Practice Assignment 6

Import speeches.csv and the EUR/US reference exchange rate (fx.csv)

```
speeches <- read.csv("speeches.csv", header = TRUE, quote = "", sep = "|" )[ ,c('date', 'contents')]
fx <- read.csv("fx.csv", header = FALSE, sep = ",", col.names = c("date", "rate", "status", "comment"))

#chech the data headers
head(fx)
```

```
##          date   rate           status comment
## 7  2021-12-03 1.1291 Normal value (A)
## 8  2021-12-02 1.1339 Normal value (A)
## 9  2021-12-01 1.1314 Normal value (A)
## 10 2021-11-30 1.1363 Normal value (A)
## 11 2021-11-29 1.1276 Normal value (A)
## 12 2021-11-26 1.1291 Normal value (A)
```

```
head(speeches)
```

```
##         date
## 1 2021-11-29
## 2 2021-11-29
## 3 2021-11-26
## 4 2021-11-25
## 5 2021-11-25
## 6 2021-11-24
##
y one of the cornerstones of our strategy in a changing world Lecture by Christine Lagarde, symmetric 2% target
## 1    SPEECH  Our strategy in a changing world  Lecture by Christine Lagarde, President of the
## 2
 specific provisions which allow the expansion of the tasks and powers assigned to the EU and its institutions. These
## 3
```

```
fx <- fx[, 1:3]

head(fx)
```

```
##          date   rate           status
## 7  2021-12-03 1.1291 Normal value (A)
## 8  2021-12-02 1.1339 Normal value (A)
## 9  2021-12-01 1.1314 Normal value (A)
## 10 2021-11-30 1.1363 Normal value (A)
## 11 2021-11-29 1.1276 Normal value (A)
## 12 2021-11-26 1.1291 Normal value (A)
```

Convert the date from character to POSIX and rate to numeric:

```
speeches$date <- as.POSIXct(speeches$date) # converts into POSIX
fx$date <- as.POSIXct(fx$date) # converts into POSIX
fx$rate <- as.numeric(as.character(fx$rate)) # converts rate into numerical
```

```
## Warning: NAs introduced by coercion
```

```
#check classesd
class(speeches$date)
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(fx$date)
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(fx$rate)
```

```
## [1] "numeric"
```

Join the dataframes

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
joined <- fx %>% left_join(speeches) %>% group_by(date) %>% summarise_each(funs(max))
```

```
## Warning: `summarise_each_()` was deprecated in dplyr 0.7.0.
## Please use `across()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

## Joining, by = "date"
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
summary(joined) #Summarize df to spot obvious issues
```

```
##       date                          rate           status
##  Min.   :1999-01-04 00:00:00   Min.   :0.8252   Length:5932
##  1st Qu.:2004-09-08 18:00:00   1st Qu.:1.1025   Class :character
##  Median :2010-05-17 12:00:00   Median :1.1983   Mode  :character
##  Mean   :2010-05-31 03:36:53   Mean   :1.1992
##  3rd Qu.:2016-02-18 06:00:00   3rd Qu.:1.3164
##  Max.   :2021-12-03 00:00:00   Max.   :1.5990
##                                NA's   :62
##    contents
##  Length:5932
##  Class :character
##  Mode  :character
##
##
##
##
```

```
head(joined)
```

```
## # A tibble: 6 x 4
##   date                 rate status           contents
##   <dttm>              <dbl> <chr>            <chr>
## 1 1999-01-04 00:00:00  1.18 Normal value (A) <NA>
## 2 1999-01-05 00:00:00  1.18 Normal value (A) <NA>
## 3 1999-01-06 00:00:00  1.17 Normal value (A) <NA>
## 4 1999-01-07 00:00:00  1.16 Normal value (A) <NA>
## 5 1999-01-08 00:00:00  1.17 Normal value (A) <NA>
## 6 1999-01-11 00:00:00  1.16 Normal value (A) <NA>
```

It doesn't look like there are any extreme outliers, min & max are in the expected range.

We have missing status values on 21.02.2003 & 24.02.2004, but both seem correct: So lets also keep them for now.

There is 62 NA's we can fix:

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
NA_fix <- na.locf(joined[,2], fromLast = TRUE) # this replaces NA with previous value.

joined <- NA_fix %>% left_join(joined) %>% group_by(date) %>% summarise_each(funs(max))  #joins the df
```

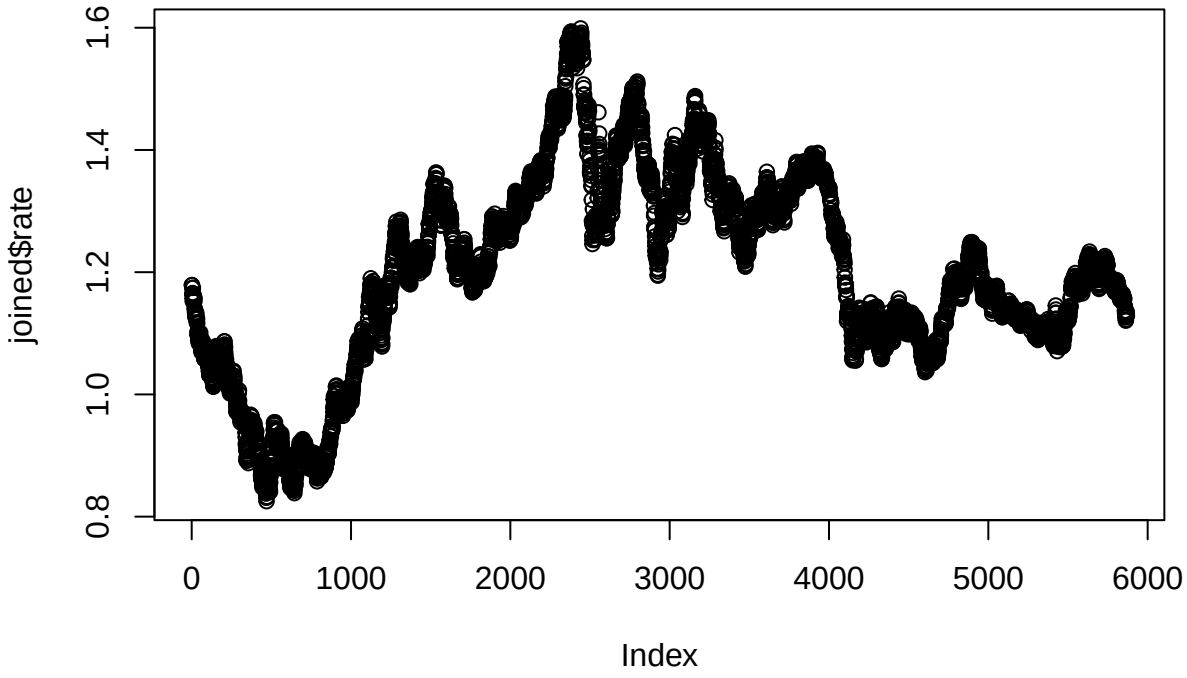
```
## Joining, by = "rate"
```

```
summary(joined) #Na's should not be present in the summary now
```

```
##       date                          rate           status
##  Min.   :1999-01-04 00:00:00   Min.   :0.8252   Length:5870
##  1st Qu.:2004-09-27 06:00:00   1st Qu.:1.1025   Class :character
##  Median :2010-06-21 12:00:00   Median :1.1983   Mode  :character
##  Mean   :2010-06-18 04:25:28   Mean   :1.1992
##  3rd Qu.:2016-03-10 18:00:00   3rd Qu.:1.3164
##  Max.   :2021-12-03 00:00:00   Max.   :1.5990
##    contents
##  Length:5870
##  Class :character
##  Mode  :character
##
##
##
```

Let's try to plot the data and see if we visually can spot any outliers

```
plot(joined$rate)
```



Now we can calculate the exchange rate return and extend the dataset with variable "good_news" and "bad_news"

```
add_return <- mutate(joined, return = rate / lag(rate)) #adds return column

df <- mutate(add_return, good_news = return >= 1.005) %>%
  mutate(add_return, bad_news = return <= 0.995) # adds good and bad news variables
```

```
#convert to numeric boolean-values

df$good_news <- as.numeric(df$good_news)
df$bad_news <- as.numeric(df$bad_news)
head(df)
```

```
## # A tibble: 6 x 7
##   date                rate status           contents return good_news bad_news
##   <dttm>             <dbl> <chr>            <chr>     <dbl>     <dbl>    <dbl>
## 1 1999-01-04 00:00:00  1.18 Normal value (A) <NA>     NA           NA       NA
## 2 1999-01-05 00:00:00  1.18 Normal value (A) <NA>      1.00         0        0
## 3 1999-01-06 00:00:00  1.17 Normal value (A) <NA>      0.996        0        0
## 4 1999-01-07 00:00:00  1.16 Normal value (A) <NA>      0.991        0        1
## 5 1999-01-08 00:00:00  1.17 Normal value (A) <NA>      1.00         0        0
## 6 1999-01-11 00:00:00  1.16 Normal value (A) <NA>      0.992        0        1
```

Below we are creating csv tables with good and bad indicators words for the exchange rate.

```
#First we remove NA's from the contents columns
df_NA_fix <- na.omit(df[,4]) # this removes NA values in the 'df' dataframe.

# Dataframe for good indicators
df_good <- df_NA_fix %>% left_join(df) %>% group_by(date) %>% summarise_each(funs(max))%>%
    select(contents, good_news)%>%
    filter(good_news == 1)
```

```
## Joining, by = "contents"
```

```
# Dataframe for bad indicators
df_bad <- df_NA_fix %>% left_join(df) %>% group_by(date) %>% summarise_each(funs(max))%>%
    select(contents, bad_news)%>%
    filter(bad_news == 1)
```

```
## Joining, by = "contents"
```

```
#clean the data and find the common words associated with good and bad indicators
library(tidytext)
library(ggplot2)
library(reshape2)

# for good indicators:
words1 <- data_frame(Text = df_good$contents) # tibble
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
good_words <- words1 %>%
  unnest_tokens(output = word, input = Text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE )
```

```
## Joining, by = "word"
```

```
head(good_words, n=23)
```

```
## # A tibble: 23 x 2
##    word          n
##    <chr>     <int>
##  1 euro       5359
##  2 â           5154
##  3 financial  4834
##  4 policy     4660
##  5 monetary   4286
##  6 de         3613
##  7 economic   2754
##  8 market     2737
##  9 central    2664
## 10 stability  2468
## # ... with 13 more rows
```

```
# for bad indicators:

words2 <- data_frame(Text = df_bad$contents)

bad_words <- words2 %>%
  unnest_tokens(output = word, input = Text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE )
```

```
## Joining, by = "word"
```

```
head(bad_words, n=23)
```

```
## # A tibble: 23 x 2
##    word          n
##    <chr>     <int>
##  1 euro       6353
##  2 â           5519
##  3 financial  5083
##  4 policy     4636
##  5 monetary   4314
##  6 market     3079
##  7 de         3058
##  8 banks      2918
##  9 central    2904
## 10 economic   2793
## # ... with 13 more rows
```

It's clear that we have to customize the stop words a little bit:

```
stop.w <- c("la", "der", "term", "de", "â", "die", "ã", "und")
stop.c <- melt(stop.w, id.vars = c('word'))
  colnames(stop.c)[1] <- "word"
stop.c
```

```
##   word
## 1   la
## 2  der
## 3 term
## 4   de
## 5    â
## 6  die
## 7    ã
## 8  und
```

```
stop_words2 <- stop.c %>% full_join(stop_words)
```

```
## Joining, by = "word"
```

Now lets re-run our code with the custom stop words:

```r
# for good indicators:
words1 <- data_frame(Text = df_good$contents) # tibble

good_words <- words1 %>%
  unnest_tokens(output = word, input = Text) %>%
  anti_join(stop_words2) %>%
  count(word, sort = TRUE )
```

```
## Joining, by = "word"
```

```r
head(good_words, n=23)
```

```
## # A tibble: 23 x 2
##    word          n
##    <chr>     <int>
##  1 euro       5359
##  2 financial  4834
##  3 policy     4660
##  4 monetary   4286
##  5 economic   2754
##  6 market     2737
##  7 central    2664
##  8 stability  2468
##  9 ecb        2317
## 10 banks      2309
## # ... with 13 more rows
```

```r
# for bad indicators:

words2 <- data_frame(Text = df_bad$contents)

bad_words <- words2 %>%
  unnest_tokens(output = word, input = Text) %>%
  anti_join(stop_words2) %>%
  count(word, sort = TRUE )
```

```
## Joining, by = "word"
```

```r
head(bad_words, n=23)
```

```
## # A tibble: 23 x 2
##    word          n
##    <chr>     <int>
##  1 euro       6353
##  2 financial  5083
##  3 policy     4636
##  4 monetary   4314
##  5 market     3079
##  6 banks      2918
##  7 central    2904
##  8 economic   2793
```

```
##  9 european   2552
## 10 stability  2438
## # ... with 13 more rows
```

```
# save good indicator table to csv:

good_indicators <- good_words %>% head(good_words, n=20)
write.csv(good_indicators, file = "good_indicators.csv")

# bad indicator table to csv:
bad_indicators <- bad_words %>% head(bad_words, n=20)
write.csv(bad_indicators, file = "bad_indicators.csv")
```

we could also produce wordclouds for the common words

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
# wordcloud for good indicators
pall = brewer.pal(5,"GnBu")
good_cloud <-wordcloud(words = good_indicators$word,
          freq = good_indicators$n,
          scale = c(5,.3),
          random.order = FALSE,
          random.color = FALSE,
          colors = rev(pall))
```


```
# wordcloud for bad indicators
pal12 = brewer.pal(5,"PuRd")
bad_cloud <- wordcloud(words = bad_indicators$word,
          freq = bad_indicators$n,
          scale = c(5,.3),
          random.order = FALSE,
          random.color = FALSE,
          colors = rev(pal12))
```

Almost the same words and their frequency has an impact, i.e. The data tells us that a bads news for euro = bad_indicator and likewise the other way around.