Introduction to statistical finance

Coursework 1 - Part B

Imperial College London Spring 2020

Abstract

Work realised by Victoire Burg (CID : 01936649), Beatrice Matteo (CID: 01899224), Paul Couturier (CID : 01922090) and Robin Mathelier (CID : 01945214) for the Coursework 1 of Introduction to statistical finance.

1 Context

To provide a broad spectrum of distribution fitting techniques, we have been searching for three different assets that would have completely different behaviours. We finally agreed on studying :

- CAC40 : the main index of Euronext Paris, that gives an indication of the stock exchange price and which is computed using the stock value of the 40 biggest listed French companies.
- BitCoin : is one of the first worldwide used cryptocurrency that would have been created by Satoshi Nakamoto, a Japanese of 40 year-old.
- Alibaba stock prices on the New York Stock exchange. Alibaba is a Chinese company specialised in e-commerce, retail, Internet, and technology. ALIBABA is interesting since it's a big company without being considered as a GAFAM. Notice : Alibaba is also listed on the Honk Kong stock exchange but we won't take this price into account.

We obtain a very diverse sample of prices series with one index, one cryptocurrency and the stock price of a huge Chinese company. Using the Yahoo Finance API through the library tidyquant we were able to retrieve the prices and plot the Figure 1.

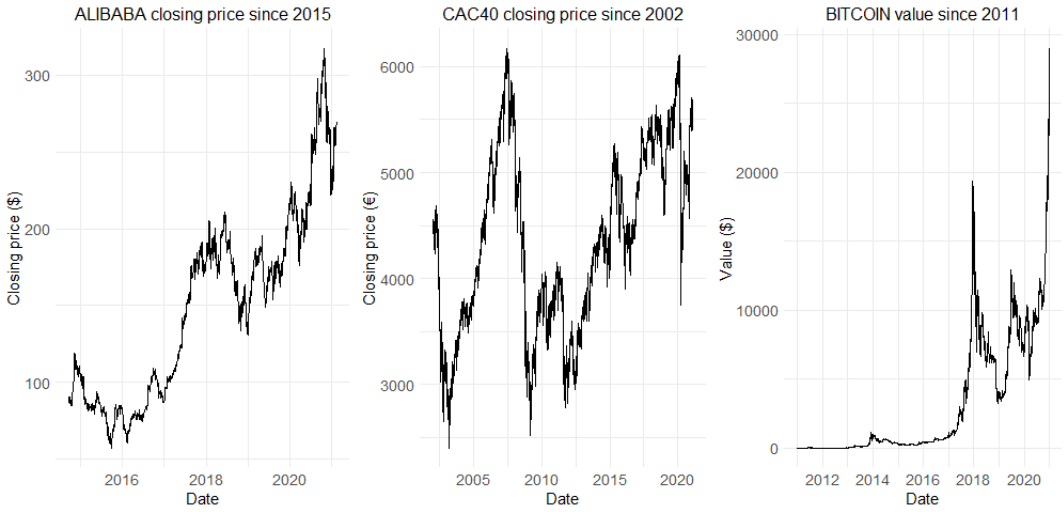

Figure 1: Price time series for the three assets

Let's explain what we can see on the Figure 1 of the price times series.

ALIBABA

In December some rumours have been spread about the possible arrest of the founder of the company, Jack Ma, after he criticised the Chinese government in an interview. This explains the drop in the stock prices that we can spot on the graph. Finally, Jack Ma has reappeared at the beginning of January explaining the rise since the beginning of 2021.

CAC40

The CAC40 is a very important index for Euronext Paris and so completely reveals the trend of the Paris stock exchange. We have selected it as it will show the 2007 crisis. Moreover, the index has been massively

impacted by the Covid crisis at the beginning of 2020.

BITCOIN
Finally for the bitcoin, we can see the spectacular peak that it has known in 2017. In addition, the Covid crisis in 2020 has vastly shook up the market, and investor have secured their position betting on the bitcoin.

2 Log returns

Once we had the closing price, we can derive the log return prices using the following formula :

$$r_t = log(s_t) - log(s_{t-1}) \tag{2.1}$$

We obtain the Figure 2.

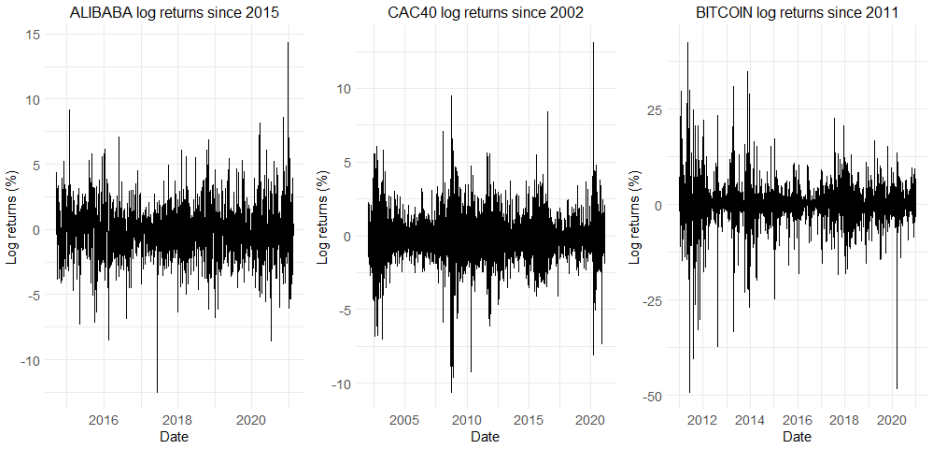

Figure 2: Plot of the log returns for the three assets

In this plot we can see what seems to be a logical log return series. However, for Alibaba series the volatility seems less clustered. We will describe this in more details in the section SF3.

3 Analyse of the stylised facts (SF1-SF3)

3.1 SF1 : distribution non-normal with heavy tails and possibly mild asymmetry

The unconditional distribution of returns (at a daily or shorter time scale) is markedly non-normal with heavy tails and possibly mild asymmetry.
To check that, we compare a histogram plus the estimated kernel density plot of the observed returns for each series with a normal distribution fitted to the data. The figure 2 indicates that, for, for the three assets, the fitted normal distribution describes very poorly the return distribution, whose peak is much more acute than the one of a normal distribution. The tails of the distributions are hard to distinguish, so we will look at the QQ-plots on figure 3. The QQ-plots suggest with the curve that the tails of the return distribution are heavier than those of the normal distribution.
We decide to analyse the distribution of the returns $r_1, ..., r_T$ more quantitatively using *skewness* and *kurtosis*. The *skewness* of the returns represented by the random variable R is defined by $\beta = \frac{\mathbb{E}[(R-\mathbb{E}[R])^3]}{\mathbb{E}[(R-\mathbb{E}[R])^2]^{\frac{3}{2}}}$ which is estimated by the sample skewness b, $b = \frac{\frac{1}{T}\sum_{t=1}^{T}(r_T-\bar{r})^3}{(\frac{1}{T}\sum_{t=1}^{T}(r_T-\bar{r})^2)^{\frac{2}{3}}}$, where $\bar{r} = \frac{1}{T}\sum_{t=1}^{T} r_t$ is the sample mean.
For a single peaked distribution $\beta < 0$ and it is a measure of the asymmetry of the distribution.
The *kurtosis* of the returns R is a measure of *peakedness* and is defined by $\kappa = \frac{\mathbb{E}[(R-\mathbb{E}[R])^4]}{\mathbb{E}[(R-\mathbb{E}[R])^2]^2}$ which is estimated by the sample *kurtosis* k, $k = \frac{\frac{1}{T}\sum_{t=1}^{T}(r_T-\bar{r})^4}{(\frac{1}{T}\sum_{t=1}^{T}(r_T-\bar{r})^2)^2}$.
To estimate the *kurtosis* and the *skewness* of the different distributions, we do a bootstrap re-sampling with 5 000 samples.
We find values which are very different from the ones obtained for the normal distribution, which shows that the returns are non-normal.

	k	k_{norm}	b	b_{norm}
CAC40	10.06	3	0.20	0
ALIBABA	6.04	3	0.17	0
BITCOIN	17.24	2.54	-0.49	0.08

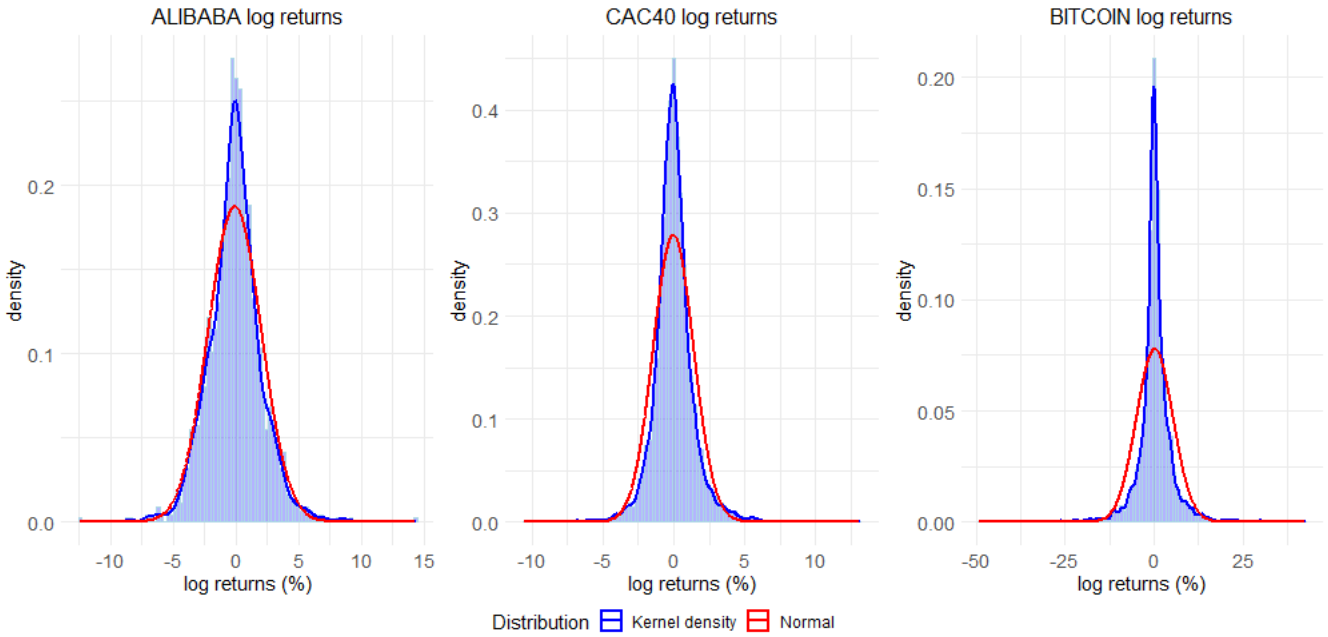

Figure 3: A histogram an a kernel density estimate of the daily returns of the three assets with a fitted normal distribution

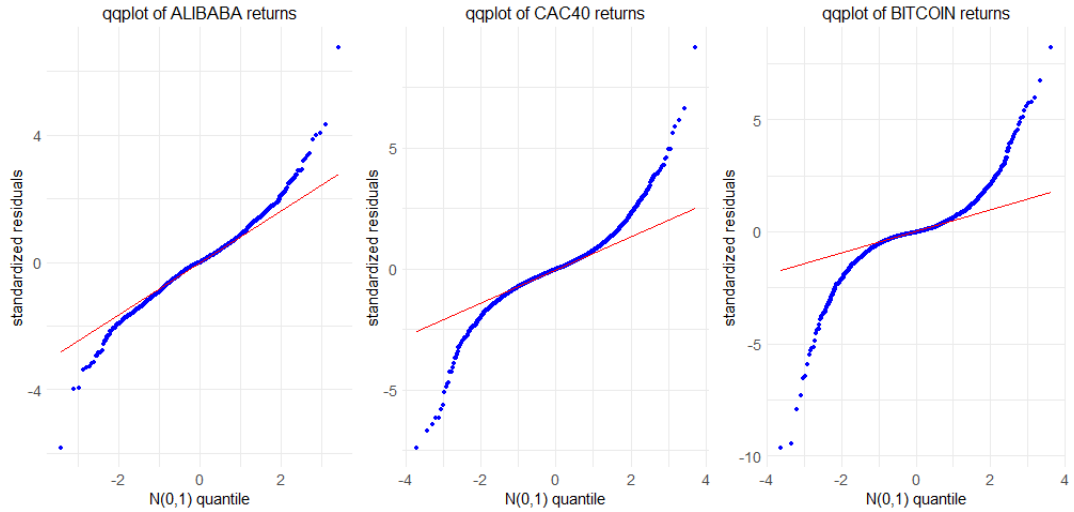

Figure 4: A histogram an a kernel density estimate of the daily returns of the three assets with a normal QQ-plot

3.2 SF2 Returns are serially uncorrelated

We have plotted the ACF for returns in the Figure 5. We can see the returns are uncorrelated since the ACF graph shows no significant auto-correlation for any lag. This is completely coherent since an autocorrelation would assess a possibility to predict the direction of the future variation. This would be quite unlikely.

3.2.1 SF3 Volatility is clustered and persistent

We plotted the ACF of the absolute returns in the Figure 6. We can see that for the CAC40 and the Bitcoin the absolute return is autocorrelated, which is perfectly logical. Indeed, we can't predict the direction of a change but still over a period of time the variation (which is the absolute return) is persistent. However,

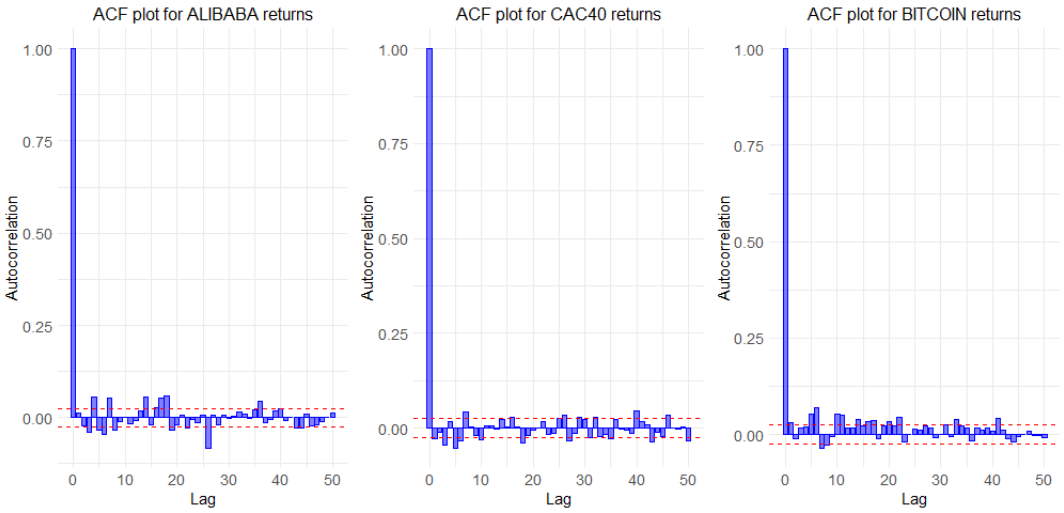

Figure 5: Empirical autocorrelation functions of the daily returns

for Alibaba, the persistence is less visible. This can be explained by the permanent great variation that the company has known since its IPO. Indeed as explained in this article [5], the company has suffered from tension between the Trump administration and the Chinese government for the past 4 years.

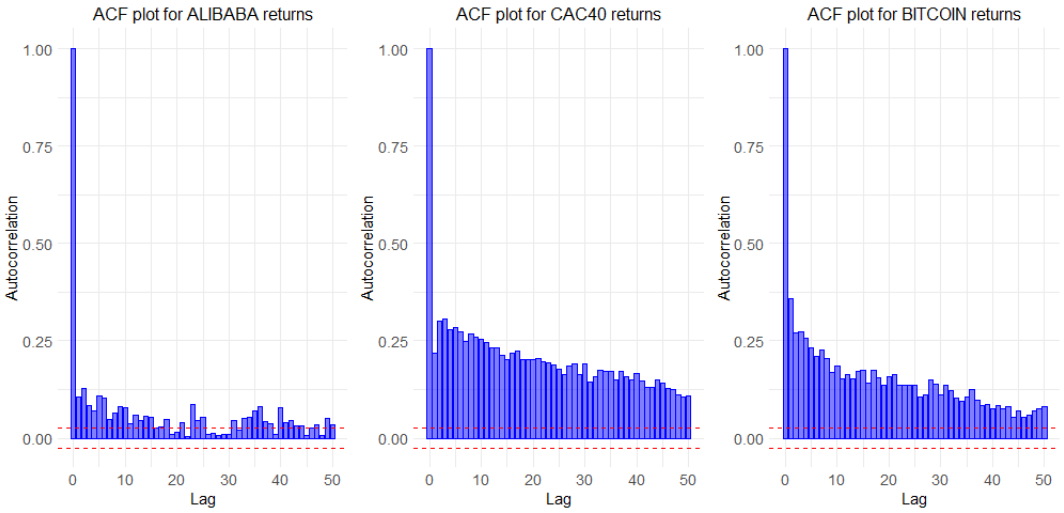

Figure 6: Empirical autocorrelation functions of the daily absolute returns

4 Examples of unconditional distribution of returns

As shown in the SF1 part, the gaussian distribution has important drawbacks when used to fit the returns. So we have been looking for various scientific papers studying different distribution to fit log return curves.

4.1 Student distribution

First we thought about the student distribution. Indeed, it has the advantage to have heavier tails than the normal distribution. The center student distribution has the following density :

$$f_T(x) = \frac{1}{\sqrt{k\pi}} \frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})} (1 + \frac{t^2}{k})^{-\frac{k+1}{2}}$$

By using a non centered normal law to construct our student distribution we can also kind of fit the skewness.

4.2 Generalized lambda distribution

We have found a very interesting article about the use of the Generalised lambda distribution (GLD) for fitting financial returns - see Article [2]. The GLD is a distribution that is only defined by its quantile function. It is parametrised by 4 values. Thus, it has sufficient degree of freedom to fit prefectly the mean, the variance,the skewness and the kurtosis of the data. Using the FKML parametrisation (others exist) we obtain the following quantile function :

$$Q_{FKML}(u) = \lambda_1 + \frac{1}{\lambda_2}\left[\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}\right]$$

In many cases its density function doesn't have an analytical form and need to be computed.

4.3 Stable distribution

Finally we found a paper that would compare the Student distribution with another one called the Stable distribution - see Articles [1] [3]. Similarly to the GLD, this distribution doesn't have in general an analytical form for its density. However, it can be perfectly defined by it's characteristic function which is :

$$\phi(t,\alpha,\beta,\gamma,\delta) = exp(it\delta - |\gamma t|^{\alpha}(1 - i\beta sgn(t)\Phi))$$

where

$$\Phi = \begin{cases} (|\gamma t|^{\alpha-1} - 1)tan(\frac{\pi\alpha}{2}) & if\ \alpha \neq 1 \\ \frac{2}{\pi}log(|\gamma t|) & if\ \alpha = 1 \end{cases}$$

This distribution has also 4 parameters like the GLD. It is a very used distribution since it's permit to generalized the central limit theorem to variable without finite variance. This seems a pretty nice property to fit returns with possibly heavy tails.

Fitting

We decide to fit the 4 models previously described: normal distribution, student distribution, stable distribution and generalized lambda distribution (gld).
To fit the student, the normal and the stable distribution we used a method of Maximum Likelihood as it is often the most common used technique.
However, to estimate the parameters of the gld distribution, we have chosen the method of moments which was in this case fitting better than with the maximum likelihood method. Moreover, it has the great advantage to optimally provide the mean, the variance,the skewness and the kurtosis. We used the special package GLDEX developped in R as shown in the article [4].

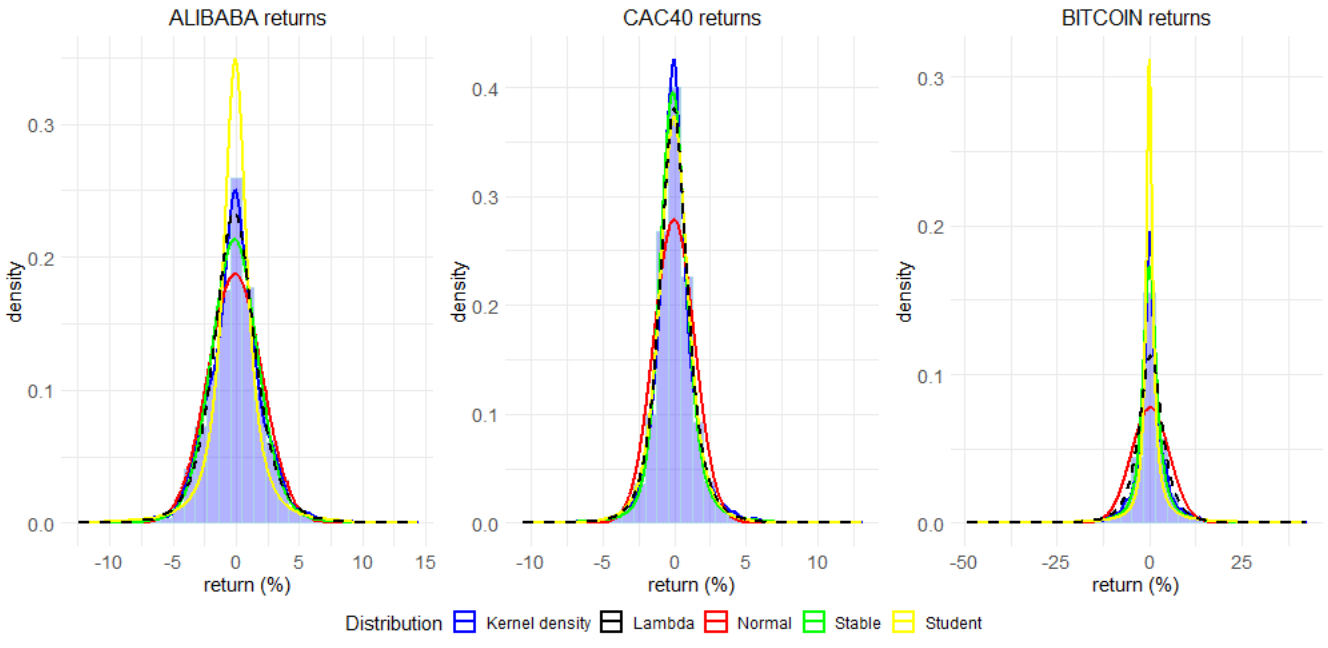

Figure 7: Graph of the 4 fitted distribution to the three assets

Goodness of fit

So we need to find distribution that fit better our empirical distribution.
In addition to the graph 7, and to see numerically the similarities between the data and the possible distribution, we compare the *kurtosis* and the *skewness* (see table below). To compute the values for the distributions Student stable normal, we use a bootstrap with a 5 000 re-sampling. However, due to possible infinite value, we had to truncate the support. For the GLD we use again the GLDEX package with the function "fun.theo.mv.gld".

	k_{data}	k_{Normal}	$k_{Student}$	k_{Stable}	k_{GLD}	s_{data}	s_{Normal}	$s_{Student}$	s_{stable}	s_{GLD}
CAC40	10.06	3.00	8.16	13.42	10.06	0.20	-0.00	0.06	0.71	0.20
ALIBABA	6.04	3.00	9.43	5.20	6.04	0.17	-0.00	0.28	0.21	0.17
BITCOIN	17.24	2.54	7.38	5.00	17.24	-0.49	0.08	0.57	0.33	-0.49

On one hand, for the CAC40 and the alibaba data, the values are coherent. On the other hand, we have the BitCoin data which presents *kurtosis* and *skewness* values anormally high. By looking at the plot 1, we may assume that those unsual values are explained by the return pic at -48. In this regard, the distribution should take into account several peaks. That's why, a gaussian mixture (assumes that all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters) or a student mixture could be undertaken. Then, the results obtained could fit better to the situation. Moreover, we can see that the GLD that fit badly CAC40 and the Bitcoin data, actually is able to fit perfectly the skewness and the kurtosis. This is because we used the method of the moment to fit the data for this distribution. Since it has 4 degree of freedom it has been able to fit perfectly the variance, the mean, the skewness and the kurtosis. However we can that doesn't mean that it is always providing a good fit in terms of curve as we see on the Figure 7.
The previous results suggest that the distributions fit better to the CAC40 and the alibaba data.

Finally, to assess the goodness of fit of the different distributions we look at the p-values of the Kolmogorov–Smirnov test which test if the two samples are obtained from the same distribution. It compares the cumulative distributions of two data sets. The null hypothesis is that both groups were sampled from populations with identical distributions. It tests for any violation of that null hypothesis: different medians, different variances, or different distributions. The R function ks.test allows to perform that test for the different distributions. It gives the following p_{values}:

p_{values}	Normal distribution	Student distribution	Stable distribution	GLD
CAC40	1.7E-14	1.2E-03	2.4E-01	7.9E-04
ALIBABA	7.0E-04	3.8E-07	5.1E-02	2.3E-01
BITCOIN	4.9E-14	4.9E-14	1.9E-02	4.9E-14

It reveals that for both the CAC40 data, the Stable distribution fits the better, since the value is above 5%. For the Bitcoin the stable remains also the best distribution even with a pvalue of 0.02 this distribution would still be rejected in many situation. For the Alibaba data, the GLD fits better but this distribution fits very badly for the other data. We see that depending on which type of asset we are looking for, we need to find adapted distributions in order to obtain good fitting.

References

[1] Robert C. Blattberg and Nicholas J. Gonedes. A comparison of the stable and student distributions as statistical models for stock prices. *The University of Chicago Press.*

[2] Yohan Chalabi, David J Scott, and Diethelm Wuertz. Flexible distribution modeling with the generalized lambda distribution.

[3] B. Mandelbrot. New methods in statistical economics. *Journal of Political Economy.*

[4] Joseph Rickert. The generalized lambda distribution and gldex package: Fitting financial return data, 2014.

[5] Jovan Lee Yuheng. Alibaba: A fundamentally strong company unfairly beaten down by the market.

A R code

```
##### Group Coursework - Part B #######

### Victoire Burg, Beatrice Matteo, Paul Couturier and Robin Mathelier ###

library(ggplot2)
library(ggpubr)
library(dplyr)
library(tidyquant)
library(zoo) ## for data processing
library(tidyverse) ## for data processing
library(jsonlite)
library(moments)
library(stabledist)
library(fBasics)
library(MASS)
library(VaRES)
library(e1071)
library(xtable)
library(gridExtra)
library(GLDEX)

my_theme <- theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.title = element_text(size = 15),
        axis.text = element_text(size = 14),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 12),
        strip.text = element_text(size = 14))

#setwd("C:/Users/robin/Dropbox/Applications/Overleaf/Coursework finance part B")

##### read CAC40 #####

CAC40 = read.csv(file = "C:/Users/paulc/Desktop/Imperial College/Spring term/
    MATH97132 - Introduction to Statistical Finance 2020-2021/Coursework/CAC40.csv")
CAC40$Date = as.Date(CAC40$Date)
CAC40$Close = as.numeric(CAC40$Close)

CAC40 = na.omit(CAC40)

Close_prices = CAC40$Close
CAC40 = CAC40[-1,]
CAC40$return = log(Close_prices[-length(Close_prices)]) - log(Close_prices[-1])
CAC40$return_100 = CAC40$return * 100

##### read bitcoin #####

BITCOIN_JSON <- fromJSON("https://api.coindesk.com/v1/bpi/historical/close.json?start
    =2011-01-01&end=2020-12-31")

dates <- as.Date(names(BITCOIN_JSON[[1]]))
number_days <- length(dates)
list_days_elapsed <- 1:(number_days-1)
list_bpi <<- c()
sapply(1:(number_days), function(i){list_bpi[i] <<- BITCOIN_JSON$bpi[[i]]})
return <- sapply(1:(number_days-1), function(i){log(list_bpi[i+1])-log(list_bpi[i])})

BITCOIN_TO_PLOT <- data.frame(Date = dates, value = list_bpi)
```

```
BITCOIN <- data.frame(number_days_elapsed = (1:(number_days-1)) ,Date = dates[-1],
    return = return)
BITCOIN$return_100 = BITCOIN$return * 100

##### read Ali Baba ####

ALIBABA <- tq_get("BABA", get = "stock.prices", from ="1990-01-01")

ALIBABA = na.omit(ALIBABA)
Close_prices = ALIBABA$close
ALIBABA = ALIBABA[-1,]
ALIBABA$return = log(Close_prices[-length(Close_prices)]) - log(Close_prices[-1])
ALIBABA$return_100 = ALIBABA$return * 100







##### plot time series ######

png(filename = "plot_time_series.png", width = 1000)

g_BITCOIN = ggplot(data = BITCOIN_TO_PLOT, aes(x = Date, y = value))
g_BITCOIN = g_BITCOIN + geom_line() + my_theme
g_BITCOIN <- g_BITCOIN + labs(x = "Date", y = "Value ($)", title = "BITCOIN value
    since 2011")
g_BITCOIN

g_CAC40 = ggplot(data = CAC40, aes(x = Date, y = Close))
g_CAC40 = g_CAC40 + geom_line() + my_theme
g_CAC40 = g_CAC40 + labs(x = "Date", y = "Closing price (???)", title = "CAC40
    closing price since 2002")
g_CAC40

g_ALIBABA = ggplot(data = ALIBABA, aes(x = date, y = close))
g_ALIBABA = g_ALIBABA + geom_line() + my_theme
g_ALIBABA = g_ALIBABA + labs(x = "Date", y = "Closing price ($)", title = "ALIBABA
    closing price since 2015")
g_ALIBABA

grid.arrange(g_ALIBABA,g_CAC40,g_BITCOIN,ncol = 3)

dev.off()

##### Question (i) - plot log returns #####

png(filename = "plot_log_returns.png", width = 1000)

g_BITCOIN <- ggplot(data=BITCOIN)
g_BITCOIN <- g_BITCOIN + geom_line(aes(x= Date, y=return_100),size=0.5)
g_BITCOIN <- g_BITCOIN + labs(x = "Date",y = "Log returns (%)",title = "BITCOIN log
    returns since 2011")
g_BITCOIN <- g_BITCOIN + my_theme

g_CAC40 <- ggplot(data=CAC40)
g_CAC40 <- g_CAC40 + geom_line(aes(x= Date, y=return_100),size=0.5)
g_CAC40 <- g_CAC40 + labs(x = "Date",y = "Log returns (%)",title = "CAC40 log returns
    since 2002")
g_CAC40 <- g_CAC40 + my_theme

g_ALIBABA <- ggplot(data=ALIBABA)
g_ALIBABA <- g_ALIBABA + geom_line(aes(x= date, y=return_100),size=0.5)
```

```
g_ALIBABA <- g_ALIBABA + labs(x = "Date",y = "Log returns (%)",title = "ALIBABA log
    returns since 2015")
g_ALIBABA <- g_ALIBABA + my_theme

grid.arrange(g_ALIBABA,g_CAC40,g_BITCOIN,ncol = 3)

dev.off()

############ SF 1 #############

mu_ALIBABA = mean(ALIBABA$return_100)
sigma_ALIBABA = sd(ALIBABA$return_100)
ALIBABA$normal = dnorm(ALIBABA$return_100, mean = mu_ALIBABA, sd = sigma_ALIBABA)

mu_CAC40 = mean(CAC40$return_100)
sigma_CAC40 = sd(CAC40$return_100)
CAC40$normal = dnorm(CAC40$return_100, mean = mu_CAC40, sd = sigma_CAC40)

mu_BITCOIN = mean(BITCOIN$return_100)
sigma_BITCOIN = sd(BITCOIN$return_100)
BITCOIN$normal = dnorm(BITCOIN$return_100, mean = mu_BITCOIN, sd = sigma_BITCOIN)

##### plot with normal fitted ######

png(filename = "log_returns_normal.png", width = 1000)

g_CAC40 = ggplot(data = CAC40)
g_CAC40 = g_CAC40 + geom_histogram(aes(x = return_100, y = ..density..), bins = 100,
    alpha = 0.3, fill = "blue", color = 'lightblue')
g_CAC40 = g_CAC40 + geom_density(aes(x = return_100, color = "Kernel density"), alpha
    = 0.8, lwd = 0.8)
g_CAC40 = g_CAC40 + geom_line(aes(x = return_100, y = normal, color = "Normal"), lwd
    = 0.8)
g_CAC40 = g_CAC40 + my_theme
g_CAC40 = g_CAC40 + scale_color_manual(name = "Distribution", values = c(
  "Kernel density" = "blue",
  'Normal' = 'red'))
g_CAC40 = g_CAC40 + labs(x = "log returns (%)", title = "CAC40 log returns")

g_ALIBABA = ggplot(data = ALIBABA)
g_ALIBABA = g_ALIBABA + geom_histogram(aes(x = return_100, y = ..density..), bins =
    100, alpha = 0.3, fill = "blue", color = 'lightblue')
g_ALIBABA = g_ALIBABA + geom_density(aes(x = return_100, color = "Kernel density"),
    alpha = 0.8, lwd = 0.8)
g_ALIBABA = g_ALIBABA + geom_line(aes(x = return_100, y = normal, color = "Normal"),
    lwd = 0.8)
g_ALIBABA = g_ALIBABA + my_theme
g_ALIBABA = g_ALIBABA + scale_color_manual(name = "Distribution", values = c(
  "Kernel density" = "blue",
  'Normal' = 'red'))
g_ALIBABA = g_ALIBABA + labs(x = "log returns (%)", title = "ALIBABA log returns")

g_BITCOIN = ggplot(data = BITCOIN)
g_BITCOIN = g_BITCOIN + geom_histogram(aes(x = return_100, y = ..density..), bins =
    100, alpha = 0.3, fill = "blue", color = 'lightblue')
g_BITCOIN = g_BITCOIN + geom_density(aes(x = return_100, color = "Kernel density"),
    alpha = 0.8, lwd = 0.8)
g_BITCOIN = g_BITCOIN + geom_line(aes(x = return_100, y = normal, color = "Normal"),
    lwd = 0.8)
g_BITCOIN = g_BITCOIN + my_theme
g_BITCOIN = g_BITCOIN + scale_color_manual(name = "Distribution", values = c(
  "Kernel density" = "blue",
  'Normal' = 'red'))
g_BITCOIN = g_BITCOIN + labs(x = "log returns (%)", title = "BITCOIN log returns")
```

```
ggarrange(g_ALIBABA, g_CAC40, g_BITCOIN, common.legend = TRUE, ncol = 3, legend = "
    bottom")

dev.off()

##### qqplot #####

png(filename = "qqplots.png", width = 1000)

g_CAC40 = ggplot(data = CAC40)
g_CAC40 = g_CAC40 + stat_qq(aes(sample = scale(return)), color = 'blue') + stat_qq_
    line(aes(sample = scale(return)), color = 'red')
g_CAC40 = g_CAC40 + my_theme
g_CAC40 = g_CAC40 + labs(y = "standardized residuals", x = "N(0,1) quantile", title =
     "qqplot of CAC40 returns")

g_ALIBABA = ggplot(data = ALIBABA)
g_ALIBABA = g_ALIBABA + stat_qq(aes(sample = scale(return)), color = 'blue') + stat_
    qq_line(aes(sample = scale(return)), color = 'red')
g_ALIBABA = g_ALIBABA + my_theme
g_ALIBABA = g_ALIBABA + labs(y = "standardized residuals", x = "N(0,1) quantile",
    title = "qqplot of ALIBABA returns")

g_BITCOIN = ggplot(data = BITCOIN)
g_BITCOIN = g_BITCOIN + stat_qq(aes(sample = scale(return)), color = 'blue') + stat_
    qq_line(aes(sample = scale(return)), color = 'red')
g_BITCOIN = g_BITCOIN + my_theme
g_BITCOIN = g_BITCOIN + labs(y = "standardized residuals", x = "N(0,1) quantile",
    title = "qqplot of BITCOIN returns")

grid.arrange(g_ALIBABA, g_CAC40, g_BITCOIN, ncol = 3)

dev.off()

###### kurtosis and skewness ######


k_CAC40 = kurtosis(CAC40$return_100, method = "moment")
s_CAC40 = skewness(CAC40$return_100)
k_CAC40_normal = kurtosis(CAC40$normal, method = "moment")
s_CAC40_normal = skewness(CAC40$normal)

k_ALIBABA = kurtosis(ALIBABA$return_100, method = "moment")
s_ALIBABA = skewness(ALIBABA$return_100)
k_ALIBABA_normal = kurtosis(ALIBABA$normal, method = "moment")
s_ALIBABA_normal = skewness(ALIBABA$normal)

k_BITCOIN = kurtosis(BITCOIN$return_100, method = "moment")
s_BITCOIN = skewness(BITCOIN$return_100)
k_BITCOIN_normal = kurtosis(BITCOIN$normal, method = "moment")
s_BITCOIN_normal = skewness(BITCOIN$normal)

k = c(k_CAC40, k_ALIBABA, k_BITCOIN)
k_norm = c(k_CAC40_normal, k_ALIBABA_normal, k_BITCOIN_normal)

s = c(s_CAC40, s_ALIBABA, s_BITCOIN)
s_norm = c(s_CAC40_normal, s_ALIBABA_normal, s_BITCOIN_normal)

xtable(data.frame(k, k_norm, row.names = c("CAC40","ALIBABA","BITCOIN")))
xtable(data.frame(s, s_norm, row.names = c("CAC40","ALIBABA","BITCOIN")))

############ SF 2 ############
```

```
png(filename = "ACF_plots_SF2.png", width = 1000)

acf_CAC40 = acf(CAC40$return, plot = FALSE, lag = 50)$acf
df_acf_CAC40 = data.frame(abs = 0:(length(acf_CAC40)-1), acf_CAC40)
g_CAC40 = ggplot(data = df_acf_CAC40)
g_CAC40 = g_CAC40 + geom_bar(aes(x = abs, y = acf_CAC40), stat = 'identity', color =
    'blue', fill = 'blue', alpha = 0.5)
g_CAC40 = g_CAC40 + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_CAC40 = g_CAC40 + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_CAC40 = g_CAC40 + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "ACF
    plot for CAC40 returns")

acf_ALIBABA = acf(ALIBABA$return, plot = FALSE, lag = 50)$acf
df_acf_ALIBABA = data.frame(abs = 0:(length(acf_ALIBABA)-1), acf_ALIBABA)
g_ALIBABA = ggplot(data = df_acf_ALIBABA)
g_ALIBABA = g_ALIBABA + geom_bar(aes(x = abs, y = acf_ALIBABA), stat = 'identity',
    color = 'blue', fill = 'blue', alpha = 0.5)
g_ALIBABA = g_ALIBABA + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_ALIBABA = g_ALIBABA + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_ALIBABA = g_ALIBABA + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "
    ACF plot for ALIBABA returns")

acf_BITCOIN = acf(BITCOIN$return, plot = FALSE, lag = 50)$acf
df_acf_BITCOIN = data.frame(abs = 0:(length(acf_BITCOIN)-1), acf_BITCOIN)
g_BITCOIN = ggplot(data = df_acf_BITCOIN)
g_BITCOIN = g_BITCOIN + geom_bar(aes(x = abs, y = acf_BITCOIN), stat = 'identity',
    color = 'blue', fill = 'blue', alpha = 0.5)
g_BITCOIN = g_BITCOIN + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_BITCOIN = g_BITCOIN + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_BITCOIN = g_BITCOIN + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "
    ACF plot for BITCOIN returns")

grid.arrange(g_ALIBABA, g_CAC40, g_BITCOIN, ncol = 3)

dev.off()

########## SF3 ##########


png(filename = "ACF_plots_SF3.png", width = 1000)

acf_CAC40 = acf(abs(CAC40$return), plot = FALSE, lag = 50)$acf
df_acf_CAC40 = data.frame(abs = 0:(length(acf_CAC40)-1), acf_CAC40)
g_CAC40 = ggplot(data = df_acf_CAC40)
g_CAC40 = g_CAC40 + geom_bar(aes(x = abs, y = acf_CAC40), stat = 'identity', color =
    'blue', fill = 'blue', alpha = 0.5)
g_CAC40 = g_CAC40 + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_CAC40 = g_CAC40 + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_CAC40 = g_CAC40 + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "ACF
    plot for CAC40 returns")

acf_ALIBABA = acf(abs(ALIBABA$return), plot = FALSE, lag = 50)$acf
df_acf_ALIBABA = data.frame(abs = 0:(length(acf_ALIBABA)-1), acf_ALIBABA)
g_ALIBABA = ggplot(data = df_acf_ALIBABA)
g_ALIBABA = g_ALIBABA + geom_bar(aes(x = abs, y = acf_ALIBABA), stat = 'identity',
    color = 'blue', fill = 'blue', alpha = 0.5)
g_ALIBABA = g_ALIBABA + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_ALIBABA = g_ALIBABA + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_ALIBABA = g_ALIBABA + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "
    ACF plot for ALIBABA returns")

acf_BITCOIN = acf(abs(BITCOIN$return), plot = FALSE, lag = 50)$acf
df_acf_BITCOIN = data.frame(abs = 0:(length(acf_BITCOIN)-1), acf_BITCOIN)
g_BITCOIN = ggplot(data = df_acf_BITCOIN)
```

```
g_BITCOIN = g_BITCOIN + geom_bar(aes(x = abs, y = acf_BITCOIN), stat = 'identity',
    color = 'blue', fill = 'blue', alpha = 0.5)
g_BITCOIN = g_BITCOIN + geom_hline(yintercept = 0.025, color = 'red', lty = 2)
g_BITCOIN = g_BITCOIN + geom_hline(yintercept = -0.025, color = 'red', lty = 2)
g_BITCOIN = g_BITCOIN + my_theme + labs(x = "Lag", y = "Autocorrelation", title = "
    ACF_plot_for_BITCOIN_returns")

grid.arrange(g_ALIBABA, g_CAC40, g_BITCOIN, ncol = 3)

dev.off()


############ part (iii) ##############


######### Student ############


rt_CAC40 = fitdistr(CAC40$return_100, dt, start = list(df = 1, ncp = 0))
df_hat_CAC40 = rt_CAC40$estimate[[1]]
ncp_hat_CAC40 = rt_CAC40$estimate[[2]]
CAC40$t = dt(CAC40$return_100, df = df_hat_CAC40, ncp = ncp_hat_CAC40)

rt_BITCOIN = fitdistr(BITCOIN$return_100, dt, start = list(df = 1, ncp = 0))
df_hat_BITCOIN = rt_BITCOIN$estimate[[1]]
ncp_hat_BITCOIN = rt_BITCOIN$estimate[[2]]
BITCOIN$t = dt(BITCOIN$return_100, df = df_hat_BITCOIN, ncp = ncp_hat_BITCOIN)

rt_ALIBABA = fitdistr(ALIBABA$return_100, dt, start = list(df = 1, ncp = 0))
df_hat_ALIBABA = rt_ALIBABA$estimate[[1]]
ncp_hat_ALIBABA = rt_ALIBABA$estimate[[2]]
ALIBABA$t = dt(ALIBABA$return_100, df = df_hat_ALIBABA, ncp = ncp_hat_ALIBABA)

######### Stable ##########

f <- function(u,a,b,c,d) {
  cat(a,b,c,d,"\n")  # Some logging (it is very slow)
  dstable(CAC40$return_100, 2*exp(a)/(1+exp(a)), 2*exp(b)/(1+exp(b))-1, exp(c), d)
}

# r_stable_CAC40 <- fitdistr(CAC40$return_100, f, list(a=1, b=0, c=log(mad(CAC40$
    return_100)), d=median(CAC40$return_100)))
# r_stable_CAC40

### gives following values (very long to run)

a_CAC40 = 1.31111593
b_CAC40 = 0.31231726
c_CAC40 = -0.32470313
d_CAC40 = -0.07071168

CAC40$stable = f(CAC40$return_100, a = a_CAC40, b = b_CAC40, c = c_CAC40, d = d_CAC40
    )


f <- function(u,a,b,c,d) {
  cat(a,b,c,d,"\n")  # Some logging (it is very slow)
  dstable(BITCOIN$return_100, 2*exp(a)/(1+exp(a)), 2*exp(b)/(1+exp(b))-1, exp(c), d)
}

# r_stable_BITCOIN <- fitdistr(BITCOIN$return_100, f, list(a=1, b=0, c=log(mad(
    BITCOIN$return_100)), d=median(BITCOIN$return_100)))
# r_stable_BITCOIN
```

```
a_BITCOIN = 0.46938594
b_BITCOIN = 0.17962342
c_BITCOIN = 0.54323727
d_BITCOIN = 0.20952216

BITCOIN$stable = f(BITCOIN$return_100, a = a_BITCOIN, b = b_BITCOIN, c = c_BITCOIN, d
    = d_BITCOIN)

f <- function(u,a,b,c,d) {
  cat(a,b,c,d,"\n")  # Some logging (it is very slow)
  dstable(ALIBABA$return_100, 2*exp(a)/(1+exp(a)), 2*exp(b)/(1+exp(b))-1, exp(c), d)
}

# r_stable_ALIBABA = fitdistr(ALIBABA$return_100, f, list(a=1, b=0, c=log(mad(ALIBABA
    $return_100)), d=median(ALIBABA$return_100)))
# r_stable_ALIBABA


a_ALIBABA = 2.29140859
b_ALIBABA = 0.20163948
c_ALIBABA = 0.28143916
d_ALIBABA = -0.10266055

ALIBABA$stable = f(ALIBABA$return_100, a = a_ALIBABA, b = b_ALIBABA, c = c_ALIBABA, d
    = d_ALIBABA)


######### generalize lambda distribution ##########

logret_CAC40 = CAC40$return_100[-1]

#Method of moment
wshLambdaMM_CAC40 <- fun.RMFMKL.mm(logret_CAC40)

fittedVal_CAC40 = dgl(CAC40$return_100, lambda1 = wshLambdaMM_CAC40[1], lambda2 =
    wshLambdaMM_CAC40[2],
                      lambda3 = wshLambdaMM_CAC40[3], lambda4 = wshLambdaMM_CAC40[4],
                param = "fmkl", inverse.eps = 1e-08,
                max.iterations = 500)

CAC40$gld = fittedVal_CAC40


logret_BITCOIN = BITCOIN$return_100[-1]

#Method of moment
wshLambdaMM_BITCOIN <- fun.RMFMKL.mm(logret_BITCOIN)

fittedVal_BITCOIN = dgl(BITCOIN$return_100, lambda1 = wshLambdaMM_BITCOIN[1], lambda2
    = wshLambdaMM_BITCOIN[2],
                      lambda3 = wshLambdaMM_BITCOIN[3], lambda4 = wshLambdaMM_BITCOIN
                          [4],
                      param = "fmkl", inverse.eps = 1e-08,
                      max.iterations = 500)

BITCOIN$gld = fittedVal_BITCOIN


logret_ALIBABA = ALIBABA$return_100[-1]

#Method of moment
wshLambdaMM_ALIBABA <- fun.RMFMKL.mm(logret_ALIBABA)
```

```
fittedVal_ALIBABA = dgl(ALIBABA$return_100, lambda1 = wshLambdaMM_ALIBABA[1], lambda2
    = wshLambdaMM_ALIBABA[2],
                        lambda3 = wshLambdaMM_ALIBABA[3], lambda4 = wshLambdaMM_ALIBABA
                            [4],
                        param = "fmkl", inverse.eps = 1e-08,
                        max.iterations = 500)

ALIBABA$gld = fittedVal_ALIBABA

######## plot ##########

png(filename = "fit_distrib.png", width = 1000)

g_CAC40 = ggplot(data = CAC40)
g_CAC40 = g_CAC40 + geom_histogram(aes(x = return_100, y = ..density..), colour = '
    lightblue', fill = "blue", alpha = 0.3)
g_CAC40 = g_CAC40 + geom_density(aes(x = return_100, colour = 'Kernel_density'),
    alpha = 0.8, lwd = 0.8, lty = 1)
g_CAC40 = g_CAC40 + geom_line(aes(x = return_100, y = normal, color = "Normal"), lwd
    = 0.8, lty = 1)
g_CAC40 = g_CAC40 + geom_line(aes(x = return_100, y = stable, color = 'Stable'), lwd
    = 0.8, lty = 1)
g_CAC40 = g_CAC40 + geom_line(aes(x = return_100, y = t, color = 'Student'), lwd =
    0.8, lty = 1)
g_CAC40 = g_CAC40 + geom_line(aes(x = return_100, y = gld, color = 'Lambda'), lwd =
    0.8, lty = 2)
g_CAC40 = g_CAC40 + scale_x_continuous(limits = c(min(CAC40$return_100),max(CAC40$
    return_100)))
g_CAC40 = g_CAC40 + scale_color_manual(name = "Distribution", values = c(
  "Kernel_density" = "blue",
  'Stable' = 'green',
  'Normal' = 'red',
  'Student' = 'yellow',
  'Lambda' = 'black'))
g_CAC40 = g_CAC40 + my_theme + labs(x = "return_(%)", title = "CAC40_returns")
g_CAC40 = g_CAC40 + theme(legend.position="bottom")

g_ALIBABA = ggplot(data = ALIBABA)
g_ALIBABA = g_ALIBABA + geom_histogram(aes(x = return_100, y = ..density..), colour =
    'lightblue', fill = "blue", alpha = 0.3)
g_ALIBABA = g_ALIBABA + geom_density(aes(x = return_100, colour = 'Kernel_density'),
    alpha = 0.8, lwd = 0.8, lty = 1)
g_ALIBABA = g_ALIBABA + geom_line(aes(x = return_100, y = normal, color = "Normal"),
    lwd = 0.8, lty = 1)
g_ALIBABA = g_ALIBABA + geom_line(aes(x = return_100, y = stable, color = 'Stable'),
    lwd = 0.8, lty = 1)
g_ALIBABA = g_ALIBABA + geom_line(aes(x = return_100, y = t, color = 'Student'), lwd
    = 0.8, lty = 1)
g_ALIBABA = g_ALIBABA + geom_line(aes(x = return_100, y = gld, color = 'Lambda'), lwd
    = 0.8, lty = 2)
g_ALIBABA = g_ALIBABA + scale_x_continuous(limits = c(min(ALIBABA$return_100),max(
    ALIBABA$return_100)))
g_ALIBABA = g_ALIBABA + scale_color_manual(name = "Distribution", values = c(
  "Kernel_density" = "blue",
  'Stable' = 'green',
  'Normal' = 'red',
  'Student' = 'yellow',
  'Lambda' = 'black'))
g_ALIBABA = g_ALIBABA + my_theme + labs(x = "return_(%)", title = "ALIBABA_returns")
g_ALIBABA = g_ALIBABA + theme(legend.position="bottom")

g_BITCOIN = ggplot(data = BITCOIN)
g_BITCOIN = g_BITCOIN + geom_histogram(aes(x = return_100, y = ..density..), colour =
    'lightblue', fill = "blue", alpha = 0.3)
```

```
g_BITCOIN = g_BITCOIN + geom_density(aes(x = return_100, colour = 'Kernel density'),
    alpha = 0.8, lwd = 0.8, lty = 1)
g_BITCOIN = g_BITCOIN + geom_line(aes(x = return_100, y = normal, color = "Normal"),
    lwd = 0.8, lty = 1)
g_BITCOIN = g_BITCOIN + geom_line(aes(x = return_100, y = stable, color = 'Stable'),
    lwd = 0.8, lty = 1)
g_BITCOIN = g_BITCOIN + geom_line(aes(x = return_100, y = t, color = 'Student'), lwd
    = 0.8, lty = 1)
g_BITCOIN = g_BITCOIN + geom_line(aes(x = return_100, y = gld, color = 'Lambda'), lwd
    = 0.8, lty = 2)
g_BITCOIN = g_BITCOIN + scale_x_continuous(limits = c(min(BITCOIN$return_100),max(
    BITCOIN$return_100)))
g_BITCOIN = g_BITCOIN + scale_color_manual(name = "Distribution", values = c(
  "Kernel density" = "blue",
  'Stable' = 'green',
  'Normal' = 'red',
  'Student' = 'yellow',
  'Lambda' = 'black'))
g_BITCOIN = g_BITCOIN + my_theme + labs(x = "return (%)", title = "BITCOIN returns")
g_BITCOIN = g_BITCOIN + theme(legend.position="bottom")


ggarrange(g_ALIBABA, g_CAC40, g_BITCOIN, common.legend = TRUE, ncol = 3, legend = "
    bottom")

dev.off()

## statistics (mean, sd, skewness, kurtosis) ##

m_CAC40 = mean(CAC40$return_100)
sd_CAC40 = sd(CAC40$return_100)
k_CAC40 = kurtosis(CAC40$return_100, method = "moment")
s_CAC40 = skewness(CAC40$return_100)

##### BOOTSTRAP ######

## Normal ##

m_rep_norm = c()
sd_rep_norm = c()
s_rep_norm = c()
k_rep_norm = c()

replicate(5000,{
  b_data = rnorm(n, mean = mu_CAC40, sd = sigma_CAC40)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_norm <<- c(m_rep_norm, mean(b_data))
  sd_rep_norm <<- c(sd_rep_norm, sd(b_data))
  s_rep_norm <<- c(s_rep_norm, skewness(b_data))
  k_rep_norm <<- c(k_rep_norm, kurtosis(b_data, method = "moment"))
})

m_CAC40_normal = mean(m_rep_norm)
sd_CAC40_normal = mean(sd_rep_norm)
s_CAC40_normal = mean(s_rep_norm)
k_CAC40_normal = mean(k_rep_norm)


## Student ##

m_rep_t = c()
sd_rep_t = c()
s_rep_t = c()
k_rep_t = c()
```

```
replicate(5000,{
  b_data = rt(n, df = df_hat_CAC40, ncp = ncp_hat_CAC40)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_t <<- c(m_rep_t, mean(b_data))
  sd_rep_t <<- c(sd_rep_t, sd(b_data))
  s_rep_t <<- c(s_rep_t, skewness(b_data))
  k_rep_t <<- c(k_rep_t, kurtosis(b_data, method = "moment"))
})

m_CAC40_t = mean(m_rep_t)
sd_CAC40_t = mean(sd_rep_t)
s_CAC40_t = mean(s_rep_t)
k_CAC40_t = mean(k_rep_t)


## Stable ##

m_rep_stable = c()
sd_rep_stable = c()
s_rep_stable = c()
k_rep_stable = c()

replicate(5000,{
  b_data = rstable(n, 2*exp(a_CAC40)/(1+exp(a_CAC40)), 2*exp(b_CAC40)/(1+exp(b_CAC40)
      )-1, exp(c_CAC40), d_CAC40)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_stable <<- c(m_rep_stable, mean(b_data))
  sd_rep_stable <<- c(sd_rep_stable, sd(b_data))
  s_rep_stable <<- c(s_rep_stable, skewness(b_data))
  k_rep_stable <<- c(k_rep_stable, kurtosis(b_data, method = "moment"))
})

m_CAC40_stable = mean(m_rep_stable)
sd_CAC40_stable = mean(sd_rep_stable)
s_CAC40_stable = mean(s_rep_stable)
k_CAC40_stable = mean(k_rep_stable)

## GLD ##

m_rep_gld = c()
sd_rep_gld = c()
s_rep_gld = c()
k_rep_gld = c()

replicate(5000,{
  b_data = rgl(n,lambda1 = wshLambdaMM_CAC40[1], lambda2 = wshLambdaMM_CAC40[2],
               lambda3 = wshLambdaMM_CAC40[3], lambda4 = wshLambdaMM_CAC40[4],
               param = "fmkl")
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_gld <<- c(m_rep_gld, mean(b_data))
  sd_rep_gld <<- c(sd_rep_gld, sd(b_data))
  s_rep_gld <<- c(s_rep_gld, skewness(b_data))
  k_rep_gld <<- c(k_rep_gld, kurtosis(b_data, method = "moment"))
})

m_CAC40_gld = mean(m_rep_gld)
sd_CAC40_gld = mean(sd_rep_gld)
s_CAC40_gld = mean(s_rep_gld)
k_CAC40_gld = mean(k_rep_gld)


# ALIBABA
```

```
## Normal ##

m_rep_norm = c()
sd_rep_norm = c()
s_rep_norm = c()
k_rep_norm = c()

replicate(5000,{
  b_data = rnorm(n, mean = mu_ALIBABA, sd = sigma_ALIBABA)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_norm <<- c(m_rep_norm, mean(b_data))
  sd_rep_norm <<- c(sd_rep_norm, sd(b_data))
  s_rep_norm <<- c(s_rep_norm, skewness(b_data))
  k_rep_norm <<- c(k_rep_norm, kurtosis(b_data, method = "moment"))
})

m_ALIBABA_normal = mean(m_rep_norm)
sd_ALIBABA_normal = mean(sd_rep_norm)
s_ALIBABA_normal = mean(s_rep_norm)
k_ALIBABA_normal = mean(k_rep_norm)


## Student ##

m_rep_t = c()
sd_rep_t = c()
s_rep_t = c()
k_rep_t = c()

replicate(5000,{
  b_data = rt(n, df = df_hat_ALIBABA, ncp = ncp_hat_ALIBABA)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_t <<- c(m_rep_t, mean(b_data))
  sd_rep_t <<- c(sd_rep_t, sd(b_data))
  s_rep_t <<- c(s_rep_t, skewness(b_data))
  k_rep_t <<- c(k_rep_t, kurtosis(b_data, method = "moment"))
})

m_ALIBABA_t = mean(m_rep_t)
sd_ALIBABA_t = mean(sd_rep_t)
s_ALIBABA_t = mean(s_rep_t)
k_ALIBABA_t = mean(k_rep_t)


## Stable ##

m_rep_stable = c()
sd_rep_stable = c()
s_rep_stable = c()
k_rep_stable = c()

replicate(5000,{
  b_data = rstable(n, 2*exp(a_ALIBABA)/(1+exp(a_ALIBABA)), 2*exp(b_ALIBABA)/(1+exp(b_
      ALIBABA))-1, exp(c_ALIBABA), d_ALIBABA)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_stable <<- c(m_rep_stable, mean(b_data))
  sd_rep_stable <<- c(sd_rep_stable, sd(b_data))
  s_rep_stable <<- c(s_rep_stable, skewness(b_data))
  k_rep_stable <<- c(k_rep_stable, kurtosis(b_data, method = "moment"))
})

m_ALIBABA_stable = mean(m_rep_stable)
sd_ALIBABA_stable = mean(sd_rep_stable)
```

```
s_ALIBABA_stable = mean(s_rep_stable)
k_ALIBABA_stable = mean(k_rep_stable)

## GLD ##

m_rep_gld = c()
sd_rep_gld = c()
s_rep_gld = c()
k_rep_gld = c()

replicate(5000,{
  b_data = rgl(n,lambda1 = wshLambdaMM_ALIBABA[1], lambda2 = wshLambdaMM_ALIBABA[2],
               lambda3 = wshLambdaMM_ALIBABA[3], lambda4 = wshLambdaMM_ALIBABA[4],
               param = "fmkl")
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_gld <<- c(m_rep_gld, mean(b_data))
  sd_rep_gld <<- c(sd_rep_gld, sd(b_data))
  s_rep_gld <<- c(s_rep_gld, skewness(b_data))
  k_rep_gld <<- c(k_rep_gld, kurtosis(b_data, method = "moment"))
})

m_ALIBABA_gld = mean(m_rep_gld)
sd_ALIBABA_gld = mean(sd_rep_gld)
s_ALIBABA_gld = mean(s_rep_gld)
k_ALIBABA_gld = mean(k_rep_gld)


# BITCOIN

## Normal ##

m_rep_norm = c()
sd_rep_norm = c()
s_rep_norm = c()
k_rep_norm = c()

replicate(5000,{
  b_data = rnorm(n, mean = mu_BITCOIN, sd = sigma_BITCOIN)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_norm <<- c(m_rep_norm, mean(b_data))
  sd_rep_norm <<- c(sd_rep_norm, sd(b_data))
  s_rep_norm <<- c(s_rep_norm, skewness(b_data))
  k_rep_norm <<- c(k_rep_norm, kurtosis(b_data, method = "moment"))
})

m_BITCOIN_normal = mean(m_rep_norm)
sd_BITCOIN_normal = mean(sd_rep_norm)
s_BITCOIN_normal = mean(s_rep_norm)
k_BITCOIN_normal = mean(k_rep_norm)


## Student ##

m_rep_t = c()
sd_rep_t = c()
s_rep_t = c()
k_rep_t = c()

replicate(5000,{
  b_data = rt(n, df = df_hat_BITCOIN, ncp = ncp_hat_BITCOIN)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_t <<- c(m_rep_t, mean(b_data))
  sd_rep_t <<- c(sd_rep_t, sd(b_data))
  s_rep_t <<- c(s_rep_t, skewness(b_data))
```

```
  k_rep_t <<- c(k_rep_t, kurtosis(b_data, method = "moment"))
})

m_BITCOIN_t = mean(m_rep_t)
sd_BITCOIN_t = mean(sd_rep_t)
s_BITCOIN_t = mean(s_rep_t)
k_BITCOIN_t = mean(k_rep_t)


## Stable ##

m_rep_stable = c()
sd_rep_stable = c()
s_rep_stable = c()
k_rep_stable = c()

replicate(5000,{
  b_data = rstable(n, 2*exp(a_BITCOIN)/(1+exp(a_BITCOIN)), 2*exp(b_BITCOIN)/(1+exp(b_
      BITCOIN))-1, exp(c_BITCOIN), d_BITCOIN)
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_stable <<- c(m_rep_stable, mean(b_data))
  sd_rep_stable <<- c(sd_rep_stable, sd(b_data))
  s_rep_stable <<- c(s_rep_stable, skewness(b_data))
  k_rep_stable <<- c(k_rep_stable, kurtosis(b_data, method = "moment"))
})

m_BITCOIN_stable = mean(m_rep_stable)
sd_BITCOIN_stable = mean(sd_rep_stable)
s_BITCOIN_stable = mean(s_rep_stable)
k_BITCOIN_stable = mean(k_rep_stable)

## GLD ##

m_rep_gld = c()
sd_rep_gld = c()
s_rep_gld = c()
k_rep_gld = c()

replicate(5000,{
  b_data = rgl(n,lambda1 = wshLambdaMM_BITCOIN[1], lambda2 = wshLambdaMM_BITCOIN[2],
               lambda3 = wshLambdaMM_BITCOIN[3], lambda4 = wshLambdaMM_BITCOIN[4],
               param = "fmkl")
  b_data = b_data[which(b_data < max(return_100) & b_data > min(return_100))]
  m_rep_gld <<- c(m_rep_gld, mean(b_data))
  sd_rep_gld <<- c(sd_rep_gld, sd(b_data))
  s_rep_gld <<- c(s_rep_gld, skewness(b_data))
  k_rep_gld <<- c(k_rep_gld, kurtosis(b_data, method = "moment"))
})

m_BITCOIN_gld = mean(m_rep_gld)
sd_BITCOIN_gld = mean(sd_rep_gld)
s_BITCOIN_gld = mean(s_rep_gld)
k_BITCOIN_gld = mean(k_rep_gld)

## TABLES ##

m = c(m_CAC40, m_ALIBABA, m_BITCOIN)
m_norm = c(m_CAC40_normal, m_ALIBABA_normal, m_BITCOIN_normal)
m_t = c(m_CAC40_t, m_ALIBABA_t, m_BITCOIN_t)
m_stable = c(m_CAC40_stable, m_ALIBABA_stable, m_BITCOIN_stable)
m_gld = c(m_CAC40_gld, m_ALIBABA_gld, m_BITCOIN_gld)

sdev = c(sd_CAC40, sd_ALIBABA, sd_BITCOIN)
sd_norm = c(sd_CAC40_normal, sd_ALIBABA_normal, sd_BITCOIN_normal)
```

```
sd_t = c(sd_CAC40_t, sd_ALIBABA_t, sd_BITCOIN_t)
sd_stable = c(sd_CAC40_stable, sd_ALIBABA_stable, sd_BITCOIN_stable)
sd_gld = c(sd_CAC40_gld, sd_ALIBABA_gld, sd_BITCOIN_gld)

k = c(k_CAC40, k_ALIBABA, k_BITCOIN)
k_norm = c(k_CAC40_normal, k_ALIBABA_normal, k_BITCOIN_normal)
k_t = c(k_CAC40_t, k_ALIBABA_t, k_BITCOIN_t)
k_stable = c(k_CAC40_stable, k_ALIBABA_stable, k_BITCOIN_stable)
k_gld = c(k_CAC40_gld, k_ALIBABA_gld, k_BITCOIN_gld)

s = c(s_CAC40, s_ALIBABA, s_BITCOIN)
s_norm = c(s_CAC40_normal, s_ALIBABA_normal, s_BITCOIN_normal)
s_t = c(s_CAC40_t, s_ALIBABA_t, s_BITCOIN_t)
s_stable = c(s_CAC40_stable, s_ALIBABA_stable, s_BITCOIN_stable)
s_gld = c(s_CAC40_gld, s_ALIBABA_gld, s_BITCOIN_gld)


xtable(data.frame(m, m_norm, m_t, m_stable, m_gld, row.names = c("CAC40","ALIBABA","
    BITCOIN")))
xtable(data.frame(sdev, sd_norm, sd_t, sd_stable, sd_gld, row.names = c("CAC40","
    ALIBABA","BITCOIN")))
xtable(data.frame(k, k_norm, k_t, k_stable, k_gld, row.names = c("CAC40","ALIBABA","
    BITCOIN")))
xtable(data.frame(s, s_norm, s_t, s_stable, s_gld, row.names = c("CAC40","ALIBABA","
    BITCOIN")))


## p-values Kolmogorov-Smirnov ##

# CAC40

pval_CAC40_norm = ks.test(CAC40$return_100, function(x) pnorm(x, mean = mean(CAC40$
    return_100), sd = sd(CAC40$return_100)), exact = TRUE)$p.value
pval_CAC40_t = ks.test(CAC40$return_100, function(x) pt(x, df = df_hat_CAC40, ncp =
    ncp_hat_CAC40), exact = TRUE)$p.value
pval_CAC40_stable = ks.test(CAC40$return_100, function(x) pstable(x, 2*exp(a_CAC40)/
    (1+exp(a_CAC40)),
                                                    2*exp(b_CAC40)/(1+exp(b_CAC40))-1,
                                                    exp(c_CAC40),
                                                    d_CAC40), exact = TRUE)$p.value
pval_CAC40_gld = ks.test(CAC40$return_100, function(x) pgl(x, lambda1 = wshLambdaMM_
    CAC40[1], lambda2 = wshLambdaMM_CAC40[2],
                                                lambda3 = wshLambdaMM_CAC40[3], lambda4 =
                                                    wshLambdaMM_CAC40[4],
                                                param = "fmkl"), exact = TRUE)$p.value

# ALIBABA

pval_ALIBABA_norm = ks.test(ALIBABA$return_100, function(x) pnorm(x, mean = mean(
    ALIBABA$return_100), sd = sd(ALIBABA$return_100)), exact = TRUE)$p.value
pval_ALIBABA_t = ks.test(ALIBABA$return_100, function(x) pt(x, df = df_hat_ALIBABA,
    ncp = ncp_hat_ALIBABA), exact = TRUE)$p.value
pval_ALIBABA_stable = ks.test(ALIBABA$return_100, function(x) pstable(x, 2*exp(a_
    ALIBABA)/(1+exp(a_ALIBABA)),
                                                    2*exp(b_ALIBABA)/(1+exp(b_ALIBABA))
                                                        -1,
                                                    exp(c_ALIBABA),
                                                    d_ALIBABA), exact = TRUE)$p.value
pval_ALIBABA_gld = ks.test(ALIBABA$return_100, function(x) pgl(x, lambda1 =
    wshLambdaMM_ALIBABA[1], lambda2 = wshLambdaMM_ALIBABA[2],
                                                lambda3 = wshLambdaMM_ALIBABA[3], lambda4 =
                                                    wshLambdaMM_ALIBABA[4],
                                                param = "fmkl"), exact = TRUE)$p.value
```

```
# BITCOIN

pval_BITCOIN_norm = ks.test(BITCOIN$return_100, function(x) pnorm(x, mean = mean(
    BITCOIN$return_100), sd = sd(BITCOIN$return_100)), exact = TRUE)$p.value
pval_BITCOIN_t = ks.test(BITCOIN$return_100, function(x) pt(x, df = df_hat_BITCOIN,
    ncp = ncp_hat_BITCOIN), exact = TRUE)$p.value
pval_BITCOIN_stable = ks.test(BITCOIN$return_100, function(x) pstable(x, 2*exp(a_
    BITCOIN)/(1+exp(a_BITCOIN)),
                                                        2*exp(b_BITCOIN)/(1+exp(b_BITCOIN))
                                                            -1,
                                                        exp(c_BITCOIN),
                                                        d_BITCOIN), exact = TRUE)$p.value
pval_BITCOIN_gld = ks.test(BITCOIN$return_100, function(x) pgl(x, lambda1 =
    wshLambdaMM_BITCOIN[1], lambda2 = wshLambdaMM_BITCOIN[2],
                                                    lambda3 = wshLambdaMM_BITCOIN[3], lambda4 =
                                                        wshLambdaMM_BITCOIN[4],
                                                    param = "fmkl"), exact = TRUE)$p.value

## Tables ##

df_pval = data.frame(norm = c(pval_CAC40_norm, pval_ALIBABA_norm, pval_BITCOIN_norm),
                     t = c(pval_CAC40_t, pval_ALIBABA_t, pval_BITCOIN_t),
                     stable = c(pval_CAC40_stable, pval_ALIBABA_stable, pval_BITCOIN_
                         stable),
                     gld = c(pval_CAC40_gld, pval_ALIBABA_gld, pval_BITCOIN_gld),
                     row.names = c("CAC40", "ALIBABA", "BITCOIN"))

xtable(df_pval, digits = -1)



# Compute the moment without troncating for the GLD



fun.moments.r(CAC40$return_100,normalise="N")

# Moments of fitted distribution:
fun.moments.r(ALIBABA$return_100,normalise="N")
#Method of moment
wshLambdaMMAli <- fun.RMFMKL.mm(ALIBABA$return_100)
# Moments of fitted distribution:
fun.theo.mv.gld(wshLambdaMMAli[1], wshLambdaMMAli[2], wshLambdaMMAli[3],
    wshLambdaMMAli[4],
                param = "fmkl", normalise="N")


fun.moments.r(BITCOIN$return_100,normalise="N")
#Method of moment
wshLambdaMMBit <- fun.RMFMKL.mm(BITCOIN$return_100)
# Moments of fitted distribution:
fun.theo.mv.gld(wshLambdaMMBit[1], wshLambdaMMBit[2], wshLambdaMMBit[3],
    wshLambdaMMBit[4],
                param = "fmkl", normalise="N")

fun.moments.r(CAC40$return_100,normalise="N")
#Method of moment
wshLambdaMMCAT <- fun.RMFMKL.mm(CAC40$return_100)
# Moments of fitted distribution:
fun.theo.mv.gld(wshLambdaMMCAT[1], wshLambdaMMCAT[2], wshLambdaMMCAT[3],
    wshLambdaMMCAT[4],
                param = "fmkl", normalise="N")
```