

# Otimização de carteiras

April 12, 2021

Nesse exercício iremos, primeiramente, analisar a correlação das ações do índice bovespa exatamente com o índice. O intuito é criar uma carteira usando as 5 ações com maior correlação com o Ibov. Analisaremos o portfólio criado com pesos iguais para cada ação (20%) e depois usaremos a fronteira eficiente de Markowitz para equilibrar os pesos de cada ação de forma a aumentar a eficiência com volatilidade mínima além de também analisar a carteira com risco ideal

## 1 Importando os pacotes

```
[1]: import yfinance as yf
import pandas as pd
pd.set_option('display.max_rows', 500)
import math
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline
import seaborn as sbs
import scipy.stats as stats
import config
import os
from datetime import datetime
import numpy as np
```

## 2 Baixando os dados históricos das ações

```
[2]: ibovurl= "http://bvmf.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?
↳Indice=ibov&idioma=pt-br"
ibov = pd.read_html(ibovurl)

# Selecionando apenas a coluna com os Tickers

tickers = ibov[0][0:]['Código'].tolist()
tickers.remove('Quantidade Teórica Total Redutor')

# Adicionando .SA em todas as ações

for i in range(0, len(tickers)):
```

```

tickers[i] = tickers[i] + '.SA'

# Buscando no yahoo finance o preço de fechamento ajustado.

dataset = yf.download(tickers=tickers, start='2015-01-01')['Adj Close']

[*****100%*****] 82 of 82 completed

```

### 3 Removendo os NA's

Para melhorar a nossa análise, vamos remover as ações que possuem mais de 25% de dados como "NA's"

```

[3]: missing_fractions = dataset.isnull().mean().sort_values(ascending=False)

missing_fractions.head(10)

drop_list = sorted(list(missing_fractions[missing_fractions > 0.25].index))

dataset.drop(labels=drop_list, axis=1, inplace=True)
dataset.shape

```

```

[3]: (1558, 73)

```

### 4 Retornos diários e acumulados

```

[4]: retorno = dataset.pct_change()
retorno_acumulado = (1 + retorno).cumprod()
retorno_acumulado.iloc[0] = 1
retorno_acumulado = pd.DataFrame(retorno_acumulado)

```

### 5 Baixando dados históricos do IBOV

Para comparação, vamos usar o ETF BOVA11.

```

[5]: ibov = yf.download('BOVA11.SA', start = '2015-01-01')['Adj Close']
IBOV = ibov / ibov.iloc[0]

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

```

## 6 Adicionando a série do IBOV ao dataframe de retornos acumulados

```
[6]: retorno_acumulado = retorno_acumulado.assign(IBOV = IBOV)
```

## 7 Correlação das ações do IBOV com o IBOV

```
[7]: correlacao = retorno_acumulado.corr()
correlacao = pd.DataFrame(correlacao)
corr_stocks_ibov = correlacao['IBOV'].sort_values(ascending = False)
corr_stocks_ibov = pd.DataFrame(corr_stocks_ibov)
corr_stocks_ibov[1:11]
```

```
[7]:
```

	IBOV
PETR4.SA	0.960602
ENGI11.SA	0.959459
LREN3.SA	0.951459
ITSA4.SA	0.950491
EQTL3.SA	0.939640
LCAM3.SA	0.937216
MRVE3.SA	0.934602
SANB11.SA	0.933713
RAIL3.SA	0.932772
ENBR3.SA	0.932207

## 8 Analisando o portfólio com pesos iguais (20%)

### 8.1 Ações selecionados

```
[8]: assets = ['PETR4.SA', 'ENGI11.SA', 'LREN3.SA', 'ITSA4.SA', 'EQTL3.SA']
pesos = np.array([0.2,0.2,0.2,0.2,0.2])
```

```
[9]: dataset = yf.download(tickers=assets, start='2015-04-02')['Adj Close']
dataset = pd.DataFrame(dataset)
```

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

```
[10]: # Retornos diários

returns = dataset.pct_change()

# Retornos anuais
ind_er = dataset.resample('Y').last().pct_change().mean()
```

```
[11]: # matrix de covariancia anualizada

cov_matrix_annual = returns.cov()*252
```

```
[12]: # Calculando a Variância do Portfólio
```

```
port_variance = np.dot(pesos.T, np.dot(cov_matrix_annual, pesos))  
print('A variância do portfólio é de', port_variance)
```

A variância do portfólio é de 0.08102249422691957

```
[13]: # Calculando a volatilidade do Portfólio
```

```
port_volatility = np.sqrt(port_variance)  
print('A volatilidade do portfólio é de', port_volatility)
```

A volatilidade do portfólio é de 0.2846445050003944

```
[14]: # Calculando o retorno anual do portfólio
```

```
portfolio_annual_return = np.sum(returns.mean() * pesos) * 252  
print('O retorno anual desse portfólio é de', portfolio_annual_return)
```

O retorno anual desse portfólio é de 0.25828609259447577

```
[15]: # Calculando o retorno anual esperando, volatilidade e a variância
```

```
percent_var = str(round(port_variance, 2) * 100) + '%'  
percent_vols = str(round(port_volatility, 2) * 100) + '%'  
percent_ret = str(round(portfolio_annual_return, 2) * 100) + '%'  
  
print('Retorno anual esperado: ' + percent_ret)  
print('Volatilidade anual/ risco: ' + percent_vols)  
print('Variância anual: ' + percent_var)
```

Retorno anual esperado: 26.0%

Volatilidade anual/ risco: 28.000000000000004%

Variância anual: 8.0%

## 8.2 Utilizando Markowitz

```
[16]: assets = ['PETR4.SA', 'ENGI11.SA', 'LREN3.SA', 'ITSA4.SA', 'EQTL3.SA']
```

```
df = yf.download(tickers=assets, start='2015-04-02')['Adj Close']  
df = pd.DataFrame(df)
```

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

```
[17]: p_ret = []  
p_vol = []  
p_weights = []
```

```
num_assets = len(df.columns)  
num_portfolios = 50000
```

```
cov_matrix = df.pct_change().apply(lambda x: np.log(1+x)).cov()
```

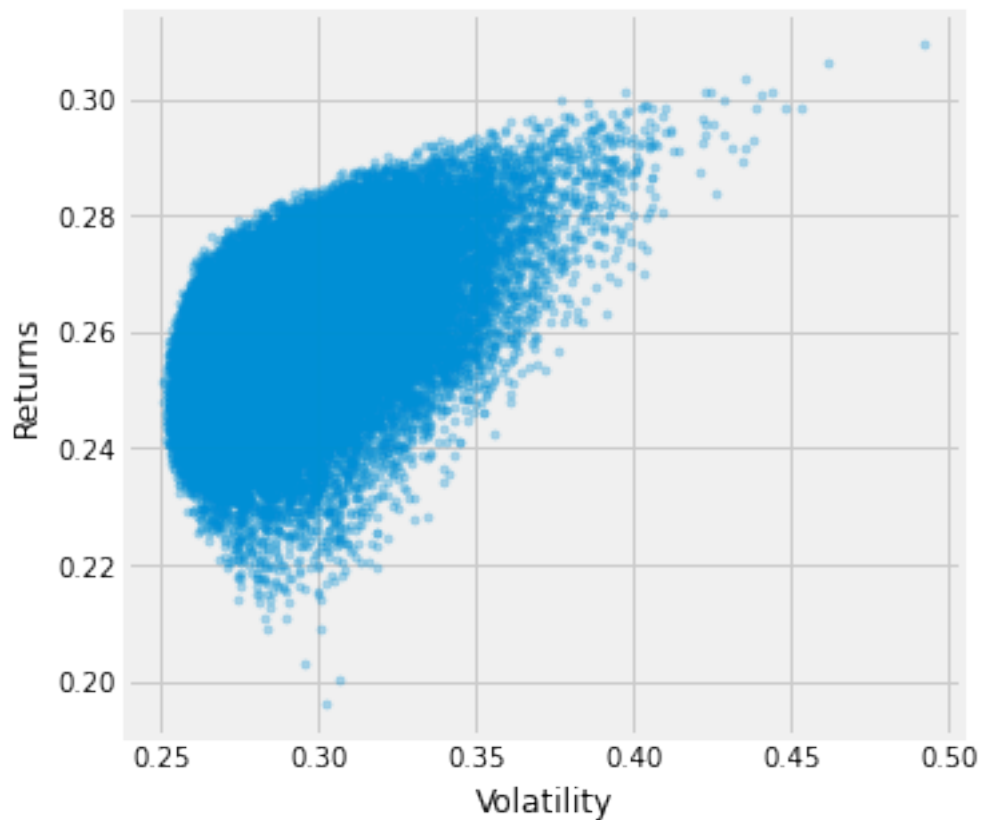
```
[18]: for portfolio in range(num_portfolios):  
    weights = np.random.random(num_assets)  
    weights = weights/np.sum(weights)  
    p_weights.append(weights)  
    returns = np.dot(weights, ind_er)  
    p_ret.append(returns)  
    var = cov_matrix.mul(weights, axis=0).mul(weights, axis=1).sum().sum()  
    sd = np.sqrt(var)  
    ann_sd = sd*np.sqrt(252)  
    p_vol.append(ann_sd)
```

```
[19]: data = {'Returns': p_ret, 'Volatility': p_vol}  
  
for counter, symbol in enumerate(df.columns.tolist()):  
    data[symbol+ 'weights'] = [w[counter] for w in p_weights]
```

```
[20]: portfolios = pd.DataFrame(data) # Dataframe dos 50 mil portfólios criados
```

```
[21]: # Plot efficient frontier  
portfolios.plot.scatter(x='Volatility', y='Returns', marker='o', s=10, alpha=0.  
    ↪3, grid=True, figsize=[5,5])
```

```
[21]: <AxesSubplot:xlabel='Volatility', ylabel='Returns'>
```



### 8.3 Pesos do portfólio mais eficiente com menor volatilidade

```
[22]: min_vol_port = portfolios.iloc[portfolios['Volatility'].idxmin()]
      # idxmin() gives us the minimum value in the column specified.
      ↪
      min_vol_port = pd.DataFrame(min_vol_port)
      min_vol_port
```

```
[22]:          9082
Returns      0.247776
Volatility    0.250985
ENGI11.SAweights  0.316274
EQTL3.SAweights  0.404900
ITSA4.SAweights  0.255545
LREN3.SAweights  0.021834
PETR4.SAweights  0.001447
```

## 8.4 Pesos do portfólio mais eficiente com “Risco Ideal”

```
[23]: # Basta calcular o Sharpe Ratio

rf = 0.0275 # risk factor
optimal_risky_port = portfolios.iloc[((portfolios['Returns']-rf)/
    ↳portfolios['Volatility']).idxmax()]
optimal_risky_port = pd.DataFrame(optimal_risky_port)
optimal_risky_port
```

```
[23]:
```

	28974
Returns	0.272244
Volatility	0.260987
ENGI11.SAweights	0.253444
EQTL3.SAweights	0.626193
ITSA4.SAweights	0.014353
LREN3.SAweights	0.044566
PETR4.SAweights	0.061444