations. Default is set to 2, which computes bigrams. If it is set to 3, then it will find collocations of bigrams and trigrams.

**min_freq:** integer indicating the minimum frequency a collocation should occur in the dataset in order to be returned.

**language:** the texts language. Default is english. The languages available can be found in the link: https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-annotation.html

### 5.2 Value

a list containing matrices with the counting of all collocations (and words) and a td-idf based collocation (and words) counting.

### 5.3 Example

This function counts the words and collocations of the texts in the PDF format. The PDF files contain monthly financial News from The Wall Street Journal and The New York Times between 2017 and 2018.

```
library(TextForecast)
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","m10","m11","m12"))
z_terms=get_terms(corpus_dates=qt[1:23],path.name=path_name,ntrms_words=10,
ngrams_number=3,st=0,ntrms_collocation=10,min_freq=10)
zz=z_terms[[2]]
#head(zz,23)
knitr::kable(head(zz, 23))
```

# 6 tf-idf function

This function computes the terms tf-idf.

## 6.1 Arguments

**x:** a input matrix x of terms counting.

## 6.2 Value

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

## 6.3 Example

```
library(TextForecast)
 data("news_data")
 X=as.matrix(news_data[,2:ncol(news_data)])
  tf_idf=tf_idf(X)
  head(tf_idf[[1]])
```

# 7 optimal alphas function

This functions computes the optimal alphas.

## 7.1 Arguments

**x:** A matrix of variables to be selected by shrinkrage methods.

**w:** Optional Argument. 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.

### 7.2 Value

**lambdas_opt:** a vector with the optimal alpha and lambda.

### 7.3 Example

Stock data is a simple tibble containing the S&P 500 return and the VIX volatility index from 1992:01 through 2018:11.

News data is a simple tibble containing the term counting of the financial news from the Wall Street Journal and the New York Times from 1992:01 to 2018:11.

```
library(TextForecast)
 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")
 print(optimal_alphas)
```

## 8 tv dictionary function

This functions selects from $X$ the most predictive terms $X^*$ using supervised machine learning techniques(Elastic Net).

### 8.1 Arguments

**x:** A matrix of variables to be selected by shrinkrage methods.

**w:** Optional Argument. A matrix or vector of variables that cannot be selected(no shrinkrage).

**y:** 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.

### 8.2 Value

$X_t^*$: a list with the coefficients and a matrix with the most predictive terms.

### 8.3 Example

This example select the most predictive words from the news database. The news database contains the terms counting of the Wall street journal and the New York Times financial news.

```
library(TextForecast)
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=x[1:(t-1),],w=w[1:(t-1),],
y=y[2:t],grid_alphas=grid_alphas,cont_folds=TRUE,family="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],newx=x,family="gaussian")
optimal_alphas1=optimal_alphas(x=x[1:(t-1),],y=y[2:t],
grid_alphas=grid_alphas,cont_folds=TRUE,family="gaussian")
x_star1=tv_dictionary(x=x[1:(t-1),],y=y[2:t],alpha=optimal_alphas1[1],
lambda=optimal_alphas1[2],newx=x,family="gaussian")
```

## 9 Optimal number of factors function

This function computes the optimal number of factors according to Ahn and Horenstein (2013).

### 9.1 Arguments

**x:** a input matrix X.

**kmax:** the maximum number of factors.

### 9.2 Value

a list with the optimal factors.

### 9.3 Example

```
library(TextForecast)
data("optimal_x")
optimal_factor <- TextForecast::optimal_number_factors(x=optimal_x,kmax=8)
head(optimal_factor[[1]])
```

# 10 Hard thresholding function

This function carries out the hard thresholding according to Bai and Ng (2008)

## 10.1 Arguments

**x:** the input matrix x.

**w:** Optional Argument. 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.

## 10.2 Value

the variables less than p-value.

## 10.3 Example

```
library(TextForecast)
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)
```

# 11 Text Forecast function

This functions computes the $h$ step ahead forecast based on textual and/or economic data.

## 11.1 Arguments

**x:** the input matrix x.

**y:** the response variable

**h:** the forecast horizon

**intercept:** TRUE for include intercept in the forecast equation.

## 11.2 Value

The h step ahead forecast

### 11.3 Example

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

## 12 Text Nowcast function

This function computes the nowcast h=0.

### 12.1 Arguments

**x:** the input matrix x.

**y:** the response variable

**intercept:** TRUE for include intercept in the forecast equation.

### 12.2 Value

the nowcast h=0 for the variable y.

### 12.3 Example

```
library(TextForecast)
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=as.matrix(optimal_factors)
z=cbind(w,pc)
t=length(y)
ncsts=text_nowcast(z,y[1:(t-1)],TRUE)
print(ncsts)
```

## 13 Top Terms function

This function computes the highest k predictive words by using the highest absolute coefficient value.

### 13.1 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

### 13.2 Value

the top k terms and the corresponding wordcloud.

### 13.3 Example

```
library(TextForecast)
 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(2,0.3),.15,"gaussian")
```

## 14 TV sentiment index function

### 14.1 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.

### 14.2 Value

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

### 14.3 Example

```
library(TextForecast)
 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,"gaussian",2)
 head(tv_index)
```

## References


Ahn, Seung C, and Alex R Horenstein. 2013. "Eigenvalue Ratio Test for the Number of Factors." *Econometrica* 81 (3). Wiley Online Library: 1203–27.

Bai, Jushan, and Serena Ng. 2008. "Forecasting Economic Time Series Using Targeted Predictors." *Journal of Econometrics* 146 (2). Elsevier: 304–17.

Godeiro, Lucas. 2018. "Ensaios Sobre Modelos de Previsao Economica." Universidade Federal da Paraíba.

Lima, Luiz Renato, Lucas Lúcio Godeiro, and Mohammed Mohsin. 2018. "Time-Varying Dictionary and the Predictive Power of Fed Minutes." In *2018 Ciret Biennial International Conference*.