Notes

Tuesday, August 9, 2022 8:06 PM

Myroslava Sánchez Andrade A01730712 | 09/08/2022

Descriptive Statistics

Used to summarize information from raw data.
Most important descriptive statistics:

- Central tendency measures

Attempts to describe a data set with a single value which represents the middle or center of its distribution.
Main central tendency measures:

- Arithmetic mean

Average value of valid values of a variable X (assuming each value has the same importance), being X an attribute of a subject.

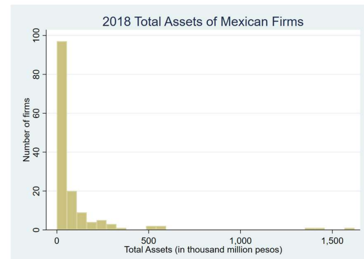
$$\bar{X} = \frac{\sum_{i=1}^{N} X_i}{N}$$

- Median

The median of a variable is the its 50 percentile, mid-point of its values sorted in descending order. If two middle points, the median will be the arithmetic average of them.

When a variable **does not follow a probability distribution** close to normal distribution, the **best measure** for central tendency is the **median**.

The mean is sensible to extreme values.



Histogram skewed to the right

- Mode

Value that most appear in a variable (calculated for discrete variables).

- Dispersion measures

Used to measure how much on average the individual values of a variable change from the mean. Variance and standard deviation reflect variability in a distribution.

- Variance and the standard deviation

The variance of a variable X is the average of squared deviations (difference between the observed values of a variable and some other value) from each individual value X_i from its mean:

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n} (X_i - \bar{X})^2 = \sigma_X^2$$

The variance is expressed in much larger units

Where:

X_i = Value i of the variable X

$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ = Arithmetic average of X

It is more used the sample variance (denominator n-1), instead of the population variance (denominator n). The *sample variance* is a more conservative value of the variance.

Rewriting the formula:

$$Var(X) = \frac{1}{(n-1)}\sum_{i=1}^{n} (X_i - \bar{X})^2 = \sigma_X^2$$

$$SD(X) = \sqrt{Var(X)} = \sqrt{\frac{1}{(n-1)}\sum_{i=1}^{n} (X_i - \bar{X})^2}$$

$$SD(X) = \frac{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}}{\sqrt{(n-1)}} = \sigma_X$$

The standard deviation is expressed in the same units as the original values

Data management

- Data transformations

- Return calculation

The return of a price is the % change of the price from one period (present period t) to the next (previous period t-1).

$$R_t = \frac{(price_t - price_{t-1})}{price_{t-1}} = \frac{price_t}{price_{t-1}} - 1$$

It is very recommended to calculate continuously compounded returns (cc returns) and cc returns instead of simple returns. Cc returns are calculated from the natural logarithm of prices.

- Natural logarithm

The natural logarithm of a number is the **exponent** that the number *e* (=2.71…) needs to be raised to get another number. The natural logarithm is the logarithm of base *e*.

Relation of *e* and the grow of financial amounts over time:
The general formula to get the final amount of an investment at the beginning of year x=1, for any interest rate R can be:

$$I_2 = I_1 * (1 + R)^1$$

The (1+R) is the growth factor of the investment.
But, if the interests are calculated each month, the investment would end up with a higher amount. The general formula would end up like this:

$$I_2 = I_1 * (\ 1 + \frac{R}{N})^{1*N}$$

A **continuously compounded** rate would give as a result the Euler constant for the growth factor.
We can generalize annual interest rate, so that e^R is the growth factor when interests are compunded every moment. On the other hand, when compounding every instant we use e^r . The relationship between the growth rate and an effective equivalent rate would be:

$$EffectiveRate = e^r - 1$$

- Continuously compounded returns

One way to calculate it is subtracting the current price(t) minus the log of the previous price (t-1). (Difference of the log of the price):

$$r_t = \log(price_t) - \log(price_{t-1})$$

Other way would be:

$$r_t = \log(\ \frac{price_t}{price_{t-1}})$$

Histogram

Illustrates how the values of a variable are distributed in its range of values (frequency plot). The most common values, least common values, the possible mean and standard deviation can be appreciated.

Probability Density Functions

- PDF of a discrete variable

Sum of probabilities of x to be equal to a specific value.

$$f(x) = P(X = x_i)$$

We can express the Cumulative Density Function (probability that x will take values less than or equal to x) as:

$$f(x) = \sum_{i=1}^{n} P(X = x_i)$$

- PDF of a continuous variable

Integration of the function f(x), where f(x) is the PDF. We calculate the probability of the continuous variable x to be within a specific range.

$$\int_{-\infty}^{\infty} f(x)\,dx = 1$$

$$\int_{a}^{b} f(x), dx = P(a \le x \le b)$$

Normal Distribution Function

The most popular continuous PDF is the well-known "bell-shaped" normal distribution defined as:

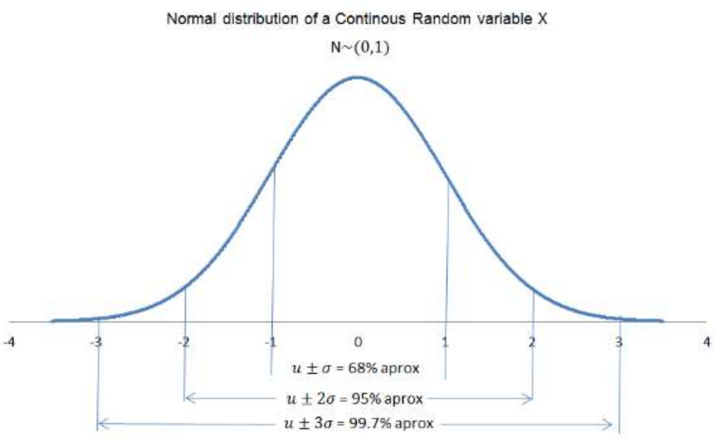
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2})}$$

Where u is the mean of the distribution and ʊ squared is the variance of the distribution. The only two parameters to be defined in order to know the behavior of the continuous random variable x are: the mean of x and the variance of x. The normal distribution is normal around u.

- For the range $(\mu - \sigma) <= x <= (\mu + \sigma)$, the area under the curve is approximately 68%

- For the range $(\mu - 2\sigma) <= x <= (\mu + 2\sigma)$, the area under the curve is approximately 95%

- For the range $(\mu - 3\sigma) <= x <= (\mu + 3\sigma)$, the area under the curve is approximately 99.7%.



2.3 Challenge: Data management and Descriptive Statistics

Wednesday, August 10, 2022 2:07 PM

Myroslava Sánchez Andrade A01730712 | 10/08/2022

Data collection and visualization

```python
import numpy as np
import pandas as pd
import pandas_datareader as pdr
```

Downloading daily prices for Bitcoin from 2017:

```python
BTC = pdr.get_data_yahoo('BTC-USD',
start="01/01/2017", interval="d")
```

Printing the content of the data

```python
BTC
```

	High	Low	Open	Close	Volume	
Date						
2017-01-01	1003.080017	958.698975	963.658020	998.325012	147775008	9
2017-01-02	1031.390015	996.702026	998.617004	1021.750000	222184992	10
2017-01-03	1044.079956	1021.599976	1021.599976	1043.839966	185168000	10
2017-01-04	1159.420044	1044.400024	1044.400024	1154.729980	344945984	11
2017-01-05	1191.099976	910.416992	1156.729980	1013.380005	510199008	10
...	...	...	...	...	...	
2022-08-06	23326.562500	22961.279297	23291.423828	22961.279297	15978259885	229
2022-08-07	23359.009766	22894.556641	22963.505859	23175.890625	15886817043	231
2022-08-08	24203.689453	23176.546875	23179.527344	23809.486328	28575544847	238
2022-08-09	23898.615234	22982.000000	23811.484375	23164.318359	23555719219	231
2022-08-10	24126.781250	22773.726562	23126.421875	24018.003906	28472848384	240

2048 rows × 6 columns

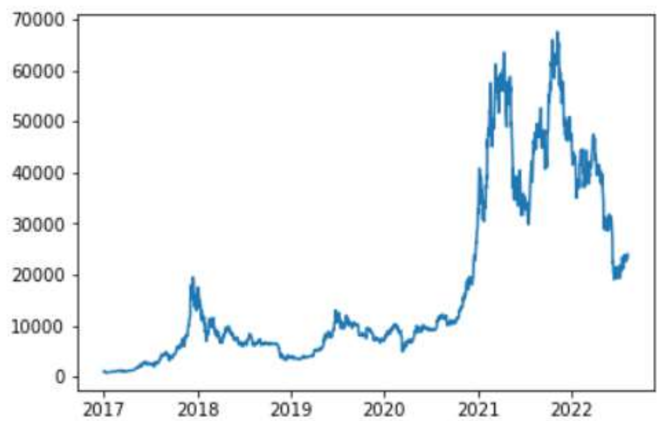
Printing the last rows of data

```
BTC.tail()
```

Date	High	Low	Open	Close	Volume	
2022-08-06	23326.562500	22961.279297	23291.423828	22961.279297	15978259885	229
2022-08-07	23359.009766	22894.556641	22963.505859	23175.890625	15886817043	231
2022-08-08	24203.689453	23176.546875	23179.527344	23809.486328	28575544847	238
2022-08-09	23898.615234	22982.000000	23811.484375	23164.318359	23555719219	231
2022-08-10	24126.781250	22773.726562	23126.421875	24018.003906	28472848384	240

Plotted data

```python
import matplotlib
from matplotlib.pyplot import*
plot(BTC["Close"])
show()
```



Printing data types of data variables

```python
BTC.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2048 entries, 2017-01-01 to 2022-08-10
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2048 non-null   float64
 1   Low        2048 non-null   float64
 2   Open       2048 non-null   float64
 3   Close      2048 non-null   float64
 4   Volume     2048 non-null   int64  
 5   Adj Close  2048 non-null   float64
dtypes: float64(5), int64(1)
memory usage: 112.0 KB
```

Calculating the simple return of Bitcoin

```python
BTC["r"] = np.log(BTC['Adj Close']) - np.log(BTC['Adj Close'].shift(1))
BTCR = BTC[['R','r']].copy()
```

Calculating cc returns:

```python
BTC["r"] = np.log(BTC['Adj Close']) - np.log(BTC['Adj Close'].shift(1))
BTCR = BTC[['R','r']].copy()
```

Describing the statistics return of a column

```python
sumret = BTC["R"].describe()
sumret
```

```
count    2047.000000
mean        0.002418
std         0.041380
min        -0.371695
25%        -0.015893
50%         0.002217
75%         0.020937
max         0.252472
Name: R, dtype: float64
```

Selecting the returns that are less than 15%

```python
BTC[BTC["R"]<-0.15]
```

Date	High	Low	Open	Close	Volume	
2017-09-14	3920.600098	3153.860107	3875.370117	3154.949951	2716310016	31
2018-01-16	13843.099609	10194.900391	13836.099609	11490.500000	18853799936	114
2018-02-05	8364.839844	6756.680176	8270.540039	6955.270020	9285289984	69
2020-03-12	7929.116211	4860.354004	7913.616211	4970.788086	53980357243	49
2022-06-13	26795.589844	22141.257812	26737.578125	22487.388672	68204556440	224

Selecting the best days of the Bitcoin

```python
BTC[BTC["R"]>0.15].sort_values(by=['R'], ascending=False)
```

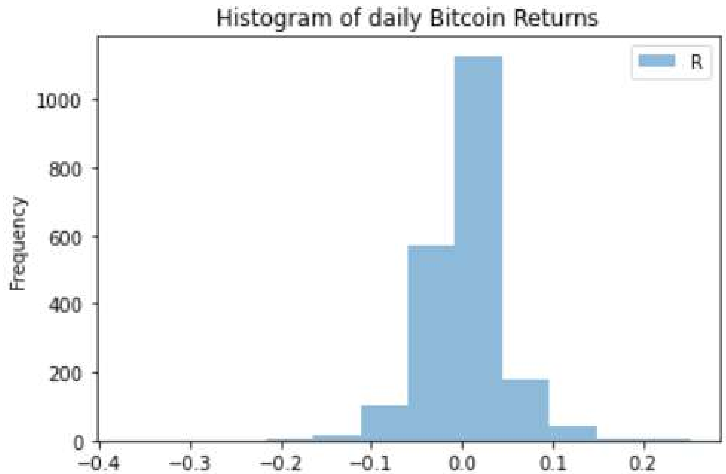
Date	High	Low	Open	Close	Volume	
2017-12-07	17899.699219	14057.299805	14266.099609	17899.699219	17950699520	17
2017-07-20	2900.699951	2269.889893	2269.889893	2817.600098	2249260032	2
2017-12-06	14369.099609	11923.400391	11923.400391	14291.500000	12656300032	14
2021-02-08	46203.929688	38076.324219	38886.828125	46196.464844	101467222687	46
2020-03-19	6329.735840	5236.968750	5245.416504	6191.192871	51000731797	6
2019-04-02	4905.954590	4155.316895	4156.919434	4879.877930	21315047816	4
2019-10-25	8691.540039	7479.984375	7490.703125	8660.700195	28705065488	8
2017-07-17	2230.489990	1932.619995	1932.619995	2228.409912	1201760000	2
2017-09-15	3733.449951	2946.620117	3166.300049	3637.520020	4148069888	3

3.2 Challenge: Histogram

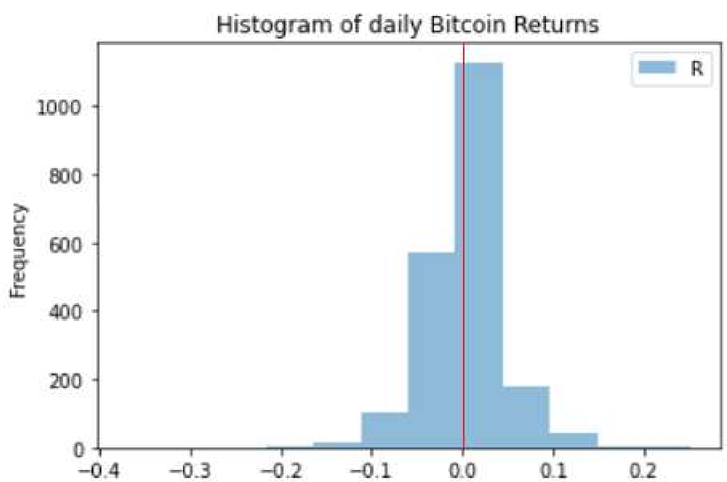
Wednesday, August 10, 2022 3:33 PM

Myroslava Sánchez Andrade A01730712 | 10/08/2022

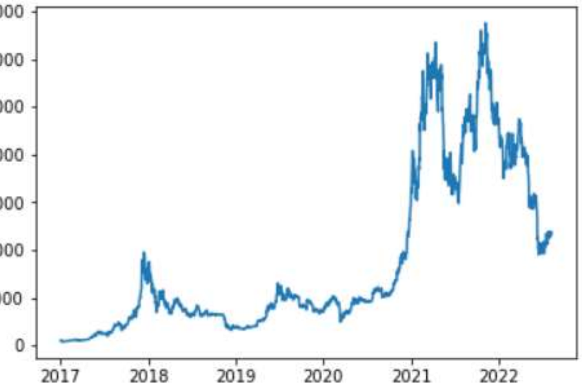
```
r_bitcoin = pd.DataFrame(BTC[["R"]])
hist=r_bitcoin.plot.hist(bins=12,alpha=0.5,title="Histogram of daily Bitco
```



AT FIRST SIGHT, IT MIGHT SEEM THAT THE DAILY FREQUENCY RETURNS OF THE BITCOIN SINCE 2017 UNTIL TODAY (08/10/2022) TENDS IN ITS MAJORITY TO BE EITHER 0 OR NEGATIVE. BUT AFTER DRAWING A VERTICAL LINE ALIGNED TO THE 0, I WAS ABLE TO REALIZE THAT THE FREQUENCY OF RETURNS TENDS TO BE IN ITS MAJORITY POSITIVE.



EVEN THOUGH THERE IS MORE PROBABILITY OF A POSITIVE RETURN, I WOULD NOT SAY THAT THE DIFFERENCE BETWEEN THE PROBABILITY OF A POSITIVE AND A NEGATIVE RETURN IS THAT MUCH. AND THIS POINT ABOVE ACTUALLY MATCHES WITH THE ANALYSIS OF THE PLOTTED GRAPH OF THE RETURNS, WHICH SHOWS AN INCREMENT OF THE RETURNS OVER TIME.



ANOTHER REALLY IMPORTANT POINT TO HIGHLIGHT IS THAT THE HISTOGRAM EXPOSES A NORMAL DISTRIBUTION (BELL-SHAPED).

5.2 Challenge: Simulating the normal distribution

Wednesday, August 10, 2022 8:23 PM

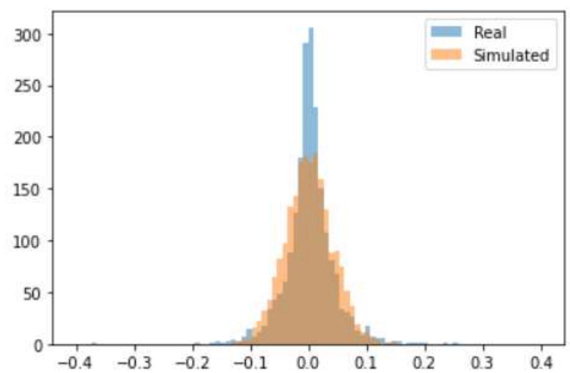
Myroslava Sánchez Andrade A01730712 | 10/08/2022

We know that the column "r" of BTC cotains the historical cc returns of Bitcoin. We use the function random.normal to simulate random returns given the mean, standard deviation and size.

```
gen_values = np.random.normal(BTC["r"].mean(), BTC["r"].std(), BTC["r"].count())
print(gen_values)
```

```
[-0.03228258 -0.04825458  0.02707137 ... -0.00301276  0.02008491
 -0.02381526]
```

```
#Showing the real distribution of historical cc returns and simulated normal distribution
sim_bitcoin = pd.DataFrame(gen_values)
matplotlib.pyplot.hist(x= r_bitcoin, bins=90,alpha=0.5,range=(-0.4, 0.4),label="Real")
matplotlib.pyplot.hist(x=sim_bitcoin,bins=90,alpha=0.5,range=(-0.4, 0.4),label="Simulated")
matplotlib.pyplot.legend(loc='upper right')
matplotlib.pyplot.show()
```



THERE IS A DIFFERENCE IN FREQUENCY WHEN X≈0. THE FREQUENCY USING THE REAL DATA WHEN THE RETURNING≈0 IS GREATER THAN IN THE SIMULATED ONE, BUT WE CAN STILL CLEARLY OBSERVE THAT BOTH DISTRIBUTIONS EXPOSE A SIMILAR NORMAL BEHAVIOUR.