Data - Import and Preparation

Stefan Glogger

August 2017

Data Import

We import the sentiment data. We also import the prices of each index over the relevant time frame.

Sentix

Read the raw sentiment data and save it in the list *sentixRaw* with each list element containing the results of the survey for the different indices. As the number of rows (dates of observation) in data differ, we extract the unique dates (*datesSentix*) and reduce the data to it. We also determine *min(datesSentix)* and *max(datesSentix)*, which we use lateron to get the stock data.

```
# install.packages("openxlsx")
library(openxlsx)
```

```
folderSentix <- (file.path(getwd(), "Data", "Sentix"))

sheets <- c("DAX","DAXm","TEC","TECm","ESX50","ESX50m","SP5","SP5m","NASDAQ","NASDAQm","NIKKEI","NIKKEIm
relevant_rows <- c("Datum","P+","Pn","P-","I+","In","I-","G+","Gn","G-")
```

```
sentixRaw <- list()

for(i in sheets){
  sentixRaw[[i]] <- read.xlsx(file.path(folderSentix, "sentix_anzahlen_bis_02092016xlsx.xlsx"),sheet=i,
  sentixRaw[[i]] <- sentixRaw[[i]][,relevant_rows]
  sentixRaw[[i]] <- sentixRaw[[i]][order(sentixRaw[[i]][,1]),]
}
```

```
unlist(lapply(sentixRaw, nrow))
```

```
##     DAX    DAXm     TEC    TECm   ESX50  ESX50m     SP5    SP5m  NASDAQ
##     803     803     803     803     803     803     803     803     803
## NASDAQm  NIKKEI NIKKEIm    BUND   BUNDm   TBOND  TBONDm
##     803     803     803     802     802     802     802
```

```
datesSentix <- unique(sentixRaw[[1]]$Datum)
for(i in names(sentixRaw)[2:length(sentixRaw)]){
  if(!(setequal(datesSentix, sentixRaw[[i]]$Datum)))
    stop("Sentix Data of different indices have not same dates. Handle manually.")
}

for(i in names(sentixRaw)){
  sentixRaw[[i]] <- unique(sentixRaw[[i]])
}
unlist(lapply(sentixRaw, nrow))
```

```
##     DAX    DAXm     TEC    TECm   ESX50  ESX50m     SP5    SP5m  NASDAQ
##     802     802     802     802     802     802     802     802     802
## NASDAQm  NIKKEI NIKKEIm    BUND   BUNDm   TBOND  TBONDm
##     802     802     802     802     802     802     802
```

```r
rm(folderSentix, sheets, relevant_rows, i)
detach("package:openxlsx", unload = T)
```

Stocks

We take data mainly from Yahoo Finance. We take closing course from *min(datesSentix)* to *max(datesSentix)* for several indexes and store in the data frame *stocks* the closing stock price at each date of the sentiment data (*datesSentix*).

We take the following as sources of the data:

- DAX *^GDAXI*
- TEC *^TECDAX*
- ESX50 *^STOXX50E*
- SP500 *^GSPC*
- NASDAQ *^NDX*
- NIKKEI *^N225*
- BUND from Sebastian: Den Bund-Future habe ich bei onvista in 5-Jahresst?cken geladen und zusammengebaut. Dezimaltrennzeichen umgestellt im .csv —- not from yahoo, manually from bundesbank *BBK01.WT0557*
- TBOND from Sebastian: Beim T-Bond ist es die 10 Year Treasury Note, auf welche das TBOND Sentiment abzielt. Diese habe ich bei FRED geladen: https://fred.stlouisfed.org/series/DGS10

```r
# install.packages("quantmod")
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

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

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

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
# ?getSymbols
```

```r
stocks <- data.frame(Datum = datesSentix)

# DAX
dax <- new.env()
getSymbols("^GDAXI", env = dax, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## Warning: ^GDAXI contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "GDAXI"
```

```r
DAX <- data.frame(dax$GDAXI[datesSentix,"GDAXI.Close"])
colnames(DAX) <- "Close" # somehow the column name cannot be given directly
DAX$Datum <- as.Date(row.names(DAX))

stocks$DAX <- merge(stocks, DAX, by = "Datum", all.x = T)$Close



# TEC
tec <- new.env()
getSymbols("^TECDAX", env = tec, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## Warning: ^TECDAX contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "TECDAX"
```

```r
TEC <- data.frame(tec$TECDAX[datesSentix, "TECDAX.Close"])
colnames(TEC) <- "Close"
TEC$Datum <- as.Date(row.names(TEC))

stocks$TEC <- merge(stocks, TEC, by = "Datum", all.x = T)$Close



# ESX50
esx50 <- new.env()
getSymbols("^STOXX50E", env = esx50, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## Warning: ^STOXX50E contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "STOXX50E"
```

```r
ESX50 <- data.frame(esx50$STOXX50E[datesSentix,"STOXX50E.Close"])
colnames(ESX50) <- "Close"
ESX50$Datum <- as.Date(row.names(ESX50))

stocks$ESX50 <- merge(stocks, ESX50, by = "Datum", all.x = T)$Close



# SP500
sp500 <- new.env()
getSymbols("^GSPC", env = sp500, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## [1] "GSPC"
```

```
SP500 <- data.frame(sp500$GSPC[datesSentix,"GSPC.Close"])
colnames(SP500) <- "Close"
SP500$Datum <- as.Date(row.names(SP500))
# sum(is.na(SP500$Close))

stocks$SP5 <- merge(stocks, SP500, by = "Datum", all.x = T)$Close


# NASDAQ
nasdaq <- new.env()
getSymbols("^NDX", env = nasdaq, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## [1] "NDX"
```

```
NASDAQ <- data.frame(nasdaq$NDX[datesSentix,"NDX.Close"])
# sum(is.na(NASDAQ[,"NDX.Close"]))
colnames(NASDAQ) <- "Close"
NASDAQ$Datum <- as.Date(row.names(NASDAQ))

stocks$NASDAQ <- merge(stocks, NASDAQ, by = "Datum", all.x = T)$Close


# NIKKEI
nikkei <- new.env()
getSymbols("^N225", env = nikkei, src = "yahoo", from = min(datesSentix), to = max(datesSentix))
```

```
## Warning: ^N225 contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "N225"
```

```
NIKKEI <- data.frame(nikkei$N225[datesSentix,"N225.Close"])
colnames(NIKKEI) <- "Close"
NIKKEI$Datum <- as.Date(row.names(NIKKEI))

stocks$NIKKEI <- merge(stocks, NIKKEI, by = "Datum", all.x = T)$Close
```

Bund

```
BUND <- read.csv(file.path(getwd(), "Data", "Bundfuture", "Bundfuture2001-2017.csv"), sep = ";")
BUND[,1] <- as.Date(BUND[,1], format = "%d.%m.%Y")
BUND <- BUND[BUND[,1] %in% datesSentix,]
BUND <- as.data.frame(BUND)

stocks$BUND <- merge(stocks, BUND, by = "Datum", all.x = T)$Schluss
```

Treasury bond

```
TBOND <- read.csv(file.path(getwd(), "Data", "10 year T-Notes", "DGS10.csv"), sep = ",")
TBOND[,1] <- as.Date(TBOND[,1], format = "%Y-%m-%d")
TBOND[,2] <- as.numeric(as.character(TBOND[,2])) # was a factor first and factors are stored via index
```

```
## Warning: NAs durch Umwandlung erzeugt
```

```r
colnames(TBOND) <- c("Datum", "DGS10")
TBOND <- TBOND[TBOND[,1] %in% datesSentix,]
TBOND <- as.data.frame(TBOND)

stocks$TBOND <- merge(stocks, TBOND, by = "Datum", all.x = T)$DGS10
```

```r
rm(BUND, DAX, ESX50, NASDAQ, NIKKEI, SP500, TBOND, TEC,
   dax, esx50, nasdaq, nikkei, sp500, tec)
detach("package:quantmod", unload = T)
```

Data Preparation

We look at how many people participated in the survey on average and remove TBOND.

We look at the number of dates on which not all stocks report prices and remove those to end up with the dates on which all data is available *datesAll*.

Sentix - number of participants in survey

NOTE: maybe also delete the “G” columns in the sentix data lateron (but it might produce quite interesting results)

```
cols <- 8:10
colnames(sentixRaw[[1]])[cols]
```

```
## [1] "G+" "Gn" "G-"
```

```
unlist(lapply(sentixRaw, function(x) {round(mean(rowSums(x[cols])), 0)}))
```

```
##     DAX    DAXm     TEC    TECm   ESX50  ESX50m     SP5    SP5m  NASDAQ
##     701     698     677     674     696     692     694     690     683
## NASDAQm  NIKKEI NIKKEIm    BUND   BUNDm   TBOND  TBONDm
##     680     647     643     628     625     160     160
```

```
rm(cols)
```

We remove TBOND, as just very few people voted for it over time in comparison to the other indices.

```
sentixRaw[["TBOND"]] <- NULL
sentixRaw[["TBONDm"]] <- NULL
stocks <- stocks[,-which(colnames(stocks)=="TBOND")]
```

```
unlist(lapply(sentixRaw, function(x) {sum(is.na.data.frame(x))}))
```

```
##     DAX    DAXm     TEC    TECm   ESX50  ESX50m     SP5    SP5m  NASDAQ
##       0       0       0       0       0       0       0       0       0
## NASDAQm  NIKKEI NIKKEIm    BUND   BUNDm
##       0       0       0       0       0
```

Stocks - na’s

There might be dates missing (we just have to look at stocks as we found the *datesSentix* as those dates, for which all sentiment is there).

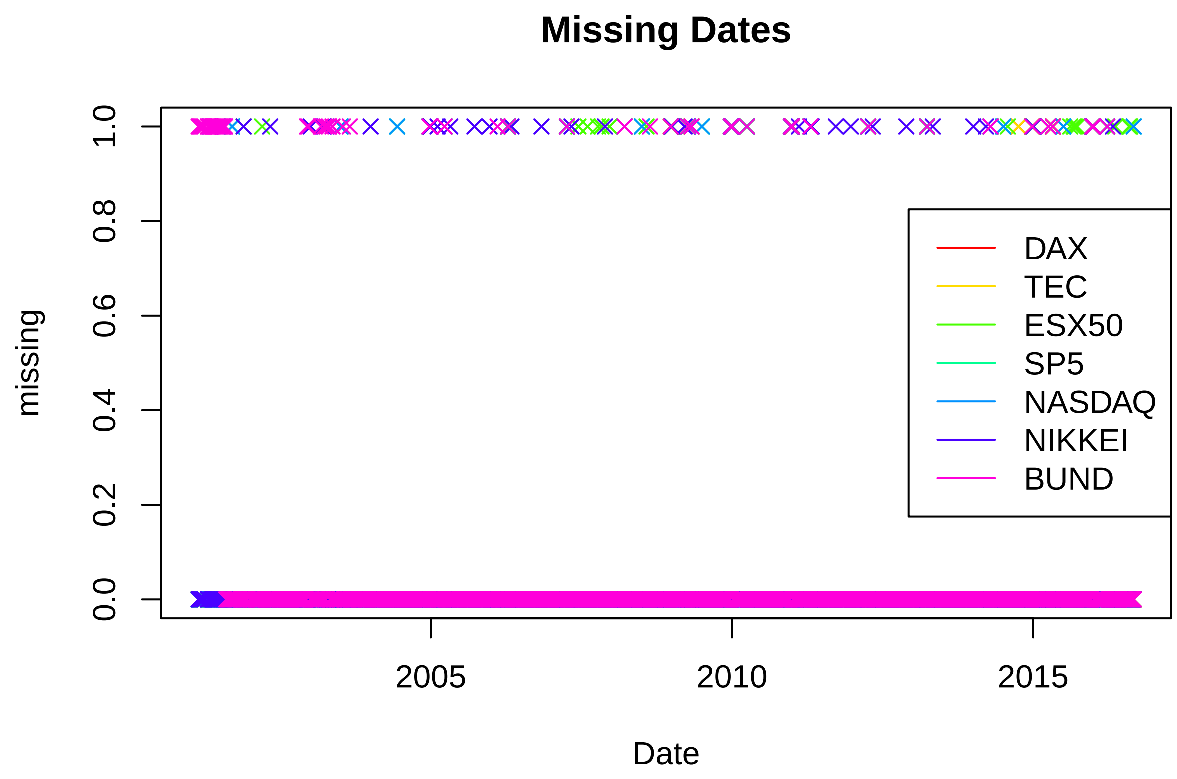
```
colSums(is.na.data.frame(stocks))
```

```
##  Datum    DAX    TEC  ESX50    SP5 NASDAQ NIKKEI   BUND
##      0     25     22     41     26     26     32     56
```

Visualize the missing dates (missing date = 1, not missing date = 0 on y-axis).

```
cols <- rainbow(ncol(stocks)-1)

plot(stocks[,1], is.na(stocks[,2]), main = "Missing Dates", ylab = "missing", xlab = "Date", col = cols
for(i in 2:(ncol(stocks)-1)){
    par(new=T)
    plot(stocks[,1], is.na(stocks[,i+1]), col = cols[i], axes = F, xlab = "", ylab = "", pch = 4)
}
legend("right", legend = colnames(stocks)[2:ncol(stocks)], col = cols, lty = 1)
```



```
pdf(file.path(getwd(), "Plot", "missingDates.pdf"), width = 10, height = 4)
cols <- rainbow(ncol(stocks)-1)
plot(stocks[,1], is.na(stocks[,2]), main = "Missing Dates", ylab = "missing", xlab = "Date", col = cols
for(i in 2:(ncol(stocks)-1)){
    par(new=T)
    plot(stocks[,1], is.na(stocks[,i+1]), col = cols[i], axes = F, xlab = "", ylab = "", pch = 4)
}
legend("right", legend = colnames(stocks)[2:ncol(stocks)], col = cols, lty = 1)
dev.off()
```

```
## pdf
##   2
```

```
rm(cols, i)
```

Determine, how many dates do have all data available.

```
nrow(stocks)
```

```
## [1] 802
```

```
nrow(stocks[complete.cases(stocks),])
```

```
## [1] 695
```

```r
nrow(stocks) - nrow(stocks[complete.cases(stocks),])
```

```
## [1] 107
```

```r
(nrow(stocks) - nrow(stocks[complete.cases(stocks),]))/nrow(stocks)
```

```
## [1] 0.1334165
```

So we would delete 13.3416459 % of the data.

delete

We delete dates with missing values.

```r
stocks <- stocks[complete.cases(stocks),]

datesAll <- stocks[,1]
rm(datesSentix)

sentixRaw <- lapply(sentixRaw, function(x) {x[(x[,1] %in% datesAll),]})

unlist(lapply(sentixRaw, nrow))
```

```
##     DAX    DAXm     TEC    TECm   ESX50  ESX50m     SP5    SP5m  NASDAQ 
##     695     695     695     695     695     695     695     695     695 
## NASDAQm  NIKKEI NIKKEIm    BUND   BUNDm 
##     695     695     695     695     695
```

other approach (not implemented)

One way of approaching this might be via linear regression of the stock data when no stock price is available. but this assumes a linear relationship and might cause trouble.