Capstone Project

Matias Garcia Mazzaro

noviembre 27, 2020

Contents

The Project

The general idea of this project is to analice the crypto market and prices. I pretend, through the exploration, detect diferents levels of correlation between peers of coins and understand the dinamic and flows of transactions. For this, is necessary divide the analysis in two or more parts. Also i will construct the dataset making web scraping. Finally i will use Machine Learning to predict the prices. This work is the final project for the Data Science Professional certificate of HarvardX.

Chapter 1

Introduction

For this work, i will choice a specific plataform of trade, is normal diferents plataforms have diferents dinamics and flow of buy and sell trade. The transfer between diferents plataforms and traders conflude in a nivelation of the prices of the total market. So, it will give us a general idea of the market and specific trade.

Is necesary make a definition and clear how i will pose the analysis of this work.

Before to start, we will calculate a long term trend of prices for the principal cryto coin. The variation on short term and flows produced by trade in the short terms. Then we will define if is necesary advance in analysis of volume of transactions.

For the long term trend data we will use historical specific data from a site who provide this daily information, this site is Investing.com. For the short variation i will construct a data set minute by minute from Coinmarketcap.com. For train the Machine Learning algorithm, i will use a dataset minute by minute published in Kaggle, is posible to have it pickin here, the information inside include data from 2012-01-01 to 2020-09-14. All datasets downloads automatically with the code, the historical data is in real time when you execute the code, and the short terms data set, which was working many hours is not posibble to include in this code, for solve it, i saved in my GitHub account. Also i include the code for construct it like a aditional information. This code is also in folder Capstone in GitHub with the name *constructing_dataset.R*.

1.1 Objetive

The objetive is undestand and predict trought diferents methods of Machine Learning, the dinamics of prices of the first cryptocoins. The general estrategy was planed, but the specific knowing will be apear in the process of exploration.

1.2 Loading the packages and libraries

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(rvest)) install.packages("rvest", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org"
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org"
if(!require(TTR)) install.packages("TTR", repos = "http://cran.us.r-project.org")
if(!require(zoo)) install.packages("zoo", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(rvest)
library(dplyr)
library(lubridate)
library(stringr)
library(ggplot2)
library(TTR)
library(zoo)
library(knitr)
```

1.3 Construct the dataset minute by minute

In this section i will explaind and show how i construct the code for scrap the minute by minute dataset, and i give a example of one or two web scraping captures, but the utility of that is making work it many hours. This chapter is a addicional informatión.

1.3.1 Read the html webpage

We read the webpage and continuosly we save the date in the moment exactly after.

```
coinmarket <- read_html("https://coinmarketcap.com/")
date <- now()
class(coinmarket)
```

```
## [1] "xml_document" "xml_node"
```

We identify the CCS selector for take the Name of coin, price, and volume. With CSS selectors we save the price of the first 100 cryptocoins, then, we take out the simbol "$", the "," and others words. Continuosly we order the data in a martix and convert it in a tibble. Then we add the data of scraping in the rows.

```
market <- html_nodes(coinmarket, ".iJjGCS , .hNpJqV , td .kDEzev , .price___3rj7O .cmc-l
length(market)
```

```
## [1] 60
```

```
market_text <- html_text(market)
market_text <- market_text %>% str_replace_all("[$]", "") #we take out the symbol "$"
market_text <- market_text %>% str_replace_all(",", "") # we take out the ","
market_text <- market_text %>% str_replace_all("\\s[A-Z]*", "") # we take out the name
market_text
```

```
##  [1] "Bitcoin"      "BTC"          "16826.44"     "312229132240" "39878281570"
##  [6] "18555868"     "Ethereum"     "ETH"          "507.26"       "57625697915"
## [11] "17365030586"  "113601740"    "XRP"          "XRP"          "0.540575"
## [16] "24514112503"  "19434297392"  "45348221180"  "Tether"       "USDT"
## [21] "1.00"         "18980730423"  "74234891389"  "18957655237"  "Bitcoinash"
## [26] "BCH"          "263.66"       "4899852640"   "3067089573"   "18584038"
## [31] "Chainlink"    "LINK"         "12.32"        "4855167435"   "1832178646"
## [36] "394009556"    "Litecoin"     "LTC"          "68.47"        "4516325816"
## [41] "6481711858"   "65962552"     "Cardano"      "ADA"          "0.135556"
## [46] "4217496796"   "1680091622"   "31112484646"  "Polkadot"     "DOT"
## [51] "4.68"         "4126626895"   "921274030"    "881378754"    "Binanceoin"
## [56] "BNB"          "28.10"        "4058277303"   "383245447"    "144406561"
```

```
market_data <- matrix(market_text, 100, 6, byrow = TRUE)
market_data <- as_tibble(market_data)
market_data <- market_data %>% mutate(date = date)
head(market_data) %>% knitr::kable()
```

V1	V2	V3	V4	V5	V6	date
Bitcoin	BTC	16826.44	312229132240	39878281570	18555868	2020-11-27 16:17:48
Ethereum	ETH	507.26	57625697915	17365030586	113601740	2020-11-27 16:17:48
XRP	XRP	0.540575	24514112503	19434297392	45348221180	2020-11-27 16:17:48
Tether	USDT	1.00	18980730423	74234891389	18957655237	2020-11-27 16:17:48
Bitcoinash	BCH	263.66	4899852640	3067089573	18584038	2020-11-27 16:17:48
Chainlink	LINK	12.32	4855167435	1832178646	394009556	2020-11-27 16:17:48

Then we can transform the class of the variables for procces and add the name in the columns.

```
market_data$V3 <- as.numeric(market_data$V3)
market_data$V4 <- as.numeric(market_data$V4)
market_data$V5 <- as.numeric(market_data$V5)
market_data$V6 <- as.numeric(market_data$V6)
colnames(market_data) <- c("Name", "Key", "Price", "Market_Cap", "Vol24hs", "Circulating
head(market_data) %>% knitr::kable()
```

Name	Key	Price	Market_Cap	Vol24hs	Circulating	Date
Bitcoin	BTC	16826.440000	312229132240	39878281570	18555868	2020-11-27 16:17:48
Ethereum	ETH	507.260000	57625697915	17365030586	113601740	2020-11-27 16:17:48
XRP	XRP	0.540575	24514112503	19434297392	45348221180	2020-11-27 16:17:48
Tether	USDT	1.000000	18980730423	74234891389	18957655237	2020-11-27 16:17:48
Bitcoinash	BCH	263.660000	4899852640	3067089573	18584038	2020-11-27 16:17:48
Chainlink	LINK	12.320000	4855167435	1832178646	394009556	2020-11-27 16:17:48

1.3.2 Others details

We need to construct others details for ensamble all the information in one complete dataset. It can be like this:

market_data1 <- full_join(market_data, market_data1) market_data1

Another thing is how set the timer for sleep the time we need. It is (in senconds) a example for sleep 1 minute;

Sys.sleep(60)

1.3.3 The Ensemble

We ensemble all in a function of **steps (m)** we want and **seconds (s)** we want to sleep the system between scrap and scrap.

```
datasetmakingfor <- function(m, s){
  i <- 1
  for(i in 1:m){
    Sys.sleep(s)
    coinmarket <- read_html("https://coinmarketcap.com/")
    date <- now()
    market <- html_nodes(coinmarket, ".iJjGCS , .hNpJqV , td .kDEzev , .price___3rj7O .c
    market_text <- html_text(market)
    market_text <- market_text %>% str_replace_all("[$]", "") #we take out the symbol "
```

```
    market_text <- market_text %>% str_replace_all(",", "") # we take out the ","
    market_text <- market_text %>% str_replace_all("\\s[A-Z]*", "") # we take out the n
    market_data <- matrix(market_text, 100, 6, byrow = TRUE)
    market_data <- as_tibble(market_data)
    market_data <- market_data %>% mutate(date = date)
    market_data$V3 <- as.numeric(market_data$V3)
    market_data$V4 <- as.numeric(market_data$V4)
    market_data$V5 <- as.numeric(market_data$V5)
    market_data$V6 <- as.numeric(market_data$V6)
    colnames(market_data) <- c("Name", "Key", "Price", "Market_Cap", "Vol24hs", "Circula
    if(i == 1){market_data1 <- market_data}
    else market_data1 <- full_join(market_data, market_data1)
    i+1
    print((i/m)*100)}
  market_data1
}
```

In the function we incorpore too a **If - else** to jump in the first loop, because is not posible to join a matrix with another that at the moment don't exist. Also we add a count _print((i/m)*100)_ for know the percentage complete of the scraping. Like we say before, we run this function only with 4 steps and 5 seconds, but the objetive is to get a dataset of 24 hours.

Only for a example;

```
dataset <- datasetmakingfor(4, 5) #4 step, sleep for 5 seconds
```

```
## [1] 25
## [1] 50
## [1] 75
## [1] 100
```

```
nrow(dataset)
```

```
## [1] 400
```

```
head(dataset) %>% knitr::kable()
```

Name	Key	Price	Market_Cap	Vol24hs	Circulating	Date
Bitcoin	BTC	16826.440000	312229132240	39878281570	18555868	2020-11-27 16:18:10
Ethereum	ETH	507.260000	57625697915	17365030586	113601740	2020-11-27 16:18:10
XRP	XRP	0.540575	24514112503	19434297392	45348221180	2020-11-27 16:18:10
Tether	USDT	1.000000	18980730423	74234891389	18957655237	2020-11-27 16:18:10
Bitcoinash	BCH	263.660000	4899872640	3067089573	18584038	2020-11-27 16:18:10
Chainlink	LINK	12.320000	4855167435	1832178646	394009556	2020-11-27 16:18:10

For be posible the analysis I saved a dataset of 24 hours in my GitHub account, it will download automatically for the next analysis. The name is **dset.csv**

Chapter 2

Analysis

In this section i will explains the process and techniques used, including data cleaning, data exploration and visualization, any insights gained, and your modeling approach. At least two different models or algorithms must be used, with at least one being more advanced than linear or logistic regression for prediction problems.

2.1 Data exploration

```
url <- "https://raw.githubusercontent.com/Avantas/Capstone/main/dset.csv"
dset <- read_csv(url)
```

If you want to have a local copy **download.file(url, "dataset.csv")**

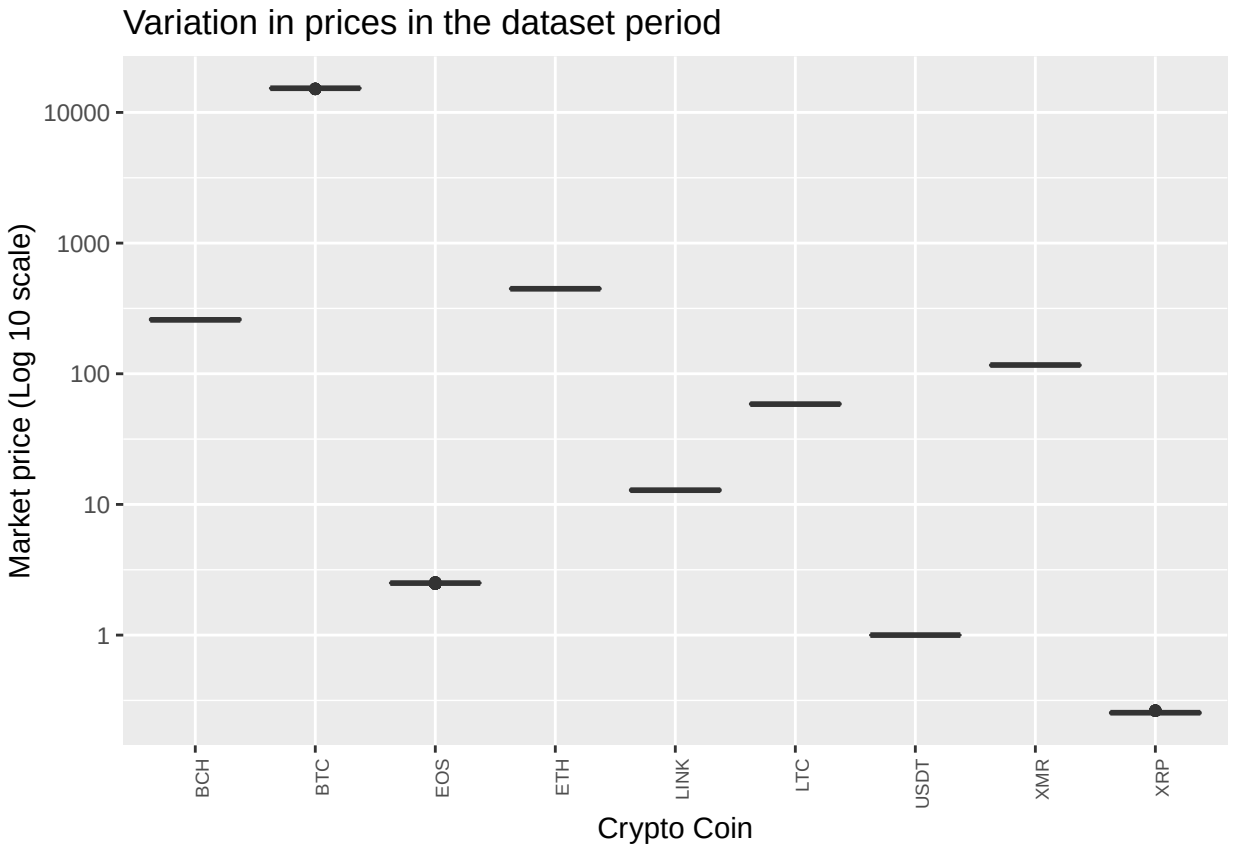
```
head(dset) %>% knitr::kable()
```

X1	Name	Key	Price	Market_Cap	Vol24hs	Circulating	Date
1	Bitcoin	BTC	15352.470000	284609678928	64262125545	18538368	2020-11-09 22:31:24
2	Ethereum	ETH	446.670000	50635045504138961665921133613345			2020-11-09 22:31:24
3	Tether	USDT	1.000000	17351235367463762670191734069000		42	2020-11-09 22:31:24
4	XRP	XRP	0.250291	113413100593112230771	453124888502		2020-11-09 22:31:24
5	Chainlink	LINK	12.670000	4961708119	1619586853	391509556	2020-11-09 22:31:24
6	Bitcoinash	BCH	264.590000	4912850732	1895163320	18567813	2020-11-09 22:31:24

We need to take out the column X1

```
dset$X1 <- NULL
```

```
dset %>% group_by(Key) %>% filter(Vol24hs >= 10^9) %>% # we filter for Crypto who marke
  ggplot(aes(Key, Price)) + geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 7)) +
  scale_y_log10() +
  xlab("Crypto Coin") +
  ylab("Market price (Log 10 scale)") +
  ggtitle("Variation in prices in the dataset period")
```

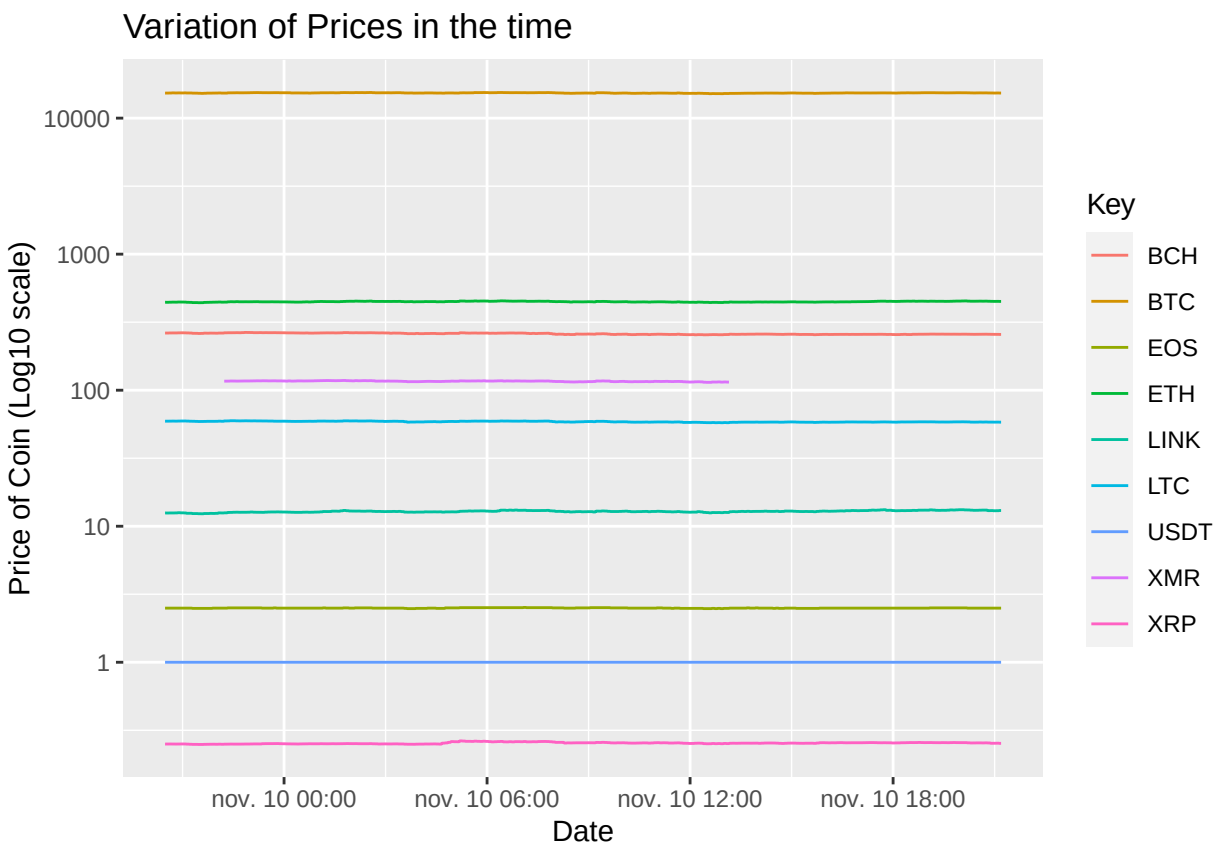



```
head(dset) %>% knitr::kable()
```

Name	Key	Price	Market_Cap	Vol24hs	Circulating	Date
Bitcoin	BTC	15352.470000	284609678926	34262125545	18538368	2020-11-09 22:31:24
Ethereum	ETH	446.670000	50635045504	13896166592	113361345	2020-11-09 22:31:24
Tether	USDT	1.000000	17351235367	46376267019	17340690004	2020-11-09 22:31:24
XRP	XRP	0.250291	11341310059	3112230771	45312488850	2020-11-09 22:31:24
Chainlink	LINK	12.670000	4961708119	1619586853	391509556	2020-11-09 22:31:24
Bitcoinash	BCH	264.590000	4912850732	1895163320	18567813	2020-11-09 22:31:24

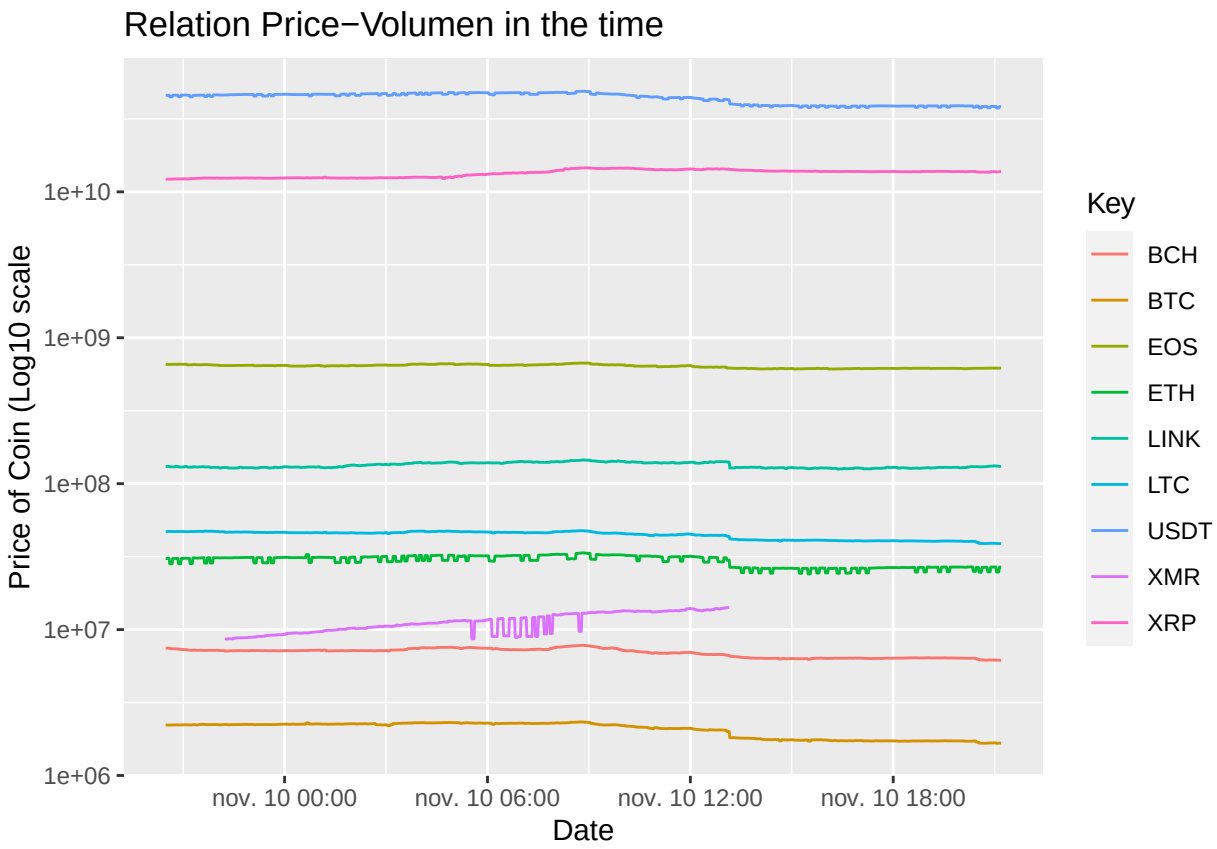
We observe the price fluctuations between diferents coins over the date

```
dset %>% group_by(Key) %>% filter(Vol24hs >= 10^9) %>%
  ggplot(aes(Date, Price, color = Key)) + geom_line() + scale_y_log10() +
  xlab("Date") + ylab("Price of Coin (Log10 scale)") + ggtitle("Variation of Prices in t
```



For observe the weight of the prices moviment, we create a relation between price and volumen

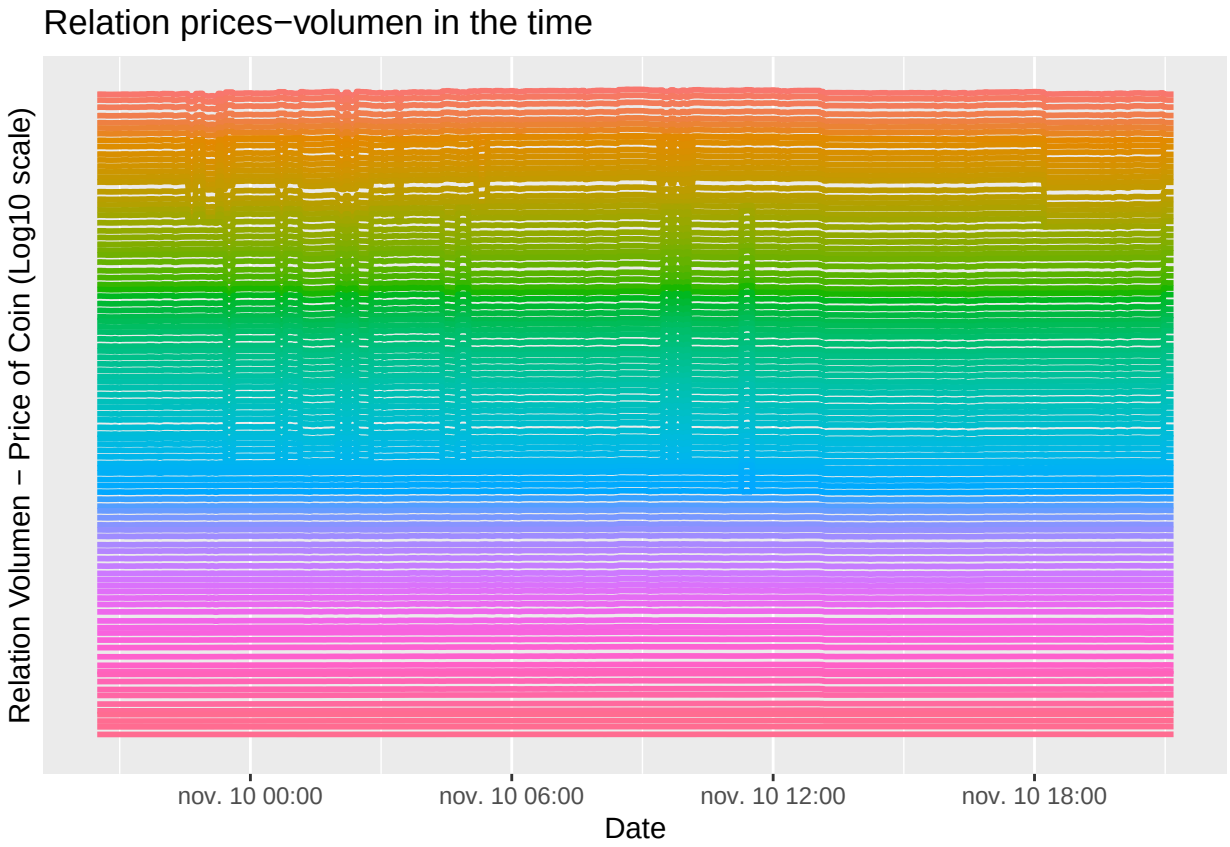
```
dset %>% mutate(relation_vol_price = Vol24hs/Price) %>%
  group_by(Key) %>% filter(Vol24hs >= 10^9) %>%
  ggplot(aes(Date, relation_vol_price, color = Key)) + geom_line() + scale_y_log10() +
  xlab("Date") + ylab("Price of Coin (Log10 scale") + ggtitle("Relation Price-Volumen in
```



In the visutalization, is posible to observe when all prices go up (by buy operations) or when some trader are moving between cryptocoins, the main usefull thing is to detect negative correlations In automatical small operations

We tray again with all cryptocoins (we take out the filter and set for better visualization)

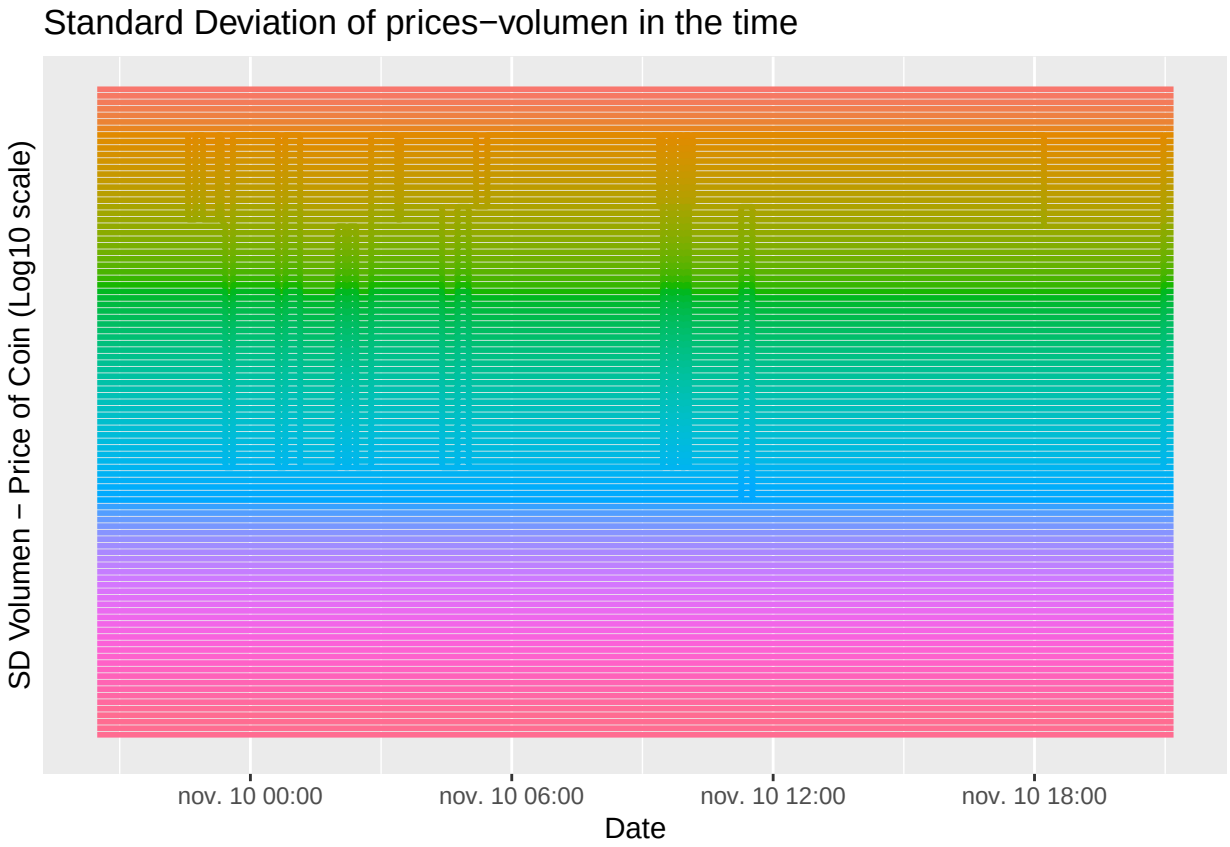
```
dset %>% mutate(relation_vol_price = Vol24hs/Price) %>%
  group_by(Key) %>%
  ggplot(aes(Date, relation_vol_price, color = Key)) +
  geom_line(show.legend = FALSE, position = "stack", size = 1) +
  scale_y_log10() +
  xlab("Date") +
  ylab("Relation Volumen - Price of Coin (Log10 scale)") +
  ggtitle("Relation prices-volumen in the time")
```



we can see two big groups of coins moving together.

We try again but filtered for sd.

```
dset %>% mutate(sd = sd(Vol24hs/Price)) %>%
  filter (sd > 50)%>%
  group_by(Key) %>%
  ggplot(aes(Date, sd, color = Key)) +
  geom_line(show.legend = FALSE, position = "stack", size = 1) +
  scale_y_log10() +
  xlab("Date") +
  ylab("SD Volumen - Price of Coin (Log10 scale)") +
  ggtitle("Standard Deviation of prices-volumen in the time")
```

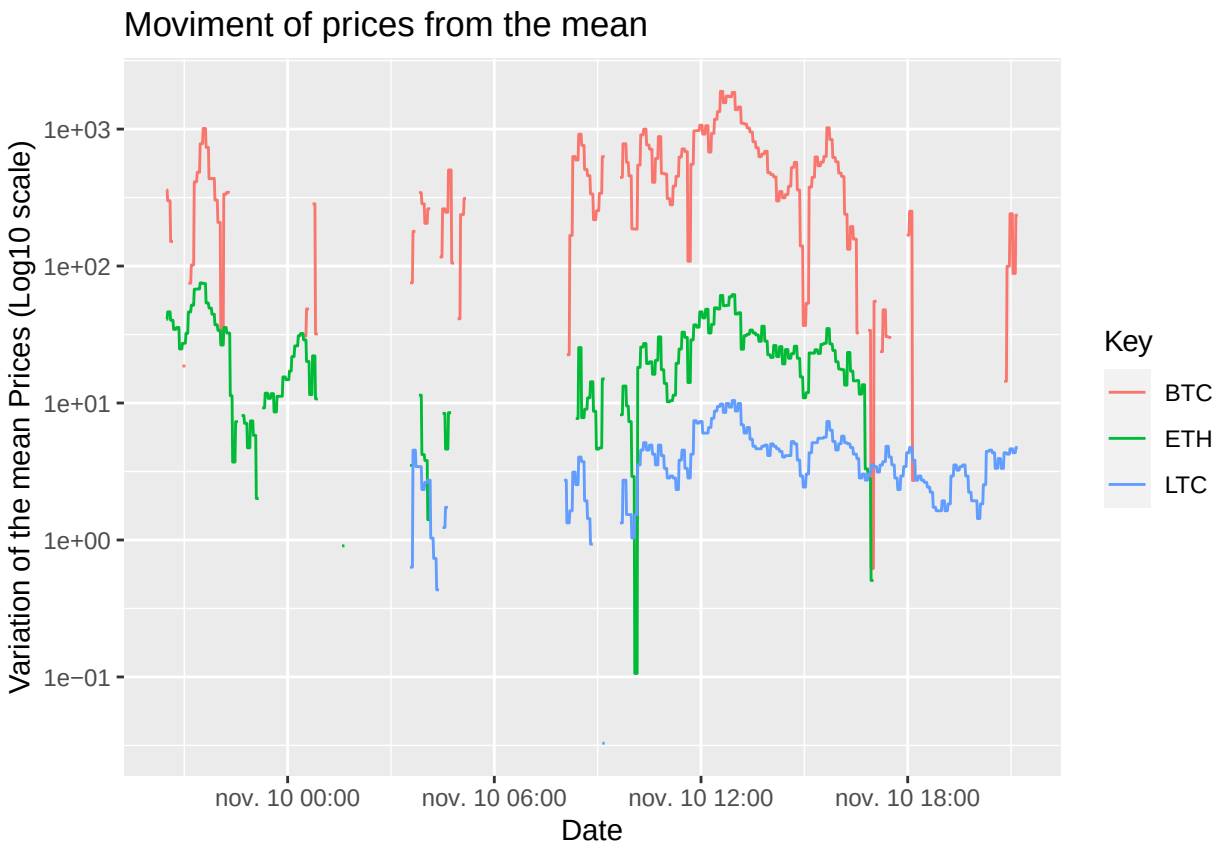


we observe al line, like a new coin that we have data for a short peroid of time, we investigate whats is it

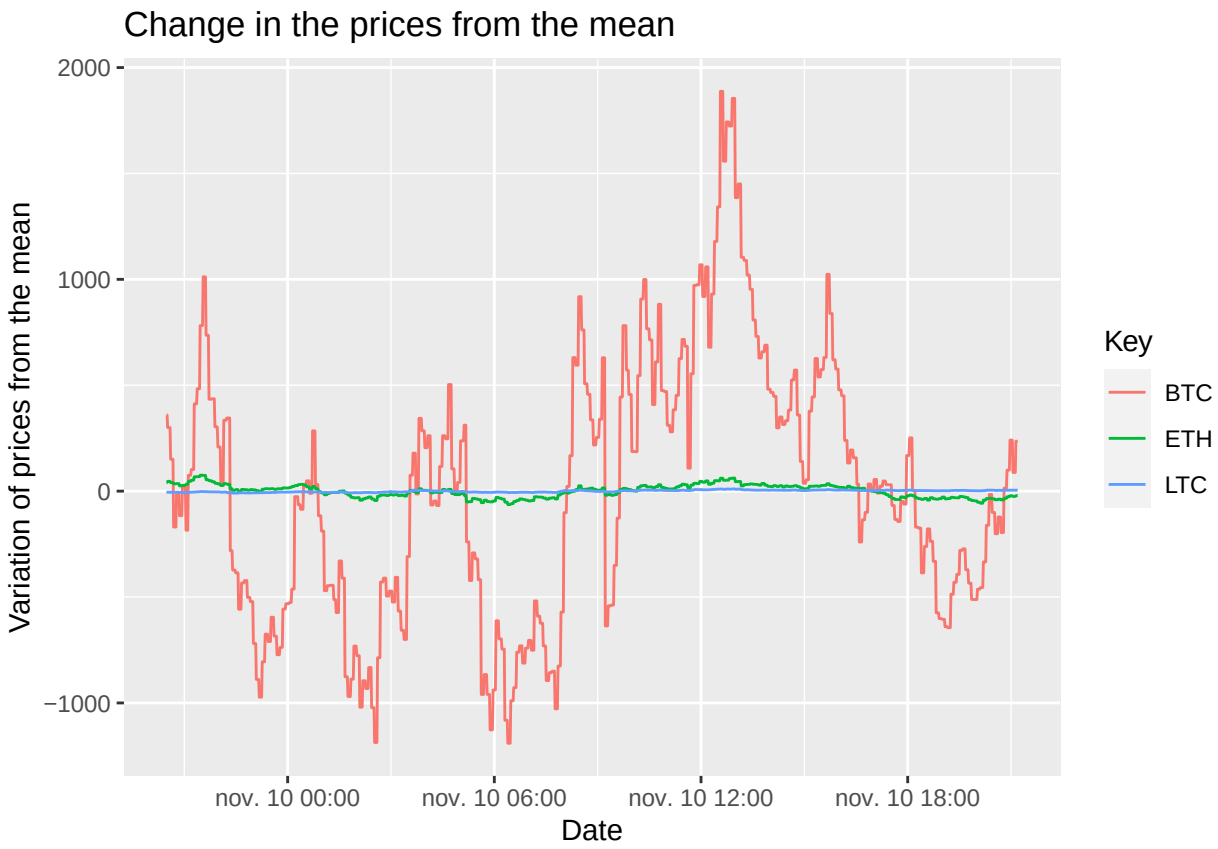
```
dset %>% group_by(Key, Vol24hs) %>% summarize(Price = mean(Price)) %>% arrange(desc(Vol
  head() %>% knitr::kable()
```

Key	Vol24hs	Price
USDT	48889741732	1
USDT	48858266675	1
USDT	48803429515	1
USDT	48737716015	1
USDT	48703237745	1
USDT	48136696001	1

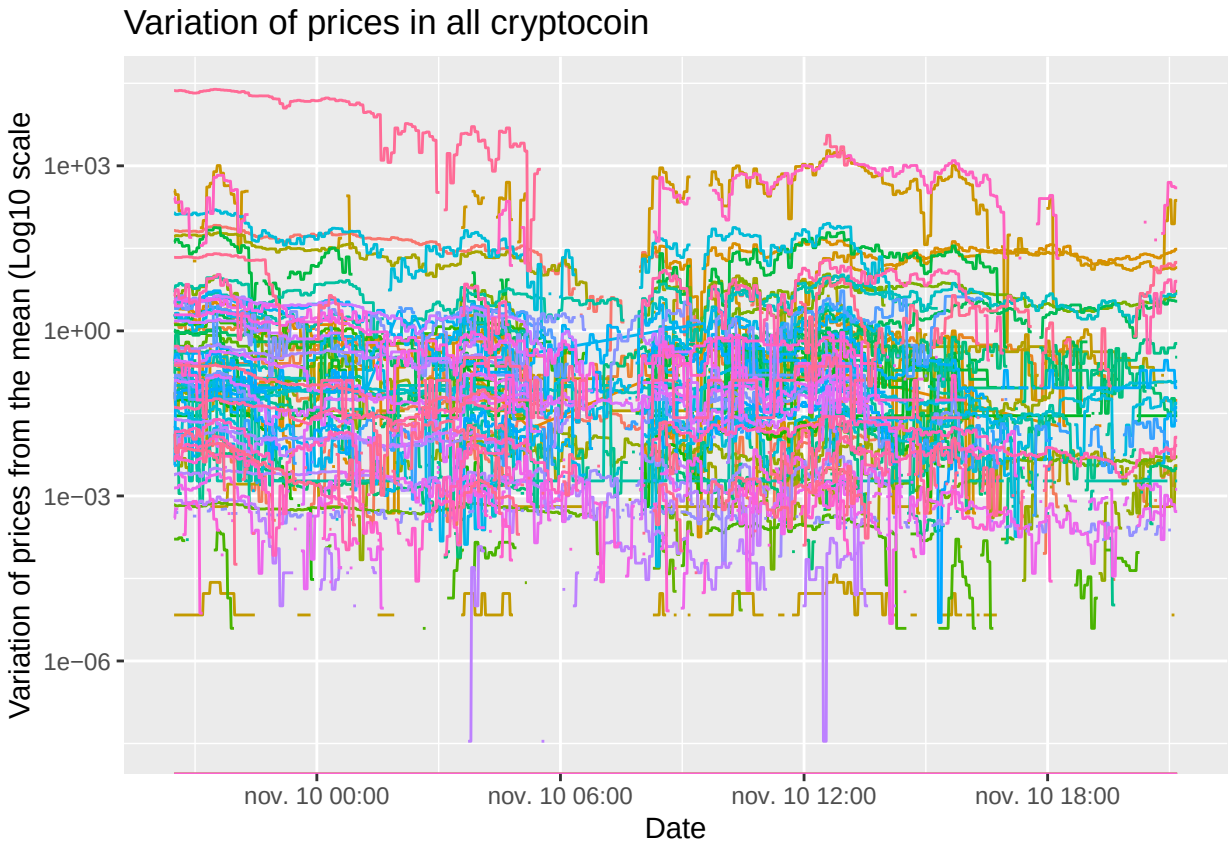
```
dset %>% group_by(Key) %>% mutate(dif = (mean(Price)-Price)*10) %>% filter(Key %in% c("
```




```
dset %>% group_by(Key) %>% mutate(dif = (mean(Price)-Price)*10) %>% filter(Key %in% c("
```



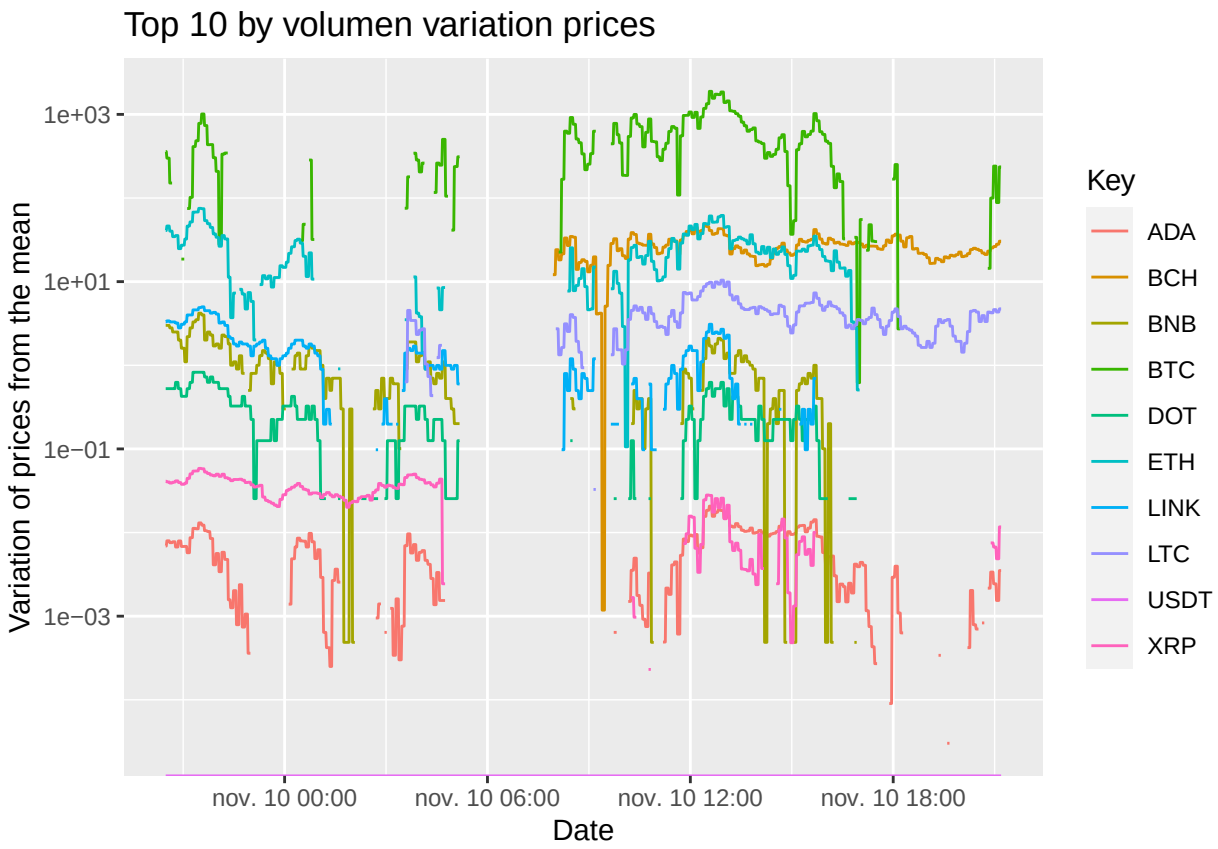

```
dset %>% group_by(Key) %>%  mutate(dif = (mean(Price)-Price)*10) %>% ggplot(aes(Date, di
  ggtitle("Variation of prices in all cryptocoin")
```




```
dset %>% group_by(Key) %>%  mutate(dif = (mean(Price)-Price)*10) %>% head() %>% knitr::
```

Name	Key	Price	Market_Cap	Vol24hs	Circulating	Date	dif
Bitcoin	BTC	15352.470000	284609678926	26262125541	18538368	2020-11-09 22:31:24	-385.7809028
Ethereum	ETH	446.670000	50635045504	13896166591	113361345	2020-11-09 22:31:24	7.3059722
Tether	USDT	1.000000	17351235367	46376267017	17340690024	2020-11-09 22:31:24	0.0000000
XRP	XRP	0.250291	11341310059	3112230771	45312488820	2020-11-09 22:31:24	0.0431008
Chainlink	LINK	12.670000	4961708119	1619586853	391509556	2020-11-09 22:31:24	1.7976389
Bitcoinash	BCH	264.590000	4912850732	1895163320	18567813	2020-11-09 22:31:24	-45.1988194

```
dset %>% group_by(Key) %>% mutate(dif = (mean(Price)-Price)*10) %>% filter(Key %in% c("
  ylab("Variation of prices from the mean") + ggtitle("Top 10 by volumen variation price
```



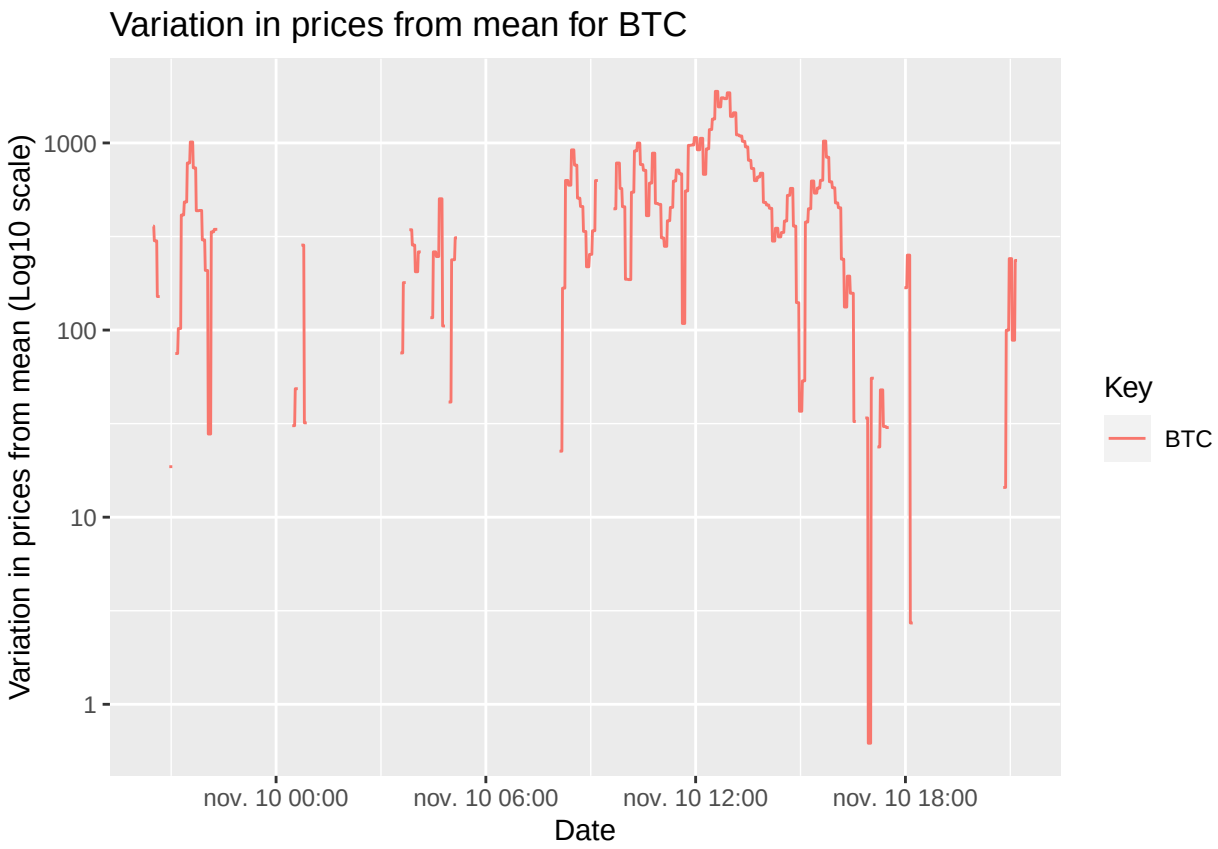

```
top20 <- dset %>% group_by(Key) %>% summarize(sum = sum(Market_Cap)) %>% arrange(desc(s
top20 <- top20 %>% top_n(20)
top20 %>% knitr::kable()
```

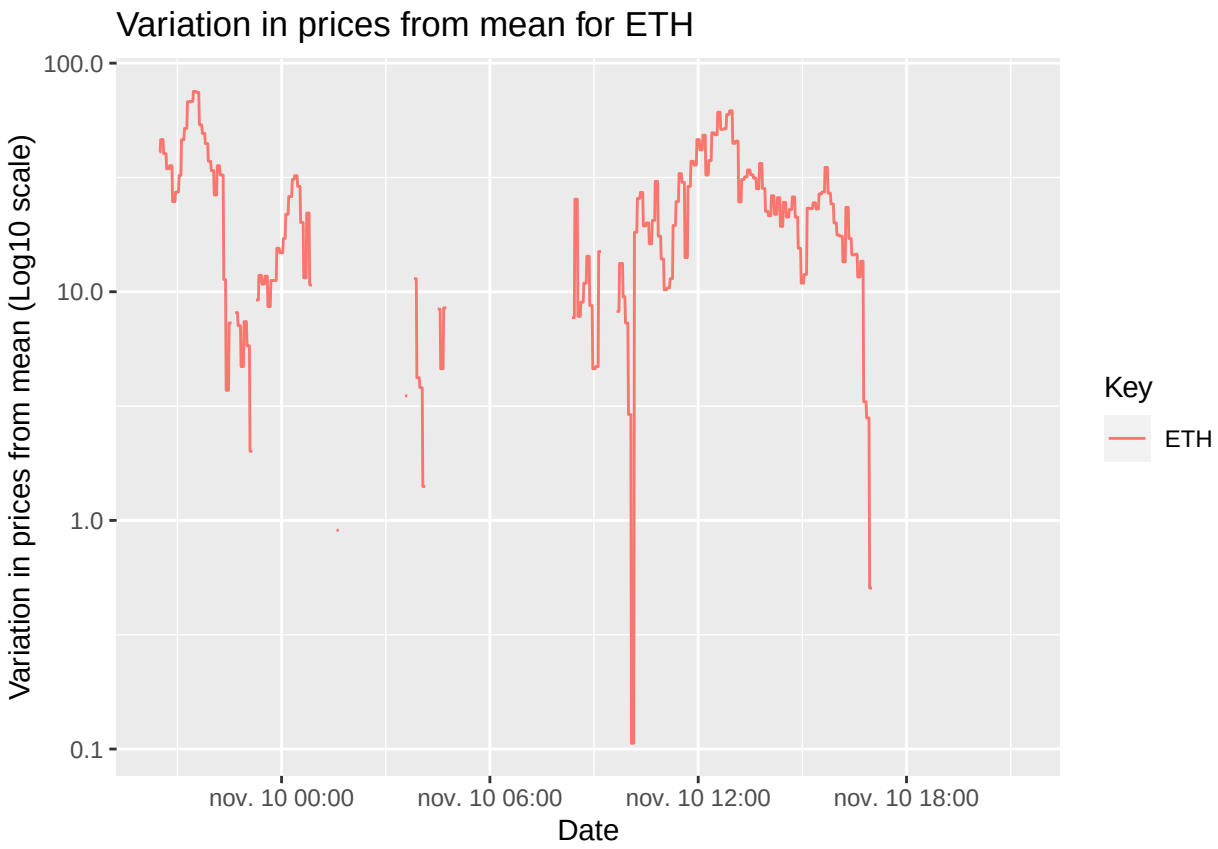
Key	sum
BTC	4.088082e+14
ETH	7.303761e+13
USDT	2.498534e+13
XRP	1.661272e+13
LINK	7.244355e+12
BCH	6.953828e+12
BNB	5.849524e+12
LTC	5.566521e+12
DOT	5.393247e+12
ADA	4.749414e+12
BSV	4.240531e+12
USDC	4.085904e+12
EOS	3.379511e+12
XMR	2.970416e+12
WBTC	2.716981e+12

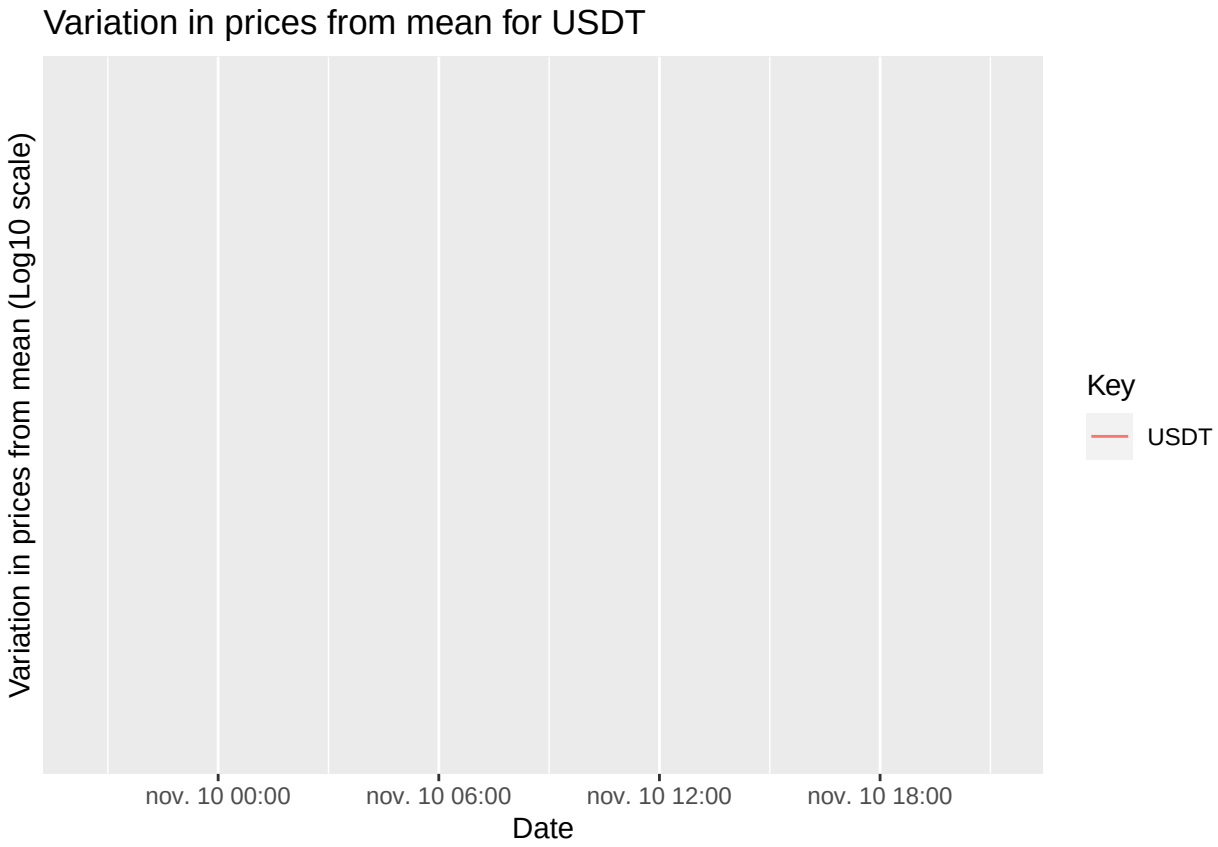
Key	sum
TRX	2.575509e+12
XLM	2.425748e+12
XTZ	2.302175e+12
CRO	2.146933e+12
LEO	1.855630e+12

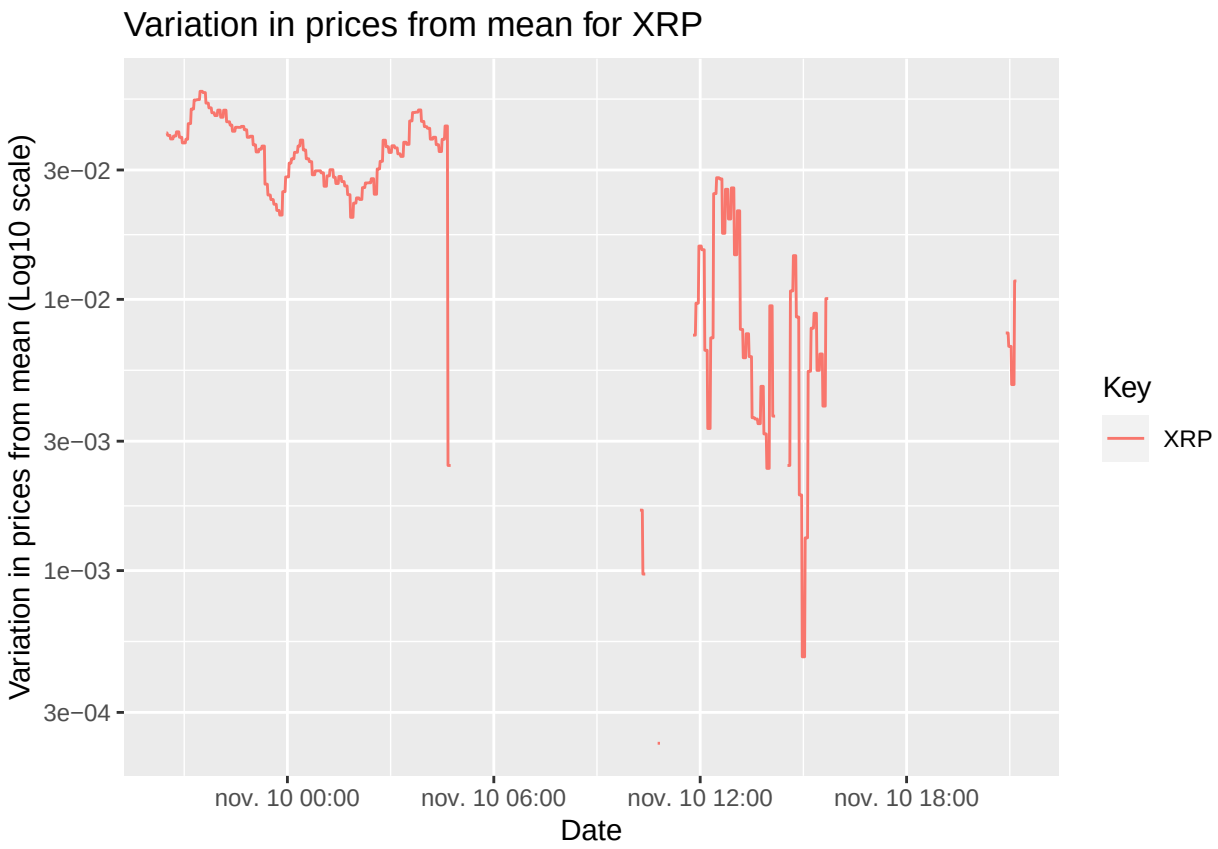
```
dset %>% group_by(Key) %>% mutate(dif = (mean(Price)-Price)*10) %>% filter(Key %in% top
```

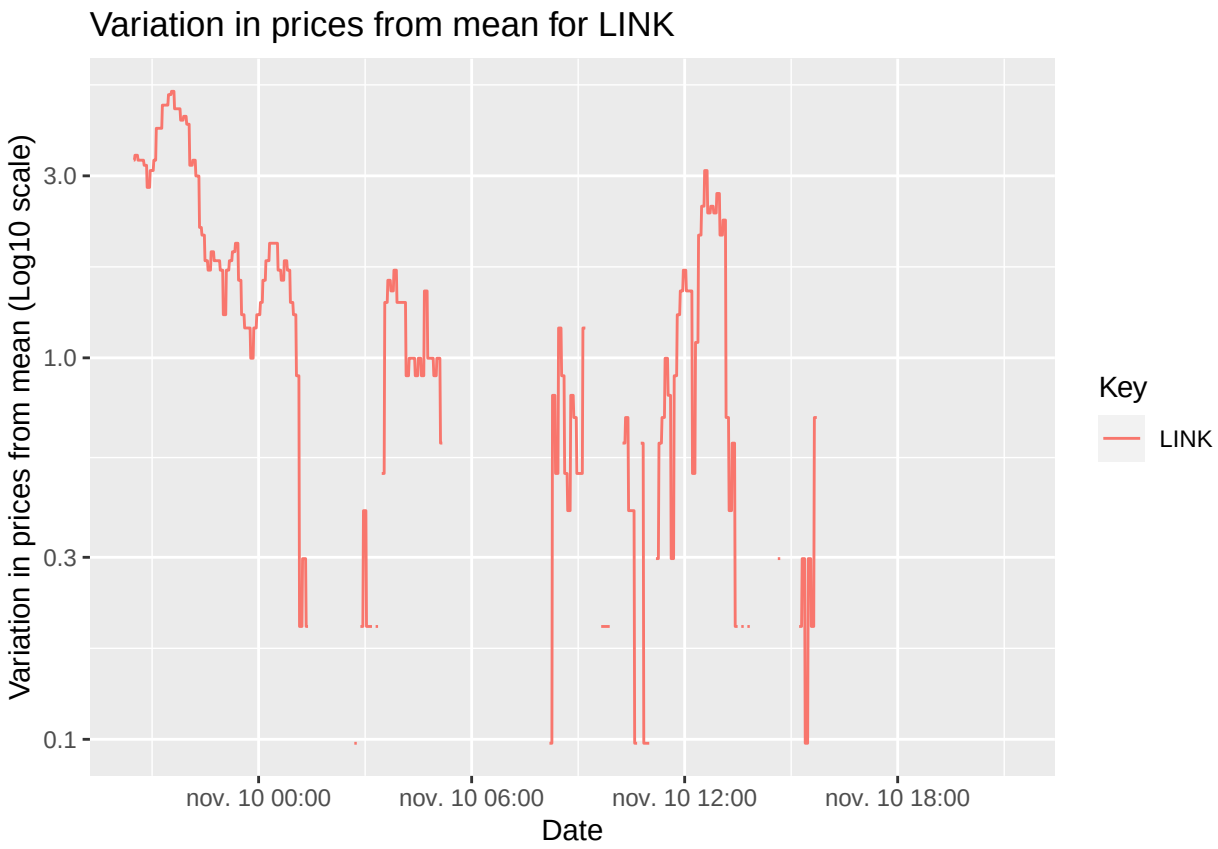
We will see the diference of price one by one, only for the top 10

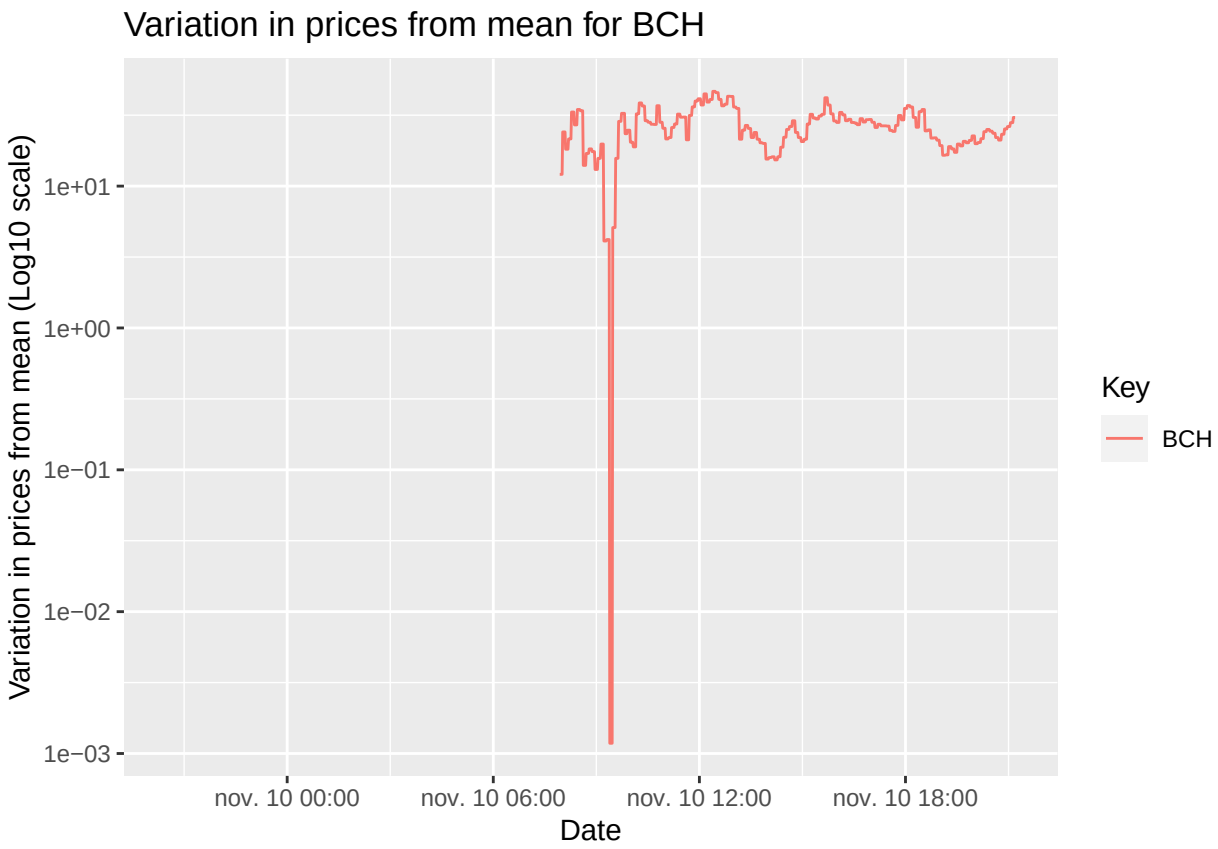

Variation in prices from mean for BTC
Variation in prices from mean (Log10 scale)
1000
100
10
1
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BTC

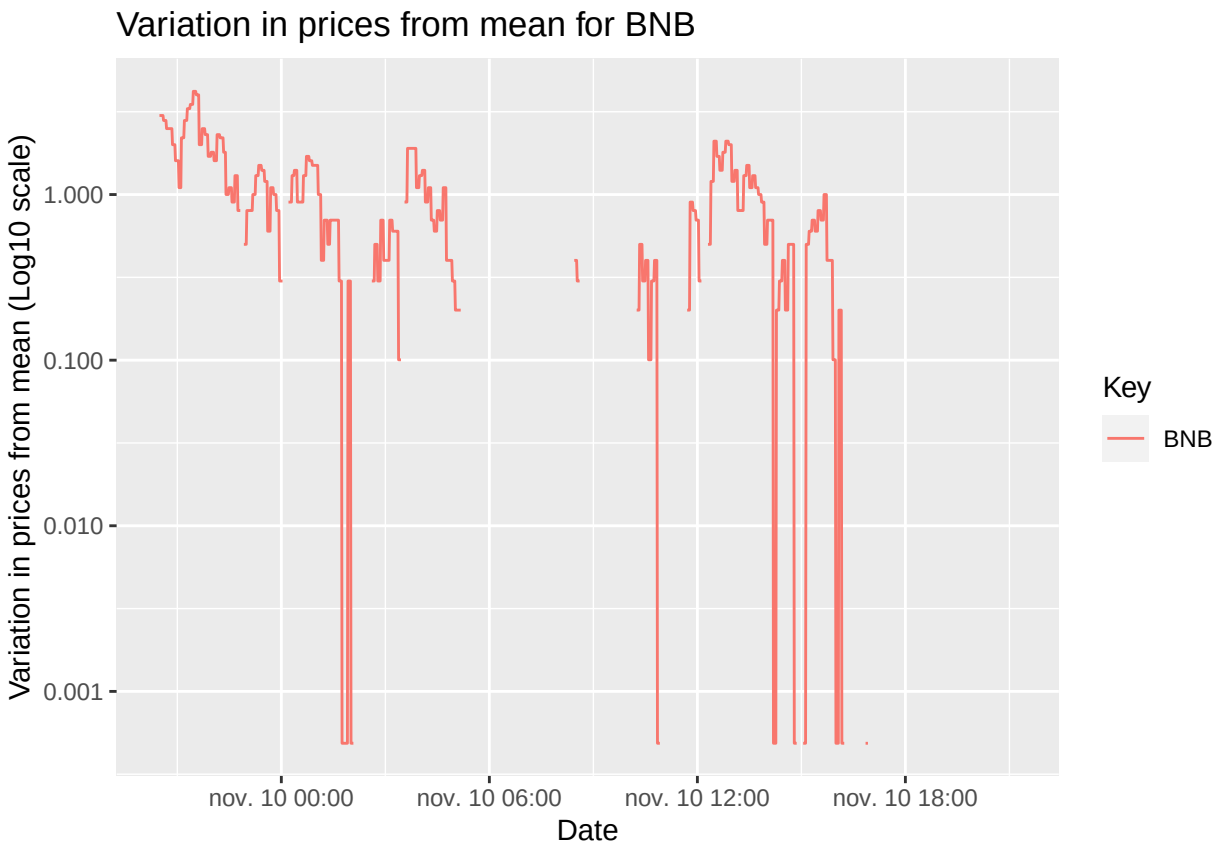

Variation in prices from mean for ETH
Variation in prices from mean (Log10 scale)
100.0
10.0
1.0
0.1
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
ETH

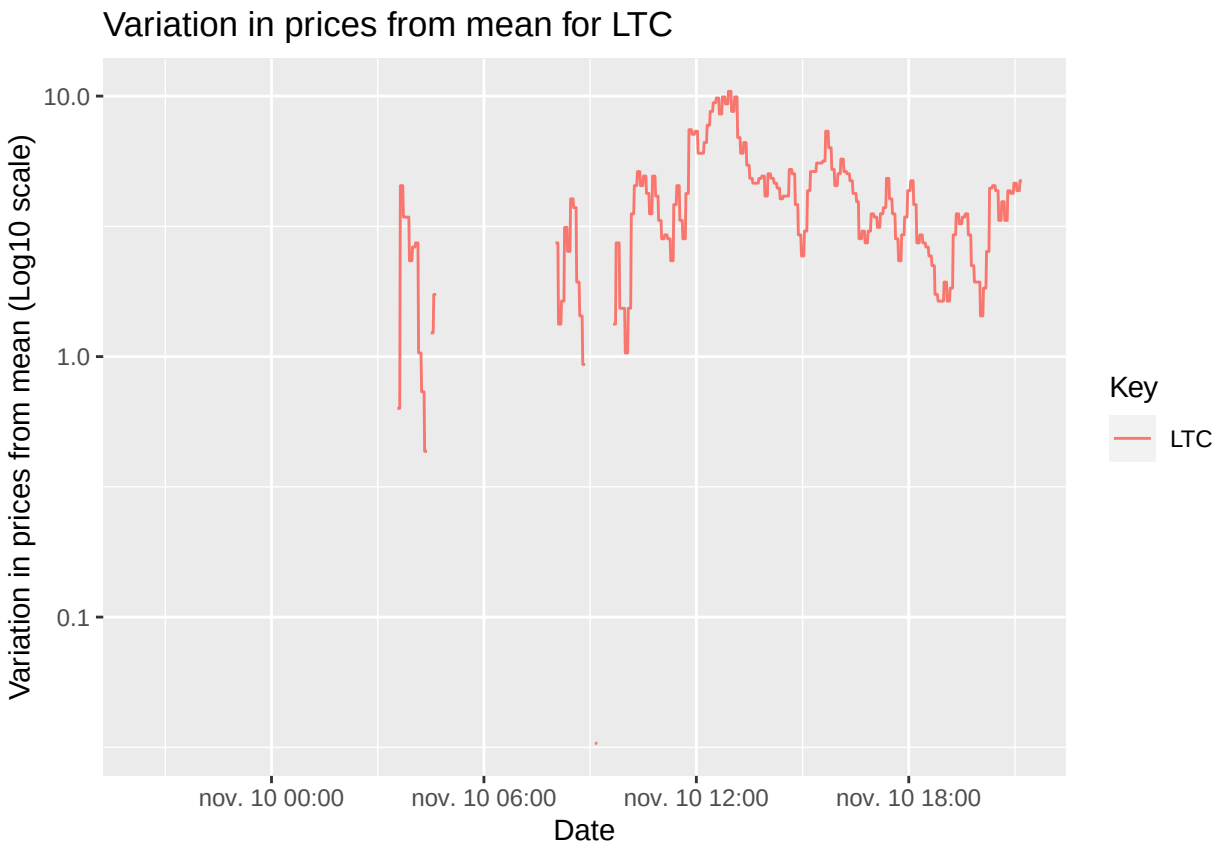

Variation in prices from mean for USDT
Variation in prices from mean (Log10 scale)
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
USDT

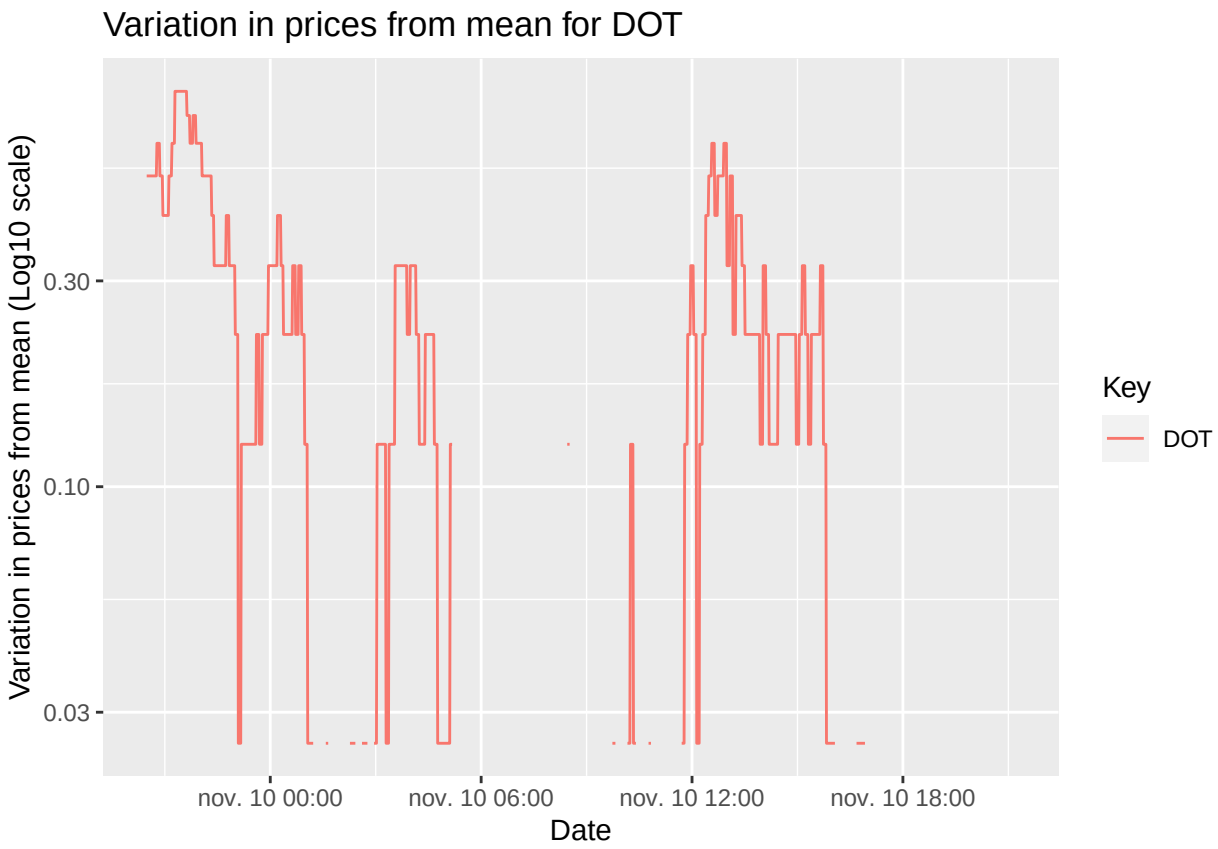

Variation in prices from mean for XRP
Variation in prices from mean (Log10 scale)
3e−02
1e−02
3e−03
1e−03
3e−04
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
XRP

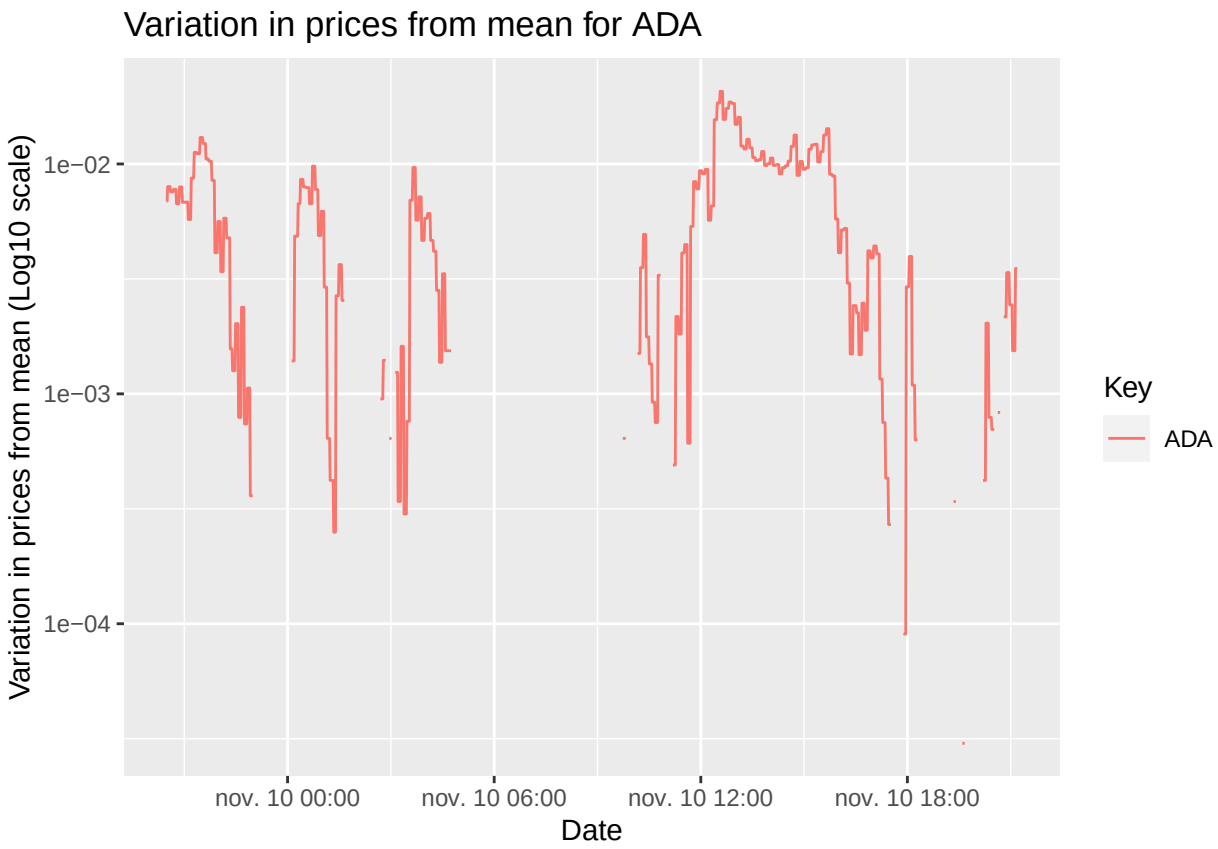

Variation in prices from mean for LINK
Variation in prices from mean (Log10 scale)
3.0
1.0
0.3
0.1
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
LINK


Variation in prices from mean for BCH
Variation in prices from mean (Log10 scale)
1e+01
1e+00
1e−01
1e−02
1e−03
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BCH

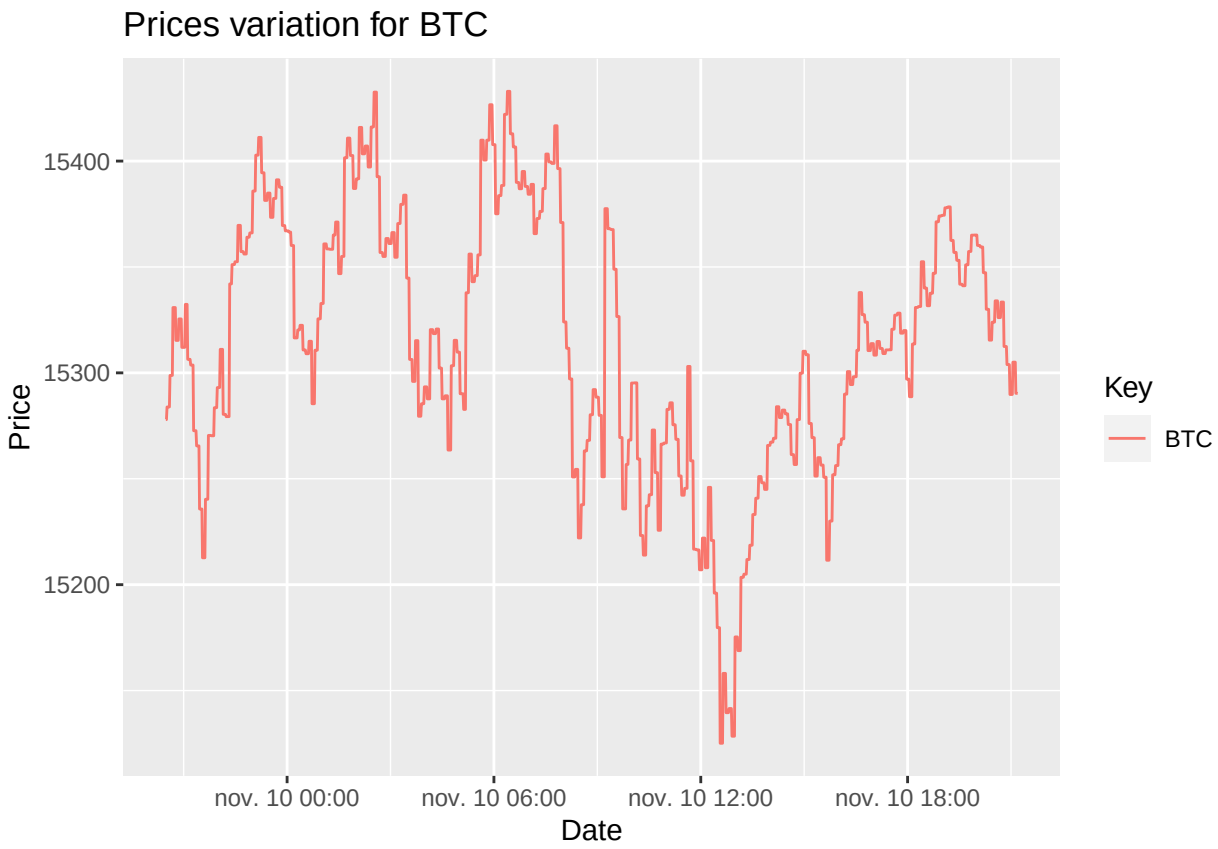

Variation in prices from mean for BNB
Variation in prices from mean (Log10 scale)
1.000
0.100
0.010
0.001
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BNB

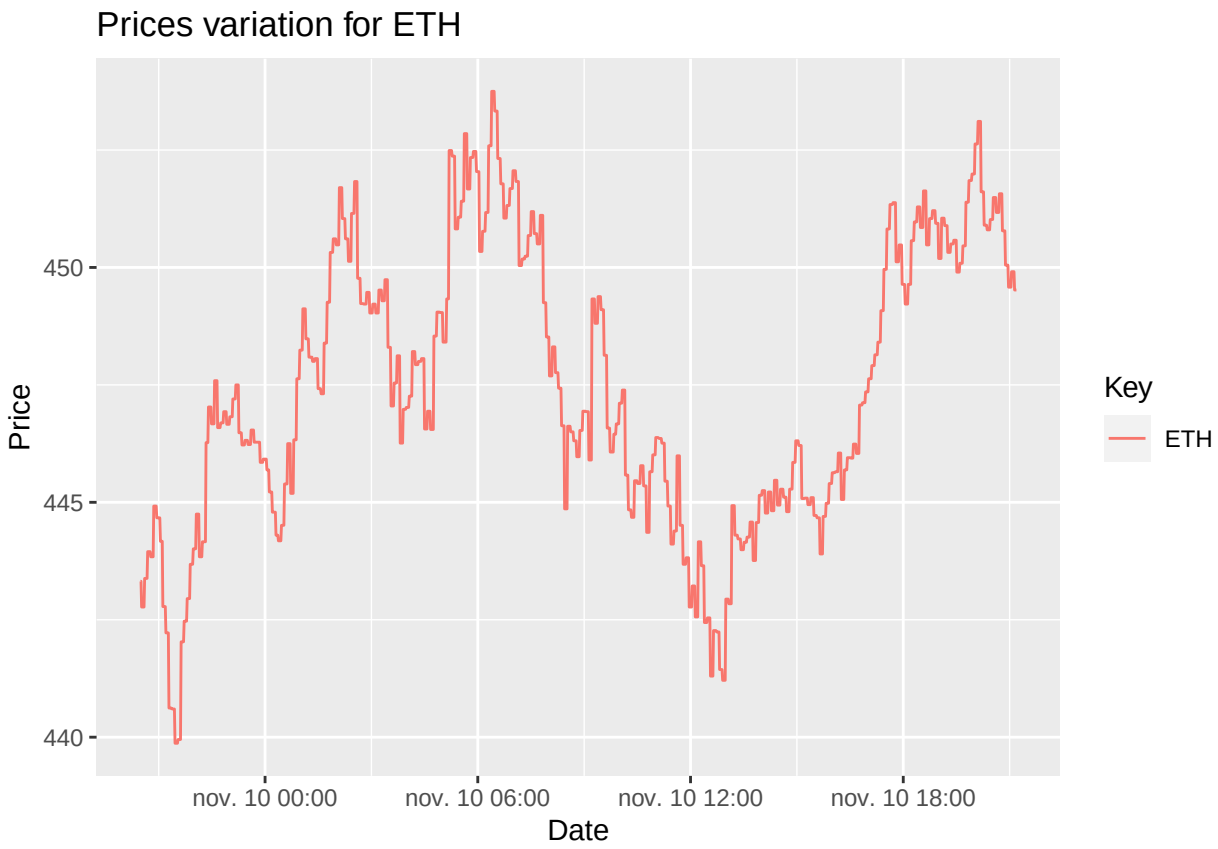

Variation in prices from mean for LTC
Variation in prices from mean (Log10 scale)
10.0
1.0
0.1
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
LTC

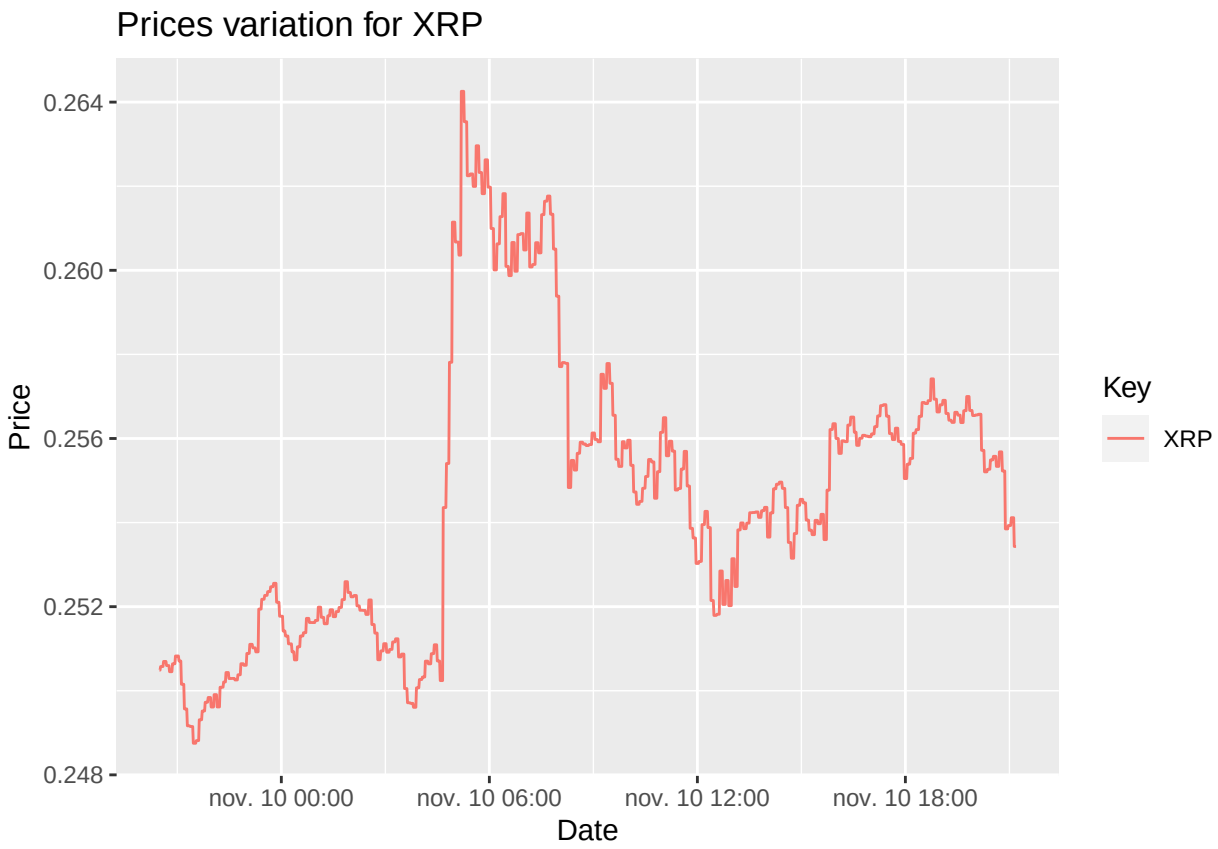

Variation in prices from mean for DOT
Variation in prices from mean (Log10 scale)
0.30
0.10
0.03
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
DOT

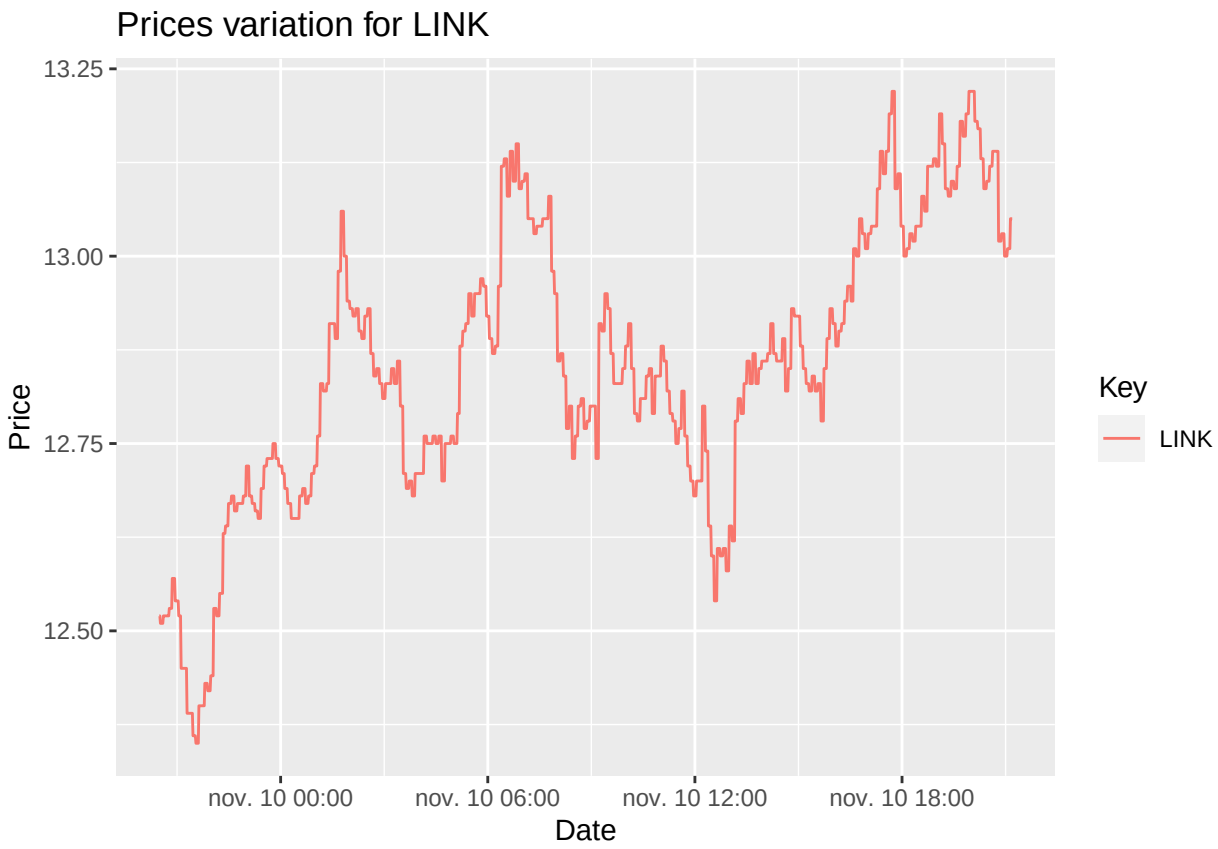


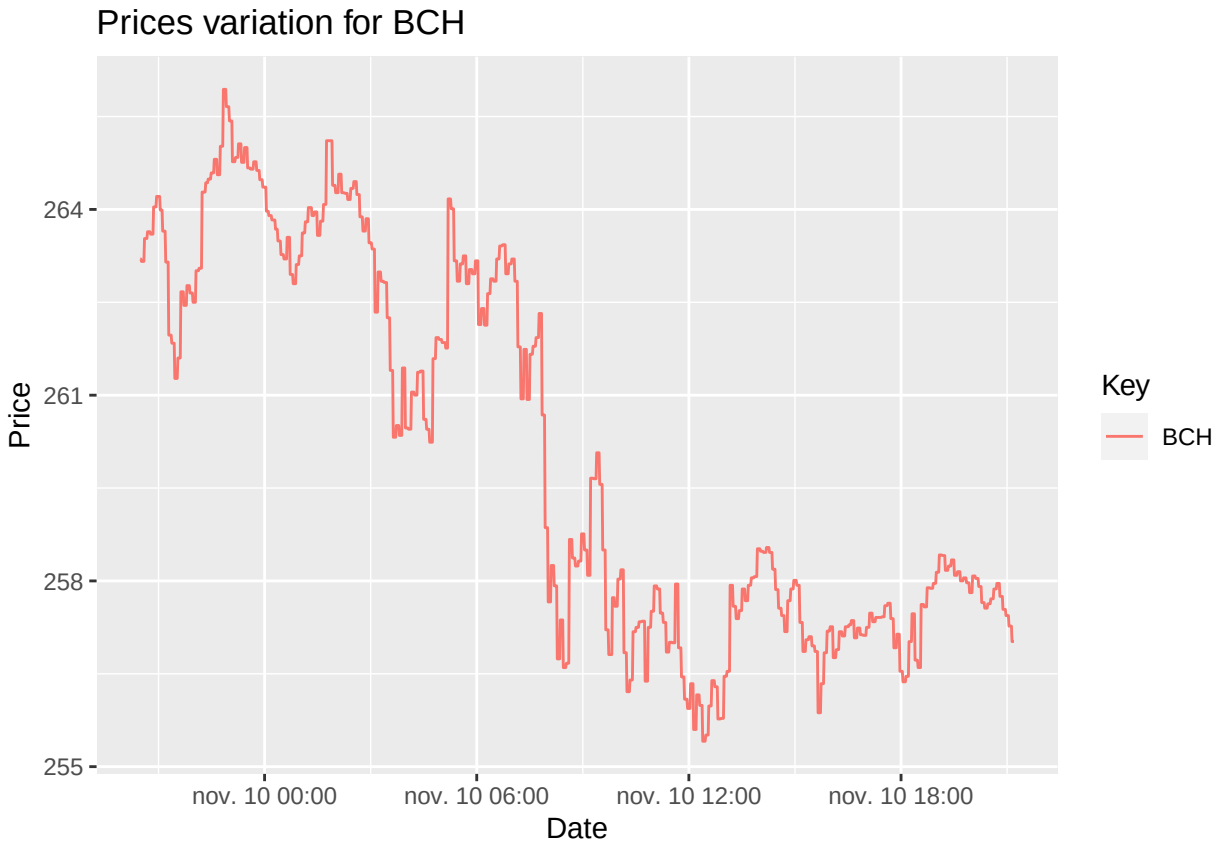
Now, we observe the real price one by one for the top 10

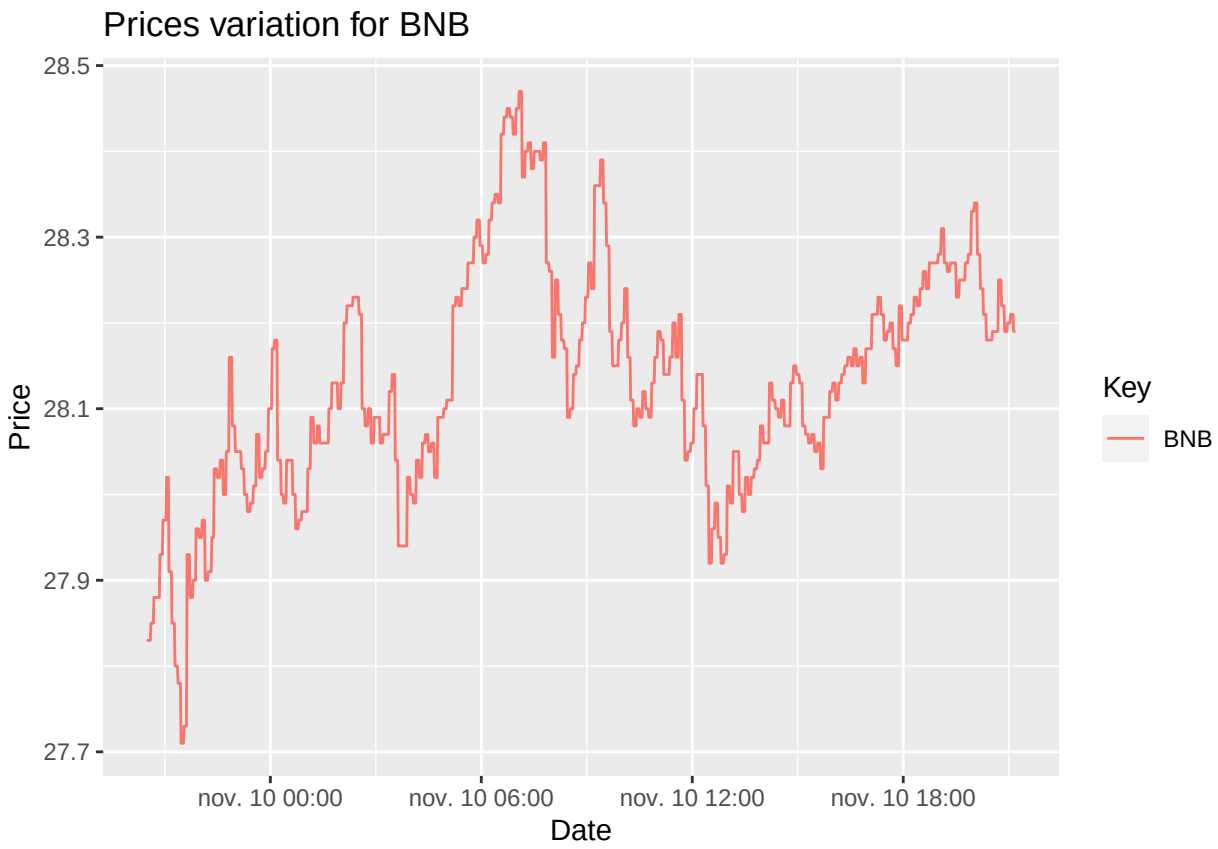

Prices variation for BTC
15400
15300
15200
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BTC

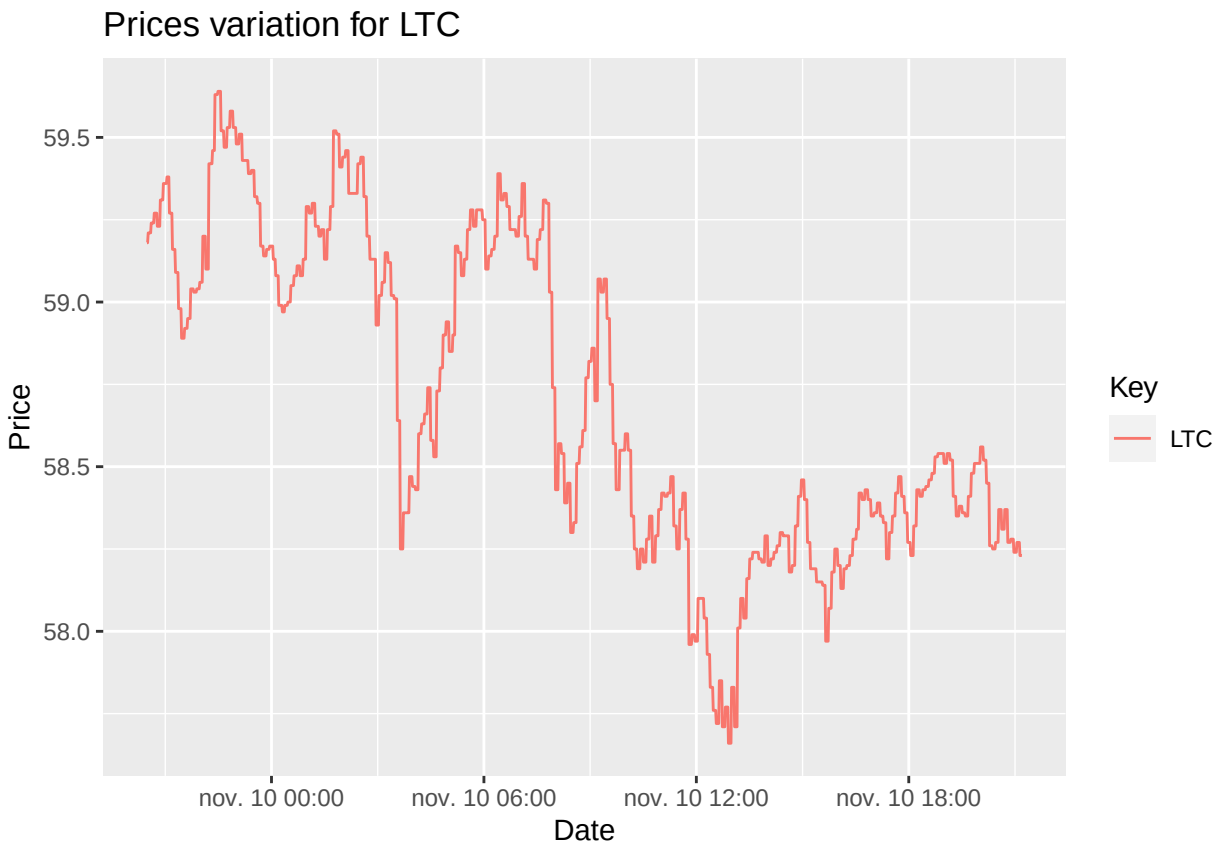

Prices variation for ETH
Price
450
445
440
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
ETH

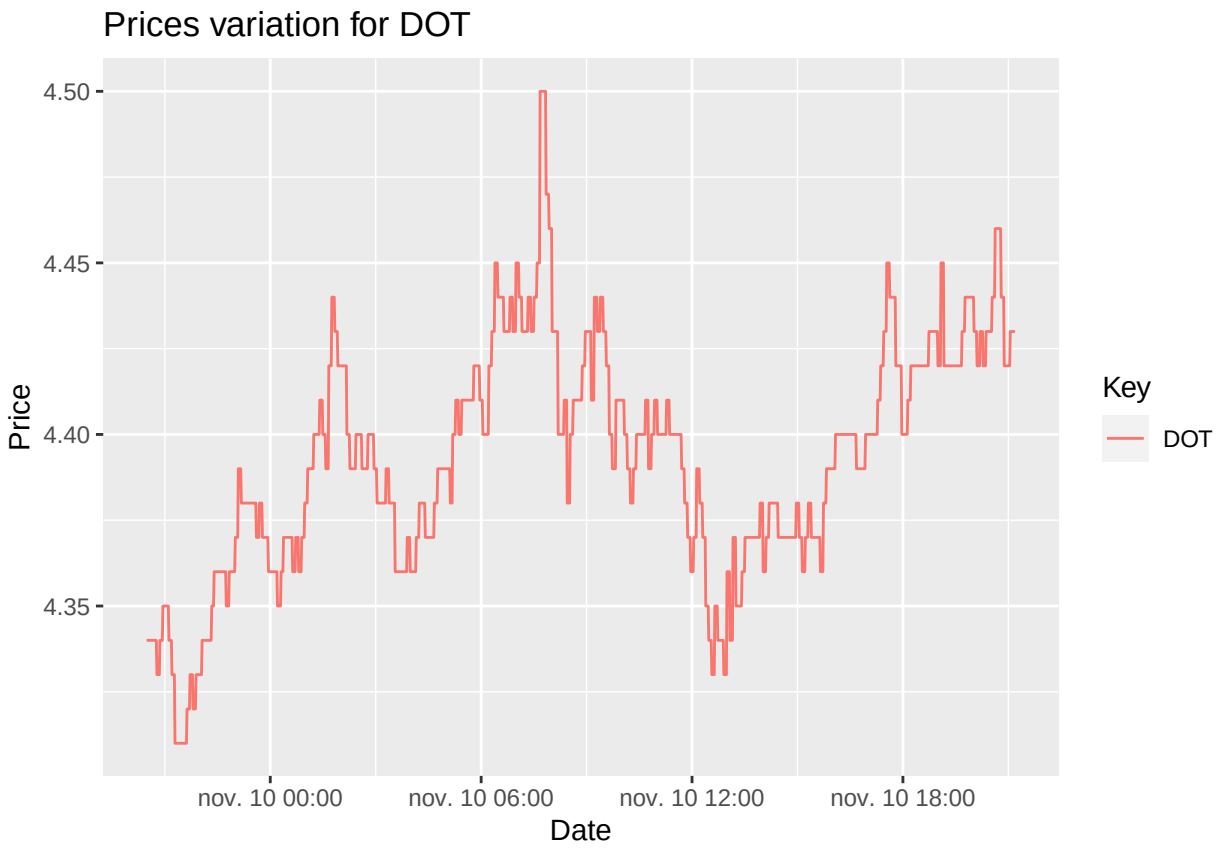

Prices variation for XRP
0.264
0.260
0.256
0.252
0.248
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
XRP

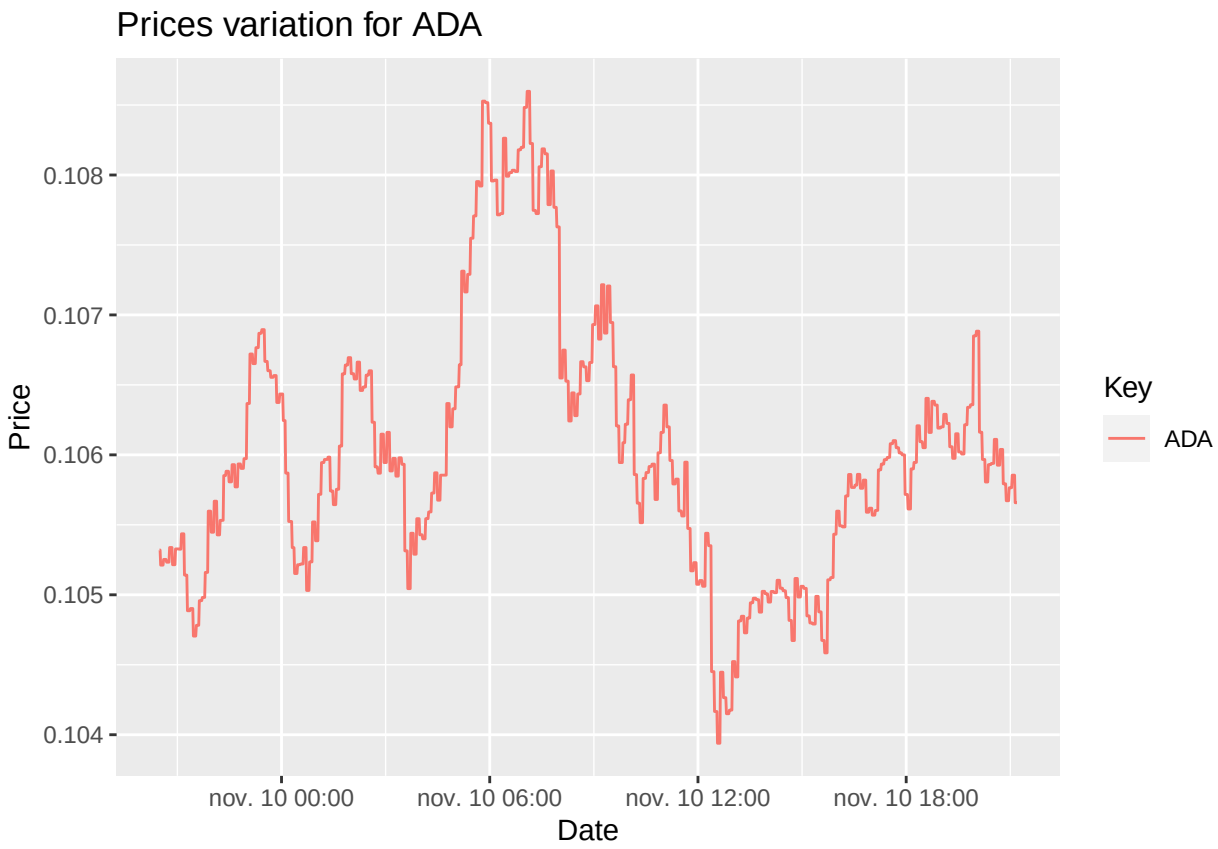

Prices variation for LINK
13.25
13.00
12.75
12.50
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
LINK


Prices variation for BCH
Price
264
261
258
255
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BCH


Prices variation for BNB
28.5
28.3
28.1
27.9
27.7
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
BNB

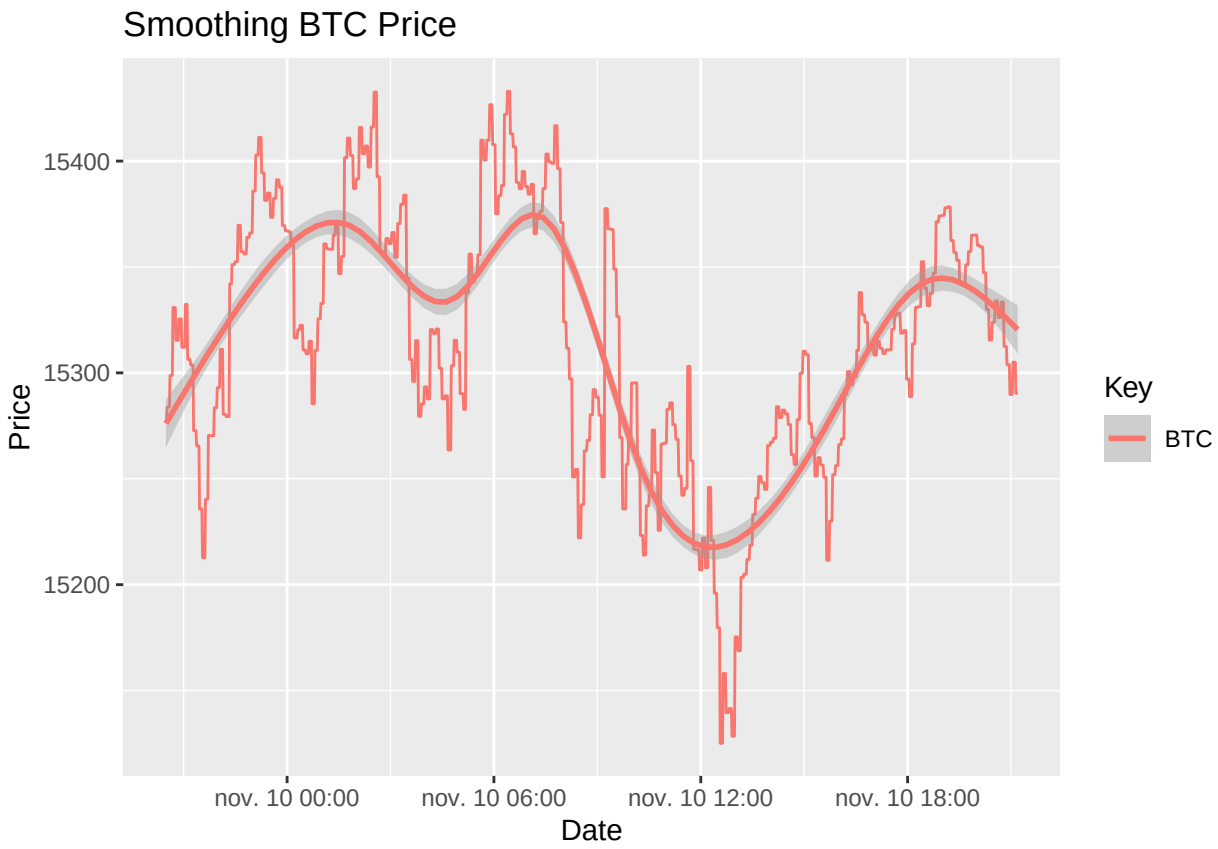

Prices variation for LTC
59.5
59.0
58.5
58.0
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
LTC


Prices variation for DOT
4.50
4.45
4.40
4.35
Price
nov. 10 00:00
nov. 10 06:00
nov. 10 12:00
nov. 10 18:00
Date
Key
DOT



We take out the graphic of USDT because is not variation.

```
dset %>% group_by(Key) %>% filter(Key == top20$Key[1]) %>% ggplot(aes(Date, Price, col=
```

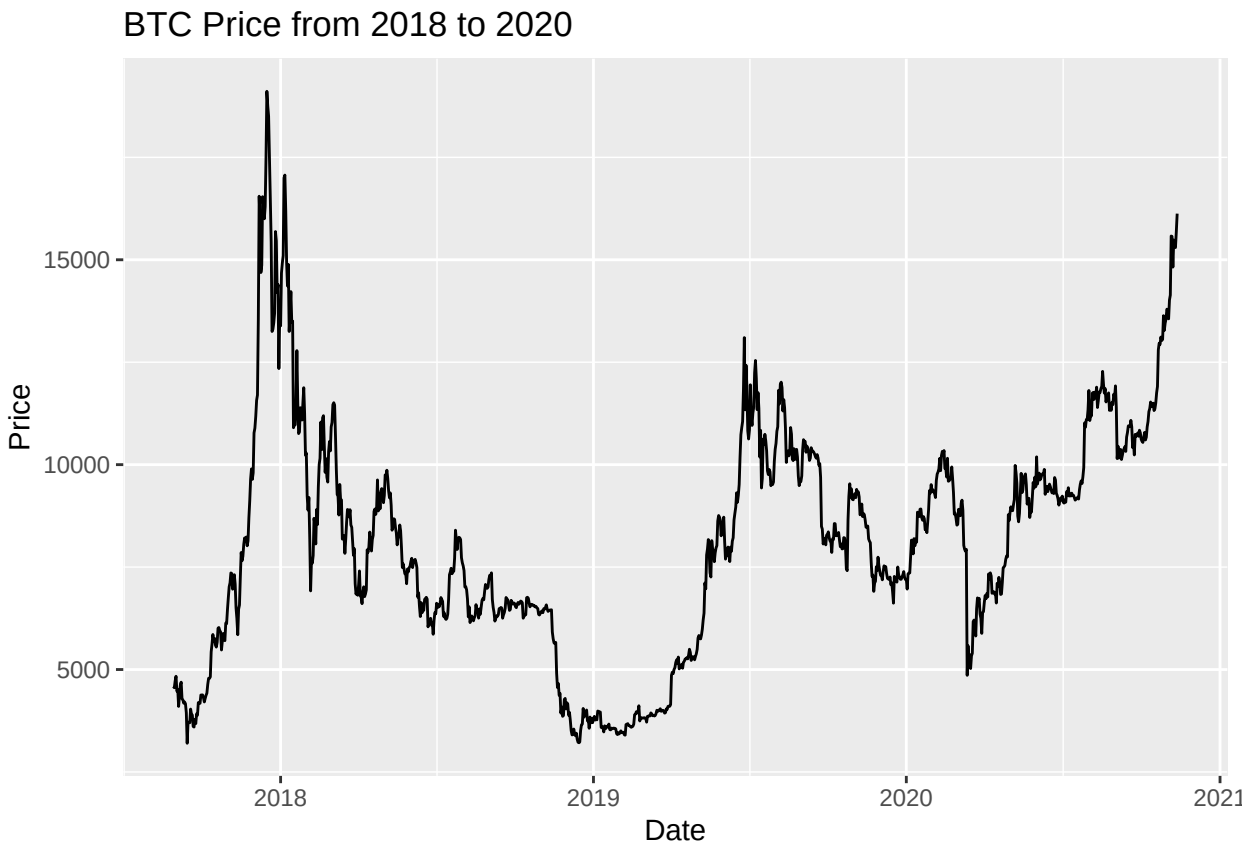


Looking the variation and volumen of the prices in the top 10 crypto coins, we will concentrate the esforces to predict the price of the bitcoin, **but** we need also a pair of other more stable coin for trade in a future.

2.2 Historical BTC data

For the historical data of BTC price i upload also a csv file in GitHub, this data was provided for Investing.com

```
url <- "https://raw.githubusercontent.com/Avantas/Capstone/main/BTC_USD%20Binance%20Hist
historical <- read_csv(url)
historical$Date <- mdy(historical$Date)
historical %>% ggplot(aes(Date, Price)) + geom_line() + ggtitle("BTC Price from 2018 to
```



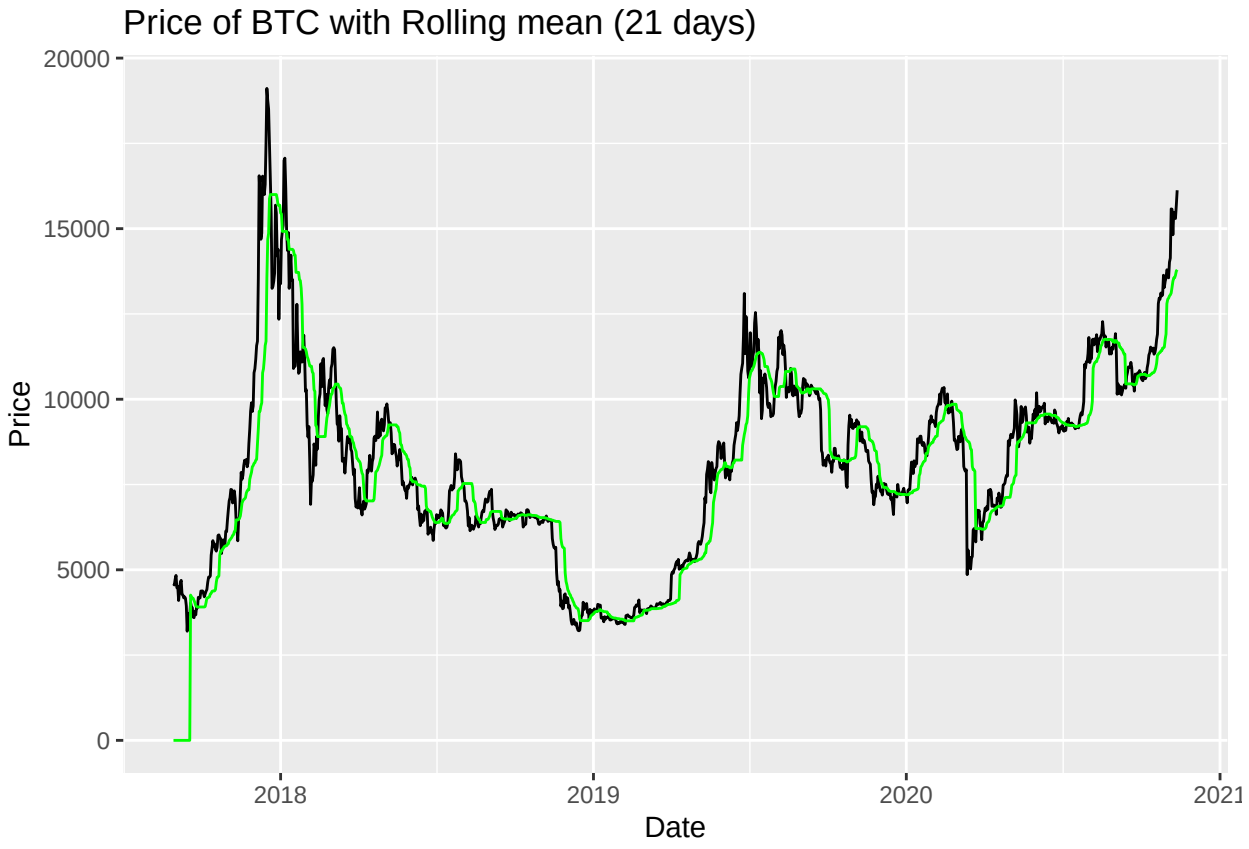
2.3 Rolling mean

We made the function of movil mean and set it in 21 days (Its knowed that there are patterns of moviments in 21 days)

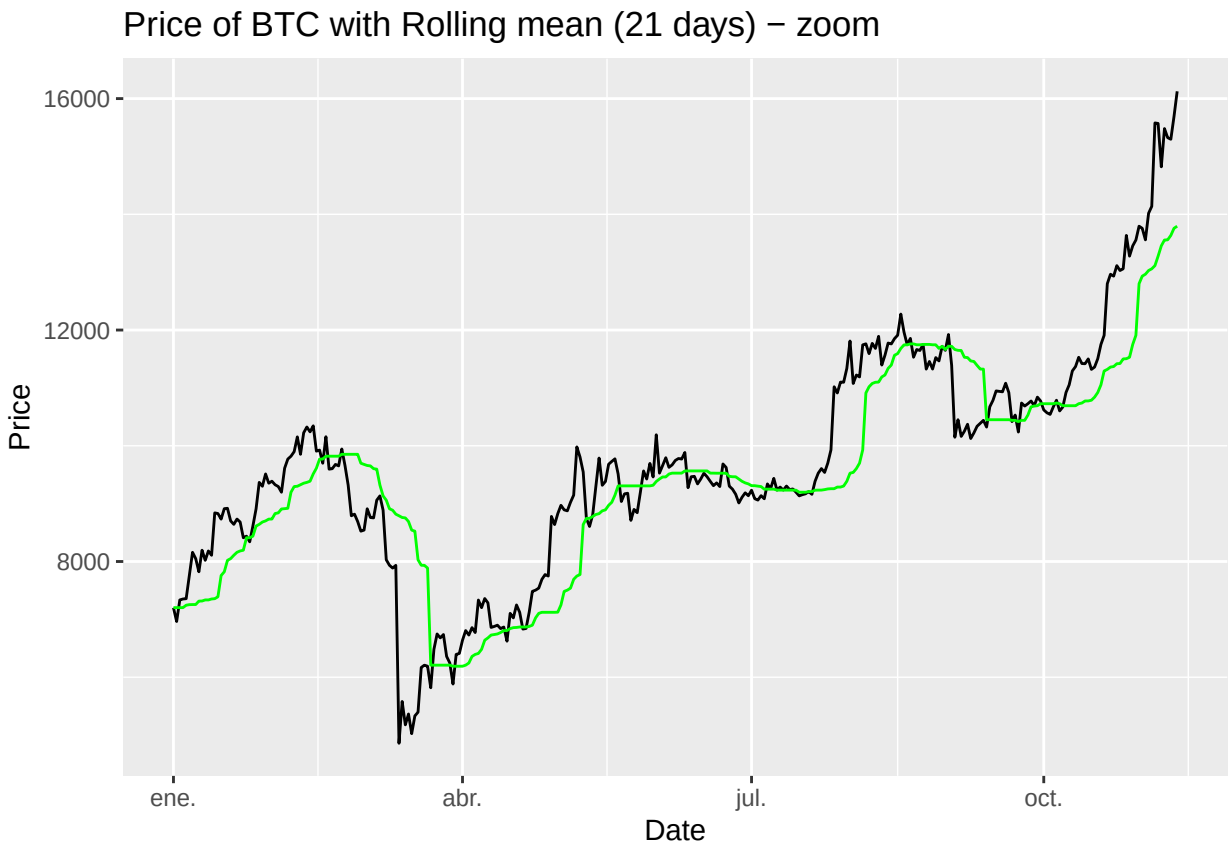
```
rollingmean <- rollmedian(historical$Price, 21)
head(rollingmean) %>% knitr::kable()
```

x
13794.5
13758.1
13635.0
13558.7
13556.1
13455.2

```
rollingmean[1153:1172] <- 0
historical <- historical %>% mutate(ma = rollingmean)
historical %>% ggplot(aes(Date, Price)) + geom_line() + geom_line(aes(Date, ma), col =
```




```
historical %>% filter(Date >= "2020-01-01") %>% ggplot(aes(Date, Price)) + geom_line() +
```



2.4 KDJ Trend Indicator

The KDJ consists of 3 lines (K, D and J - hence the name of the indicator) and 2 levels. The K and the D are the same lines that you see when using the Stochastic Oscillator. The line J, on the other hand, represents the divergence of the value D from the K. The convergence of these lines is a sign of emerging trading opportunities. As with the Stochastic Oscillator, the overbought and oversold levels correspond to the times when the trend is about to reverse. By default, these levels were obtained at 20% and 80%. Both can be adjusted for additional sensitivity / fewer false alerts.

```
KDJ <- historical %>% select(High, Low, Price)
KDJ <- KDJ %>% stoch()
head(KDJ) %>% knitr::kable()
```

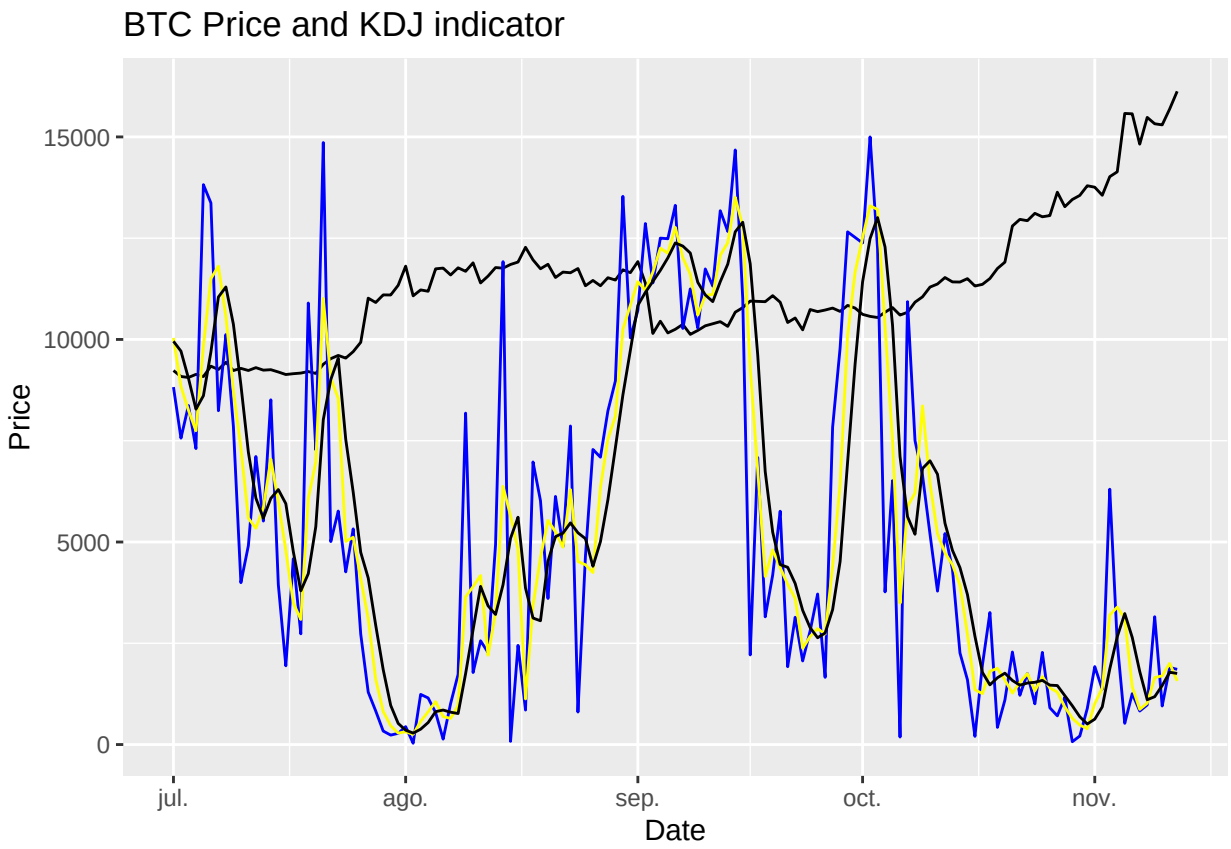
fastK	fastD	slowD
NA	NA	NA
NA	NA	NA
NA	NA	NA
NA	NA	NA

fastK	fastD	slowD
NA	NA	NA
NA	NA	NA

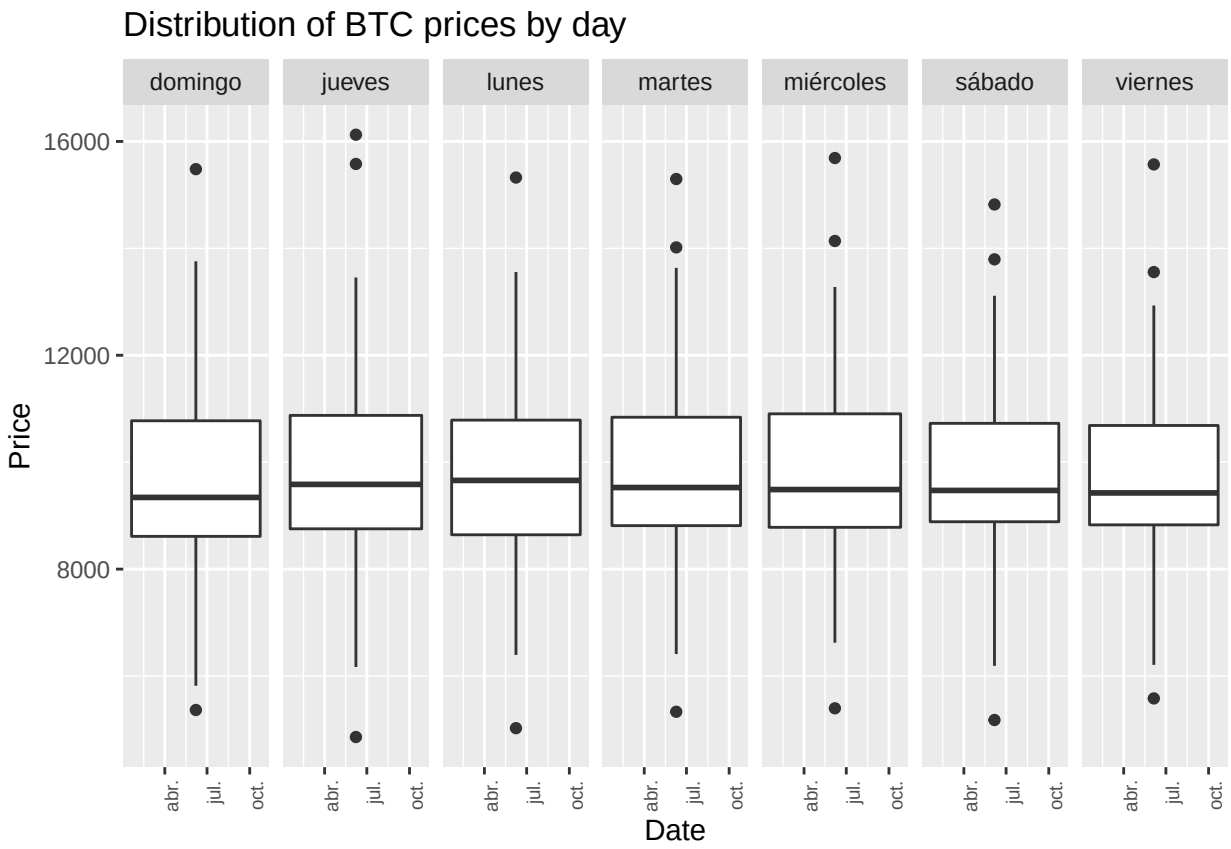
```
KDJ_matrix_K <- matrix(KDJ[-1:-13,1],(839+333-13),1)
KDJ_matrix_D <- matrix(KDJ[-1:-15,2],(839+333-15),1)
KDJ_matrix_J <- matrix(KDJ[-1:-17,3],(839+333-17),1)
KDJ_matrix_D[1158:1159] <- 0
KDJ_matrix_J[1156:1159] <- 0
KDJ_matrix_K[1160:1172] <- 0
KDJ_matrix_J[1160:1172] <- 0
KDJ_matrix_D[1160:1172] <- 0
KDJ_matrix <- matrix(c(KDJ_matrix_K, KDJ_matrix_D, KDJ_matrix_J), 1172,3)
head(KDJ_matrix) %>% knitr::kable()
```

0.1237767	0.1046671	0.1170639
0.1268054	0.1335223	0.1189264
0.0634191	0.1130022	0.0971097
0.2103422	0.1102546	0.0787598
0.0652453	0.0680723	0.0737800
0.0551764	0.0579526	0.1202393

```
historical %>% mutate(K = KDJ_matrix[,1] , D = KDJ_matrix[,2], J = KDJ_matrix[,3]) %>%
  filter(Date >= "2020-07-01") %>% ggplot(aes(Date, Price)) + geom_line() +
  geom_line(aes(Date, K*15000), col = "blue") +
  geom_line(aes(Date, D*15000), col = "yellow") +
  geom_line(aes(Date, J*15000), col = "black") +
  ggtitle("BTC Price and KDJ indicator")
```




```
historical  %>% mutate(day = weekdays(historical$Date)) %>%
  filter(Date >= "2020-01-01") %>% group_by(day) %>%
  ggplot(aes(Date, Price)) + geom_boxplot() +
  facet_grid(. ~ day) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 7)) +
  ggtitle("Distribution of BTC prices by day")
```



2.5 We explore the dataset from kaggle

We can acces to a dataset of kaggle minute by minute, but, this data set have information only until 14 th of september this year. Indeed, is interesting for the future analysis or train the algorithm in the short term.

```
url1 <- "https://www.kaggle.com/mczielinski/bitcoin-historical-data?select=bitstampUSD_1
ds_k <- read_csv(url1)
```

```
ds_kaggle <- read_csv("bitstampUSD_1min_data_20120101_to_20200914.csv")
head(ds_kaggle) %>% knitr::kable()
```

Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
1325317920	4.39	4.39	4.39	4.39	0.4555809	2	4.39
1325317980	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1325318040	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1325318100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1325318160	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1325318220	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
ds_kaggle$Timestamp <- as.POSIXct(ds_kaggle$Timestamp, origin="1970-01-01")
head(ds_kaggle)
```

```
## # A tibble: 6 x 8
##   Timestamp             Open   High    Low  Close `Volume_(BTC)`
##   <dttm>               <dbl>  <dbl>  <dbl>  <dbl>          <dbl>
## 1 2011-12-31 04:52:00   4.39   4.39   4.39   4.39          0.456
## 2 2011-12-31 04:53:00 NaN    NaN    NaN    NaN           NaN
## 3 2011-12-31 04:54:00 NaN    NaN    NaN    NaN           NaN
## 4 2011-12-31 04:55:00 NaN    NaN    NaN    NaN           NaN
## 5 2011-12-31 04:56:00 NaN    NaN    NaN    NaN           NaN
## 6 2011-12-31 04:57:00 NaN    NaN    NaN    NaN           NaN
## # ... with 2 more variables: `Volume_(Currency)` <dbl>, Weighted_Price <dbl>
```

```
min(ds_kaggle$Timestamp)
```

```
## [1] "2011-12-31 04:52:00 -03"
```

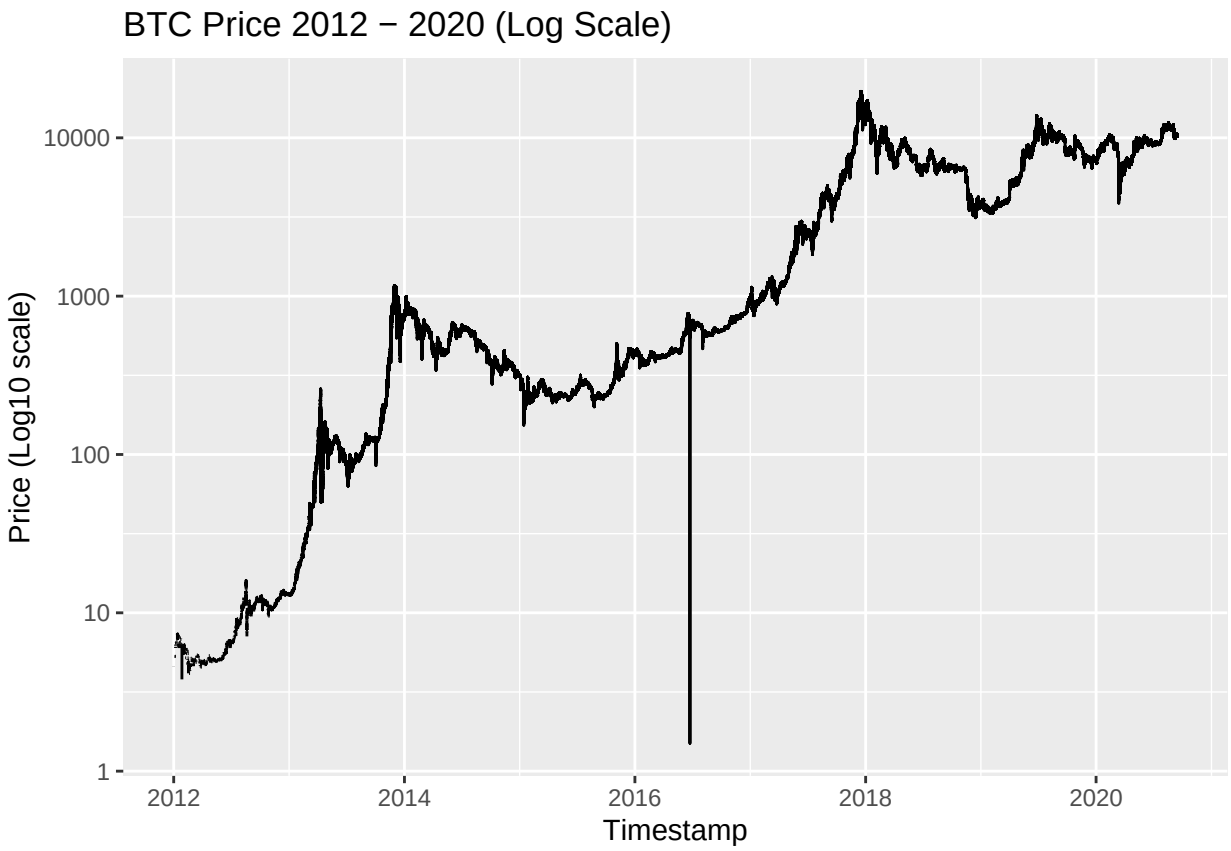
```
max(ds_kaggle$Timestamp)
```

```
## [1] "2020-09-13 21:00:00 -03"
```

2.5.1 Log Scale

The Stock and Flow model is always discused by diferents traders.

```
ds_kaggle %>% ggplot(aes(Timestamp, Close)) + geom_line() + scale_y_log10() +
  ylab("Price (Log10 scale)") + ggtitle("BTC Price 2012 - 2020 (Log Scale)")
```

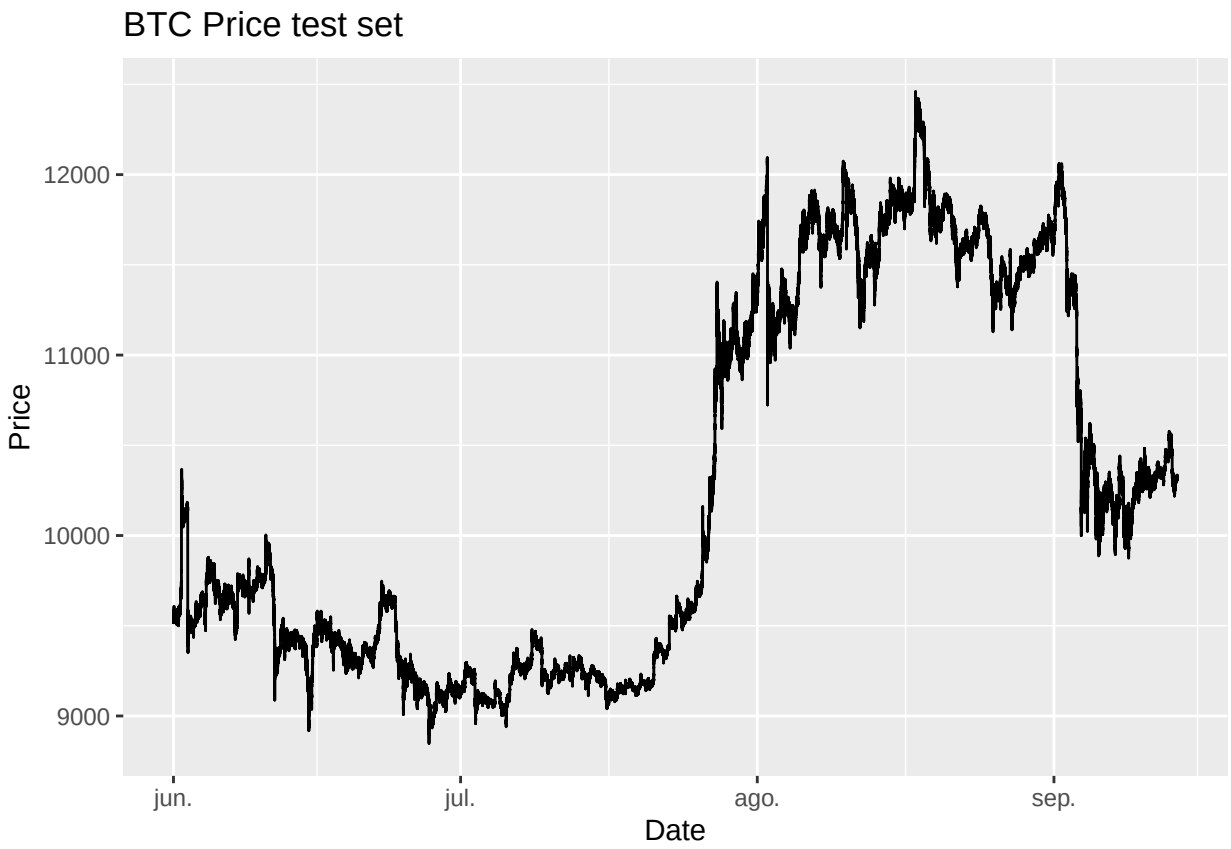


2.6 Machine Learning Prediction

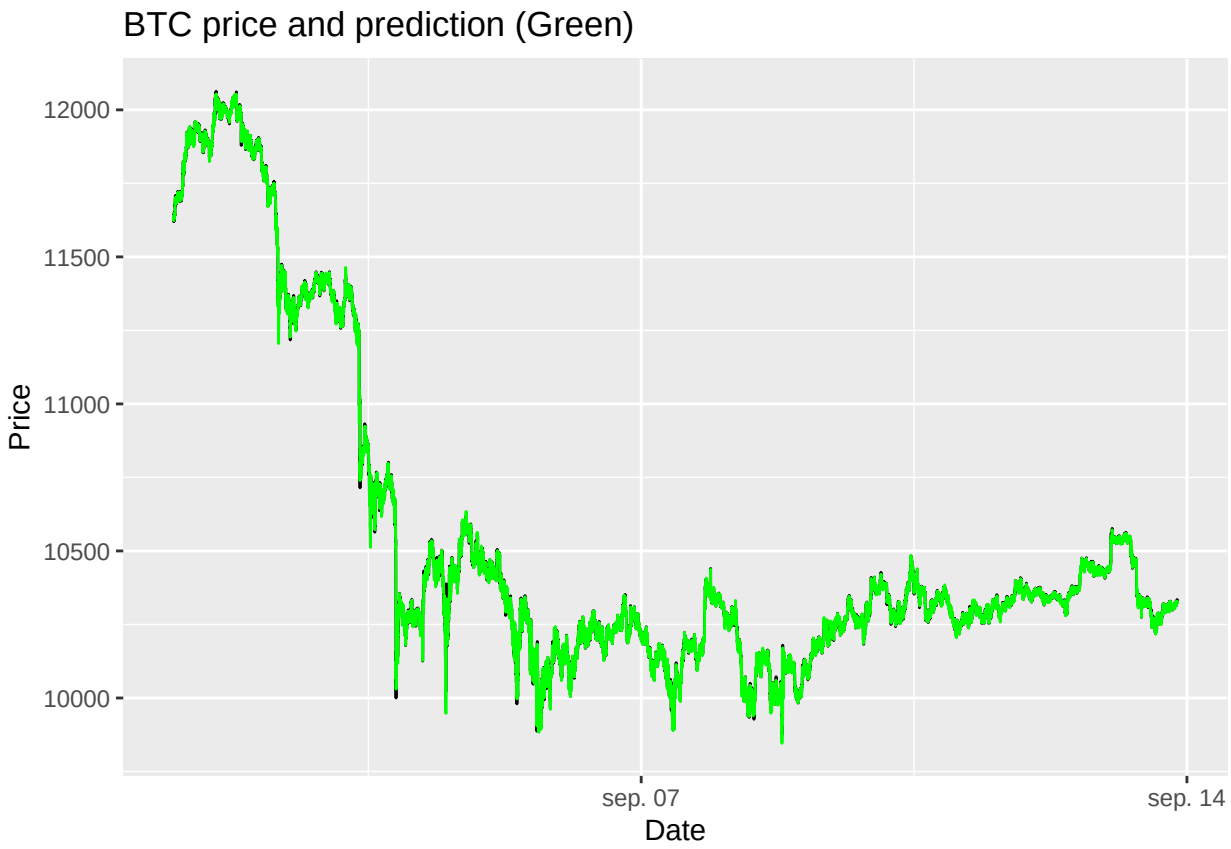
```
reduced_kaggle <- ds_kaggle %>% filter(Timestamp >= "2018-01-01")
```

We use the glm method

```
test_set <- reduced_kaggle %>% filter(Timestamp >= "2020-06-01", !is.na(Close))
train_set <- reduced_kaggle %>% filter(Timestamp < "2020-06-01", !is.na(Close))

test_set %>% ggplot(aes(Timestamp, Close)) + geom_line() +
  ylab("Price") + xlab("Date") + ggtitle("BTC Price test set")
```



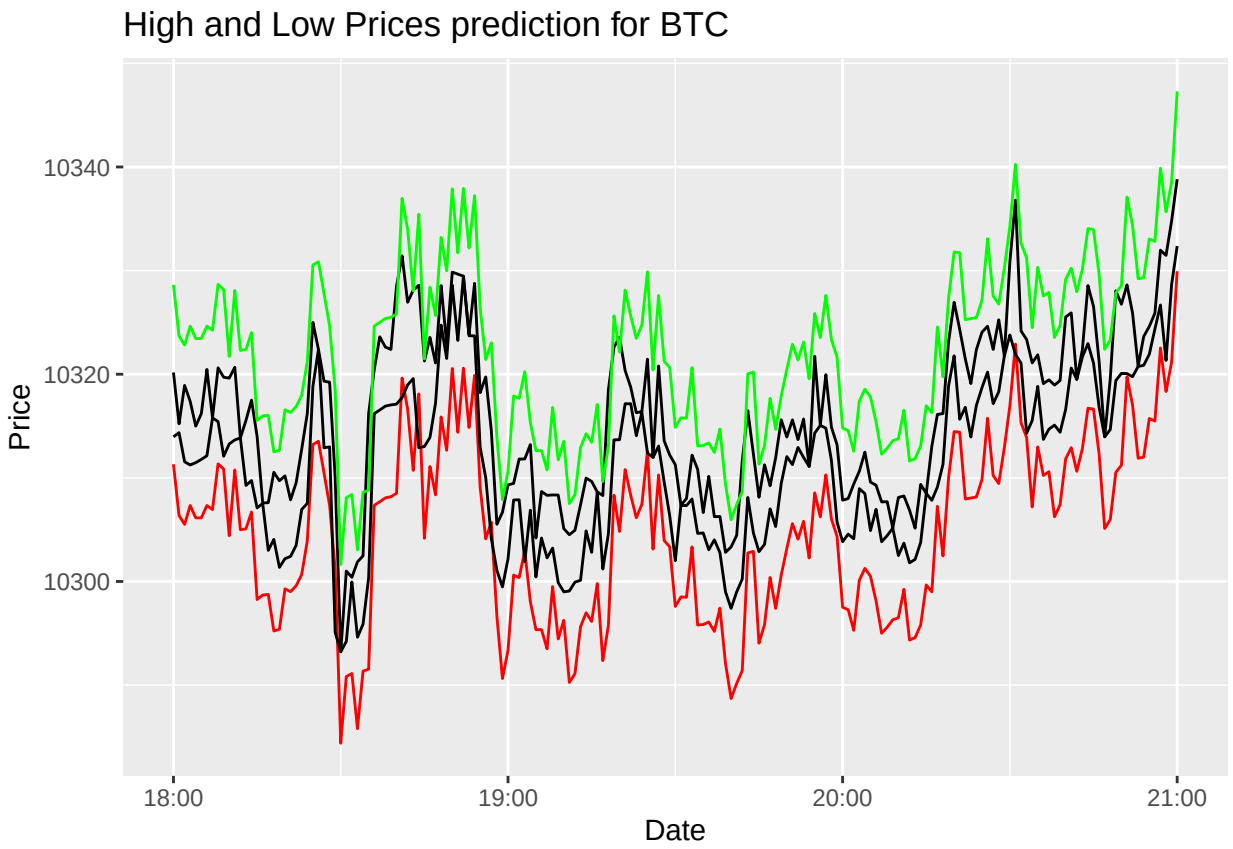

```
train_glm <- train(Close ~ ., method = "glm", data = train_set)

y_hat_glm <- predict(train_glm, test_set, type = "raw")

test_set %>% mutate(y_hat_glm = y_hat_glm) %>% filter(Timestamp >= "2020-09-01") %>% ggp
```



The correlation between the Price of close with Open Price High and Low is near to perfect, but the problem is we will have not all this information. It is making a prediction by row and by minute.

We try to predict the high and low values with the Open Price.

```
train_glm_high <- train(High ~ Open, method = "glm", data = train_set)
train_glm_low <- train(Low ~ Open, method = "glm", data = train_set)
y_hat_glm_high <- predict(train_glm_high, test_set, type = "raw")
y_hat_glm_low <- predict(train_glm_low, test_set, type = "raw")
test_set %>% mutate(y_hat_glm_high = y_hat_glm_high, y_hat_glm_low = y_hat_glm_low) %>%
  filter(Timestamp >= "2020-09-13 18:00") %>% ggplot(aes(Timestamp, High)) +
  geom_line() +
  geom_line(aes(Timestamp, y_hat_glm_high),col = "green") +
  geom_line(aes(Timestamp, y_hat_glm_low),col = "red") +
  geom_line(aes(Timestamp, Low), col ="black") + ggtitle("High and Low Prices prediction
```




```
test_set %>% filter(Timestamp >= "2020-09-13 18:00") %>% head() %>% knitr::kable()
```

Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
2020-09-13 18:00:00	10320.17	10320.17	10313.96	10315.22	1.8287349	18863.255	10314.92
2020-09-13 18:01:00	10315.22	10315.22	10314.36	10314.36	0.3100561	3198.252	10315.07
2020-09-13 18:02:00	10314.36	10318.92	10311.55	10316.18	3.9124152	40352.137	10313.87
2020-09-13 18:03:00	10316.19	10317.38	10311.24	10311.35	0.2270818	2342.420	10315.31
2020-09-13 18:04:00	10314.99	10314.99	10311.47	10311.54	0.4935460	5089.281	10311.67
2020-09-13 18:05:00	10314.99	10316.20	10311.78	10316.20	0.9834636	10144.626	10315.20

They are not good predictors.

```
test_set %>% head() %>% knitr::kable()
```

Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
2020-06-01 00:00:00	9530.01	9541.39	9530.01	9536.62	5.2551676	50135.3015	9540.191
2020-06-01 00:01:00	9541.21	9547.85	9537.39	9543.54	2.9884966	28528.2967	9546.036
2020-06-01 00:02:00	9543.54	9545.65	9536.06	9536.79	5.2079182	49677.7248	9538.884
2020-06-01 00:03:00	9539.88	9545.61	9534.86	9545.61	2.0350560	19425.5986	9545.486
2020-06-01 00:04:00	9537.35	9537.35	9536.62	9536.62	0.0151896	144.8614	9536.869
2020-06-01 00:05:00	9543.01	9544.30	9540.97	9544.30	0.0407025	388.4031	9542.483

2.7 Machine learning dataset future

We need to add the price for tomorrow in the same row with the predictors, so, we can train the algorithm for the price of tomorrow.

```
Price <- historical$Price
histor <- historical %>% mutate(tomorrow = NA)
histor$tomorrow[2:1172] <- Price[1:1171]
```

Future price, tomorrow

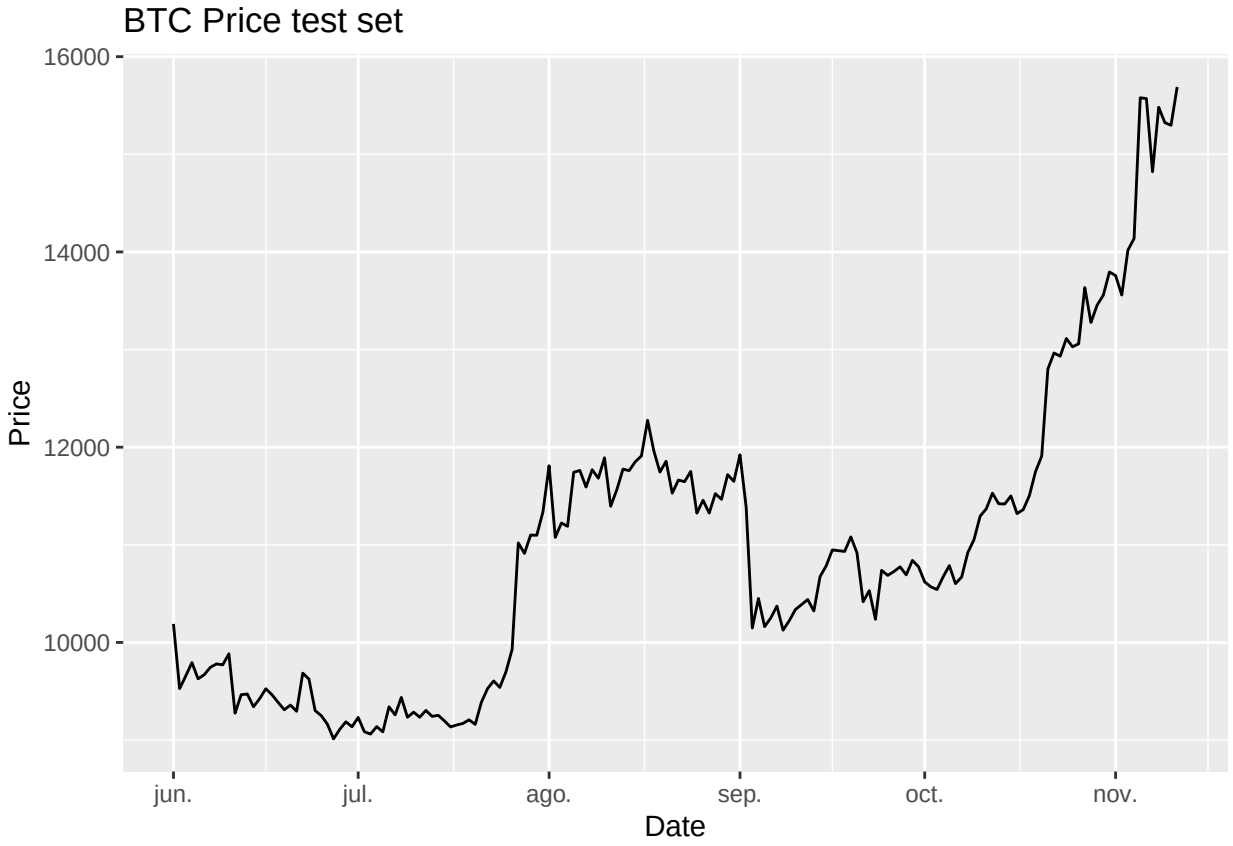
```
histor %>% head() %>% knitr::kable()
```

Date	Price	Open	High	Low	Vol.	Change %	ma	tomorrow
2020-11-12	16125.8	15686.7	16125.8	15581.0	102.19K	2.78%	13794.5	NA
2020-11-11	15690.0	15311.8	15867.4	15311.8	78.45K	2.57%	13758.1	16125.8
2020-11-10	15297.2	15329.3	15346.7	15235.3	61.68K	-0.18%	13635.0	15690.0
2020-11-09	15324.5	15483.8	15608.5	15208.8	108.98K	-1.02%	13558.7	15297.2
2020-11-08	15481.8	14801.9	15518.4	14801.9	65.57K	4.46%	13556.1	15324.5
2020-11-07	14821.1	15566.0	15590.0	14820.7	101.41K	-4.81%	13455.2	15481.8

2.7.1 Glm method

We predict

```
test_set_histor <- histor %>% filter(Date >= "2020-06-01", !is.na(tomorrow)) %>%
  select(Date, Price, Open, High, Low, tomorrow)
train_set_histor <- histor %>% filter(Date < "2020-06-01", !is.na(tomorrow)) %>%
  select(Date, Price, Open, High, Low, tomorrow)

test_set_histor %>% ggplot(aes(Date, Price)) + geom_line() + ggtitle("BTC Price test set
```




```
head(train_set_histor)
```

```
## # A tibble: 6 x 6
##   Date       Price  Open  High   Low tomorrow
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 2020-05-31 9461. 9700. 9700. 9400.   10190
## 2 2020-05-30 9695  9432. 9734. 9353.    9461.
## 3 2020-05-29 9425. 9591. 9601. 9355.    9695
## 4 2020-05-28 9565. 9205  9611. 9119     9425.
```

```
## 5 2020-05-27 9197  8837  9217. 8819.    9565.
## 6 2020-05-26 8843  8902. 9002  8700.    9197
```

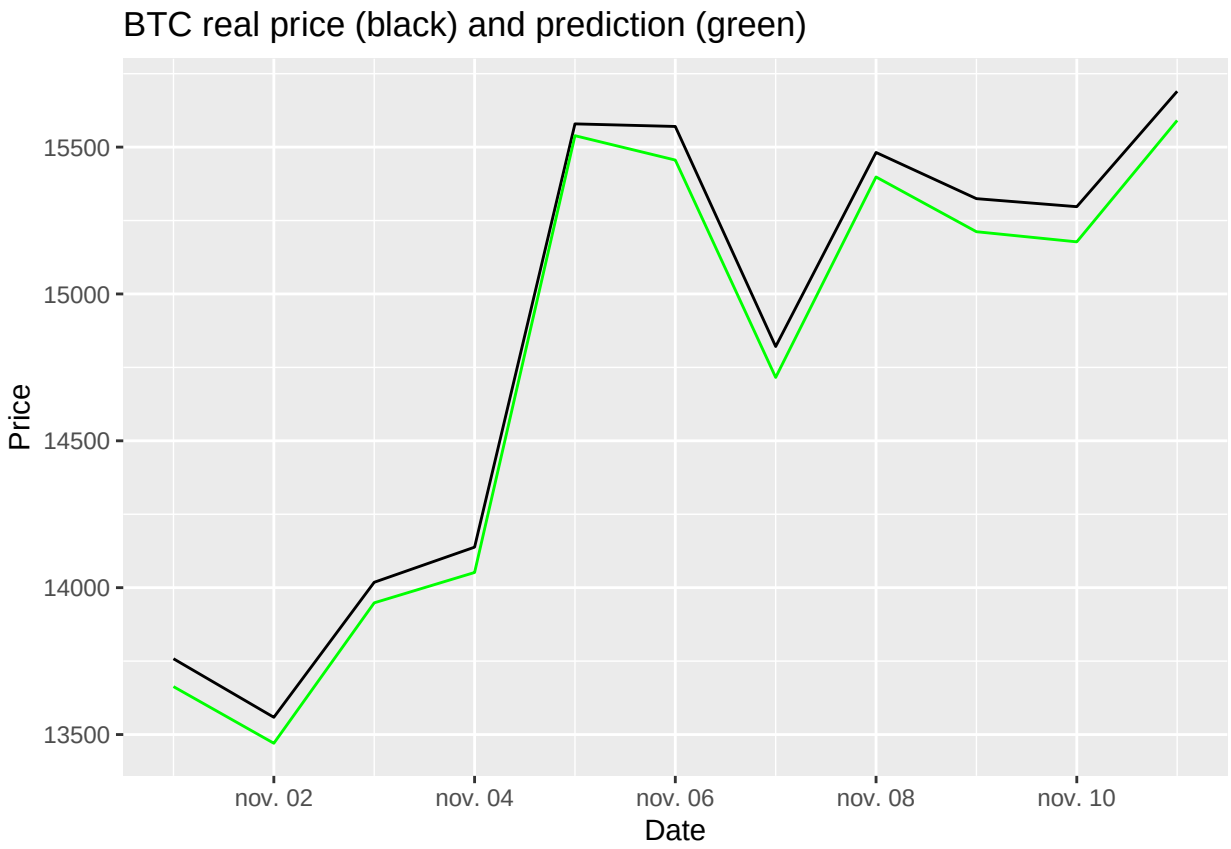
```
head(test_set_histor)
```

```
## # A tibble: 6 x 6
##   Date        Price   Open   High    Low tomorrow
##   <date>      <dbl>  <dbl>  <dbl>  <dbl>    <dbl>
## 1 2020-11-11 15690  15312. 15867. 15312.   16126.
## 2 2020-11-10 15297. 15329. 15347. 15235.   15690
## 3 2020-11-09 15324. 15484. 15608. 15209.   15297.
## 4 2020-11-08 15482. 14802. 15518. 14802.   15324.
## 5 2020-11-07 14821. 15566  15590  14821.   15482.
## 6 2020-11-06 15570. 15592. 15825. 15472    14821.
```

```
train_glm <- train(tomorrow ~ ., method = "glm", data = train_set_histor)
train_glm
```

```
## Generalized Linear Model
##
## 1007 samples
##    5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1007, 1007, 1007, 1007, 1007, 1007, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   419.0554  0.9766249  252.3858
```

```
y_hat_glm <- predict(train_glm, test_set_histor, type = "raw")

test_set_histor %>% mutate(y_hat_glm = y_hat_glm) %>%
  filter(Date >= "2020-11-01") %>% ggplot(aes(Date, Price)) +
  geom_line() + geom_line(aes(Date, y_hat_glm),col = "green") + ggtitle("BTC real price
```




```
rmse_glm <- RMSE(y_hat_glm, test_set_histor$Price)
```

We normalize the RMSE.

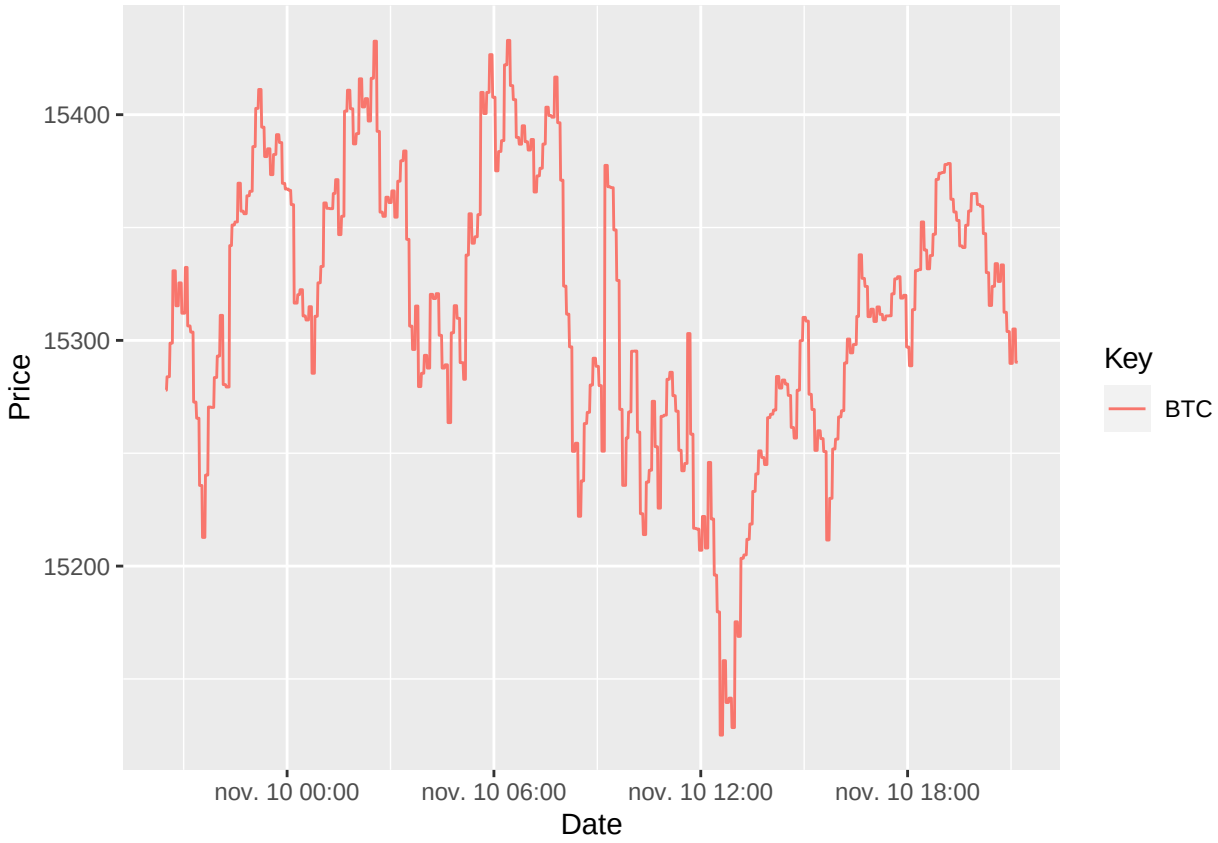
That normalisation doesn't really produce a percentage (e.g. 1 doesn't mean anything in particular), and it isn't any more or less valid than any other form of normalisation. It depends on the distribution of that data.

```
rmse_glm/(max(test_set_histor$Price)-min(test_set_histor$Price))
```

```
## [1] 0.007335916
```

I go back to price minute by minute to set the bands

```
dset %>% group_by(Key) %>% filter(Key == top20$Key[1]) %>% ggplot(aes(Date, Price, col=
```



We take the trend of the last day and knowing where the price is going, we can choose to establish points for buy and sell in a couple of hours.

We try again with other tecnics kNN

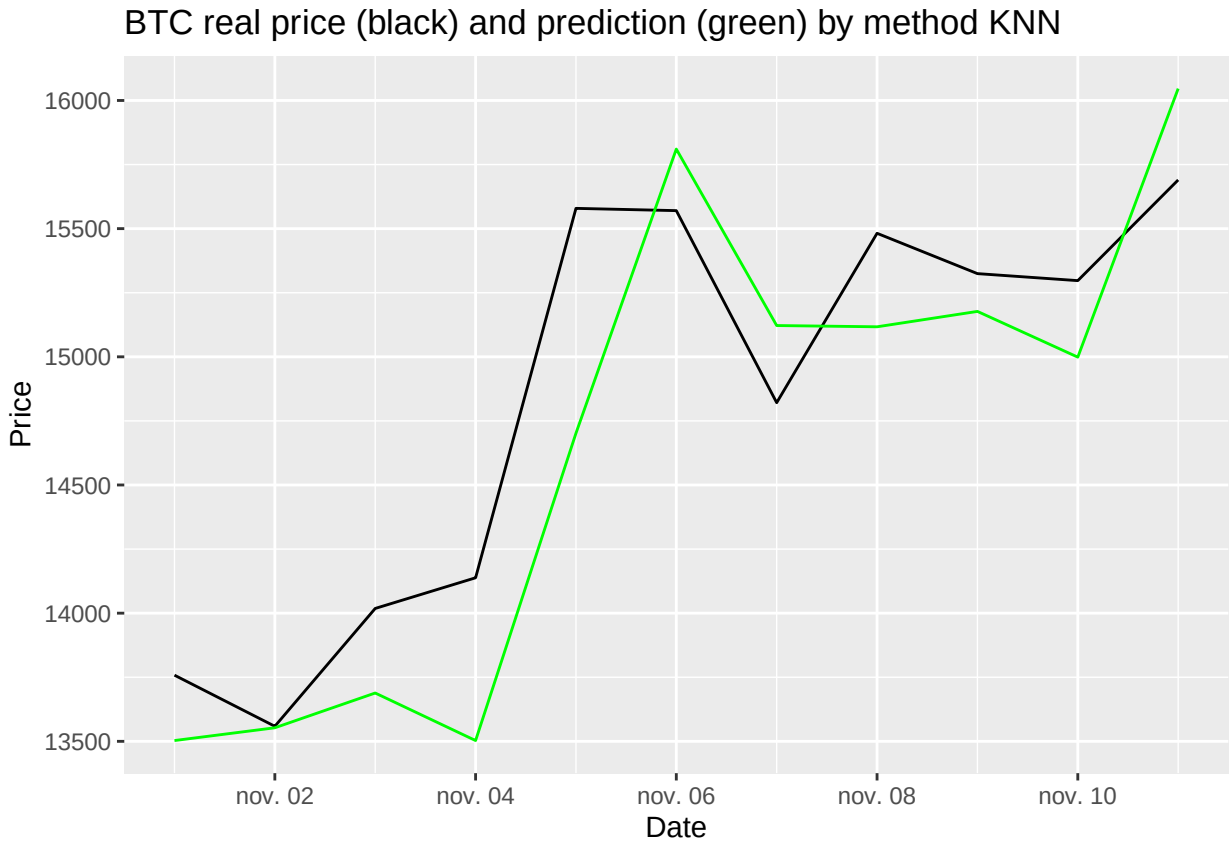
2.7.2 Knn method

```
train_knn <- train(tomorrow ~ ., method = "knn", data = train_set_histor)
train_knn
```

```
## k-Nearest Neighbors
##
## 1007 samples
##    5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1007, 1007, 1007, 1007, 1007, 1007, ...
## Resampling results across tuning parameters:
##
```

```
##   k  RMSE      Rsquared   MAE
##   5  516.6746  0.9648195  306.5821
##   7  503.3505  0.9666734  296.9752
##   9  495.0276  0.9678531  290.3415
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 9.
```

```
y_hat_knn <- predict(train_knn, test_set_histor, type = "raw")

test_set_histor %>% mutate(y_hat_knn = y_hat_knn) %>% filter(Date >= "2020-11-01") %>% g
```




```
rmse_knn <- RMSE(y_hat_knn, test_set_histor$Price)
rmse_knn %>% knitr::kable()
```

x
191.3683

We normalize the RMSE.

```
rmse_knn/(max(test_set_histor$Price)-min(test_set_histor$Price))
```

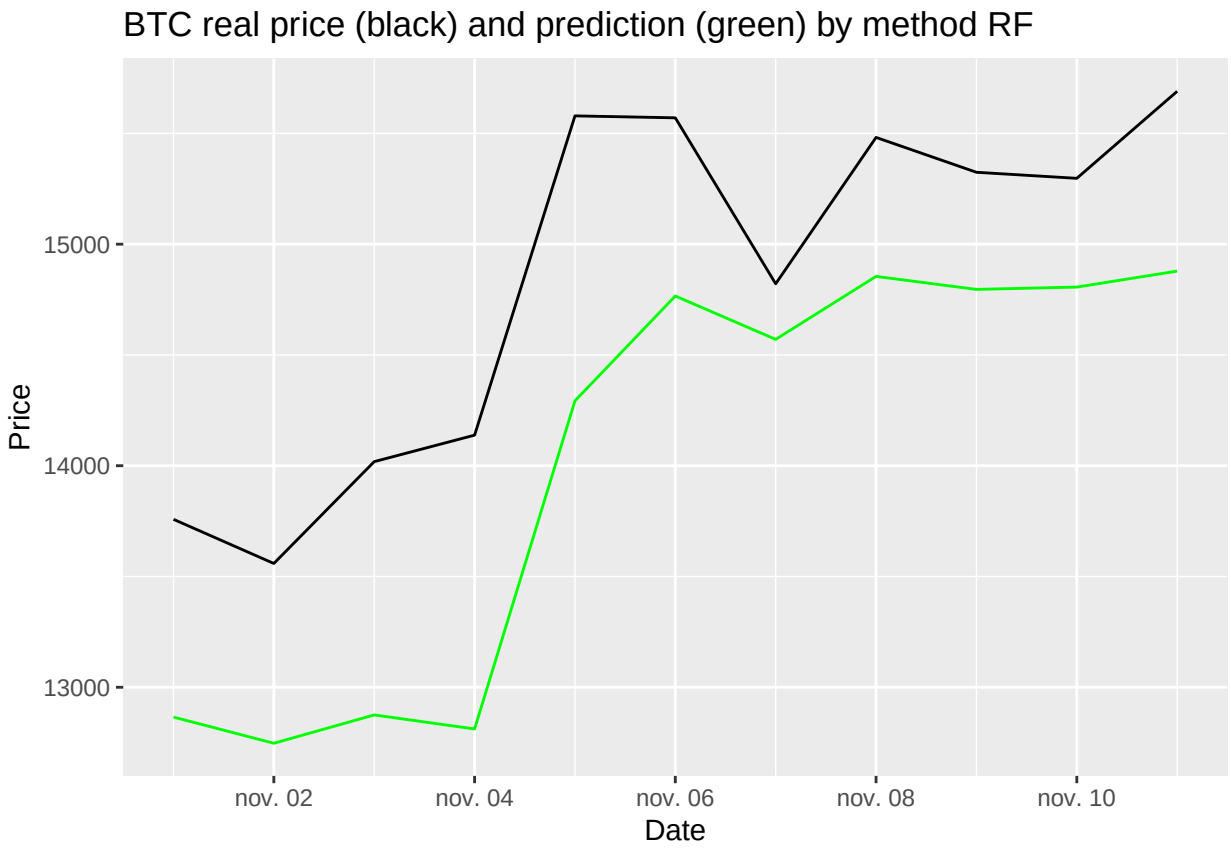
```
## [1] 0.02865653
```

2.7.3 Random forest method

```
train_rf <- train(tomorrow ~ ., method = "rf", data = train_set_histor)
train_rf
```

```
## Random Forest
##
## 1007 samples
##    5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1007, 1007, 1007, 1007, 1007, 1007, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     439.0995  0.9747408  270.8424
##   3     440.1632  0.9745623  273.8745
##   5     446.9333  0.9738001  279.7706
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
y_hat_rf <- predict(train_rf, test_set_histor, type = "raw")

test_set_histor %>% mutate(y_hat_rf = y_hat_rf) %>% filter(Date >= "2020-11-01") %>% ggp
```




```
rmse_rf <- RMSE(y_hat_rf, test_set_histor$Price)
rmse_rf %>% knitr::kable()
```

x
398.8221

We normalize the RMSE.

```
rmse_rf/(max(test_set_histor$Price)-min(test_set_histor$Price))
```

```
## [1] 0.05972178
```

Chapter 3

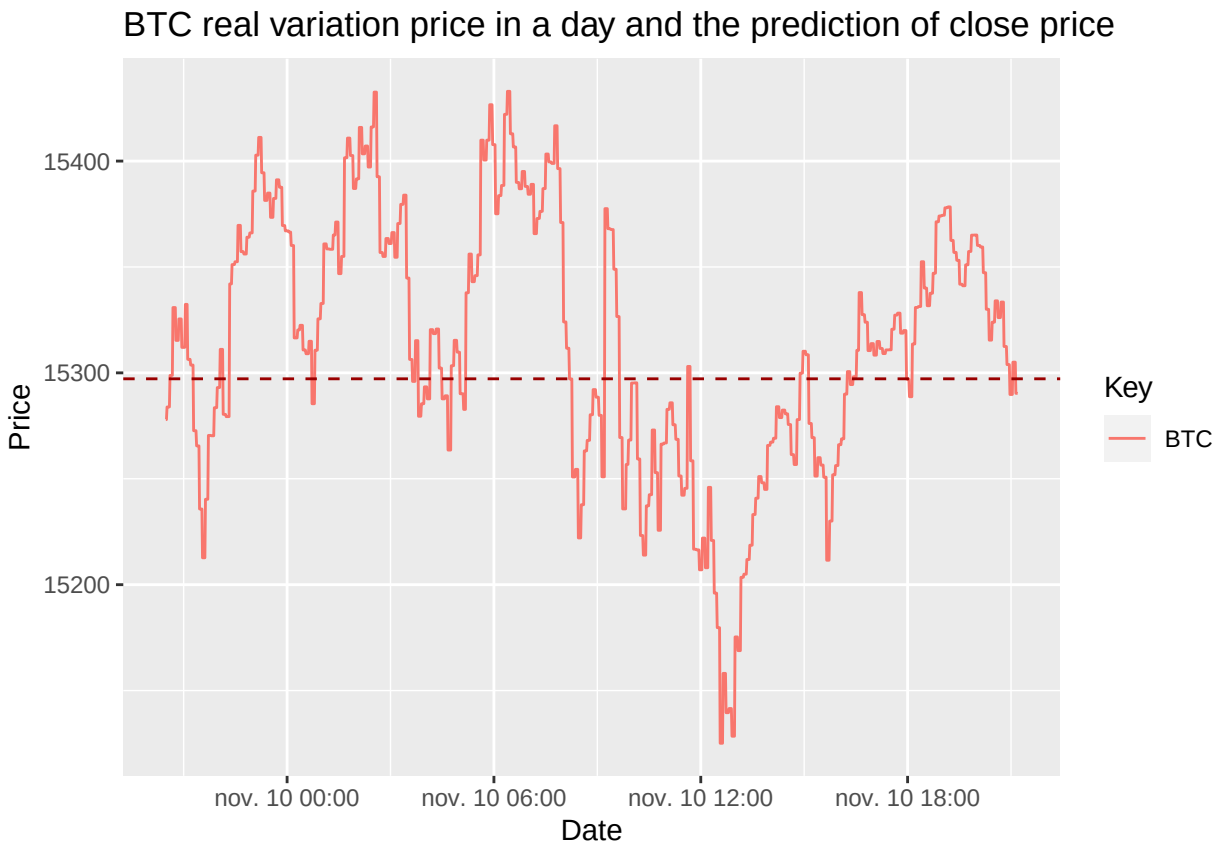
Results

So, the method "glm" is the best to predict the close price for tomorow using like predictor the data from today. It is very good, knowing the close price of tomorrow, its permits us buy and sell in a period of time in the day, like we see in the daily dataset.

```
day09 <- test_set_histor %>% filter(Date == "2020-11-09")
day09 %>% knitr::kable()
```

Date	Price	Open	High	Low	tomorrow
2020-11-09	15324.5	15483.8	15608.5	15208.8	15297.2

```
dset %>% group_by(Key) %>% filter(Key == top20$Key[1]) %>%
  ggplot(aes(Date, Price, col= Key)) + geom_line() +
  geom_hline(aes(yintercept = day09$tomorrow), colour="#990000", linetype="dashed") + gg
```



In the Graphic we can see the line with the prediction at the finish of the day, this prediction, we construct, exist before the minutes advance in the day, so, we can see when the price is high and low.

Is posible to ensemble all in a real time function. Is not posible realice in this code because we need to generate The final and rmd document.

Chapter 4

Conclusion

We was looking the comportament of prices for three datasets, there are a lot of information for know more the dinamic of prices. Also needed made a photography in a specific moment for this analysis. I continued looking the prices days after of this capture, and not all the time the prediction is good, specially when the market is going up, there are a lot of players and is not easy predict a price, but it can be posible. Again, is posible ensemble all thogether in a function and also fit the model. Like a observation, at the moment i am finishing this work, the CSS selectors are changed in the webpage of Coinmarketcap.com, it are think is necesary be carrefull and control for errors in the web scraping, principally if we will make trade. This work shows the powerfull of tecnics of Machine Learning for predict prices. I will continue developing and fitting. You can follow me in my GitHub account with the name Avantas, and also in Twitter. Finally, I would like to thank the entire team behind the platform that make learning possible.