



CURSO

Python aplicado a finanzas



MÓDULO 5

¿Qué es un Portafolio?

MÓDULO 5

¿Qué es un Portafolio?

- Una cartera de inversiones o cartera de valores o portafolio, es una determinada combinación de activos financieros en los cuales se invierte.
- Una cartera de inversiones puede estar compuesta por una combinación de algunos instrumentos de renta fija y renta variable.
- Dependiendo del tipo de inversor que se sea, conservador, de crecimiento, agresivo, la ponderación de los diferentes activos será diferente.
- https://es.wikipedia.org/wiki/Cartera_de_valores

Rendimiento Acumulado

```
[ ] import pandas as pd

pd = pd.DataFrame([100,50, 75, 37.5])
pd.columns= ["Close"]

[ ] pd["rendimiento"] = pd.Close.pct_change()

[ ] pd["rendimiento_manual"] = (pd.Close / pd.Close.shift())-1

[ ] pd["rendimiento_acum"] = ((1 + pd.Close.pct_change()).cumprod()-1) *100
```

```
[ ] pd
```

	Close	rendimiento	rendimiento_manual	rendimiento_acum
0	100.0	NaN	NaN	NaN
1	50.0	-0.5	-0.5	-50.0
2	75.0	0.5	0.5	-25.0
3	37.5	-0.5	-0.5	-62.5

```
[ ] data = yf.download(['AMZN','TSLA','NFLX'], start='2000-01-01')['Adj Close'].dropna()
print(data.head())
print(data.tail())
```

[*****100%*****] 3 of 3 completed

	AMZN	NFLX	TSLA
Date			
2010-06-29	108.610001	16.082857	4.778
2010-06-30	109.260002	15.521429	4.766
2010-07-01	110.959999	15.665714	4.392
2010-07-02	109.139999	15.297143	3.840
2010-07-06	110.059998	15.324286	3.222
	AMZN	NFLX	TSLA
Date			
2021-11-18	3696.060059	682.020020	1096.380005
2021-11-19	3676.570068	678.799988	1137.060059
2021-11-22	3572.570068	659.200012	1156.869995
2021-11-23	3580.040039	654.059998	1109.030029
2021-11-24	3580.409912	658.289978	1116.000000

```
[ ] data
```

	AMZN	NFLX	TSLA
Date			
2010-06-29	108.610001	16.082857	4.778000
2010-06-30	109.260002	15.521429	4.766000
2010-07-01	110.959999	15.665714	4.392000
2010-07-02	109.139999	15.297143	3.840000
2010-07-06	110.059998	15.324286	3.222000
...
2021-11-18	3696.060059	682.020020	1096.380005
2021-11-19	3676.570068	678.799988	1137.060059
2021-11-22	3572.570068	659.200012	1156.869995
2021-11-23	3580.040039	654.059998	1109.030029
2021-11-24	3580.409912	658.289978	1116.000000

2874 rows x 3 columns



Cantidad de Acciones a comprar S&P 500

1) Obtener los tickers del S&P 500

```
[1] import pandas as pd

tables = pd.read_html("https://en.wikipedia.org/wiki/List_of_S%26P_500_companies")

tickers = tables[0]["Symbol"]

tickers
```

0	MMM
1	ABT
2	ABBV
3	ABMD
4	ACN
	...
500	YUM
501	ZBRA
502	ZBH
503	ZION
504	ZTS

Name: Symbol, Length: 505, dtype: object

Cantidad de Acciones a comprar S&P 500

2) Obtenemos datos del mercado

```
[ ]  
import requests  
  
symbol = "MMM"  
  
SANDBOX_API_TOKEN = "Tpk_72c8bb43275041e3b90a922f7512171e"  
  
api_url = f"https://sandbox.iexapis.com/stable/stock/{symbol}/quote?token={SANDBOX_API_TOKEN}"  
  
data = requests.get(api_url).json()  
data  
  
[ ] stocks_columns = ["Ticker", "Price", "MarketCap", "SharesToBuy"]  
  
stocks_dataframe = pandas.DataFrame(columns = stocks_columns)  
  
for symbol in tickers:  
    endpoint = f"https://sandbox.iexapis.com/stable/stock/{symbol}/quote?token={IEX_CLOUD_API_TOKEN}"  
    data = requests.get(endpoint).json()  
    stocks_dataframe = stocks_dataframe.append(pandas.Series([symbol,  
                                                                data["latestPrice"],  
                                                                data["marketCap"],  
                                                                "N/A"],  
                                                                index = stocks_columns),  
                                                ignore_index = True)  
  
stocks_dataframe
```

Cantidad de Acciones a comprar S&P 500

3) Tamaño de portafolio y posición para cada archivo

```
[ ] portfolio_value = 10000000
```

```
[ ] position_size = portfolio_value / len(stocks_dataframe)
```

```
[ ] position_size
```

```
19801.980198019803
```

Cantidad de Acciones a comprar S&P 500

4) Cálculo de cantidad de activos a comprar con el monto disponible

```
[ ] stocks_dataframe.SharesToBuy = position_size // stocks_dataframe.Price
```

```
[ ] stocks_dataframe
```

	Ticker	Price	MarketCap	SharesToBuy
0	MMM	185.43	106941700053	106.0
1	ABT	131.03	222499836424	151.0
2	ABBV	119.49	215791225494	165.0
3	ABMD	338.12	15772456423	58.0
4	ACN	378.37	247952903317	52.0
...
500	YUM	128.20	38010837427	154.0
501	ZBRA	619.63	31996772132	31.0
502	ZBH	135.43	27739763492	146.0
503	ZION	70.81	10862996665	279.0
504	ZTS	230.90	106761343425	85.0

505 rows × 4 columns


Cantidad de Acciones a comprar Panel Lider

```
data = data[0]
```

```
data["Símbolo"] = data.Símbolo.str.split(" ")
```

```
data["Símbolo"] = data["Símbolo"].apply(lambda x: x[0])
```

```
#data["Símbolo"]  
data["ÚltimoOperado"]
```



0	95.40
1	226.20
2	311.00
3	905.00
4	72.30
5	5.69
6	103.40
7	559.50
8	62.40
9	204.70
10	176.00
11	278.50
12	3911.50
13	150.85
14	85.80
15	214.40
16	88.20
17	207.55
18	58.50
19	125.50
20	24.30
21	823.10

Name: ÚltimoOperado, dtype: float64

Cantidad de Acciones a comprar Panel Lider

```
stocks_columns = ["Ticker", "Price", "SharesToBuy"]

stocks_dataframe = pd.DataFrame(columns = stocks_columns)

for i in range(len(data)):

    stocks_dataframe = stocks_dataframe.append(pd.Series([data.iloc[i,0],
                                                            data.iloc[i,1],
                                                            "N/A"],
                                                            index = stocks_columns),
                                                ignore_index = True)

stocks_dataframe
```

	Ticker	Price	SharesToBuy
0	ALUA	95.40	N/A
1	BBAR	226.20	N/A
2	BMA	311.00	N/A
3	BYMA	905.00	N/A
4	CEPU	72.30	N/A
5	COME	5.69	N/A
6	CRES	103.40	N/A
7	CVH	559.50	N/A
8	EDN	62.40	N/A
9	GGAL	204.70	N/A
10	HARG	176.00	N/A
11	LOMA	278.50	N/A
12	MIRG	3911.50	N/A
13	PAMP	150.85	N/A
14	SUPV	85.80	N/A
15	TECO2	214.40	N/A
16	TGNO4	88.20	N/A
17	TGSU2	207.55	N/A
18	TRAN	58.50	N/A
19	TXAR	125.50	N/A
20	VALO	24.30	N/A
21	YPFD	823.10	N/A

Cantidad de Acciones a comprar Panel Lider

```
portfolio_value = 10000000
```

```
position_size = portfolio_value / len(stocks_dataframe)
```

```
position_size  
stocks_dataframe.SharesToBuy = position_size // stocks_dataframe.Price
```

```
stocks_dataframe
```

	Ticker	Price	SharesToBuy
0	ALUA	95.40	4764.0
1	BBAR	226.20	2009.0
2	BMA	311.00	1461.0
3	BYMA	905.00	502.0
4	CEPU	72.30	6286.0
5	COME	5.69	79884.0
6	CRES	103.40	4395.0
7	CVH	559.50	812.0
8	EDN	62.40	7284.0
9	GGAL	204.70	2220.0
10	HARG	176.00	2582.0
11	LOMA	278.50	1632.0
12	MIRG	3911.50	116.0
13	PAMP	150.85	3013.0
14	SUPV	85.80	5297.0
15	TECO2	214.40	2120.0
16	TGNO4	88.20	5153.0
17	TGSU2	207.55	2190.0
18	TRAN	58.50	7770.0
19	TXAR	125.50	3621.0
20	VALO	24.30	18705.0
21	YPFD	823.10	552.0

Correlación y Matriz de Correlación (Pearson)

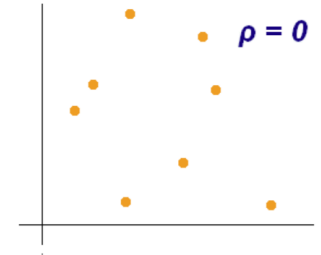
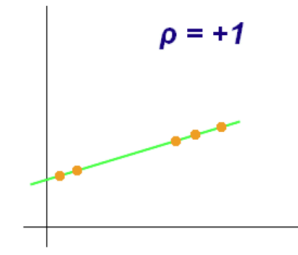
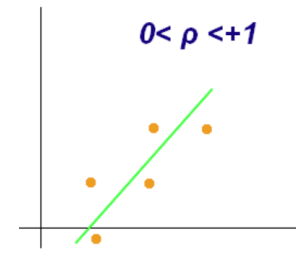
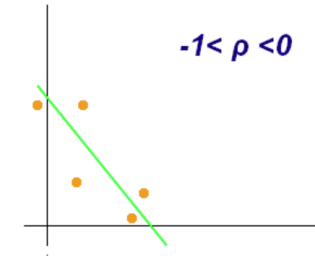
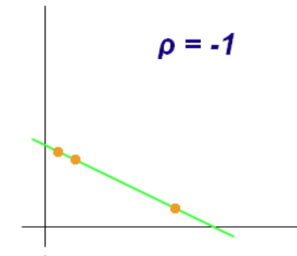
- En estadística, el coeficiente de correlación de Pearson es una medida de dependencia lineal entre dos variables aleatorias cuantitativas.
- A diferencia de la covarianza, la correlación de Pearson es independiente de la escala de medida de las variables.
- De manera menos formal, podemos definir el coeficiente de correlación de Pearson como un índice que puede utilizarse para medir el grado de relación de dos variables siempre y cuando ambas sean cuantitativas y continuas.



En Pandas .corr()



- Calcula la correlación de columnas por pares, excluyendo NA / valores nulos
- **Parámetros:** Método Pearson, Kendall, Spearman
- **Pearson:** Coeficiente de correlación estándar.
- **Kendall:** Coeficiente de correlación de Kendall Tau.
- **Spearman:** Correlación de rango de Spearman.



```
#tickers = ['BBAR','BMA','GGAL','SUPV']
tickers = ["AAPL","X","BABA","TSLA","GOLD"]
```

```
df = yf.download(tickers, auto_adjust=True)["Close"]
```

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

```
df = df.pct_change() * 100
```

```
df
```

	AAPL	BABA	GOLD	TSLA	X
Date					
1980-12-12	NaN	NaN	NaN	NaN	NaN
1980-12-15	-5.217059	NaN	NaN	NaN	NaN
1980-12-16	-7.339781	NaN	NaN	NaN	NaN
1980-12-17	2.475081	NaN	NaN	NaN	NaN
1980-12-18	2.899232	NaN	NaN	NaN	NaN
...
2021-11-18	2.853599	-11.127612	-2.048596	0.676761	-0.857139
2021-11-19	1.697604	-2.270202	-1.507780	3.710397	0.946890
2021-11-22	0.292744	-2.650706	-2.271600	1.742207	4.282215
2021-11-23	0.242206	-2.166587	-1.819104	-4.135293	0.156437
2021-11-24	0.328356	2.139758	-0.205872	0.628474	-1.874273

10327 rows × 5 columns



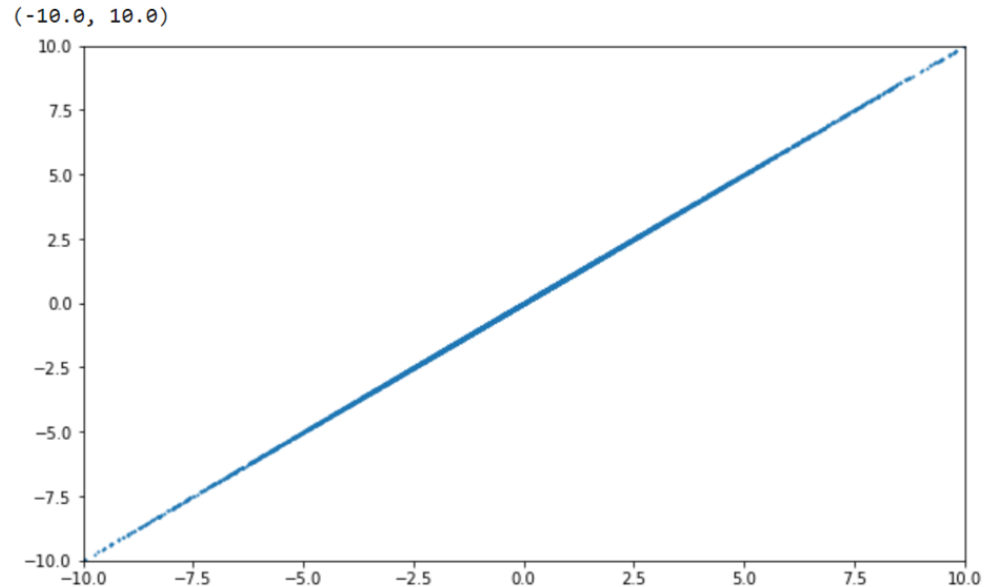
```
df.corr()
```

	BBAR	BMA	GGAL	SUPV
BBAR	1.000000	0.705642	0.541859	0.807860
BMA	0.705642	1.000000	0.754066	0.803181
GGAL	0.541859	0.754066	1.000000	0.836293
SUPV	0.807860	0.803181	0.836293	1.000000

Gráficas de Correlación

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10,6))
#ax.scatter(df.GGAL, df.GGAL, s=1)
#ax.scatter(df.BBAR, df.BBAR, s=1)
ax.scatter(df.X, df.X, s=1)
ax.set_xlim(-10,10)
ax.set_ylim(-10,10)
```



```
self.file = None
self.fingerprints = set()
self.logdups = True
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = open(os.path.join(path, 'log.txt'), 'a')
    self.file.seek(0)
    self.fingerprints.update(fingerprints)

@classmethod
def from_settings(cls, settings):
    debug = settings.getboolean('debug')
    return cls(job_dir=settings.get('job_dir'))

def request_seen(self, request):
    fp = self.request_fingerprint(request)
    if fp in self.fingerprints:
        return True
    else:
        return False
```

Sharpe de Cartera

- El **ratio Sharpe**, nombrado así por su creador William Sharpe, es una métrica que ayuda a los inversores a medir la eficiencia de una inversión teniendo en cuenta tanto la rentabilidad como el riesgo asumido.
- Se calcula como: **Ratio Sharpe = (Rentabilidad del fondo o de la cartera – Rentabilidad del activo libre de riesgo) / Volatilidad (desviación típica) del fondo o de la cartera.**
- Mide la rentabilidad adicional por encima de la rentabilidad del activo libre de riesgo por unidad de volatilidad asumida.
- Cuanto mayor sea el ratio sharpe de una inversión, mejor será el rendimiento ajustado al riesgo de la inversión.
- En el caso de ser negativo, la rentabilidad de la inversión no habrá superado al activo libre de riesgo.
- En la práctica, el **ratio Sharpe** es utilizado como herramienta de análisis con la que comparar productos similares, facilitando la elección óptima entre distintas inversiones.



Sharpe de Cartera

```
import yfinance as yf
import numpy as np
import pandas as pd
import random, requests
```

```
data = yf.download(['AMZN', 'TSLA', 'NFLX'], start='2000-01-01', end='2021-01-01')['Adj Close']
```

data

[*****100%*****] 3 of 3 completed

	AMZN	NFLX	TSLA
Date			
2000-01-03	89.375000	NaN	NaN
2000-01-04	81.937500	NaN	NaN
2000-01-05	69.750000	NaN	NaN
2000-01-06	65.562500	NaN	NaN
2000-01-07	69.562500	NaN	NaN
...
2020-12-24	3172.689941	513.969971	661.770020
2020-12-28	3283.959961	519.119995	663.690002
2020-12-29	3322.000000	530.869995	665.989990
2020-12-30	3285.850098	524.590027	694.780029
2020-12-31	3256.929932	540.729980	705.669983

5284 rows × 3 columns

```
retornos = ((1 + data.pct_change()).cumprod()-1) *100
retornos.plot(legend=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fbb59824090>



```
retornos = np.log((data/data.shift(1)).dropna())
retornos.head()
```

	AMZN	NFLX	TSLA
Date			
2010-06-30	0.005967	-0.035532	-0.002515
2010-07-01	0.015439	0.009253	-0.081723
2010-07-02	-0.016538	-0.023808	-0.134312
2010-07-06	0.008394	0.001773	-0.175470
2010-07-07	0.030160	0.099480	-0.019430

```
pond = np.random.random(len(data.columns))
pond = pond/np.sum(pond)
pond
```

array([0.83752418, 0.04206442, 0.12041139])

```
np.random.random(3)
```

array([0.02820778, 0.74807252, 0.70654583])

Sharpe de Cartera

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_{ij}$$

```
r={}
r['retorno'] = np.sum( (retornos.mean() * pond * 252))
r['volatilidad'] = np.sqrt(np.dot(pond, np.dot(retornos.cov()*252, pond)))
r['sharpe'] = r['retorno'] / r['volatilidad']
r['ponderaciones'] = pond
r
```

```
{'ponderaciones': array([0.83752418, 0.04206442, 0.12041139]),
 'retorno': 0.3426253611237615,
 'sharpe': 1.1355211463924377,
 'volatilidad': 0.3017340207289717}
```

Markowitz

```
pip install yfinance
```

```
import yfinance as yf, numpy as np, pandas as pd, matplotlib.pyplot as plt

def markowitz(data, q=1000):
    retornos = np.log((data/data.shift(1)).dropna())
    datos , pesos, datosTickers = [] , [] , []

    for i in range(q):
        pond = np.array(np.random.random(len(data.columns)))
        pond = pond/np.sum(pond)
        pesos.append(pond)
        r={}
        r['retorno'] = np.sum( (retornos.mean() * pond * 252))
        r['volatilidad'] = np.sqrt(np.dot(pond, np.dot(retornos.cov()*252, pond)))
        r['sharpe'] = r['retorno'] / r['volatilidad']
        datos.append(r)

    for ticker in data.columns:
        d = {}
        d['ticker'] = ticker
        d['retorno'] = retornos[ticker].mean() * 252
        d['volatilidad'] = retornos[ticker].std() * (252**0.5)
        d['sharpe'] = d['retorno'] / d['volatilidad']
        datosTickers.append(d)
```



Markowitz

```
datosTickers = pd.DataFrame(datosTickers).set_index('ticker')
datos = pd.DataFrame(datos)
optimo = datos.loc[datos.sharpe.idxmax()]
mejor_port = pesos[datos.sharpe.idxmax()]
datosTickers['ponderacion_optima'] = mejor_port

plt.figure(figsize=(14,11))
plt.scatter(datos.volatilidad, datos.retorno, c=datos.sharpe, s=2, cmap='magma')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatilidad')
plt.ylabel('Retorno')
plt.scatter(optimo.volatilidad, optimo.retorno, c='tab:green', s=2000)
plt.text(optimo.volatilidad, optimo.retorno, 'Optimo', fontsize=12, c='w', ha='center', va='center')

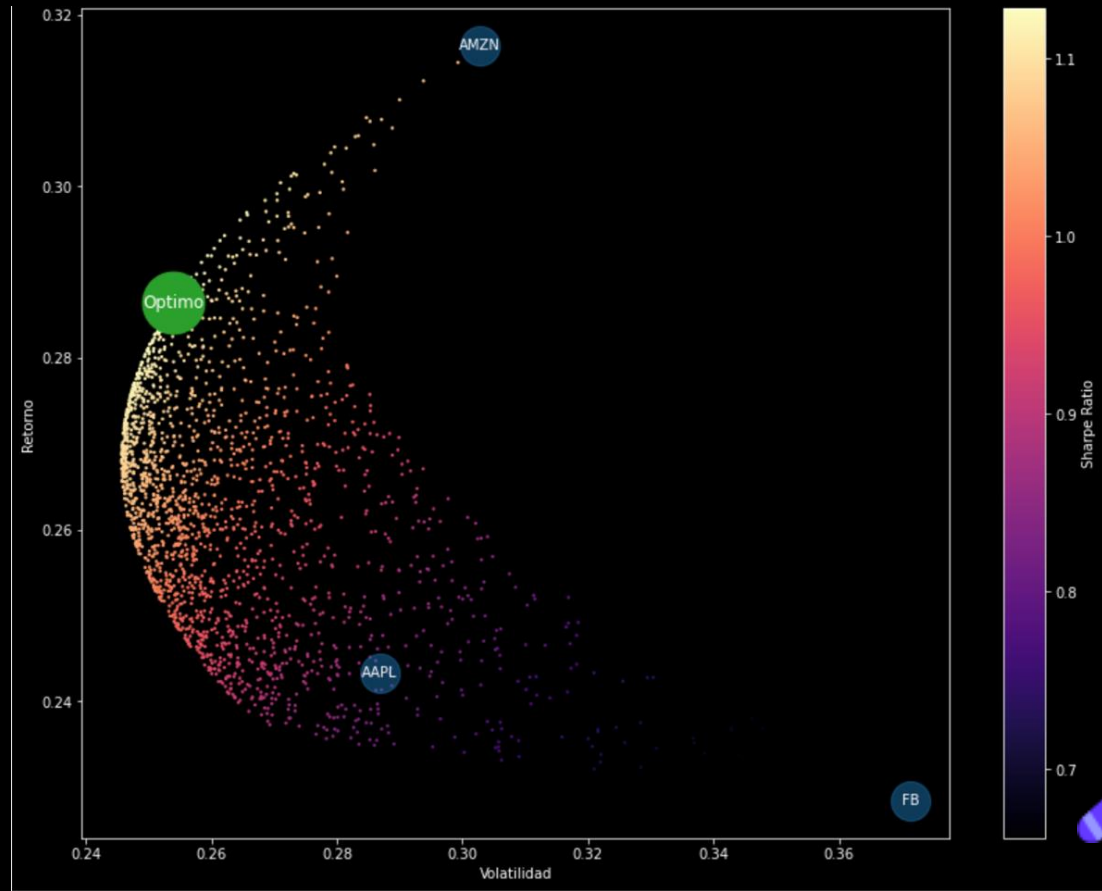
for ticker in data.columns:
    vol = datosTickers.loc[ticker, 'volatilidad']
    ret = datosTickers.loc[ticker, 'retorno']
    plt.scatter(vol, ret, c='tab:blue', alpha=0.5, s=800)
    plt.text(vol, ret, ticker, c='w', ha='center', va='center')

return(datosTickers.round(3), optimo)

plt.style.use('dark_background')
df = yf.download(['AAPL', 'AMZN', 'FB'], start='2000-01-01', end='2021-01-01')['Adj Close']
df = df.loc[~(df==0).any(axis=1)]
pond, optimo = markowitz(df, q=2000)
print(pond, '\n\nPortafolio Optimo:\n', optimo, sep='')
```



Markowitz



```
[*****100%*****] 3 of 3 completed
```

ticker	retorno	volatilidad	sharpe	ponderacion_optima
AAPL	0.243	0.287	0.849	0.367
AMZN	0.316	0.303	1.045	0.598
FB	0.228	0.371	0.615	0.036

Portafolio Optimo:
retorno 0.286479
volatilidad 0.253869
sharpe 1.128453
Name: 968, dtype: float64

