Optimización de portafolios

Esthefany Ortiz Gutierrez 148571
Allan Eduardo Rosas García 160630
Antonio Manguart Páez 175701

Mayo 2019

1 Introducción

El objetivo de este trabajo es la implementación de la solución al modelo de selección de portafolios de Markowitz y revisar la teoría que origina el problema. Para la solución principal se ha usado la librería CVXOPT [1] de python que sirve para resolver problemas de optimización convexa pero también se presenta una solución simple al resolver el sistema de ecuaciones por condiciones de optimalidad y otra por por simulación y con un ejemplo para 2 activos se deja establecido de froma teorica como sería el sistema a resolver por el método de Newton.

2 Antecedentes

La teoría moderna de portafolio [2] (MPT) es una teoría de inversión que busca maximizar el retorno y minimizar el riesgo. Esta teoría fue desarrollada por el economista Henry Markowitz [3] en 1952.

Con anterioridad al trabajo de Markowitz, por 1938, John Burr Williams [4] escribió un libro llamado "La teoría del valor de la inversión" donde describe el modelo de descuento de dividendos en donde el objetivo de hacer una buena inversión consistía en seleccionar activos individuales que presentaran buenas rentabilidades por dividendo y buenas perspectivas sin tomar en cuenta ningún otro factor.

MPT se basa en construir carteras de activos para maximizar el retorno esperado en función del nivel de riesgo. Bajo este punto de vista una alta rentabilidad va necesariamente acompañada de un nivel más alto de riesgo.

El modelo de Markowitz se basa en varios supuestos sobre el comportamiento de los inversores, en [LL10] se enlistan los siguientes:

- Los inversores pueden estimar una distribución de probabilidad de posibles rendimientos durante un período entre la compra y venta.
- Los inversores tienen funciones de utilidad por períodos únicos en los que maximizan la utilidad en el marco minimizar el monto marginal invertido.
- Los inversores utilizan la variabilidad sobre los posibles valores de rendimiento para medir el riesgo.
- Los inversores sólo utilizan el rendimiento esperado y el riesgo para tomar decisiones de inversión.

[1] el sitio oficila puede ser cosultado en [Mar]
[2] Ver [LL10] para mas detalles
[3] Markowitz recibió el premio nobel en 1990 por este trabajo
[4] Este documento [Wil38] describe el pensamiento de la epoca antes de Markowtiz

- El rendimiento esperado y el riesgo son medidos por los dos primeros momentos de la distribución de probabilidad del rendimiento, el valor esperado y la variación.
- El retorno es deseable; El riesgo debe ser evitado.

3 Teoría modernda de portafolios MPT

Markowitz sugiere dos enfoques de optimización restringida para obtener los pesos óptimos de la cartera:

- Minimizar el riesgo o la variación de la cartera, sujeto a que la cartera alcance alguna tasa esperada de retorno, y también sujeto a que la suma de los pesos de la cartera sean igual a uno.
- Máximizar la tasa esperada del retorno del portafolio sujeto a un riesgo dado y con la suma de los pesos de la cartera igual a uno.

3.1 Activos y portafolios

Un protafolio es una inversión compuesta por N activos A_n, con retornos R_n para $n \in \{1, 2, 3, ..., N\}$ donde se ha invertido una cantidad W en todo el portafolio. Sea W_n [5] la cantidad invertida en el activo A_n podemos escribir

$$\sum_{n=1}^{N} W_n = W \tag{1}$$

Consideramos $w_n = \frac{W_n}{W}$ y expresamos esta suma en términos de sus valores realtivos

$$\sum_{n=1}^{N} w_n = 1 \tag{2}$$

y

$$\sum_{n=1}^{N} w_n R_n = R \tag{3}$$

El retorno esperado del portafolio se define como:

$$\rho = E(R) = E(\sum_{n=1}^{N} w_n R_n) = \sum_{n=1}^{N} w_i r_n \tag{4}$$

Donde $r_n = E(R_n)$ es el retorno esperado de cada activo A_n

[5] Valores negativos de W_n son interpretados como short selling

Se define la covarianza entre cualesquiera dos activos i y j del portafolio como:

$$s_{ij} = E[(R_i - r_i)(R_j - r_j)] \tag{5}$$

De esta forma podemos expresar la matriz de varianzas y covarianzas

$$S = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{bmatrix} \tag{6}$$

Por último para terminar de caracterizar el portafolio se define el riesgo como la medida que cuantifica que tanta dispersión existe alrededor del retorno esperado entonces tomamos la varianza del portafolio como esta medida y la expresamos como:

$$s^2 = E(|R - \rho|^2) = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j s_{ij} = w^T S w \tag{7}$$

Donde $w = [w_1, w_2, ..., w_n]^T$ es el vector de pesos (inversión de cada activo)

3.2 Formulación del problema

En este documento nos centramos en resolver el problema con el enfoque de minimización establecido por Markowitz el cual indica que un portafolio es optimo si para un retorno esperado ρ el portafolio tiene mínima vairanza s^2.

Encontrar tal portafolio requerie resolver el siguiente problema con restricciones:

$$w = arg \min_{w} \{w^T S w\} \tag{8}$$

Sujeto a

$$w^T u = \sum_{n=1}^{N} w_n = 1 \tag{9}$$

$$w^T r = \sum_{n=1}^{N} w_n r_n = \rho \tag{10}$$

Donde $u = [1, 1, ..., 1]^T$ y $r = [r_1, r_2, ..., r_N]^T$

Entonces el problema de optimización de portafolios es un problema de optimización cuadrática con restricciones [6]

[6] En [CT06] se pueden consultar la construcción de ejemplos simples

4 Datos usados para ajustar pesos

Los datos que usaremos para ajustar el modelo corresponden a acciones de las siguientes empresas:

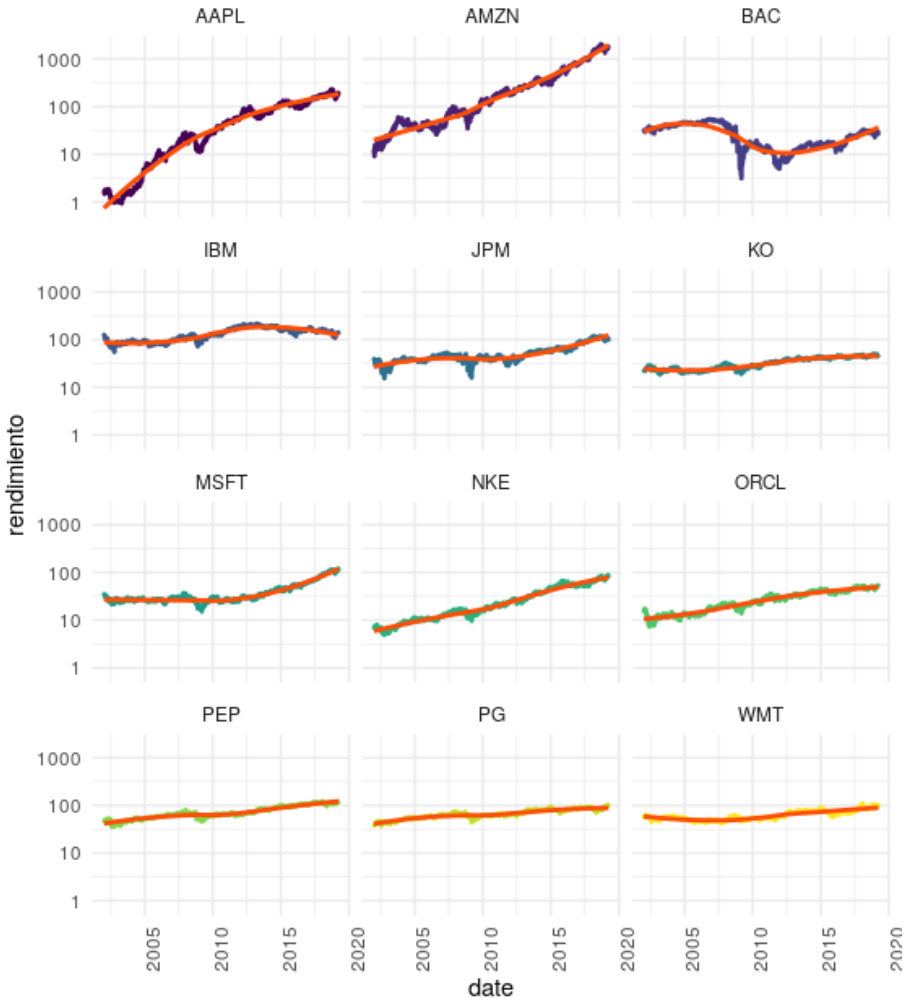
- apple (AAPL)
- amazon (AMZN)
- microsoft (MSFT)
- coca cola (KO)
- pepsi (PEP)
- bank of america (BAC)
- jp morgan (JPM)
- nike (NKE)
- oracle (ORCL)
- IBM (IBM)
- procter and gamble (PG)
- walmart (WMT)

Para obtener estos datos se creó un script [7] en R para automatizar la descarga

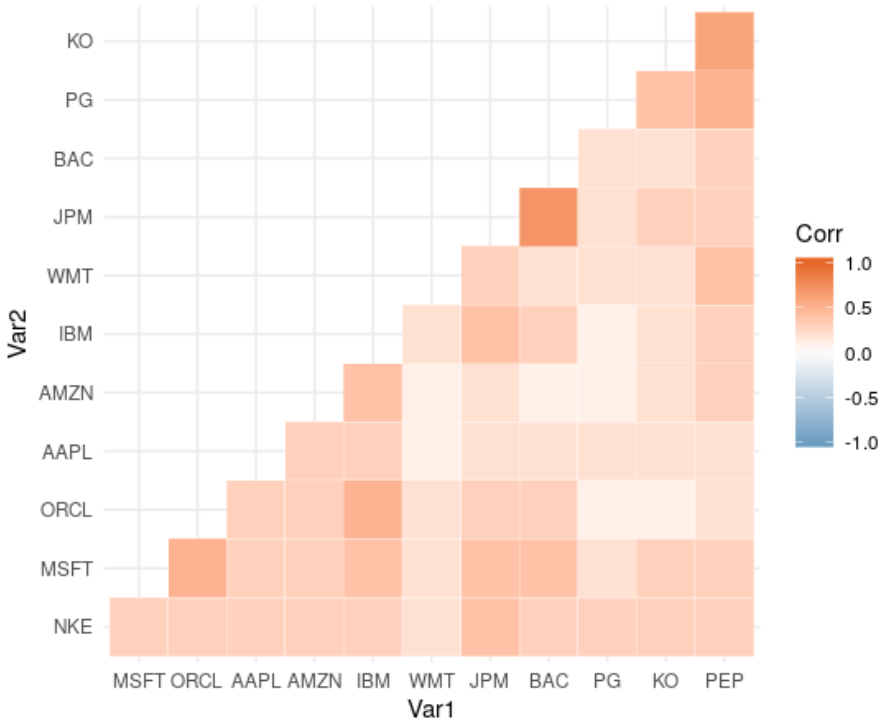
A partir de los datos descargados, exploraremos un poco su estructura [8] y analizáremos un poco como se relacionan entre ellos y el tiempo y con base en esta información aproximaremos un criterio para saber donde sería bueno invertir, todo esto con el fin de poder contrastar un poco los resultados intuitivos vs los resultados obtenidos a partir del uso de MPT.

Los precios diarios de los activos seleccionados son:

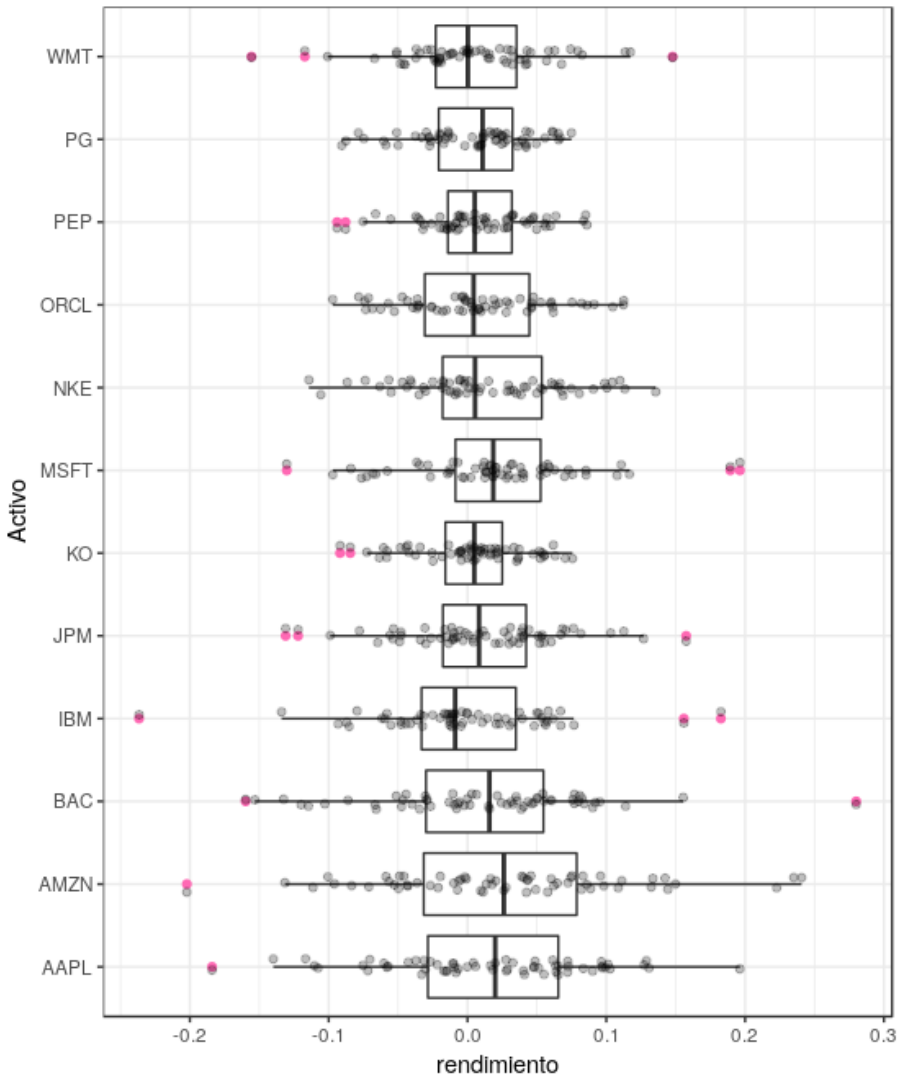
[7] El script para descargar datos se encuentra en el apéndice C
[8] El script del EDA se encuentra en el apéndice D



Esté chart muestra los precios diarios de cada acción desde el 2002 hasta el 2019, donde observamos que AAPL y AMZN hisoricamente han incrementado el valor de sus acciones con una tendencia creciente muy fuerte.



Esta gráfia de correlaciones considera todos los periodos de los rendimientos mensuales y los activos que se mueven juntos de forma mas fuerte son JPM y BAC, despues PEP y KO.



La gráfica muestra la distribución interquanrtile de los rendimientos de las acciones desde 2014 hasta 2019.

De esta última gráfica observamos que KO tiene la distribución mas concentrada i.e. es la que menos varia (tiene menor riesgo), AMZN es la que mas ha variado (pasa algo similar con AAPL) pero si consideramos el tiempo también sabemos (Por la primera gráfica) que el valor de las acciones de AMZN son los que mas han crecido.

Otro dato importante que notamos es sobre la mediana de los rendimientos

de IBM ya que su valor es negativo, y al revisar la gráfica 1 observamos que tiene desde el 2013 una tendencia a la baja en sus precios diarios.

Con base en esta información podriamos proponer una inversión mayoritaria en acciones de AMZN y AAPL sin usar MPT.

4.1 Solución por simulación

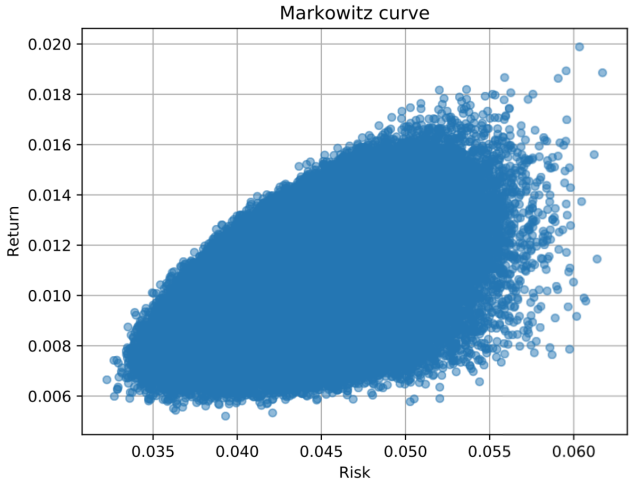
La primera solución que trabajaremos está dada por simulación, es un método heurístico pero muy simple de implementar y el obetivo es tener un primer resultado sin profundizar en la teoría

Supongamos que se tienen n activos, para simular un portafolio lo único que necesitamos es obtener la matriz de covarianza y la esperanza de los rendimientos de cada activo.

Result: Simulación de portafolio[a]
$y_0 = x_0$
for *t=0 to total simulaciones* **do**
  W = obten un vector que sume 1 aleatorio
  $E(R_p) = W^T R$
  $\sigma_p^2 = W^T S W$
end
return W

[a] El código en python puede ser consultado en el apéndice C

Al final tenemos un cantidad muy grande de portafolios aleatorios, y para cada uno vemos cual es su retorno y su rendimiento. Si extraemos esta información y calculamos la raíz cuadrada de la varianza por cada portafolio podemos graficarlo de la siguiente forma.



Hay muchos casos donde para un riesgo dado existen mayores rendimientos, y donde se obtiene el máximo por cada activo le llamamos frontera eficiente, que es la linea superior de los puntos azules.

5 Optimización

El problema de optimización de portafolios puede ser resuelto de la siguiente manera:

1. Incorporar las restricciones a la función objetivo por medio de penalizaciones (multiplicadores de lagrange)
2. Expresar el problema via la matriz de Karush–Kuhn–Tucker
3. Resolver por algún método de optimización numérica (descenso, newton, etc.)

Entonces considerando el problema original:

$$\min_{w} w^T S w \tag{11}$$

sujeto a:

$$\sum_{i=1}^{n} w_i = 1$$

$$\sum_{i=1}^{n} w_i r_i = \rho$$

Incorporamos las restricciones:

$$L(w, \lambda_1, \lambda_2) = w^T S w - \lambda_1 (w^T u - 1) - \lambda_2 (w^T r - \rho)$$

la condición de optimalidad indica que para resolver el problema general:

$$\min_x f(x) \tag{12}$$

sujeto a:

$$Ax = b$$

se debe cumplir:

- f es convexa y 2 veces continuamente diferenciable
- $A \epsilon R^{pxn}$ con rango $A = p$
- Asumimos que p^* es finita y acotada

y las condiciones de optimalidad: x^* es optimal si y sólo si existe v^* tal que

$$\nabla f(x^*) + A^T v^* = 0$$
$$Ax^* = b$$

Entonces para el problema general de minimización cuadrática [9] con restricciones de igualdad

$$\min_x \frac{1}{2} x^T P x + q^T x + r \tag{13}$$

sujeto a:

$$Ax = b$$

escribimos las condiciones de optimalidad de la siguiente forma:

$$\begin{bmatrix} P & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ v^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}$$

Ejemplo: En el caso del problema de optimización de portafolios si consideramos los activos A y B las condiciones de optimalidad quedan expresadas de la siguiente forma:

$$\begin{bmatrix} 2\sigma_A^2 & 2\sigma_{AB} & 1 & r_A \\ 2\sigma_{AB} & 2\sigma_B^2 & 1 & r_B \\ 1 & 1 & 0 & 0 \\ r_A & r_B & 0 & 0 \end{bmatrix} \begin{bmatrix} w_A^* \\ w_B^* \\ \lambda_1^* \\ \lambda_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \rho \end{bmatrix}$$

Para encontrar los pesos tendriamos que resolver el sistema de ecuaciones [10].

[9] Consultar el capítulo 4.4 de [BV04]

[10] En el apendice B se encuentra el código para resolver este problema de forma muy simple haciendo uso de de la función solve

5.1 Solución al problema de optimización de portafolios por el método de Newton

El problema consiste en resolver la aproximación de segundo orden, ver [NW06]

$$\min \hat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v \tag{14}$$

sujeto a:

$$A(x+v) = b$$

De la linearización de las condiciones de optimalidad se sigue :

$$\nabla f(x+v) + A^T w \approx \nabla f(x) + \nabla^2 f(x) v + A^T w = 0 \tag{15}$$
$$A(x+v) = b$$

El paso de Newton Δx_{nt} de f en un x factible esta dado por la solución v de

$$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$$

Ejemplo: En el caso del problema de optimización de portafolios si consideramos los activos A y B se debe resolver:

$$\begin{bmatrix} 2\sigma_A^2 & 2\sigma_{AB} & 1 & r_A \\ 2\sigma_{AB} & 2\sigma_B^2 & 1 & r_B \\ 1 & 1 & 0 & 0 \\ r_A & r_B & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta w_A^* \\ \Delta w_B^* \\ \lambda_1^* \\ \lambda_2^* \end{bmatrix} = \begin{bmatrix} -(2\sigma_A^2 w_A + 2\sigma_{AB} w_B) \\ -(2\sigma_{AB} w_A + 2\sigma_B^2 w_B) \\ 0 \\ 0 \end{bmatrix}$$

5.2 Algoritmo programado en CVXOPT para la solución al problema de optimización de portafolios

La libreria CVXOPT nos ayuda a resolver problemas de optimización cuadrática con restricciones donde sólo tenemos que especificar el problema en su forma general:

$$\min \frac{1}{2} x^T P x + q^T x \tag{16}$$

sujeto a:

$$Gx \leq h$$
$$Ax = b$$

Cargamos las librerias que usaremos

```
import numpy as np
import pandas as pd
from cvxopt import blas, solvers, matrix
```

Especificamos los parametros de nuestro problema y creamos una función para encontrar los pesos óptimos, está función que presentamos tiene la capacidad de cálcular pesos para $w_i \geq 0$ o bien encontrar pesos para permitir ventas en corto.

Cuando la función no acepte ventas en corto se activara la restricción:

$$Gx \leq h$$

Que para el problema de optimziación de portafolios es de la forma:

$$w_i > 0$$

Y por lo tanto debe ser modificada de la siguiente forma

$$-w_i < 0$$

Esto con el fin de ser congruentes con la forma general del problema de optimización cuadratica

Cuando las ventas en corto sean aceptadas las matrices G y h serán nulas.

```
def markowitz(r_i, sigma, rho, Ventas_en_corto = False):
    ##########################################################################
    # Se defienen las matrices principales con la equivalencia
    # entre el problema general de optimizacion y el problema
    # de optimizacion de markowitz
    ##########################################################################
    d = len(sigma)
    P = matrix(sigma)
    q = matrix(np.zeros(d, float))
    A = matrix([
        np.ones(d, float).tolist(), r_i.tolist()
    ]).trans()
    b = matrix([1.0, float(rho)])
    ##########################################################################
    # Se define la restriccion para cuando las ventas en corto
    # no estan permitidas
    ##########################################################################
    diagonal = np.zeros((d, d), float)
    np.fill_diagonal(diagonal, -1.0)

    G = matrix(diagonal)
    h = matrix(np.zeros(d, float))
    ##########################################################################
    # Se revisa la condicion de Ventas en corto para asignar
    # valores a G y h
```

```
    ##################################################################
    if Ventas_en_corto == True:
        G=None
        h=None
    else:
        G=G
        h=h
    ##################################################################
    # Una vez que estan definidos todos los elementos se
    # resuelve el problema usando solvers.qp
    ##################################################################
    sol = solvers.qp(P=P, q=q, G=G, h=h, A=A, b=b)
    w = list(sol['x'])
    ##################################################################
    return w
```

6 Comparación de resultados

Para probar los resultados de está función usaremos los datos descargados y contrastaremos con los resultados obtenidos por una función simple en R que resuleve el problema de condiciones de optimalidad, el script de la función de R puede ser consultado en el apéndice B.

Probamos con un rendimiento de .009

6.1 Función markowitz con ventas en corto

Cargamos los datos con los que vamos a trabajar

```
return_data = pd.read_csv('monthly_return.csv')
return_data = return_data[
        [i for i in return_data.keys() if i not in ('date')]
        ]
return_data = return_data.T.values
```

Obtenemos el vector promedio de los rendimientos y la matriz de varianzas y covarianzas

```
mean_returns = np.mean(return_data, axis=1)
sigma_returns = np.cov(return_data)
```

habilitamos el parametro para permitir ventas en corto

```
markowitz(mean_returns, sigma_returns, .009, Ventas_en_corto = True)
```

6.2 Función markowitz para pesos $w_i > 0$

Ahora probaremos con la restricción de no permitir ventas en corto usando la función en CVXOPT:

```
markowitz(mean_returns, sigma_returns, .009, Ventas_en_corto = False)
```

6.3 Función en R para solución por condiciones de optimalidad

cargamos las librerias que ocuparemos

```
library(readr)
library(dplyr)
```

cargamos los datos y obtenemos los inputs necesarios

```
returns<- 'monthly_return.csv' %>%
  read_csv %>%
  select(-date)

sigma_returns<-cov(returns)
mean_returns<-colMeans(returns)
```

Aplicamos la función

```
markowitz_simple_con_condicones_de_optimalidad(mean_returns,
                                               sigma_returns,
                                               .009)
```

6.4 Resultados

Activos	Markowitz con $w_i > 0$	Markowitz con short selling	Markowitz R
AAPL	0.08015190946661055	0.07915794106844312	0.079157941
AMZN	0.04181040751829357	0.036079832504524914	0.036079833
MSFT	0.005946541493249802	0.021507255888848352	0.021507256
KO	0.12891674633867195	0.1259844942559241	0.125984494
PEP	0.03905755811070346	0.0525305451552263	0.052530545
JPM	1.8151264568346091e-06	0.011552341289544884	0.011552341
BAC	2.755324102535624e-07	-0.045930434536501454	-0.045930435
NKE	0.11815216755220508	0.12718427165620566	0.127184272
ORCL	0.04290046093711035	0.052203417482306856	0.052203417
IBM	0.0006533481772138372	-0.0018511569959980506	-0.001851157
PG	0.3337369963710057	0.33798150505511976	0.337981505
WMT	0.2086717733760687	0.2035999871763553	0.203599987

7 Conclusiones

Exploramos distintas formas de optimización de portafolios, evaluamos cómo sería el proceso de optimización utilizando simulaciones, planteando el problema de optimización expresándose usando la matriz de Karosh Kuhn Tucker y resolviendo el sistema de ecuaciones, planteamos la solución teórica utilizando el método de Newton, y por último utilizando el paquete de CVXOPT.

Comparando las respuestas usando Markowitz con short selling en CVXOPT, y Markowitz resolviendo la matriz en R usando un solve, obtenemos las mismas respuestas.

Las respuestas también son congruentes con la "intuición", es decir, nosotros escogimos minimizar el riesgo, dado un retorno fijo, en este caso, nosotros escogimos un retorno bastante conservador de .09, esto implica, que nuestro modelo tendrá mucha flexibilidad para minimizar la varianza. Y podemos ver en los resultados, que así fue, y que las acciones a las que se le dieron prioridad, fueron acciones, que si observamos las gráficas, con muy poca varianza, como PG, WMT, y KO. Y acciones con mucho mayor rendimiento pero mucha varianza, como lo es Amazon y Apple, se les dio un peso pequeño.

Algunos siguientes pasos que no se realizaron, pero se podrían realizar, es comparar el costo computacional de los algoritmos explorados. Nosotros usamos pocas acciones, pero si aumentamos el número de acciones de manera significativa, por ejemplo, si estamos evaluando todas las acciones disponibles en el mercado, quizás los métodos realizados no sean computacionalmente factibles, entonces sería explorar métodos de escalamiento o de cómputo distribuido o paralelo.

8 Apéndice

A Código en R para bajar acciones

Esta función sirve para bajar los precios diarios y los retornos mensuales de un conjunto de acciones dadas. Recibe como input un conjunto de acciones fechas inicial y fecha final, el output de la función es una lista que contiene un data frame con los precios diarios y otro data frame con los rendimientos mensuales

```
Se definen las librerias que se usaran
#############################################################
library(quantmod)
library(purrr)
library(readr)
library(dplyr)
library(tibble)
#############################################################

obtener_rendimientos<-function(Activos,
                               Fecha_inicial,
                               Fecha_final){

  # bajamos los activos
  portafolio_prices<-Activos %>%
    map(~getSymbols.yahoo(.,
                          from=Fecha_inicial,
                          to = Fecha_final,
                          periodicity = "daily",
                          auto.assign=FALSE)[,4])

  # Obtenemos los precios diarios a partir de
  # portafolio_prices
  precios_diarios<-portafolio_prices %>%
    map(~as.data.frame(.) %>%
          rownames_to_column(var = "date")) %>%
    reduce(left_join, by = "date") %>%
    set_names(c("date", Activos)) %>%
    mutate(date = date %>% as.Date)

  # Obtenemos los rendimientos mensuales a
  # partir de portafolio_prices
  rendimientos_mensuales<-portafolio_prices %>%
    map(~monthlyReturn(.) %>%
          as.data.frame %>%
          rownames_to_column(var = "date")) %>%
    reduce(left_join, by = "date") %>%
```

```
    set_names(c("date", Activos)) %>%
    mutate(date = date %>% as.Date)

  precios_y_rendimientos<-list(
    PreciosDiarios = precios_diarios,
    RendimientosMensuales = rendimientos_mensuales)

  return(precios_y_rendimientos)
}
```

B Código simple en R para solución por condiciones de optimalidad

Está función cálcula los pesos optimos w_i con ventas en corto. el output es un data.frame con los w_i's optimos y coeficientes de penalización. La función recibe los siguientes inputs:

- r_i un vactor que contiene todos los promedios de los activos
- Sigma una matriz de varianzas y covarianzas de los activos
- rho un rendimiento deseado

```
library(readr)
library(dplyr)

markowitz_simple_con_condicones_de_optimalidad<-function(r_i,
                                                         Sigma,
                                                         rho){
  ##################################################################
  # Definimos la matriz de Karush Kuhn Tucker
  kkt_matrix<-(2*Sigma) %>%
    rbind(1) %>%
    rbind(r_i) %>%
    cbind(c(rep(1.0,ncol(Sigma)),0.0, 0.0)) %>%
    cbind(c(r_i, 0, 0))
  ##################################################################
  # Definimos la parte derecha del sistema de
  # ecuaciones a resolver
  b<-c(rep(0,ncol(Sigma)),1, rho)
  ##################################################################
  # Se cAclucla la solucion del problema
  w_i<-solve(kkt_matrix, b, tol = 1e-7) %>%
    as.data.frame
```

```
  ##################################################################
  return(w_i)
}
```

C Código en Python para la simulación

```
# -*- coding: utf-8 -*-
"""
This script optimize portfolio returns using Markowitz curve
"""
import numpy as np
import matplotlib.pyplot as plt
from markowitz import markowitz
import pandas as pd
import matplotlib
import random
matplotlib.use('Agg')
from matplotlib.backends.backend_pdf import PdfPages

return_data = pd.read_csv('monthly_return.csv')
return_data = return_data[
    [i for i in return_data.keys() if i not in ('date')]
    ]
return_data = return_data.T.values
n_portfolios = 100000
mean_returns = np.mean(return_data, axis=1)
sigma_returns = np.cov(return_data)


def generate_portfolio():
    """
    For a given matrix with returns, generate random portfolio
    W are the weights of stocks in a given portfolio
    sum(w) = 1
    Sigma is the covariance matrix
    Expected return form portfolio = w * x^T
    Portfolio variance = w * Sigma * w^T
    The simulation process consist in generate some
    random weights and get the variance and expected returns
    for those random weights. If you do this many times
    you will find the optimal values for a particular
    portfolio allocation balancing
    risk and reward
    """
```

```
    w = np.random.random(len(return_data))
    w /= np.sum(w)

    x = np.asmatrix(mean_returns)
    w_matrix = np.asmatrix(w)
    covariance_matrix = np.asmatrix(sigma_returns)

    expected_return = w_matrix * x.T
    risk = np.sqrt(w_matrix * covariance_matrix * w_matrix.T)
    return risk, expected_return


def main():
    """
    General calls
    """
    # Get random portfolio
    random_risk, random_return = np.column_stack(
        [generate_portfolio() for i in range(n_portfolios)]
        )
    random_risk = [j[0] for j in random_risk]
    random_return = [k[0] for k in random_return]

    returns_random_low = random.sample(random_return, k=10000)
    optimals = [markowitz(
        mean_returns, sigma_returns, i
        ) for i in returns_random_low]
    optimal_means = [np.dot(i, mean_returns) for i in optimals]

    optimal_variance = [np.sqrt(
        float(np.asmatrix(i) * sigma_returns * np.asmatrix(i).T)
        ) for i in optimals]

    # Plot results
    pdf = PdfPages('markowitz_curve.pdf')
    plt.figure()
    plt.plot(random_risk, random_return, 'o', markersize=5, alpha=0.5)
    plt.plot(optimal_variance, optimal_means, 'o', alpha=0.7, color='red')
    plt.xlabel('Risk')
    plt.grid()
    plt.ylabel('Return')
    plt.title('Markowitz curve')
    pdf.savefig()
    plt.close()
    pdf.close()
```

```
if __name__ == "__main__":
    main()
```

D Código en R para EDA

```
#########################################################################
################################## EDA ##################################
#########################################################################
# Cargamos las librerias que usaremos
library(readr)
library(ggplot2)
library(tidyr)
library(dplyr)
library(ggridges)
library(viridis)
library(lubridate)
library(ggcorrplot)
#########################################################################
# Cargamos los datos que usaremos
retornos_mensuales<-read_csv("monthly_return.csv")
precios_diarios<-read_csv("daily_price.csv")
#########################################################################
# Boxplots para rendimientos mensuales del 2014 al 2019
retornos_mensuales %>%
  gather(Activo, rendimiento, -date) %>%
  mutate(Year = year(date) ) %>%
  filter(Year %in% c(2014, 2015, 2016, 2017, 2018, 2019)) %>%
  ggplot(aes(x = Activo, y = rendimiento)) +
  #scale_y_log10() +
  geom_boxplot(outlier.colour = "hotpink") +
  geom_jitter(position = position_jitter(width = 0.1, height = 0),
              alpha = 1/4)+
  theme_bw() +
  coord_flip()
#########################################################################
# correlaciones mensuales para todos los datos
round(cor(retornos_mensuales %>% select(-date)), 1) %>%
  ggcorrplot(hc.order = TRUE, type = "lower",
             outline.col = "white",
             ggtheme = ggplot2::theme_gray,
             colors = c("#6D9EC1", "white", "#E46726")) +
  theme_minimal()
#########################################################################
# Series de los precios diarios
precios_diarios %>%
  gather(Activo, rendimiento, -date) %>%
```

```
mutate(Year = year(date) ) %>%
ggplot(aes(x = date, y = rendimiento)) +
geom_line(aes(color = Activo), size = 1) +
stat_smooth(
  color = "#FC4E07", fill = "#FC4E07",
  method = "loess") +
scale_y_log10()+
scale_color_viridis(discrete=TRUE) +
theme_minimal() +
theme(legend.position = "none",
      axis.text.x = element_text(angle = 90)) +
facet_wrap(~Activo, ncol=3)
```

Referencias

[Wil38] John Burr Williams. *The theory of investment value*. Tech. rep. 1938.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CT06] Gerard Cornuejols and Reha Tütüncü. *Optimization methods in finance*. Vol. 5. Cambridge University Press, 2006.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[LL10] Cheng-Few Lee and John Lee. *Handbook of quantitative finance and risk management*. Springer Science & Business Media, 2010.

[Mar] Joachim Dahl Martin Andersen. *Python Software for Convex Optimization*. URL: `https://cvxopt.org/`. (accessed: 01.05.2019).