Thesis data collection script

Jesse Keränen

10/10/2023

This is script used to clean up the data from Datastream. Data was downloaded from Datastream in multiple small excel files. With this script these files are merged and missing observations are cleaned. Finally ready data set is saved so that it can be loaded and used at later point.

Load the needed libraries.

```r
library(readxl)
library(data.table)
library(ggplot2)
library(stringr)
library(openxlsx)
library(tikzDevice)
```

Load the static stock level data for each contry.

```r
static_s <- unique(data.table(read_excel("Data/w02/DS_static_omxs.xlsx")))
static_h <- unique(data.table(read_excel("Data/w02/DS_static_omxh.xlsx")))
static_o <- unique(data.table(read_excel("Data/w02/DS_static_omxo.xlsx")))
static_c <- unique(data.table(read_excel("Data/w02/DS_static_omxc.xlsx")))

# Name of the corresponding country is added
static_s[, Country := "Sweden"]
static_h[, Country := "Finland"]
static_o[, Country := "Norway"]
static_c[, Country := "Denmark"]

static_screens <- function(dt, variable, filter) {
  return(dt[get(variable) %in% filter, ])
}
# Remove stocks for which Country Of Listing is different than the name of the corresponding country.
static_s <- static_screens(static_s, "GEOLN", c("SWEDEN"))
static_h <- static_screens(static_h, "GEOLN", c("FINLAND"))
static_o <- static_screens(static_o, "GEOLN", c("NORWAY"))
static_c <- static_screens(static_c, "GEOLN", c("DENMARK"))

# Remove stocks for which Geographical Classification of Company is different than the name of the corr
static_s <- static_screens(static_s, "GEOGN", c("SWEDEN"))
static_h <- static_screens(static_h, "GEOGN", c("FINLAND"))
static_o <- static_screens(static_o, "GEOGN", c("NORWAY"))
static_c <- static_screens(static_c, "GEOGN", c("DENMARK"))
```

```
# Remove stocks for which Country if ISIN Issuer doesn't match the two letter code for corresponding co
static_s <- static_screens(static_s, "GGISN", c("SE"))
static_h <- static_screens(static_h, "GGISN", c("FI"))
static_o <- static_screens(static_o, "GGISN", c("NO"))
static_c <- static_screens(static_c, "GGISN", c("DK"))

# Remove stocks that are trated in different currency than corresponding country's currency.
static_s <- static_screens(static_s, "PCUR", c("SK"))
static_h <- static_screens(static_h, "PCUR", c("M", "E"))
static_o <- static_screens(static_o, "PCUR", c("NK"))
static_c <- static_screens(static_c, "PCUR", c("DK"))
```

```
omxs_key_words <- paste("(^|\\s+)", c("CONVERTED INTO", "USE", "CONVERTED-", "CONVERTED - SEE"), "($|\\
omxh_key_words <- paste("(^|\\s+)", c("USE"), "($|\\s+)", sep = "", collapse = "|")
omxc_key_words <- paste("(^|\\s+)", c("\\)CSE\\)"), "($|\\s+)", sep = "", collapse = "|")

key_word_removal <- function(dt, variable, key_word) {
  return(dt[!(get(variable) %like% key_word), ])
}
# Remove stocks that have keywords specific for Sweden in their NAME, ENAME or ECNAME
static_s <- key_word_removal(static_s, "NAME", omxs_key_words)
static_s <- key_word_removal(static_s, "ENAME", omxs_key_words)
static_s <- key_word_removal(static_s, "ECNAME", omxs_key_words)

# Remove stocks that have keywords specific for Finland in their NAME, ENAME or ECNAME
static_h <- key_word_removal(static_h, "NAME", omxh_key_words)
static_h <- key_word_removal(static_h, "ENAME", omxh_key_words)
static_h <- key_word_removal(static_h, "ECNAME", omxh_key_words)

# Remove stocks that have keywords specific for Denmark in their NAME, ENAME or ECNAME
static_c <- key_word_removal(static_c, "NAME", omxc_key_words)
static_c <- key_word_removal(static_c, "ENAME", omxc_key_words)
static_c <- key_word_removal(static_c, "ECNAME", omxc_key_words)
```

```
# Append the contry specific data.tables
static <- rbindlist(list(static_s, static_h, static_o, static_c))

# Remove stocks that are not classified as equity, are not primary security within company or are not p
static <- static_screens(static, "TYPE", "EQ")
static <- static_screens(static, "MAJOR", "Y")
static <- static_screens(static, "ISINID", "P")

key_words <-  paste("(^|\\s+)", c("1000DUPL", "DULP", "DUP", "DUPE", "DUPL", "DUPLI", "DUPLICATE", "XSQ

# Rest of the keyword cleaning. Removing e.g. preferred stock, warrants etc..
static <- key_word_removal(static, "NAME", key_words)
static <- key_word_removal(static, "ENAME", key_words)
static <- key_word_removal(static, "ECNAME", key_words)

write.table(paste(static[1:1500, Type], collapse = ", "), file = "Data/ID_1.txt", sep = "\n", row.names
write.table(paste(static[1501:length(static$Type), Type], collapse = ", "), file = "Data/IDs_2.txt", se
```

Function to download monthly data. As parameters function takes path where file to be loaded is located, name od the variable that is being loaded and which county's data is being loaded. Function loads the data, transforms it to tidy format and gets rid of missing values. Function also sets both name of the company and the value of the corresponding variable (e.g. market value) to correct format. Additionally function add new column which indicates in which stock exchange stock is being traded. Finally function returns a data.table.

Derive first variables from monthly data. If company went bankruptcy or is delisted for some other reason, last available return index and market value is repeated until end of the dataset. Therefore we use TIME variable from Datastream (which tells the time of the last available observation) to define when company should be excluded for the dataset.

To avoid forward looking information we only consider accounting figures from year n to be available for the investor in July n + 1. For this we create help column "hcol" that is used to merge annual data to correct period in monthly dataset.

```
monthly_data <- monthly_data[Date <= TIME] # Remove de-listed companies

monthly_data[, year := year(Date)]
monthly_data[, month := month(Date)]
monthly_data[, hcol := ifelse(month >= 7, year-1, year-2)] # Create help column for merging the yearly
```

```
fx_rates <- data.table(read_excel("Data/w02/FX_rates.xlsx", .name_repair = "minimal"))
colnames(fx_rates) <- as.character(fx_rates[1, ])
fx_rates <- fx_rates[-1, ]
fx_rates[, `:=` (Code = NULL, Date = as.POSIXct(Date, tz = "UTC"))]

fx_rates <- melt(fx_rates, id.vars = "Date", value.name = "FX", variable.name = "PCUR")
fx_rates <- fx_rates[, `:=` (PCUR = as.character(PCUR), FX = as.numeric(FX))]

monthly_data <- merge(monthly_data, fx_rates, by = c("Date", "PCUR"))

cols <- c("RI", "MV")
monthly_data[, (paste("USD", cols, sep = ".")) := lapply(.SD, function(x) x / FX), .SDcols = cols]

monthly_data[, `:=` (L.MV = shift(MV), L.USD.MV = shift(USD.MV)), by = Company]

# Calculation of some variables requires MV from Last December
DEC.MV <- monthly_data[month(Date) == 12, .(Company, year=year(Date), DEC.MV = MV, DEC.USD.MV = USD.MV)
DEC.MV[, year := year + 1]
monthly_data <- merge(monthly_data, DEC.MV, by = c("year", "Company"), all.x = T)
```

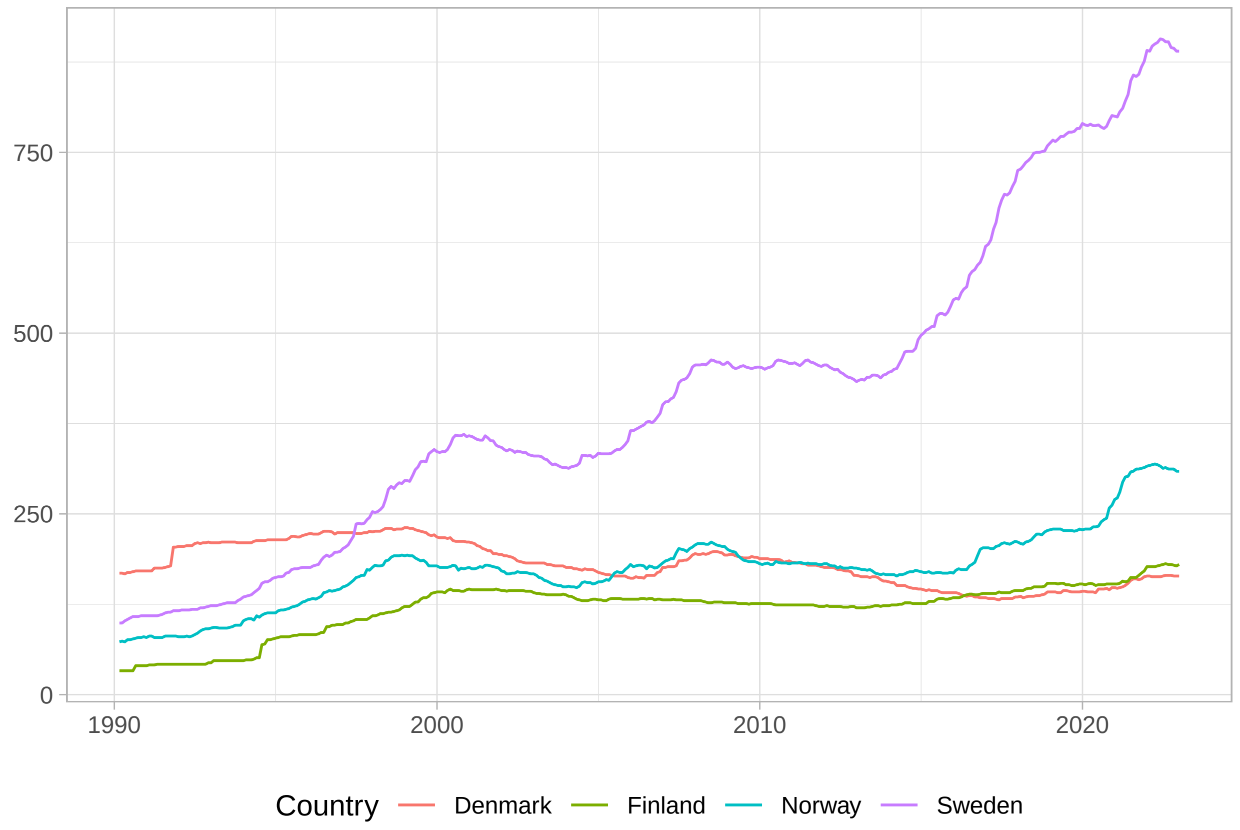
Development of the number of companies in each market.

```
number_of_companies <- monthly_data[, .(Count = uniqueN(Company)), by=c("Date", "Country")]

tikz(file = "number_of_companies.tex", width = 6, height = 4)
plot <- ggplot(number_of_companies, aes(Date, Count, color = Country)) + geom_line() +
    theme_light() + theme(axis.title.y = element_blank(),
                          axis.title.x = element_blank(), legend.position = "bottom")
print(plot)
dev.off()
```

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

```
plot
```



Following two functions are used to prepare the annual data. Functions work quite similar to monthly data collection function. Main difference is that we don't require for each variable a value. E.g. it could be the case that information for deferred taxes (WC03263) is not available. Instead of excluding this company for corresponding periods totally, we set the deferred taxes to 0 for that company for that period. It is defined separately for each variable if such a transformation is reasonable. Variables that are subject to this procedure are stored in the array called "replace_zero".

```
replace_zero <- c("WC03263", "WC03501")

get_annual_data <- function(var) {
  # browser()
  path <- paste("Data/w02/", var, ".xlsx", sep = "")
  dt <- as.data.table(read_excel(path, .name_repair = "minimal"))
  colnames(dt) <- as.character(dt[1, ])
  dt <- tail(dt, -1)
  dt <- melt(dt, id.vars = "Code")
  dt[is.na(value), value := ifelse(var %in% replace_zero, 0, NA)]
  dt <- dt[!is.na(value) & variable != "#ERROR"]
  dt[, `:=` (value=as.numeric(value), variable=gsub(paste("[(]", var,"[)]",
                                              sep = ""),"", variable), Code = as.numeric(Code))]
  colnames(dt) <- c("Date", "Company", var)
  return(dt)
}

variables <- c("WC02999", "WC01001", "WC03501", "WC02001", "WC01501", "WC04860", "WC03263", "WC01551")
```

```r
annual_data <- as.list(lapply(variables, function(x) {
  get_annual_data(x)
}))
annual_data <- Reduce(function (...) { merge(..., all = TRUE) }, annual_data) # Merge the data.tables f

annual_data <- merge(annual_data, static[, .(Type, PCUR)], by.x = "Company", by.y = "Type")
# annual_data <- na.omit(annual_data) # Remove NA
setorder(annual_data, Company, -Date)
```

```r
fx_rates[, `:=` (Year = year(Date), Month = month(Date))]
july_fx_rates <- fx_rates[, .SD[which.max(Date), ], by = c("Year", "Month", "PCUR")][Month == 12, ]

annual_data <- merge(annual_data, july_fx_rates[, -c("Month", "Date")], by.x = c("Date", "PCUR"), by.y =

annual_data[, (paste("USD", variables, sep = ".")) := lapply(.SD, function(x) x / FX), .SDcols = variabl

annual_data[, `:=` (L.WC02999 = shift(WC02999), L.USD.WC02999 = shift(USD.WC02999)), by = Company]
```

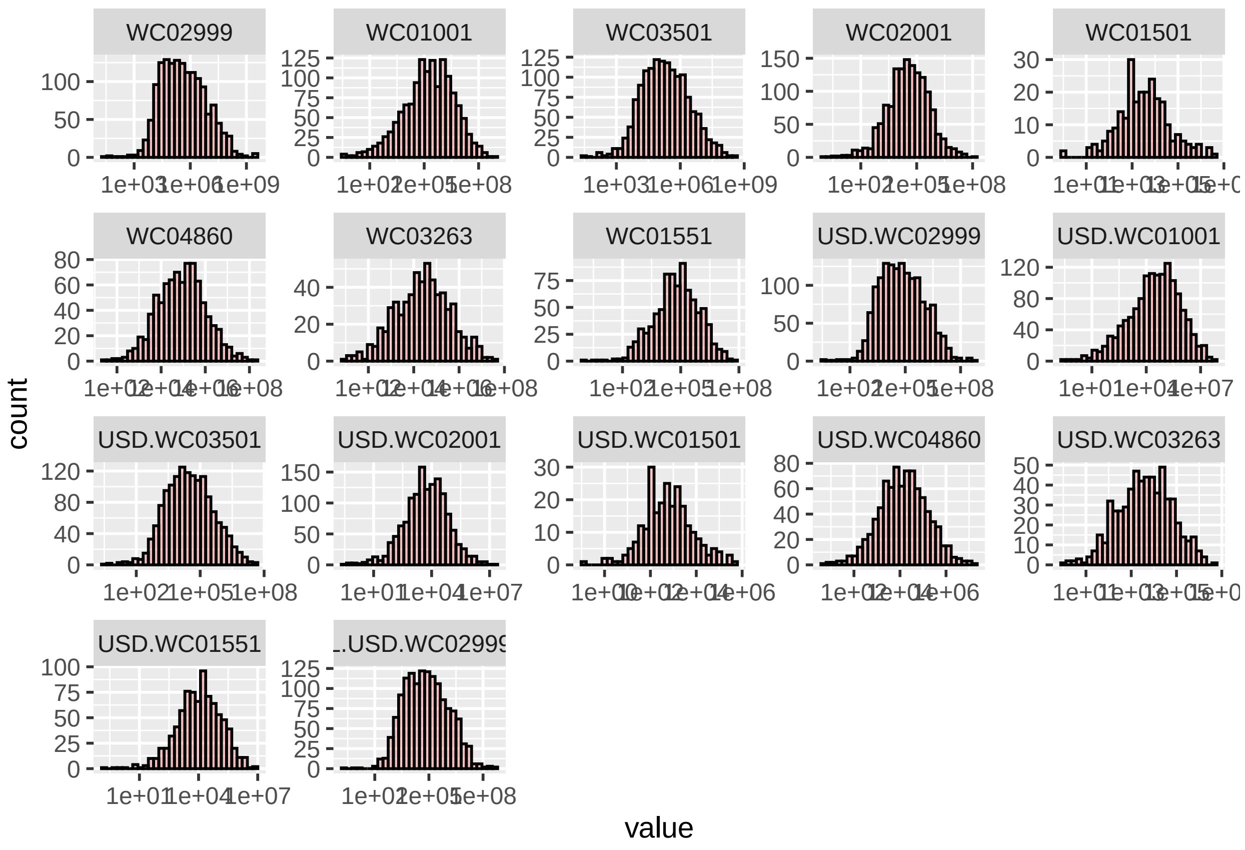
As typical for financial data we have big outliers in our data. Outliers can be either extraordinary chracteristics of a company or they can be mistakes in the data. Either way we don't want our results to be driven by the outliers. Therefore we winsorize all variables. If value of the variable is bigger than 99% of the values in given year then value will be replaced by 99% breakpoint value. Same is done for values smaller than 1%.

```
## Warning in self$trans$transform(x): NaNs produced
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

```
## Warning: Removed 26936 rows containing non-finite values (‘stat_bin()‘).
```

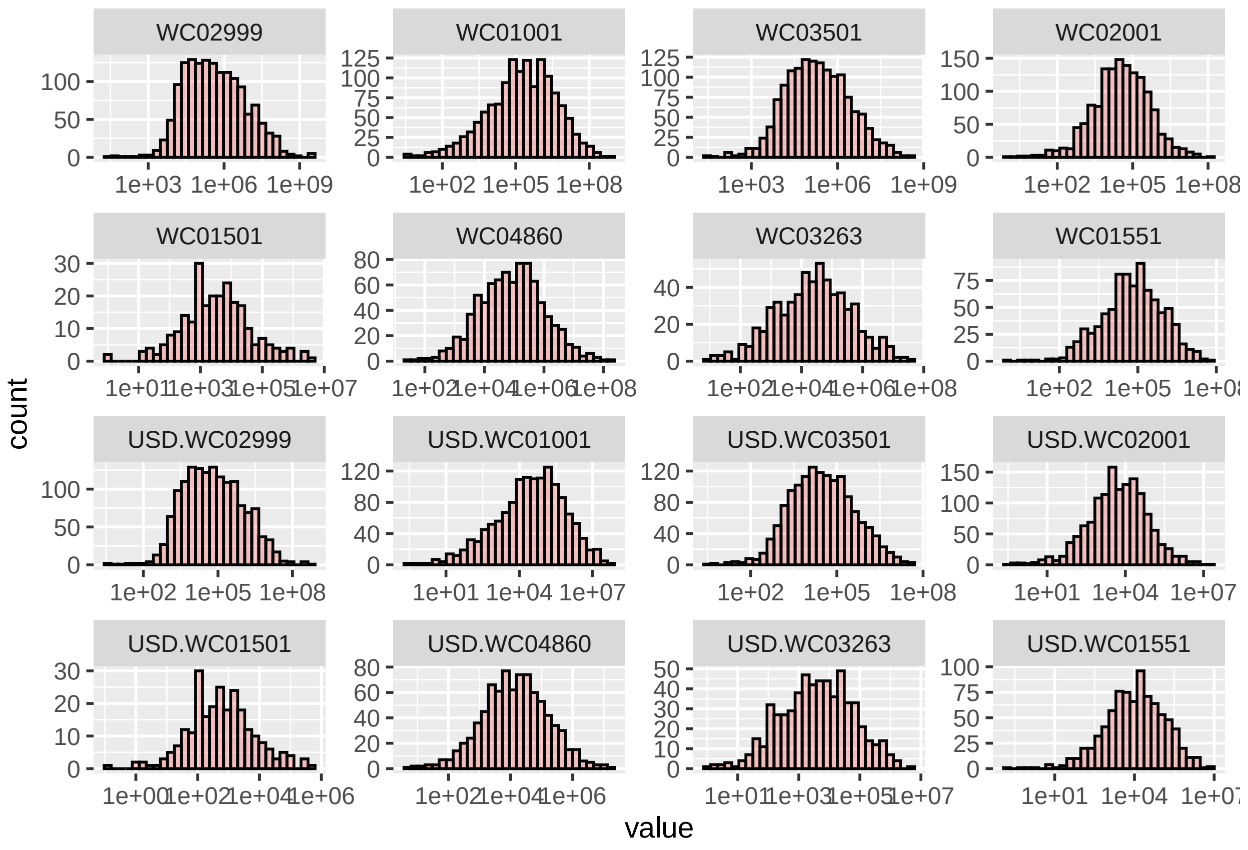


```
## Warning in self$trans$transform(x): NaNs produced

## Warning in self$trans$transform(x): Transformation introduced infinite values
## in continuous x-axis

## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.

## Warning: Removed 25702 rows containing non-finite values (‘stat_bin()‘).
```



Finally ready monthly and annual datasets can be merged. Help column is used from monthly dataset and reason is explained above.

```
data <- merge(monthly_data, annual_data[, -c("PCUR", "FX")], by.x = c("Company", "hcol"), by.y = c("Comp
# data <- na.omit(data)
```

```
data[, RET := (RI / shift(RI)) - 1, by = Company] # Monthly return local currency
data[, USD.RET := (USD.RI / shift(USD.RI)) - 1, by = Company] # Monthly return USD

winsorize(data, "USD.RET", 0.001)
data <- data[USD.RET <= 9.9, ] # Remove returns bigger than 900%
# Abnormal short term reversal removal
data <- data[, .SD[!((USD.RET > 3 | shift(USD.RET) > 3) & (1 + USD.RET) * (1 + shift(USD.RET)) < 0.5), 
```

Instead of using raw accounting data, usually some kind of ratios are used in anomaly literature. In the next chunck stock characteristics used in this study are calculated.

```
data[, CashToAsset := WC02001 / WC02999, by = Company]

data[, CapitalTurnOver := WC01001 / shift(WC02999), by = Company]

data[, BE := WC03501 + WC03263, by = Company]
data[, BEME := BE / (DEC.MV * 1000), by = Company]

data[, BEME.M := BE / (L.MV * 1000), by = Company]
```

```
data[, INV := (WC02999 - shift(WC02999, 12)) / shift(WC02999, 12), by = Company]

data[, CFP := WC04860 / (DEC.MV * 1000), by = Company]

data[, Leverage := (WC02999 - WC03501) / (DEC.MV * 1000), by = Company]

data[, SP := WC01001 / (DEC.MV * 1000), by = Company]

data[, EP := WC01551 / (DEC.MV * 1000), by = Company]

data[, ROA := WC01551 / shift(WC02999, 12), by = Company]

data[, ROE := WC01551 / shift(BE), by = Company]

data[, MOM7 := shift(USD.RI, 7, type = "lag")/shift(USD.RI, 12, type = "lag") - 1,
   by = Company]

data[, MOM12 := shift(USD.RI, 2, type = "lag")/shift(USD.RI, 12, type = "lag") - 1,
   by = Company]

data[, MOM24 := shift(USD.RI, 2, type = "lag")/shift(USD.RI, 24, type = "lag") - 1,
   by = Company]

data[, MOM2 := shift(MOM12)]

var <- c("RET", "BEME", "INV", "EP", "CashToAsset", "CapitalTurnOver", "CFP", "Leverage", "SP", "ROA",

invisible(lapply(var, function(x) {
  # winsorize(data, x, 0.01)
}))

data[, (var) := lapply(.SD, function(x) {
  replace(x, is.infinite(x), NA)
}), .SDcols = var]
```

Idea of this study is to mimic actual investment decision setting. Therefore we use lagged variables in the regressions because this information would have been available for an investor in a given time. To make running regressions simpler we add for each variable used in the regression lagged version (monthly variables).

```
data[, `:=` (L.CashToAsset = shift(CashToAsset), L.CapitalTurnOver = shift(CapitalTurnOver), L.INV = shi

data <- data[is.finite(rowSums(data[, ..var]))] # Remove infinite values
```

```
# Total number of unique companies
data[year >= 1997, uniqueN(Company), by = Country]
```

```
##    Country  V1
## 1: Finland 214
## 2:  Norway 376
## 3:  Sweden 880
## 4: Denmark 260
```

```r
data[year >= 1997, uniqueN(Company)]
```

```
## [1] 1730
```

```r
company_count <- data[year >= 1997, .(number_of_companies = uniqueN(Company)), by = c("Country", "Date")
company_count <- company_count[, .(MIN = min(number_of_companies), MAX = max(number_of_companies),
                 MEAN = mean(number_of_companies)), by = Country]

company_count_tot <- data[year >= 1997, .(Country = "Nordic", number_of_companies = uniqueN(Company)), 
company_count_tot <- company_count_tot[, .(Country = "Nordic", MIN = min(number_of_companies),
                                          MAX = max(number_of_companies), MEAN = mean(number_of_compan
company_count <- rbind(company_count, company_count_tot)


market_value <- data[, .(MEAN.MV = mean(L.USD.MV, na.rm = T), MEDIAN.MV = median(L.USD.MV, na.rm = T),
                         SUM.MV = sum(L.USD.MV)), by = c("Country", "Date")]
market_value <- market_value[, .(MEAN.MV = mean(MEAN.MV), MEDIAN.MV = mean(MEDIAN.MV),
                                 SUM.MV = mean(SUM.MV)), by = Country]

market_value_tot <-  data[, .(Country = "Nordic", MEAN.MV = mean(L.USD.MV, na.rm = T),
                              MEDIAN.MV = median(L.USD.MV, na.rm = T), SUM.MV = sum(L.USD.MV)), by = c(
market_value_tot <- market_value_tot[, .(Country = "Nordic", MEAN.MV = mean(MEAN.MV), MEDIAN.MV = mean(
                                        SUM.MV = mean(SUM.MV))]
market_value <- rbind(market_value, market_value_tot)

country_chacteristics <- merge(company_count, market_value, by = "Country")
setorder(country_chacteristics, Country)
```

```r
# describe(na.omit(data[, ..var]))
means <- data[, (lapply(.SD, mean, na.rm = T)),.SDcols = var, by = Date]
means <- melt(means, id.vars = "Date", value.name = "MEAN.Nordic")

sds <- data[, (lapply(.SD, sd, na.rm = T)),.SDcols = var, by = Date]
sds <- melt(sds, id.vars = "Date", value.name = "SD.Nordic")

temp <- merge(means, sds, by = c("Date", "variable"))

means <- data[, (lapply(.SD, mean, na.rm = T)),.SDcols = var, by = c("Date", "Country")]
means[, Country := paste("MEAN.", Country, sep = "")]
means <- melt(means, id.vars = c("Date", "Country"), value.name = "Mean")
means <- dcast(means, Date + variable ~ Country, value.var = "Mean")

sds <- data[, (lapply(.SD, sd, na.rm = T)),.SDcols = var, by = c("Date", "Country")]
sds[, Country := paste("SD.", Country, sep = "")]
sds <- melt(sds, id.vars = c("Date", "Country"), value.name = "SD")
sds <- dcast(sds, Date + variable ~ Country, value.var = "SD")

temp2 <- merge(means, sds, by = c("Date", "variable"))
temp3 <- merge(temp, temp2, by = c("Date", "variable"))

sddd <- temp3[, lapply(.SD, mean, na.rm = T), .SDcols = colnames(temp3[, -c("Date", "variable")]), by =
```

Final data.table is saved to the data folder for later use.

```r
save(data, file = "Data/data.Rdata")
```