Data - Derivations

Stefan Glogger

August 2017

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-"
```

```r
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"
```

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

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

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

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

```r
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)
```

Stocks

We calculate discrete returns for each date and each stock.

returns

Discrete returns. Be aware that we "loose" the first date now, as we have no idea of the return on day one. Therefore we might also exclude the first date for the other (sentix) variables. We will go on with carefully matching the dates to always consider information ot the actual day.

```
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)
C <- cov(ret)
```

```
# sentixRaw <- lapply(sentixRaw, function(x) {x <- x[2:nrow(x), ]})
# sDisp <- lapply(sDisp, function(x) {x <- x[2:nrow(x), ]})
# sHerf <- lapply(sHerf, function(x) {x <- x[2:nrow(x), ]})
#
# stocks <- stocks[2:nrow(stocks), ]
# datesAll <- datesAll[2:nrow(datesAll)]
```

time window

bull and bear

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

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

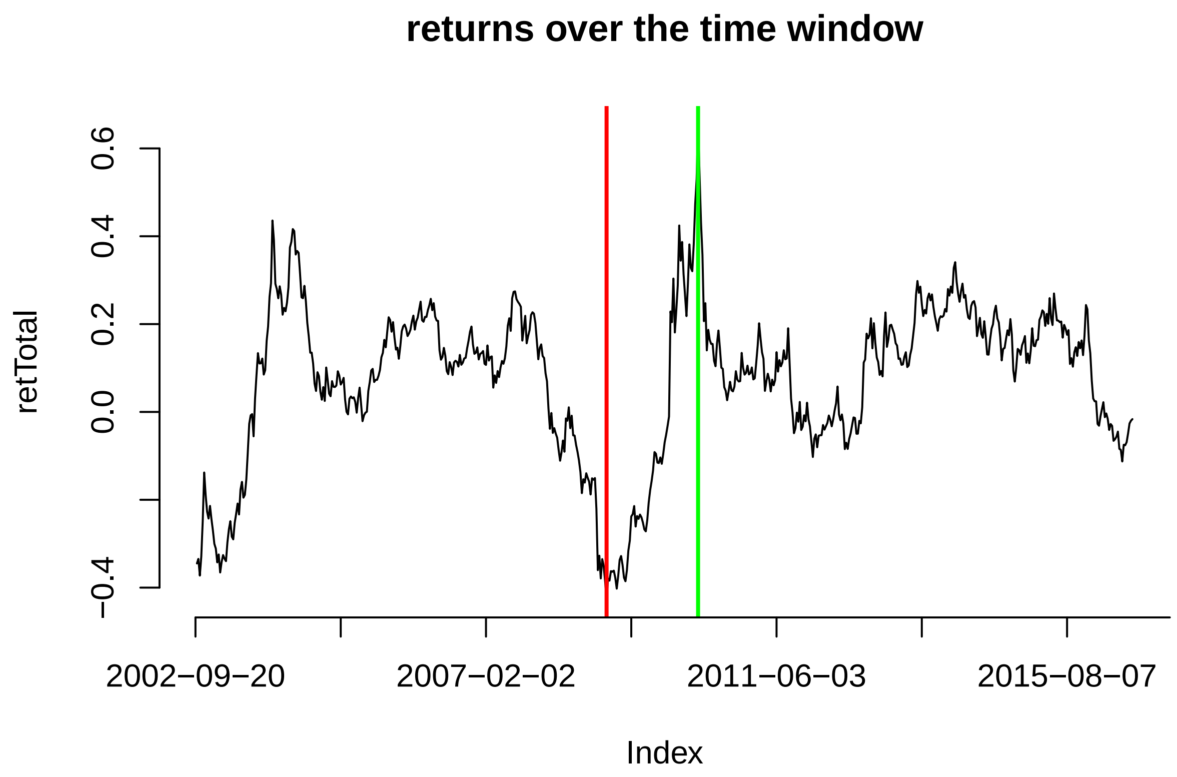
As we calculate with closing prices, we assume that the return is actually of that day (or better spoken of that week). We investment at the very beginning to the opening price, which should be rathly the closing price of the day (week) before).

```
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) # 2 -> columnwise
}

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

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

# dates of which the returns have been calculated
datesEvalBear <- rownames(ret)[(iMin):(iMin+l-1)]
datesEvalBull <- rownames(ret)[(iMax):(iMax+l-1)]
class(datesEvalBear) <- "Date"
class(datesEvalBull) <- "Date"
```

additional visualization of the resturns over each time window

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



last data

We also look at the most actual data.

```
datesEvalLast <- rownames(ret)[(nrow(ret)-l+1):nrow(ret)]
class(datesEvalLast) <- "Date"
```

used later for storing results. trick *deparse(substitute())* to get an error when a window is deleted.

```
datesNames <- c(deparse(substitute(datesEvalBear)), deparse(substitute(datesEvalBull)), deparse(substit
```

remove variables

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

TODO further consideration

For the moment, I (Stefan) don't think that the regressing is thoroughly based, so this (updating of code) is skipped for now.

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)")))
}
```

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:

```r
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)