Chapter_01_Descriptive_Statistics

June 4, 2022

1 Descriptive statistics

This chapter will explain the most important statistics to describe a dataset in the financial world. Indeed, these methods are helpful in portfolio management, financial analysis, and trading.

1.0.1 After this Chapter you will be able to:

- Compute and understand how to interprate the mean
- Compute and understand how to interprate the median
- Compute and understand how to interprate mode
- Compute and understand how to interprate the variance
- Compute and understand how to interprate the standard deviation
- Compute and understand how to interprate the covariance
- Compute and understand how to interprate the variance-covariance matrix
- Compute and understand how to interprate the skweness ?
- Compute and understand how to interprate the kurtosis ?

1.0.2 Exercises (Trading / Portfolio management):

- Compute the risk/return of a financial asset
- Compute the correlation between asset properly

Join our community: https://discord.gg/wXjNPAc5BH

Read our book: https://www.amazon.com/gp/product/B09HG18CYL

Quantreo's YouTube channel: https://www.youtube.com/channel/UCp7jckfiEglNf_Gj62VR0pw

[ ]:
```
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.7/dist-
packages (from yfinance) (1.1.5)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-
packages (from yfinance) (1.19.5)
Collecting lxml>=4.5.1
  Downloading lxml-4.6.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.m
anylinux_2_24_x86_64.whl (6.3 MB)
     |                | 6.3 MB 35.8 MB/s
Requirement already satisfied: multitasking>=0.0.7 in
```

```
/usr/local/lib/python3.7/dist-packages (from yfinance) (0.0.10)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.7/dist-
packages (from yfinance) (2.23.0)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.24->yfinance) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-
packages (from pandas>=0.24->yfinance) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.7.3->pandas>=0.24->yfinance) (1.15.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.20->yfinance) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.20->yfinance) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.20->yfinance)
(2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests>=2.20->yfinance) (2.10)
Installing collected packages: lxml, yfinance
  Attempting uninstall: lxml
    Found existing installation: lxml 4.2.6
    Uninstalling lxml-4.2.6:
      Successfully uninstalled lxml-4.2.6
Successfully installed lxml-4.6.4 yfinance-0.1.67
```

[ ]:
```
import numpy as np
import pandas as pd
import yfinance as yf
from scipy.stats import skew, kurtosis
```

[ ]:
```
# Import some data
df = yf.download("GOOG")["Adj Close"].pct_change(1).dropna()
df
```

```
[**********************100%**********************]  1 of 1 completed
```

[ ]:
```
Date
2014-03-28    0.002740
2014-03-31   -0.005393
2014-04-01    0.018295
2014-04-02   -0.000282
2014-04-03    0.004833
                 …
2021-12-06    0.008953
2021-12-07    0.029486
2021-12-08    0.004620
2021-12-09   -0.004132
2021-12-10    0.003842
```

```
Name: Adj Close, Length: 1942, dtype: float64
```

2 Central tendency measure

2.0.1 Mean

[ ]:
```python
# -------- Mean with numpy ------------
mean = np.mean(df, axis=0)*100
print(f"Daily Mean: {'%.2f' % mean} %")

# Annualization of the mean return
annual_mean = mean * 252
print(f"Mean Annual: {'%.2f' % annual_mean} % ")

# day mean return --> monthly mean return
monthly_mean = mean * 21
print(f"Monthly Mean: {'%.2f' % monthly_mean} %")
```

```
Daily Mean: 0.10 %
Mean Annual: 25.10 %
Monthly Mean: 2.09 %
```

2.0.2 Median

[ ]:
```python
# -------- Median with numpy ------------
median = np.median(df, axis=0)*100
print(f"Daily Median: {'%.2f' % median} %")

# Annualization of the mean return
annual_median = median * 252
print(f"Yearly Median: {'%.2f' % annual_median} % ")

# day mean return --> monthly mean return
monthly_median = median * 21
print(f"Monthly Median: {'%.2f' % monthly_median} %")
```

2.0.3 Centile

[ ]:
```python
# -------- Centile with numpy ------------
centile_10 = np.quantile(df, 0.1, axis=0)*100
print(f"Centile 10%: {'%.2f' % centile_10} %")

centile_50 = np.quantile(df, 0.5, axis=0)*100
print(f"Centile 50%: {'%.2f' % centile_50} %")

centile_99 = np.quantile(df, 0.99, axis=0)*100
print(f"Centile 99%: {'%.2f' % centile_99} %")
```

```
Centile 10%: -1.63 %
Centile 50%: 0.10 %
Centile 99%: 4.27 %
```

3 Standard dispersion measurement

3.0.1 Variance

[ ]:
```python
# -------- Variance with numpy ------------
var = np.var(df, axis=0)*100
print(f"Daily Median: {'%.2f' % var} %")

# Annualization of the mean return
annual_var = var * 252
print(f"Median Annual: {'%.2f' % annual_var} % ")

# day mean return --> monthly mean return
monthly_var = var * 21
print(f"Monthly Mean: {'%.2f' % monthly_var} %")
```

```
Daily Median: 0.03 %
Median Annual: 6.75 %
Monthly Mean: 0.56 %
```

3.0.2 Standard deviation

[ ]:
```python
# -------- Stadard-Deviation with numpy ------------
std = np.std(df, axis=0)*100
print(f"Daily Volatility: {'%.2f' % std} %")

# Annualization of the mean return
annual_std = std * np.sqrt(252)
print(f"Annual Volatility: {'%.2f' % annual_std} % ")

# day mean return --> monthly mean return
monthly_std = std * np.sqrt(21)
print(f"Monthly Volatility: {'%.2f' % monthly_std} %")
```

```
Daily Volatility: 1.64 %
Annual Volatility: 25.98 %
Monthly Volatility: 7.50 %
```

3.0.3 Skweness

[ ]:
```python
# -------- Skweness with numpy ------------
skw = skew(df)
print(f"Skweness: {'%.2f' % skw} ")
```

```
Skweness: 0.47
```

3.0.4 Kurtosis

[ ]:
```
# -------- Kurtosis with numpy ------------
kurto = kurtosis(df)
print(f"Kurtosis: {'%.2f' % kurto}")
```

```
Kurtosis: 9.68
```

4 Relationship measurement

4.0.1 Variance Covariance matrix

[ ]:
```
# Import several assets
df = yf.download(["GOOG","EURUSD=X"])["Adj Close"].pct_change(1).dropna()
```

```
[**********************100%**********************]  2 of 2 completed
```

[ ]:
```
# Variance Covariance matrix
mat = np.cov(df, rowvar=False)
mat
```

[ ]:
```
array([[ 2.40234701e-05, -1.13876470e-06],
       [-1.13876470e-06,  2.58833065e-04]])
```

4.0.2 Covariance

[ ]:
```
# Covariance
mat[0][1]
```

[ ]:
```
-1.1387646967552557e-06
```

4.0.3 Correlation

[ ]:
```
# Correlation matrix
df.corr()
```

[ ]:

	EURUSD=X	GOOG
EURUSD=X	1.000000	-0.014441
GOOG	-0.014441	1.000000

5 EXERCISES

5.0.1 Exercise 1: Compute the annualized risk return couple for Microsoft stock price (Yahoo symbol: MSFT). Don't forget to use the variations price

[ ]:
```
# Import the prices
df = yf.download("MSFT")["Adj Close"].pct_change(1).dropna()

# Compute risk return
```

```
mean = np.mean(df) * 252 * 100
vol = np.std(df) * np.sqrt(252) * 100

print(f"MSFT | \t returns: {'%.2f' % mean} % \t volatility: {'%.2f' % vol} %")
```

```
[**********************100%**********************]  1 of 1 completed
MSFT |   returns: 29.85 %        volatility: 33.81 %
```

5.0.2 Exercise 2: Compute the covariance and the correlation matrix for the following assets: [“AMZN”, “MSFT”, “GOOG”, “EURUSD=X”, “BTC-USD”]

[ ]:
```
df = yf.download(["AMZN", "MSFT", "GOOG", "EURUSD=X", "BTC-USD"])["Adj Close"].
↪pct_change(1).dropna()
df.cov()
```

```
[**********************100%**********************]  5 of 5 completed
```

[ ]:

	AMZN	BTC-USD	EURUSD=X	GOOG	MSFT
AMZN	2.480388e-04	0.000042	-2.160787e-07	1.383901e-04	0.000140
BTC-USD	4.191048e-05	0.001521	-1.706122e-06	5.484372e-05	0.000065
EURUSD=X	-2.160787e-07	-0.000002	1.793829e-05	-8.180438e-07	-0.000001
GOOG	1.383901e-04	0.000055	-8.180438e-07	1.888534e-04	0.000138
MSFT	1.400978e-04	0.000065	-1.026322e-06	1.383320e-04	0.000196

[ ]:
```
df.corr()
```

[ ]:

	AMZN	BTC-USD	EURUSD=X	GOOG	MSFT
AMZN	1.000000	0.068243	-0.003239	0.639415	0.635379
BTC-USD	0.068243	1.000000	-0.010330	0.102344	0.119769
EURUSD=X	-0.003239	-0.010330	1.000000	-0.014055	-0.017308
GOOG	0.639415	0.102344	-0.014055	1.000000	0.718987
MSFT	0.635379	0.119769	-0.017308	0.718987	1.000000

[ ]:
```

```