

Dow Jones monthly performance

Team B


2/14/2022

For these exercises, we need to load the following packages:

```
library(readxl)
library(tidyverse)
library(lubridate)
```

1. Import the Excel spreadsheet as a tibble named dj, which stands for ‘Dow Jones’. Make sure you correctly cut off non-data rows at the top and bottom.

Let us first import the data, with the `read_excel` function from the `readxl` library.



```
# Read excel sheet and skip metadata at start
dj <- as_tibble(read_excel("dja-performance-report-monthly.xls", skip = 5))


# Remove the last 5 rows, which are unwanted
dj <- dj[1:(nrow(dj) - 5), ]
```

2. Remove all columns except ‘Effective Date’ and ‘Close Value’.

```
# Subset to only retain the relevant columns
dj <- dj[c("Effective Date", "Close Value")]
```

3. Column names with spaces are awkward to work with in R. Change the names of the columns so that they are short and do not contain spaces: date and close_value.

Let us rename the column names with ones that are easier to work with.



```
# Change the column name from x to y,
# at whichever index the column name was previously x
# names(data)[name(data) == x] <- y


names(dj)[names(dj) == "Effective Date"] <- "date"
names(dj)[names(dj) == "Close Value"] <- "close_value"

# Alternatively, we could simply do this
colnames(dj) <- c("date", "close_value")
```

4. What are the R classes of the columns?

Let’s find out what class the columns belong to. We can apply the `class` function to every column in `dj` with the help of `lapply`.

```
lapply(dj, class)
```



```
## $date
## [1] "character"
##
## $close_value
## [1] "numeric"
```

5. Append a column month with the month that is implicit in the date column.

We can use the `mdy` function, and pipe its output to the `month` function. Both of these are included in the `lubridate` library.

```
dj$month <-
  mdy(dj$date) |>
  month()
```

6. Answer the question: are all months in `dj` in consecutive calendrical order? The result should be a logical value (i.e. `TRUE` or `FALSE`). Do not use a `for`-loop. You may find the functions `lead()` and `lag()` in `dplyr` useful for this task. Be careful! Suppose we had the dates “6/30/1867” and “7/31/1869” in consecutive rows of `dj`. The months appear to be consecutive (6 and 7), but they are in different years; thus, they are not consecutive calendrical months. Ensure that your code handles such cases correctly.

Let us first append a year column, so that we can make sure that the years are consecutive too. We can take inspiration from the code for the previous answer.

```
dj$year <-
  mdy(dj$date) |>
  year()
```

There are multiple ways we can approach this problem. Let us look at one approach where we `lag` the months and years by 1; that is, we create two new columns, `lagged_year` and `lagged_month` which contain the `year` and `month` of the previous row respectively. By default, `lag` *lags* (or shifts) the elements by 1 row forward. We could’ve also used `lead` to shift the elements one row behind.

```
dj$lagged_year <- lag(dj$year)
dj$lagged_month <- lag(dj$month)
```

Unless the lagged row is the first month of the year, the difference between `lagged_year` and `year` should always be 0, and the difference between the `lagged_month` and `month` should always be one. When `lagged_month` is 12 and `month` is 1, `year - lagged_year` should be 1. Thus, we can simply add 12 to the difference whenever this is the case. Let us append a new column that calculates this difference.

```
dj$difference <-
  ((dj$year - dj$lagged_year) * 12) +
  (dj$month - dj$lagged_month)
```

Thus, if the value of difference is not equal to 1 in any row, the months are not consecutive anymore. We can simply use the `all()` function to check that all values are equal to 1. We should also not include NA values by using the `na.rm` argument, to skip over the first row which is NA for both lagged columns.

```
is_consecutive <- all(dj$difference == 1, na.rm = TRUE)
is_consecutive
```

```
## [1] FALSE
```

7. In the previous problem, you should have found that not all months are in consecutive calendrical order. Use R to find the date just before the gap(s). Perform some research about the reason for the gap(s). Summarise your findings in maximally 10 sentences.

Uh oh! The months are not consecutive. We can find out where this is the case by using `filter` from the `dplyr` library (which `tidyverse` conveniently includes for us). `filter` also drops rows where the condition evaluates to `NA`.

```
filter(dj, difference != 1)
```

```
## # A tibble: 1 x 7
##   date      close_value month  year lagged_year lagged_month difference
##   <chr>          <dbl> <dbl> <dbl>    <dbl>    <dbl>    <dbl>
## 1 12/31/1914      54.6   12  1914      1914         7         5
```

Since `difference` is 5, `lagged_month` is 7, and `lagged_year` is 1914, we know that the only data before this was from July of the same year. Thus, August, September, and November were skipped in 1914. Let's find out why.

July 1914 marked the start of WWI. The breakout of war probably generated strong pessimism in economic growth and hence the stock market, resulting in the stock exchange closing down for a few months before it reopened. Thus, Dow Jones index data during those months are missing.



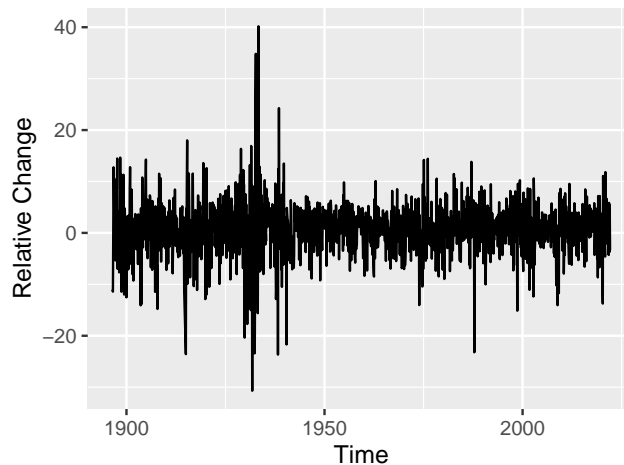
8. Append a column `rel_change_pct` with the relative change of the Dow Jones (in percent) compared to its value at the start of the month. In the first row, enter `NA` as a sign that there is no closing value in the previous month; Charles Dow calculated the eponymous index for the first time in May 1896. Again, do not use a for-loop.

```
# Calculate the percentage change from previous month
dj$rel_change_pct <- 100 * (lead(dj$close_value) / dj$close_value - 1)

# We also need to shift all rows of rel_change_pct down by 1 to match to the
# correct months. The first row is automatically filled with NA
dj$rel_change_pct <- lag(dj$rel_change_pct)
```

9. Make a quick-and-dirty plot of the Dow Jones's relative change as a function of time using the following code:

```
# Remove first row because it contains NA
ggplot(dj[-1, ], aes(mdy(date), rel_change_pct)) +
  geom_line() +
  labs(
    x = "Time",
    y = "Relative Change"
  )
```



10. Which month saw the largest relative increase in the history of the Dow Jones? Perform some research about the reason for the stock market rally. Summarise your findings in maximally 10 sentences.

```
# We can see that the largest relative increase happened in April, 1933
dj$date[which.max(dj$rel_change_pct)]
```

```
## [1] "04/29/1933"
```

The April of 1933 marked the implementation of several important policies of U.S. President Franklin D. Roosevelt's New Deal. These include an announcement that the U.S. will leave the gold standard, a monetary system in which currency is backed by gold. These policies were a tailwind for the stock market as it creates a positive economic outlook and boosts market confidence.

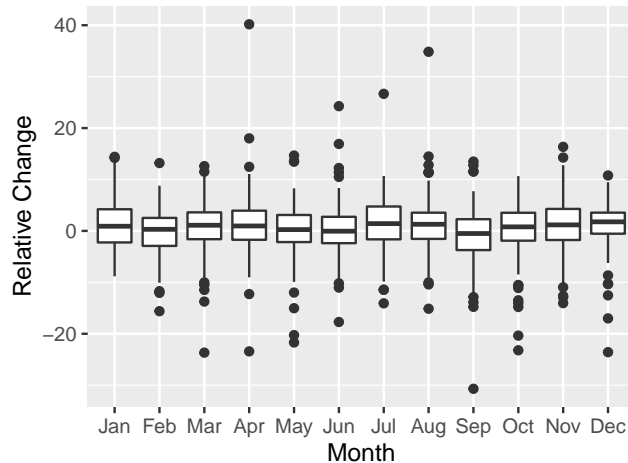
11. For the purpose of plotting, append a column `month_abb` that contains the abbreviated name of the month as a factor (e.g. "Jan", "Feb"). You can find the abbreviations in R's built-in vector `month.abb`. Sort levels in chronological order from "Jan" to "Dec".

```
# Add a column as a factor
dj$month_abb <- as.factor(month.abb[dj$month])

# Sort the levels of the factor in chronological order
dj[["month_abb"]] <- factor(dj[["month_abb"]], levels = c(month.abb))
```

12. Make a quick-and-dirty box plot of relative change as a function of month. Which preliminary conclusion can you draw from the box plot?

```
ggplot(dj[-1, ], aes(month_abb, rel_change_pct)) +
  geom_boxplot() +
  labs(
    x = "Month",
    y = "Relative Change"
  )
```



At first glance, it seems that there are not much variation among months. The median of each month are all around/slightly above zero, and the spreads of data are also similar. This shows how box plot is a good tool to visualize distribution but not ideal to make comparisons.

13. Which month has the highest median relative change? Which month has the lowest (i.e. the most strongly negative) change? No for-loop! In Quantitative Reasoning, you learned how to get the answer with `aggregate()`.

```
# Aggregating with median function
agg_median <- aggregate(rel_change_pct ~ month_abb,
  data = dj,
  median
)

arranged_changes <- arrange(agg_median, rel_change_pct)

# Lowest change
arranged_changes[1, ]

##   month_abb rel_change_pct
## 1      Sep    -0.510084

# Highest change
arranged_changes[12, ]

##   month_abb rel_change_pct
## 12     Dec     1.764781
```


We can see that September is the month with lowest negative median relative change while December has the highest change.

14. Now we know the differences in the median between different months, but are these differences statistically significant?

The Kruskal-Wallis test is a nonparametric statistical method to infer whether samples come from the same distribution. To test the hypothesis that there are monthly differences, the null hypothesis is that the relative changes come from the same distribution in all months. Reactivate your Quantitative Reasoning knowledge to interpret the result of:

```
kruskal.test(rel_change_pct ~ month, data = dj)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: rel_change_pct by month  
## Kruskal-Wallis chi-squared = 31.371, df = 11, p-value = 0.0009616
```

Since the p-value  0.001282, is very small, we have sufficient evidence to reject the null hypothesis that the relative changes come from the same distribution in all months at even 1% significance level. As such, we can conclude with relatively large confidence that the relative changes do not come from the same distribution in all months. This means that it is possible that the stock market tends to perform better or worse in certain months compared to other months.

