MATH70079 - Introduction to Statistical Finance

Coursework Part B

Authors:
Félix Hovine (01354188), Victor Khamesi (02091191), Cyrus Yavari(02158751) and Tinius Mellbye (02155151)

February, 2021

In this report, we will look at the Visa Inc. stock (Ticker: V) traded on The New York Stock Exchange (NYSE). Firstly, we will look at the stock's return time series. Note that we will throughout use the term "return" to mean "log-return" unless otherwise stated. Secondly, we investigate to what degree the stock is in line with the stylised facts. Lastly, a proposal distribution more suitable than the Normal distribution will be presented, namely, the class of stable distributions, which in fact contains the Normal and Cauchy distribution to name a few.

(i) Log-return time series

The Visa stock's evolution from 2010 throughout 2021 is presented in Figure 1. One may observe a relatively stable upwards trend in the daily closing prices up to the beginning of 2020. In late February and early March 2020, many stock markets around the world fell as a result of the Covid-19 pandemic, which was followed by quantitative easing contributing to a relatively fast recovery. This appear to be reflected in the Visa stock as well. The disruption is also apparent in the plots of the returns and absolute returns, the middle and bottom plot of Figure 1. Note that the returns time series shown in Figure 1 looks like "noise" and modelling it directly from this plot would be a tough task. However, further studies in this report will reveal some characteristic properties of these return series.

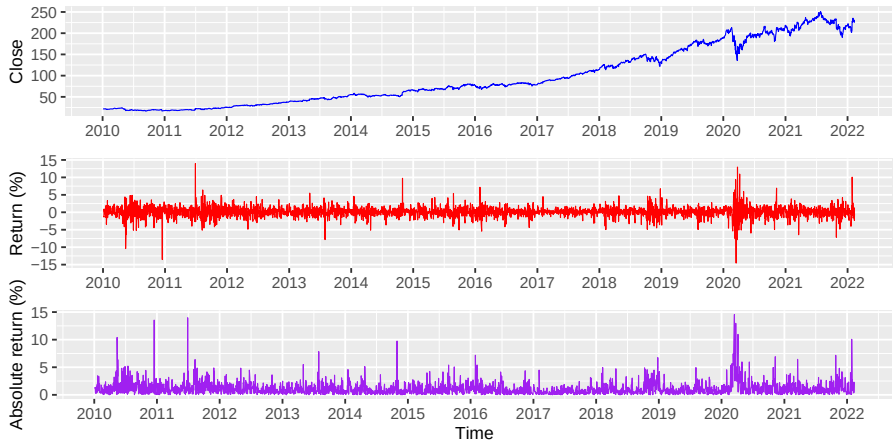


Figure 1: A trace plot of the daily closing prices from 2010 up to now (14$^{\text{th}}$ February 2022) is depicted in the top plot, showing an overall increasing trend. In 2020 a clear disruption occurred. The disruption is reflected in the plot of the log-returns (middle) and absolute log-returns (bottom) showing high volatility.

(ii) Stylised facts $\mathbf{SF_1 - SF_3}$

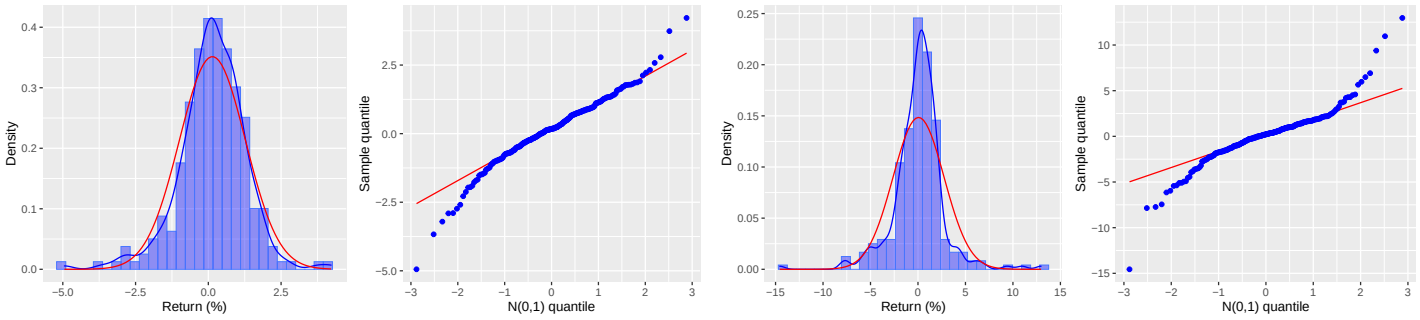
We now split the Visa stock daily data into three years: 2019, 2020 and 2021, in order to check whether the stock exhibits the following stylised facts ($\mathbf{SF_1 - SF_3}$).

- **SF$_1$**, The unconditional distribution of return (at a daily or shorter time scale) is markedly non-normal with heavy tails and possibly mild asymmetry.
- **SF$_2$**, Returns are serially uncorrelated.
- **SF$_3$**, Volatility is clustered and persistent.

SF$_1$: returns are not normally distributed

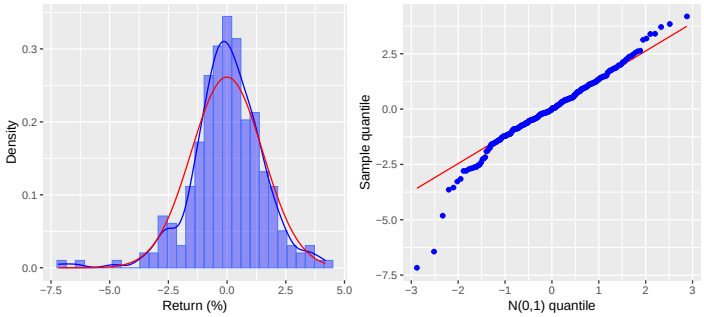
One can check the first stylised fact by analysing the histogram and the kernel density estimate of the returns, along with a normal distribution fitted to the data and also a QQ-plot. Figure 2 exhibits that for each year the returns are non-normal with heavy tails.

In order to quantify it, one may look at the sample excess kurtosis and sample skewness which measures respectively the tail weight and symmetry. By using `R` functions to calculate the kurtosis and skewness is given in Table 1. It can be seen that there was some disruption in early 2020 and mid-2021. By including the standard error computed using bootstrap, the excess kurtosis is well above zero for all three years. Note that the estimates for the mean return all have high standard errors: this quantity is therefore very hard to estimate correctly. The skewness is different from zero for all three years, even when including the standard error. There is a chance the skewness is zero in 2020 but the kurtosis is so large the distribution is clearly non normal and one would say that stylised **SF$_1$** is satisfied.



(a) 2019

(b) 2020



(c) 2021

Figure 2: Histogram and kernel density (blue line) of daily returns for each year, along with a normal density fitted on sample values in red line (left). QQ-plot of returns compared to a normal density line $y = x$

	2019	2020	2021
$\hat{r}$	0.1403 (0.0713)	0.0601 (0.1672)	-0.0037 (0.0942)
$\hat{s}$	1.1352 (0.0746)	2.6873 (0.2447)	1.5257 (0.1036)
$\hat{b}$	-0.4987 (0.3480)	-0.1129 (0.7152)	-0.6771 (0.3590)
$\hat{k}$	2.4612 (0.8834)	6.5745 (1.8894)	2.8289 (1.1641)

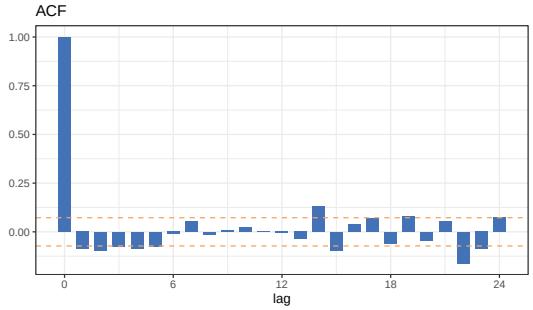
Table 1: Sample statistics of the daily returns for each considered year. Standard Errors reported in parentheses are computed using non-parametric bootstrap. Where $\hat{r}$ is the mean return, $\hat{s}$ is the standard deviation of the return, $\hat{b}$ is the skewness and $\hat{k}$ is the kurtosis.

SF_2: Returns are serially uncorrelated

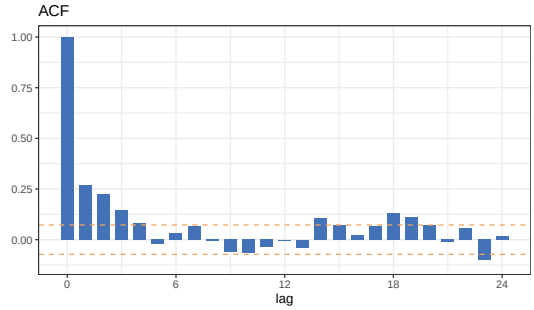
The second stylised fact indicates that the returns should be serially uncorrelated. To assess if the three time series are aligned with this fact, we will use the empirical auto-correlation function which is given by the following:

$$\hat{\rho}(k) := \frac{\sum_{t=k+1}^{T} (r_t - \bar{r})(r_{t-k} - \bar{r})}{\sum_{t=1}^{T} (r_t - \bar{r})^2}, \quad k = 0, 1, \ldots, T-1$$

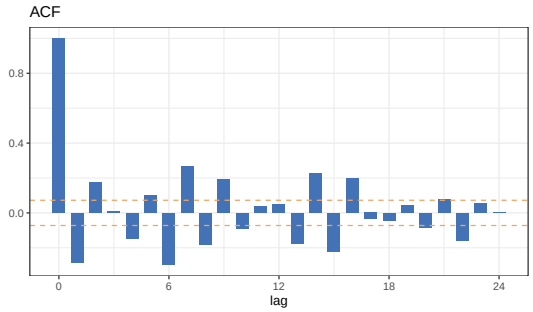
where $\bar{r}$ denotes the mean return on the period. The empirical ACF for each year are plotted in Figure 3.



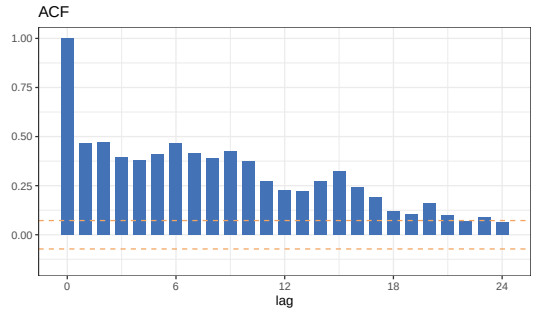
(a) 2019 ACF plot for log-returns



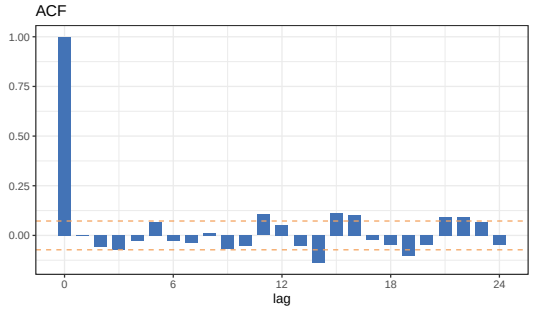
(b) 2019 ACF plot for absolute log-returns



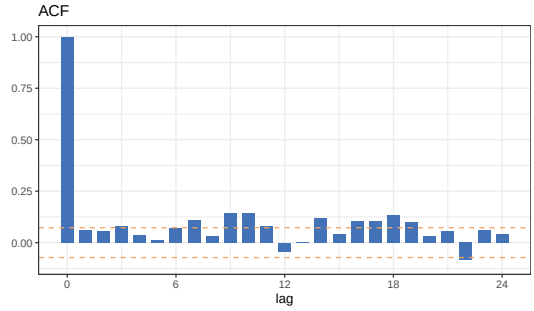
(c) 2020 ACF plot for log-returns



(d) 2020 ACF plot for absolute log-returns



(e) 2019 ACF plot for log-returns



(f) 2021 ACF plot for absolute log-returns

Figure 3: Auto-correlation plots for log-returns and absolute log-returns

For years 2019 and 2021 the auto correlations are not significantly different from zero. However, this is different in 2020 where there seem to be some points of large auto-correlation. The large auto-correlation could be attributed to the Covid disruption in 2020. So we can conclude that 2019 and 2021 satisfy $\mathbf{SF_2}$ but 2020 might not.

$\mathbf{SF_3}$: volatility is clustered and persistent

We then check if the volatility is clustered and persistent by looking at the auto correlation plot of the absolute returns, shown in Figure 3. This measures volatility as a large price change one day leads to a large price change the next day and it decays slowly with lag time. This is seen in 2020 and for the first few lag times in 2019. However, this is not seen in 2021 and one may conclude that $\mathbf{SF_3}$ holds for 2020 but not for 2021 or 2019. A reason which might explain why it does not hold for each year may be due to the time window chosen. Indeed, we are only looking at a one-year period, while this stylised fact might only hold for longer period of time (here a year only contains roughly 250 data points for each business day, whereas considering these plots on multiple consecutive years would have given significantly better results).

(iii) Distribution modelling for the returns

The unconditional distribution of logarithmic (log) returns was shown to be non-normal in Section (**ii**). More specifically, it was highlighted that the distribution has a leptokurtic shape with an acute peak and heavy tails compared to the normal distribution. Furthermore, for the three time series, slight asymmetry was observed. The observed returns can be described well by a distribution which is unimodal, skewed and has heavy tails. Distributions which exhibit such properties are the Stable distributions. These distributions form a general class of probability distributions, and can be defined by location-scale transformations (Ibe, 2013).

The most general stable distribution is characterized by four parameters:

- The tail index, $\alpha \in (0, 2]$. It dictates the probability in the extreme tails. A smaller value of *alpha*, will yield heavier tails.
- The skewness index, $\beta \in [-1, 1]$. When β is negative, the distribution is negatively skewed, and when it is postive, the distribution is positively skewed.
- The scale parameter, $\sigma > 0$. This parameter is a measure of dispersion of the density function.
- The location parameter, $\mu \in \mathbb{R}$. This parameter can shift the distribution either to the right or left.

In this report, we will investigate how does the most general Stable distribution compares to the Cauchy and Normal distributions. It is important to note that the Cauchy distribution is a special case of the stable distribution, with $\alpha = 1$ and $\beta = 0$. Moreover the Normal is also a special case, with $\alpha = 2$ and $\beta = 0$. By fitting, the Stable, Cauchy and Normal, we will be able to see if it is necessary to increase the number of parameters and thus the complexity of our models in order to have a good fit.

The scale and location parameters for the Normal distribution was found using the maximum likelihood estimates. For the scale parameter, it is equivalent to the biased standard deviation and for the location parameter, it is the sample mean of the returns. However, for the Cauchy distribution, the MLEs for the parameters do not have a closed-form. As a result, direct optimization of the log-likelihood must be performed. For both distributions, the function `fitdistr()` from the `MASS` package was used. For the Gaussian distribution, the function uses the closed-form MLEs and for the Cauchy distribution, numerical optimisation is used.

Finally, for the general Stable distribution, the quantiles method was used and was developed by McCulloch (1986). This method estimates the four parameters using five pre-determined sample quantiles with the help of accompanying tables. Empirical research done by Kateregga et al (2017) have shown that this method is consistent with the Maximum Likelihood method. It is worth noting that the McCulloc is only valid for tail index

between 0.6 and 2. However, this restriction was not an issue as α was consistently larger than 0.6. The function used to perform the quantiles method is `McCullochParametersEstim()` which is from the package `StableEstim`.

The estimates for the parameters of the Normal, Cauchy and Stable distribution are shown in Table 2 and 3.

	Normal		**Cauchy**	
	μ	σ	x_0	λ
2019	0.1403	1.1329	0.2000	0.5852
2020	0.0601	2.6820	0.2697	1.0747
2021	-0.0037	1.5227	0.0183	0.7850

Table 2: Estimate of the parameters for the Normal and Cauchy distribution using the `fitdistr()` function

	Stable			
	α	β	γ	δ
2019	1.7160	-0.4080	0.6739	0.2162
2020	1.4170	-0.2640	1.2364	0.2961
2021	1.6470	-0.1890	0.8963	0.0078

Table 3: Estimate of the parameters for the Stable distribution using the `McCullochParametersEstim()` function

Figure 4, 5 and 6 display the the density histogram and empirical cumulative distribution functions for the log-returns in 2019, 2020 and 2021 respectively. In the density histogram figure, the Normal, Cauchy, Stable probability density functions using the parameters derived in Table 2 and 3 were also plotted. In addition, for the ECDF plots, the cumulative distribution functions were also plotted for the three different densities.

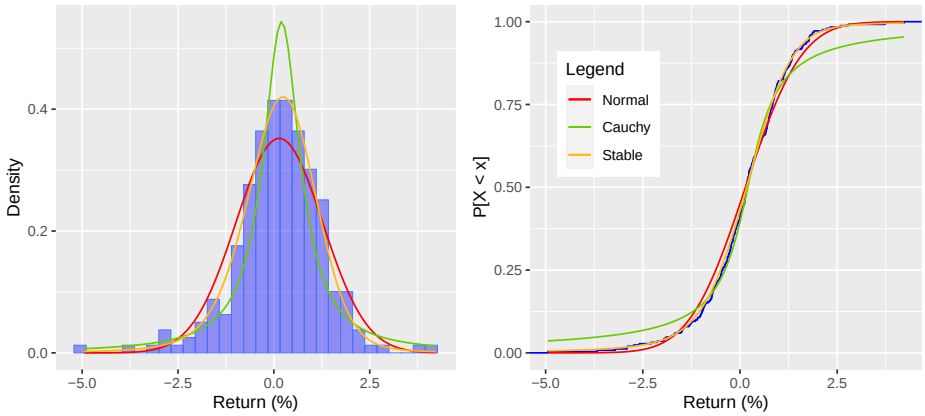


Figure 4: (Left) Density histogram for the log-returns in 2019 with the Normal, Cauchy and Stable probability density functions. (Right) ECDF for the log-returns in 2019 with the CDF of the Normal, Cauchy and Stable probability density functions

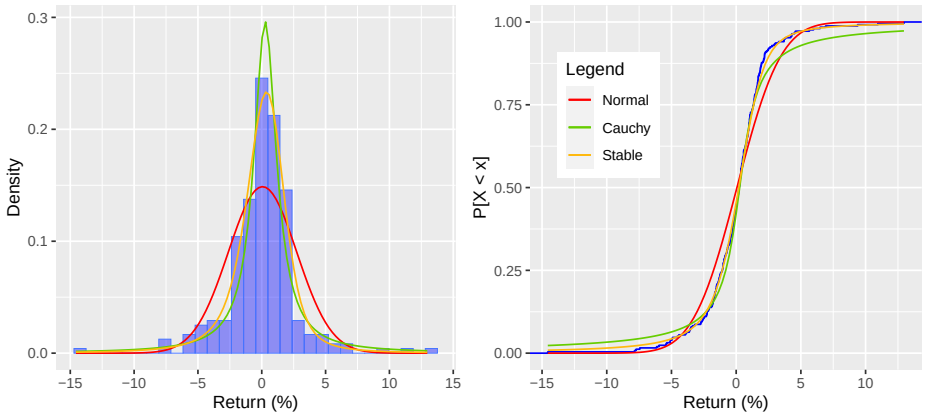


Figure 5: (Left) Density histogram for the log-returns in 2020 with the Normal, Cauchy and Stable probability density functions. (Right) ECDF for the log-returns in 2020 with the CDF of the Normal, Cauchy and Stable probability density functions

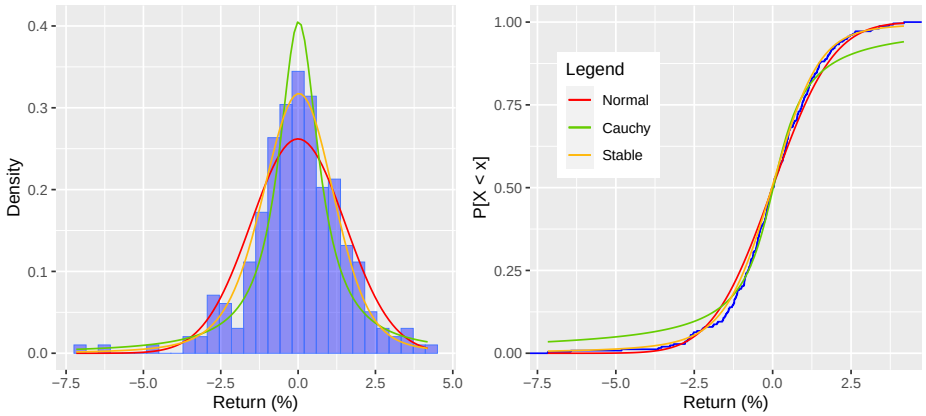


Figure 6: (Left) Density histogram for the log-returns in 2021 with the Normal, Cauchy and Stable probability density functions. (Right) ECDF for the log-returns in 2021 with the CDF of the Normal, Cauchy and Stable probability density functions

From Figure 4, 5 and 6, one can see that the Stable distribution seems to describe the best the unconditional distribution of returns compared to the Normal and Cauchy distributions. The Cauchy distribution has tails that are too heavy (this can be seen especially in the CDF plot) and seems to be too narrow at the mode. The Normal distribution is too large at the mode, and has tails which seems to be slightly too small compared to the data. On the other hand, one can see that the Stable distribution is almost identical to the ECDF for the three consecutive years.

To corroborate the visual assessment, we used the Kolmogorov–Smirnov (KS) test to assess the goodness of fit quantitatively. The test is non-parametric. It can be used to assess whether or not a sample comes from a population with a specific probability distribution by computing the maximum vertical distance between the theoretical CDF and the ECDF D. Therefore, smaller is D, better is the fit. In a KS test, the null hypothesis is that the sample was indeed generated by the population probability distribution. In Table 4, 5 and 6, the p-values of each KS test are presented. If the p-value is lower than a 5% threshold, one should reject the null hypothesis, and thus the observed log-returns are not sampled from the specified probability density.

	D	p-value
Normal	0.0637	0.2585
Cauchy	0.0773	0.0983
Stable	0.0386	0.8479

Table 4: KS tests results for 2019

	D	p-value
Normal	0.1175	0.0019
Cauchy	0.0670	0.2055
Stable	0.0364	0.8903

Table 5: KS tests results for 2020

	D	p-value
Normal	0.0702	0.1667
Cauchy	0.0649	0.2392
Stable	0.0439	0.7176

Table 6: KS tests results for 2021

From Table 4, 5 and 6, the p-value and D value are the greatest for the Stable distribution for 2019, 2020 and 2021, showing that there is greater evidence that the observed data originates from this distribution rather than the Cauchy and Normal distributions. The difference in the D statistic value between the Stable and the other distributions is large, giving more confidence that the stable is the most appropriate distribution to model the return series for each year. This is consistent since the Stable distribution has more parameters than the two others, and therefore is more flexible.

From the visual and quantitative analysis, it seems that the log-returns could be modelled by a Stable distribution for each year. However, the parameter of that distribution should be re-calculated for each series as the distribution parameters seem to change every year according to Table 3. This can easily be explained by financial periods: for instance, 2020 was characterised by the pandemic and quantitative easing which impact a lot the distribution parameters (changes in prices were higher in magnitude than usually) compared to a *bull market* period for example.

References

Ibe, Oliver (2013). *Markov processes for stochastic modeling*. Newnes.

Kateregga, Michael, Sure Mataramvura and David Taylor (2017). 'Parameter estimation for stable distributions with application to commodity futures log-returns'. In: *Cogent Economics & Finance* 5.1, p. 1318813.

McCulloch, J Huston (1986). 'Simple consistent estimators of stable distribution parameters'. In: *Communications in Statistics-Simulation and Computation* 15.4, pp. 1109–1136.

Appendix

A Code

```
library(tidyquant)
library(ggplot2)
library(ggpubr)
library(scales)
library(boot)
library(dplyr)
library(cowplot)

# Taken from https://rh8liuqy.github.io/ACF_PACF_by_ggplot2.html
ggplot.corr <- function(data, lag.max = 24, ci = 0.95, large.sample.size = TRUE,
                        horizontal = TRUE,...) {

  if(horizontal == TRUE) {numofrow <- 1} else {numofrow <- 2}

  list.acf <- acf(data, lag.max = lag.max, type = "correlation", plot = FALSE)
  N <- as.numeric(list.acf$n.used)
  df1 <- data.frame(lag = list.acf$lag, acf = list.acf$acf)
  df1$lag.acf <- dplyr::lag(df1$acf, default = 0)
  df1$lag.acf[2] <- 0
  df1$lag.acf.cumsum <- cumsum((df1$lag.acf)^2)
  df1$acfstd <- sqrt(1/N * (1 + 2 * df1$lag.acf.cumsum))
  df1$acfstd[1] <- 0
  df1 <- select(df1, lag, acf, acfstd)

  list.pacf <- acf(data, lag.max = lag.max, type = "partial", plot = FALSE)
  df2 <- data.frame(lag = list.pacf$lag,pacf = list.pacf$acf)
  df2$pacfstd <- sqrt(1/N)

  if(large.sample.size == TRUE) {
    plot.acf <- ggplot(data = df1, aes( x = lag, y = acf)) +
      geom_area(aes(x = lag, y = qnorm((1+ci)/2)*acfstd), fill = "#B9CFE7") +
      geom_area(aes(x = lag, y = -qnorm((1+ci)/2)*acfstd), fill = "#B9CFE7") +
      geom_col(fill = "#4373B6", width = 0.7) +
      scale_x_continuous(breaks = seq(0,max(df1$lag),6)) +
      scale_y_continuous(name = element_blank(),
                         limits = c(min(df1$acf,df2$pacf),1)) +
      ggtitle("ACF") +
      theme_bw()

    plot.pacf <- ggplot(data = df2, aes(x = lag, y = pacf)) +
      geom_area(aes(x = lag, y = qnorm((1+ci)/2)*pacfstd), fill = "#B9CFE7") +
      geom_area(aes(x = lag, y = -qnorm((1+ci)/2)*pacfstd), fill = "#B9CFE7") +
      geom_col(fill = "#4373B6", width = 0.7) +
      scale_x_continuous(breaks = seq(0,max(df2$lag, na.rm = TRUE),6)) +
      scale_y_continuous(name = element_blank(),
                         limits = c(min(df1$acf,df2$pacf),1)) +
      ggtitle("PACF") +
      theme_bw()
  }
  else {
    plot.acf <- ggplot(data = df1, aes( x = lag, y = acf)) +
      geom_col(fill = "#4373B6", width = 0.7) +
```

```
      geom_hline(yintercept = qnorm((1+ci)/2)/sqrt(N),
                 colour = "sandybrown",
                 linetype = "dashed") +
      geom_hline(yintercept = - qnorm((1+ci)/2)/sqrt(N),
                 colour = "sandybrown",
                 linetype = "dashed") +
      scale_x_continuous(breaks = seq(0,max(df1$lag),6)) +
      scale_y_continuous(name = element_blank(),
                         limits = c(min(df1$acf,df2$pacf),1)) +
      ggtitle("ACF") +
      theme_bw()

    plot.pacf <- ggplot(data = df2, aes(x = lag, y = pacf)) +
      geom_col(fill = "#4373B6", width = 0.7) +
      geom_hline(yintercept = qnorm((1+ci)/2)/sqrt(N),
                 colour = "sandybrown",
                 linetype = "dashed") +
      geom_hline(yintercept = - qnorm((1+ci)/2)/sqrt(N),
                 colour = "sandybrown",
                 linetype = "dashed") +
      scale_x_continuous(breaks = seq(0,max(df2$lag, na.rm = TRUE),6)) +
      scale_y_continuous(name = element_blank(),
                         limits = c(min(df1$acf,df2$pacf),1)) +
      ggtitle("PACF") +
      theme_bw()
  }
  plot.acf
}


options("getSymbols.warning4.0" = FALSE)
options("getSymbols.yahoo.warning" = FALSE)

### Question 1

# Download data
data <- tq_get(x = "V", from = "2010-01-01", to = today())
head(data)

# Compute log returns and absolute log returns
data$log_return = c(NA, 100 * diff(log(data$close), lag = 1))
data$abs_log_return = abs(data$log_return)

returns <- ggplot(data, aes(x = date, y = log_return)) +
  geom_line(size=.3, color="red") +
  scale_x_date(date_breaks = "year" , date_labels = "%Y") +
  theme(axis.title.x = element_blank(),
        axis.title = element_text(size = 10)) +
  labs(y = "Return (%)")

close <- ggplot(data, aes(x = date, y = close)) +
  geom_line(size=.3, color="blue") +
  scale_x_date(date_breaks = "year" , date_labels = "%Y") +
  theme(axis.title.x = element_blank(),
        axis.title = element_text(size = 10)) +
  labs(y = "Close")

abs_returns <- ggplot(data, aes(x = date, y = abs_log_return)) +
```

```
  geom_line(size=.3, color="purple") +
  scale_x_date(date_breaks = "year" , date_labels = "%Y") +
  labs(y = "Absolute return (%)", x = "Time") +
  theme(axis.title = element_text(size = 10))

ggarrange(close, returns, abs_returns,
          ncol = 1, nrow = 3)


### Question 2

meanFunc <- function(x,i){mean(x[i])}
sdFunc <- function(x,i){sd(x[i])}
skewFunc <- function(x,i){skewness(x[i])}
kurtFunc <- function(x,i){kurtosis(x[i])}

# Splitting
data_2019 <- data[data$date >= "2019-01-01" & data$date <= "2019-12-31",]
data_2020 <- data[data$date >= "2020-01-01" & data$date <= "2020-12-31",]
data_2021 <- data[data$date >= "2021-01-01" & data$date <= "2021-12-31",]

# 2019
# SF1: returns are not normally distributed
mean_2019 <- mean(data_2019$log_return)
sd_2019 <- sd(data_2019$log_return)

hist_2019 <- ggplot(data_2019, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  geom_density(lwd = .5, colour = "blue") +
  stat_function(fun = dnorm, args = list(mean = mean_2019, sd = sd_2019),
                colour="red") +
  labs(x="Return (%)", y="Density")


qq_2019 <- ggplot(data_2019, aes(sample = log_return)) +
  stat_qq_line(colour="red") +
  stat_qq(colour="blue") +
  labs(x="N(0,1) quantile", y="Sample quantile")

ggarrange(hist_2019, qq_2019,
          ncol = 2, nrow = 1)

# Mean, SD, skew, kurtosis and Std.Errors estimates using bootstrap
r <- mean(data_2019$log_return)
sd_r <- sd(boot(data_2019$log_return, meanFunc, 1e4)$t)
s <- sd(data_2019$log_return)
sd_s <- sd(boot(data_2019$log_return, sdFunc, 1e4)$t)
b <- skewness(data_2019$log_return)
sd_b <- sd(boot(data_2019$log_return, skewFunc, 1e4)$t)
k <- kurtosis(data_2019$log_return)
sd_k <- sd(boot(data_2019$log_return, kurtFunc, 1e4)$t)
res <- rbind(c(r, s, b, k), c(sd_r, sd_s, sd_b, sd_k))
round(res, 4)

# SF.2: returns are not correlated
ggplot.corr(data = data_2019$log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)
```

```
# SF.3: volatility is clustered and persistent
ggplot.corr(data = data_2019$abs_log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)


# 2020
# SF1: returns are not normally distributed
mean_2020 <- mean(data_2020$log_return)
sd_2020 <- sd(data_2020$log_return)

hist_2020 <- ggplot(data_2020, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  geom_density(lwd = .5, colour = "blue") +
  stat_function(fun = dnorm, args = list(mean = mean_2020, sd = sd_2020),
                colour="red") +
  labs(x="Return (%)", y="Density")


qq_2020 <- ggplot(data_2020, aes(sample = log_return)) +
  stat_qq_line(colour="red") +
  stat_qq(colour="blue") +
  labs(x="N(0,1) quantile", y="Sample quantile")

ggarrange(hist_2020, qq_2020,
          ncol = 2, nrow = 1)

r <- mean(data_2020$log_return)
sd_r <- sd(boot(data_2020$log_return, meanFunc, 1e4)$t)
s <- sd(data_2020$log_return)
sd_s <- sd(boot(data_2020$log_return, sdFunc, 1e4)$t)
b <- skewness(data_2020$log_return)
sd_b <- sd(boot(data_2020$log_return, skewFunc, 1e4)$t)
k <- kurtosis(data_2020$log_return)
sd_k <- sd(boot(data_2020$log_return, kurtFunc, 1e4)$t)
res <- rbind(c(r, s, b, k), c(sd_r, sd_s, sd_b, sd_k))
round(res, 4)

# SF.2: returns are not correlated
ggplot.corr(data = data_2020$log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)

# SF.3: volatility is clustered and persistent
ggplot.corr(data = data_2020$abs_log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)

# 2021
# SF1: returns are not normally distributed
mean_2021 <- mean(data_2021$log_return)
sd_2021 <- sd(data_2021$log_return)

hist_2021 <- ggplot(data_2021, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  geom_density(lwd = .5, colour = "blue") +
  stat_function(fun = dnorm, args = list(mean = mean_2021, sd = sd_2021),
                colour="red") +
```

```
  labs(x="Return (%)", y="Density")


qq_2021 <- ggplot(data_2021, aes(sample = log_return)) +
  stat_qq_line(colour="red") +
  stat_qq(colour="blue") +
  labs(x="N(0,1) quantile", y="Sample quantile")

ggarrange(hist_2021, qq_2021,
          ncol = 2, nrow = 1)

r <- mean(data_2021$log_return)
sd_r <- sd(boot(data_2021$log_return, meanFunc, 1e4)$t)
s <- sd(data_2021$log_return)
sd_s <- sd(boot(data_2021$log_return, sdFunc, 1e4)$t)
b <- skewness(data_2021$log_return)
sd_b <- sd(boot(data_2021$log_return, skewFunc, 1e4)$t)
k <- kurtosis(data_2021$log_return)
sd_k <- sd(boot(data_2021$log_return, kurtFunc, 1e4)$t)
res <- rbind(c(r, s, b, k), c(sd_r, sd_s, sd_b, sd_k))
round(res, 4)

# SF.2: returns are not correlated
ggplot.corr(data = data_2021$log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)

# SF.3: volatility is clustered and persistent
ggplot.corr(data = data_2021$abs_log_return, lag.max = 24, ci= 0.75,
            large.sample.size = FALSE, horizontal = TRUE)


### Question 3

# Normal distribution
# Cauchy
# Stable distribution

library(MASS)
library(StableEstim)
library(stabledist)

# 2019
normal_parameters_2019 <- fitdistr(data_2019$log_return, "normal")$estimate
cauchy_parameters_2019 <- fitdistr(data_2019$log_return, "cauchy")$estimate
stable_parameters_2019 <- McCullochParametersEstim(data_2019$log_return)

hist_density_2019 <- ggplot(data_2019, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  stat_function(fun = dnorm, args = list(mean = normal_parameters_2019["mean"],
                                         sd = normal_parameters_2019["sd"]),
                colour="red") +
  stat_function(fun = dcauchy, args = list(location = cauchy_parameters_2019["location"],
                                           scale = cauchy_parameters_2019["scale"]),
                colour="chartreuse3") +
  stat_function(fun = dstable, args = list(alpha = stable_parameters_2019["alpha"],
                                           beta = stable_parameters_2019["beta"],
                                           gamma = stable_parameters_2019["gamma"],
```

```
                                        delta = stable_parameters_2019["delta"]),
                colour="darkgoldenrod1") +
  labs(x="Return (%)", y="Density")


ecdf_2019 <- ggplot(data_2019, aes(log_return)) +
  stat_ecdf(geom = "step", colour="blue") +
  stat_function(fun = pnorm, args = list(mean = normal_parameters_2019["mean"],
                                         sd = normal_parameters_2019["sd"]),
                aes(colour="Normal")) +
  stat_function(fun = pcauchy, args = list(location = cauchy_parameters_2019["location"],
                                           scale = cauchy_parameters_2019["scale"]),
                aes(colour="Cauchy")) +
  stat_function(fun = pstable, args = list(alpha = stable_parameters_2019["alpha"],
                                           beta = stable_parameters_2019["beta"],
                                           gamma = stable_parameters_2019["gamma"],
                                           delta = stable_parameters_2019["delta"]),
                aes(colour="Stable")) +
  scale_colour_manual(name="Legend",
                      values=c(Normal="red", Cauchy="chartreuse3", Stable ="darkgoldenrod1"),
                      guide = guide_legend(override.aes = list(
                        linetype = c("solid", "solid", "solid"),
                        shape = c(NA, NA, NA)))) +
  labs(x="Return (%)", y="P[X < x]") +
  theme(legend.position=c(.2,.7))

ggarrange(hist_density_2019, ecdf_2019,
          ncol = 2, nrow = 1)

results_ks_2019 <- matrix(nrow=3, ncol=2, dimnames=list(c("Normal", "Cauchy", "Stable"),
                                                        c("D", "p-value")))
ks_norm_2019 <- ks.test(data_2019$log_return, "pnorm",
                        normal_parameters_2019[1],
                        normal_parameters_2019[2])
results_ks_2019[1, 1] <- ks_norm_2019$statistic
results_ks_2019[1, 2] <- ks_norm_2019$p.value
ks_cauchy_2019 <- ks.test(data_2019$log_return, "pcauchy",
                        cauchy_parameters_2019[1],
                        cauchy_parameters_2019[2])
results_ks_2019[2, 1] <- ks_cauchy_2019$statistic
results_ks_2019[2, 2] <- ks_cauchy_2019$p.value
ks_stable_2019 <- ks.test(data_2019$log_return, "pstable",
                          stable_parameters_2019[1],
                          stable_parameters_2019[2],
                          stable_parameters_2019[3],
                          stable_parameters_2019[4])
results_ks_2019[3, 1] <- ks_stable_2019$statistic
results_ks_2019[3, 2] <- ks_stable_2019$p.value


# 2020
normal_parameters_2020 <- fitdistr(data_2020$log_return, "normal")$estimate
cauchy_parameters_2020 <- fitdistr(data_2020$log_return, "cauchy")$estimate
stable_parameters_2020 <- McCullochParametersEstim(data_2020$log_return)

hist_density_2020 <- ggplot(data_2020, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
```

```
  stat_function(fun = dnorm, args = list(mean = normal_parameters_2020["mean"],
                                         sd = normal_parameters_2020["sd"]),
                colour="red") +
  stat_function(fun = dcauchy, args = list(location = cauchy_parameters_2020["location"],
                                           scale = cauchy_parameters_2020["scale"]),
                colour="chartreuse3") +
  stat_function(fun = dstable, args = list(alpha = stable_parameters_2020["alpha"],
                                           beta = stable_parameters_2020["beta"],
                                           gamma = stable_parameters_2020["gamma"],
                                           delta = stable_parameters_2020["delta"]),
                colour="darkgoldenrod1") +
  labs(x="Return (%)", y="Density")


ecdf_2020 <- ggplot(data_2020, aes(log_return)) +
  stat_ecdf(geom = "step", colour="blue") +
  stat_function(fun = pnorm, args = list(mean = normal_parameters_2020["mean"],
                                         sd = normal_parameters_2020["sd"]),
                aes(colour="Normal")) +
  stat_function(fun = pcauchy, args = list(location = cauchy_parameters_2020["location"],
                                           scale = cauchy_parameters_2020["scale"]),
                aes(colour="Cauchy")) +
  stat_function(fun = pstable, args = list(alpha = stable_parameters_2020["alpha"],
                                           beta = stable_parameters_2020["beta"],
                                           gamma = stable_parameters_2020["gamma"],
                                           delta = stable_parameters_2020["delta"]),
                aes(colour="Stable")) +
  scale_colour_manual(name="Legend",
                      values=c(Normal="red", Cauchy="chartreuse3", Stable ="darkgoldenrod1"),
                      guide = guide_legend(override.aes = list(
                        linetype = c("solid", "solid", "solid"),
                        shape = c(NA, NA, NA)))) +
  labs(x="Return (%)", y="P[X < x]") +
  theme(legend.position=c(.2,.7))

ggarrange(hist_density_2020, ecdf_2020,
          ncol = 2, nrow = 1)

results_ks_2020 <- matrix(nrow=3, ncol=2, dimnames=list(c("Normal", "Cauchy", "Stable"),
                                                        c("D", "p-value")))
ks_norm_2020 <- ks.test(data_2020$log_return, "pnorm",
                        normal_parameters_2020[1],
                        normal_parameters_2020[2])
results_ks_2020[1, 1] <- ks_norm_2020$statistic
results_ks_2020[1, 2] <- ks_norm_2020$p.value
ks_cauchy_2020 <- ks.test(data_2020$log_return, "pcauchy",
                          cauchy_parameters_2020[1],
                          cauchy_parameters_2020[2])
results_ks_2020[2, 1] <- ks_cauchy_2020$statistic
results_ks_2020[2, 2] <- ks_cauchy_2020$p.value
ks_stable_2020 <- ks.test(data_2020$log_return, "pstable",
                          stable_parameters_2020[1],
                          stable_parameters_2020[2],
                          stable_parameters_2020[3],
                          stable_parameters_2020[4])
results_ks_2020[3, 1] <- ks_stable_2020$statistic
results_ks_2020[3, 2] <- ks_stable_2020$p.value
```

```
# 2021
normal_parameters_2021 <- fitdistr(data_2021$log_return, "normal")$estimate
cauchy_parameters_2021 <- fitdistr(data_2021$log_return, "cauchy")$estimate
stable_parameters_2021 <- McCullochParametersEstim(data_2021$log_return)

hist_density_2021 <- ggplot(data_2021, aes(x = log_return)) +
  geom_histogram(aes(y = ..density..), bins=30, fill="blue", alpha=0.4,
                 colour="royalblue1", lwd=.2) +
  stat_function(fun = dnorm, args = list(mean = normal_parameters_2021["mean"],
                                         sd = normal_parameters_2021["sd"]),
                colour="red") +
  stat_function(fun = dcauchy, args = list(location = cauchy_parameters_2021["location"],
                                           scale = cauchy_parameters_2021["scale"]),
                colour="chartreuse3") +
  stat_function(fun = dstable, args = list(alpha = stable_parameters_2021["alpha"],
                                           beta = stable_parameters_2021["beta"],
                                           gamma = stable_parameters_2021["gamma"],
                                           delta = stable_parameters_2021["delta"]),
                colour="darkgoldenrod1") +
  labs(x="Return (%)", y="Density")


ecdf_2021 <- ggplot(data_2021, aes(log_return)) +
  stat_ecdf(geom = "step", colour="blue") +
  stat_function(fun = pnorm, args = list(mean = normal_parameters_2021["mean"],
                                         sd = normal_parameters_2021["sd"]),
                aes(colour="Normal")) +
  stat_function(fun = pcauchy, args = list(location = cauchy_parameters_2021["location"],
                                           scale = cauchy_parameters_2021["scale"]),
                aes(colour="Cauchy")) +
  stat_function(fun = pstable, args = list(alpha = stable_parameters_2021["alpha"],
                                           beta = stable_parameters_2021["beta"],
                                           gamma = stable_parameters_2021["gamma"],
                                           delta = stable_parameters_2021["delta"]),
                aes(colour="Stable")) +
  scale_colour_manual(name="Legend",
                      values=c(Normal="red", Cauchy="chartreuse3", Stable ="darkgoldenrod1"),
                      guide = guide_legend(override.aes = list(
                        linetype = c("solid", "solid", "solid"),
                        shape = c(NA, NA, NA)))) +
  labs(x="Return (%)", y="P[X < x]") +
  theme(legend.position=c(.2,.7))

ggarrange(hist_density_2021, ecdf_2021,
          ncol = 2, nrow = 1)

results_ks_2021 <- matrix(nrow=3, ncol=2, dimnames=list(c("Normal", "Cauchy", "Stable"),
                                                        c("D", "p-value")))
ks_norm_2021 <- ks.test(data_2021$log_return, "pnorm",
                        normal_parameters_2021[1],
                        normal_parameters_2021[2])
results_ks_2021[1, 1] <- ks_norm_2021$statistic
results_ks_2021[1, 2] <- ks_norm_2021$p.value
ks_cauchy_2021 <- ks.test(data_2021$log_return, "pcauchy",
                          cauchy_parameters_2021[1],
                          cauchy_parameters_2021[2])
results_ks_2021[2, 1] <- ks_cauchy_2021$statistic
```

```
results_ks_2021[2, 2] <- ks_cauchy_2021$p.value
ks_stable_2021 <- ks.test(data_2021$log_return, "pstable",
                          stable_parameters_2021[1],
                          stable_parameters_2021[2],
                          stable_parameters_2021[3],
                          stable_parameters_2021[4])
results_ks_2021[3, 1] <- ks_stable_2021$statistic
results_ks_2021[3, 2] <- ks_stable_2021$p.value
```