10 Data

Stefan Glogger

18 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"
```

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

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

```r
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).

```r
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)
```

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)