Problem 2 (13 credits)

HW3

Michael Brucek (mbrucek), Eduardo Chavez (echavezh), Kevin Grady (grady133), Danny Moncada (monca016), Tian Zhang (zhan7003)

March 25, 2020

```r
suppressWarnings(suppressPackageStartupMessages({
  library(assertthat) # we use validate_that() function
  library(readr) # we use read_csv()
  library(TSA)
  library(lubridate)
  library(xts)
  library(forecast)
  library(ggplot2)
  library(dplyr)
  library(tseries)
}))

set.seed(12345)
```

Spurious Regression

For this problem, we would look at one of the most important and dangerous concepts in time series – a *spurious regression*. Spurious regression will arise any time you forget about time series nature of your data and attempt to use a linear regression for hypothesis testing.

For this problem, we would try to use real Tesla stock prices and regress it on an absolutely arbitrarily generated random walk that was just generated by you.

Intuitively, you should understand that there could be no relationship between an arbitrary independent random walk that you just generated on your computer and a real price of Tesla stock. Yet, a linear regression will consistently find a "significant" relationship.

Question 1 (3 credits)

First, let's load TESLA stock *closing* prices from the CSV file and select *only the time window for the last 1 year* (from 2017-02-09 till 2018-02-08 including both). Also, please plot the selected time window in ggplot.

Please save the resulting data into data.frame `df1`. Please make sure that the data.frame has two columns:

- `Index` which describes the date of the observation
- `TSLA` which describe the *closing* price of Tesla stock for that day

Please learn to work with compressed CSV files directly. Do not ungzip the file in your code. Use `read_csv`.

Also, for the entire assignment, the unit tests will suggest the proper format of your answer. If you provided the answer to the question in the proper format, the `validate_that` unit tests would all report TRUE.

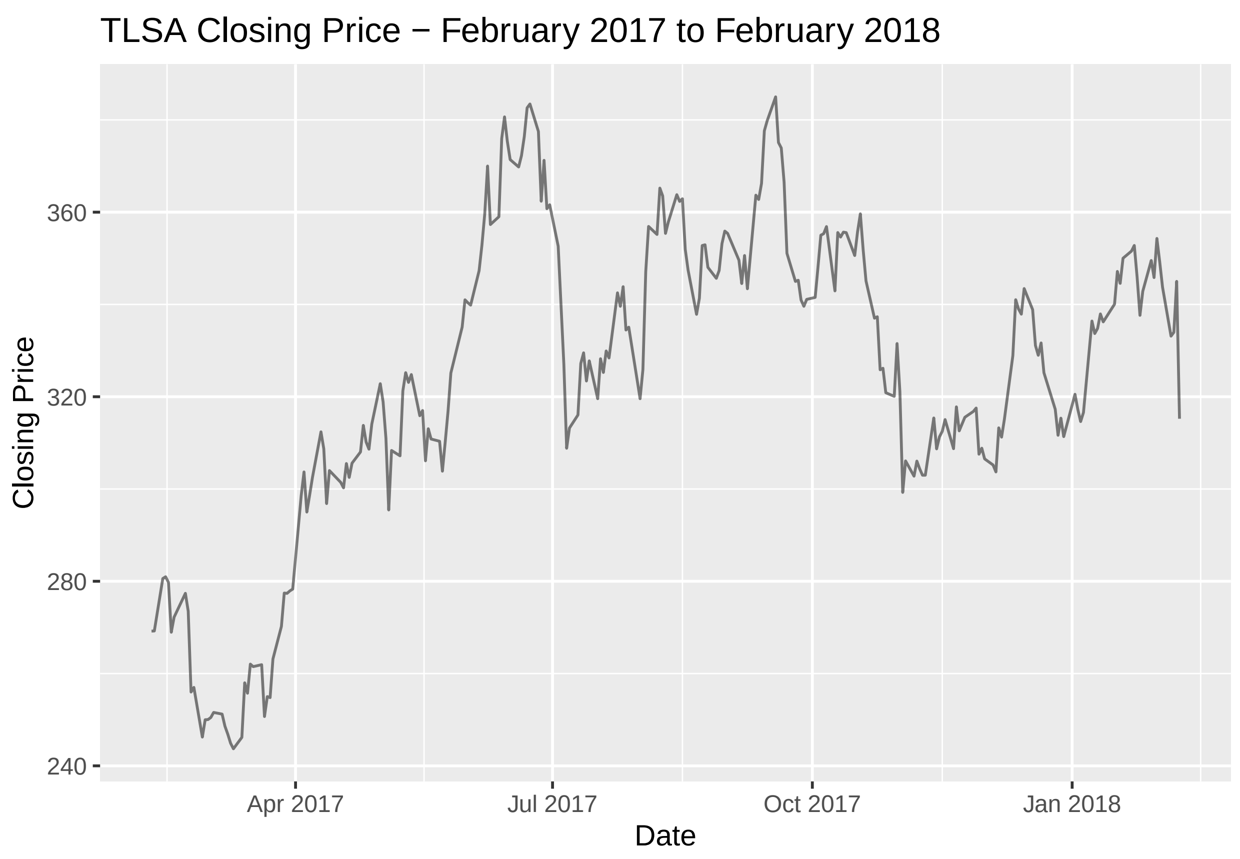
```
TSLA <- read_csv("TSLA.csv.gz") # Please do not change this line
```

```
## Parsed with column specification:
## cols(
##   Index = col_date(format = ""),
##   TSLA.Open = col_double(),
##   TSLA.High = col_double(),
##   TSLA.Low = col_double(),
##   TSLA.Close = col_double(),
##   TSLA.Volume = col_double()
## )
```

```
head(TSLA)
```

```
## # A tibble: 6 x 6
##   Index      TSLA.Open TSLA.High TSLA.Low TSLA.Close TSLA.Volume
##   <date>         <dbl>     <dbl>    <dbl>      <dbl>       <dbl>
## 1 2010-06-29      19        25       17.5       23.9    18783276
## 2 2010-06-30      25.8      30.4     23.3       23.8    17194394
## 3 2010-07-01      25        25.9     20.3       22.0     8229863
## 4 2010-07-02      23        23.1     18.7       19.2     5141807
## 5 2010-07-06      20        20       15.8       16.1     6879296
## 6 2010-07-07      16.4      16.6     15.0       15.8     6924914
```

```
# you will need to select the appropriate time period
df1 <- TSLA[TSLA$Index >= "2017-02-09" & TSLA$Index <= "2018-02-08",] %>% select(1, 5) %>%
            rename(TSLA = TSLA.Close)

# don't forget to plot the selected window of stock prices
p1 <- ggplot(data = df1, aes(x = Index, y = TSLA)) + geom_line(alpha = 0.5) +
          ggtitle("TLSA Closing Price - February 2017 to February 2018") +
      labs(x = "Date", y = "Closing Price")
p1
```



```r
# --------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# --------------------
validate_that(nrow(df1) == 252L) # should be 252 trading days
validate_that(df1[[1,"Index"]] == as.Date("2017-02-09")) # should start with 2017-02-09
validate_that(all(colnames(df1) == c("Index","TSLA"))) # these specific column names
validate_that("ggplot" %in% class(p1)) # don't forget to plot it!
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

Question 2 (1 credit)

Please simulate an arbitrary sample path from a pure random walk and save it into a numeric vector `Xt`

```r
set.seed(12345) # Please do not change the seed

N = 252L
e = rnorm(N)

Xt <- cumsum(e)
```

```r
# --------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# --------------------

validate_that(length(Xt) == 252L) # should be a vector with 252 numbers
```

```
## [1] TRUE
```

Question 3 (1 credit)

Please regress the raw Tesla closing stock prices on this simulated random walk, save and report the results. Please save the estimated model into `lm_3` variable and print `summary(lm3)` as well.

Unless you are lucky, you should see that the linear model feels that the random walk that you just simulated on your computer seems to "explain" the Tesla stock price. This is the classic spurious regression.

```r
tsla_close <- df1$TSLA

lm_3 <- lm(Xt ~ tsla_close)

# don't forget to print the estimates
summary(lm_3)
```

```
##
## Call:
## lm(formula = Xt ~ tsla_close)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.471  -5.471  -1.000   5.438  14.219
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -47.71088    3.98539  -11.97   <2e-16 ***
## tsla_close    0.20789    0.01223   17.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.442 on 250 degrees of freedom
## Multiple R-squared:  0.5362, Adjusted R-squared:  0.5344
## F-statistic: 289.1 on 1 and 250 DF,  p-value: < 2.2e-16
```

```r
# --------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# --------------------
validate_that("lm" %in% class(lm_3))
```

```
## [1] TRUE
```

Question 4 (1 credit)

You will probably remember that for financial assets it is the percentage change that is important so your first suggestion might be to compute the logs of Tesla stock prices and regress the logs of the prices on your random walk `Xt`.

Try doing that! Please compute `df4` data.frame that adds one more column: `log_TSLA` which is just a log of a Tesla stock price on that day.

Please regress `log_TSLA` on random walk `Xt` and save the estimated model into `lm_4`. Don't forget to print `summary(lm4)` as well. You should see that it doesn't seem to affect the spurious statistical significance.

```
# Transform TSLA closing price to LOG of closing price
df4 <- df1 %>% mutate(log_TSLA = log(TSLA))

# Save as a new variable
log_tsla_close <- df4$log_TSLA

# Perform spurious regression using log price instead
lm_4 <- lm(Xt ~ log_tsla_close)

# don't forget to print the estimates
summary(lm_4)
```

```
##
## Call:
## lm(formula = Xt ~ log_tsla_close)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.047  -5.350  -1.044   5.104  13.841
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -361.919     21.272  -17.01   <2e-16 ***
## log_tsla_close   66.070      3.682   17.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.254 on 250 degrees of freedom
## Multiple R-squared:  0.5629, Adjusted R-squared:  0.5612
## F-statistic:   322 on 1 and 250 DF,  p-value: < 2.2e-16
```

```
# --------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# --------------------
validate_that(nrow(df4) == nrow(df1))
validate_that("lm" %in% class(lm_4))
validate_that(all(colnames(df4) == c("Index","TSLA","log_TSLA"))) # these specific column names
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

Question 5 (1 credit)

Now it is time to unfold the mystery. It looks like the main problem is that $\log(TSLA)$ is actually following a random walk.

$$\log(\text{TSLA}_t) = \log(\text{TSLA}_{t-1}) + e_t$$

where $e_t \sim N(0, \sigma^2)$ and σ^2 is called the volatility (of Tesla stock returns).

Let's verify it.

- Please do an `adf.test` on $\log(TSLA)$ from `df4` to find out and print the results. What is your conclusion about whether it has a random walk or not?
- Please run an `auto.arima` on $\log(TSLA)$ from `df4` and see if it discovers any random walk components

```
adf.test(df4$log_TSLA)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df4$log_TSLA
## Dickey-Fuller = -2.0608, Lag order = 6, p-value = 0.5502
## alternative hypothesis: stationary
```

```
# The Augmented Dickey-Fuller test slightly tell us that this process is stationary, but
# it is not definitive.  The p-value is only 0.55, which is not normally enough for us to
# reject the null hypothesis that the process is not stationary.

auto.arima(df4$log_TSLA)
```

```
## Series: df4$log_TSLA
## ARIMA(0,1,0)
##
## sigma^2 estimated as 0.0005424:  log likelihood=587.57
## AIC=-1173.15   AICc=-1173.13   BIC=-1169.62
```

```
# The Auto Arima is much stronger evidence that this is indeed a random walk.  It tells
# us that the model is ARIMA(0,1,0), which tell us this is a random walk.  It is a cumul-
# ative sum of an i.i.d. process which itself is known as ARIMA(0,0,0).  So we set this
# to TRUE.

q5_log_TSLA_is_random_walk <- TRUE # Please change to TRUE or FALSE
```

```
# -------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# -------------------
validate_that(is.logical(q5_log_TSLA_is_random_walk))
```

```
## [1] TRUE
```

Question 6 (1 credit)

Now let's do it the correct way. We cancel random walks by taking first differences. Let's see if a *change* in log-Tesla stock prices (this is called "stock return") is explained by a *change* in X_t.

- Hint:
  - take a look at `diff()` function to do first-differences

Please save the results into `lm_6` objects and display the summary. You should see that there is no longer any significant relationship that remains.

Remember: be very-very-very careful when doing hypothesis testing on non-stationary time series!

These issues are not limited to stock prices. They may come up even when studying stuff like *the effect of discounts on the total membership*. As long as the outcome variable has a random walk component, you are risking a spurious regression.

```r
# Perform first order differencing on the log of the closing price for TSLA
log_tsla_close_diff <- diff(df4$log_TSLA, differences = 1)

lm_6 <- lm(Xt[1:251] ~ log_tsla_close_diff)

# don't forget to print the results
summary(lm_6)
```

```
##
## Call:
## lm(formula = Xt[1:251] ~ log_tsla_close_diff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.887  -8.214   3.494   7.367  12.787
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          19.6577     0.5958  32.991   <2e-16 ***
## log_tsla_close_diff -18.3180    25.5872  -0.716    0.475
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.436 on 249 degrees of freedom
## Multiple R-squared:  0.002054,   Adjusted R-squared:  -0.001954
## F-statistic: 0.5125 on 1 and 249 DF,  p-value: 0.4747
```

```r
# --------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# --------------------
validate_that("lm" %in% class(lm_6))
```

```
## [1] TRUE
```

Question 7 (3 credits)

Finally, you might still be puzzled about what log has to do with percentages and why taking the first differences in logs are called "stock returns". You may also have heard about the term *volatility* of stock returns. This question will attempt to define both of these.

As we discussed, for financial instruments it is the *percent* change that is important not the absolute change. This is why financial news are obsessed with some stock gaining 10% in a day or with global markets dropping by 5% etc.

There are several ways to compute the percent change for Tesla stock:

1. *Daily* returns:

$$\text{Return}_t = \frac{\text{Price}_t - \text{Price}_{t-1}}{\text{Price}_{t-1}}$$

Daily return is a traditional financial performance measure.

2. Instantaneous daily return rate (also called "logarithmic return"):

$$r_t = \log(\text{Price}_t) - \log(\text{Price}_{t-1})$$

Logarithmic return is an important measure in continuous-time finance (which models high-frequency trading) and captures an instantaneous growth rate of the stock price just like a derivative captures the slope of growth.

You will see that the two return measures will be very close to each other. This is the reason why I am so freely applying log-transformation everytime we care about percent change rather than absolute change.

Please compute both of these measures for Tesla stock for each day and plot the histograms for each one. To do so, please create data.frame `df7` that has the following columns:

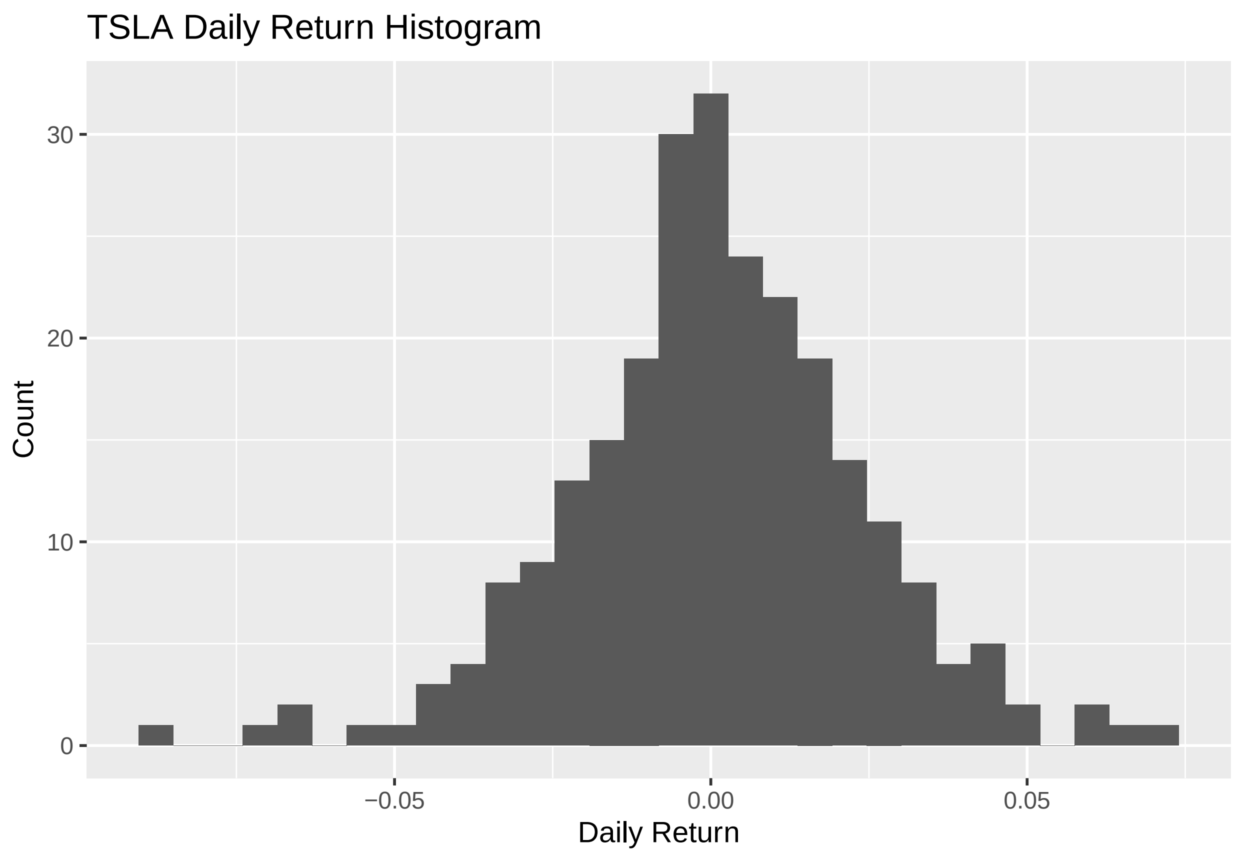
- `Index`: the date of the observation
- `TSLA`: the closing price as before
- `log_TSLA`: the log of closing price as before
- `daily_return`: this is the newly computed daily return of TSLA stock price
- `daily_return_rate`: this is the newly computed daily return rate of TSLA stock price

You should see that `daily_return` is quite similar to the instantaneous `daily_return_rate`.

- Hints:
  - the very first day in our working data (2017-02-09) will not have a well defined return since it is the first day in the data. Feel free to assume the return is zero on that day.

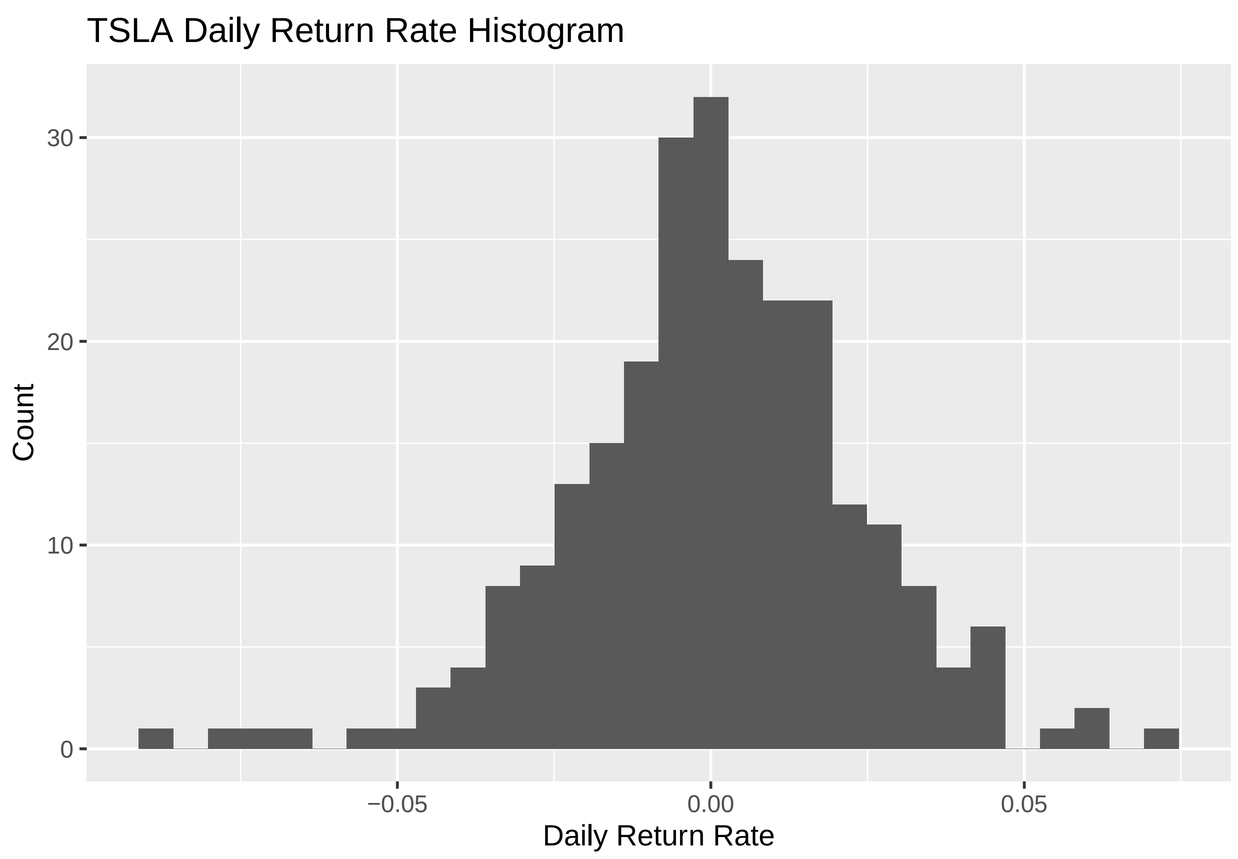
```r
# Cretae a dataframe using the instructions above

# Create a price variable to save some typing when creating the DF
price <- df1$TSLA

df7 <- data.frame(Index = df1$Index, TSLA = price, log_TSLA = log(price),
                  # Subtract the lag of price and divde by the lag of price
                  daily_return = (price - lag(price)) / lag(price),
                   # Subtract the log of price by the log of the LAG of price
                   daily_return_rate = log(price) - log(lag(price)))

# Replace the daily return and daily return rate with zeros per our assumption
df7[is.na(df7)] <- 0

# Plot the daily return as a histogram
p7_daily <- ggplot(data = df7, aes(daily_return)) + geom_histogram() +
  ggtitle("TSLA Daily Return Histogram") + labs(x = "Daily Return", y = "Count")
p7_daily
```

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



```
# Plot the daily return rate as a histogram
p7_logarithmic <- ggplot(data = df7, aes(daily_return_rate)) + geom_histogram() +
  ggtitle("TSLA Daily Return Rate Histogram") + labs(x = "Daily Return Rate", y = "Count")
p7_logarithmic
```

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



```r
# -------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# -------------------
validate_that(nrow(df7) == nrow(df1))
validate_that("ggplot" %in% class(p7_daily))
validate_that("ggplot" %in% class(p7_logarithmic))
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

Question 8 (2 credits)

Finally, please calculate the *daily* and *annualized* volatility of Tesla stock return rate (that is, for logarithmic return)

- Daily volatility is just a standard deviation of the distribution of daily logarithmic return that you computed in the previous question.

Also, in finance, traders usually use annualized volatility rather than daily volatility. Remember that for the random walk:

$$\text{Var}(Y_t) = t$$

and therefore,

$$\text{Std.Dev}(Y_t) = \sqrt{t}$$

There are 252 trading days in a year so $t = 252$. In other words:

- Annualized volatility of a stock is defined as daily volatility times $\sqrt{252}$.

```
# Please replace NA with the computed plot

logaritmic_daily_return_volatility_TSLA <- sd(df7$daily_return_rate)
logaritmic_annualized_return_volatility_TSLA <- sd(df7$daily_return_rate) * sqrt(252)

logaritmic_daily_return_volatility_TSLA
```

```
## [1] 0.02327829
```

```
logaritmic_annualized_return_volatility_TSLA
```

```
## [1] 0.3695313
```

```
# -------------------
# Below are unit tests
# (all should produce true once you solve Hw)
# -------------------
validate_that(is.numeric(logaritmic_daily_return_volatility_TSLA))
validate_that(is.numeric(logaritmic_annualized_return_volatility_TSLA))
```

```
## [1] TRUE
## [1] TRUE
```