Proyecto 1, Modelos de Gestión Financiera

Sebastian Puerto

25 de septiembre de 2019

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import csv
from tabulate import tabulate
np.set_printoptions(precision=6) #Mostrar numeros con maximo seis digit
np.set_printoptions(suppress=True) # Suprimir uso de notacion cientific
```

In [2]:
```
N = 23 # Numero de periodos
mYah = 0 # Numero de acciones extraidas de Yahoo
mFin = 6 # Numero de acciones extraidas de Finance.com
M = mYah + mFin # Numero de acciones

precios = np.zeros((M, N+1))
retornos = np.zeros((M, N))
```

Extrayendo columnas de los archivos de Yahoo Finance:

In [3]:
```
archivosYahoo = ['csvs/EC.csv','csvs/CIB.csv']
```

In [4]:
```
for k in range(mYah):
    archivo = archivosYahoo[k]
    lector = csv.reader(open(archivo))
    lector.__next__() #Ignorar primer renglon

    for i in range(N+1):
        precios[k, i] = lector.__next__()[5] #Extraer la 5ta columna: p
        if (i > 0):
            retornos[k, i-1] = (precios[k, i] - precios[k, i-1])/precio
```

Extrayendo columnas de los archivos de Investing.com

In [5]:
```
 archivosFin = [ 'csvs/CFV-2.csv', 'csvs/FTSE-2.csv', 'csvs/IMI-2.csv',
```

In [6]:
```
for k in range(mYah, mYah + mFin):
    archivo = archivosFin[k - mYah]
    lector = csv.reader(open(archivo))
    lector.__next__() #Ignorar primer renglon

    for i in range(N, -1, -1):
        precios[k, i] = lector.__next__()[1] #Extraer la 1era columna: p
        if (i < N):
            retornos[k, i] = (precios[k, i+1] - precios[k, i])/precios[l
```

In [7]:
```
np.shape(retornos)
```

Out[7]:
```
(6, 23)
```

TODO: Explicar, a la luz de la lectura 1, la naturaleza de esos activos.

3

Vector de Rendimientos promedio

In [8]:
```
retProm = np.mean(retornos, 1, keepdims = True) # Hallar el promedio de

print("Tamaño de matriz de rentabilidades:", np.shape(retProm)) # En efe
print("Retornos promedio:\n", retProm)
```

```
Tamaño de matriz de rentabilidades: (6, 1)
Retornos promedio:
 [[0.001484]
 [0.010866]
 [0.005016]
 [0.019599]
 [0.018384]
 [0.003606]]
```

Matriz de Covarianzas

In [9]:
```python
S = np.zeros((M,M)) # Inicializacion en 0's

for k in range(M): # Iterar con k sobre activos
    for l in range(M): # Iterar con l sobre activos
        for i in range(N): # Iterar sobre el tiempo con i
            # Para la combinacion de activos k y l se suma la contribuc
            S[k, l] += (retornos[k, i] - retProm[k])*(retornos[l, i] -

S = S/N

print("Matriz de covarianzas:\n", S)
```

```
Matriz de covarianzas:
 [[ 0.008515  0.003032  0.00307   0.002884  0.00071  -0.001638]
 [ 0.003032  0.00234   0.001852  0.002165  0.000496 -0.000257]
 [ 0.00307   0.001852  0.003651  0.00176   0.001199 -0.000463]
 [ 0.002884  0.002165  0.00176   0.003089  0.000381 -0.00086 ]
 [ 0.00071   0.000496  0.001199  0.000381  0.006055 -0.00046 ]
 [-0.001638 -0.000257 -0.000463 -0.00086  -0.00046   0.007506]]
```

In [10]:
```python
varianzas = np.array([S[i, i] for i in range(M)]).reshape((M, 1))
desvs = np.sqrt(varianzas)

print("\nVarianzas:\n", varianzas)


def varPort(x): # Funcion de calculo de varianza de un portafolio x^T S
    return x.T.dot(S).dot(x)[0,0]

print(varPort(np.array([1,0,0,0,0,0]).reshape(M,1))) # Verificacion de 
print(varPort(np.array([0,0,0,1,0,0]).reshape(M,1))) # Verificacion de 
```

```

Varianzas:
 [[0.008515]
 [0.00234 ]
 [0.003651]
 [0.003089]
 [0.006055]
 [0.007506]]
0.008514831703589519
0.0030887769402972985
```

Matriz de Covarianzas es Definida Positiva

Matriz Cuadrada

In [11]:
```python
np.shape(S)
```

Out[11]:
```
(6, 6)
```

Simétrica

In [12]:
```
S - S.transpose() # Debería ser igual a su transpuesta,  y lo es pues s
```

Out[12]:
```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

Definida Positiva

Usamos la caracterización vista en clase: los subdeterminantes en línea son positivos. En particular vemos que el determinante de la matriz es distinto de 0.

In [13]:
```
for k in range(1,M):
    submatPpal = S[:k, :k]
    detp = np.linalg.det(submatPpal)
    print("El determinante de la submatriz  de tamaño ", np.shape(subma
```

```
El determinante de la submatriz  de tamaño  (1, 1)  que contiene la en
trada 1,1, es decir la matriz
 [[0.008515]]  es  0.008514831703589522 != 0

El determinante de la submatriz  de tamaño  (2, 2)  que contiene la en
trada 1,1, es decir la matriz
 [[0.008515 0.003032]
 [0.003032 0.00234 ]]  es  1.073195456246791e-05 != 0

El determinante de la submatriz  de tamaño  (3, 3)  que contiene la en
trada 1,1, es decir la matriz
 [[0.008515 0.003032 0.00307 ]
 [0.003032 0.00234  0.001852]
 [0.00307  0.001852 0.003651]]  es  2.240673462900374e-08 != 0

El determinante de la submatriz  de tamaño  (4, 4)  que contiene la en
trada 1,1, es decir la matriz
 [[0.008515 0.003032 0.00307  0.002884]
 [0.003032 0.00234  0.001852 0.002165]
 [0.00307  0.001852 0.003651 0.00176 ]
 [0.002884 0.002165 0.00176  0.003089]]  es  2.4269164187072262e-11 !=
0

El determinante de la submatriz  de tamaño  (5, 5)  que contiene la en
trada 1,1, es decir la matriz
 [[0.008515 0.003032 0.00307  0.002884 0.00071 ]
 [0.003032 0.00234  0.001852 0.002165 0.000496]
 [0.00307  0.001852 0.003651 0.00176  0.001199]
 [0.002884 0.002165 0.00176  0.003089 0.000381]
 [0.00071  0.000496 0.001199 0.000381 0.006055]]  es  1.36818180405477
3e-13 != 0
```

In [14]:
```
Sinv = np.linalg.inv(S)
print(Sinv)
```

```
[[ 239.905682 -260.093503  -72.693514    9.855839   10.037876   40.682
241]
 [-260.093503 1792.633235 -271.221102 -889.34298   -15.336563 -114.821
325]
 [ -72.693514 -271.221102  505.248761  -21.704035  -67.987994   -0.643
032]
 [   9.855839 -889.34298   -21.704035  970.796515   21.156713   82.809
258]
 [  10.037876  -15.336563  -67.987994   21.156713  178.171117   10.812
737]
 [  40.682241 -114.821325   -0.643032   82.809258   10.812737  148.271
907]]
```

In [15]:
```
S.dot(Sinv)
```

Out[15]:
```
array([[ 1.,  0.,  0.,  0., -0.,  0.],
       [-0.,  1.,  0., -0.,  0., -0.],
       [ 0.,  0.,  1., -0.,  0., -0.],
       [-0., -0.,  0.,  1.,  0., -0.],
       [ 0., -0.,  0., -0.,  1., -0.],
       [-0.,  0.,  0., -0.,  0.,  1.]])
```

4

Parámetros de la teoría

In [16]:
```
u = np.ones((M, 1))

A = u.T.dot(Sinv.dot(u))[0,0]
B = u.T.dot(Sinv.dot(retProm))[0,0]
C = retProm.T.dot(Sinv.dot(retProm))[0,0]
D = A*C - B**2

print("A =", A, "\t B =", B, "\t C =", C, ", entonces D =", D)
```

```
A = 758.0484469476713    B = 9.45611444074225    C = 0.240103968678626
46 , entonces D = 92.59234024639102
```

5

Ecuación general de los portafolios óptimos dados los parámetros de la teoría A, B, C y el parámetro μ: $x*(\mu) = (\frac{C-B\mu}{D})S^{-1}\hat{u} + (\frac{A\mu-B}{D})S^{-1}\bar{r}$.

La varianza de estos portafolios, dada como función de μ tiene la fórula $\sigma^2(\mu) = \frac{A\mu^2-2B\mu+C}{D}$, la cual llamamos la frontera eficiente.

En nuestro caso, como:

In [17]:
```
def xOptMu(mu): # Funcion para calculo del portafolio optimo dado param
    return ((C - B*mu)/D) * Sinv.dot(u) + ((A*mu - B)/D) * Sinv.dot(ret


#Test para ver que xOptMu funciona bien: comprobar que en mu = B/A se h
#print(xOptMu(B/A)) # Portafolio calculado
#print(1/A * Sinv.dot(u)) # Portafolio optimo teorico
# print(varPort(xOptMu(B/A)), 1/A) # Varianzas calculadas y teorica

rentDeseadas = np.linspace(0., 0.03, 9)

portOptMu = np.zeros( (M, len(rentDeseadas)) )

for i in range(len(rentDeseadas)):
    portOptMu[:,i] = xOptMu(rentDeseadas[i]).flatten()

print("\t\t PORTAFOLIOS EFICIENTES")
print("Cada fila corresponde a un activo y cada columna a un valo de mu
print(tabulate(portOptMu, rentDeseadas))
```

```
                  PORTAFOLIOS EFICIENTES
Cada fila corresponde a un activo y cada columna a un valo de mu disti
nto

       0.0     0.00375      0.0075     0.01125       0.015     0.01875
0.0225     0.02625        0.03
----------  ----------  ----------  ----------  ----------  ----------
----------  ----------  ----------
 0.15218     0.0936206   0.0350612  -0.0234982  -0.0820576  -0.140617
-0.199176   -0.257736   -0.316295
 0.667353    0.562632    0.457911    0.353189    0.248468    0.143747
0.0390256  -0.0656957  -0.170417
 0.40858     0.313909    0.219239    0.124568    0.0298975  -0.0647732
-0.159444   -0.254114   -0.348785
-0.566745   -0.327538   -0.0883313   0.150876    0.390083    0.62929
0.868497    1.1077      1.34691
 0.0243601   0.071309    0.118258    0.165207    0.212156    0.259105
0.306053    0.353002    0.399951
 0.314272    0.286067    0.257863    0.229658    0.201454    0.173249
0.145044    0.11684     0.0886353
```

En todos los portafolios hay posiciones en corto para al menos un activo.

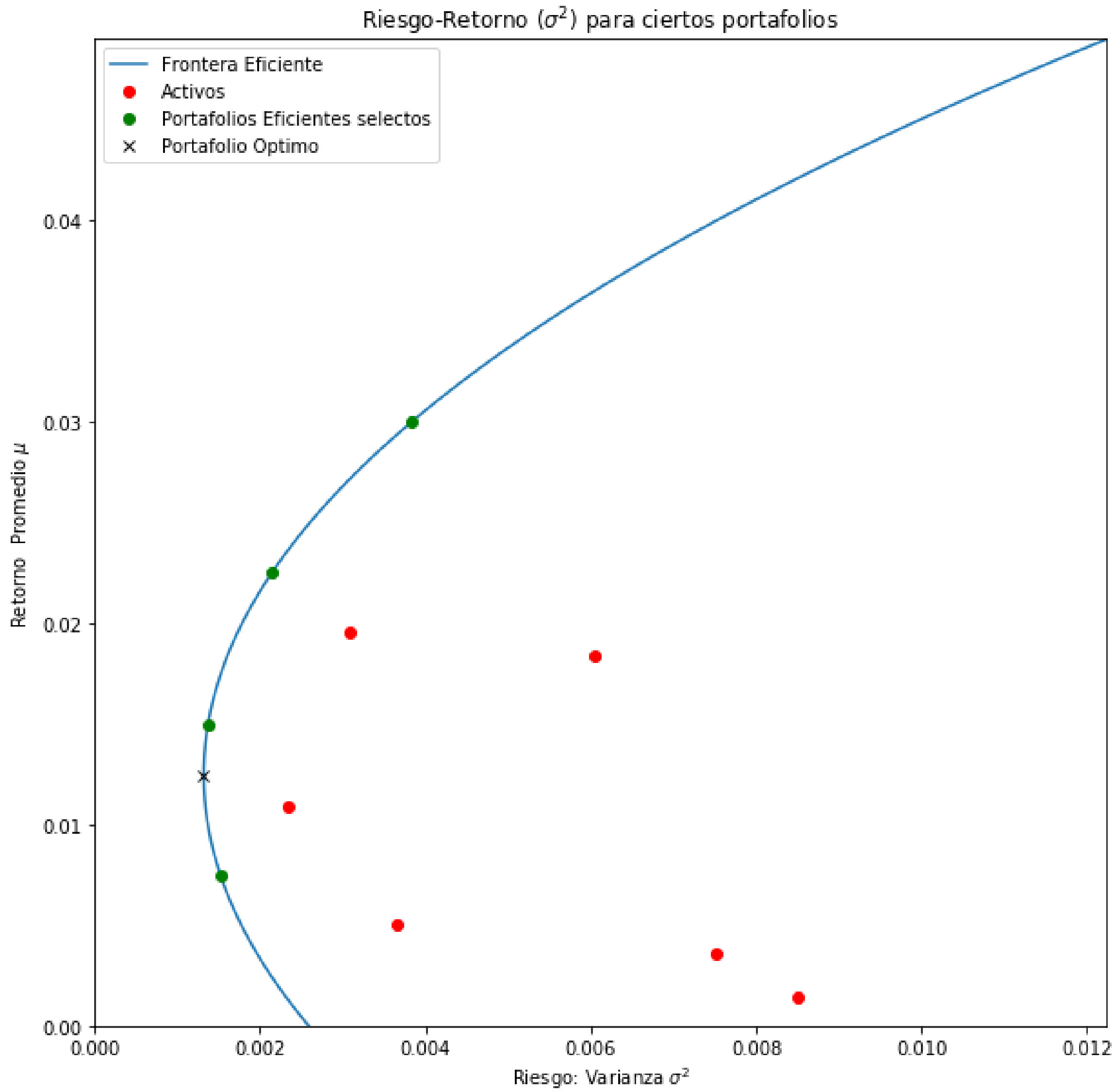
Escogeremos los siguientes 4 portafolios eficientes:

In [18]:
```
musElegidos = [rentDeseadas[2], rentDeseadas[4], rentDeseadas[6], rentD
# Matriz cuya entrada i sera el portafolio i de tamaño Mx1
portElegidos = [xOptMu(mu).reshape(M,1) for mu in musElegidos]
varElegidos = [varPort(xOptMu(mu)) for mu in musElegidos]
```

6

In [19]:
```python
mus = np.linspace(-0.005, 2.5*np.max(retProm), 200)
varFront = (A*mus**2 - 2*mus*B + C)/D

plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma^2$) para ciertos portafolios")

plt.xlabel("Riesgo: Varianza $\sigma^2$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = max(varFront))
plt.ylim(bottom = 0, top = max(mus))

# Frontera eficiente
plt.plot(varFront, mus)

# Activos elegidos
plt.plot(varianzas, retProm, 'ro')

# 4 Portafolios Eficientes
plt.plot(varElegidos, musElegidos, 'go')

plt.plot(1/A, B/A, 'kx')

plt.legend(("Frontera Eficiente", "Activos", "Portafolios Eficientes se
```

Out[19]:
```
<matplotlib.legend.Legend at 0x7fee18fbaef0>
```



7

Calculo de la tasa libre de riesgo correspondiente a cada portafolio eficiente, es decir el corte de su linea de mercado con el eje de rentabilidad.

Funciones Ayudantes

La rentabilidad promedio del portafolio x se halla como $\bar{r}_x = \bar{r}^T x$.

La línea de mercado de capital correspondiente a un portafolio eficiente es la línea tangente a la frontera eficiente que que pasa por este portafolio. Para calcular su fómrmula basta diferenciar implícitamente la fórmula de la frontera eficiente $\sigma^2(\mu) = \frac{A\mu^2 - 2B\mu + C}{D}$. Primero pasamos la D a multiplicar, así que obtenemos: $D\sigma^2 = A\mu^2 - 2B\mu + C$. Al derivar implícitamente respecto a σ (coordenada x) obtenemos $2D\sigma = 2A\mu\mu' - 2B\mu'$. De aquí se deduce que la pendiente de la línea de mercado de capital tiene pendiente $\mu' = \frac{2D\sigma^*}{2A\mu^* - 2B}$ (dados σ^*, μ^* de la frontera eficiente). Esto significa que la linea de mercado de capital tenga una ecuación de la forma $\mu = \tau + \mu'_{\sigma*,\mu*}\sigma$; como (σ^*, μ^*) pertenece a esta recta, se deduce que el punto de corte τ es $\tau = \frac{2A(\mu^*)^2 - 2B\mu^* - 2\sigma^2 D}{2A\mu^* - 2B}$. En el punto 10 usamos la formula hallada del problema de optimización de pendiente y comprobamos que estas 2 formulas coinciden, al menos para los portafolios escogidos.

In [20]:
```
def muPort(port): # Funcion que calcula la rentabilidad promedio de un p
    return port.T.dot(retProm)[0,0]

# Funcion que calcula el punto de corte de la linea de mercado de capita
# de rentabilidad mu
def corteMu(mu):
    var = varPort(xOptMu(mu))
    return (2*A*mu**2 - 2*B*mu - 2*var*D)/(2*A*mu - 2*B)

# Verificacion de que la funcion que calucula el rendimiento de un porta
#print(muPort(np.array([1, 0, 0, 0, 0, 0]).reshape(m, 1))) #0.0014838
#print(muPort(np.array([0, 0, 0, 0, 1, 0]).reshape(m, 1))) #0.018384

# Verificacion de que tauMu es calculado correctamente
#print(tauMu(muPort(portOptMu[:,8].reshape(m, 1))), corteMu(muPort(port(
#print(tauMu(100))
#print(B/A)
```

Cálculo de los τs correspondientes a los valores de μ escogidos:

In [21]:
```
tausCalculados = [corteMu(mu) for mu in musElegidos]
print("mu's: \t", musElegidos)
print("tau's: \t", tausCalculados)
```

```
mu's:    [0.0075, 0.015, 0.0225, 0.03]
tau's:   [0.04486721780257211, -0.05132227236926794, -0.00359756335347
6338, 0.003280267417311185]
```

Calculo de la frontera eficiente para una región buena

In [22]:
```
muMax = 2.5*np.max(retProm) # Mu maximo para el cual se calcula la fron
sigMax =  np.sqrt((A*muMax**2 - 2*muMax*B + C)/D) #Limite derecho del e

mus = np.linspace(-0.0025, muMax, 200)
stdFront = np.sqrt((A*mus**2 - 2*mus*B + C)/D)
```

PRIMER PORTAFOLIO EFICIENTE

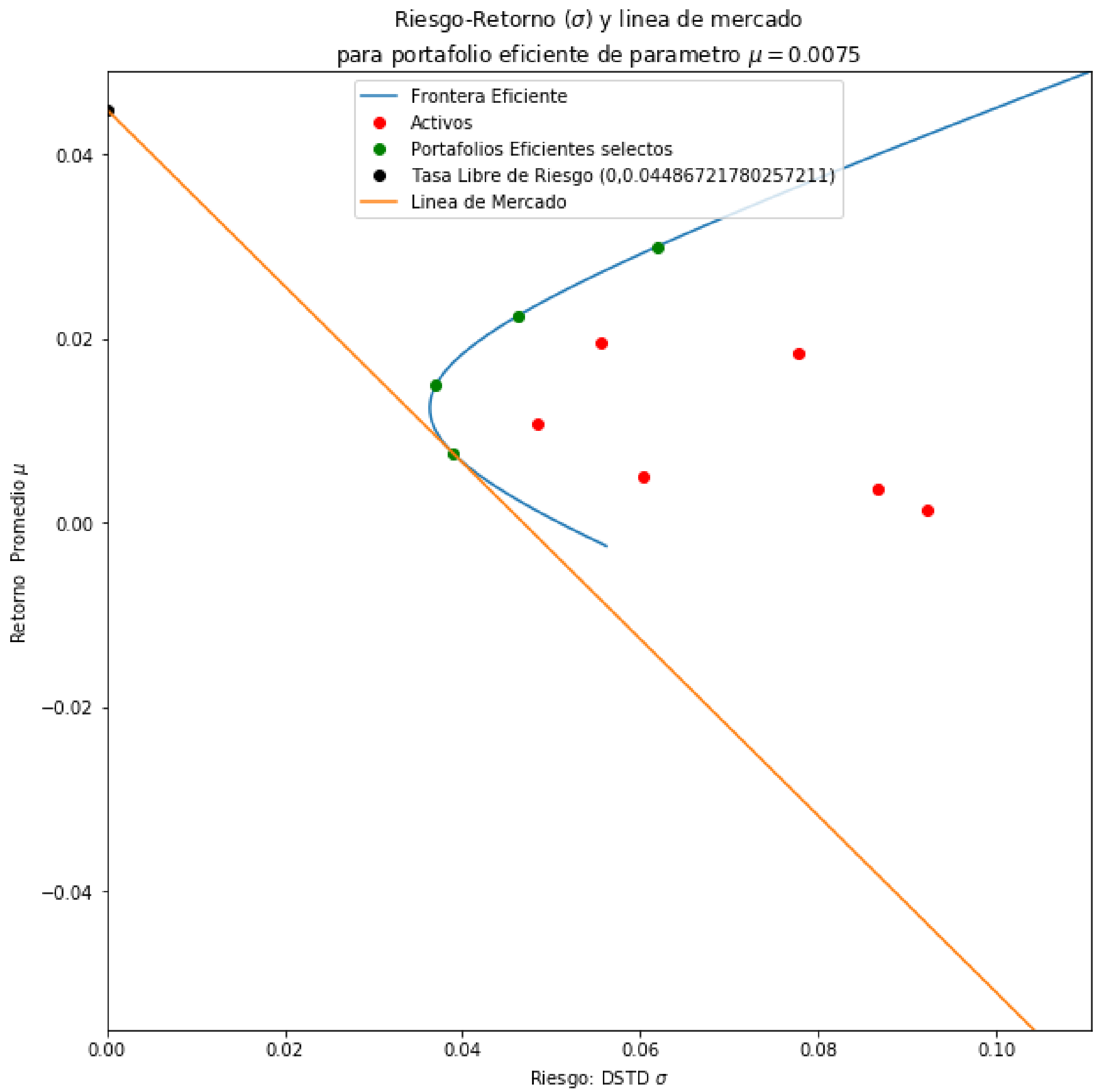
In [23]:

```
    # PRIMER PORTAFOLIO

#Indice para el portafolio actual
# UNICO PARAMETRO QUE SE CAMBIA PARA LAS GRAFICAS SIGUIENTES
i = 0

######################################################
mu = musElegidos[i]
port = portElegidos[i]
var = varPort(port)
std = np.sqrt(var)
tau = corteMu(mu)

# Pendiente de la linea de mercado de capital
m = (mu - tau)/(std-0)


print("El portafolio eficiente para el cual se está graficando su línea
      xOptMu(musElegidos[i]))
print("El cual tiene un rendimiento promedio de: ", mu,
      "una varianza de", var,
      "y le corresonde una tasa libre de riesgo de", tau)


plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma$) y linea de mercado \npara portafol

plt.xlabel("Riesgo: DSTD $\sigma$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = sigMax)
plt.ylim(bottom = -0.055, top = muMax)

# Frontera eficiente
plt.plot(stdFront, mus)

# Activos elegidos
plt.plot(np.sqrt(varianzas), retProm, 'ro')

# 4 Portafolios Eficientes
plt.plot(np.sqrt(varElegidos), musElegidos, 'go')

# LINEA DE MERCADO Y CORTE CON EL EJE DE RENDIMIENTOS: TASA LIBRE DE RI
plt.plot([0], [tau], 'ko')
xs = np.linspace(0, sigMax, 100)
ys = m*xs + tau
plt.plot(xs, ys)

plt.legend(("Frontera Eficiente", "Activos", "Portafolios Eficientes se
            "Tasa Libre de Riesgo (0," + str(tau)+")", "Linea de Mercad
```

```
El portafolio eficiente para el cual se está graficando su línea de me
rcado y tasa libre de riesgo es:
 [[ 0.035061]
 [ 0.457911]
```

```
 [ 0.219239]
 [-0.088331]
 [ 0.118258]
 [ 0.257863]]
El cual tiene un rendimiento promedio de:  0.0075 una varianza de 0.00
1521750901136676 y le corresonde una tasa libre de riesgo de 0.0448672
1780257211
```

Out[23]: `<matplotlib.legend.Legend at 0x7fee184a7e48>`



El portafolio optimo que se halla con este valor de μ no es deseable, pues hay portafolios con mayores rendimientos y menores riesgos disponibles. Por lo tanto, la línea de mercado hallada no es útil.

SEGUNDO PORTAFOLIO

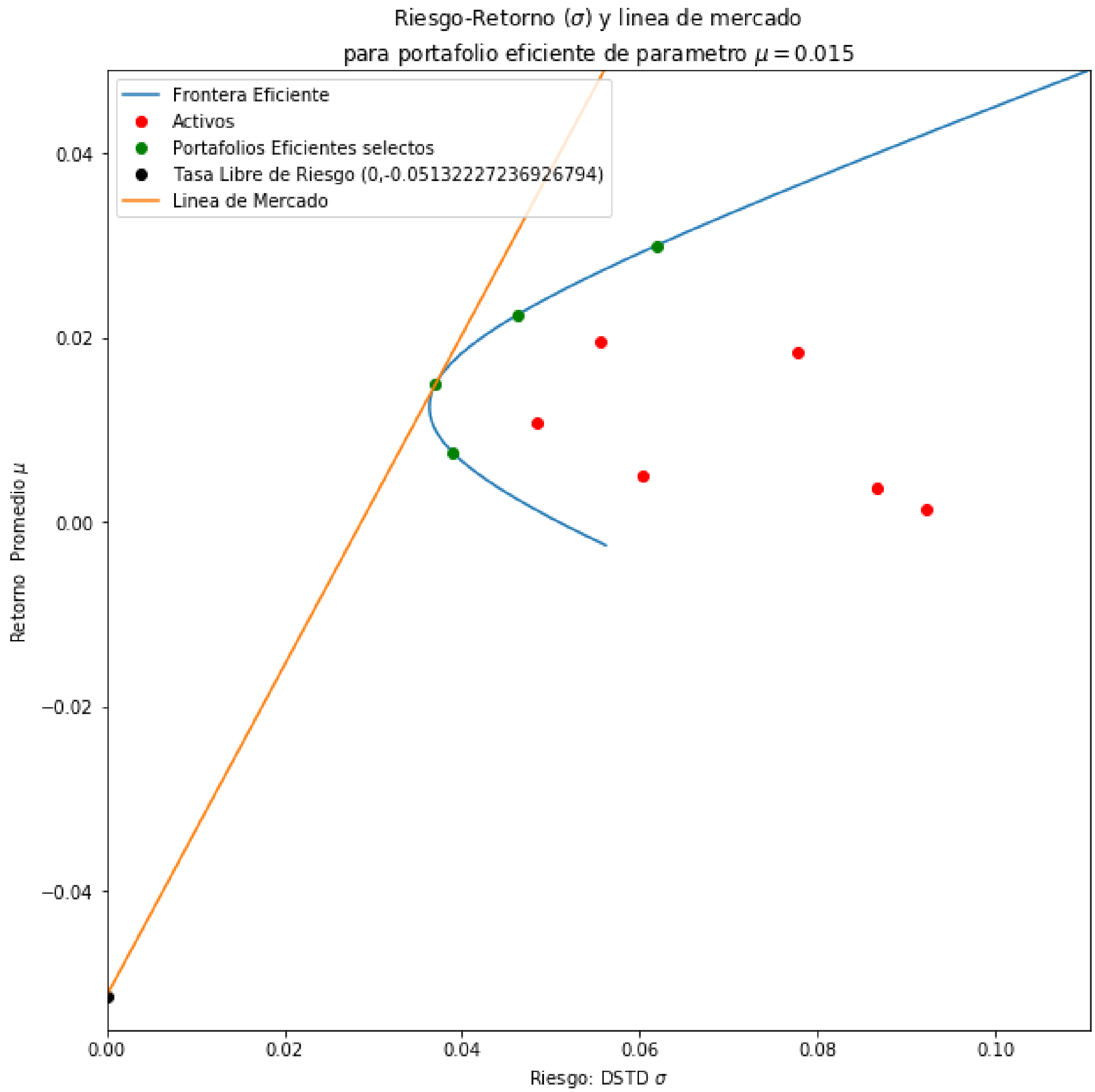
In [24]:

```
    # SEGUNDO PORTAFOLIO

#Indice para el portafolio actual
# UNICO PARAMETRO QUE SE CAMBIA PARA LAS GRAFICAS SIGUIENTES
i = 1

######################################################
mu = musElegidos[i]
port = portElegidos[i]
var = varPort(port)
std = np.sqrt(var)
tau = corteMu(mu)

# Pendiente de la linea de mercado de capital
m = (mu - tau)/(std-0)


print("El portafolio eficiente para el cual se está graficando su línea
      xOptMu(musElegidos[i]))
print("El cual tiene un rendimiento promedio de: ", mu,
      "una varianza de", var,
      "y le corresonde una tasa libre de riesgo de", tau)


plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma$) y linea de mercado \npara portafol

plt.xlabel("Riesgo: DSTD $\sigma$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = sigMax)
plt.ylim(bottom = -0.055, top = muMax)

# Frontera eficiente
plt.plot(stdFront, mus)

# Activos elegidos
plt.plot(np.sqrt(varianzas), retProm, 'ro')

# 4 Portafolios Eficientes
plt.plot(np.sqrt(varElegidos), musElegidos, 'go')

# LINEA DE MERCADO Y CORTE CON EL EJE DE RENDIMIENTOS: TASA LIBRE DE RI
plt.plot([0], [tau], 'ko')
xs = np.linspace(0, sigMax, 100)
ys = m*xs + tau
plt.plot(xs, ys)

plt.legend(("Frontera Eficiente", "Activos", "Portafolios Eficientes se
            "Tasa Libre de Riesgo (0," + str(tau)+")", "Linea de Mercado
```

```
El portafolio eficiente para el cual se está graficando su línea de m
ercado y tasa libre de riesgo es:
 [[-0.082058]
 [ 0.248468]
 [ 0.029897]
```

```
 [ 0.390083]
 [ 0.212156]
 [ 0.201454]]
El cual tiene un rendimiento promedio de:  0.015 una varianza de 0.00
13714032465502388 y le corresonde una tasa libre de riesgo de -0.0513
2227236926794
```

Out[24]:
```
<matplotlib.legend.Legend at 0x7fee18247f98>
```



TERCER PORTAFOLIO

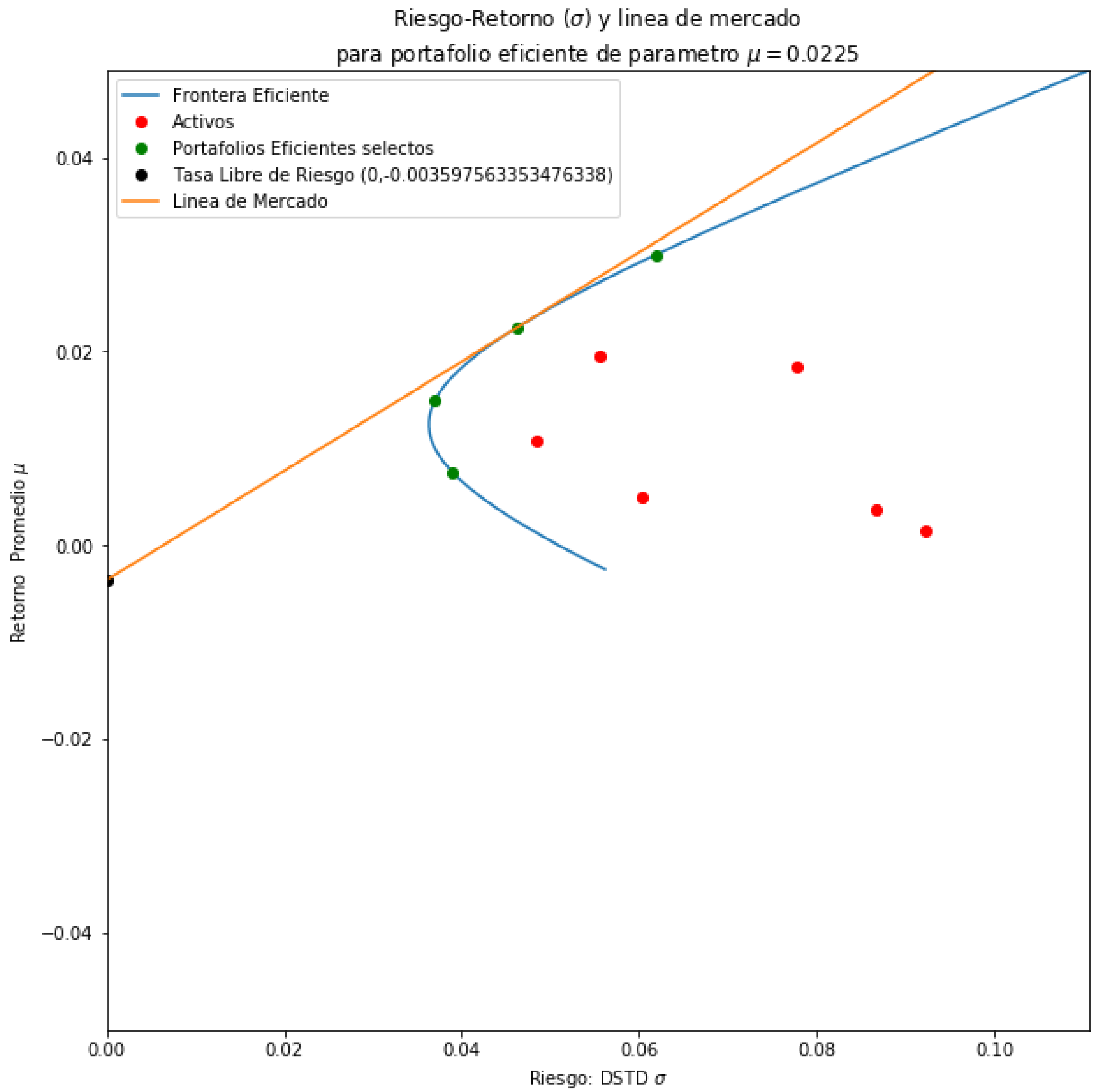
In [25]:

```
    # TERCER PORTAFOLIO

#Indice para el portafolio actual
# UNICO PARAMETRO QUE SE CAMBIA PARA LAS GRAFICAS SIGUIENTES
i = 2

######################################################
mu = musElegidos[i]
port = portElegidos[i]
var = varPort(port)
std = np.sqrt(var)
tau = corteMu(mu)

# Pendiente de la linea de mercado de capital
m = (mu - tau)/(std-0)


print("El portafolio eficiente para el cual se está graficando su línea
      xOptMu(musElegidos[i]))
print("El cual tiene un rendimiento promedio de: ", mu,
      "una varianza de", var,
      "y le corresonde una tasa libre de riesgo de", tau)


plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma$) y linea de mercado \npara portafol

plt.xlabel("Riesgo: DSTD $\sigma$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = sigMax)
plt.ylim(bottom = -0.05, top = muMax)

# Frontera eficiente
plt.plot(stdFront, mus)

# Activos elegidos
plt.plot(np.sqrt(varianzas), retProm, 'ro')

# 4 Portafolios Eficientes
plt.plot(np.sqrt(varElegidos), musElegidos, 'go')

# LINEA DE MERCADO Y CORTE CON EL EJE DE RENDIMIENTOS: TASA LIBRE DE RI
plt.plot([0], [tau], 'ko')
xs = np.linspace(0, sigMax, 100)
ys = m*xs + tau
plt.plot(xs, ys)

plt.legend(("Frontera Eficiente", "Activos", "Portafolios Eficientes se
            "Tasa Libre de Riesgo (0," + str(tau)+")", "Linea de Mercado
```

```
El portafolio eficiente para el cual se está graficando su línea de m
ercado y tasa libre de riesgo es:
 [[-0.199176]
 [ 0.039026]
```

```
 [-0.159444]
 [ 0.868497]
 [ 0.306053]
 [ 0.145044]]
El cual tiene un rendimiento promedio de:  0.0225 una varianza de 0.0
021420869651279212 y le corresonde una tasa libre de riesgo de -0.003
597563353476338
```

Out[25]: <matplotlib.legend.Legend at 0x7fee17c62eb8>



CUARTO PORTAFOLIO

In [26]:

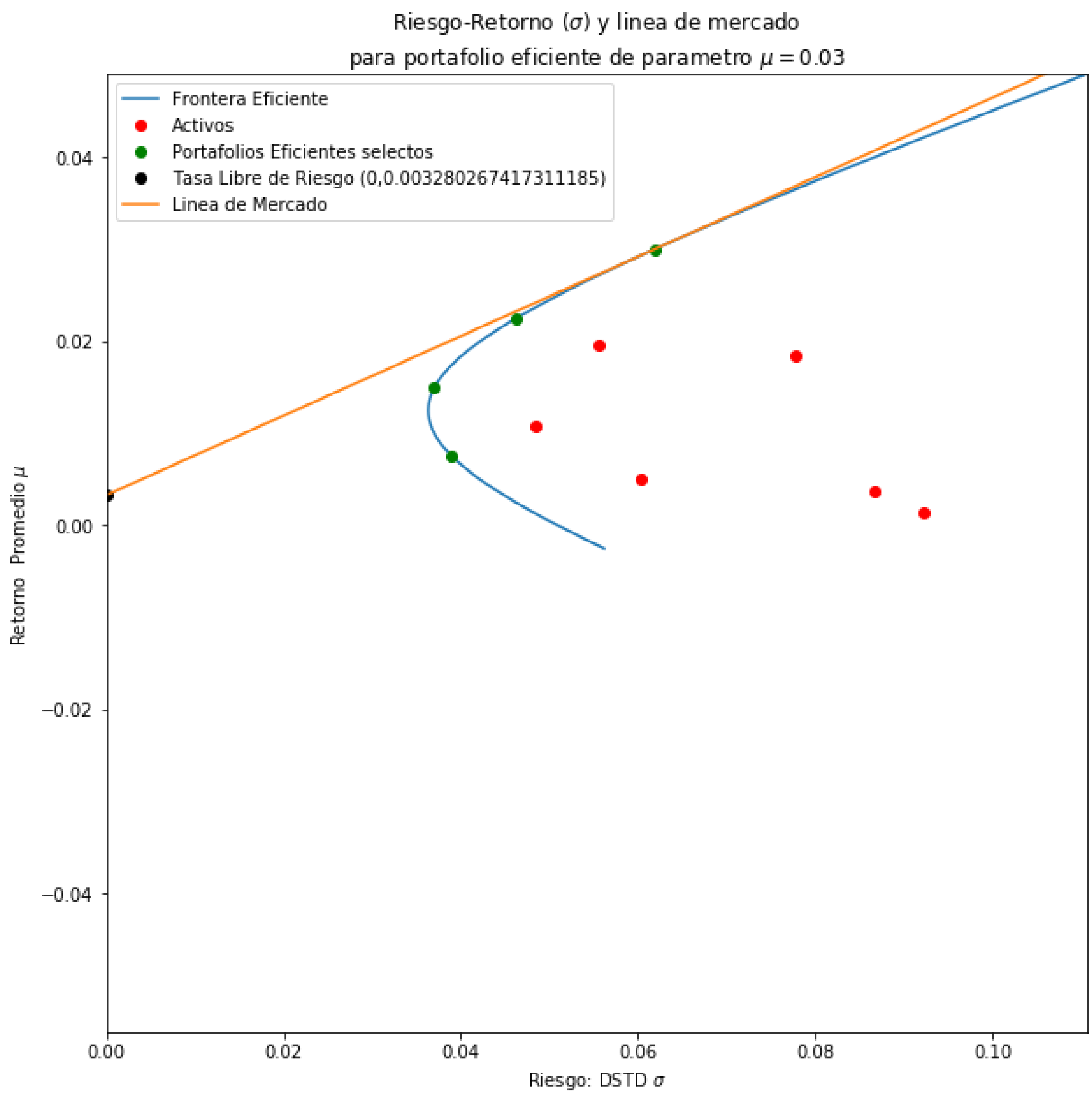
```
    # CUARTO PORTAFOLIO

#Indice para el portafolio actual
# UNICO PARAMETRO QUE SE CAMBIA PARA LAS GRAFICAS SIGUIENTES
i = 3

######################################################
mu = musElegidos[i]
port = portElegidos[i]
var = varPort(port)
std = np.sqrt(var)
tau = corteMu(mu)

# Pendiente de la linea de mercado de capital
m = (mu - tau)/(std-0)


print("El portafolio eficiente para el cual se está graficando su línea
      xOptMu(musElegidos[i]))
print("El cual tiene un rendimiento promedio de: ", mu,
      "una varianza de", var,
      "y le corresonde una tasa libre de riesgo de", tau)


plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma$) y linea de mercado \npara portafol

plt.xlabel("Riesgo: DSTD $\sigma$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = sigMax)
plt.ylim(bottom = -0.055, top = muMax)

# Frontera eficiente
plt.plot(stdFront, mus)

# Activos elegidos
plt.plot(np.sqrt(varianzas), retProm, 'ro')

# 4 Portafolios Eficientes
plt.plot(np.sqrt(varElegidos), musElegidos, 'go')

# LINEA DE MERCADO Y CORTE CON EL EJE DE RENDIMIENTOS: TASA LIBRE DE RI
plt.plot([0], [tau], 'ko')
xs = np.linspace(0, sigMax, 100)
ys = m*xs + tau
plt.plot(xs, ys)

plt.legend(("Frontera Eficiente", "Activos", "Portafolios Eficientes se
            "Tasa Libre de Riesgo (0," + str(tau)+")", "Linea de Mercad
```

```
El portafolio eficiente para el cual se está graficando su línea de m
ercado y tasa libre de riesgo es:
 [[-0.316295]
 [-0.170417]
 [-0.348785]
```

```
 [ 1.346911]
 [ 0.399951]
 [ 0.088635]]
El cual tiene un rendimiento promedio de:  0.03 una varianza de 0.003
8338020568697267 y le corresonde una tasa libre de riesgo de 0.003280
267417311185
```

Out[26]:

```
<matplotlib.legend.Legend at 0x7fee17a0a588>
```



8

Si la ecuación de la asíntota es $\sigma = m'\mu + b'$, hallando el $lim_{\mu\to\infty} \frac{\sigma_{x*}(\mu)}{m\mu' + b'}$ concluimos que $m' = \sqrt{A/D}$. En ese caso $b' = lim_{\mu\to\infty}\left(\sigma_{x*}(\mu) - \sqrt{A/D}\mu\right) = -\frac{B}{\sqrt{AD}}$.

Despejando μ como funcion de σ, la recta queda reescrita como $\mu = \sqrt{\frac{D}{A}}\sigma + \frac{B}{A}$, es decir que tiene pendiente $m = \sqrt{\frac{D}{A}}$ e intercepto $\tau^* = \frac{B}{A}$

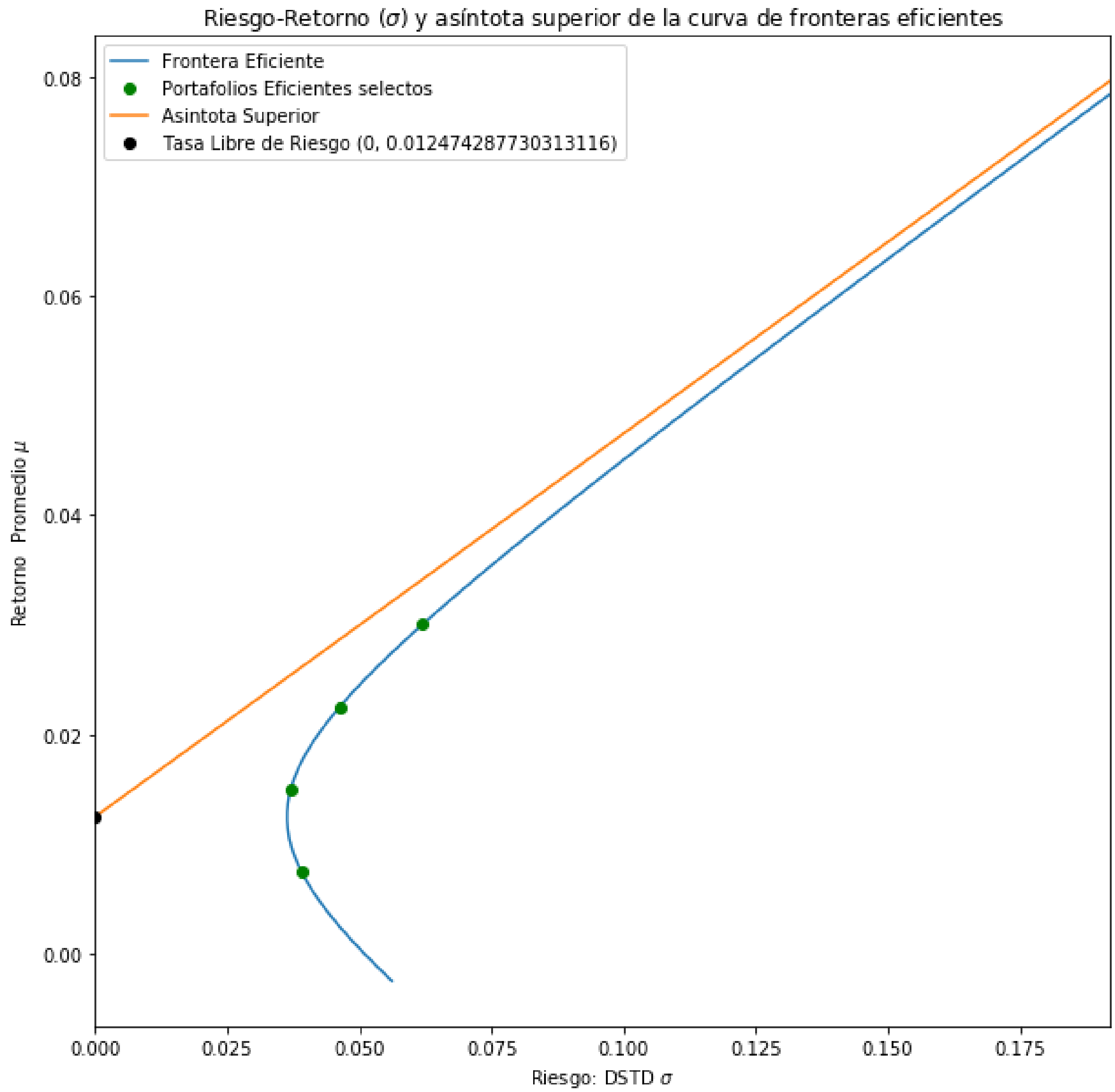
In [27]:
```
muMax = 4*np.max(retProm) # Mu maximo para el cual se calcula la fronte
sigMax =  np.sqrt((A*muMax**2 - 2*muMax*B + C)/D) #Limite derecho del e

mus = np.linspace(-0.0025, muMax, 200)
stdFront = np.sqrt((A*mus**2 - 2*mus*B + C)/D)
```

In [28]:
```
m = np.sqrt(D/A)
tauOpt = B/A

print("El corte de la asintota con el eje de rendimientos, tau* =", tau
      "se interpreta como la tasa libre de riesga máxima que es compati
      encontrado. La asintota, entonces, representa un límite a las pos
      y riesgos que son razonables en combinacion con 'el portafolio ef
```

```
El corte de la asintota con el eje de rendimientos, tau* = 0.012474287
730313116 se interpreta como la tasa libre de riesga máxima que es com
patible con algún portafolio eficiente       encontrado. La asintota, 
entonces, representa un límite a las posibles combinaciones de rendimi
entos       y riesgos que son razonables en combinacion con 'el portaf
olio eficiente' de rendimiento y riesgo infinito.
```

In [29]:
```
plt.figure(figsize = (10,10))
plt.title("Riesgo-Retorno ($\sigma$) y asíntota superior de la curva de

plt.xlabel("Riesgo: DSTD $\sigma$")
plt.ylabel("Retorno  Promedio $\mu$")

plt.xlim(left = 0, right = sigMax)
#plt.ylim(bottom = -0.055, top = muMax)

# Frontera eficiente
plt.plot(stdFront, mus)

# 4 Portafolios Eficientes
plt.plot(np.sqrt(varElegidos), musElegidos, 'go')


# ASINTOTA SUPERIOR
xs = np.linspace(0, sigMax, 100)
ys = m*xs + tauOpt
plt.plot(xs, ys)

plt.plot([0], [tauOpt], 'ko')

#

plt.legend(("Frontera Eficiente", "Portafolios Eficientes selectos",
            "Asintota Superior", "Tasa Libre de Riesgo (0, " + str(tauO
```

Out[29]:
```
<matplotlib.legend.Legend at 0x7fee1781ad30>
```



9

La frontera eficiente esta descrita por la fórmula $\sigma^2(\mu) = \frac{A\mu^2 - 2B\mu + C}{D}$, la cual, al ser derivada e igualada a 0 respecto a μ nos dice que el valor mínimo de riesgo (ya sea varianza o desviacion estándar) se encuentra cuando $\mu^{**} = \frac{B}{A} = \tau^*$, para el cual la varianza del portafolio correspondiente será $\sigma^2_{**} = \frac{1}{A}$ y el portafolio estará dado, entonces, por $x^{**} = \frac{1}{A} S^{-1} \hat{u}$. En nuestro caso:

In [30]:
```
muOpt = B/A
varOpt = 1/A
sigOpt = np.sqrt(varOpt)
portOpt = xOptMu(muOpt)

print("Rentabilidad Óptima:\t", muOpt)
print("Varianza Óptima:\t", varOpt)
print("Desviación estandar Óptima:\t", sigOpt)
print("Portafolio Óptimo:\n", portOpt)
```

```
Rentabilidad Óptima:	 0.012474287730313116
Varianza Óptima:	 0.0013191768996118408
Desviación estandar Óptima:	 0.036320474936485075
Portafolio Óptimo:
 [[-0.042617]
 [ 0.319   ]
 [ 0.09366 ]
 [ 0.228971]
 [ 0.180534]
 [ 0.22045 ]]
```

10

Usando las fórmulas que definen a los valores

Para hallar el rendimiento promedio el portafolio x usamos $\bar{r}_x = \bar{r}^T \cdot x$ (función muPort).

Para hallar su varianza usamos $\sigma_x = x^T S x$ (función varPort).

La tasa libre de riesgo la hallamos como el punto de corte la línea de mercado de capital, es decir aquella línea tangente a la frontera eficiente que pasa por el portafolio, la cual vimos que tenía fórmula $\tau = \frac{2A(\mu^*)^2 - 2B\mu^* - 2\sigma^2 D}{2A\mu^* - 2B}$ (función corteMu).

In [31]:
```
datos = []

for i in range(len(musElegidos)):
    mu = muPort(portElegidos[i])
    var = varPort(portElegidos[i])
    std = np.sqrt(var)
    tau = corteMu(mu)
    datos.append([mu, var, std, tau])

print("TABLA CALCULADA CON FÓRMULAS DE DEFINICIÓN PARA LOS 4 PORTAFOLIOS
print(tabulate(datos, ["Rentabilidad Promedio", "Varianza", "Desv. Std.
```

```
TABLA CALCULADA CON FÓRMULAS DE DEFINICIÓN PARA LOS 4 PORTAFOLIOS EFIC
IENTES ELEGIDOS

  Rentabilidad Promedio    Varianza    Desv. Std.    Tasa L.R compatib
le
-----------------------  ----------  ------------  -------------------
--
                 0.0075  0.00152175     0.0390096             0.044867
2
                 0.015   0.0013714      0.0370325            -0.051322
3
                 0.0225  0.00214209     0.0462827            -0.003597
56
                 0.03    0.0038338      0.0619177             0.003280
27
```

Usando las fórmulas halladas al optimizar respecto a μ

Al resolver el problema de optimización de varianza de los portafolios dada la rentabilidad promedio μ, se deduce que su varianza está dada por la fórmula $\sigma^2(\mu) = \frac{A\mu^2 - 2B\mu + C}{D}$ (funcion varMu).

Utilizamos la misma formula de τ que en la tabla anterior.

In [32]:
```
def varMu(mu):
    return (A*mu**2 - 2*B*mu + C)/(D)

datos = []

for i in range(len(musElegidos)):
    mu = musElegidos[i]
    var = varMu(mu)
    std = np.sqrt(var)
    tau = corteMu(mu)
    datos.append([mu, var, std, tau])

print("TABLA CALCULADA CON FÓRMULAS DEL PRIMER PROBLEMA DE OPTIMIZACION
print(tabulate(datos, ["Rentabilidad Promedio", "Varianza", "Desv. Std.
```

```
TABLA CALCULADA CON FÓRMULAS DEL PRIMER PROBLEMA DE OPTIMIZACION PARA
LOS 4 PORTAFOLIOS EFICIENTES ELEGIDOS

  Rentabilidad Promedio    Varianza    Desv. Std.    Tasa L.R compatib
le
-----------------------  ----------  ------------  -------------------
--
                 0.0075  0.00152175     0.0390096             0.044867
2
                 0.015   0.0013714      0.0370325            -0.051322
3
                 0.0225  0.00214209     0.0462827            -0.003597
56
                 0.03    0.0038338      0.0619177             0.003280
27
```

Usando las fórmulas que se deducen al resolver el problema de optimización respecto a τ

Ahora usamos las fórmulas halladas de la solución al problema de optimización convexo de la funcion $Max\frac{\bar{r}_x-\tau}{\sigma_x}$ para un parámetro τ bajo la condición $x \cdot \hat{u} = 1$.

La rentabilidad promedio viene dada por la fórmula $\mu(\tau) = \frac{C-\tau B}{B-\tau A}$ (funcion muTau).

La varianza del portafolio eficiente hallado viene dado por la fórmula $\sigma^2(\tau) = \frac{\mu(\tau)-\tau}{B-\tau A}$ (funcion varTau).

In [33]:
```
datos = []

def tauMu(mu):
    return (mu*B - C)/(mu*A - B)

def muTau(tau):
    return (C - tau*B)/(B - tau*A)

def varTau(tau):
    return (muTau(mu) - tau)/(B - tau * A)

for i in range(len(musElegidos)):
    mu = muTau(tausCalculados[i]) # Taus calculados son los valores cal
    var = varMu(mu)
    std = np.sqrt(var)
    tau = tauMu(mu)
    datos.append([mu, var, std, tau])

print("TABLA CALCULADA CON FÓRMULAS DE DEFINICIÓN PARA LOS 4 PORTAFOLIOS
print(tabulate(datos, ["Rentabilidad Promedio", "Varianza", "Desv. Std.
```

```
TABLA CALCULADA CON FÓRMULAS DE DEFINICIÓN PARA LOS 4 PORTAFOLIOS EFIC
IENTES ELEGIDOS

  Rentabilidad Promedio    Varianza    Desv. Std.    Tasa L.R compatib
le
-----------------------  ----------  ------------  -------------------
--
                 0.0075  0.00152175     0.0390096             0.044867
2
                 0.015   0.0013714      0.0370325            -0.051322
3
                 0.0225  0.00214209     0.0462827            -0.003597
56
                 0.03    0.0038338      0.0619177             0.003280
27
```

Al comparar las 3 tablas, vemos que son en efecto todos los valores coinciden, a pesar de haberse usado fórmulas distintas.

11

Esogemos el portafolio con $\mu = 0.03$, para el cual $\tau = 0.00328027$ y $\sigma^2 = 0.0038338$

In [34]:
```
portPM = portElegidos[3]
```

Podemos o no usar el hecho de que $cov(r_x, r_y) = x^T S y$ para definir la función covarianza.

Funciones ayudantes:

In [35]:
```
def covarianza(portX, portY):
    #cov = 0
    # Calculo de la rentabilidad promedio de los portafolios
    #muX = muPort(portX)
    #print(muX)
    #muY = muPort(portY)
    #print(muY)
   #
    #for k in range(M):
    #    cov += (portX[k]  - muX)*(portY[k] - muY)
    #
    #return cov[0]/M
    return portX.T.dot(S).dot(portY)

# Test para comprobar que la funcion de covarianza funciona
#print(covarianza(portElegidos[1], portElegidos[1]))

def beta1(portX, portPM):
    return  (covarianza(portX, portPM)/covarianza(portPM, portPM))[0,0]

def beta2(portX, portPM):
    muX = muPort(portX)
    muPM = muPort(portPM)
    tau = tauMu(muPM)
    return  (muX - tau)/(muPM - tau)
```

Representamos los activos como portafolios:

In [36]:
```
# Matriz cuya entrada k sera el portafolio correspondiente al activo k
portActivos = np.zeros((M, M, 1))

for k in range(M):
    # El activo k se representa como el portafolio que tiene un 1 en la
    portActivos[k, k] = 1

print(portActivos[0])
print(portPM)
```

```
[[1.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
[[-0.316295]
 [-0.170417]
 [-0.348785]
 [ 1.346911]
 [ 0.399951]
 [ 0.088635]]
```

Tabulamos el resultado de las 2 formulas para beta:

In [37]:
```python
datos = []

betas1 = []
betas2 = []
for k in range(M):
    betas1.append(beta1(portActivos[k], portPM))
    betas2.append(beta2(portActivos[k], portPM))

# Para tabular
datos.append(betas1)
datos.append(betas2)

print("BETAS DE CADA ACTIVO RESPECTO A PORTAFOLIO PM ESCOGIDO, SEGUN LAS
print(tabulate(datos,["Activo "+ str(k) for k in range(1, M+1)]))
```

```
BETAS DE CADA ACTIVO RESPECTO A PORTAFOLIO PM ESCOGIDO, SEGUN LAS 2 FO
RMULA
  Activo 1    Activo 2    Activo 3    Activo 4    Activo 5    Activo 6
----------  ----------  ----------  ----------  ----------  ----------
-0.0672316    0.283891   0.0649758    0.610742    0.565275   0.0121788
-0.0672316    0.283891   0.0649758    0.610742    0.565275   0.0121788
```

Como los 2 renglones son iguales, concluimos que las 2 formulas son equivalentes.

Referencias

- https://finance.yahoo.com/quote/CIB/history?period1=1505883600&period2=1568955600&interval=1mo&filter=history&frequency=1mo (https://finance.yahoo.com/quote/CIB/history?period1=1505883600&period2=1568955600&interval=1mo&filter=history&frequency=1mo)
- https://finance.yahoo.com/quote/AVH/history?period1=1505883600&period2=1568955600&interval=1mo&filter=history&frequency=1mo (https://finance.yahoo.com/quote/AVH/history?period1=1505883600&period2=1568955600&interval=1mo&filter=history&frequency=1mo)
- https://finance.yahoo.com/quote/CIB/history?p=CIB&.tsrc=fin-srch (https://finance.yahoo.com/quote/CIB/history?p=CIB&.tsrc=fin-srch)
- https://es.investing.com/equities/exito-historical-data (https://es.investing.com/equities/exito-historical-data)
- https://es.investing.com/equities/grupoargos-historical-data (https://es.investing.com/equities/grupoargos-historical-data)
- https://es.investing.com/indices/ftse-colombia-historical-data (https://es.investing.com/indices/ftse-colombia-historical-data)
- https://es.investing.com/equities/carton-de-colombia-sa-historical-data (https://es.investing.com/equities/carton-de-colombia-sa-historical-data)
- https://www.investing.com/equities/mineros-sa-historical-data (https://www.investing.com/equities/mineros-sa-historical-data)