Case study: Stock Market

Overview

In this case study, you will jointly analyse historic data of three major stock indices, the **Dow Jones**, the **DAX**, and the **Nikkei**, and the exchange rates between the **US dollar**, the **Euro**, and the **Yen**. Using this data, you will address several questions. How large was the impact of the recent financial crisis on the respective stock markets? How correlated is the development between the stock markets? What is the relationship between stock market returns and exchange rates? To address these questions, you will import several data files, while tuning import parameters to match the idiosyncracies of the data. You will merge the data files into a single data frame, and mutate the data to reflect changes in index price and exchange rate. You will analyze correlations of stock indices among themselves and to exchange rates. You create illustrative plots for each of the analyses. Below you will find several tasks that will guide you through these steps. For the most part these tasks require you to make use of what you have learned in the sessions **Data I/O**, **Data Wrangling**, and **Statistics**. However, they will also go beyond what you have learned, in particular when it comes to plotting. In those cases, the tasks will provide the necessary code and guidance.

Table 1: Variables of data sets "^DJI.csv", "^GDAXI.csv", "^N225.csv"

Variable	Description
Date	Current day
Open	Day's price when the stock exchange opened
High	Day's highest price
Low	Day's lowest price
Close	Day's closing price
Adj Close	Adjusted closing price that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open

Table 2: Variables of data sets "euro-dollar.txt", "yen-dollar.txt"

Variable	Description
Date (currently unnamed)	Current day
Rate (currently unnamed)	Day's exchange rate in terms of 1 US Dollar. E.g., a value of .75 means that the respective currenty is worth a fraction of .75 of 1 US Dollar

Data I/O

1. Open a new R script and save it as a new file called `finance_case_study.R`. At the top of the script, using comments, write your name and the date. Then, load the `tidyverse`package. Here's how the top of your script should look:

```
## NAME
## DATE
## Sales Data - Case Study

library(tidyverse)
```

2. Load in the stock index data sets, "^DJI.csv", "^GDAXI.csv", and "^N225.csv", from the data folder, using the `read_csv()`-function. In so doing, set an explicit `na`-argument to account for the fact that "^GDAXI.csv" "^N225.csv" use a specific character string to represent missings in the data. To identify the NA-character string in the data open one of them (in a standard text viewer, e.g., *textedit*).

```
# Load index data from local data folder
dow <- read_csv(file = '^DJI.csv')
dax <- read_csv(file = '^GDAXI.csv', na = 'null')
nik <- read_csv(file = '^N225.csv', na = 'null')
```

3. Load in the exchange rate data sets, "euro-dollar.txt" and "yen-dollar.txt", from the data folder, using the `read_delim()`-function and `\t` as the `delim`-argument (separates by *tabulator*). Observe the inferred data types and the variables names. There's something wrong. First, we want the columns to be called `Date` and `Rate` (and not be taken from the first row). To achieve this, include the `col_names`-argument and assign to it a vector that contains the variable names. Second, we want the `Date`-variable to be of type `date`. To achieve this, use the `parse_date` function with `format = '%d %b %Y'`, which specifies the exact format the dates are formatted in, and overwrite the existing `Date`-variable.

```
# Load exchange rate data from local data folder
eur_usd <- read_delim(file = 'euro-dollar.txt', delim = '\t', col_names = c('Date', 'Rate'))
yen_usd <- read_delim(file = 'yen-dollar.txt', delim = '\t', col_names = c('Date', 'Rate'))

# change Date to date type
eur_usd$Date <- parse_date(eur_usd$Date, format = '%d %b %Y')
yen_usd$Date <- parse_date(yen_usd$Date, format = '%d %b %Y')
```

4. Get a first impression of the datasets using `print()`, `typeof()` and `str()`. What type are the data objects, what variables do they contain?

```
# Inspect data
typeof(dow) ; str(dow)
```

```
## [1] "list"
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    8364 obs. of  7 variables:
##  $ Date     : Date, format: "1985-01-29" "1985-01-30" ...
##  $ Open     : num  1278 1297 1283 1277 1272 ...
##  $ High     : num  1295 1305 1293 1286 1295 ...
##  $ Low      : num  1267 1279 1273 1270 1269 ...
##  $ Close    : num  1293 1288 1287 1278 1290 ...
##  $ Adj Close: num  1293 1288 1287 1278 1290 ...
##  $ Volume   : int  13560000 16820000 14070000 10980000 11630000 13800000 14610000 11440000 8000000 1
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 7
##   .. ..$ Date     :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
##   .. ..$ Open     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ High     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Low      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Close    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Adj Close: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
```

```
##   .. ..$ Volume   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
typeof(dax) ; str(dax)
```

```
## [1] "list"
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    7811 obs. of  7 variables:
##  $ Date     : Date, format: "1987-12-30" "1987-12-31" ...
##  $ Open     : num  1005 NA NA 956 996 ...
##  $ High     : num  1005 NA NA 956 996 ...
##  $ Low      : num  1005 NA NA 956 996 ...
##  $ Close    : num  1005 NA NA 956 996 ...
##  $ Adj Close: num  1005 NA NA 956 996 ...
##  $ Volume   : int  0 NA NA 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 7
##   .. ..$ Date     :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
##   .. ..$ Open     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ High     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Low      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Close    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Adj Close: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Volume   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
typeof(nik) ; str(nik)
```

```
## [1] "list"
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    13722 obs. of  7 variables:
##  $ Date     : Date, format: "1965-01-05" "1965-01-06" ...
##  $ Open     : num  1258 1264 1274 1286 NA ...
##  $ High     : num  1258 1264 1274 1286 NA ...
##  $ Low      : num  1258 1264 1274 1286 NA ...
##  $ Close    : num  1258 1264 1274 1286 NA ...
##  $ Adj Close: num  1258 1264 1274 1286 NA ...
##  $ Volume   : int  0 0 0 0 NA 0 0 0 NA 0 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 7
##   .. ..$ Date     :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
```

```
##   .. ..$ Open     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ High     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Low      : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Close    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Adj Close: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Volume   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
typeof(eur_usd) ; str(eur_usd)
```

```
## [1] "list"
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    6546 obs. of  2 variables:
##  $ Date: Date, format: "1999-01-04" "1999-01-05" ...
##  $ Rate: num  1.19 1.18 1.16 1.17 1.16 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 2
##   .. ..$ Date: list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ Rate: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
typeof(yen_usd) ; str(yen_usd)
```

```
## [1] "list"
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    8765 obs. of  2 variables:
##  $ Date: Date, format: "1990-01-02" "1990-01-03" ...
##  $ Rate: num  0.00684 0.00686 0.00698 0.00695 0.00694 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 2
##   .. ..$ Date: list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ Rate: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

Data Wrangling

5. Create a single data frame from all five datasets that includes only the variables containing the dates, the stock index (unadjusted) closing prices, and the exchange rates. Do this by *piping* (i.e., using `%>%`) together several `inner_join()`-functions, joining the data sets one-by-one using the `Date` variable. At the end, rename the variables appropriately using the `rename()`-function. Note, in joining two

data sets you can control the naming of overlapping variable names using the `suffix`-argument of the `inner_join()`-function. Observe that `inner_join()` takes care of the fact that different dates are available in each of the data sets by only retaining dates for which all data sets provide data on.

```
# create single data frame
data = dow %>% select('Date','Close') %>%
  inner_join(dax %>% select('Date','Close'), by = 'Date', suffix = c('_dow','_dax')) %>%
  inner_join(nik %>% select('Date','Close'), by = 'Date') %>%
  inner_join(eur_usd, by = 'Date') %>%
  inner_join(yen_usd, by = 'Date', suffix = c('_eur','_yen')) %>%
  rename('Close_nik' = 'Close')
```

6. Create new variables containing the change in index price and exchange rate for each variable using the `mutate`-function(). To compute the change, use the `diff()`-function. Since `diff()` will return `n - 1` change values, add an `NA` at the first position of the change variable à la `c(NA, diff(my_variable))`.

```
# create variables representing day-to-day changes
data = data %>% mutate(
  Close_dow_change = c(NA, diff(Close_dow)),
  Close_dax_change = c(NA, diff(Close_dax)),
  Close_nik_change = c(NA, diff(Close_nik)),
  Rate_eur_change = c(NA, diff(Rate_eur)),
  Rate_yen_change = c(NA, diff(Rate_yen))
)
```

7. Create a variable containing merely the year of the date variable using `mutate()` and `year()` from the `lubridate`-package (should have been installed with the `tidyverse`). First, load the `lubridate`-package.

```
# load lubridate
library(lubridate)

# create year variable
data = data %>% mutate(year = year(Date))
```

8. Create a long version of data frame, in which variables occupy different rows rather than columns, using the `gather()`-function from the `tidyr`-package (also part of the `tidyverse`). To do this use the command below. You may have to (or want to) change the object/variable names. The first two arguments (given we used *pipes*) to the `gather`-function specify the names of the new variables, the third and fourth specify the names of the variable containing the dates and years with a leading hyphen. Check out the example in the `?gather`-help file. When done, inspect the object and compare it (visually) to the original, wide version.

```
# create long version of data frame
long_data = data %>% gather(variable, value, -Date, -year)
```

Statistics (& Plotting)

8. Plot the price development of the three stock indices using the following command (using the long data object). The code uses a modern and very powerful plotting package called `ggplot2`, which you will be introduced to on the second weekend of the course. For the code to work, you may have to adjust the object and variable names to match the ones in your data frame. Inspecting the plot, do you see a significant drop anywhere?

```
# create long version of data frame
temp_data <- long_data %>% filter(variable %in% c("Close_dow","Close_dax","Close_nik"))
```

```
ggplot(data = temp_data, mapping = aes(x = Date, y = value)) +
  geom_line() +
  facet_grid(~variable)
```

9. Calculate the overall stock index price change per year. Use `group_by()`, `summarize()`, and `sum()` on the stock index change variables. Remember there were `NA`'s in two of the stock index price variables. When was the biggest drop in stock index prices?

```
# calculate aggregate change per year
aggregate_change <- data %>%
  group_by(year) %>%
  summarize(
    mean_dow_change = sum(Close_dow_change),
    mean_dax_change = sum(Close_dax_change, na.rm = T),
    mean_nik_change = sum(Close_nik_change, na.rm = T)
)
aggregate_change
```

```
## # A tibble: 20 x 4
##     year mean_dow_change mean_dax_change mean_nik_change
##    <dbl>           <dbl>           <dbl>           <dbl>
##  1 1999.            NA             1529.           5903.
##  2 2000.          -709.            -455.          -5561.
##  3 2001.          -766.           -1254.          -4436.
##  4 2002.         -1680.           -2112.          -2235.
##  5 2003.          2112.             718.            327.
##  6 2004.           329.             180.           1335.
##  7 2005.           -65.5           1108.           3219.
##  8 2006.          1746.            1189.           1504.
##  9 2007.           903.            1470.          -1918.
## 10 2008.         -4697.           -3257.          -6448.
## 11 2009.          1880.            1147.           1513.
## 12 2010.          1021.             957.           -318.
## 13 2011.           648.           -1016.          -1774.
## 14 2012.           721.            1714.           1940.
## 15 2013.          3566.            1940.           5896.
## 16 2014.          1479.             253.           1159.
## 17 2015.          -379.             937.           1583.
## 18 2016.          2159.             738.             80.7
## 19 2017.          4957.            1266.           3439.
## 20 2018.          -455.            -960.          -2395.
```

10. Now that you know when the biggest drop occured, do you not wonder which stock index suffered the biggest loss? Compare the overall losses to the index price on the first trading day of that year. To do this, first identify the first date available for that year and then `filter()` the data based on the respective character string to retrieve that day's closing prices. Then divide the overall stock price change by that years first closing price. Which stock index suffered the greatest relative loss?

```
aggregate_change %>%
  filter(year == 2008) %>%
  select(-1) / data %>%
  filter(Date == "2008-01-04") %>%
  select(Close_dow, Close_dax, Close_nik)
```

```
##   mean_dow_change mean_dax_change mean_nik_change
## 1      -0.3669855      -0.4171147      -0.4389109
```

11. One driver behind the results observed for the last two tasks is that modern financial markets are closely intertwined, to the extend that a change in one market can bring about a change in the markets. Evaluate this aspect of financial markets by correlating the stock index change variables among each other using `cor()`. Note that `cor()` can take a data frame as an argument to produce the full correlation matrix among all variables. Also, don't forget about the `NA`s - there is an argument for `cor()` to deal with them. How closely are the stock indices related and which ones are most closely related?

```
data %>%
  select(Close_dow_change, Close_dax_change, Close_nik_change) %>%
  cor(., use = 'complete.obs')
```

```
##                   Close_dow_change Close_dax_change Close_nik_change
## Close_dow_change        1.0000000        0.5702655        0.1685651
## Close_dax_change        0.5702655        1.0000000        0.3178292
## Close_nik_change        0.1685651        0.3178292        1.0000000
```

12. Evaluate the stability of the relationships between financial markets by analyzing the correlations for each year. Note that here have to specify each pairwise correlation separately in order to `summarize()` the correlation for each year (rather than computing the entire correlation matrix).

```
data %>%
  group_by(year) %>%
  summarize(
    cor_dow_dax = cor(Close_dow_change, Close_dax_change, use = 'complete.obs'),
    cor_dow_nik = cor(Close_dow_change, Close_nik_change, use = 'complete.obs'),
    cor_dax_nik = cor(Close_dax_change, Close_nik_change, use = 'complete.obs')
)
```

```
## # A tibble: 20 x 4
##     year cor_dow_dax cor_dow_nik cor_dax_nik
##    <dbl>       <dbl>       <dbl>       <dbl>
##  1 1999.       0.453      0.0730       0.192
##  2 2000.       0.308     -0.0519       0.195
##  3 2001.       0.667      0.255        0.287
##  4 2002.       0.654      0.196        0.236
##  5 2003.       0.695      0.127        0.249
##  6 2004.       0.468      0.166        0.344
##  7 2005.       0.390      0.0625       0.327
##  8 2006.       0.594      0.100        0.266
##  9 2007.       0.537      0.0874       0.394
## 10 2008.       0.613      0.212        0.527
## 11 2009.       0.732      0.175        0.260
## 12 2010.       0.695      0.255        0.329
## 13 2011.       0.796      0.185        0.351
## 14 2012.       0.725      0.212        0.325
## 15 2013.       0.566      0.161        0.256
## 16 2014.       0.567      0.149        0.176
## 17 2015.       0.538      0.254        0.295
## 18 2016.       0.605      0.214        0.387
## 19 2017.       0.554      0.367        0.395
## 20 2018.       0.378      0.108        0.481
```

13. Another important index of the financial markets is the exchange rate to other currencies. Generally, it is assumed that a strong economcy translates in both a strong stock index and a strong currency relative to other currencies. First, evaluate whether changes in exchange rates correlate with each other the same way that stock indices do.

```
data %>%
  select(Rate_eur_change, Rate_yen_change) %>%
  cor(., use = 'complete.obs')
```

```
##                  Rate_eur_change Rate_yen_change
## Rate_eur_change        1.0000000       0.3913418
## Rate_yen_change        0.3913418       1.0000000
```

14. Now evaluate whether exchange rates vary as a function of the difference between stock indices. I.e., does, for instance, a large difference between Dow Jones and DAX translate into a strong Dollar relative to the EURO? Based on the above intuition this should be the case. However, economic theory has also produced an alternative hypothesis. That is, foreign investors who benefit from a rise in stock index price may be motivated to sell their holdings and exhange them for their own currency to mantain a neutral position. This would, in effect, depreciate the currency at the same time as the stock index is outperforming, thus producing a negative relationship. What do you think? Ask the data.

```
data %>%
  summarize(
    cor_dow_dax = cor(Close_dow - Close_dax, Rate_eur, use = 'complete.obs'),
    cor_dow_nik = cor(Close_dow - Close_nik, Rate_yen, use = 'complete.obs')
  )
```

```
## # A tibble: 1 x 2
##   cor_dow_dax cor_dow_nik
##         <dbl>       <dbl>
## 1      0.0869       0.569
```

15. Now, evaluate the stability of the above relationship for each year separately? Can you make out a stable pattern? No? I guess it depends.

```
data %>%
  group_by(year) %>%
  summarize(
    cor_dow_dax = cor(Close_dow - Close_dax, Rate_eur, use = 'complete.obs'),
    cor_dow_nik = cor(Close_dow - Close_nik, Rate_yen, use = 'complete.obs')
  )
```

```
## # A tibble: 20 x 3
##     year cor_dow_dax cor_dow_nik
##    <dbl>       <dbl>       <dbl>
##  1 1999.      -0.639     -0.501
##  2 2000.      -0.222     -0.479
##  3 2001.      -0.393     -0.221
##  4 2002.       0.162     -0.287
##  5 2003.       0.772     -0.320
##  6 2004.       0.253      0.427
##  7 2005.       0.831      0.862
##  8 2006.       0.776      0.0745
##  9 2007.      -0.136      0.824
## 10 2008.       0.825      0.788
## 11 2009.       0.837      0.740
## 12 2010.       0.714      0.932
## 13 2011.      -0.334      0.459
## 14 2012.      -0.248      0.727
## 15 2013.      -0.235      0.898
## 16 2014.      -0.856      0.781
```

```
## 17 2015.       0.601      0.819
## 18 2016.      -0.108      0.831
## 19 2017.       0.784      0.629
## 20 2018.       0.470      0.792
```

References

https://www.weforum.org/agenda/2015/07/whats-the-relationship-between-stock-returns-and-exchange-rates/
http://www.faz.net/aktuell/finanzen/aktien/eine-geschichte-des-dax-die-wichtigsten-ereignisse-12970011.html