Clustering Cryptocurrencies - K-Means

Kühni, Romaine; Exchange Student
Drianne, Pierre-Matthias; Exchange Student
Bader, Michael; MIA, 3rd Semester

at Universität St. Gallen

Ladislaus von Bortkiewicz Chair of Statistics

C.A.S.E.-Center for Applied Statistics and Economics

International Research Training Group

Humboldt-Universität zu Berlin
lvb.wiwi.hu-berlin.de

www.case.hu-berlin.de

irtg1792.hu-berlin.de

link to Github

Outline

I - Introduction & Motivation
- Introduction
- Coingecko

II - Exploratory Analysis
- Plotting Prices & Returns
- Stationarity Test
- Autocorrelation Test
- Statistical Summary
- Explorative Clustering

III - Clustering
- Principal Component Analysis
- Linear Factor Analysis

IV - Conclusion
- Discussion of the Results
- Limitations & Further Studies

1 - Introduction

Why Clustering cryptocurrencies?

⊡ The behaviour of cryptocurrencies is not yet very well understood.

⊡ The decentralised and data-rich nature of cryptocurrency transactions creates a lot of accessible data that can be studied easily.

⊡ Advances in technology and know-how have made machine-learning (ML) more and more accessible.

⊡ Combining machine-learning with cryptocurrencies allows us to acquire skills in two very important fields of the future.

⊡ It also allows for findings that are achievable for a small group of motivated students with limited prior knowledge of Python and ML.

1- Web scrapping daily returns from coingecko

```
# Calling Coin Gecko API
## Import the data
#cg = CoinGeckoAPI()
#coins_market = cg.get_coins_markets('usd')
#df_coins_market = pd.DataFrame(coins_market)
```

```
## Get the prices for each crypto for 365 days (01.01.19 - 31.12.19)
#prices = []
#for i in df_coins_market['id']:
#    a = cg.get_coin_market_chart_range_by_id(i, 'usd', 1546300800, 1577750400)
#    b = a['prices']
#    c = []
#   for j in range(len(b)):
#        c.append(b[j][1])
#    prices.append([i, c])
#del a, b, c, i, j

## Creating the prices, returns and market caps dataframes

#frame = pd.DataFrame(prices)
#coins = frame[1].apply(pd.Series)
#frame = coins.set_index(frame[0])
#frame = frame.dropna(axis=0).transpose()
#frame.to_csv('CC_Prices.csv',index=True,header=True)
```

Approach

We used the public and free API of Coingecko.com. We initially included all coins of the service in our download of daily returns for the year 2019. We wanted to study a long, recent time-period that is free from the impact of the pandemic, hence 2019

The process of cleaning the data includes dropping all coins that have missing data and arranging the daily returns into a pandas data frame.

Limitations

- Our code ran into error 429. Our loop sometimes created too many requests in too short a time period

- The data we received from the website also was not always consistent

- There is no guarantee the API will be available in the future and our study be repeatable

II- Exploratory analysis

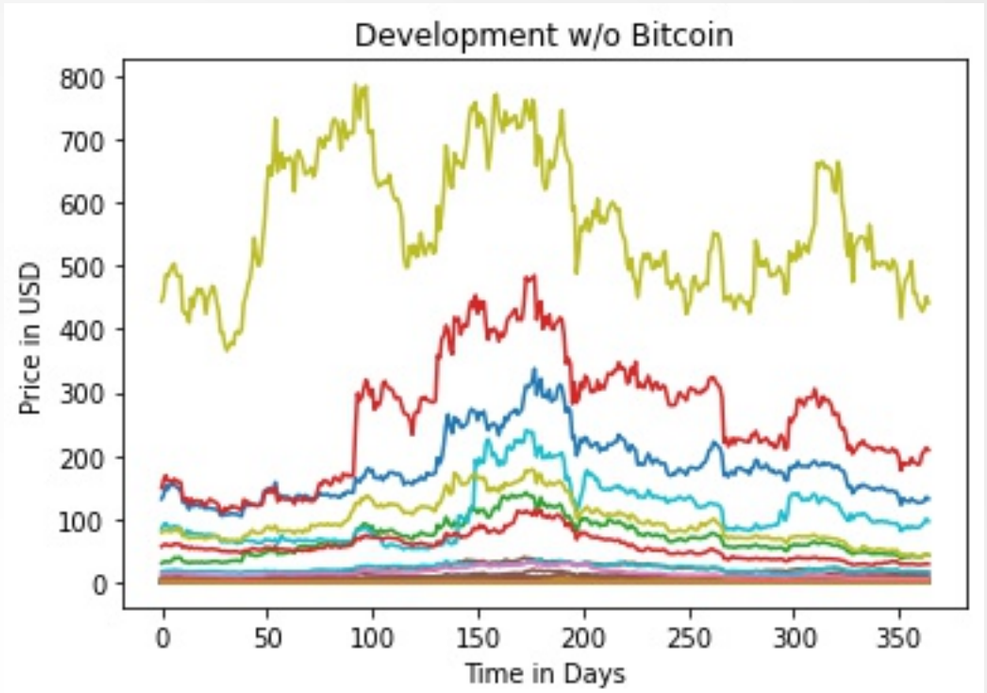
The following couple of slides give insights into the dataset used to work with and helped identifying potential patterns. The analysis follows the structure below:

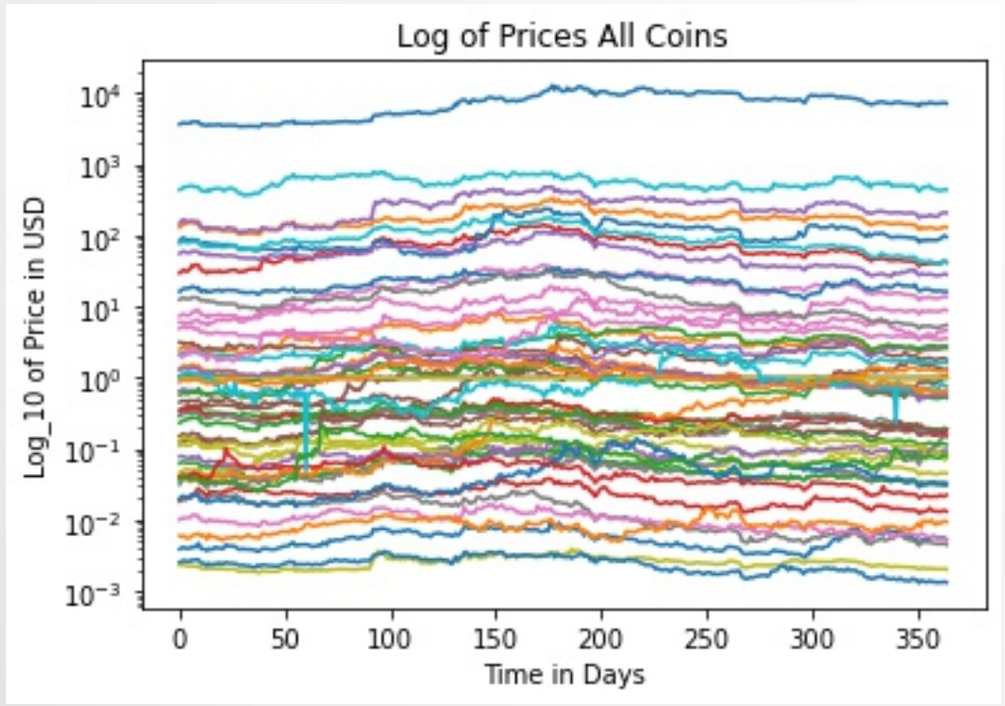
1) Plotting the prices over the period of various combinations of coins
2) CCs returns and market caps
3) Stationarity test
4) Autocorrelation test
5) Statistical summary of cc
6) Scatterplot of correlated cc
7) Explorative Clustering with K-Means

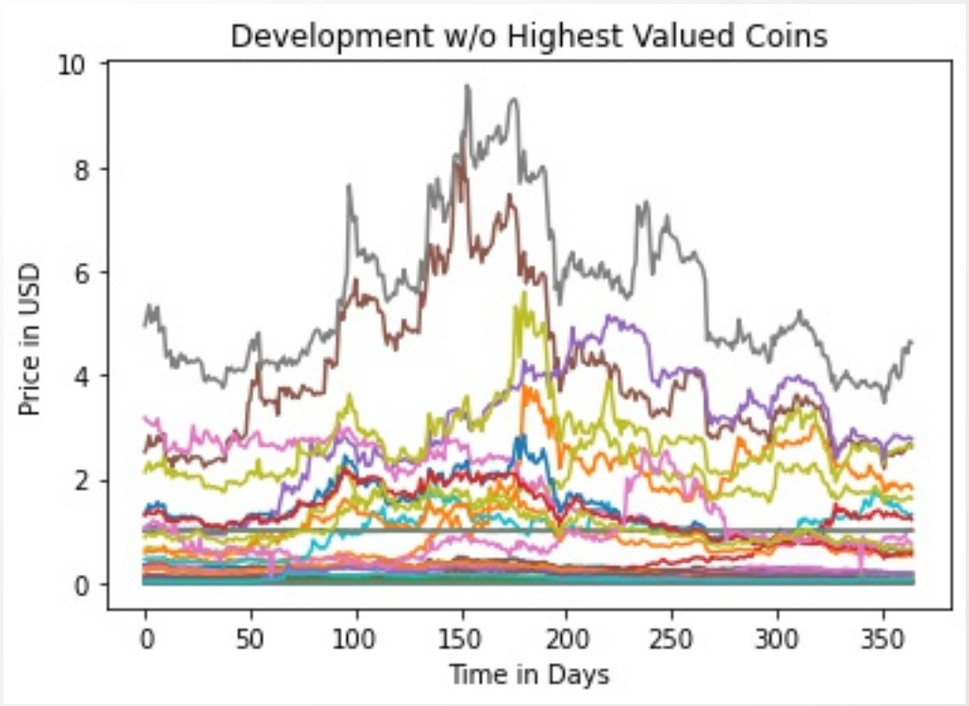
II - Exploratory analysis
1) Plotting the prices over the period of various combinations of coins

Observation

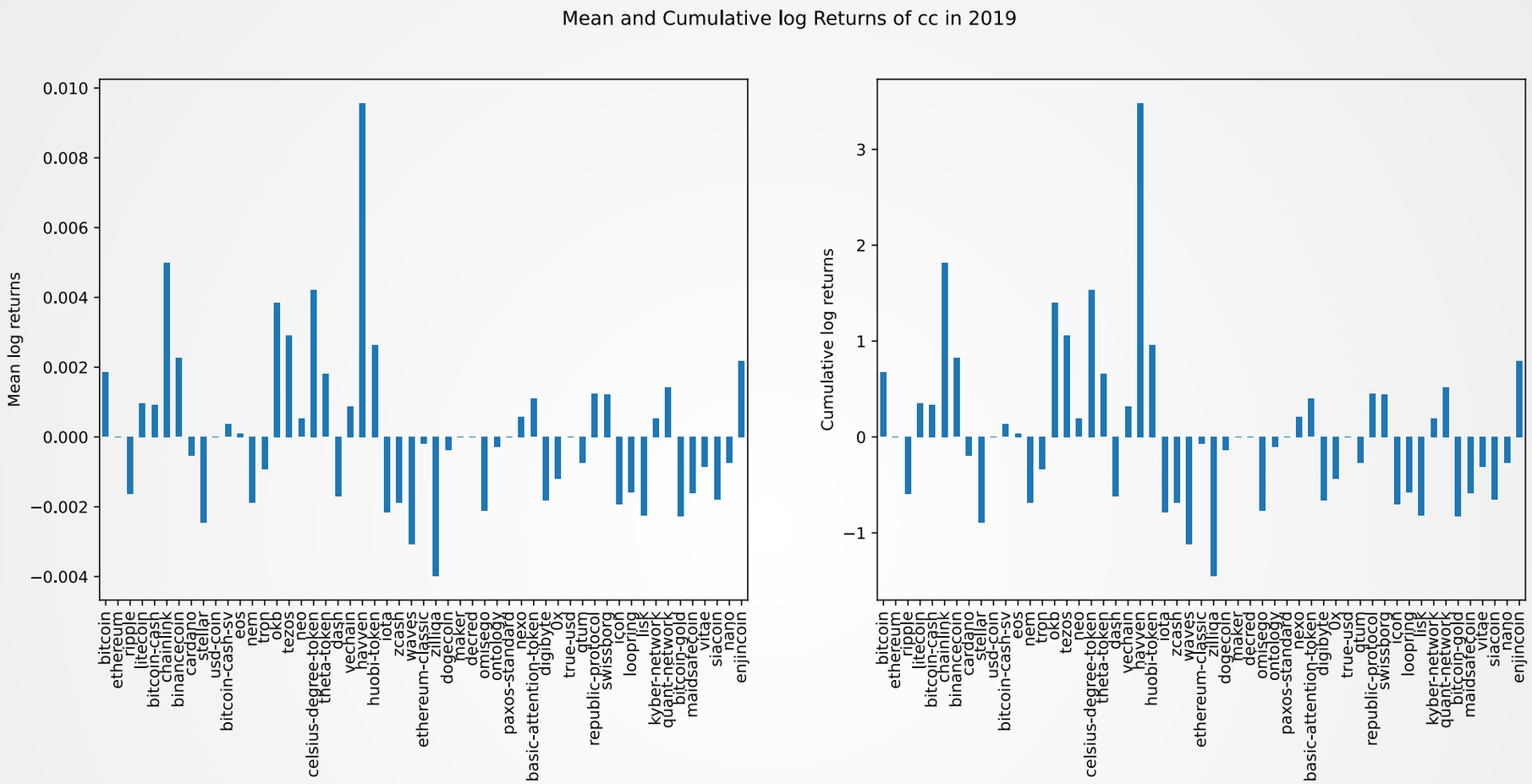
We see that the development of all coins in 2019 appears very noisy. Even when we ignore the highest performing coins no clear patterns are emerging. However, we start seeing that some time series start resembling each other. This gives encouragement to further study patterns within the data.







II - Exploratory analysis
2) CCs log-returns and prices/market caps

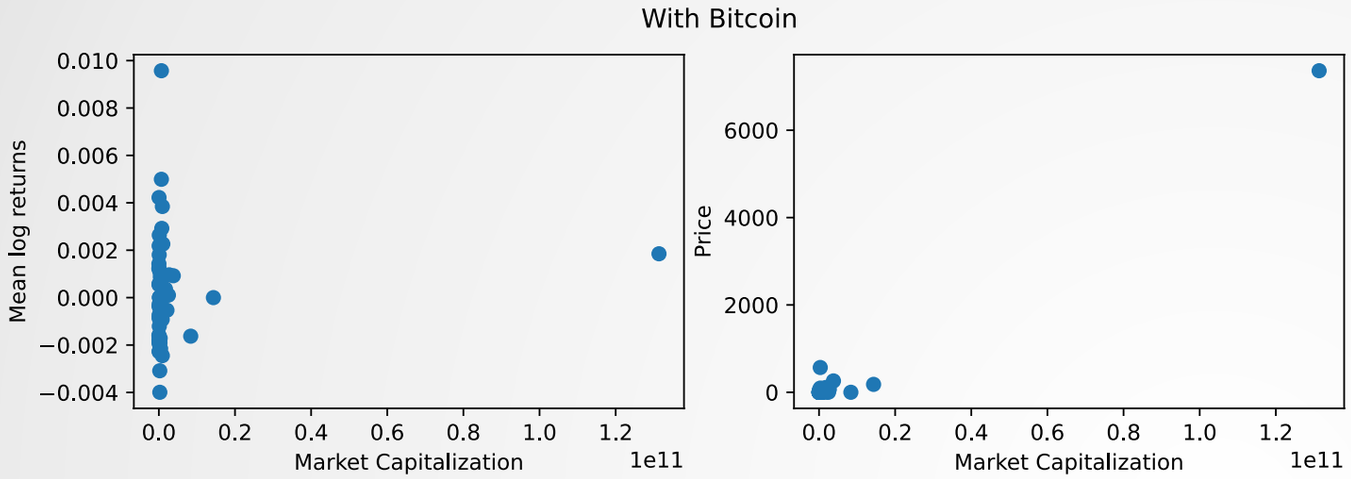


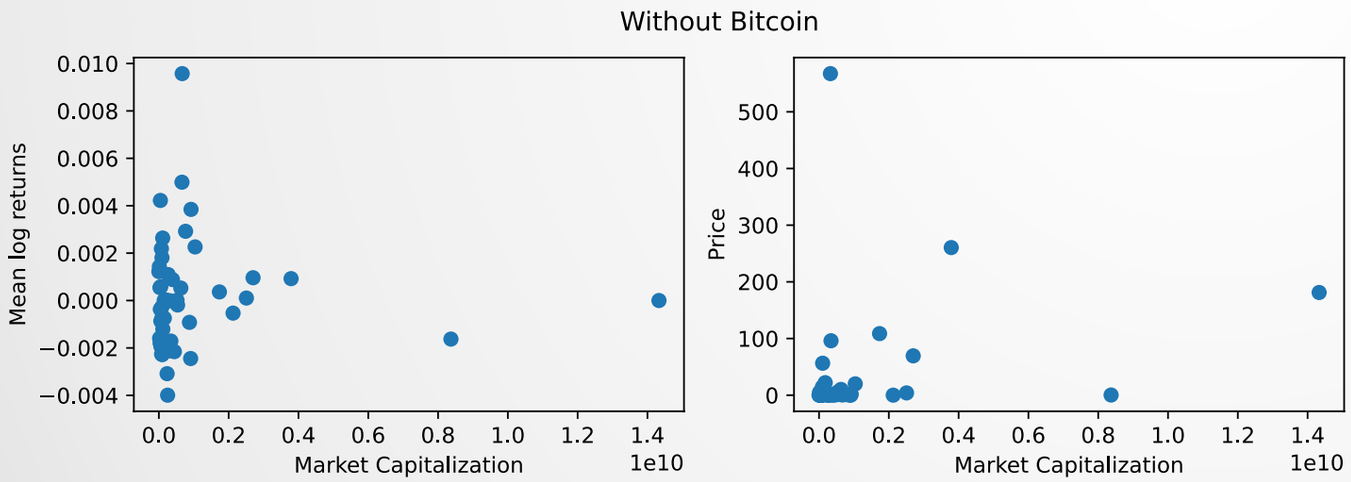
Observation

The cumulative log-returns for 2019 are all mostly in the range of -1 to 1,5; the average is 0.76.

Most cryptos are in a narrow range, while only a few show extreme values. This might make finding clusters more difficult down the road.

II - Exploratory analysis
2) CCs log-returns and prices/market caps





Observation

Market Capitalization - Mean Returns and Market Capitalization - Mean Price:

- Cryptocurrencies are gathered in a small area
- We see in both analysis that the high market capitalization of the bitcoin disturb a graphical analysis
- When ignoring bitcoin, the concentration of coins starts to give way to a more nuanced picture, however, a strong concentration is still visible in both plots

Conclusion:

The presence of the bitcoin and its extremely large Market capitalization and high price might affect the clustering.

If we decide to use the market capitalization and the price to create our clusters, one might compare the results with and without the bitcoin as it may have too much weight in the computation.

II - Exploratory analysis
3) Stationarity test

Observation

The graphic on the right is an extraction of the output table for the augmented Dickey-Fuller test with alpha = 0.05. The analysis has shown, that **all our coins pass this test**. This is important as the stationarity of time series is an important prerequisite for further statistical analysis. Therefore, we can keep working with the set of coins we have been using so far.

Cryptocurrency	Stationary
bitcoin	TRUE
ethereum	TRUE
ripple	TRUE
litecoin	TRUE
bitcoin-cash	TRUE
chainlink	TRUE
binancecoin	TRUE
cardano	TRUE
stellar	TRUE
usd-coin	TRUE
bitcoin-cash-sv	TRUE
eos	TRUE
nem	TRUE
tron	TRUE
okb	TRUE
tezos	TRUE
neo	TRUE

	Stationary
celsius-degree-token	TRUE
theta-token	TRUE
dash	TRUE
vechain	TRUE
havven	TRUE
huobi-token	TRUE
iota	TRUE
zcash	TRUE
waves	TRUE
ethereum-classic	TRUE
zilliqa	TRUE
dogecoin	TRUE
maker	TRUE
decred	TRUE
omisego	TRUE
ontology	TRUE
paxos-standard	TRUE

	Stationary
nexo	TRUE
basic-attention-token	TRUE
digibyte	TRUE
0x	TRUE
true-usd	TRUE
qtum	TRUE
republic-protocol	TRUE
swissborg	TRUE
icon	TRUE
loopring	TRUE
lisk	TRUE
kyber-network	TRUE
quant-network	TRUE
bitcoin-gold	TRUE
maidsafecoin	TRUE
vitae	TRUE
siacoin	TRUE
nano	TRUE
enjincoin	TRUE

II - Exploratory analysis
4) Autocorrelation test

Procedure

Autocorrelation of cryptocurrencies has been tested with the following features:

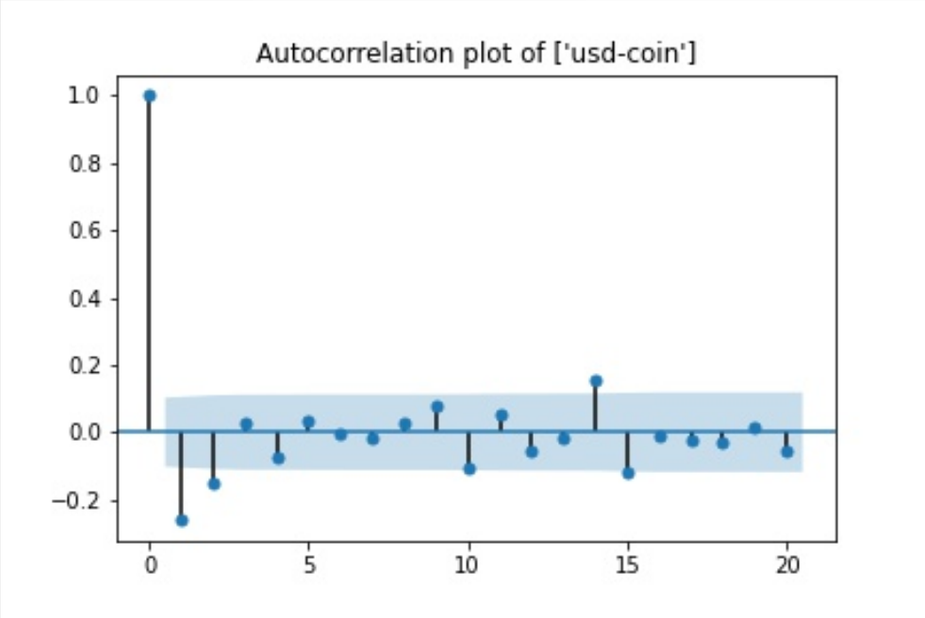
- **Number of Lags**: 20
- **Alpha**: 0.05
- **Threshold above which a crypto is considered as being autocorrelated to a given lag**: ±0.25

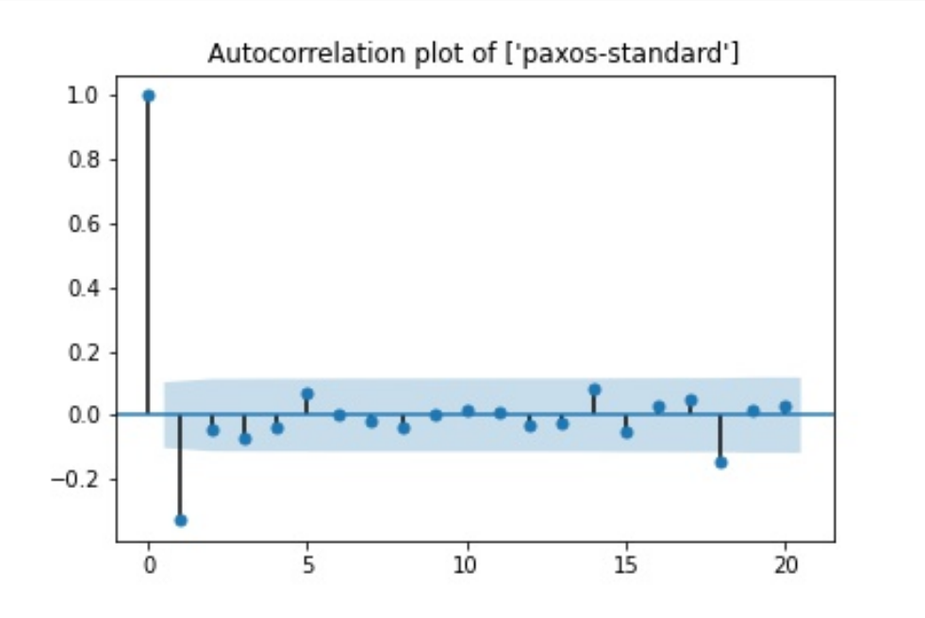
Observation

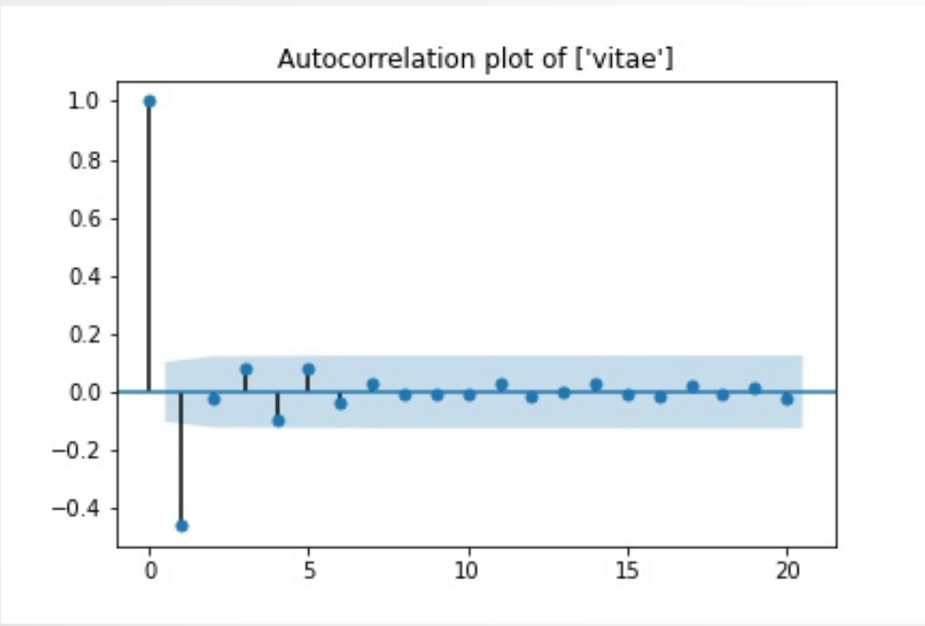
Four **coins have shown an autocorrelation.** Autocorrelation plots are displayed in the next slide. Concerned cc are “USD-coin”, “paxos-standard”, “true-USD” and “vitae”.

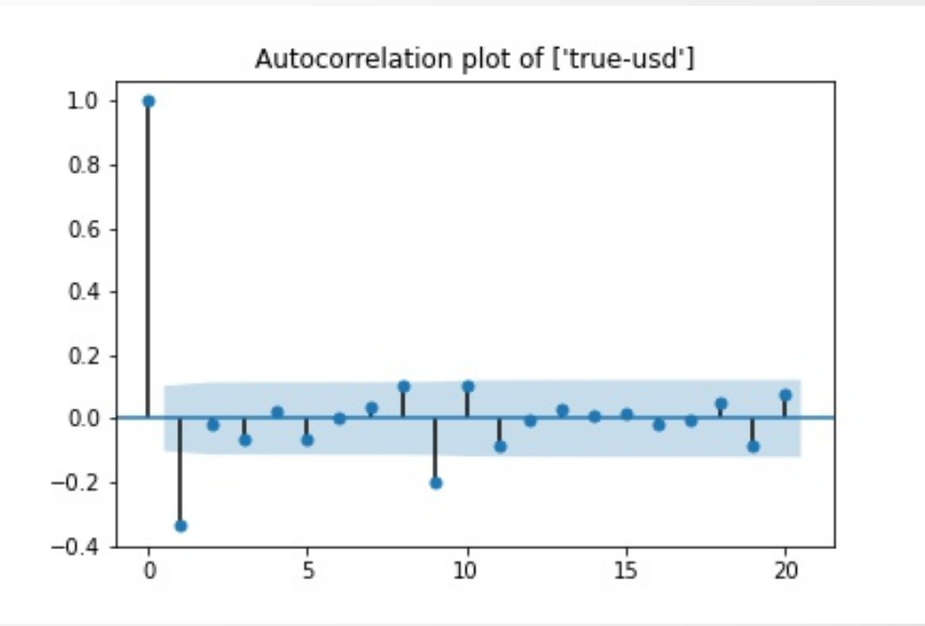
The pattern is always very similar. A relatively high autocorrelation is observed in the first lag and decrease then. Nevertheless, some dispersed lags also present some autocorrelation.

Autocorrelation plots





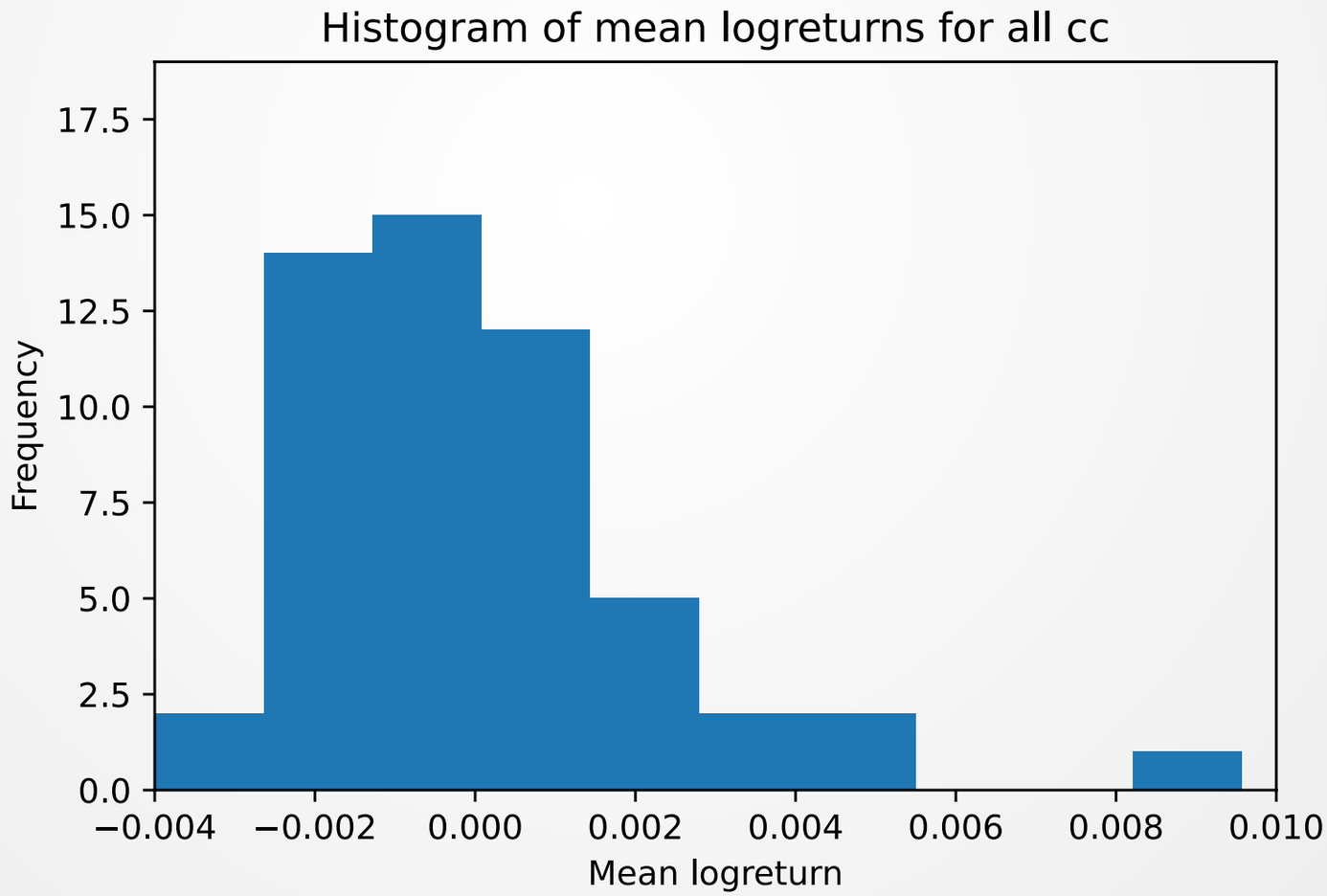




II - Exploratory analysis
5) Statistical summary of the CCs

Observation

The mean log-returns of all the CCs are highly concentrated in a small area (slightly negative).

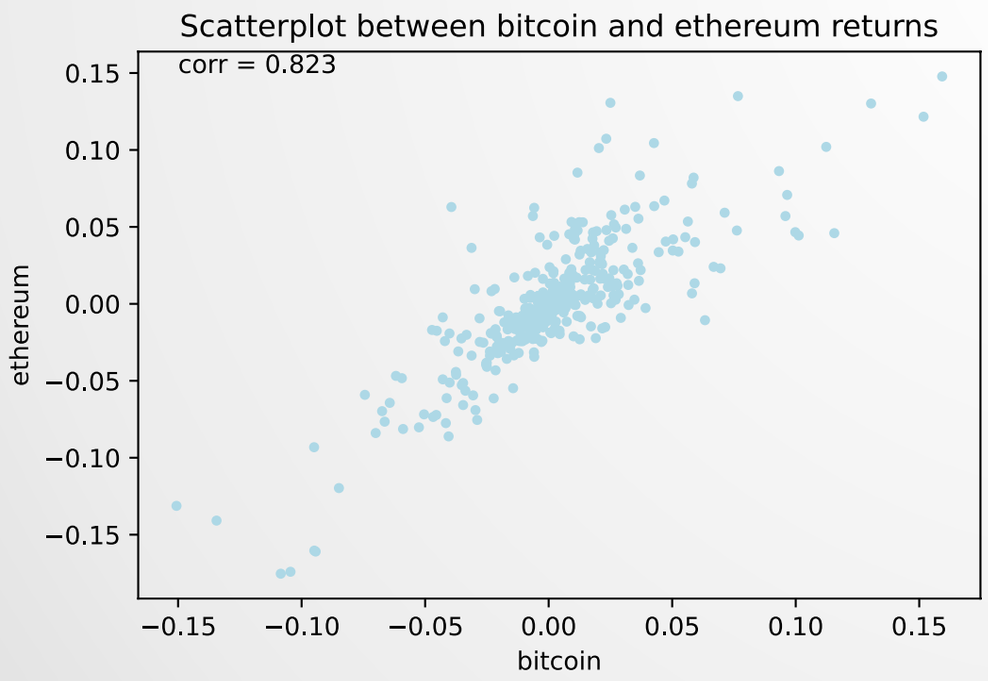


II - Exploratory analysis
6) Scatterplot of correlated CCs 1/4

Observation

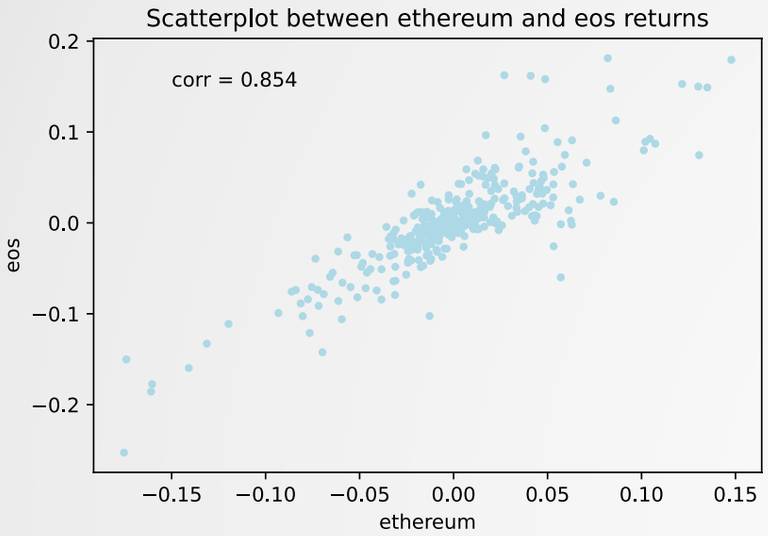
Our analysis shows that 16 CC pairs are highly correlated in their log returns (i.e. bigger than 0.8). As an example, the scatter plot of bitcoin's ethereum's returns is shown to the right. This correlation is an indication that certain CCs react similarly to a third variable. Therefore, we would expect to be able to have some success in clustering later on.

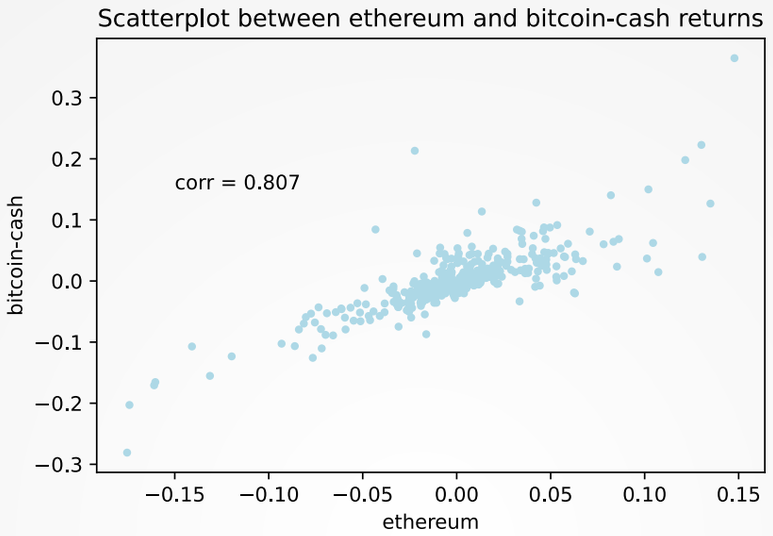
The table on the right shows the concerned CC pairs whose scatterplots are displayed in next two slides.

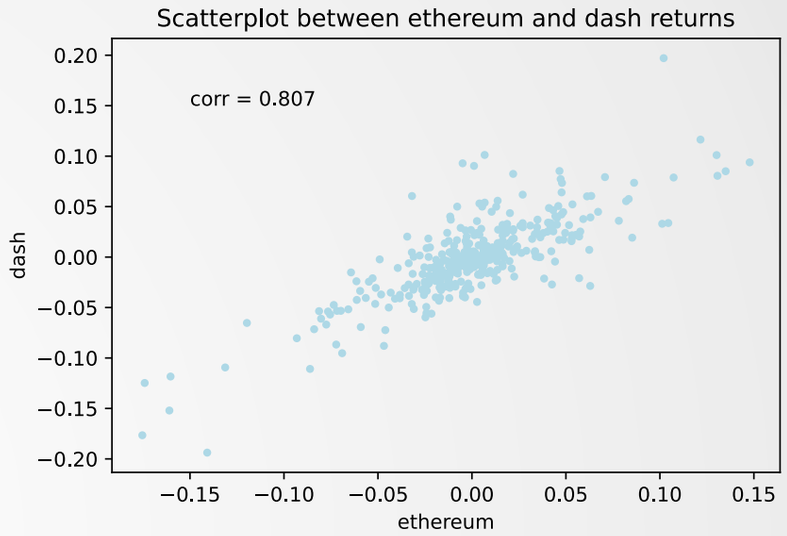


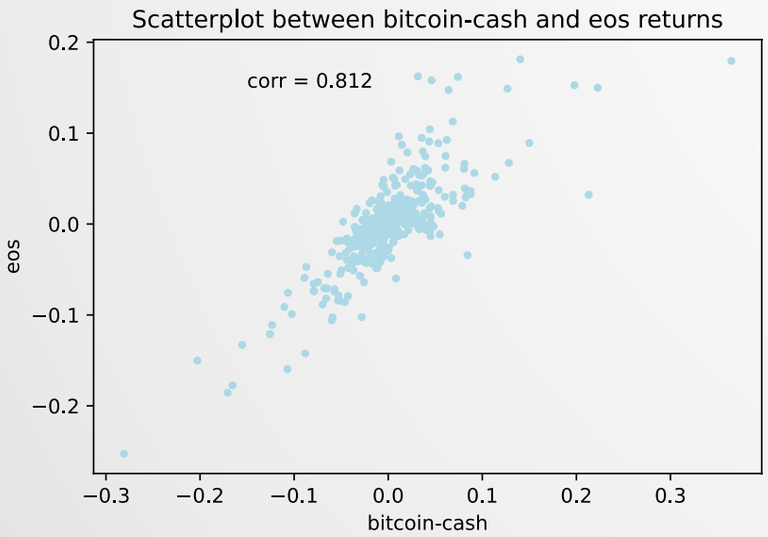
Crypto 1	Crypto 2	Pair correlation
bitcoin	ethereum	823
ethereum	ripple	807
ethereum	litecoin	822
ethereum	bitcoin-cash	807
ethereum	cardano	824
ethereum	eos	854
ethereum	dash	807
ripple	cardano	814
litecoin	bitcoin-cash	805
litecoin	cardano	812
litecoin	eos	836
bitcoin-cash	eos	812
cardano	eos	815
neo	ontology	819
neo	qtum	819
omisego	qtum	835

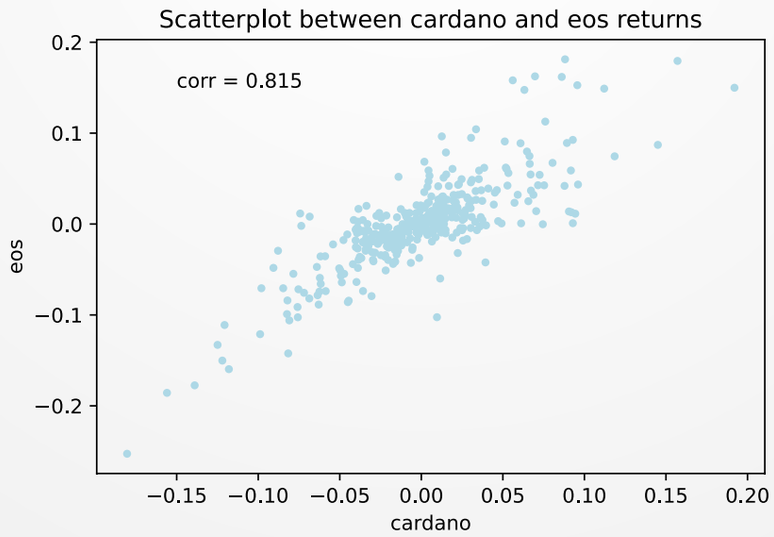
II - Exploratory analysis
6) Scatterplot of correlated CCs 2/4

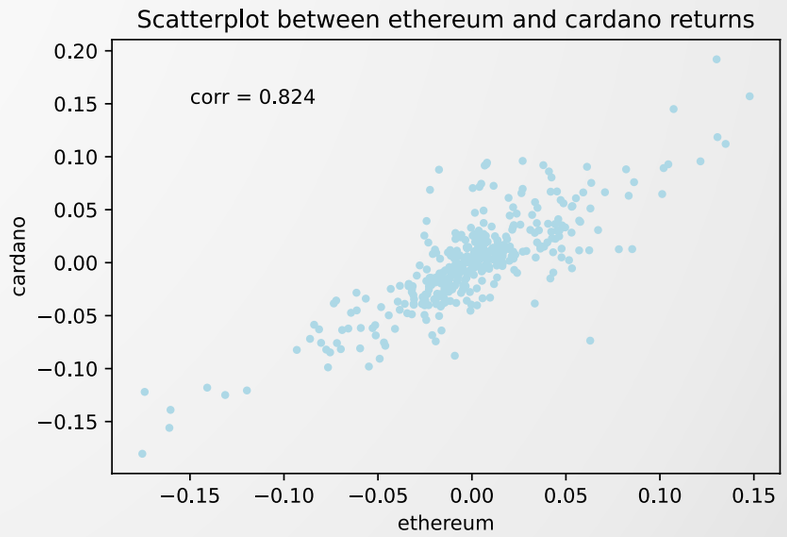




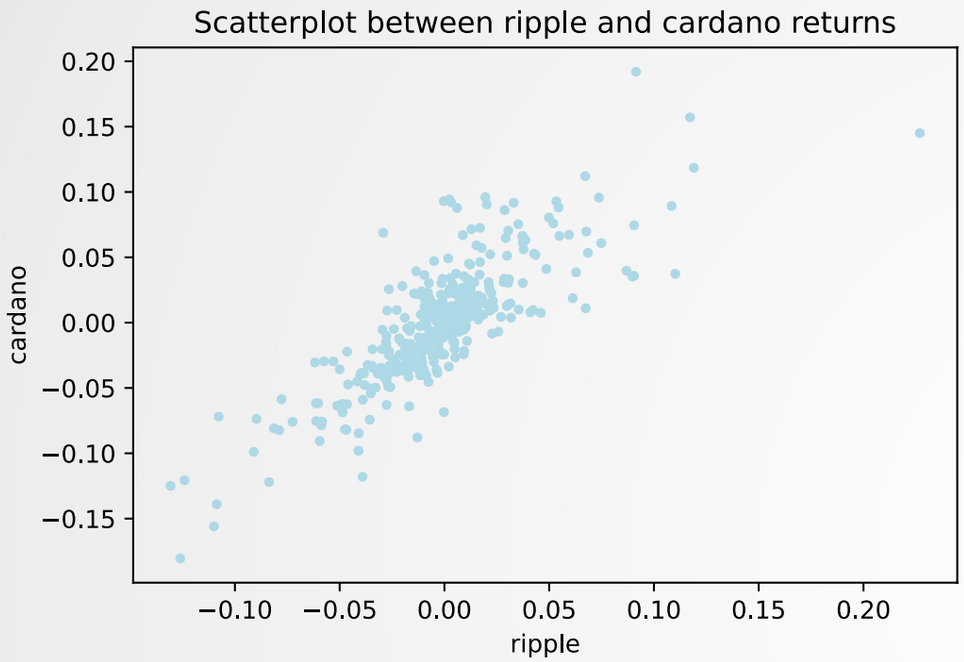


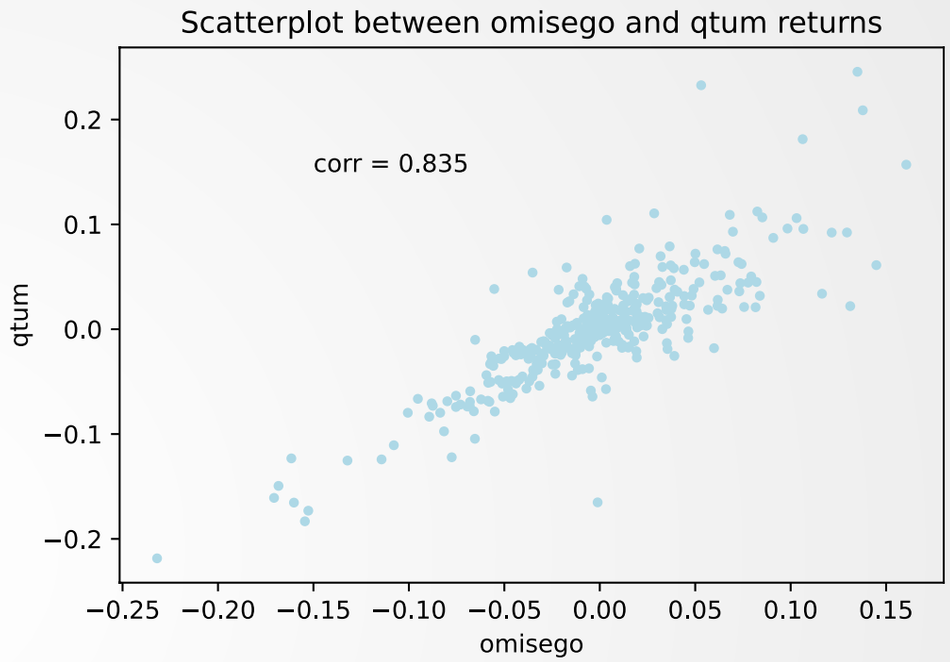


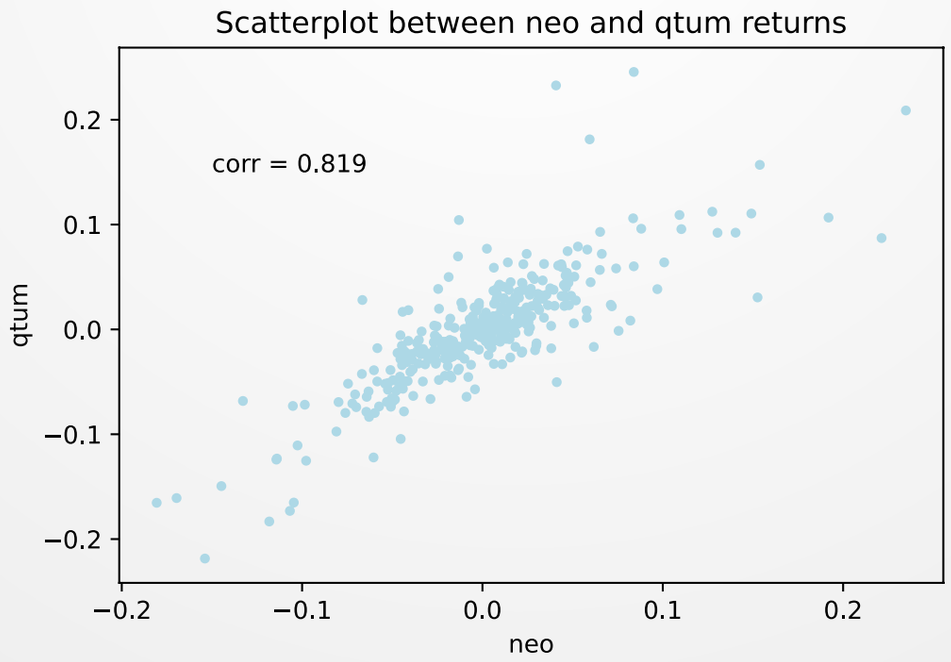




II - Exploratory analysis
6) Scatterplot of correlated CCs 4/4



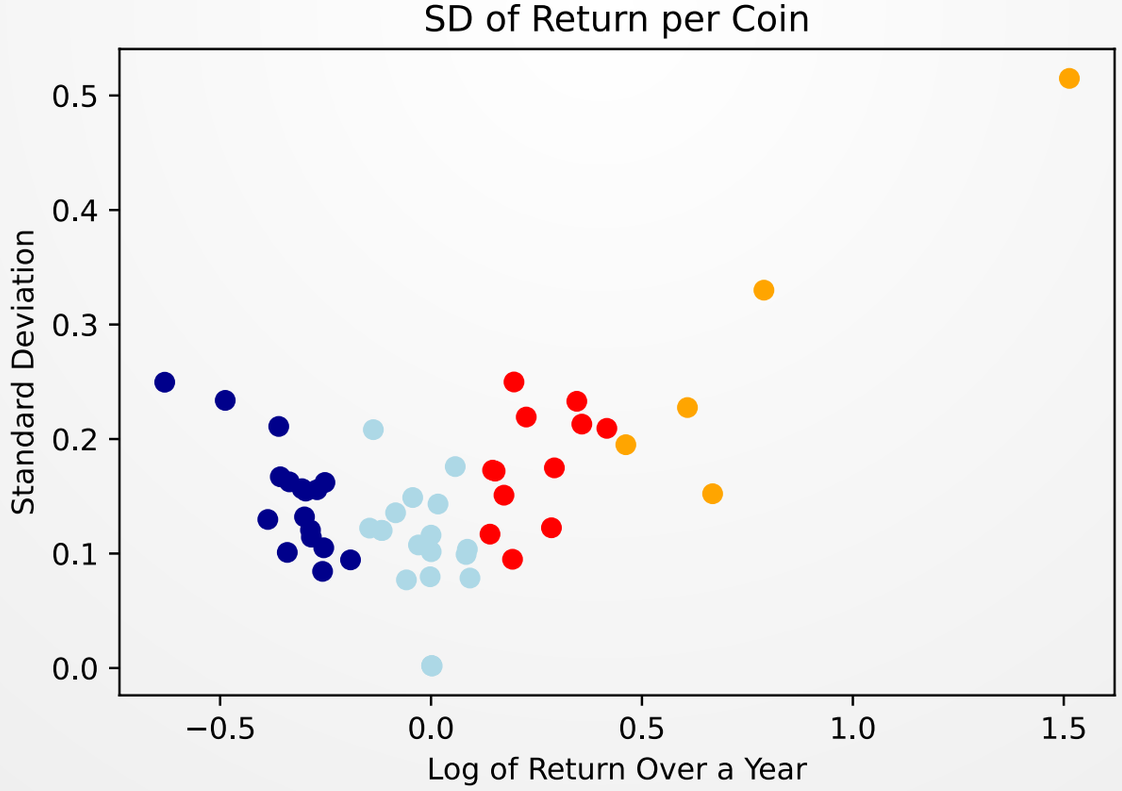




II - Exploratory analysis
7) Exploratory clustering with Kmeans

Observation

As a warm-up for the most sophisticated clustering efforts in the second part, we applied the Kmeans method to our data set. While the clustering has been successful and the presence of clusters is plausible, we also see that there tends to be a linear relationship between many coins' returns and standard deviations, respectively a quadratic relationship with a minimum around (0|0)



III - Clustering - Variables & Methods used

Variables:

- Log-returns
- Variances
- Skewness
- Kurtosis
- Quantile ($\alpha \leq 0.05$)
- Quantile ($\alpha \leq 0.10$)
- Quantile ($\alpha \leq 0.15$)
- Quantile ($\alpha > 0.85$)
- Quantile ($\alpha > 0.90$)
- Quantile ($\alpha > 0.95$)

We used a **rolling window on 14 days** each time; We then shifted from 365 to a final number of **351 steps** in our time-series.

Methods:

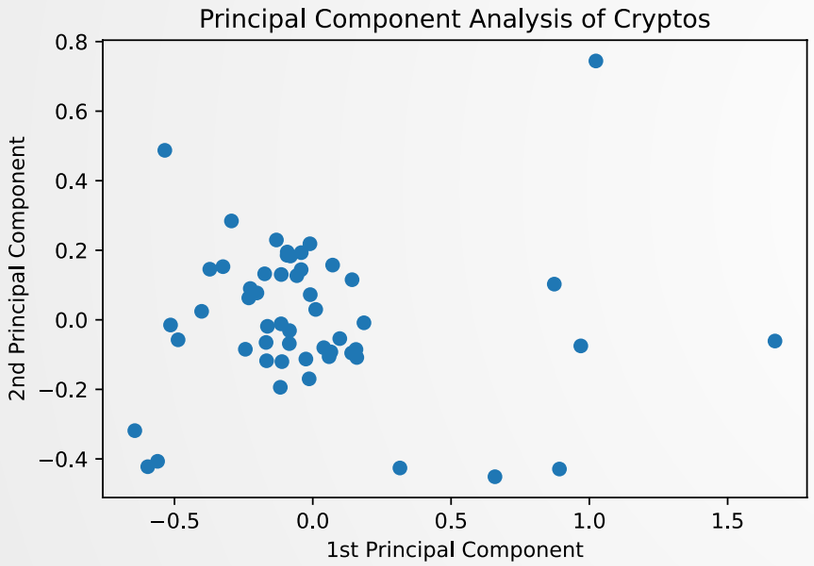
To reduce the information, we tried two different methods: A **Principal Component Analysis** (PCA) and a **Linear Factor Model** (LFM).

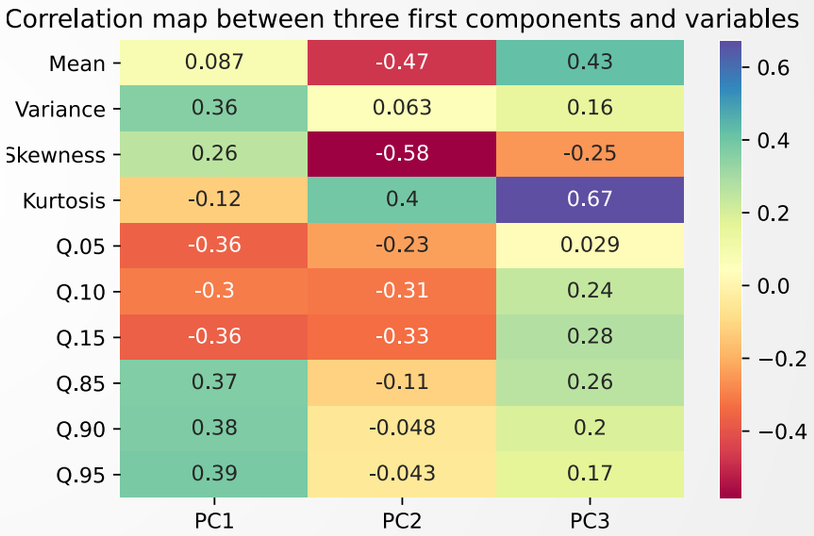
Then, we used the **K-Means** method to cluster the data.

III - Clustering
Method 1: PCA + Means

Principal components are linear combinations of the variables. The principal component analysis explains the total variance in order to reduce the information.

Kaiser-Guttman rule cannot be applied in our case as none of our component has a value greater than 1. We used a PCA with 3 components, even if they are "weak" in order to have part of the variance explained. This is the main limitation of this method. The figure on the left represents the cc log returns on the two first components for the first window.

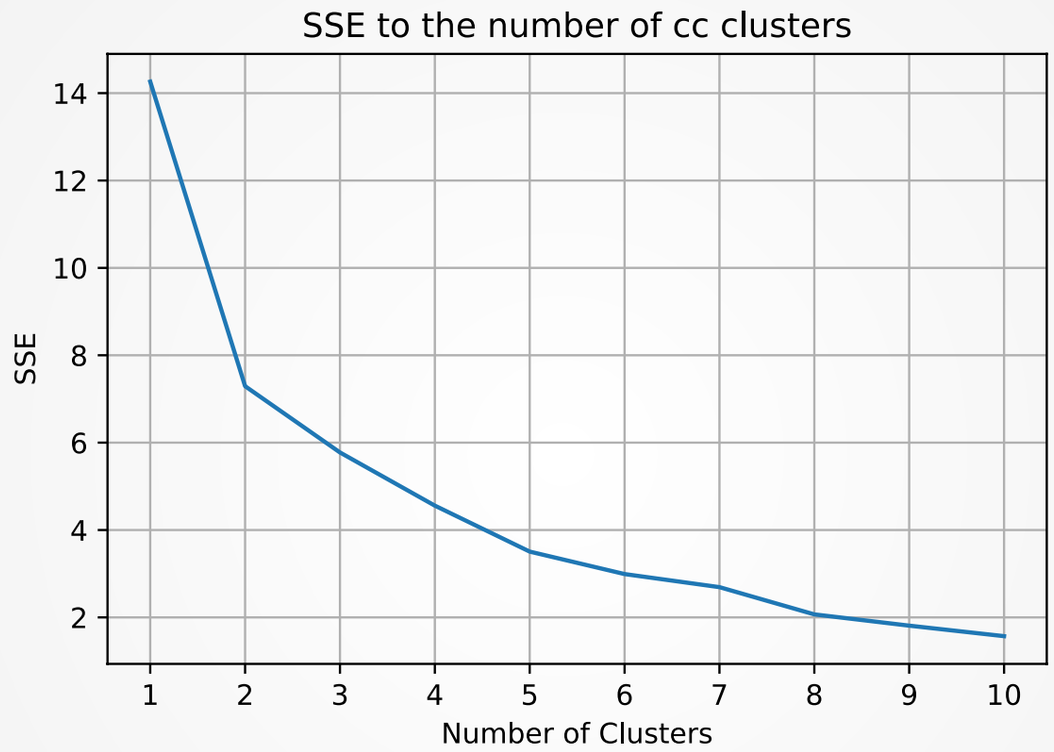




On the right, a correlation map between the selected components and the variables is displayed. While the **first component** seems to be more correlated to the different **quantiles**, the **two others** seem rather to explain the information contained with the **first moments**. Precisely, the **third component** present a high correlation to the **kurtosis**.

III - Clustering
Method 1: PCA + Kmeans

We enter these components in a K-Means clustering method with 3 components.



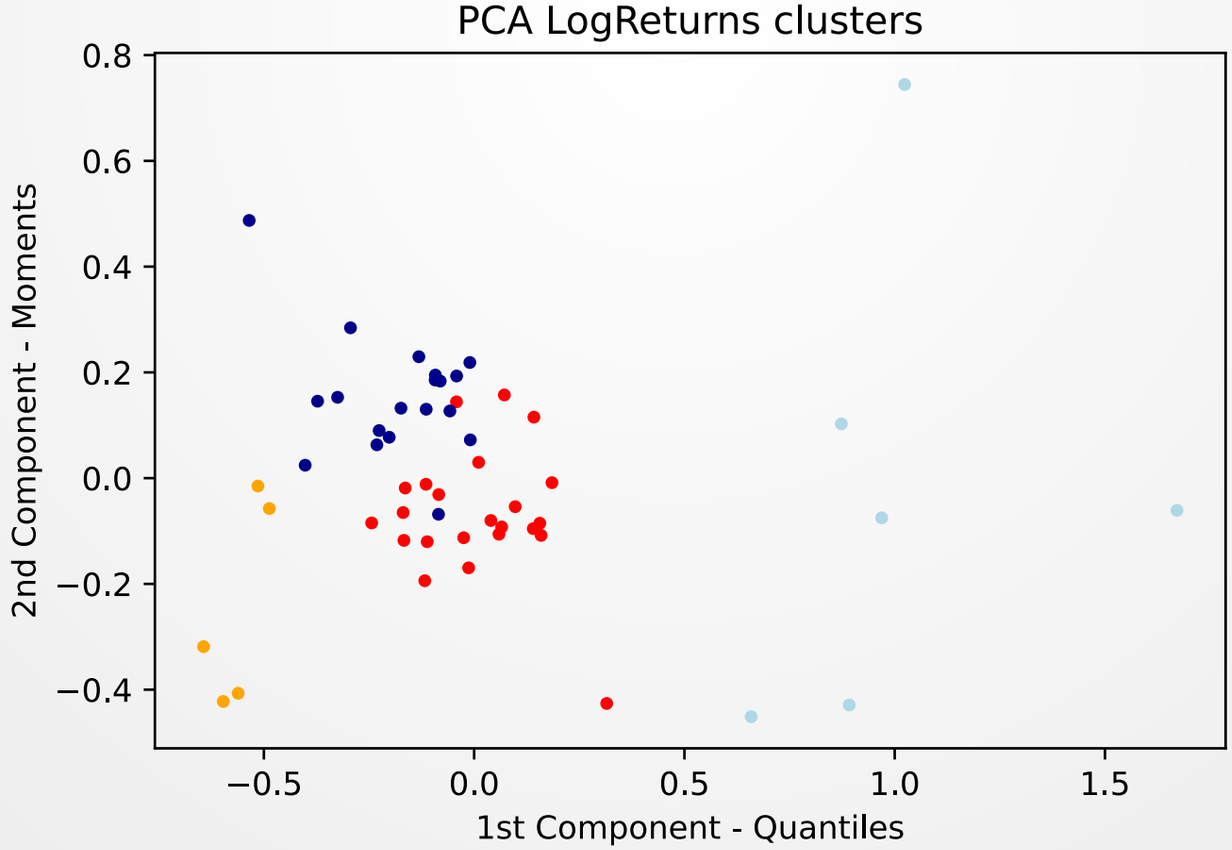
After iterating on many windows, we can see than **4 clusters** seems to appear as a good choice. Indeed, we need a trade-off between the SSE and the number of clusters. The more clusters we have, the lower the SSE. However, too many clusters would not be useful for our analysis.

Finally, as the data in all the windows act differently, it is not always the optimal choice for every window.

III - Clustering
Method 1: PCA + Kmeans: Clustering the first rolling window

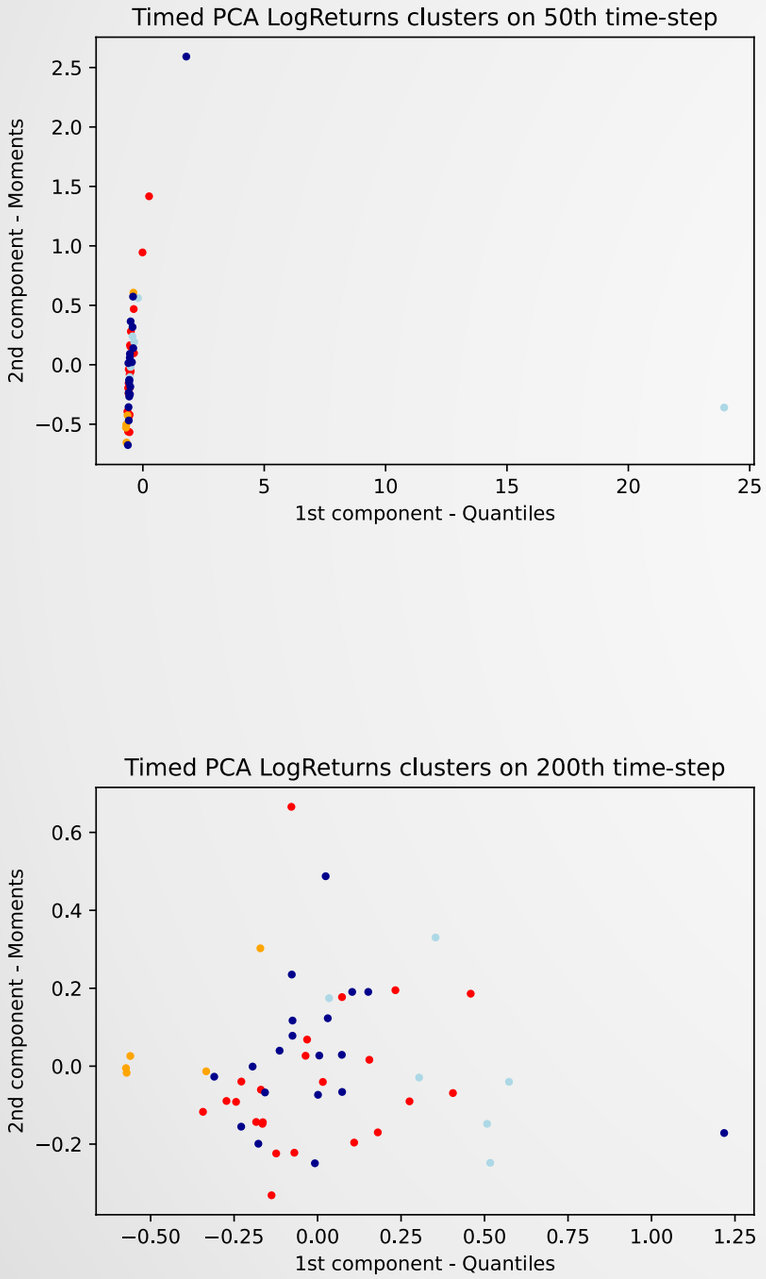
Since we had chosen our number of clusters (4) based on the SSE criteria, we were able to cluster cc log returns. Those have been then clustered for the first rolling window, as showed on figure below.

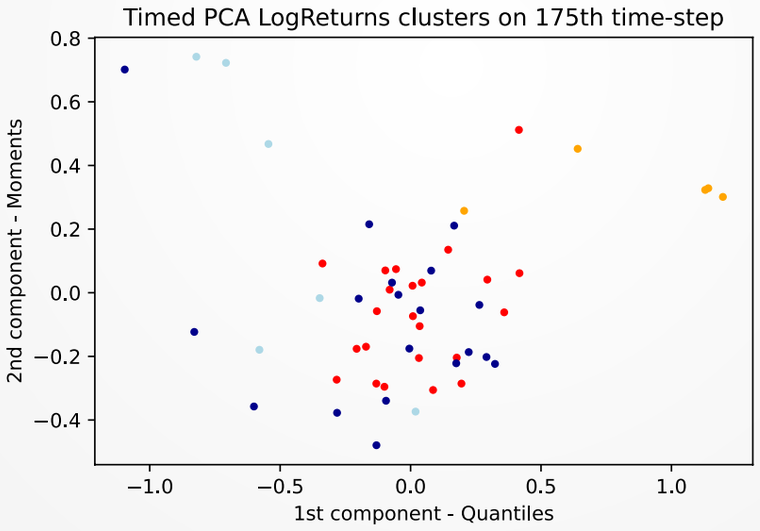
Results are quite encouraging since cc seem to be well differentiated on these two dimensions. However, we can observe the limit of choosing 4 clusters. Indeed, dark blue and red clusters might be gathered into a single cluster.

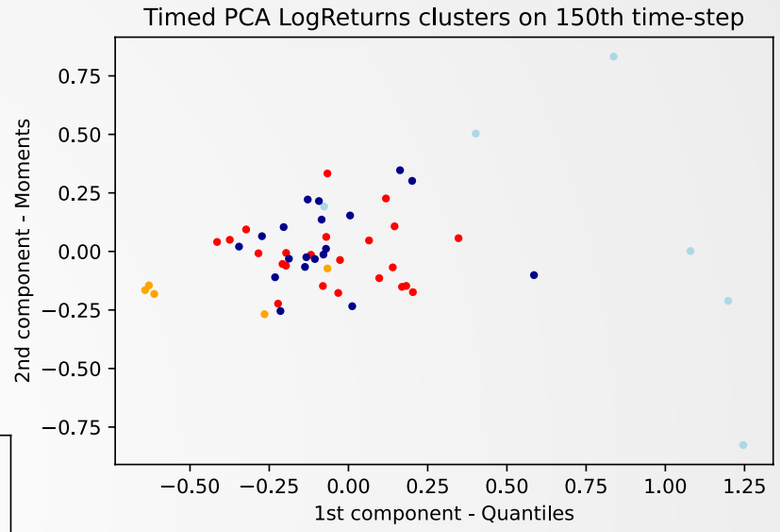


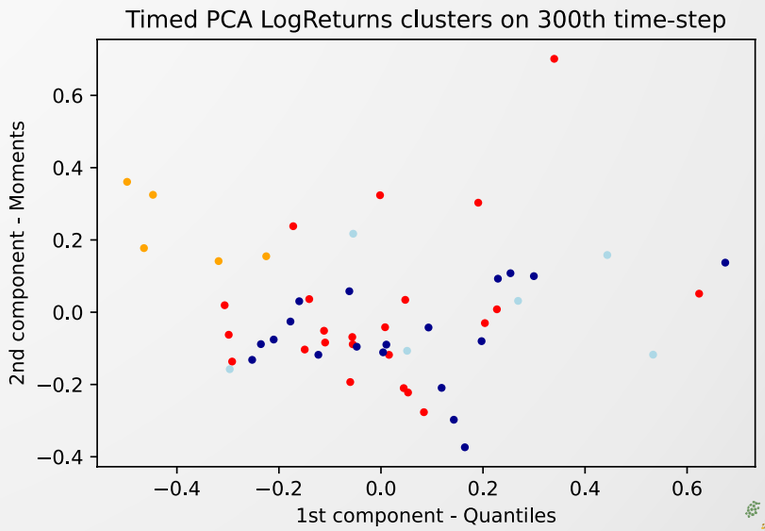
III - Clustering
Method 1: PCA + Kmeans: Clustering on all the windows

Then, this clustering method was applied to all rolling windows. It gave the following clusters for arbitrary chosen rolling windows.









III - Clustering
Method 1: PCA + Kmeans: Clustering on all the windows

Conclusions from last slide are obviously difficult to draw due to the few number of rolling windows illustrated and from the reduced number of dimensions plotted (2 instead of 3).

For this reason, the **distribution** of all CC log returns in each cluster has been computed. It shows thus, in which percentage is a given cc found in a given cluster.

Red values represent the clusters in which CC log returns are most of the time in. We can observe than, except for a few, cc are **highly concentrated in the first cluster**.

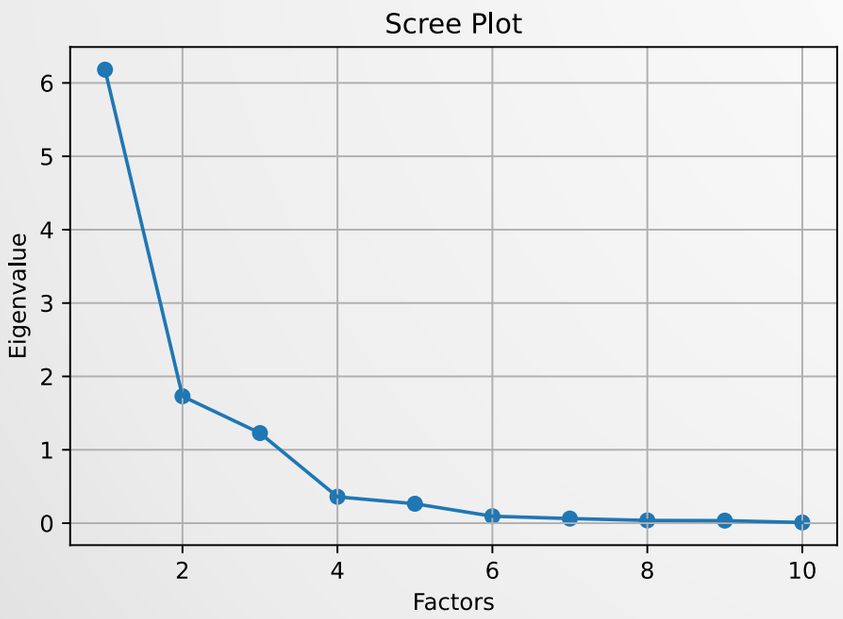
	Cluster1	Cluster2	Cluster3	Cluster4		Cluster1	Cluster2	Cluster3	Cluster4		Cluster1	Cluster2	Cluster3	Cluster4
bitcoin	**0.46**	0.16	0.17	0.21	**celsius-degree-token**	0.24	0.23	**0.32**	0.21	**nexo**	**0.35**	0.21	0.24	0.19
ethereum	**0.46**	0.17	0.19	0.18	**theta-token**	**0.42**	0.19	0.23	0.16	**basic-attention-token**	**0.41**	0.18	0.24	0.17
ripple	**0.45**	0.17	0.19	0.19	**dash**	**0.5**	0.17	0.18	0.15	**digibyte**	**0.48**	0.17	0.2	0.15
litecoin	**0.46**	0.17	0.19	0.18	**vechain**	**0.41**	0.19	0.21	0.19	**0x**	**0.46**	0.2	0.2	0.14
bitcoin-cash	**0.55**	0.17	0.14	0.14	**havven**	0.16	**0.3**	0.29	0.25	**true-usd**	**0.36**	0.13	0.21	0.3
chainlink	**0.41**	0.17	0.24	0.19	**huobi-token**	**0.41**	0.17	0.21	0.21	**qtum**	**0.49**	0.2	0.16	0.15
binancecoin	**0.51**	0.15	0.19	0.15	**iota**	**0.54**	0.14	0.19	0.14	**republic-protocol**	**0.3**	0.24	0.26	0.19
cardano	**0.49**	0.15	0.21	0.15	**zcash**	**0.51**	0.17	0.18	0.15	**swissborg**	**0.33**	0.25	0.24	0.18
stellar	**0.52**	0.16	0.16	0.16	**waves**	**0.49**	0.15	0.21	0.16	**icon**	**0.49**	0.2	0.16	0.14
usd-coin	**0.37**	0.12	0.21	0.3	**ethereum-classic**	**0.47**	0.21	0.17	0.15	**loopring**	**0.39**	0.22	0.21	0.17
bitcoin-cash-sv	**0.46**	0.19	0.21	0.14	**zilliqa**	**0.44**	0.17	0.25	0.13	**lisk**	**0.48**	0.19	0.18	0.15
eos	**0.42**	0.19	0.21	0.18	**dogecoin**	**0.49**	0.17	0.18	0.16	**kyber-network**	**0.28**	0.26	0.26	0.21
nem	**0.46**	0.21	0.15	0.17	**maker**	**0.47**	0.21	0.17	0.15	**quant-network**	0.19	0.27	**0.28**	0.25
tron	**0.49**	0.16	0.21	0.14	**decred**	**0.5**	0.19	0.16	0.15	**bitcoin-gold**	**0.6**	0.13	0.16	0.1
okb	**0.39**	0.21	0.21	0.18	**omisego**	**0.51**	0.16	0.2	0.12	**maidsafecoin**	**0.46**	0.22	0.2	0.12
tezos	**0.39**	0.19	0.29	0.12	**ontology**	**0.46**	0.18	0.2	0.16	**vitae**	0.11	**0.44**	0.24	0.21
neo	**0.49**	0.16	0.21	0.14	**paxos-standard**	**0.37**	0.12	0.21	0.3	**siacoin**	**0.5**	0.17	0.19	0.14
										nano	**0.45**	0.19	0.21	0.15
										enjincoin	**0.29**	0.26	0.28	0.16

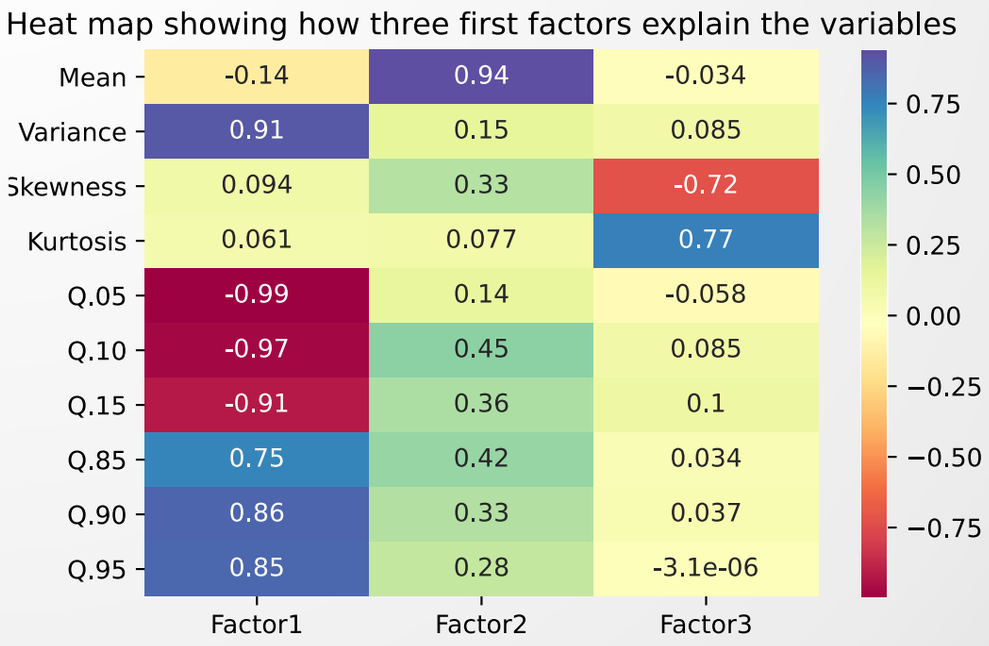
III - Clustering
Method 2: Factor analysis + Kmeans: Reducing the number of dimensions

The linear factor analysis enables to reduce the number of dimensions of a given database by transforming initial variables into factors. These factors are linear transformations of the initial variables.

Their determination is based on the eigenvalues of the initial database. The larger are the eigenvalues, the better are these factors explaining the behaviour of the variables. Only eigenvalues superior to 1 are considered since it means that a factor is able to explain more than a single variable.

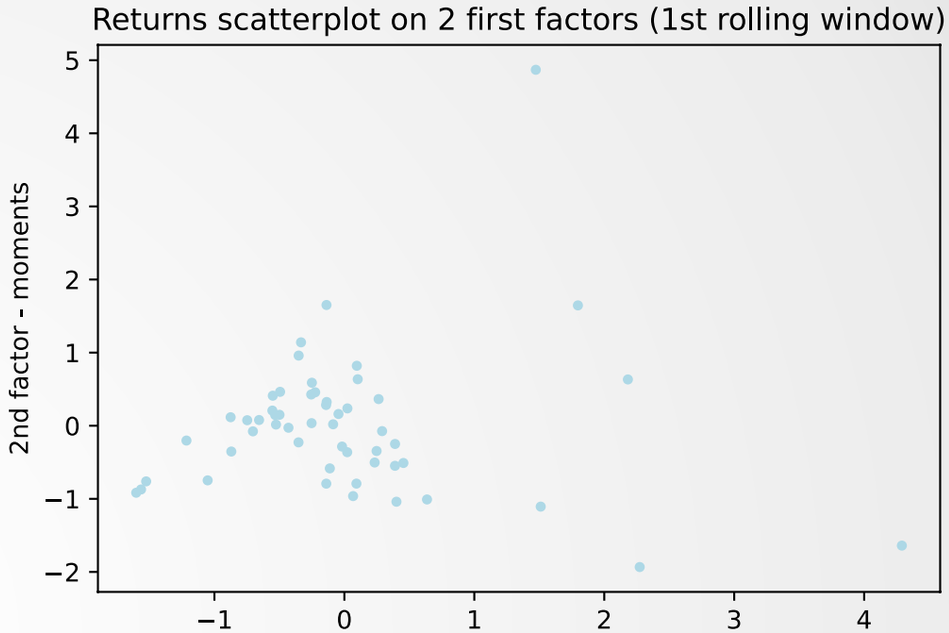
As displayed on this scree plot, from the 10 initial variables, **3 eigenvalues** are superior to 1. Then, it represents the number of **linear factors** we will keep working with. Similarly to the PCA, the **1st factor** seems to explain the **quantiles** while the **others** explain the first **moments**.

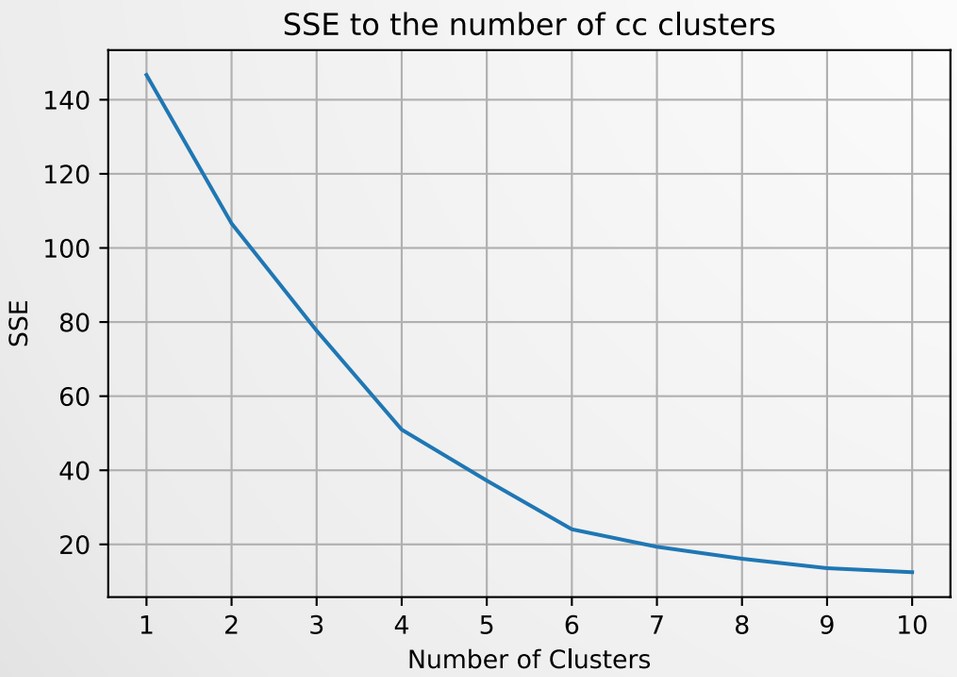




III - Clustering
Method 2: Factor analysis + Kmeans: Determining the number of clusters

Before clustering, let's have a look to the projection of CC log returns on the two first factors (in the first rolling window). From this projection, we can observe that there is a **high concentration of CC in bottom-left** and then some cc are dispersed on the right. Obviously, it might have been interesting to project these CC log-returns on the three factors but we decided to limit ourselves to 2D figures.



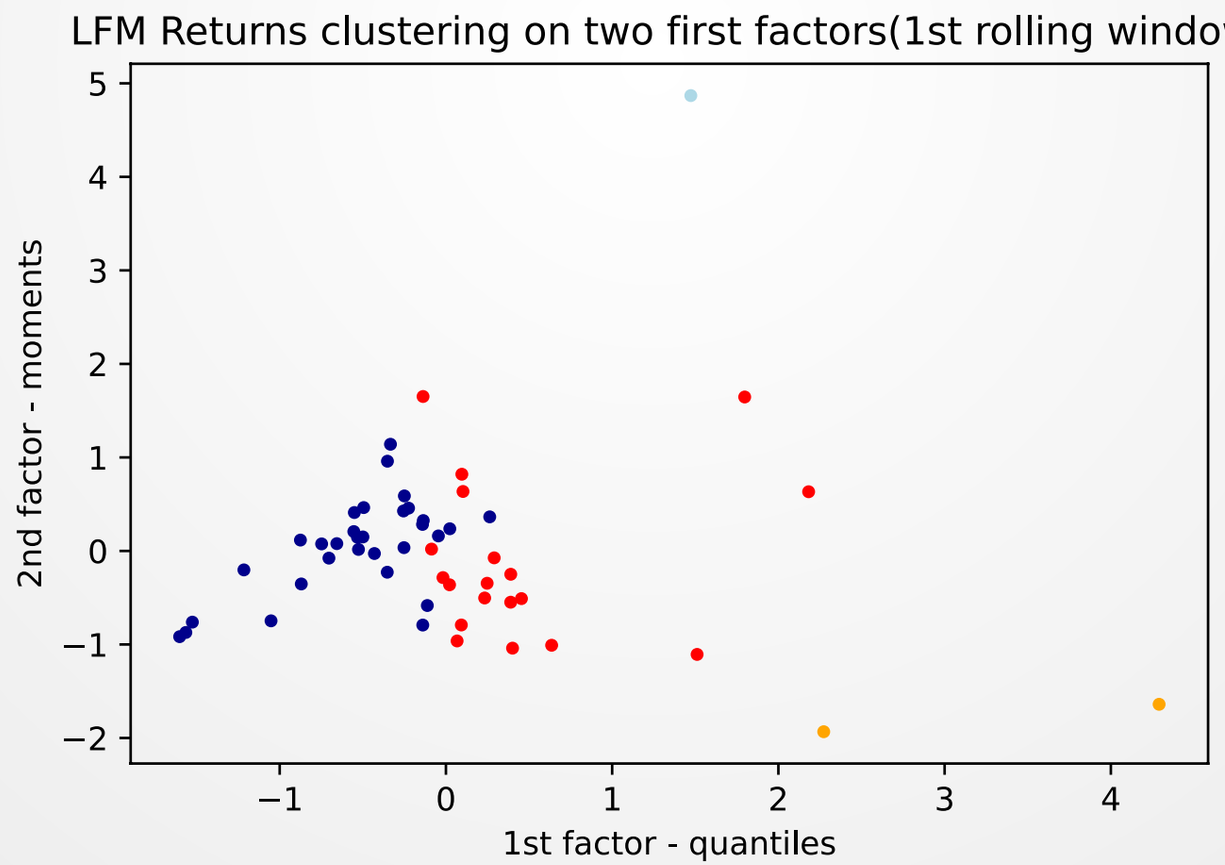


Then, the number of clusters needs to be defined. Again, the sum of squared errors will be used as main criteria. From this figure on the left, a fair tradeoff seems to be **4 clusters**. Indeed, it allows to have an acceptable SSE (not too high) without having too much different clusters (too many clusters is not interesting since it leads to only show the behaviour of a few CC).

III - Clustering
Method 2: Factor analysis + Kmeans: Clustering the first rolling window

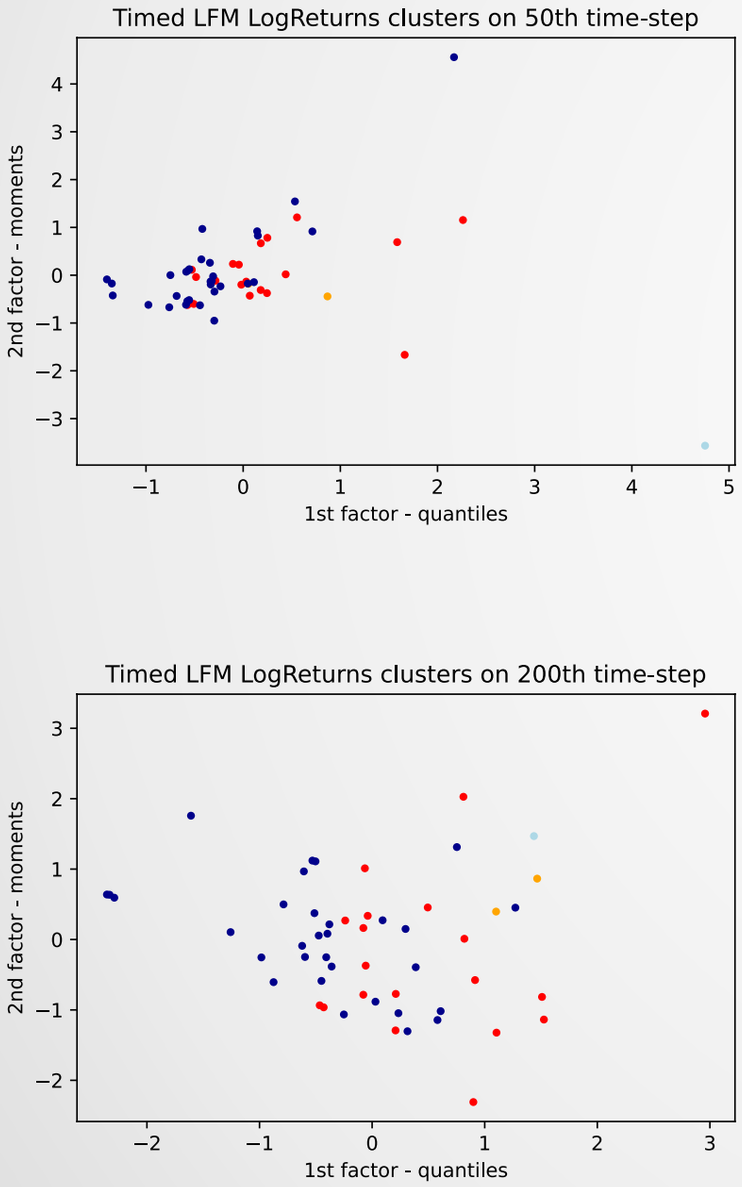
Since we had chosen our number of clusters (4) based on the SSE criteria, we were able to cluster CC log returns. Those have been then clustered for the first rolling window, as showed on left figure.

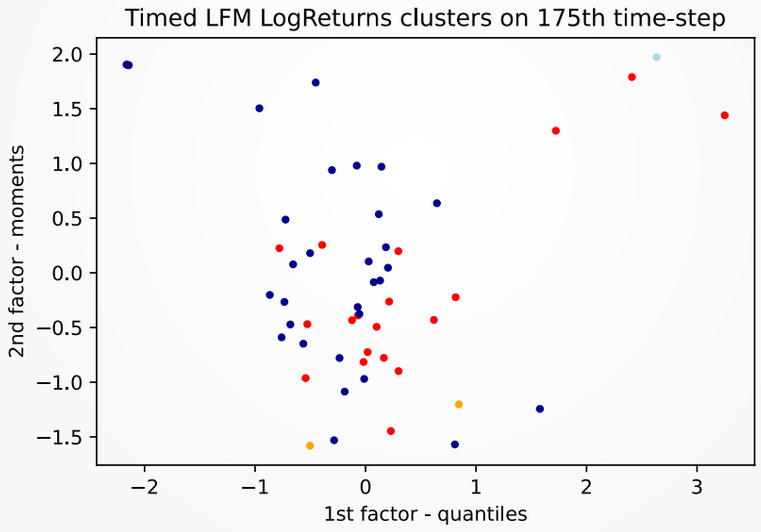
Results are quite encouraging since CC on bottom-left seem to be well differentiated. However, we can observe the limit of choosing 4 clusters since there is only one CC in the second cluster (in light blue).

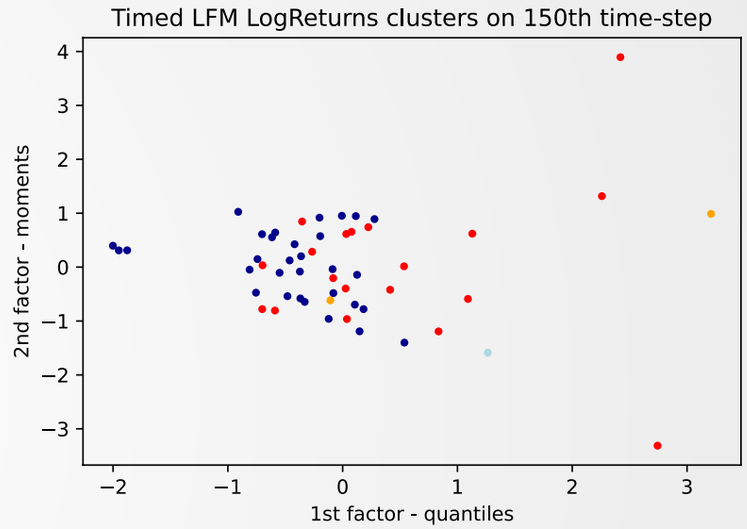


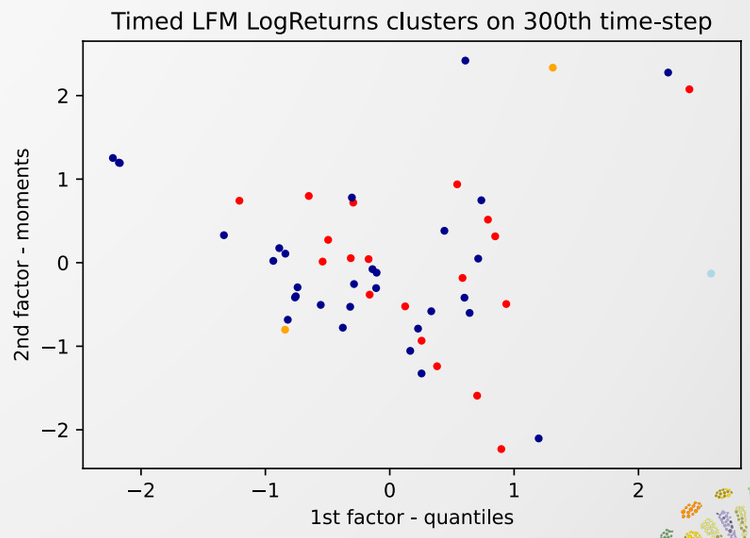
III - Clustering
Method 2: Factor analysis + Kmeans: Clustering on all windows

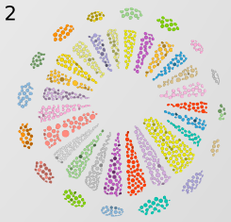
Then, this clustering method was applied to all rolling windows. It gave the following clusters for arbitrary chosen rolling windows.











III - Clustering
Method 2: Factor analysis + Kmeans: Clustering on all windows

Conclusions from last slide are obviously difficult to draw due to the few number of rolling windows illustrated and from the reduced number of dimensions plotted (2 instead of 3).

For this reason, the **distribution** of all CC log returns in each cluster has been computed. It shows thus, in which percentage is a given cc found in a given cluster.

Red values represent the clusters in which CC log returns are most of the time in. We can observe than, except for a few, CC are **highly concentrated in the first cluster**.

	Cluster1	Cluster2	Cluster3	Cluster4		Cluster1	Cluster2	Cluster3	Cluster4		Cluster1	Cluster2	Cluster3	Cluster4
bitcoin	**0.45**	0.23	0.15	0.17	**celsius-degree-token**	0.25	0.27	**0.28**	0.2	**nexo**	**0.3**	0.26	0.2	0.24
ethereum	**0.44**	0.25	0.14	0.17	**theta-token**	**0.36**	0.28	0.16	0.2	**basic-attention-token**	**0.33**	0.2	0.26	0.21
ripple	**0.42**	0.24	0.15	0.19	**dash**	**0.47**	0.23	0.12	0.18	**digibyte**	**0.39**	0.22	0.2	0.19
litecoin	**0.46**	0.22	0.17	0.15	**vechain**	**0.34**	0.22	0.24	0.21	**0x**	**0.3**	0.29	0.21	0.21
bitcoin-cash	**0.49**	0.21	0.14	0.16	**havven**	0.23	0.21	**0.32**	0.24	**true-usd**	**0.4**	0.18	0.15	0.27
chainlink	**0.35**	0.24	0.2	0.21	**huobi-token**	**0.34**	0.26	0.2	0.21	**qtum**	**0.39**	0.29	0.17	0.14
binancecoin	**0.47**	0.21	0.14	0.18	**iota**	**0.46**	0.21	0.14	0.19	**republic-protocol**	**0.29**	0.26	0.21	0.24
cardano	**0.44**	0.3	0.12	0.13	**zcash**	**0.48**	0.23	0.14	0.15	**swissborg**	0.29	**0.31**	0.19	0.2
stellar	**0.43**	0.23	0.15	0.19	**waves**	**0.4**	0.24	0.17	0.19	**icon**	**0.39**	0.28	0.16	0.17
usd-coin	**0.4**	0.19	0.15	0.27	**ethereum-classic**	**0.45**	0.26	0.15	0.14	**loopring**	**0.35**	0.31	0.17	0.17
bitcoin-cash-sv	**0.38**	0.21	0.23	0.17	**zilliqa**	**0.39**	0.25	0.17	0.19	**lisk**	**0.36**	0.25	0.19	0.19
eos	**0.41**	0.26	0.15	0.18	**dogecoin**	**0.39**	0.24	0.16	0.21	**kyber-network**	0.26	**0.31**	0.19	0.23
nem	**0.41**	0.24	0.16	0.19	**maker**	**0.4**	0.22	0.18	0.21	**quant-network**	0.18	**0.33**	0.25	0.24
tron	**0.4**	0.26	0.17	0.17	**decred**	**0.41**	0.22	0.16	0.21	**bitcoin-gold**	**0.52**	0.22	0.11	0.15
okb	**0.35**	0.26	0.2	0.19	**omisego**	**0.45**	0.24	0.16	0.15	**maidsafecoin**	**0.35**	0.27	0.19	0.19
tezos	**0.44**	0.22	0.18	0.16	**ontology**	**0.39**	0.25	0.17	0.19	**vitae**	0.12	0.29	0.27	**0.31**
neo	**0.45**	0.26	0.17	0.13	**paxos-standard**	**0.4**	0.18	0.15	0.27	**siacoin**	**0.44**	0.2	0.19	0.17
										nano	**0.41**	0.26	0.18	0.15
										enjincoin	0.26	**0.33**	0.2	0.2

IV - Discussion of the results (1/2)

We can draw same conclusions for the two methods (even if they do not show exactly the same clusters). The cryptocurrencies are mainly in the same cluster (the first one). In other words, they do not differentiate one from another in their behaviour on average. There are different explanations for that:

- We use cryptocurrencies with the **highest market capitalization**. Those are the bigger ones, which means that they succeed on the market. They are more likely to act in the same way.
- We use only data from **only one year** (2019). Taking a longer period would maybe yield bigger differences.
- We delete cryptocurrencies with null values. Consequently, we have only the **"surviving cryptocurrencies"**. The ones that disappeared during this period are not in our sample.
- Maybe by using other variables we would have larger gaps between the cryptocurrency patterns.
- The **size of the rolling window** has an impact. With higher rolling window, we notice that, even if they are still concentrated, cryptocurrencies start to act differently as they are a more dispersed through the cluster. Consequently, we can say that through longer periods, cryptocurrencies can distinguish more distinctly one from another.

IV - Discussion of the results (2/2)

Following the exploratory analysis, we saw that the cryptocurrencies have the same pattern and our cluster analysis confirms that. Moreover, a few exceptions act sightly differently by being more often in another cluster.

PCA vs Factor Analysis

Due to the weak eigenvalues (lower than 1) of its components, the results of the PCA are not as strong as the ones of the Factor Analysis. Indeed, the components of the PCA might not take enough information into account.

However, both analysis have the same conclusions. The distribution of the cryptocurrencies in the clusters change throughout the windows, but the majority of the cryptocurrencies are more often in the first cluster.

The comparison of the two methods might be redundant as in both cases the first factors/components seem to explain the same variables.

IV - Limitations of the analysis

Limitations

- We only looked at a limited set of coins in a limited set of time
- There are outliers we did not study closer
- PCA analysis not well suited for the analysis (due to the low eigenvalues of its components)
- The high market capitalization of Bitcoin can disturb results. Our study could be repeated without it

Further Studies

- Increase the frequency of the observations
- Use other variables
- Study longer and/or different time frames
- Avoid survivorship bias by including coins that have *some* missing values and deal with the gap in the data in an appropriate manner
- Consider different clustering methods