In [330]:
```
pip install yfinance
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.7/d
ist-packages (0.1.63)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/pytho
n3.7/dist-packages (from yfinance) (2.23.0)
Requirement already satisfied: lxml>=4.5.1 in /usr/local/lib/python3.
7/dist-packages (from yfinance) (4.6.3)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python
3.7/dist-packages (from yfinance) (1.1.5)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.
7/dist-packages (from yfinance) (1.19.5)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/
python3.7/dist-packages (from yfinance) (0.0.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/l
ib/python3.7/dist-packages (from pandas>=0.24->yfinance) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python
3.7/dist-packages (from pandas>=0.24->yfinance) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/d
ist-packages (from python-dateutil>=2.7.3->pandas>=0.24->yfinance)
(1.15.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python
3.7/dist-packages (from requests>=2.20->yfinance) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/p
ython3.7/dist-packages (from requests>=2.20->yfinance) (2021.5.30)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/py
thon3.7/dist-packages (from requests>=2.20->yfinance) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.
1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.20->yfi
nance) (1.24.3)
```

Librerías

In [369]:
```
import yfinance as yf
import random as rndm
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.stattools import adfuller
from scipy.stats import wishart
import scipy.stats as sp
```

https://github.com/dieraq/-TSE_Tarea_07 (https://github.com/dieraq/-TSE_Tarea_07)

Ejercicio 1

Reproducir la figura del ejercicio 1 utilizando los siguientes parámetros: $q = 0.5, r = 0.5$. Se han usado unidades arbirtrarias de tal manera que $\mathcal{R}^2 = 1$ cuando $\mathcal{G} = 1$.

Solución

Calculamos: </br>

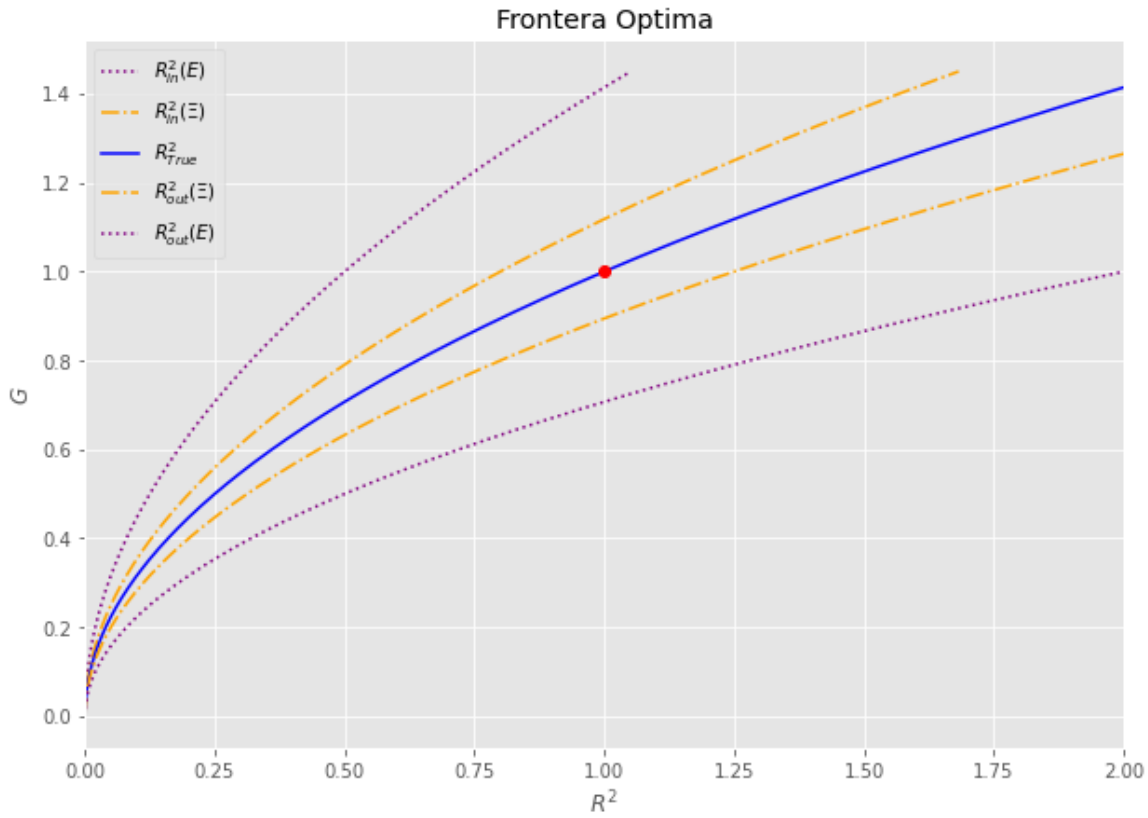
$$\mathcal{R}^2_{true} = \frac{\mathcal{G}^2}{p(1+r)}$$
$$\mathcal{R}^2_{in} = \mathcal{R}^2_{true}(1-q)$$
$$\mathcal{R}^2_{out} = \frac{\mathcal{R}^2_{true}}{1-q}$$
$$\mathcal{R}^2_{in}(\Xi) = \frac{\mathcal{G}^2}{p}\frac{r+q}{(1+r)(r+q(r+1))}$$
$$\mathcal{R}^2_{out}(\Xi) = \frac{\mathcal{G}^2}{p}\frac{r+q+rq}{(r+q)(1+r)}$$

In [332]:
```
G = np.linspace(0,1.45,1000) # Ganancia esperada del portafolio
q = 0.5
r = 0.5
p = 1/(1+r)
# In and Out and true
R_t =  lambda G: (G**2)/(p*(1+r))
R_i = lambda R: R*(1-q)
R_o = lambda R: R/(1-q)
R_true = R_t(G).flatten()
R_in = R_i(R_true).flatten()
R_out = R_o(R_true).flatten()
R_in_or = (G**2/p)*((r+q)/((1+r)*(r+q*(r+1))))
R_out_or = (G**2*(r+q+r*q))/(p*(r+q)*(1+r))
```

In [333]:
```
fig = plt.figure(figsize = (10, 7))
#plt.style.use('ggplot')
plt.plot(R_in, G,ls="--", label ="$R_{in}^2(E)$", c ="purple",linestyl
e='dotted')
plt.plot(R_in_or, G,ls="--", label ="$R_{in}^2(\Xi)$", c ="orange",lin
estyle='dashdot')
plt.plot(R_true, G, label ="$R_{True}^2$", c ="blue",linestyle='solid
')
plt.plot(R_out_or, G, label ="$R_{out}^2(\Xi)$", c ="orange",linestyle
='dashdot')
plt.plot(R_out, G, label ="$R_{out}^2(E)$", c ="purple",linestyle='dot
ted')
plt.plot(1,1, marker="o", color="red")
plt.xlim((0, 2))
plt.title("Frontera Optima")
plt.xlabel("$R^2$")
plt.ylabel("$G$")
plt.legend()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: Matpl
otlibDeprecationWarning: Saw kwargs ['ls', 'linestyle'] which are all
aliases for 'linestyle'.  Kept value from 'linestyle'.  Passing multi
ple aliases for the same property will raise a TypeError in 3.3.
  This is separate from the ipykernel package so we can avoid doing i
mports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: Matpl
otlibDeprecationWarning: Saw kwargs ['ls', 'linestyle'] which are all
aliases for 'linestyle'.  Kept value from 'linestyle'.  Passing multi
ple aliases for the same property will raise a TypeError in 3.3.
  after removing the cwd from sys.path.
```

Out[333]: <matplotlib.legend.Legend at 0x7f76ae6a2910>



Ejercicio 2

La figura de este ejercicio muestra la frontera óptima para el proceso verdadeor, dentro de muestra y fuera de muestra en el caso de una matriz Wishart $W_q(0, \sigma^2 I)$ donde $q = 0.5, p = 100, \sigma^2 = 0.2$ y $g = 1$

Reproduzca la figura del ejercicio 3 y responda las siguientes preguntas:

Solución

Calculamos: </br>

$$\mathcal{R}^2_{true} = \frac{\mathcal{G}^2}{g'\Sigma^{-1}g}$$
$$\mathcal{R}^2_{in} = \frac{\mathcal{G}^2}{g'E^{-1}g}$$
$$\mathcal{R}^2_{out} = \frac{\mathcal{C}^2 g'E^{-1}\tilde{E}E^{-1}g}{g'E^{-1}g}$$

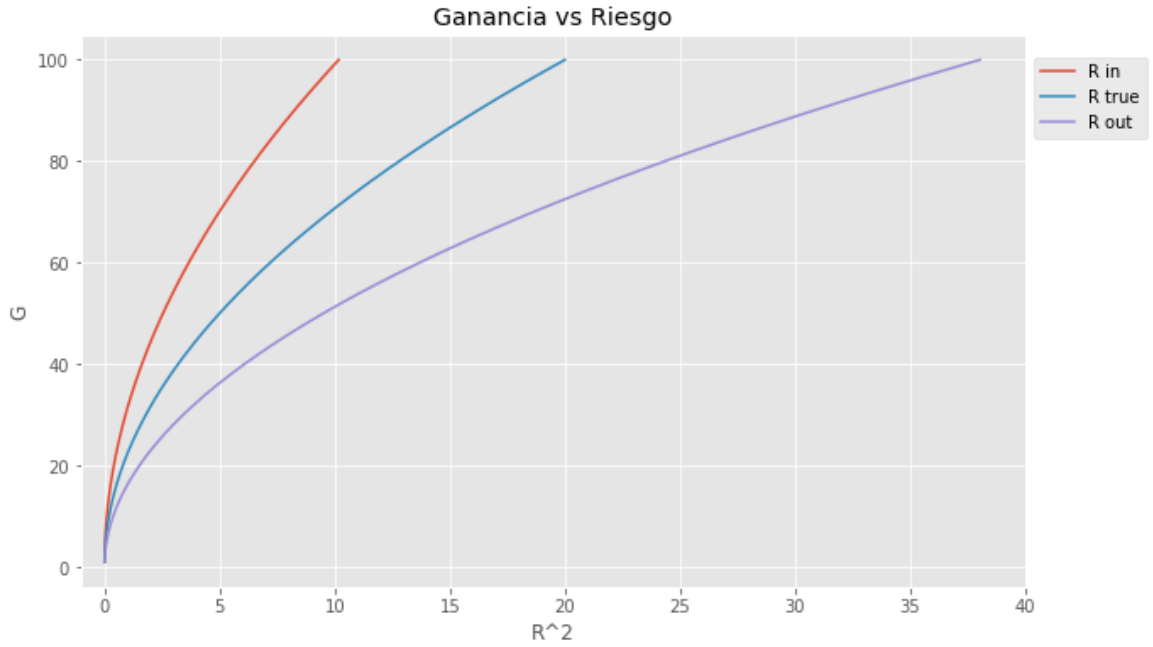
- Con matriz de Covarianzas

In [334]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=200, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[334]: <matplotlib.legend.Legend at 0x7f76ae6ab290>



- Correlaciones

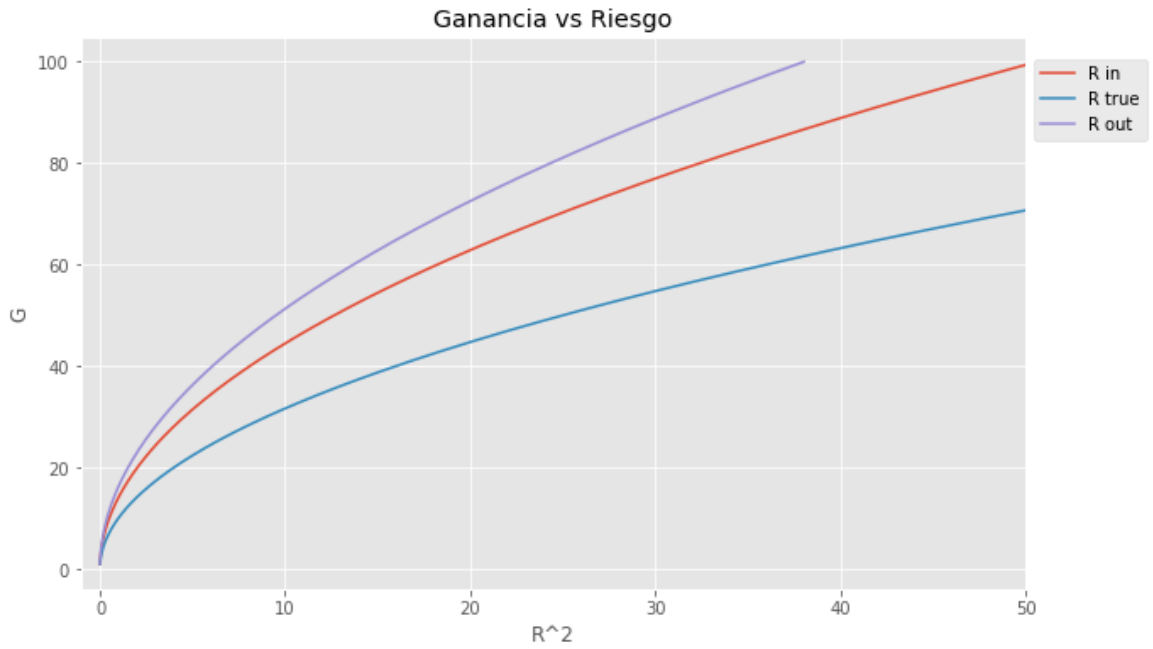
Se sabe que la matriz de correlaciones C está dada por las estradas $c_{ij} = \frac{cov(X_i,X_j)}{\sqrt{var(X_i)var(X_j)}}$. En este caso como $\Sigma_{ii} = 0.2$ y $\Sigma_{ij} = 0$ para toda $i \neq j$, entonces tenemos que $C = \Sigma/\sigma^2$ ya que en este caso en general se cumple que $var(X_i) = var(X_j) = 0.2$.

La implemtación que se muestra a continuación se basa en este hecho sustituyendo Σ por C.

In [335]:
```
""" Correlaciones In-sample  True  Out-sample """
def R_t(G, C, g):
    return G**2 / (g.T@np.linalg.inv(C)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E,C, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@C@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
#np.random.seed(seed=47)
C = S/0.2
E = W/(200*0.2)
G = np.linspace(1,100,200)
""" Correlaciones In-sample  True  Out-sample """
R_true = R_t(G,C,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1,50))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[335]: <matplotlib.legend.Legend at 0x7f76ae4d5150>



Podemos corroborar que tienen comportamientos similares, sin embargo, la matriz de correlaciones sobreestima el riesgo asociado del portafolio a comparación de la matriz de covarianzas.

- ¿Cómo cambia el comportamiento al variar p, n o $q = p/n$?

Notemos que $q <= 1$. Así, aseguramos el marco teórico en el cual nos enfocamos. Por lo anterior, notemos que conforme el número de elementos a analizar p crece, para concordancia en la teoría n, el número de observaciones, debe crecer en alguna proporción con respecto a n.

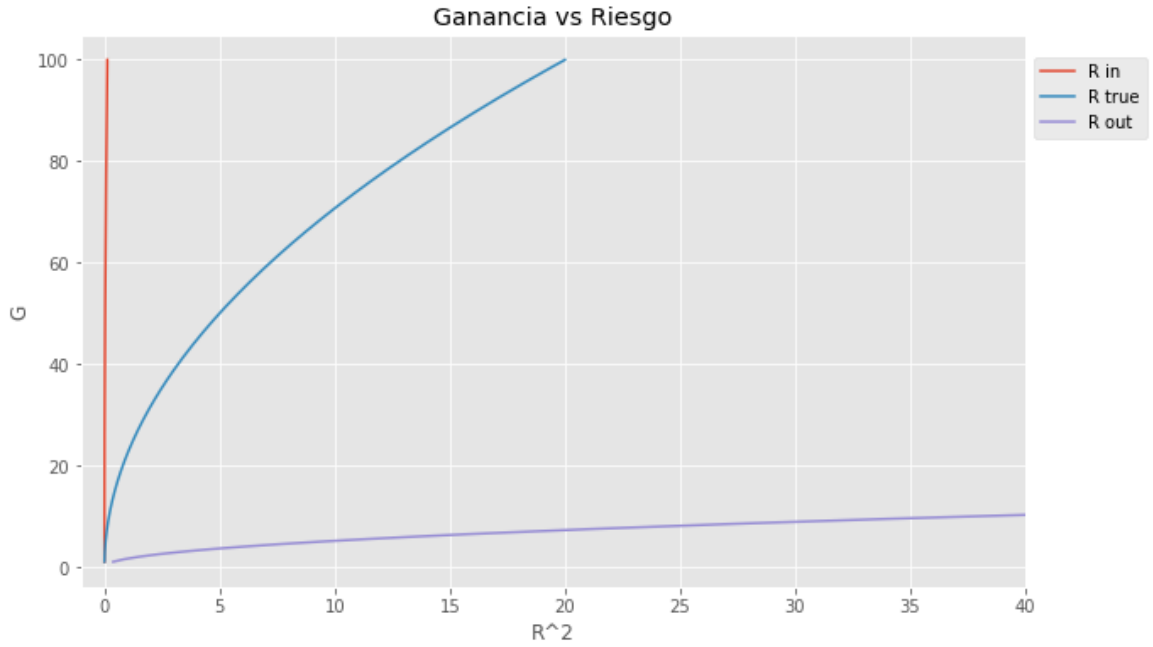
Sin importar la elección de n y p cuando $q = 1$, el sesgo entre los riesgos dentro de muestra y fuera de muestra incrementa de modo que la frontera óptima dentro de muestra diverge y la frontera fuera de muestra se acerca al eje vertical.

In [336]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=100, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[336]: <matplotlib.legend.Legend at 0x7f76ae46bcd0>



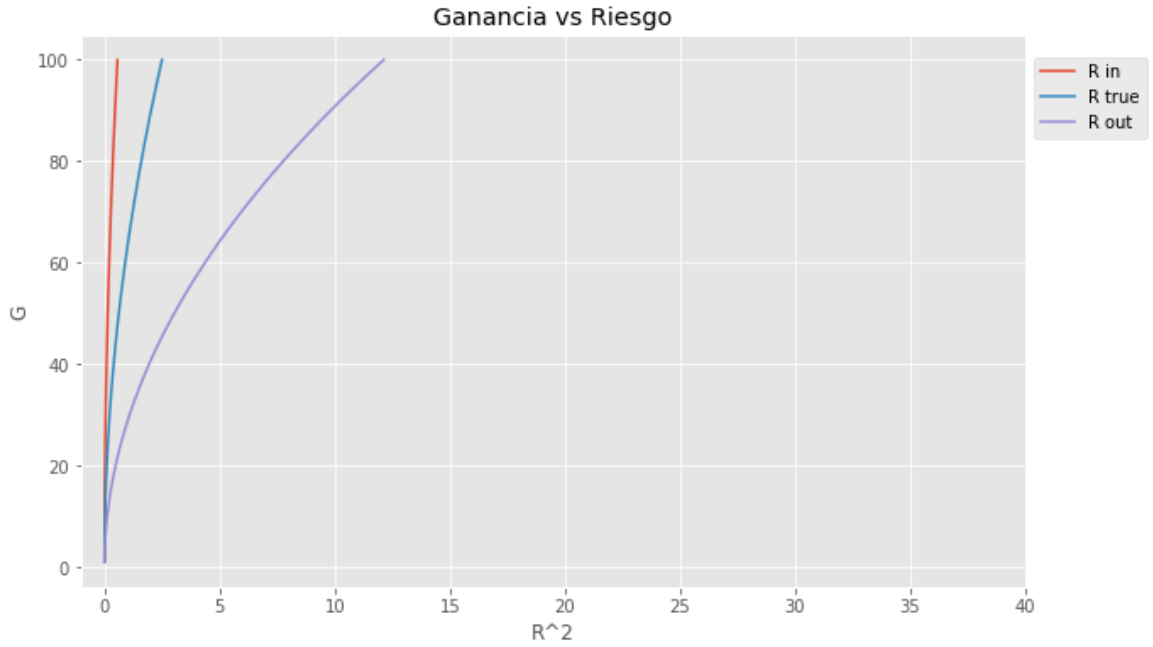
Si q decrece, se infiere que n es mayor que p de modo que la frontera óptima real converge a la dentro de muestra y la fuera de muestra se aleja.

In [337]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 800
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=1000, scale=S, size =1, random_state = None)
E = W/1000

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[337]: <matplotlib.legend.Legend at 0x7f76ae3d6c90>



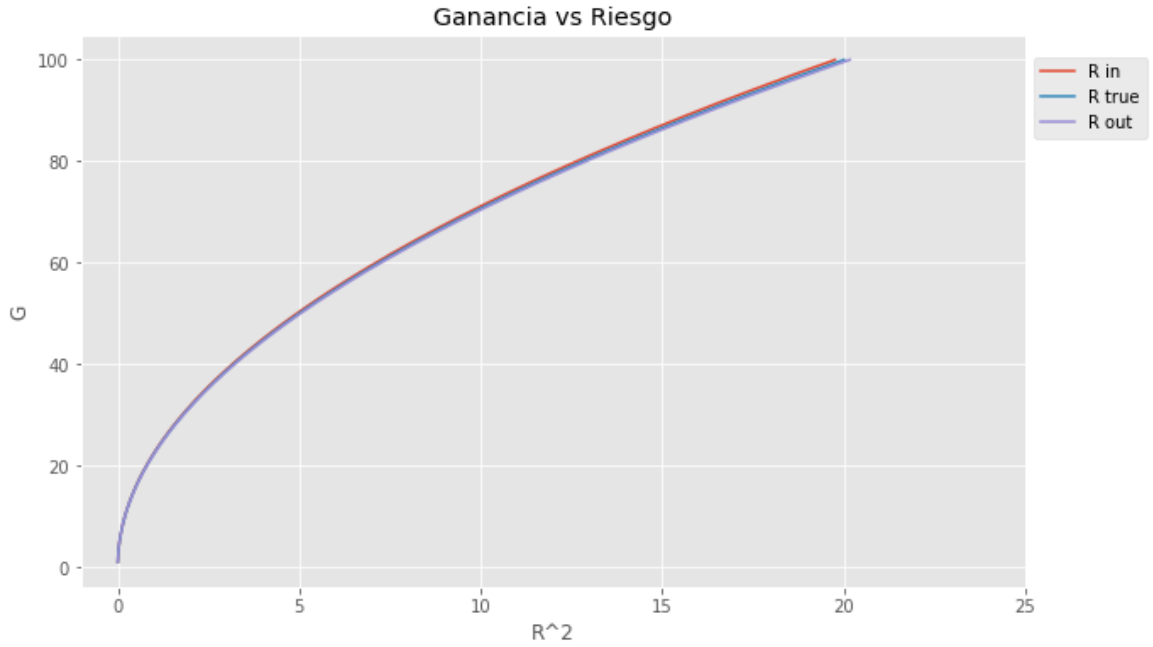
Si n constante y p incrementa de manera que $q \to 1$ se observa que las tres fronteras se acercan más al eje vertical.

In [338]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=10000, scale=S, size =1, random_state = None)
E = W/10000

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 25))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[338]: <matplotlib.legend.Legend at 0x7f76ae35c0d0>



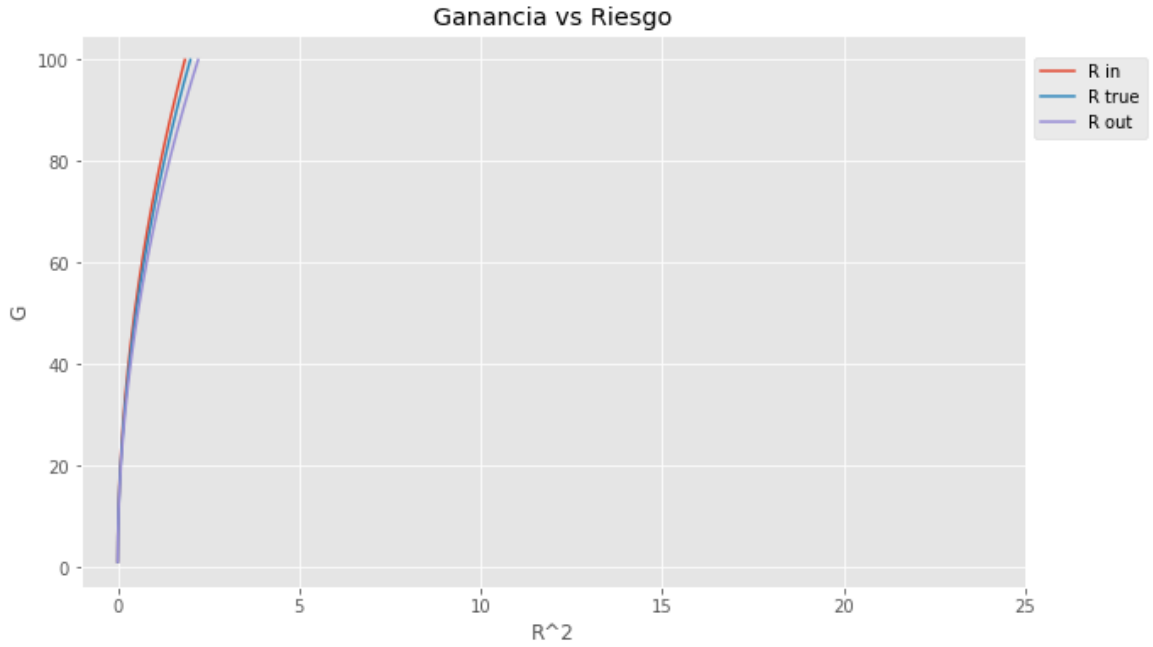
Conforme n se aleja mucho más de p podemos observar que el sesgo decrece, el mismo caso en el que $q \to 0$.

In [339]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 1000
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=10000, scale=S, size =1, random_state = None)
E = W/10000

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 25))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[339]: <matplotlib.legend.Legend at 0x7f76ae2e0590>



Si p incrementa de manera que $p \to \infty$ con $p << n$ entonces se observa que las tres fronteras divergen.

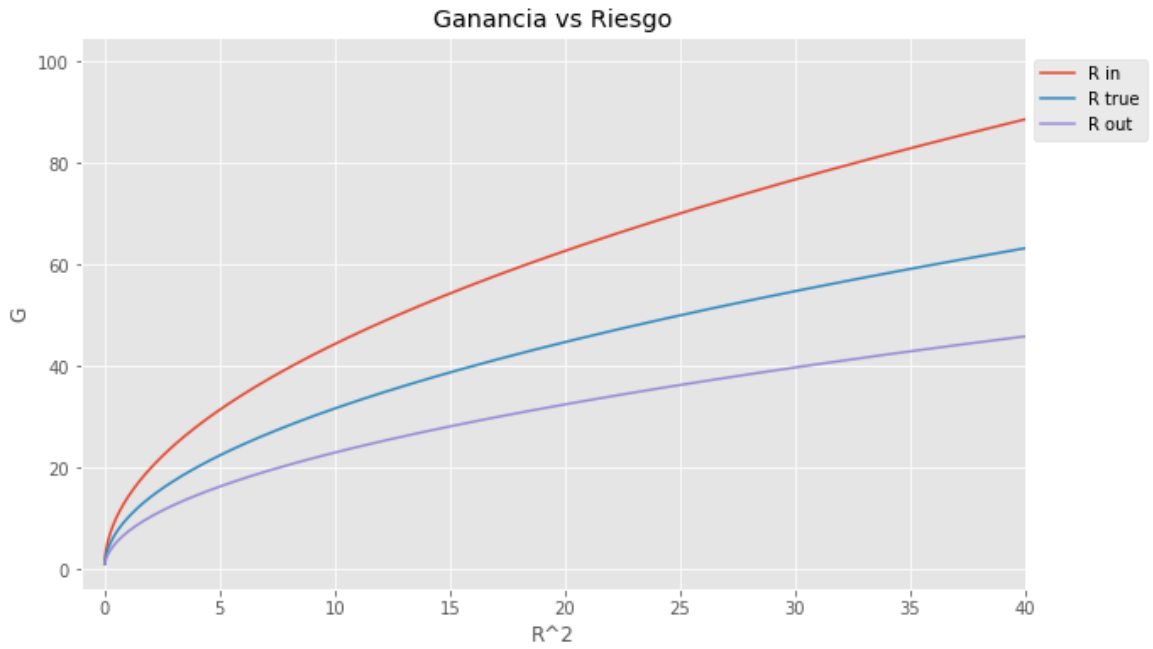
- ¿Cómo cambia el comportamiento de los resultados al variar el valor de σ^2?

In [340]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*1
W = wishart.rvs(df=200, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[340]: <matplotlib.legend.Legend at 0x7f76af3698d0>

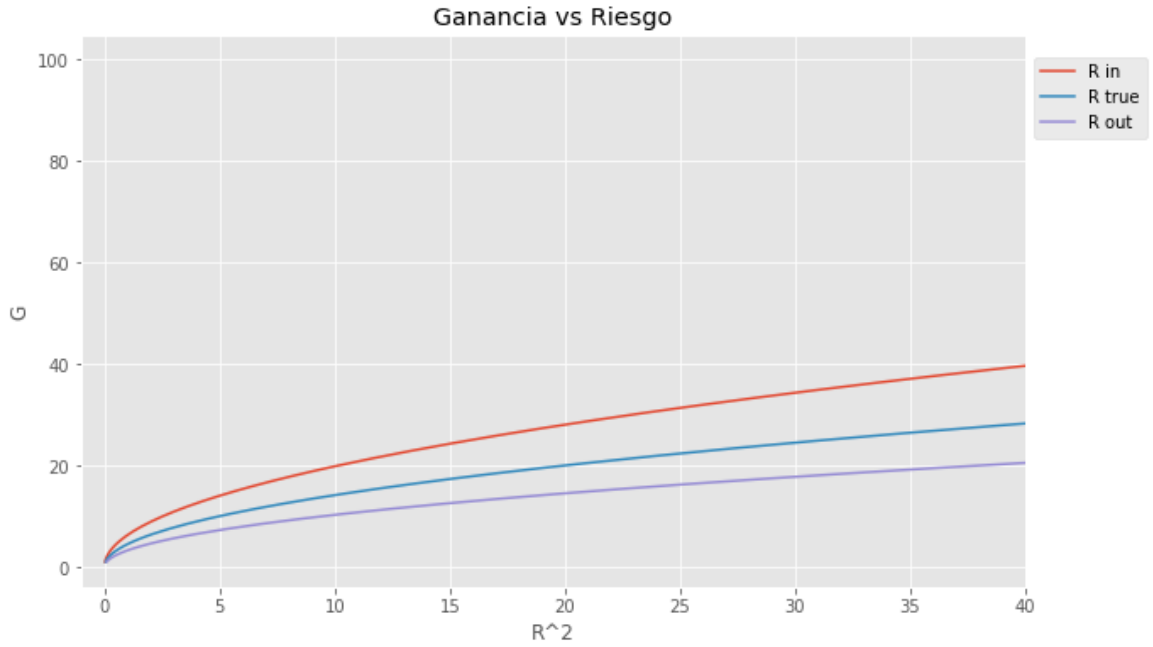


In [341]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones((p,1))
G = np.linspace(1,100,200)
S = np.identity(p)*5
W = wishart.rvs(df=200, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[341]: <matplotlib.legend.Legend at 0x7f76ae6714d0>



Podemos observar que conforme σ^2 incrementa, las fronteras se acercan y asimilan el eje horizontal.

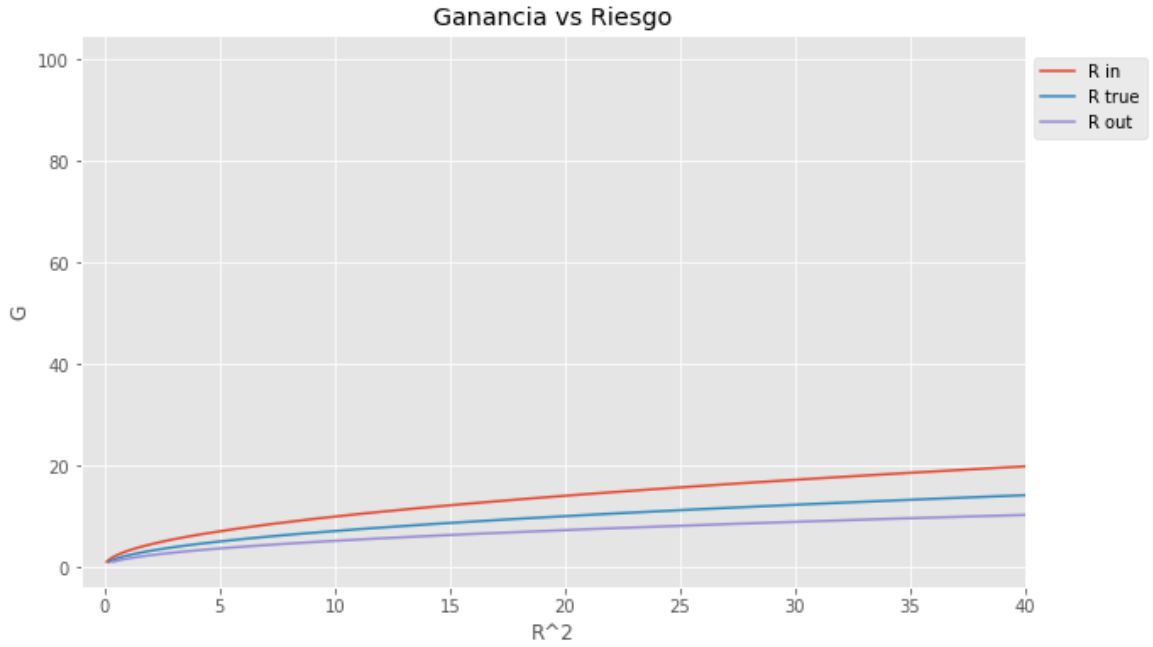
- ¿Cómo cambia el comportamiento ante otra elección del valor de la ganancia esperada del portafolio g?$

In [342]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones(p)*0.1
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=200, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[342]: <matplotlib.legend.Legend at 0x7f76af465290>

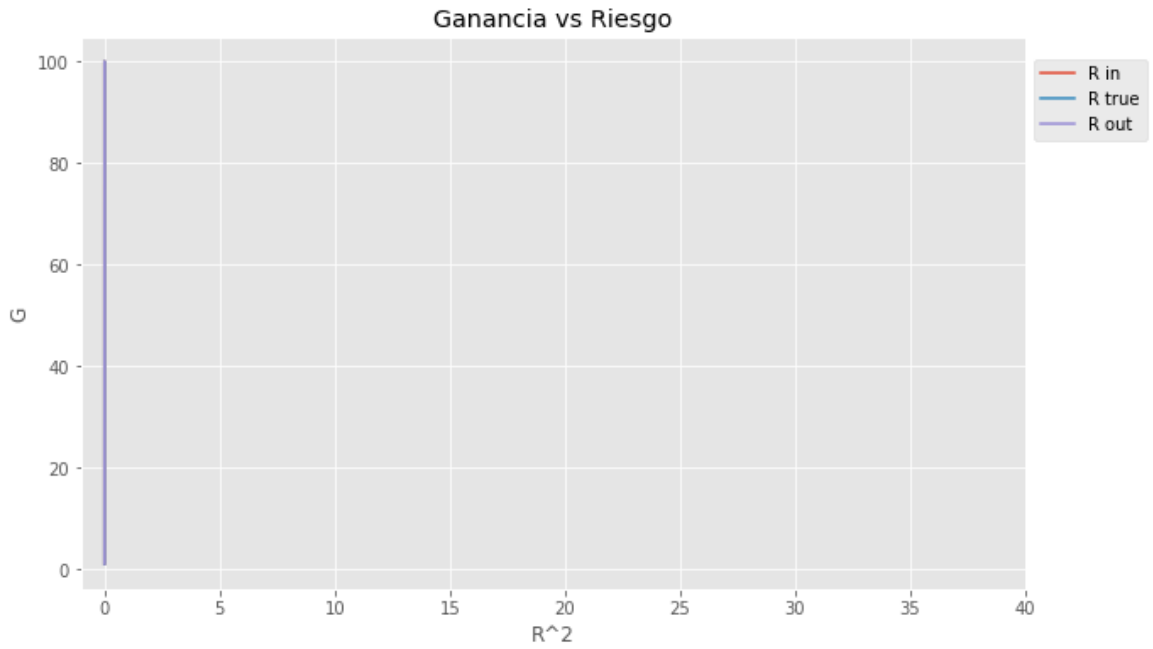


In [343]:
```
""" Implementacion Rs """
def R_t(G, S, g):
    return G**2 / (g.T@np.linalg.inv(S)@g).flatten()
def R_in(G, E, g):
    return G**2 / (g.T@np.linalg.inv(E)@g).flatten()
def R_out(G, E, S, g):
    return ((G**2)*(g.T@np.linalg.inv(E)@S@np.linalg.inv(E)@g) / ((g.T
@np.linalg.inv(E)@g)**2)).flatten()
""" Parametros Sigma y E , Dominio G"""
np.random.seed(seed=47)
p = 100
g = np.ones(p)*100
G = np.linspace(1,100,200)
S = np.identity(p)*0.2
W = wishart.rvs(df=200, scale=S, size =1, random_state = None)
E = W/199

""" Covarianzas In-sample  True  Out-sample """
R_true = R_t(G,S,g) # True
R_in = R_in(G,E,g) # In-sample
R_out = R_out(G,E, S, g) # Out-sample
fig = plt.figure(figsize = (10, 6))
plt.xlim((-1, 40))
plt.plot(R_in.flatten(), G, label ="R in")
plt.plot(R_true.flatten(), G, label ="R true")
plt.plot(R_out.flatten(), G, label ="R out")
plt.title("Ganancia vs Riesgo")
ax = plt.axes()
ax.set_xlabel('R^2')
ax.set_ylabel('G')
plt.legend(loc="upper left", bbox_to_anchor = (1,.975))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as
a previous axes currently reuses the earlier instance.  In a future v
ersion, a new instance will always be created and returned.  Meanwhil
e, this warning can be suppressed, and the future behavior ensured, b
y passing a unique label to each axes instance.
```

Out[343]: <matplotlib.legend.Legend at 0x7f76ae1a6610>



Sí $\sigma^2 \to 0$ entonces, las fronteras se van a 0 y cuando $\sigma^2 \to \infty$ las fronteras divergen.

Ejercicio 3

Realizar el siguiente procedimiento para aplicar estimadores alternativos al método RIE en un portafolio con datos reales.

- Obtener $p = 50$ series financieras para $n = 200$ días de transacción.

Solución

Tomamos una lista de activos que cotizan bajo criterio de capitalización de mercado, es decir, se seleccionaron aleatoriamente dentro del TOP 100 de capitalización de mercado de los datos financieros de yahoo!; más petróleo crudo, oro, plata, cobre y algunas criptomonedas como lo son Bitcoin, Ehtereum, Dogecoin, Link y Tether.

In [344]:
```
prices_list = ["FUJIY","MSBHF","HMC","NTDOY","HG=F","GC=F","CL=F","BTC
-USD","LINK-USD","DOGE-USD","USDT-USD",
"ETH-USD","GME","AMC","AAPL","MSFT","GOOG","AMZN","TM","FB","TSLA","BR
KB","JPM","TSMN","TCEHYN","NVDA","SONY",
"V","BAC","SMSN.IL","JNJ","BABA","WMT","NESNN","UNH","MA","HD","PG","A
SMLN","ROGN","DIS","PYPL","ATCC",
"ADBE","NFLX","XOM","CMCSA","ORCL","PFE","CSCO","NKE","KO","TMO","V
Z","NVON","LLY","DHR","PEP","ABT",
"MRK","ACN","CVX","COST","C","AVGO","T","WFC","ABBV","MS","AZN","NV
S","MCD","TXN"]
print(len(prices_list))
```

```
73
```

Posteriormente procedemos a extraer una muestra de 50 activos aleatoriamente.

In [345]:
```
print(prices_list)
# select a subset without replacement
print('Muestra :')
rndm.sample(prices_list, 3)
```

```
['FUJIY', 'MSBHF', 'HMC', 'NTDOY', 'HG=F', 'GC=F', 'CL=F', 'BTC-USD',
'LINK-USD', 'DOGE-USD', 'USDT-USD', 'ETH-USD', 'GME', 'AMC', 'AAPL',
'MSFT', 'GOOG', 'AMZN', 'TM', 'FB', 'TSLA', 'BRKB', 'JPM', 'TSMN', 'T
CEHYN', 'NVDA', 'SONY', 'V', 'BAC', 'SMSN.IL', 'JNJ', 'BABA', 'WMT',
'NESNN', 'UNH', 'MA', 'HD', 'PG', 'ASMLN', 'ROGN', 'DIS', 'PYPL', 'AT
CC', 'ADBE', 'NFLX', 'XOM', 'CMCSA', 'ORCL', 'PFE', 'CSCO', 'NKE', 'K
O', 'TMO', 'VZ', 'NVON', 'LLY', 'DHR', 'PEP', 'ABT', 'MRK', 'ACN', 'C
VX', 'COST', 'C', 'AVGO', 'T', 'WFC', 'ABBV', 'MS', 'AZN', 'NVS', 'MC
D', 'TXN']
Muestra :
```

Out[345]: ['LINK-USD', 'TXN', 'BTC-USD']

In [346]:
```
ok = yf.download("SMSN.IL", period="3y",interval = "1d")
jamon = list(ok['Close'][-365:-1])
jamon.reverse()
print(jamon)
```

```
[**********************100%**********************]  1 of 1 completed
[1483.5, 1538.0, 1527.5, 1549.0, 1562.5, 1565.0, 1584.0, 1640.5, 163
2.0, 1653.5, 1596.5, 1585.0, 1580.0, 1624.5, 1618.5, 1640.5, 1639.5,
1634.0, 1611.5, 1613.5, 1623.0, 1639.5, 1670.0, 1658.0, 1647.5, 1663.
5, 1635.5, 1606.0, 1596.0, 1618.5, 1618.0, 1575.0, 1538.5, 1554.5, 15
85.5, 1570.0, 1580.5, 1596.0, 1638.5, 1694.5, 1735.0, 1794.0, 1773.0,
1799.5, 1813.5, 1773.0, 1727.0, 1716.5, 1740.0, 1712.0, 1691.0, 1720.
0, 1713.5, 1733.0, 1721.0, 1719.5, 1697.0, 1742.0, 1768.5, 1740.0, 17
44.0, 1750.5, 1756.5, 1732.5, 1778.0, 1799.0, 1776.5, 1772.0, 1769.0,
1783.5, 1784.5, 1804.5, 1814.0, 1791.5, 1758.5, 1765.0, 1762.5, 1766.
0, 1789.0, 1840.0, 1827.0, 1814.0, 1817.0, 1826.0, 1821.0, 1833.5, 18
37.5, 1864.5, 1846.0, 1818.5, 1822.5, 1808.0, 1781.5, 1796.0, 1779.5,
1782.0, 1782.0, 1790.0, 1753.0, 1763.0, 1757.0, 1783.0, 1741.0, 1744.
0, 1799.0, 1855.0, 1853.5, 1821.5, 1837.0, 1816.0, 1821.0, 1849.0, 18
66.0, 1866.5, 1889.0, 1866.5, 1847.5, 1846.0, 1863.0, 1853.0, 1879.5,
1885.0, 1891.5, 1888.0, 1849.0, 1862.0, 1884.0, 1910.0, 1924.5, 1856.
0, 1824.0, 1816.0, 1795.0, 1811.0, 1786.0, 1797.0, 1809.0, 1825.0, 18
08.0, 1829.0, 1833.0, 1840.0, 1816.0, 1813.0, 1829.0, 1770.0, 1815.0,
1794.0, 1789.0, 1830.0, 1860.0, 1872.0, 1906.0, 1850.0, 1911.0, 1853.
0, 1855.0, 1871.0, 1894.0, 1848.0, 1863.0, 1914.0, 1921.0, 1891.0, 18
89.0, 1855.0, 1853.0, 1864.0, 1874.0, 1854.0, 1893.0, 1900.0, 1859.0,
1840.0, 1901.0, 1910.0, 1959.0, 1997.0, 1962.0, 2034.0, 1979.0, 1974.
0, 1912.0, 2006.0, 2064.0, 2012.0, 2052.0, 2058.0, 2016.0, 1913.0, 18
81.0, 1907.0, 1895.0, 1825.0, 1847.0, 1804.0, 1759.0, 1668.0, 1643.0,
1631.0, 1646.0, 1678.0, 1690.0, 1695.0, 1686.0, 1670.0, 1688.0, 1691.
0, 1653.0, 1685.0, 1661.0, 1595.0, 1586.0, 1544.0, 1514.0, 1554.0, 15
50.0, 1513.0, 1525.0, 1514.0, 1450.0, 1452.0, 1469.0, 1483.0, 1499.0,
1424.0, 1382.0, 1385.0, 1355.0, 1366.0, 1346.0, 1352.0, 1310.0, 1308.
0, 1270.0, 1263.0, 1279.0, 1282.0, 1330.0, 1329.0, 1341.0, 1327.0, 13
45.0, 1337.0, 1314.0, 1302.0, 1300.0, 1331.0, 1330.0, 1328.0, 1318.0,
1300.0, 1298.0, 1278.0, 1282.0, 1272.0, 1276.0, 1267.0, 1254.0, 1258.
0, 1237.0, 1237.0, 1257.0, 1244.0, 1253.0, 1267.0, 1281.0, 1308.0, 13
00.0, 1281.0, 1244.0, 1242.0, 1240.0, 1222.0, 1194.0, 1158.0, 1172.0,
1149.0, 1153.0, 1177.0, 1178.0, 1193.0, 1195.0, 1187.0, 1178.0, 1179.
0, 1227.0, 1233.0, 1235.0, 1220.0, 1237.0, 1261.0, 1237.0, 1220.0, 12
21.0, 1222.0, 1212.0, 1205.0, 1198.0, 1206.0, 1220.0, 1237.0, 1221.0,
1171.0, 1136.0, 1133.0, 1142.0, 1162.0, 1136.0, 1138.0, 1125.0, 1137.
0, 1118.0, 1122.0, 1098.0, 1099.0, 1113.0, 1128.0, 1153.0, 1111.0, 11
10.0, 1104.0, 1103.0, 1090.0, 1096.0, 1080.0, 1080.0, 1077.0, 1082.0,
1094.0, 1088.0, 1077.0, 1072.0, 1043.0, 1091.0, 1114.0, 1155.0, 1148.
0, 1150.0, 1171.0, 1128.0, 1131.0, 1064.0, 1043.0, 1025.0, 1020.0, 10
13.0, 996.5, 986.5, 1012.0, 1046.0, 1031.0, 1012.0, 969.0, 977.5, 99
0.5, 981.5, 988.5, 999.5, 992.0, 1006.0, 983.5, 1003.0, 1043.0]
```

Realizamos la consulta de los precios de los activos a través de la api **`yfinance`** en python, de momento consultamos 3 años para homologar después 200 días para todos los activos:

In [347]:
```
rndm.seed(7)
sequence = [i for i in range(len(prices_list))]
prices_names = rndm.sample(prices_list, 50)
price_samples = {}

for price_name in prices_names:
  query = yf.download(price_name, period="3y",interval = "1d")
  prices = list(query['Close'][-731:-1])
  price_samples[price_name] = prices
```

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

- Calcular los retornos diarios y mostrar la distribución de los retornos (agregados) en un histograma.

Solución

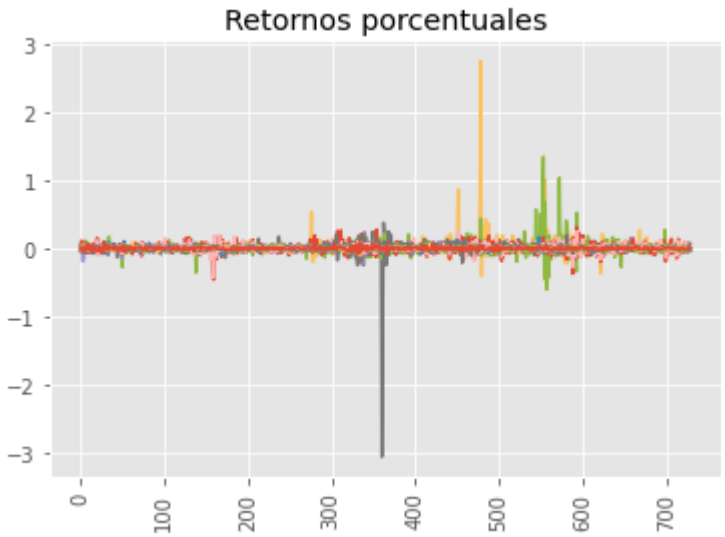
In [348]:
```
retornos = {}
for name in prices_names:
  retorno = [(price_samples[name][i]-price_samples[name][i-1])/price_s
amples[name][i-1] for i in range(1,730)]
  retornos[name] = retorno
```

In [407]:
```
retornos_m = pd.DataFrame(retornos)
agregado_por_dia = retornos_m.sum(axis=1)
sns.histplot(agregado_por_dia)
plt.title("Distribución de retornos acumulados por día")
```

Out[407]:
```
Text(0.5, 1.0, 'Distribución de retornos acumulados por día')
```

- Muestre en una tabla los estadísticos descriptivos de los retornos: media, std, curtosis, asimetría, test raíz unitaria, test JB.

In [409]:
```
plt.title('Retornos porcentuales')
plt.plot(retornos_m)
plt.xticks(rotation=90)
plt.show()
```



Solución

In [349]:
```
for name in prices_names:
  if len(price_samples[name]) < 730:
    print(name)
retornos_m = pd.DataFrame(retornos)
scaler = StandardScaler()
scaler.fit(retornos_m)
retornos_std = scaler.transform(retornos_m)
retornos_std = pd.DataFrame(retornos_std)
retornos_std = retornos_std.set_axis(prices_names, axis=1, inplace=Fal
se)
retornos_std.describe()
# Falta realizar tests
```

Out[349]:

	PYPL	FB	NKE	CL=F	DOGE-USD	GM
count	7.290000e+02	7.290000e+02	7.290000e+02	7.290000e+02	7.290000e+02	7.290000e+(
mean	1.370646e-18	2.019799e-17	1.241196e-17	-2.252047e-17	-1.709351e-17	6.217401e-
std	1.000687e+00	1.000687e+00	1.000687e+00	1.000687e+00	1.000687e+00	1.000687e+(
min	-6.372842e+00	-6.306306e+00	-5.778772e+00	-2.373251e+01	-2.999979e+00	-5.443896e+(
25%	-5.200769e-01	-4.877305e-01	-4.319011e-01	-5.926561e-02	-2.444966e-01	-3.072945e-(
50%	1.125755e-02	-3.556281e-03	-1.131601e-02	4.542804e-02	-8.779525e-02	-9.909465e-(
75%	4.953181e-01	5.322957e-01	4.652707e-01	1.486579e-01	5.900748e-02	1.599065e-(
max	5.540178e+00	4.674989e+00	7.562195e+00	2.956935e+00	1.985218e+01	1.198207e+(

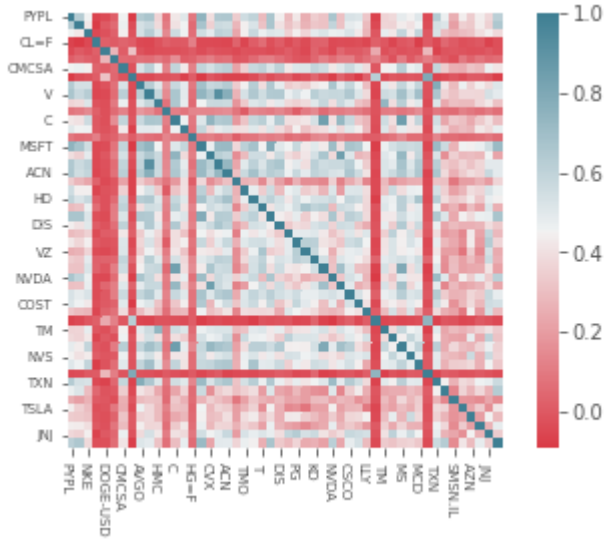
In [405]:
```
descriptivos=retornos_std.describe()
kurtosis=sp.kurtosis(retornos_std,axis=0)
asimetria=sp.skew(retornos_std,axis=0)
descriptivos.loc['kurtosis'] = kurtosis
descriptivos.loc['asimetria'] = asimetria
descriptivos.drop(index=['count', 'min', '25%', '50%', '75%', 'max'])
descriptivos
```

Out[405]:

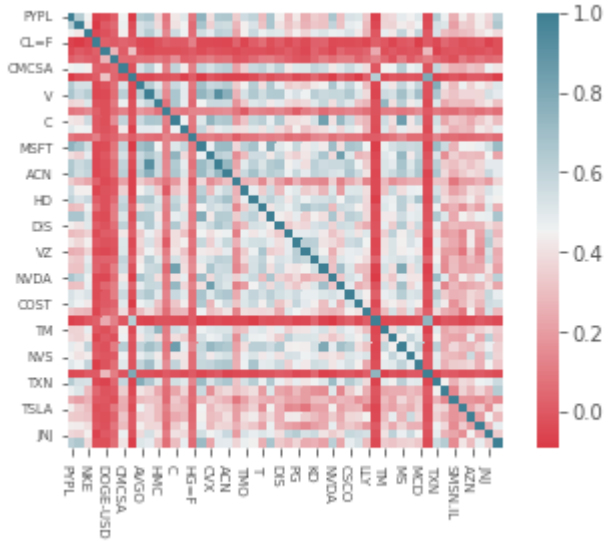
	PYPL	FB	NKE	CL=F	DOGE-USD	
count	7.290000e+02	7.290000e+02	7.290000e+02	7.290000e+02	7.290000e+02	7.29000(
mean	1.370646e-18	2.019799e-17	1.241196e-17	-2.252047e-17	-1.709351e-17	6.21740
std	1.000687e+00	1.000687e+00	1.000687e+00	1.000687e+00	1.000687e+00	1.00068
min	-6.372842e+00	-6.306306e+00	-5.778772e+00	-2.373251e+01	-2.999979e+00	-5.44389(
25%	-5.200769e-01	-4.877305e-01	-4.319011e-01	-5.926561e-02	-2.444966e-01	-3.07294
50%	1.125755e-02	-3.556281e-03	-1.131601e-02	4.542804e-02	-8.779525e-02	-9.90946
75%	4.953181e-01	5.322957e-01	4.652707e-01	1.486579e-01	5.900748e-02	1.59906
max	5.540178e+00	4.674989e+00	7.562195e+00	2.956935e+00	1.985218e+01	1.19820
kurtosis	6.276088e+00	4.780082e+00	1.258772e+01	4.450734e+02	2.177776e+02	4.86801(
asimetria	2.463960e-01	-4.645193e-02	7.179216e-01	-1.956816e+01	1.190625e+01	4.65279(

- Calcular la matriz de covarianza y mostrar su mapa de calor.

In [350]:
```
cov_m = retornos_std.cov()
ax = sns.heatmap(
    cov_m,
   # vmin=0, vmax=0.01, #center=0,
    cmap=sns.diverging_palette(10, 220, n=300),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=270,
    horizontalalignment='right',
    size = 7
)
ax.set_yticklabels(
    ax.get_yticklabels(),
    #rotation=45,
    #horizontalalignment='right',
    size = 6.5
);
```



In [351]:
```
# Verificamos correlaciones
corr_m = retornos_std.corr()
ax = sns.heatmap(
    corr_m,
    #vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(10, 220, n=300),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=270,
    horizontalalignment='right',
    size = 7
)
ax.set_yticklabels(
    ax.get_yticklabels(),
    #rotation=45,
    #horizontalalignment='right',
    size = 6.5
);
```



- Considerar las ganancias esperadas $\mathbf{g} = 1$ y calcular la frontera óptima dentro y fuera de muestra. Para esto dividir la matriz de datos en dos periodos el primero lo usará para el caso dentro de muestra y el segundo para el caso fuera de muestra de tal manea que $q = 1/2$.

Solución

Dividimos la muestra en dos:

In [352]:
```
in_sample = {}
out_sample = {}
# PARTIMOS MUESTRA
for name in prices_names:
  in_sample[name] = retornos_std[name][528:628]
  out_sample[name] = retornos_std[name][629:729]
# IN-SAMPLE
E_in = pd.DataFrame(in_sample)
# OUT-SAMPLE
E_out = pd.DataFrame(out_sample)
```

In [353]:
```
E_in.describe()
```

Out[353]:

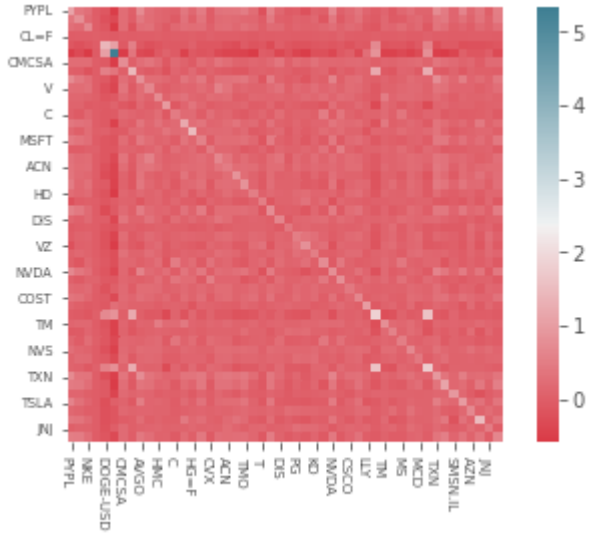
	PYPL	FB	NKE	CL=F	DOGE-USD	GME	CMCSA	1
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	
mean	-0.052409	-0.007511	-0.073852	0.055983	0.104615	0.377048	0.028919	
std	1.061175	0.889042	0.830007	0.168271	1.190139	2.312067	0.870279	
min	-2.570228	-1.862730	-2.076353	-0.521343	-2.719971	-5.443896	-2.242650	
25%	-0.662039	-0.632567	-0.579676	-0.039576	-0.470418	-0.401770	-0.450316	
50%	-0.015109	-0.081018	-0.132895	0.069822	-0.118969	-0.097267	-0.012803	
75%	0.645793	0.582579	0.417902	0.146581	0.506031	0.489327	0.475143	
max	2.852340	3.133502	2.352107	0.491606	7.203994	11.982067	3.637265	

In [354]:
```
E_out.describe()
```

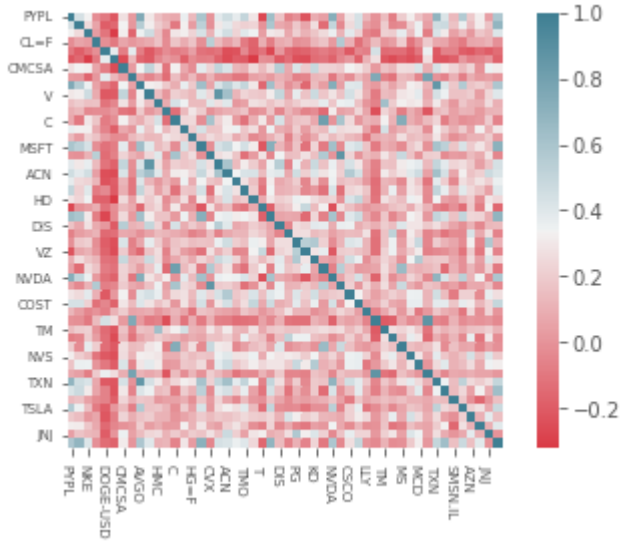
Out[354]:

	PYPL	FB	NKE	CL=F	DOGE-USD	GME	CMCSA	1
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	
mean	-0.042513	-0.033668	-0.002987	0.046285	-0.077089	-0.053773	-0.052006	
std	0.631277	0.692384	0.958059	0.147969	0.410805	0.568055	0.755897	
min	-2.554221	-2.205844	-3.126833	-0.551260	-1.376869	-2.506555	-4.129274	
25%	-0.370190	-0.445900	-0.349489	-0.046459	-0.300231	-0.342104	-0.305717	
50%	-0.003167	-0.042490	-0.048484	0.048590	-0.082852	-0.095500	0.000538	
75%	0.380590	0.424433	0.218266	0.142374	0.063378	0.101972	0.361492	
max	1.374665	2.256984	7.562195	0.445493	1.879066	2.385033	1.380230	

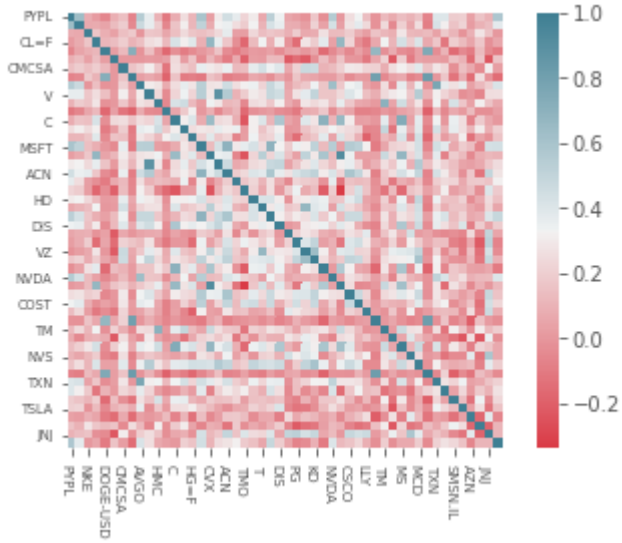
In [355]:
```
# Verificamos correlaciones
cov_m = E_in.cov()
ax = sns.heatmap(
    cov_m,
   # vmin=0, vmax=0.01, #center=0,
    cmap=sns.diverging_palette(10, 220, n=300),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=270,
    horizontalalignment='right',
    size = 7
)
ax.set_yticklabels(
    ax.get_yticklabels(),
    #rotation=45,
    #horizontalalignment='right',
    size = 6.5
);
```



In [356]:
```
# Verificamos correlaciones
corr_m = E_in.corr()
ax = sns.heatmap(
    corr_m,
    #vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(10, 220, n=300),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=270,
    horizontalalignment='right',
    size = 7
)
ax.set_yticklabels(
    ax.get_yticklabels(),
    #rotation=45,
    #horizontalalignment='right',
    size = 6.5
);
```



In [357]:
```
# Verificamos correlaciones
corr_m = E_out.corr()
ax = sns.heatmap(
    corr_m,
    #vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(10, 220, n=300),
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=270,
    horizontalalignment='right',
    size = 7
)
ax.set_yticklabels(
    ax.get_yticklabels(),
    #rotation=45,
    #horizontalalignment='right',
    size = 6.5
);
```



In [358]:
```
# Mimsa longitud
print(len(out_sample[name]))
print(len(in_sample[name]))
```

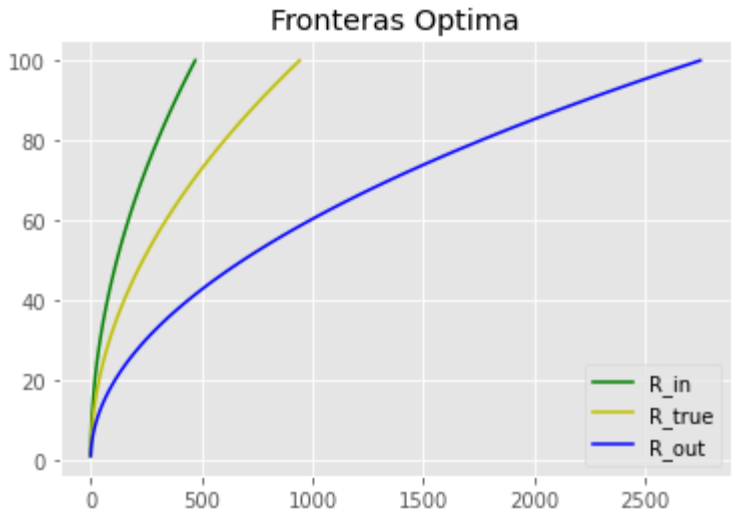
```
100
100
```

In [359]:
```
g = np.array([1]*50)
g = g.reshape(50,1)
q = 1/2
```

In [367]:
```
# R
factor1 = np.matmul(np.linalg.inv(np.array(E_in.corr())),g)
# auxiliar
factor2 = np.matmul(np.transpose(g),np.linalg.inv(np.array(E_in.corr
()))) # auxiliar
G = np.linspace(1,100,100)  # Dominio de G
# In and Out and true
R_i = lambda G: (G**2)/(np.matmul(np.transpose(g),factor1))
R_o = lambda G: (G**2)*(np.matmul(np.matmul(factor2,np.array(E_out.cor
r())),factor1))/(np.matmul(np.transpose(g),factor1))**2
R_t = lambda R: R/(1-q)
R_in_cs = R_i(G).flatten()
R_true_cs = R_t(R_in(G).flatten()).flatten()
R_out_cs = R_o(G).flatten()
# Plots
plt.style.use('ggplot')
plt.plot(R_i(G).flatten(), G,   c ="g", label = "R_in")
plt.plot(R_t(R_in(G).flatten()).flatten(), G,   c ="y", label = "R_tru
e")
plt.plot(R_o(G).flatten(), G,   c ="b", label = "R_out")

plt.title('Fronteras Optima')
plt.legend()
```

Out[367]: <matplotlib.legend.Legend at 0x7f76ac934850>



- Aplique el método de clipping descrito en la ec. (5) de "J. Bun, J.P. Bouchaud, y M. Potters, Cleaning correlation matrices, Risk magazine, (2016)". Mostrar las fronteras con sesgo y sin sesgo en la misma figura.

Solución

Sacamos la cota superior de la distribución de Marcenko-Pastur y sólo conservamos aquellos valores superiores a esta cota, dados por K. Así

$$\gamma = \frac{Tr(E) - \sum_{k=K+1}^{p} \lambda_k}{p - K}.$$

Sustituimos aquellos valores propios menores que la cota superior por γ y nuestro estimador va a estar dado por:

$$\Xi^{clip} = \sum_{k=1}^{p} \xi_k^{clip} u_k u_k'$$

con u_k el respectivo vector propio del eigenvalor k.

In [362]:
```
""" Marcenko Pastur """
p = 50
n = 100
q = 1/2
l_upper = (1 + np.sqrt(q))**2
l_lower = (1 - np.sqrt(q))**2
l = np.linspace(l_upper,l_lower , 100)

MP = lambda l : np.sqrt((maximo - l)*(l - minimo))/(2*q*np.pi*l)
```

In [363]:
```
""" Cliping """
# Eigenvalores y eigenvectores muestrales
lam_in, v_in = np.linalg.eigh(E_in.corr())
lam_in = np.sort(lam_in)[::-1]
v_in = v_in[np.argsort(lam_in)[::-1]]
# In-sample
l_sup_in =  lam_in[lam_in>l_upper]
print(l_sup_in)
print('supremos :',len(l_sup_in))
gm_in = (np.trace(E_in.corr()) - sum(list(lam_in)[3:]))/(50-3)
l_clip_in = list(lam_in)
l_clip_in[3:] = [gm_in for l in l_clip_in[3:]]
E_clip_in = sum([np.matmul(np.reshape(v_in[:,i],(50,1)),np.reshape(v_i
n[:,i],(1,50))) for i in range(50)])
print('gamma :',gm_in)
print('traza E in sample :',np.trace(E_in.corr()))
print('traza E_clip in sample :',np.trace(E_clip_in))
```

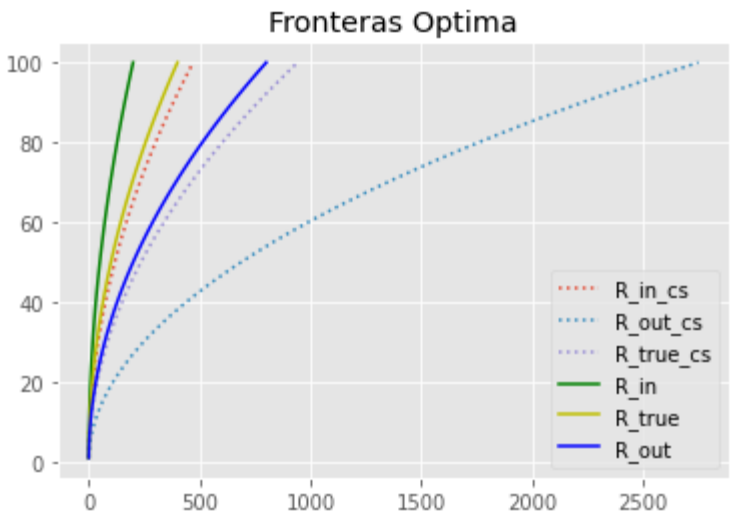
```
[11.22773644  5.64035872  3.62480913  2.92684511]
supremos : 4
gamma : 0.43601924009634335
traza E in sample : 50.0
traza E_clip in sample : 50.0
```

In [364]:
```
""" Cliping """
# Eigenvalores y eigenvectores muestrales
lam_out, v_out = np.linalg.eigh(E_out.corr())
lam_out = np.sort(lam_out)[::-1]
v_out = v_out[np.argsort(lam_out)[::-1]]
# Out-sample
l_sup_out =  lam_out[lam_out>l_upper]
print(l_sup_out)
print('supremos :',len(l_sup_out))
gm_out = (np.trace(E_out.corr()) - sum(list(lam_out)[4:]))/(50-4)
l_clip_out = list(lam_out)
l_clip_out[4:] = [gm_out for l in l_clip_out[4:]]
E_clip_out = sum([np.matmul(np.reshape(v_out[:,i],(50,1)),np.reshape(v
_out[:,i],(1,50))) for i in range(50)])
print('gamma :',gm_out)
print('traza E :',np.trace(E_out.corr()))
print('traza E_clip :',np.trace(E_clip_out))
```

```
[11.19584272  5.10264256  4.07724106  3.73064239]
supremos : 4
gamma : 0.5240514942340952
traza E : 50.0
traza E_clip : 50.0
```

In [365]:
```
""" R """
factor1 = np.matmul(np.linalg.inv(np.array(E_clip_in)),g)
# auxiliar
factor2 = np.matmul(np.transpose(g),np.linalg.inv(np.array(E_clip_i
n))) # auxiliar
G = np.linspace(1,100,100)  # Dominio de G
# In and Out and true
R_in =  lambda G: (G**2)/(np.matmul(np.transpose(g),factor1))
R_out = lambda G: (G**2)*(np.matmul(np.matmul(factor2,E_clip_out),fact
or1))/((np.matmul(np.transpose(g),factor1))**2)
R_true = lambda R: R/(1-q)
# Plots
plt.style.use('ggplot')
plt.plot(R_in_cs, G, label = "R_in_cs",linestyle='dotted')
plt.plot(R_out_cs, G, label = "R_out_cs",linestyle='dotted')
plt.plot(R_true_cs, G, label = "R_true_cs",linestyle='dotted')
plt.plot(R_in(G).flatten(), G,   c ="g", label = "R_in")
plt.plot(R_true(R_in(G).flatten()).flatten(), G,   c ="y", label = "R_
true")
plt.plot(R_true(R_in(G).flatten()).flatten()/(1-q), G,   c ="b", label
= "R_out")
R_in_cs
plt.title('Fronteras Optima')
plt.legend()
```

Out[365]: <matplotlib.legend.Legend at 0x7f76acb54850>



- Aplique el método de contración lineal descrito en la ec. (3) del mismo artículo para valores de $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Mostrar los resultados en la misma figura.

Solución

Contracción lineal básica: combinación lineal de la estimación muestral y la matriz identidad:

$$\Xi := \alpha E + (1-\alpha) I_N$$

con E igual a la matriz de correlaciones muestral de la matriz dentro de muestra y fuera de muestra.
Calculamos: </br>

$$\mathcal{R}^2_{in} = \frac{\mathcal{C}^2}{g' E^{-1} g}$$

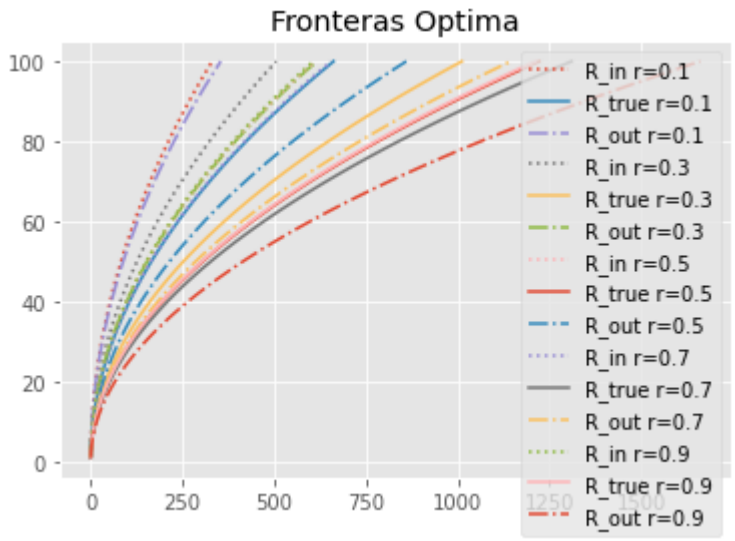
$$\mathcal{R}^2_{true} = \frac{\mathcal{R}^2_{in}}{1-q}$$

$$\mathcal{R}^2_{out} = \frac{\mathcal{C}^2 g' E^{-1} \tilde{E} E^{-1} g}{g' E^{-1} g}$$

In [366]:
```
""" Contracción lineal """
alpha = [0.1,0.3,0.5,0.7,0.9]

for a in alpha:
  """ R """
  E_bas_in = a*np.array(E_in.corr())+(1-a)*np.identity(50)
  E_bas_out = a*np.array(E_out.corr())+(1-a)*np.identity(50)
  factor1 = np.matmul(np.linalg.inv(E_bas_in),g)                # auxili
ar
  factor2 = np.matmul(np.transpose(g),np.linalg.inv(np.array(E_bas_i
n))) # auxiliar
  G = np.linspace(1,100,100)  # Dominio de G
  # In and Out and true
  R_in =  lambda G: (G**2)/(np.matmul(np.transpose(g),factor1))
  R_out = lambda G: (G**2)*(np.matmul(np.matmul(factor2,E_bas_out),fac
tor1))/((np.matmul(np.transpose(g),factor1))**2)
  R_true = lambda R: R/(1-q)
  # Plots
  plt.style.use('ggplot')
  plt.plot(R_in(G).flatten() ,G, label = "R_in r="+str(a),linestyle='d
otted')
  plt.plot(R_true(R_in(G).flatten()).flatten(), G, label = "R_true r
="+str(a),linestyle='solid')
  plt.plot(R_out(G).flatten(), G, label = "R_out r="+str(a),linestyle=
'dashdot')

  plt.title('Fronteras Optima')
  plt.legend(loc = 'upper right')
```



In [412]:
```
!jupyter nbconvert --to html T7_TSE_DRA.ipynb
```

```
[NbConvertApp] Converting notebook T7_TSE_DRA.ipynb to html
[NbConvertApp] Writing 1185285 bytes to T7_TSE_DRA.html
```