The Magic in R

Stefan Glogger

17 August 2017

Overview

Please look at the titles to get an overview of what is done when. You can also refer to the introducing sentences of each main title.

Packages, Functions and Parameters

First we load all the relevant packages (are saved separetly in *libraries.R*). Then we show the global parameters (*parameters.R*) and also load the functions (which are shown when needed).

```
source("parameters.R", echo = T)
```

```
##
## > folderData <- file.path(getwd(), "Data")
##
## > folderPlotIndexSent <- file.path(getwd(), "Plot_Index_Sent")
##
## > targetRpa <- 0.06
##
## > targetRpm <- ((1 + targetRpa/100)^(1/12) - 1)
##
## > targetVolpa <- 0.04
##
## > w <- rep(1, 3)
##
## > dateMinEvalLast <- as.Date("2015-06-15")
##
## > dateMaxEvalLast <- as.Date("2016-08-19")
##
## > numAsset <- 7
##
## > sentixDataNames <- c("sentixI1disp", "sentixP1disp",
## +     "sentixG1disp", "sentixI1herf", "sentixG1herf", "sentixP1herf",
## +     "sentixI6disp", " ..." ... [TRUNCATED]
```

```
source("functions.R")
```

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 *dateMin* and *dateMax*, which we use lateron to get the stock data.

```
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
```

```
(dateMin <- min(datesSentix))
```

```
## [1] "2001-02-23"
```

```
(dateMax <- max(datesSentix))
```

```
## [1] "2016-09-02"
```

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

Stocks

We take data mainly from Yahoo Finance. We take closing course from *dateMin* to *dateMax* 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

```
# install.packages("quantmod")
library(quantmod)
# ?getSymbols
```

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

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

```
## '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"
```

```
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 = dateMin, to = dateMax)
```

```
## 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"
```

```
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 = dateMin, to = dateMax)
```

```
## 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"
```

```
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 = dateMin, to = dateMax)
```

```
## [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 = dateMin, to = dateMax)
```

```
## [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 = dateMin, to = dateMax)
```

```
## 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(folderData, "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(folderData, "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
```

```
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
```

```
rm(BUND, DAX, ESX50, NASDAQ, NIKKEI, SP500, TBOND, TEC,
   dax, esx50, nasdaq, nikkei, sp500, tec, i)
```

```
## Warning in rm(BUND, DAX, ESX50, NASDAQ, NIKKEI, SP500, TBOND, TEC, dax, :
## Objekt 'i' nicht gefunden
```

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

```
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).

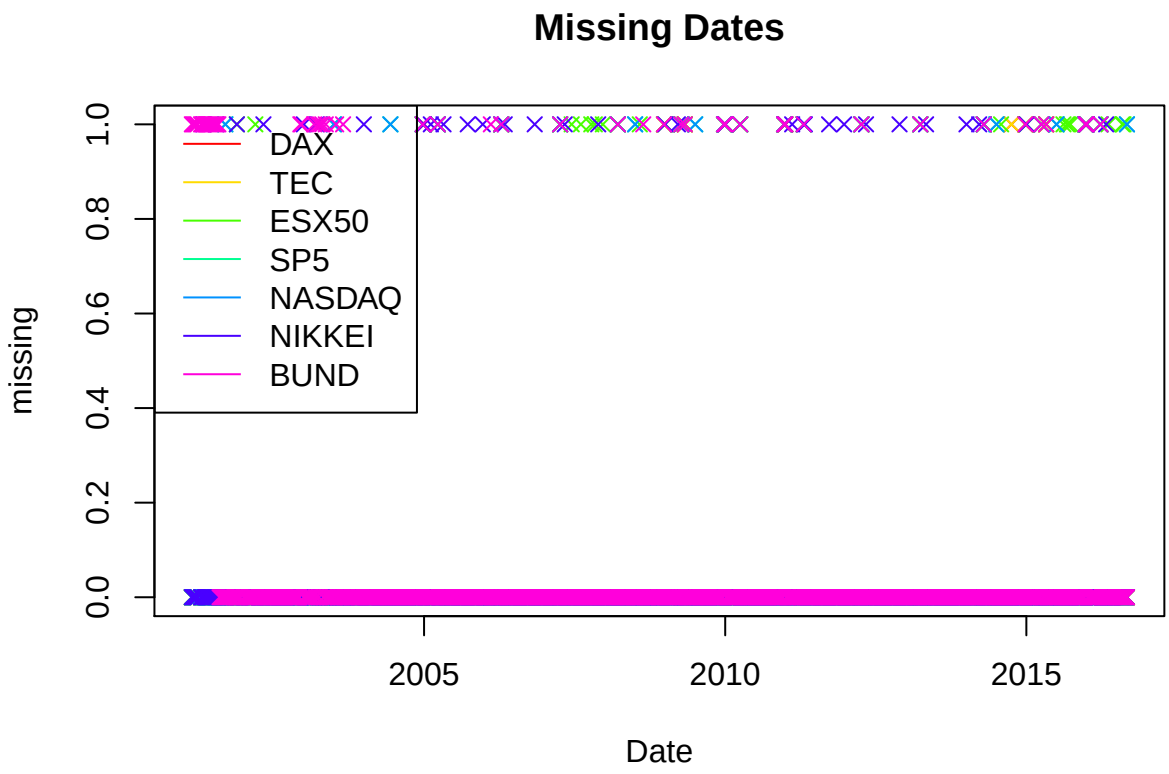
```
colSums(is.na.data.frame(stocks))
```

```
##  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("topleft", legend = colnames(stocks)[2:ncol(stocks)], col = cols, lty = 1)
```




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

Determine, how many dates do have all data available.

```
nrow(stocks)
```

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

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

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

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

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

```
(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.

```
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
```

approach

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.

Data Derivations

We calculate dispersion and herfindah for the sentix data.

Sentix

Dispersion

We measure dispersion of the results of the survey (at each date) as its variance.

Fix one date. Let X_i be the respond of participant i to the future state of the stock with $X_i = 1$ representing, he has positive opinion, $X_i = 0$ neutral, $X_i = -1$ negative.

Then we calculate the dispersion of X as:

$$\text{disp}(X) = \text{Var}(X), \text{where} X = (X_1, ...X_n)$$

In alignment to Dominik's code, we perform the calculation for each index, each group of persons (private, institutional and all), and both time periods (1 month, 6 month).

We produce a list named *sDisp*. Each list element (e.g. P1, P6, I1, ...) contains a data frame with the dispersion for each index (column) at each date (row).

```
sDisp <- list()

colnames(sentixRaw[[1]])
```

```
##  [1] "Datum" "P+"    "Pn"    "P-"    "I+"    "In"    "I-"    "G+"
##  [9] "Gn"    "G-"
```

```
groupP <- c("P+", "Pn", "P-")
groupI <- c("I+", "In", "I-")
groupG <- c("G+", "Gn", "G-")
sDispColumn <- function(dat, group){
  res <- numeric(nrow(dat))
  for(i in 1:length(res)){
    res[i] <- var(c(rep(1, dat[i, group[1]]), rep(0, dat[i, group[2]]), rep(-1, dat[i, group[3]])))
  }
  return(res)
}

names(sentixRaw)
```

```
##  [1] "DAX"     "DAXm"    "TEC"     "TECm"    "ESX50"   "ESX50m"  "SP5"
##  [8] "SP5m"    "NASDAQ"  "NASDAQm" "NIKKEI"  "NIKKEIm" "BUND"    "BUNDm"
```

```
(period1 <- names(sentixRaw)[2*((0:(length(sentixRaw)/2-1)))+1])
```

```
## [1] "DAX"    "TEC"    "ESX50"  "SP5"    "NASDAQ" "NIKKEI" "BUND"
```

```
(period6 <- names(sentixRaw)[2*((0:(length(sentixRaw)/2-1)))+2])
```

```
## [1] "DAXm"    "TECm"    "ESX50m"  "SP5m"    "NASDAQm" "NIKKEIm" "BUNDm"
```

```
sDispDataFrame <- function(period, group){
  res <- data.frame(Datum = datesAll)

  res$DAX <- sDispColumn(sentixRaw[[period[1]]], group)
```

```
  res$TEC <- sDispColumn(sentixRaw[[period[2]]], group)
  res$ESX50 <- sDispColumn(sentixRaw[[period[3]]], group)
  res$SP5 <- sDispColumn(sentixRaw[[period[4]]], group)
  res$NASDAQ <- sDispColumn(sentixRaw[[period[5]]], group)
  res$NIKKEI <- sDispColumn(sentixRaw[[period[6]]], group)
  res$BUND <- sDispColumn(sentixRaw[[period[7]]], group)

  return(res)
}

sDisp[["P1"]] <- sDispDataFrame(period1, groupP)
sDisp[["P6"]] <- sDispDataFrame(period6, groupP)
sDisp[["I1"]] <- sDispDataFrame(period1, groupI)
sDisp[["I6"]] <- sDispDataFrame(period6, groupI)
sDisp[["G1"]] <- sDispDataFrame(period1, groupG)
sDisp[["G6"]] <- sDispDataFrame(period6, groupG)


# we get a problem as the helping formulas are hard coded
if((ncol(sDisp[[1]])-1) != length(period1))
  stop("Fatal error. Check 'sDispDataFrame'. number of Indices changed")

rm(groupP, groupI, groupG, sDispColumn,
   period1, period6, sDispDataFrame)
```

herfindah

We compute a weighted negative Herfindahl Index, which is a measure of dispersion as given in https://www.federalreserve.gov/pubs/feds/2014/201435/201435pap.pdf. Negative value lets higher values indicate greater dispersion.

At each fixed date, the weighted negative Herfindahl Index is computed by:

$$\text{herf}(X) = -\left[\left(\frac{|\{X_i : X_i = 1\}|}{|\{X_1, ..., X_n\}|}\right)^2 + 2\left(\frac{|\{X_i : X_i = 0\}|}{|\{X_1, ..., X_n\}|}\right)^2 + \left(\frac{|\{X_i : X_i = -1\}|}{|\{X_1, ..., X_n\}|}\right)^2\right]$$

Code in analogy to Dominik's.

We produce a list named *sHerf*. Each list element (e.g. P1, P6, I1, ...) contains a data frame with the dispersion for each index (column) at each date (row).

```
sHerf <- list()

colnames(sentixRaw[[1]])
```

```
##  [1] "Datum" "P+"    "Pn"    "P-"    "I+"    "In"    "I-"    "G+"
##  [9] "Gn"    "G-"
```

```
groupP <- c("P+", "Pn", "P-")
groupI <- c("I+", "In", "I-")
groupG <- c("G+", "Gn", "G-")
sHerfColumn <- function(dat, group){
  res <- numeric(nrow(dat))
  for(i in 1:length(res)){
```

```
    s <- sum(dat[i, group])
    res[i] <- -1*( (dat[i, group[1]]/s)^2 + 2*(dat[i, group[2]]/s)^2 + (dat[i, group[3]]/s)^2 )
  }
  return(res)
}

names(sentixRaw)
```

```
##  [1] "DAX"     "DAXm"    "TEC"     "TECm"    "ESX50"   "ESX50m"  "SP5"
##  [8] "SP5m"    "NASDAQ"  "NASDAQm" "NIKKEI"  "NIKKEIm" "BUND"    "BUNDm"
```

```
(period1 <- names(sentixRaw)[2*((0:(length(sentixRaw)/2-1)))+1])
```

```
## [1] "DAX"    "TEC"    "ESX50"  "SP5"    "NASDAQ" "NIKKEI" "BUND"
```

```
(period6 <- names(sentixRaw)[2*((0:(length(sentixRaw)/2-1)))+2])
```

```
## [1] "DAXm"    "TECm"    "ESX50m"  "SP5m"    "NASDAQm" "NIKKEIm" "BUNDm"
```

```
sHerfDataFrame <- function(period, group){
  res <- data.frame(Datum = datesAll)

  res$DAX <- sHerfColumn(sentixRaw[[period[1]]], group)
  res$TEC <- sHerfColumn(sentixRaw[[period[2]]], group)
  res$ESX50 <- sHerfColumn(sentixRaw[[period[3]]], group)
  res$SP5 <- sHerfColumn(sentixRaw[[period[4]]], group)
  res$NASDAQ <- sHerfColumn(sentixRaw[[period[5]]], group)
  res$NIKKEI <- sHerfColumn(sentixRaw[[period[6]]], group)
  res$BUND <- sHerfColumn(sentixRaw[[period[7]]], group)

  return(res)
}

sHerf[["P1"]] <- sHerfDataFrame(period1, groupP)
sHerf[["P6"]] <- sHerfDataFrame(period6, groupP)
sHerf[["I1"]] <- sHerfDataFrame(period1, groupI)
sHerf[["I6"]] <- sHerfDataFrame(period6, groupI)
sHerf[["G1"]] <- sHerfDataFrame(period1, groupG)
sHerf[["G6"]] <- sHerfDataFrame(period6, groupG)


# we get a problem as the helping formulas are hard coded
if((ncol(sHerf[[1]])-1) != length(period1))
  stop("Fatal error. Check 'sHerfDataFrame'. number of Indices changed")

rm(groupP, groupI, groupG, sHerfColumn,
   period1, period6, sHerfDataFrame)
```

TODO further consideration

regress Sentiment

We first regress each sentiment on the other sentiments and just go with the non-explained intercept. From these, we calculate the covariance matrix.

```
i <- sentixDataNames[1]
parse(text = paste0(i, "Reg", " <- ", "regSent(", i, ")"))
for (i in sentixDataNames){
    eval(parse(text = paste0(i, "Reg", " <- ", "regSent(", i, ")")))
}

sentixDataNamesReg <- c()
i = 1
parse(text = paste0("sentixDataNamesReg <- ", "c(sentixDataNamesReg, \"", sentixDataNames[i], "Reg\")"))
for(i in sentixDataNames){
    eval(parse(text = paste0("sentixDataNamesReg <- ", "c(sentixDataNamesReg, \"", i, "Reg\")")))
}
```

```
i <- sentixDataNames[i]
parse(text = paste0(i, "RegCov", " <- ", "cov(", i, "Reg)"))
for(i in sentixDataNames){
    eval(parse(text = paste0(i, "RegCov", " <- ", "cov(", i, "Reg)")))
}
```

returns

Discrete returns. First return ist 0 to start of with (first date).

```
ret <- as.matrix(stocks[2:nrow(stocks),2:ncol(stocks)]/stocks[1:(nrow(stocks)-1),2:ncol(stocks)] - 1)
rownames(ret) <- stocks[2:nrow(stocks), 1]

mu <- colMeans(ret)
S <- cov(ret)
```

find time window

Determine length of time window (l). Calculate return for all stocks (*retWindow*) for all possible time windows (l, l+1, l+2, ..., T). Equal weights for all returns. Calculate (arithmetic) average of all returns at each possible time window (*retTotal*). Choose the one with lowest (*datesEvalBear*) and highest (*datesEvalBull*).

$$\text{retWindow}_{\text{stock}} = \prod_{k=1}^{l} (1 + \text{ret}_{\text{stock}}(k)) - 1$$

Take care as *ret* already contains return from day before to actual day (in each row).

```
l <- 50

retWindow <- matrix(0, nrow = nrow(ret)-l+1, ncol = ncol(ret))
rownames(retWindow) <- rownames(ret)[l:nrow(ret)]
class(rownames(retWindow)) <- "Date"
for(i in 1:nrow(retWindow)){
    retWindow[i,] <- apply(ret[i:(i+l-1),]+1, 2, function(x) prod(x)-1)
}

retTotal <- numeric(nrow(retWindow))
retTotal <- apply(retWindow, 1, mean)
names(retTotal) <- rownames(retWindow)
```

```
iMin <- which(retTotal==min(retTotal))
iMax <- which(retTotal==max(retTotal))

# need l+1 values (start, end (= where max is), l steps in btw)
datesEvalBear <- rownames(ret)[(iMin-1):(iMin+l-1)]
datesEvalBull <- rownames(ret)[(iMax-1):(iMax+l-1)]
class(datesEvalBear) <- "Date"
class(datesEvalBull) <- "Date"
```

additional visualization of the resturns over each time window

```
plot(retTotal, type = "l", axes = FALSE)
abline(v = iMin, col = "red")
abline(v = iMax, col = "green")
axis(1, pretty(1:length(retTotal)), names(retTotal)[pretty(1:length(retTotal))+1])
axis(2)
```

remove variables

```
rm(retWindow, retTotal)
rm(iMin, iMax)
```

regression

regress one on all others

We regress one sentiment variable on all other sentiment variables and take the residuals.

```
regSentResidual
```

```
sentixI1dispResiduals50 <- regSentResidual(sentixI1disp, consider = 50, func = mean)
summary(sentixI1dispResiduals50)

sentixI1dispResiduals10 <- regSentResidual(sentixI1disp, consider = 10, func = mean)
summary(sentixI1dispResiduals10)
```

That is not useful! The values differ after the 16th position after decimal point.

Look at what causes this good explanation of one variable by its others:

```
dat <- sentixI1disp
for(k in colnames(dat)){
    # generate formula (regress one column on all the others while using 'consider' previous points)
    print(form <- as.simple.formula(setdiff(colnames(dat), k), k))
    print(summary(lm(form, data = dat[max((200-50),1):200,])))
}
```

do (correct?) adoptation

get Covariance to 0 by regressing one on all before and so on (compare to Portfolio Analysis Theorem 3.5)