ENGIN 604 Introducción a Python para las Finanzas
Aplicación: Capital Asset Pricing Model - Pauta

Profesor: *Gabriel E. Cabrera*
Ayudante: *Alex Den Braber*

Desarrollado por William Sharpe (1964), el CAPM (*Capital Asset Pricing Model*) parte de la idea que toda inversión contiene dos tipos de riesgo: sistemático (riesgo de mercados que no se pueden diversificar) y no sistemático (componente del retorno de la acción que no correlaciona con los movimientos del mercado). la diversificación de portafolios puede remover el riesgo no sistemático, sin embargo, no es el caso para el riesgo sistemático.

El CAPM es una forma de medir el riesgo sistemático. William Sharpe encuentra que el retorno de una acción o portafolio es:

$$R_{i,t} = R_f + \beta_i(R_{m,t} - R_f)$$

Donde:

- $R_{i,t}$: Retorno esperado del portafolio o acción.
- R_f: Tasa libre de riesgo.
- $R_{m,t}$: Retorno esperado del mercado.
- $R_{m,t} - R_f$: *Equity market premium*
- β_i: El beta de CAPM.

De acuerdo al CAPM, el β es la única medida relevante de riesgo, definido como:

$$\beta_i = \frac{cov(R_{p,t}, R_{m,t})}{\sigma_m^2}$$

Si se considera el β como el segundo coeficiente de una regresión lineal simple por MCO:

$$R_{i,t} - R_f = \alpha_i + \beta_i(R_{m,t} - R_f) + \varepsilon_{i,t}$$

Si bien el modelo anterior no es el CAPM, la regresión permite estimar el β de CAPM. De hecho si $\alpha_i = 0$ y $Cov(R_{m,t}, \varepsilon_{i,t}) = 0$ entonces:

$$\begin{aligned} R_{i,t} - R_f &= \beta_i(R_{m,t} - R_f) \\ R_{i,t} &= R_f + \beta_i(R_{m,t} - R_f) \end{aligned}$$

Se obtiene la ecuación de CAPM.

1. Aplicación

1. Importe los archivos `stocks.pkl`, `sp500.pkl` y `rfree.pkl`.
   - `stocks.pkl`: índices de Facebook, Apple, Amazon, Netflix y Google desde 2014:12 hasta 2020:12.
   - `sp500.pkl`: Precio al cierre del índice del S&P 500 desde 2014:12 hasta 2020:12.
   - `rfree.pkl`: Tasa libre de riesgo desde 2015:01 hasta 2020:12.

```
import numpy as np
import pandas as pd

# carga índices
stocks = pd.read_pickle('stocks.pkl')

# carga s&p 500
sp500 = pd.read_pickle('sp500.pkl')

# tasa libre de riesgo
Rf = pd.read_pickle('rfree.pkl')
```

2. Genere una función que permita calcular el retorno aritmético, definido como:

$$R_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}} \quad i = 1, .., 5 \quad t = 1, .., 73$$

```
# se crea la función
def returns(x):

    '''
    Parameters
    ----------
    x : Array de dimensión tx1

    Returns
    -------
    Retorno aritmético de dimensión (t-1)x1
    '''

    arithmetic_return = np.diff(x, axis = 0) / x[:-1]
    return(arithmetic_return)
```

3. Aplique la función creada en (2) a cada elemento (precios al cierre) almacenado en el diccionario.

```
# diccionario vacio
dict_return = {}

for ticker, close in stocks.items():
    print('Calculando retorno de ticker: ', ticker)
    dict_return[ticker] = returns(close)
```

```
## Calculando retorno de ticker:  AAPL
## Calculando retorno de ticker:  AMZN
## Calculando retorno de ticker:  FB
## Calculando retorno de ticker:  GOOG
## Calculando retorno de ticker:  NFLX
```

4. Aplique la función creada en (2) al índice del S&P 500. Luego reste la tasa libre de riesgo.

$$R_{m,t} = R_{sp500} - R_f$$

```
# retorno de mercado
Rm = returns(sp500)

# risk premium
Rm_Rf = Rm - Rf
```

5. Genere una función que permita ponderar cada retorno por un peso fijo. Los pesos deben sumar 1 y el *output* debe tener restada la tasa libre de riesgo R_f.

$$R_{p,t} = \mathbf{R} \cdot \mathbf{W} = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1i} \\ r_{21} & r_{22} & \dots & r_{2i} \\ \vdots & \vdots & \vdots & \vdots \\ r_{t1} & r_{t2} & \dots & r_{ti} \end{pmatrix}_{t \times i} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_i \end{pmatrix}_{i \times 1}$$

Pruebe su función con los siguientes pesos:

Cuadro 1: Indice & pesos

Índice	Pesos
AAPL	0.25
AMZN	0.25
FB	0.20
GOOG	0.20
NFLX	0.10

```
# pesos (suman 1)
w = np.array([0.25, 0.25, 0.20, 0.20, 0.10]).reshape((-1,1))

# se crea la función
def portfolio(x, w, rf):

    '''
    Parameters
    ----------
    x : Diccionario que contiene i arrays de dimensión t1
    w : Pesos de dimensión i1
    rf: Tasa libre de riesgo de dimensión t1

    Returns
    -------
    Portafolio ponderado de dimensión t1
    '''

    ret_list = []

    for name, r_i in x.items():
        ret_list.append(r_i)

    ret_mat = np.concatenate(ret_list, axis=1)
    rp = ret_mat.dot(w) - rf

    return(rp)

# exceso de retorno
Rp_Rf = portfolio(dict_return, w, Rf)
```

6. Genere una función que permita obtener los $\hat{\beta}$ de una regresión lineal mediante su forma matricial:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'y)$$

```
# se crea la función
def ols(x,y):

    '''
    Parameters
    ----------
    x : matriz ti (Rm - Rf)
    y : matriz ti (Rp - Rf)

    Returns
    -------
    Alfa y Beta
    '''

    m,n = x.shape

    x_with_ones = np.hstack((np.ones((m,1)),x))

    x_t_x = x_with_ones.T.dot(x_with_ones)
    inv_x_t_x = np.linalg.inv(x_t_x)

    betas = inv_x_t_x.dot(x_with_ones.T.dot(y))

    return(betas)

# parámetros
ols(Rm_Rf,Rp_Rf)
```

```
## array([[0.0172864 ],
##        [1.12976075]])
```

7. Muestre que el resultado anterior es igual a:

$$\hat{\beta} = \frac{cov(R_{p,t}, R_{m,t})}{\sigma_m^2}$$

```
# beta portfolio
cov_var_mat = np.cov(np.concatenate([Rp_Rf,Rm_Rf], axis = 1), rowvar=False)

print('El Beta de CAPM es: ', cov_var_mat[0,1] / cov_var_mat[1,1])
```

```
## El Beta de CAPM es:  1.1297607468830584
```

8. Verifique el resultado obtenido en (6) utilizando la librería `statsmodels`.

```
import statsmodels.api as sm

# añade columna de 1
Rm_Rf_with_const = sm.add_constant(Rm_Rf)

# estimación por OLS
mod = sm.OLS(Rp_Rf,Rm_Rf_with_const)
# ajuste del modelo
```

```
res = mod.fit()

# resultado
print(res.summary(res))
```

```
res.params # parámetros
```

```
## array([0.0172864 , 1.12976075])
```