# Summer Research Project on Green Finance

Dorothea Langner
Supervisor Fred Espen Benth
University of Oslo, Department of Risk and Stochastics

June-August 2024

## 1 Explanation of Code

First, load the required packages.

[1]:
```
{
    "tags": [
        "remove-cell"
    ]
}
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import pandas as pd
import numpy as np #for algebraic computations
import matplotlib.pyplot as plt
from patsy import dmatrices
from statsmodels.stats.correlation_tools import cov_nearest #to get valid␣
 ↪covariance matrix
import yfinance as yf
```

The following tickers are of ETFs which exclusively represent one sector. We take data from June 1st, 2021 to June 1st, 2024. Let us get a glimpse of the data.

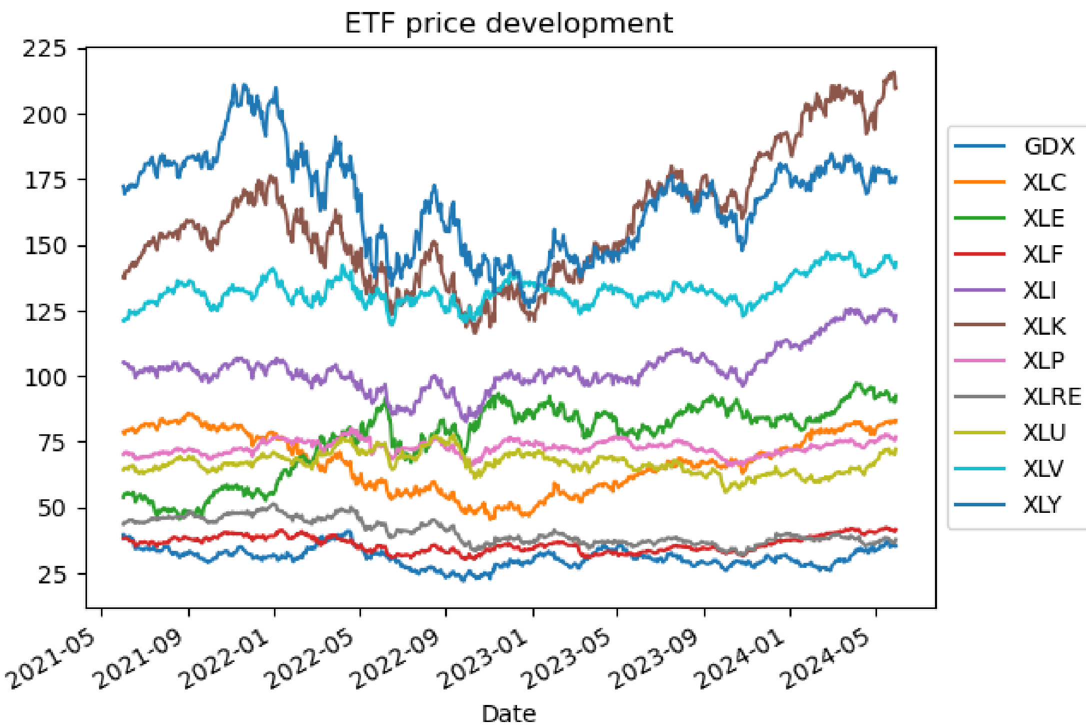
[3]:
```
tickers =␣
 ↪sorted(['GDX','XLC','XLY','XLP','XLE','XLF','XLV','XLI','XLRE','XLK','XLU'])

d = len(tickers)

data = yf.download(tickers,start='2021-06-01',end='2024-06-01')
print(data.head())
```

```
[*********************100%%***********************]  11 of 11 completed

Price        Adj Close                                                   \
Ticker             GDX        XLC        XLE        XLF         XLI
Date
2021-06-01   39.490002  78.632080  53.788052  38.069267  105.168709
```

```
2021-06-02  39.540001  78.392715  54.790005  38.129028  104.869743
2021-06-03  38.169998  77.864120  54.938812  38.218674  104.640526
2021-06-04  38.680000  78.981163  55.305862  38.318279  104.979355
2021-06-07  38.650002  79.390083  55.067772  38.069267  104.251862

Price                                                                 ...  \
Ticker             XLK        XLP       XLRE        XLU         XLV  ...
Date                                                                  ...
2021-06-01  137.478821  70.041199  43.557484  64.242035  121.095505  ...
2021-06-02  138.427155  70.299255  44.161350  64.579308  120.856445  ...
2021-06-03  137.139420  70.735954  44.072254  64.966187  121.224998  ...
2021-06-04  139.774796  70.984077  44.111855  64.866989  121.613472  ...
2021-06-07  139.744843  70.864983  44.527630  64.986031  122.051758  ...

Price        Volume                                                          \
Ticker          XLC       XLE       XLF       XLI      XLK      XLP     XLRE
Date
2021-06-01  3301400  36285100  35710700   9456900  6453700  8893800  5512300
2021-06-02  2782000  33946700  37537500   7547800  5049100  6411300  4599000
2021-06-03  2812300  29380800  54877500  10342300  6360600  9410000  3447300
2021-06-04  2333200  26329400  27250900   7118200  5591300  6078200  3061300
2021-06-07  3321100  20035200  37401800   8298700  5892000  7306900  5659700

Price
Ticker           XLU       XLV      XLY
Date
2021-06-01  11026100  13368300  4680600
2021-06-02  10245700  13245100  3531600
2021-06-03  11853600  13244400  4351400
2021-06-04   7234800  11228900  3216600
2021-06-07   6911200  13978500  3194900

[5 rows x 66 columns]
```

In the following, we will consider the daily adjusted closing price of the ETFs. To get a better overview of the data, look at the price development over the considered period.

[4]:
```
#plot daily adjusted closing price of selcted ETFs
def plot_prices():
    data['Adj Close'].plot(title='ETF price development').
 ↪legend(bbox_to_anchor=(1.0, 0.5), loc='center left')
    plt.show()

plot_prices()
```




The aim of our analysis is to find the Markowitz portfolio for a return period of one year. Thus, we need to compute the the return rates of each ETF over the span of one year, and then form an average over them. Furthermore, the covariances of the ETFs over a yearly span need to be computed. Since the data is only available from June 1st, 2021 to May 31st, 2024, we will use this time period.

[5]:
```
#manual function to get yearly data (1 year interval not supported by yfinance)
#data only available from 2021-06-01 to 2024-05-31
data_y = data.loc[(data.index=='2021-06-01')|
                  (data.index=='2022-06-01')|(data.index=='2023-06-01')|
                  (data.index=='2024-05-31')]

#computations including all ETFs
#compute covariance matrix (on yearly basis)
C_all = data_y['Adj Close'].cov()
```

First, we will perform the computations of the covariance matrix and the average return rates for all ETFs. Later on, we will do those for a subset of our chosen ETFs which we classify as "green" or not "non-green".

[6]:
```
#compute yearly returns
data_y_ret = data_y['Adj Close'].pct_change()

#get mean and standard deviation of yearly returns
```

```
data_y_av_ret = data_y_ret.describe(include='all').loc['mean']

#compute average yearly return rates
av_ret = data_y_av_ret.to_numpy()
av_ret = av_ret.reshape((d,1))
cov_all = C_all.to_numpy()
```

[7]:

```
#do this for green portfolios, removed ETFs
def green(noticker):
    def get_indices(lst, targets):
        return [index for index, element in enumerate(lst) if element in targets]
    indices = get_indices(tickers,noticker)
    tickers_green = [x for x in tickers if x not in noticker]
    data_y_green = data_y.drop(noticker,level=1,axis=1)
    C_green = data_y_green['Adj Close'].cov()
    data_y_green_ret = data_y_green['Adj Close'].pct_change()
    data_y_green_av_ret = data_y_green_ret.describe(include='all').loc['mean']
    av_green_ret = data_y_green_av_ret.to_numpy()
    dg = len(tickers_green)
    av_green_ret = av_green_ret.reshape((dg,1))
    cov_green = C_green.to_numpy()
    return tickers_green,av_green_ret,cov_green,dg,indices
```

Next, we need to consider the problem that the empirical covariance matrix is not valid, meaning it is not positive semidefinite, or in particular for the computations of the Markowitz portfolio, not positive definite, meaning it is invertible. The empirical covariance matrix is symmetric. If the empirical covariance matrix is invalid, we can find the nearest covariance function, which is positive semidefinite, in the Frobenius-norm. There is no nearest positive definite matrix, but we will use the strategy of taking $C' = C + a\mathrm{Id}$, where a is a positive constant big enough such that $C'$ is positive definite. Then, the infimum over the $a > 0$ such that $C'$ is positive definite can be somewhat considered the "nearest" positive definite matrix to C. The following code provides these computations and checks if we actually do have a positive definite matrix.

[8]:

```
def is_posdef(A):
    try:
        _ = np.linalg.cholesky(A)
        return True
    except np.linalg.LinAlgError:
        return False

def PD(A):
    if is_posdef(A):
        return A
    else:
        A = 1/2*(A+A.T)
        eigvals = list(np.linalg.eigvalsh(A))
        for i in range(len(eigvals)):
            if abs(eigvals[i])<=np.finfo(float).eps:
```

```
                eigvals[i]=0
        ev_min = min(list(filter(lambda num: num!=0,eigvals)))
        if ev_min < 0:
            A = A + (-ev_min+1e-15)*np.eye(len(A))
        else:
            A = A + 1e-15*np.eye(len(A))
        return A

def PSD(A):
  E = np.linalg.eigvalsh(A)
  if np.all(E > -1e-15):
      return A
  else:
      return cov_nearest(A,threshold=1e-15,method='nearest')
```

Note that the empirical covariance matrix we are given has very small below-zero eigenvalues. Additionally, the nearest positive semi-definite matrix is not positive definite, so we need the other function to get to an invertible covariance matrix. Next, we compute the constants needed for the Markowitz portfolio, as well as the latter itself. Note that in the inverse of the covariance function is computed via computing the pseudoinverse. Since the matrix is positive definite, so invertible, these two coincide in theory. Numerically, the two can be quite different if the algorithms encounter numerical instability. The numpy algorithm to compute the inverse uses LU-decomposition, whereas the pseudoinverse function uses singular value decomposition. The latter is numerically more accurate and stable. Therefore, this method is used to compute the inverse matrix of the "nearest" positive definite covariance matrix. Since the empirical covariance matrix has very small below-zero eigenvalues, the results computing the Markowitz portfolio via the above described method versus directly the pseudoinverse of the empirical covariance matrix are almost identical (but not exactly identical).

[9]:
```
#compute Markowitz portfolio
def constants(avret,cov,d):
    cov = PD(cov)
    cov_inv = np.linalg.pinv(cov)
    a = np.ones((1,d)) @ cov_inv @ avret
    a = a.item()
    b = avret.T @ cov_inv @ avret
    b = b.item()
    c = np.ones((1,d)) @ cov_inv @ np.ones((d,1))
    c = c.item()
    return a,b,c

def markowitz_portf(avret,cov,r,d):
    cov = PD(cov)
    cov_inv = np.linalg.pinv(cov)
    a = constants(avret, cov,d)[0]
    b = constants(avret, cov,d)[1]
    c = constants(avret, cov,d)[2]
```

```python
    w_0 = (b*np.ones((d,1))-a*avret)/(b*c-a**2)
    w_r = (c*avret-a*np.ones((d,1)))/(b*c-a**2)
    #the Markowitz portfolio:
    w_star = cov_inv@(w_0+r*w_r)
    s_2 = c*((r-a/c)**2)/(b*c-a**2)+1/c #variance
    return (np.sqrt(s_2),w_star)
```

From the hyperbolic equation dependent on the variance and the expected return, one can deduce the formula of the efficient portfolio front. We also compute the inverse, a function taking the parameter r.

[10]:
```python
#efficient portfolio front equation
def effport(a,b,c,s):
    return np.sqrt((s**2-1/c)*(b*c-a**2)/c)+a/c

#equation as function of r
def inverse(a,b,c,r):
    return np.sqrt(c*(r-a/c)**2/(b*c-a**2)+1/c)
```

To visualise each efficient portfolio front, and all of them together for comparison, the code below is given.

[11]:
```python
#plot efficient portfolio front
def plot_effport(avret,cov,d,color,label):
    a = constants(avret, cov,d)[0]
    b = constants(avret, cov,d)[1]
    c = constants(avret, cov,d)[2]
    s = np.linspace(0,50,num=1000)
    plt.plot(s,effport(a,b,c,s),color=color,linewidth=1,label=label)
    plt.title('Efficient portfolio front')
    plt.xlabel('risk sigma')
    plt.ylabel('expected return r')
    plt.legend()
    plt.show()

#plot both efficient portfolio fronts together
def plot_together(lists):
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    s = np.linspace(0,50,num=1000)
    plt.plot(s,effport(a1,b1,c1,s),color='navy',linewidth=1,label='all')
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
```

```
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        string = ','.join(str(elm) for elm in lists[i])
        plt.plot(s,effport(a,b,c,s),linewidth=1,label='no '+string)
    plt.title('Efficient portfolio front')
    plt.xlabel('risk sigma')
    plt.ylabel('expected return r')
    plt.legend()
    plt.show()
```

Now, we compute the loss functions. The first one, at arbitrary fixed risk level, calculates the differences in expected returns between the Markowitz portfolio including all ETFs and that only including the ETFs considered to be "green". The second one, at arbitrary fixed return value, computes the increase of risk with the Markowitz portfolios including all and only a subset of our ETFs.

[12]:
```
#expected loss in return at fixed risk
def losses_s(s,lists):
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    val = []
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        val.append(effport(a1,b1,c1,s)-effport(a,b,c,s))
    return val

#increase in risk at fixed expected return
def losses_r(r,lists):
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    losses = []
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        losses.append(inverse(a,b,c,r)-inverse(a1,b1,c1,r))
    return losses
```

Lastly, we plot the loss functions, the individual portfolios, and all the portfolios together in comparison.

[13]:
```
#plot loss in r
def plot_loss_s(lists):
    s = np.linspace(0,50,num=1000)
    for i in range(len(lists)):
        string = ','.join(str(elm) for elm in lists[i])
        plt.plot(s,losses_s(s,lists)[i],linewidth=1,label='loss without '+string)
    plt.title('Expected losses in return at same risk')
    plt.xlabel('risk')
    plt.ylabel('expected loss')
    plt.legend()
    plt.show()

#plot increase in s
def plot_loss_r(lists):
    r = np.linspace(0,1,num=1000)
    for i in range(len(lists)):
        string = ','.join(str(elm) for elm in lists[i])
        plt.plot(r,losses_r(r,lists)[i],linewidth=1,label='increased risk␣
↪without '+string)
    plt.title('Increase in risk at same expected return')
    plt.xlabel('expected return')
    plt.ylabel('increase in risk')
    plt.legend()
    plt.show()

def plot_investments(tickers,avret,cov,r,d):
    values = list(markowitz_portf(avret,cov,r,d)[1].flat)
    colors = ['g' if m > 0 else 'r' for m in values]
    plt.bar(tickers,values,width=.5,color=colors)
    plt.xlabel('ETFs')
    plt.ylabel('share in portfolio')
    plt.title('Markowitz portfolio for r= '+str(r))
    plt.show()

def plot_investments_together(r,lists):
    n = len(lists)+1
    values1 = list(markowitz_portf(av_ret,cov_all,r,d)[1].flat)
    x_axis = np.arange(len(tickers))
    plt.bar(x_axis - len(lists)/8,values1,color='navy',label='all',width=1/(n+1))
    for i in range(len(lists)):
        values =␣
↪list(markowitz_portf(green(lists[i])[1],green(lists[i])[2],r,green(lists[i])[3])[1].
↪flat)
        for j in range(len(green(lists[i])[4])):
```

```
            values.insert(green(lists[i])[4][j],0)
        string = ','.join(str(elm) for elm in lists[i])
        plt.bar(x_axis - len(lists)/8+(i+1)*1/(n+1),values,label='no␣
↪'+string,width=1/(n+1))
    plt.xticks(x_axis,tickers)
    plt.xlabel('ETFs')
    plt.ylabel('share in portfolio')
    plt.title('Markowitz portfolios for r= '+str(r))
    plt.legend()
    plt.show()
```

Finally, look at the results considering a portfolio without the XLE ETF which invests into oil and gas companies, as well as suppliers and infrastructure to such companies, and a portfolio without the XLE and GDX ETFs. The GDX ETF has holdings in gold mining and related companies. First, look at the risk and Markowitz portfolio share values for r=0.1. Including all ETFs, we have

[14]:
```
#plot results
print(markowitz_portf(av_ret, cov_all, 0.1,d))
```

```
(7.252516529565316, array([[-0.0106992 ],
       [-0.00761719],
       [ 0.38609441],
       [ 0.0835073 ],
       [ 0.04982632],
       [-0.09589748],
       [ 0.05232146],
       [ 0.11949097],
       [ 0.22472966],
       [ 0.14806207],
       [ 0.05018169]]))
```

[15]:
```
print(markowitz_portf(green(['XLE'])[1],green(['XLE'])[2], 0.
↪1,green(['XLE'])[3]))
```

```
(16.2849530030226, array([[-0.15343974],
       [-0.2741187 ],
       [ 0.07622152],
       [ 0.00285427],
       [ 0.36811886],
       [ 0.15192447],
       [ 0.10927762],
       [ 0.4738317 ],
       [ 0.45644526],
       [-0.21111527]]))
```

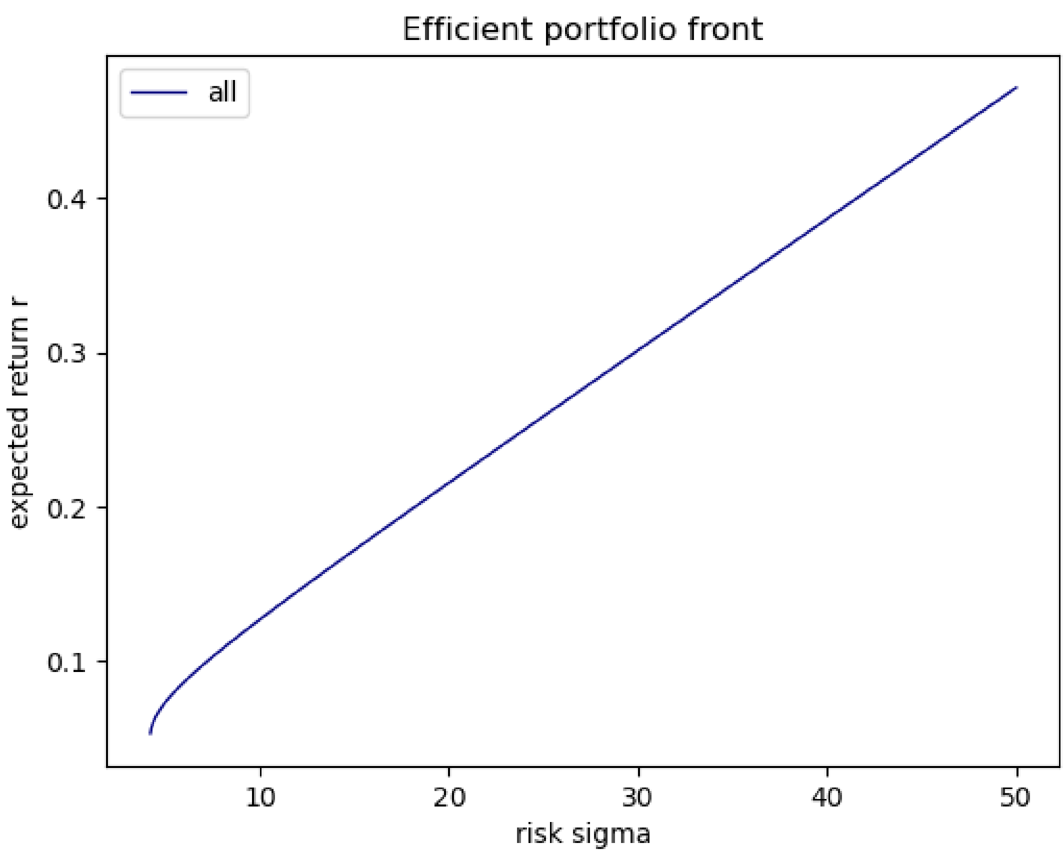
[16]:
```
print(markowitz_portf(green(['XLE','GDX'])[1],green(['XLE','GDX'])[2], 0.
↪1,green(['XLE','GDX'])[3]))
```

```
(17.460970149014873, array([[-0.24420588],
```

```
[ 0.04802758],
[ 0.00553271],
[ 0.4381719 ],
[ 0.13002666],
[ 0.05814815],
[ 0.37393555],
[ 0.39526243],
[-0.20489911]]))
```

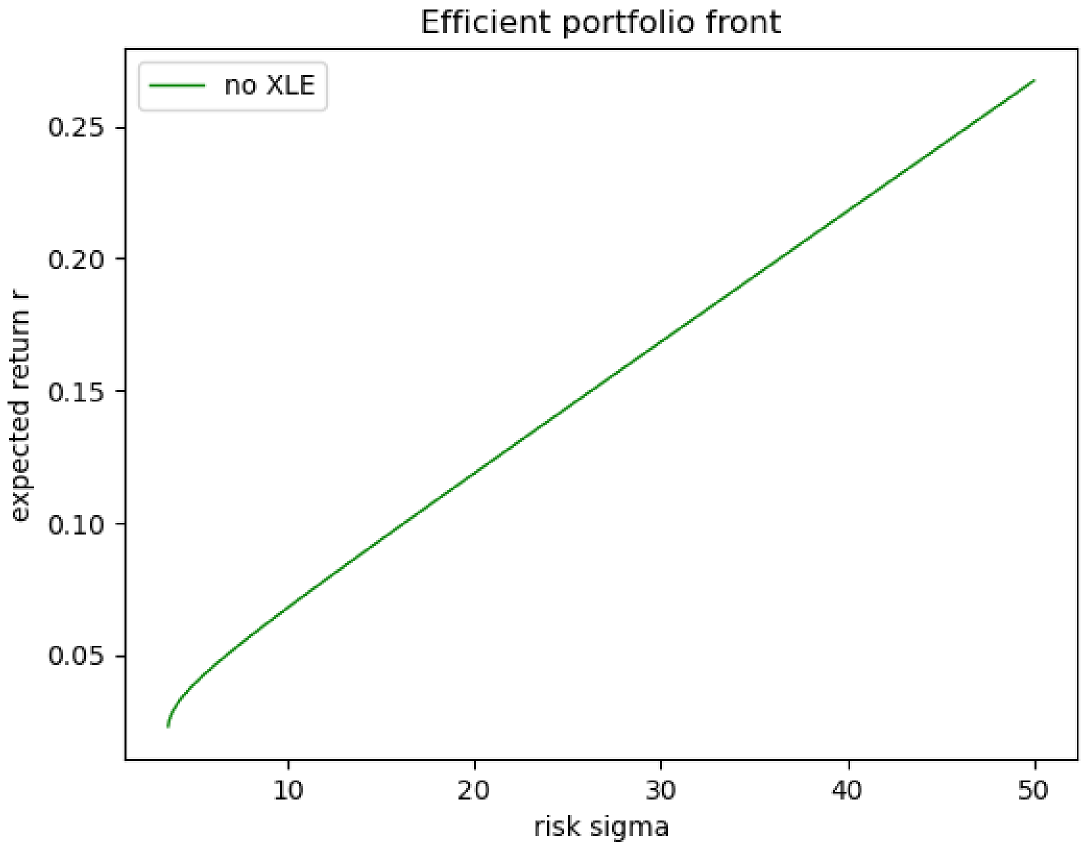
Next, look at the efficient portfolio front investing into all ETFs,

[17]:
```
plot_effport(av_ret,cov_all,d,'navy','all')
```



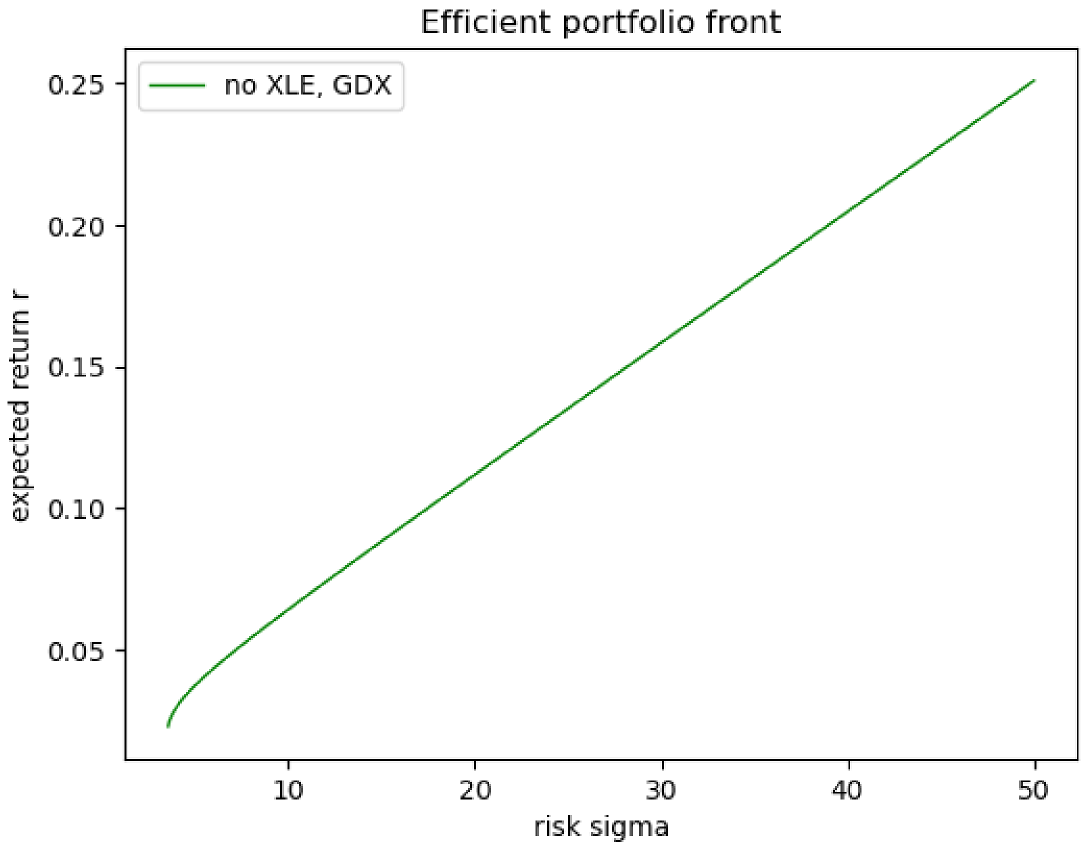

not into XLE,

[18]:
```
plot_effport(green(['XLE'])[1],green(['XLE'])[2],green(['XLE'])[3],'green','no␣
↪XLE')
```



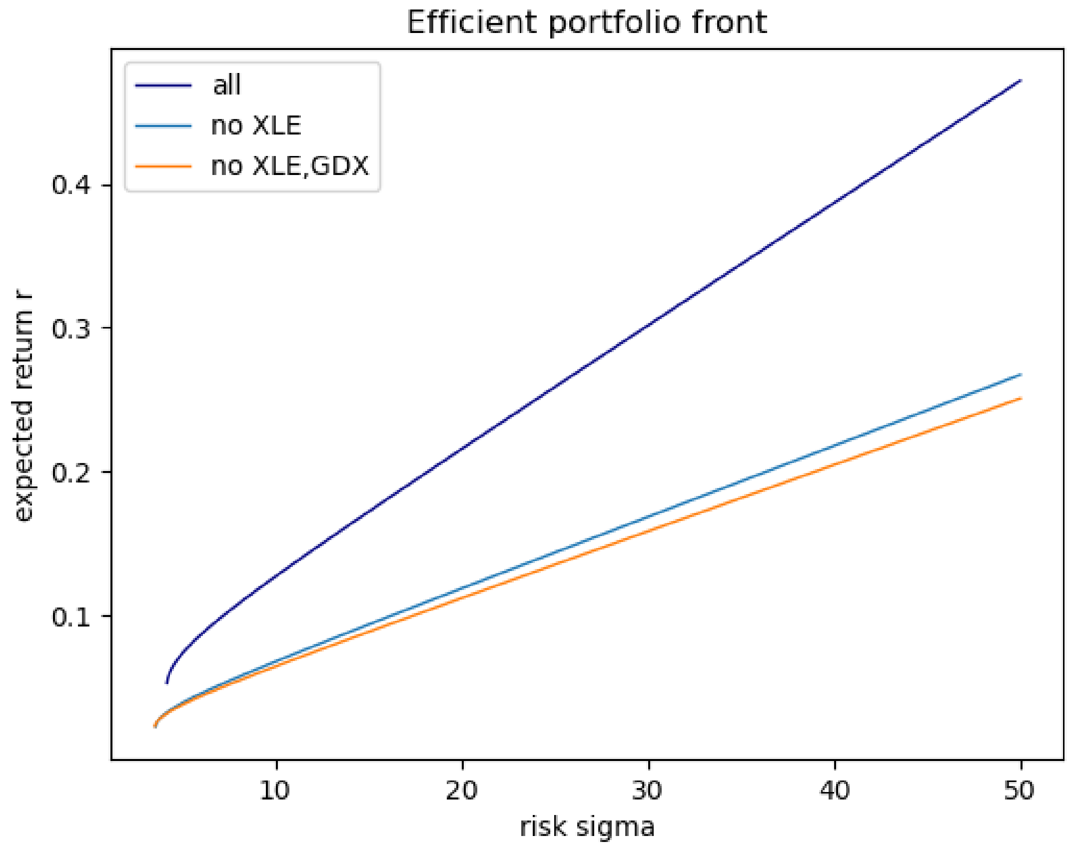

and not into XLE and GDX.

```
[19]: plot_effport(green(['XLE','GDX'])[1],green(['XLE','GDX'])[2],green(['XLE','GDX'])[3],'green','n
 ↪XLE, GDX')
```



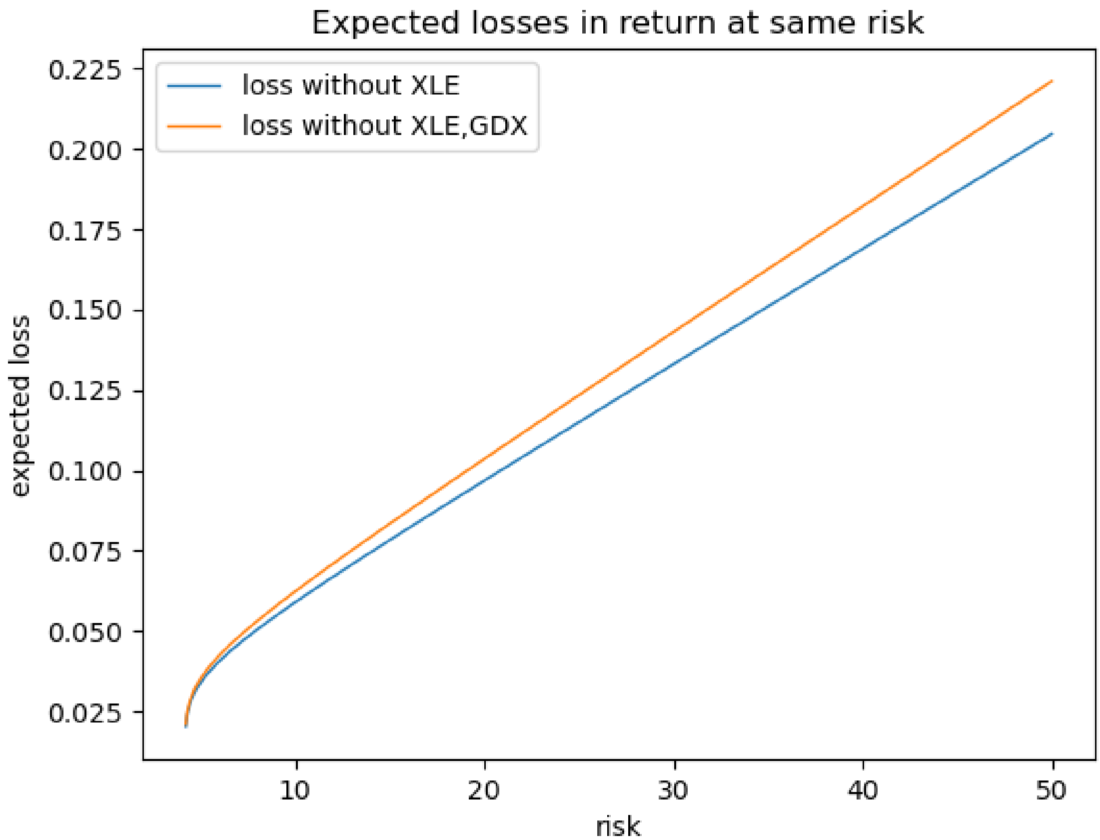

Now, look at all of them together to see the differences.

[20]:
```
plot_together([['XLE'],['XLE','GDX']])
```

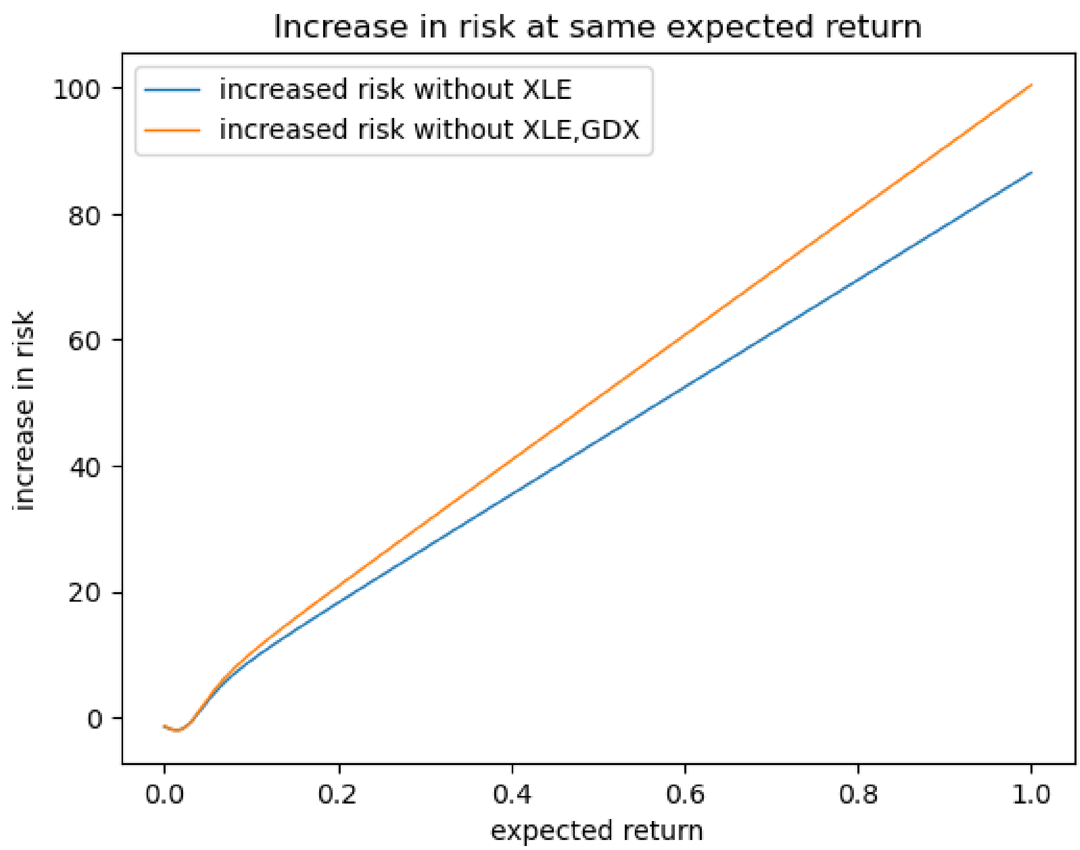



Look at the two loss functions, first in r, then in s.

[21]:
```
plot_loss_s([['XLE'],['XLE','GDX']])
```

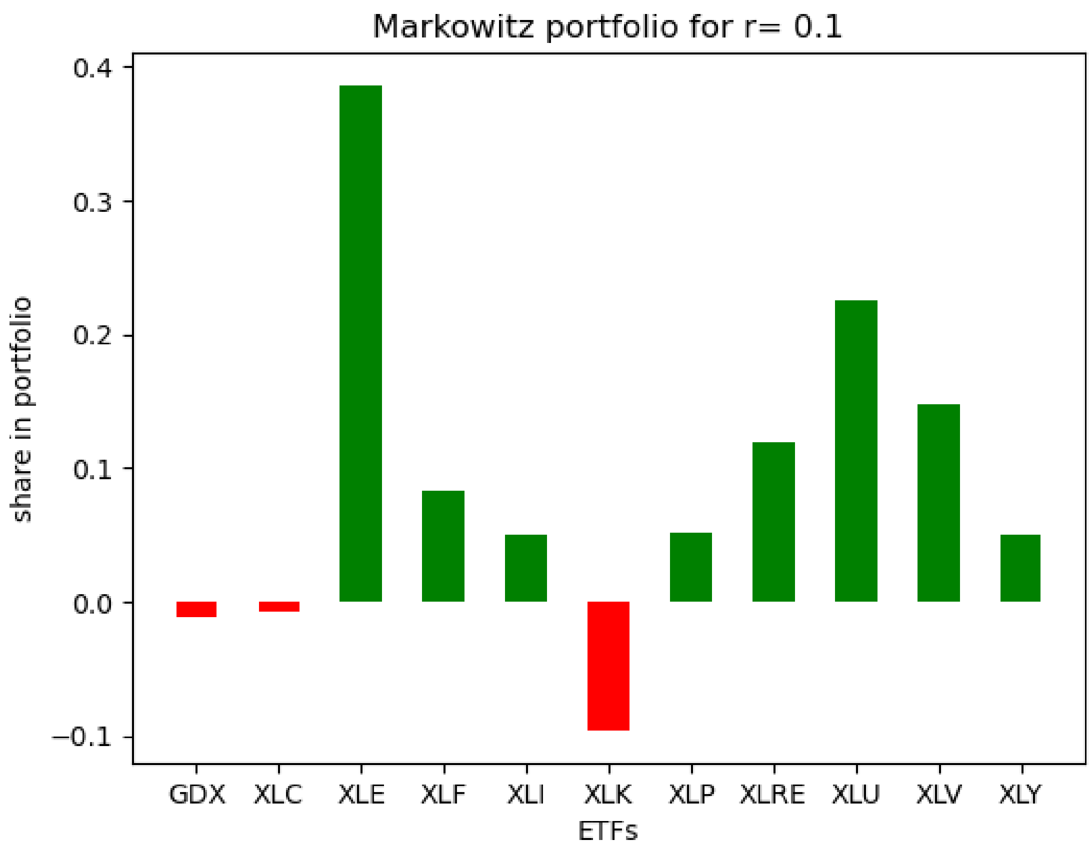



[22]:
```
plot_loss_r([['XLE'],['XLE','GDX']])
```



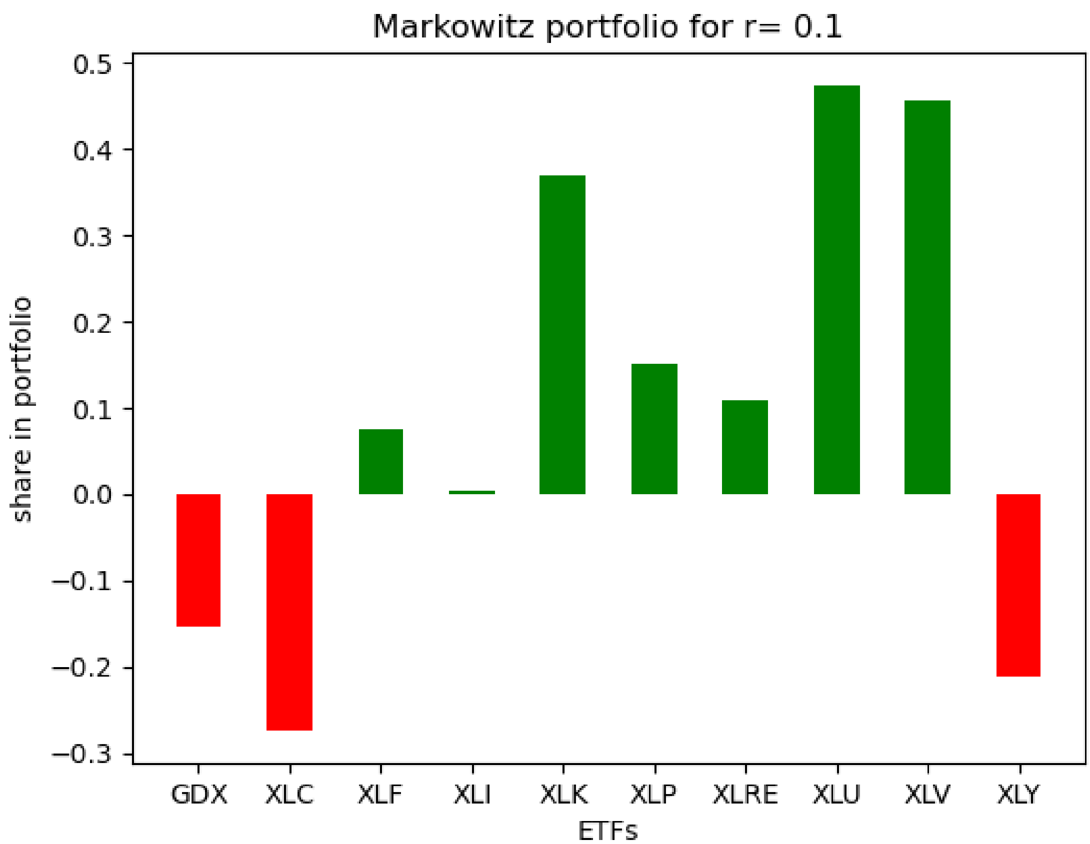

Below is the Markowitz portfolio with all ETFs visualised, at expected return level $r = 0.1$.

[23]:
```
plot_investments(tickers,av_ret, cov_all, 0.1, d)
```

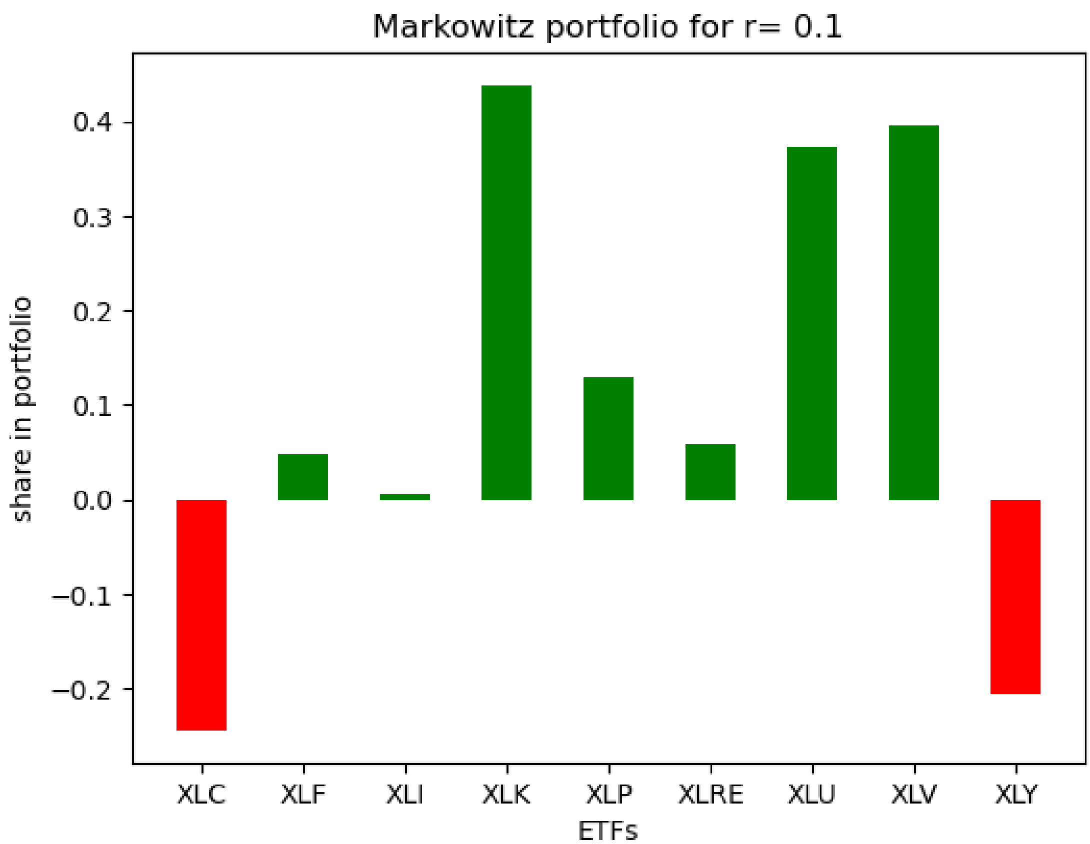



At the same value for r, look at the Markowitz portfolios first without XLE and then without XLE and GDX.

[24]:
```
plot_investments(green(['XLE'])[0],green(['XLE'])[1],green(['XLE'])[2],0.
↪1,green(['XLE'])[3])
```

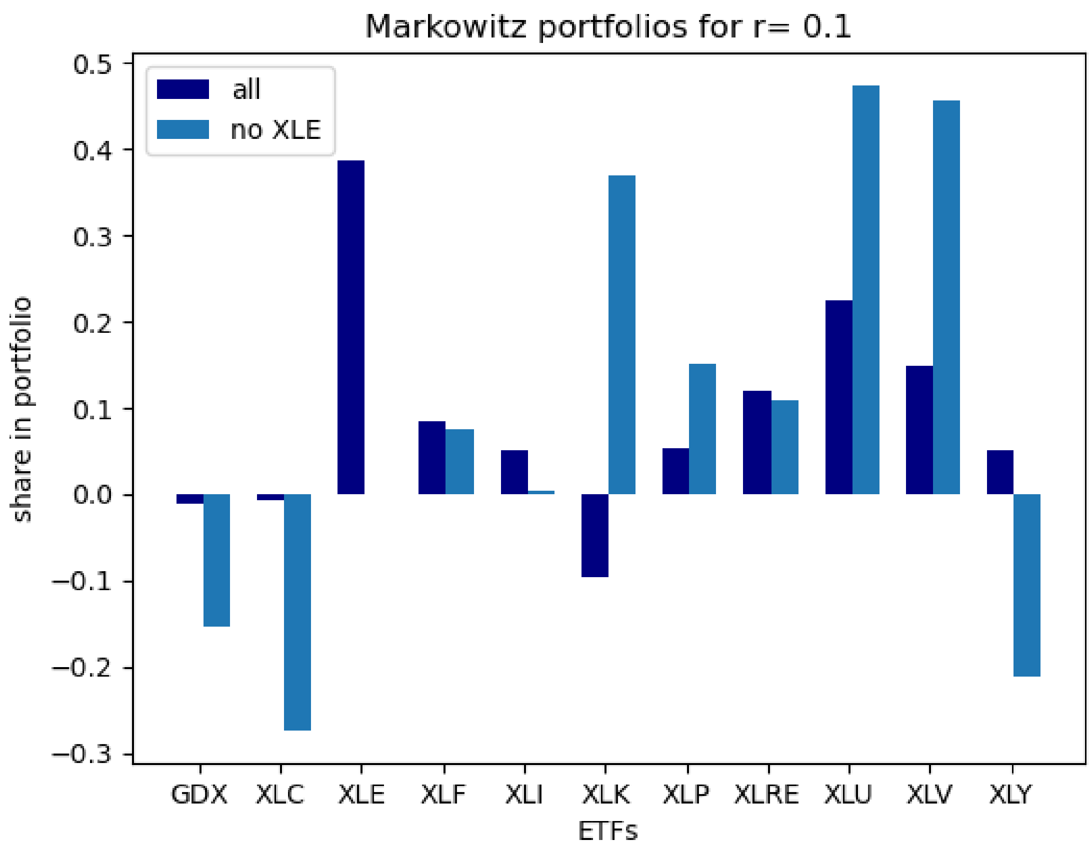



[25]:
```
plot_investments(green(['XLE','GDX'])[0],green(['XLE','GDX'])[1],green(['XLE','GDX'])[2],0.
↪1,green(['XLE','GDX'])[3])
```

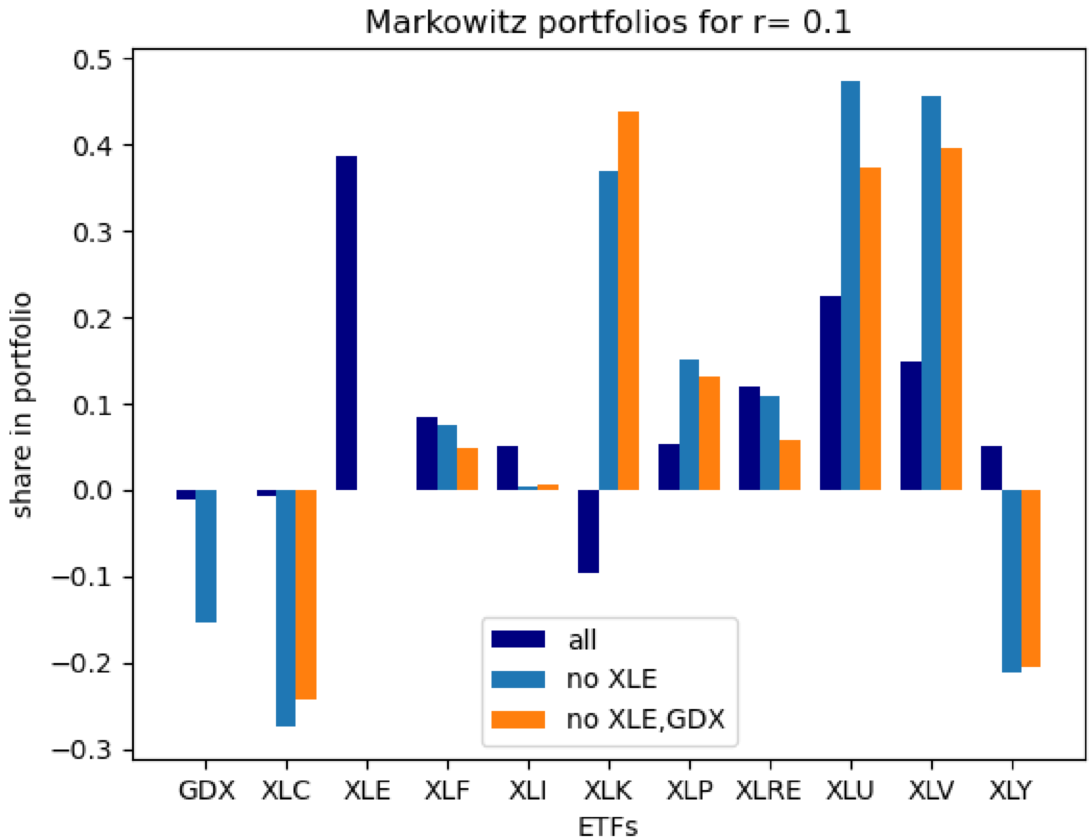



Lastly, the above portfolios are plotted together.

[26]:
```
plot_investments_together(0.1,[['XLE']])
```




[27]:
```
plot_investments_together(0.1, [['XLE'],['XLE','GDX']])
```




Another portfolio strategy to consider is investing based on ESG. This strategy takes into consideration how much a company is exposed to or involved in environmental, social, and governance issues, and how it deals with those. The issue with an ESG based portfolio is that comparable ESG rankings are hard to find, and they all weigh different ESG issues in different ways. There is no such thing as a universal ESG index or rating. Nevertheless, constructing a portfolio on such criteria is a more and more pressing issue, and by using a ranking instead of just eliminating for example oil and gold related companies, there may be a more nuanced portfolio opportunity. There might be companies in these sectors which already take steps to improve their environmental impact, and there may be companies in other sectors which do not fulfill our ESG requirements. We will use the ESG Risk Ratings from Morningstar Sustainalytics [Sus23]. These consider the general exposure of a company to certain ESG risks, as well as their management with those. The problem with this ranking is that it does not only give a raw index as to how much the company follows ESG criteria, but looks at how high the risk of the stock price of this company falling is in regards to ESG issues. If one only wants to select companies by for example how environmentally friendly they are, this index may not be the right choice. Nevertheless, it allows for comparison amongst all companies of all different sectors, which makes it the most powerful tool found for the computations below.

Since the ranking used provides an index of (almost) each individual company, we first need to compute the ESG ranking of our ETFs. Note that there is only one company for which there was no rating available, GE Vernova Inc. (GEV), as this company only entered the US stock market in April 2024, when it was founded through a merging of General Electric's Energy spin-offs. Due to the fact that many ETFs have holdings in a lot of different stocks, the data analysis is reduced to the holdings of each ETF which make up 1% or more of the assets held in the ETF. Furthermore, to handle the above issued missing ESG rating for the GEV stock the holdings in this stock will be ignored. The ETF XLI has 1.27% of its holdings in GEV stocks, and since this number is fairly small and there are no other ratings missing, we proceed by ignoring this holding.

First, the data for our ETFs is read in manually by setting up data frames which contain the stocks and share of these in the ETF, as well as their particular ESG rating. Note that a higher ranking means higher ESG risk, so we want to choose companies or ETFs with a low ESG ranking.

[28]:
```
#create ESG dataframe
data_gdx = pd.DataFrame([['NEM',0.1247,21.4],['AEM',0.0831,21.1],['GOLD',0.
↪0752,29.5],['WPM',0.0617,7.1],
            ['FNV',0.0570,6.7],['2899.HK',0.0442,36.7],['GFI',0.0412,23.
↪9],['AU',0.0388,24.4],
            ['KGC',0.0368,23.7],['NST.AX',0.0366,31.0],['RGLD',0.0319,8.
↪5],['PAAS',0.0294,24.2],
            ['AGI',0.0246,35.1],['HMY',0.0218,32.9],['EDV CN',0.0297,17.3],['EVN.
↪AX',0.0187,27.4],
            ['1818.HK',0.0180,48.3],['BVN',0.0169,41.4],['BTG',0.0135,24.
↪2],['HL',0.0130,32.6],
            ['EGO',0.0120,20.0],['OR',0.0120,10.
↪0]],columns=['ticker','share','ESG'])

data_xlc = pd.DataFrame([['META',0.2269,32.7],['GOOGL',0.1416,24.8],['GOOG',0.
↪1195,24.8],['NFLX',0.0484,15.5],
            ['TMUS',0.0476,25.0],['EA',0.0457,13.3],['T',0.0451,22.1],['VZ',0.
↪0423,18.2],
            ['CHTR',0.0389,23.7],['CMCSA',0.0384,22.6],['DIS',0.0368,15.
↪0],['TTWO',0.0341,16.0],
            ['OMC',0.0246,14.5],['WBD',0.0216,18.1],['LYV',0.0181,21.5],['IPG',0.
↪0155,8.7],
            ['NWSA',0.0141,10.0],['MTCH',0.0114,16.7],['FOXA',0.0114,12.2]],
                        columns=['ticker','share','ESG'])

data_xle = pd.DataFrame([['XOM',0.2629,41.3],['CVX',0.1725,35.3],['COP',0.
↪0833,33.1],['EOG',0.0452,34.4],
            ['SLB',0.0412,19.2],['MPC',0.0406,30.3],['PSX',0.0377,33.0],['VLO',0.
↪0330,30.5],
            ['WMB',0.0327,21.2],['OKE',0.0300,25.0],['OXY',0.0261,37.7],['HES',0.
↪0257,32.0],
            ['KMI',0.0243,17.8],['FANG',0.0219,37.3],['BKR',0.0210,19.
↪4],['HAL',0.0188,23.9],
            ['DVN',0.0187,31.6],['TRGP',0.0179,31.9],['CTRA',0.0128,32.
↪7],['MRO',0.0103,38.7]],
            columns=['ticker','share','ESG'])
```

```
data_xlf = pd.DataFrame([['BRK.B',0.1304,27.3],['JPM',0.1006,27.3],['V',0.
↪0767,15.0],['MA',0.0654,15.6],
            ['BAC',0.0483,24.3],['WFC',0.0372,35.9],['GS',0.0262,24.2],['SPGI',0.
↪0247,11.5],
            ['AXP',0.0230,18.3],['PGR',0.0216,19.8],['MS',0.0214,24.8],['C',0.
↪0202,22.1],
            ['BLK',0.0192,18.4],['SCHW',0.0189,23.4],['CB',0.0189,22.4],['MMC',0.
↪0185,21.5],
            ['FI',0.0157,17.7],['BX',0.0157,23.9],['ICE',0.0137,18.6],['CME',0.
↪0125,17.1],
            ['MCO',0.0116,14.6],['PYPL',0.0112,16.4],['USB',0.0108,24.
↪9],['PNC',0.0107,23.7],
            ['AON',0.0104,15.3],['AJG',0.0100,20.
↪5]],columns=['ticker','share','ESG'])

data_xli = pd.DataFrame([['GE',0.0475,34.5],['CAT',0.0443,29.1],['UBER',0.
↪0383,23.2],['HON',0.0373,27.1],
            ['RTX',0.0369,29.6],['UNP',0.0363,20.0],['ETN',0.0341,18.1],['BA',0.
↪0267,36.6],
            ['ADP',0.0267,15.1],['LMT',0.0265,28.6],['DE',0.0263,16.0],['UPS',0.
↪0262,18.8],
            ['WM',0.0202,18.8],['TT',0.0202,15.1],['TDG',0.0196,38.2],['GD',0.
↪0180,33.9],
            ['ITW',0.0175,22.8],['CSX',0.0174,21.1],['PH',0.0172,27.1],['EMR',0.
↪0163,22.8],
            ['NOC',0.0162,26.7],['CTAS',0.0161,17.0],['FDX',0.0153,19.
↪0],['MMM',0.0149,40.3],
            ['PCAR',0.0148,24.6],['CARR',0.0141,16.7],['NSC',0.0134,23.
↪3],['GEV',0.0127,np.nan],
            ['CPRT',0.0125,15.7],['JCI',0.0123,16.1],['URI',0.0114,15.
↪7],['LHX',0.0112,20.1],
            ['GWW',0.0107,16.0],['PAYX',0.0106,16.7],['PWR',0.0105,36.
↪7],['RSG',0.0104,18.5],
            ['OTIS',0.0104,18.6],['AME',0.0103,21.1],['VRSK',0.0103,16.
↪3],['CMI',0.0100,18.8],
            ['IR',0.0100,10.2]],columns=['ticker','share','ESG'])
#GEV only in market since April 2024, no rating yet

data_xlk = pd.DataFrame([['MSFT',0.2219,14.2],['AAPL',0.2156,16.8],['NVDA',0.
↪0595,13.2],['AVGO',0.0541,18.9],
            ['AMD',0.0252,13.3],['QCOM',0.0232,13.4],['ADBE',0.0228,14.
↪0],['CRM',0.0225,14.4],
            ['ORCL',0.0219,14.7],['AMAT',0.0192,11.6],['ACN',0.0185,8.
↪6],['CSCO',0.0183,12.9],
```

```
            ['TXN',0.0169,21.9],['INTU',0.0166,16.9],['MU',0.0153,18.6],['IBM',0.
↪0153,13.3],
            ['NOW',0.0145,15.0],['LRCX',0.0134,12.2],['INTC',0.0125,15.
↪3],['ADI',0.0110,18.1],
            ['KLAC',0.0108,16.2]],columns=['ticker','share','ESG'])

data_xlp = pd.DataFrame([['PG',0.1480,26.3],['COST',0.1435,26.2],['WMT',0.
↪1092,23.9],['KO',0.0907,24.2],
            ['PM',0.0468,26.8],['PEP',0.0444,20.8],['MDLZ',0.0378,21.4],['MO',0.
↪0341,32.2],
            ['CL',0.0340,25.0],['TGT',0.0282,17.1],['KMB',0.0201,27.5],['STZ',0.
↪0180,26.0],
            ['GIS',0.0161,25.8],['SYY',0.0154,15.3],['KVUE',0.0153,17.
↪0],['KDP',0.0152,23.7],
            ['MNST',0.0147,32.8],['KR',0.0142,23.2],['ADM',0.0138,31.6],['DG',0.
↪0119,21.2],
            ['HSY',0.0116,21.7],['CHD',0.0114,21.0],['EL',0.0112,24.0],['KHC',0.
↪0111,32.6]],
            columns=['ticker','share','ESG'])

data_xlre = pd.DataFrame([['PLD',0.1035,10.6],['AMT',0.0929,12.6],['EQIX',0.
↪0735,13.0],['WELL',0.0585,12.2],
             ['DLR',0.0490,12.2],['SPG',0.0489,12.5],['PSA',0.0474,11.7],['O',0.
↪0452,15.5],
             ['CCI',0.0429,12.0],['EXR',0.0342,14.1],['CSGP',0.0306,21.
↪1],['VICI',0.0298,13.9],
             ['AVB',0.0293,8.1],['CBRE',0.0275,6.3],['IRM',0.0266,12.4],['EQR',0.
↪0240,11.4],
             ['WY',0.0215,15.2],['SBAC',0.0211,9.7],['INVH',0.0210,16.
↪0],['VTR',0.0207,11.2],
             ['ARE',0.0187,13.1],['ESS',0.0185,11.6],['MAA',0.0168,11.
↪7],['DOC',0.0141,11.3],
             ['HST',0.0131,12.9],['KIM',0.0129,10.4],['UDR',0.0126,12.
↪9],['CPT',0.0120,14.2],
             ['REG',0.0104,11.3]],columns=['ticker','share','ESG'])
#WELL data taken from Welltower OP LLC

data_xlu = pd.DataFrame([['NEE',0.1407,24.9],['SO',0.0814,28.1],['DUK',0.0730,26.
↪8],['CEG',0.0672,28.3],
            ['SRE',0.0449,23.2],['AEP',0.0437,22.1],['D',0.0394,28.0],['PCG',0.
↪0360,30.4],
            ['PEG',0.0347,21.2],['EXC',0.0329,18.8],['ED',0.0296,21.1],['VST',0.
↪0280,29.3],
            ['XEL',0.0279,26.3],['EIX',0.0261,24.0],['AWK',0.0240,18.7],['WEC',0.
↪0237,22.8],
```

```
['DTE',0.0217,31.3],['ETR',0.0213,24.9],['PPL',0.0196,26.9],['ES',0.
↪0192,18.1],
            ['CNP',0.0187,24.8],['FE',0.0187,28.0],['AEE',0.0177,26.0],['NRG',0.
↪0172,34.2],
            ['CMS',0.0166,20.3],['LNT',0.0123,17.1],['AES',0.0119,23.
↪5],['EVRG',0.0115,29.2],
            ['NI',0.0111,20.6]],columns=['ticker','share','ESG'])

data_xlv = pd.DataFrame([['LLY',0.1302,23.6],['UNH',0.0826,17.0],['JNJ',0.
↪0655,21.3],['MRK',0.0605,21.1],
            ['ABBV',0.0560,26.8],['TMO',0.0397,12.7],['ABT',0.0341,22.
↪2],['AMGN',0.0305,22.7],
            ['DHR',0.0301,10.7],['PFE',0.0289,17.7],['ISRG',0.0281,19.
↪5],['ELV',0.0232,10.0],
            ['VRTX',0.0224,19.3],['SYK',0.0213,23.9],['BSX',0.0208,22.
↪1],['REGN',0.0203,16.8],
            ['MDT',0.0196,22.2],['CI',0.0182,13.0],['GILD',0.0157,21.7],['BMY',0.
↪0154,21.2],
            ['MCK',0.0146,13.4],['ZTS',0.0142,15.1],['CVS',0.0141,18.7],['BDX',0.
↪0127,23.7],
            ['HCA',0.0124,27.8]],columns=['ticker','share','ESG'])

data_xly = pd.DataFrame([['AMZN',0.2428,29.3],['TSLA',0.1350,24.7],['HD',0.
↪0944,12.8],['MCD',0.0389,25.8],
            ['BKNG',0.0382,17.2],['LOW',0.0360,11.8],['TJX',0.0347,15.
↪5],['NKE',0.0321,18.7],
            ['SBUX',0.0249,22.3],['CMG',0.0243,20.0],['ABNB',0.0178,23.
↪7],['ORLY',0.0175,11.8],
            ['MAR',0.0165,20.3],['GM',0.0151,28.3],['HLT',0.0151,16.2],['AZO',0.
↪0143,11.0],
            ['ROST',0.0136,17.2],['F',0.0128,23.0],['DHI',0.0116,21.6],['YUM',0.
↪0104,20.5],
            ['LEN',0.0101,26.0]],columns=['ticker','share','ESG'])
```

Next, we will compute the ESG rating of each considered ETF by first normalising the shares such that their sum is 1, then multiplying the adjusted share value with the respective ESG value, and then summing these results up.

[29]:
```
def ESG_value(df):
    total = df.dropna()['share'].sum()
    #ignore missing values instead of computing as if ESG value=0
    df.share *=1/total
    df['esg_rel'] = df.share*df.ESG
    ESG = df['esg_rel'].sum()
    return ESG
```

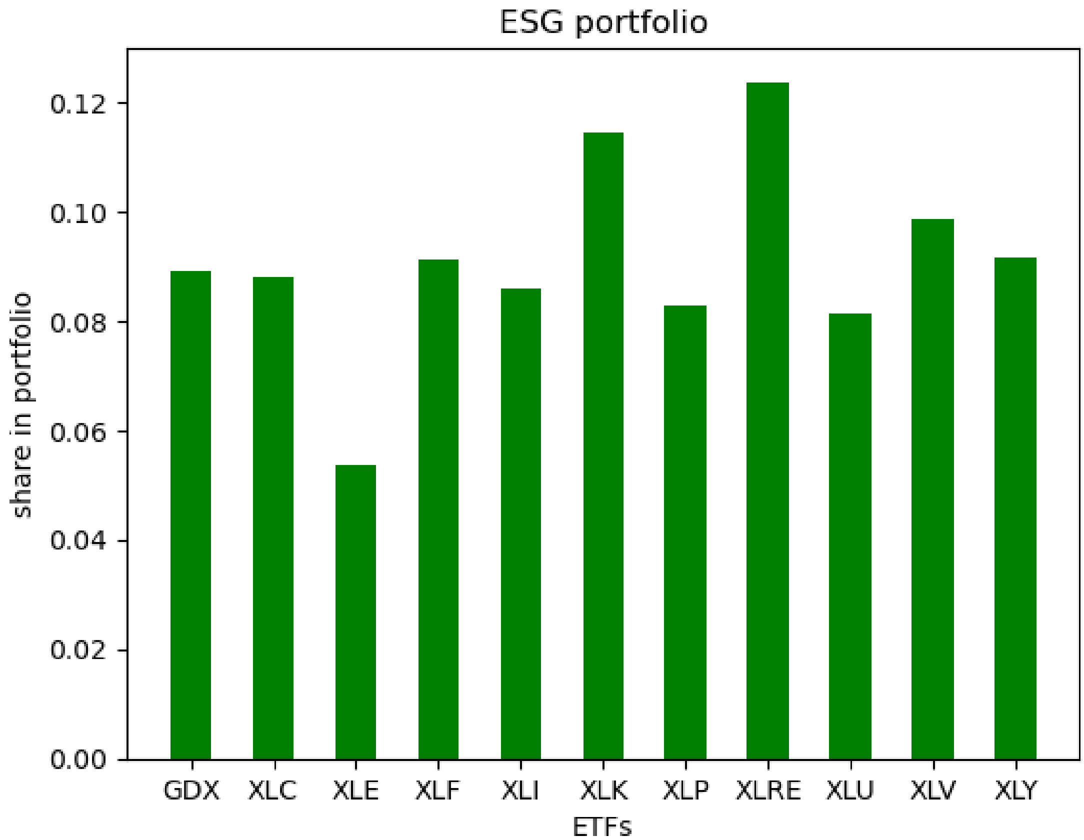
The ESG ranking used starts at 0 and is open end to the positive numbers, but according to their documentation [Sus23], 95% of the companies have a rating below 50. In fact, the highest values in our use is 48.3. In order to invest less, and not more, into companies with higher ESG rating, we thus mirror our ratings at value 50. Then we proceed to compute our portfolio by investing according to the ETFs ESG ratings.

[30]:
```
def ESG_portf(dfs):
    val = []
    for i in range(len(dfs)):
        val.append(50-ESG_value(dfs[i]))
    total = sum(val)
    val = [x*1/total for x in val]
    return val
```

Looking at this portfolio, we see that there are no negative values appearing. This is due to the fact that the rating only has positive values, and by the way we mirrored them we preserved this property. One could consider shorting, but there is no obvious natural value to select as a limit on the ESG rating according to which one would choose shorting. Thus, we will not consider this here.

[31]:
```
def plot_ESG_portf(dfs):
    values = ESG_portf(dfs)
    colors = ['g' if m > 0 else 'r' for m in values]
    plt.bar(tickers,values,width=.5,color=colors)
    plt.xlabel('ETFs')
    plt.ylabel('share in portfolio')
    plt.title('ESG portfolio')
    plt.show()
```

[32]:
```
dfs = [data_gdx,data_xlc,data_xle,data_xlf,data_xli,data_xlk,data_xlp,data_xlre,
       data_xlu,data_xlv,data_xly]
plot_ESG_portf(dfs)
```




Now, after obtaining our ESG portfolio, we want to know its expected return and risk.

[33]:
```
def ESG_ret_risk(dfs):
    av = av_ret.flatten().tolist()
    val = []
    for i in range(len(dfs)):
        val.append(av[i]*ESG_portf(dfs)[i])
    matr = np.array(ESG_portf(dfs))
    s_2 = matr.reshape((1,11)) @ cov_all @ matr.reshape((11,1))
    s = np.sqrt(s_2.item())
    return sum(val),s
```

[34]:
```
print(ESG_ret_risk(dfs))
```

```
(0.04967839434089261, 7.9739154241304355)
```

Let us now look at how this portfolio performs in comparison to the Markowitz portfolios computed before.

[35]:
```
#plot both efficient portfolio fronts together
def plot_with_ESG(lists,dfs):
```

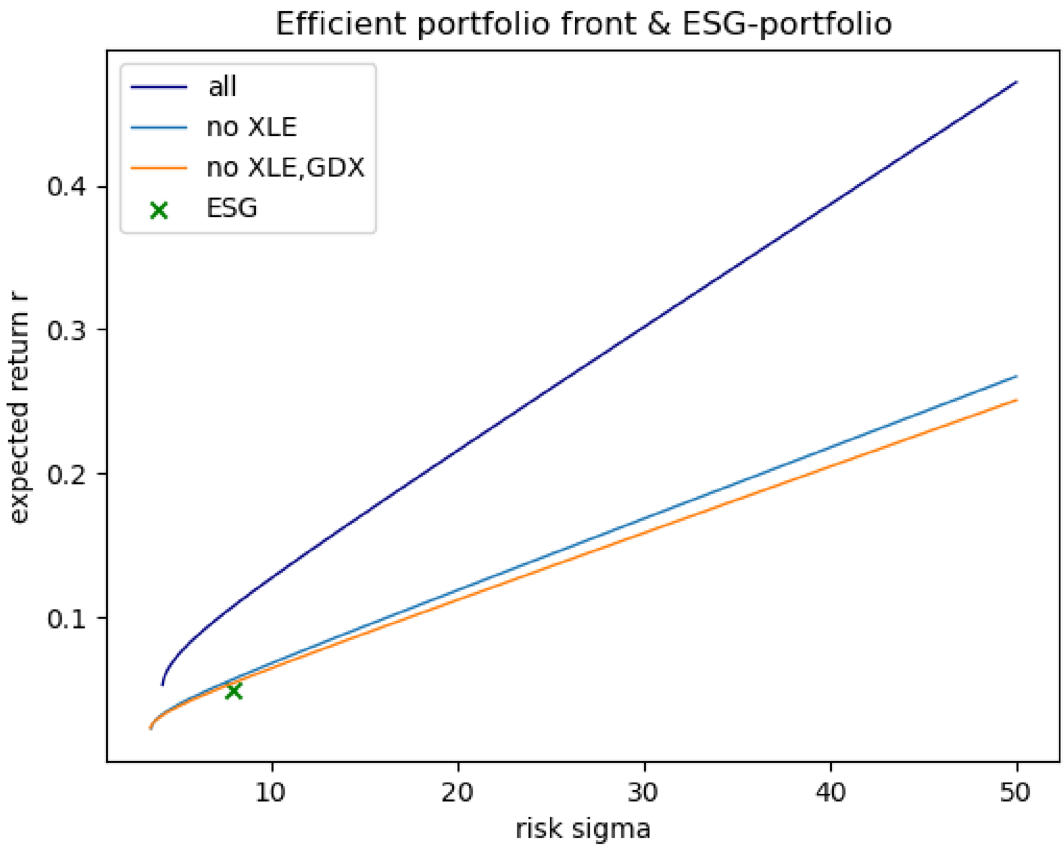
```
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    s = np.linspace(0,50,num=1000)
    plt.plot(s,effport(a1,b1,c1,s),color='navy',linewidth=1,label='all')
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        string = ','.join(str(elm) for elm in lists[i])
        plt.plot(s,effport(a,b,c,s),linewidth=1,label='no '+string)
    plt.
↪scatter(ESG_ret_risk(dfs)[1],ESG_ret_risk(dfs)[0],c='green',marker='x',label='ESG')
    plt.title('Efficient portfolio front & ESG-portfolio')
    plt.xlabel('risk sigma')
    plt.ylabel('expected return r')
    plt.legend()
    plt.show()
```

[36]:

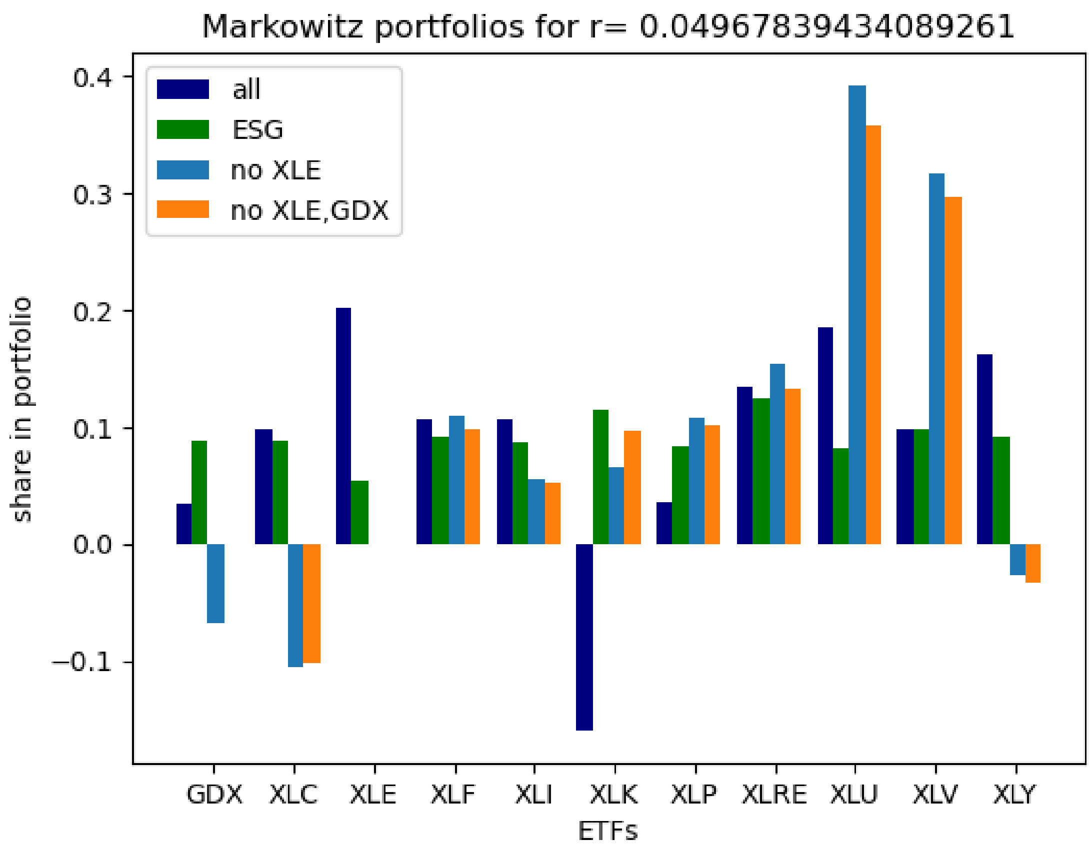
```
#plot portfolios together
def plot_with_ESG_investments(r,lists,dfs):
    n=len(lists)+2
    values1 = list(markowitz_portf(av_ret,cov_all,r,d)[1].flat)
    x_axis = np.arange(len(tickers))
    plt.bar(x_axis - (len(lists)+1)/8,values1,color='navy',label='all',width=1/
↪(n+1))
    values2 = ESG_portf(dfs)
    plt.bar(x_axis -(len(lists)+1)/8+1/
↪(n+1),values2,label='ESG',color='green',width=1/(n+1))
    for i in range(len(lists)):
        values =␣
↪list(markowitz_portf(green(lists[i])[1],green(lists[i])[2],r,green(lists[i])[3])[1].
↪flat)
        for j in range(len(green(lists[i])[4])):
            values.insert(green(lists[i])[4][j],0)
        string = ','.join(str(elm) for elm in lists[i])
        plt.bar(x_axis - (len(lists)+1)/8+(i+2)*1/(n+1),values,label='no␣
↪'+string,width=1/(n+1))
    plt.xticks(x_axis,tickers)
    plt.xlabel('ETFs')
    plt.ylabel('share in portfolio')
    plt.title('Markowitz portfolios for r= '+str(r))
    plt.legend()
```

```
    plt.show()
```

[37]:
```
plot_with_ESG([['XLE'],['XLE','GDX']], dfs)
```




[38]:
```
plot_with_ESG_investments(ESG_ret_risk(dfs)[0],[['XLE'],['XLE','GDX']],dfs)
```




The losses in expected return at same risk and the increase in risk at same expected return compared to the Markowitz portfolios found before are computed by the following functions.

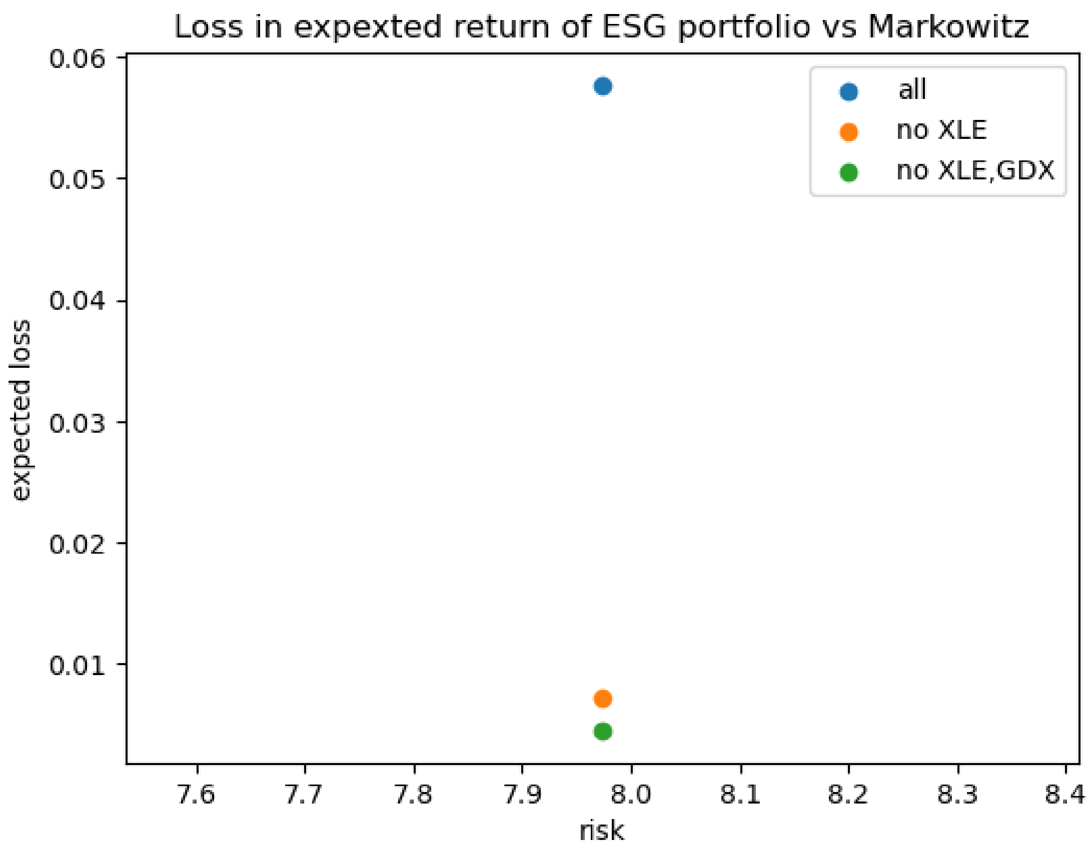
[39]:
```
#loss functions
def loss_ESG_s(lists,dfs):
    s = ESG_ret_risk(dfs)[1]
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    val = []
    val.append(effport(a1,b1,c1,s)-ESG_ret_risk(dfs)[0])
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        val.append(effport(a,b,c,s)-ESG_ret_risk(dfs)[0])
    return val
```

```
def loss_ESG_r(lists,dfs):
    r = ESG_ret_risk(dfs)[0]
    a1 = constants(av_ret, cov_all,d)[0]
    b1 = constants(av_ret, cov_all,d)[1]
    c1 = constants(av_ret, cov_all,d)[2]
    losses = []
    losses.append(ESG_ret_risk(dfs)[1]-inverse(a1,b1,c1,r))
    for i in range(len(lists)):
        a =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[0]
        b =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[1]
        c =␣
↪constants(green(lists[i])[1],green(lists[i])[2],green(lists[i])[3])[2]
        losses.append(ESG_ret_risk(dfs)[1]-inverse(a,b,c,r))
    return losses
```

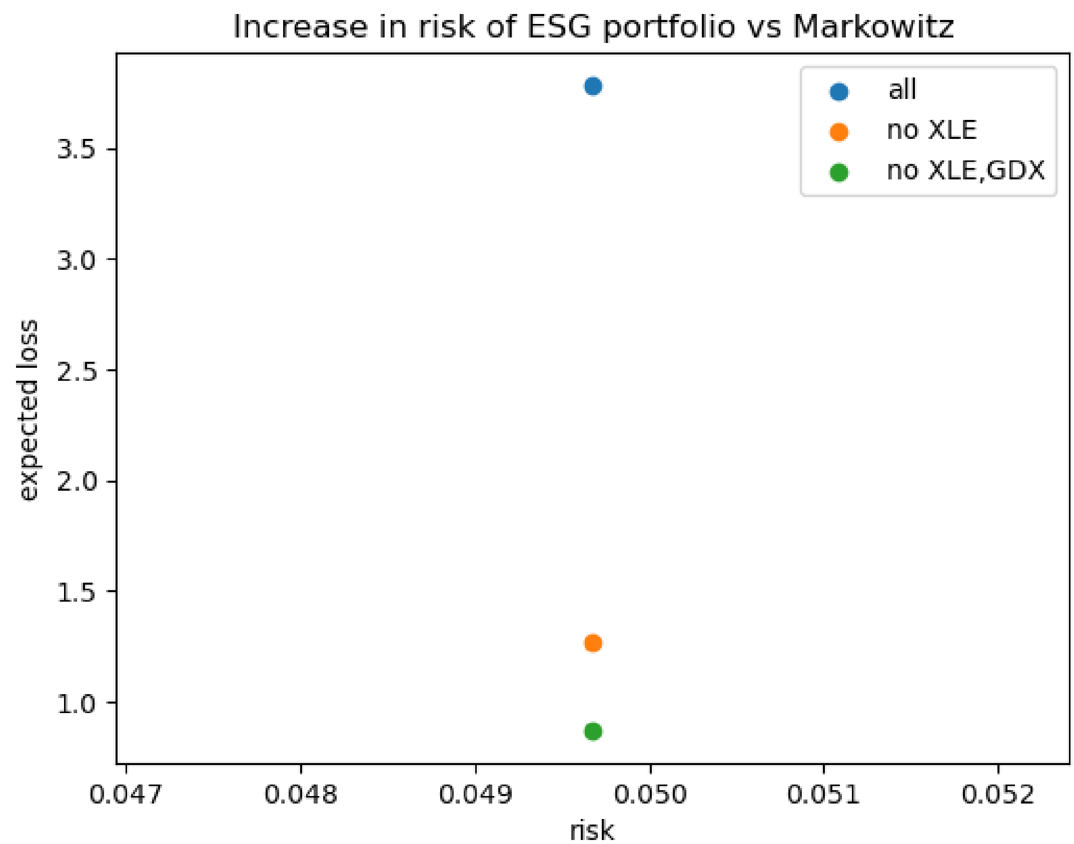
We visualise those losses via the functions below.

[40]:
```
#plot loss functions
def plot_loss_ESG_s(lists,dfs):
    s = ESG_ret_risk(dfs)[1]
    plt.scatter(s,loss_ESG_s(lists,dfs)[0],marker='o',label='all')
    for i in range(1,len(loss_ESG_s(lists,dfs))):
        string = ','.join(str(elm) for elm in lists[i-1])
        plt.scatter(s,loss_ESG_s(lists,dfs)[i],marker='o',label='no '+string)
    plt.title('Loss in expexted return of ESG portfolio vs Markowitz')
    plt.xlabel('risk')
    plt.ylabel('expected loss')
    plt.legend()
    plt.show()

def plot_loss_ESG_r(lists,dfs):
    r = ESG_ret_risk(dfs)[0]
    plt.scatter(r,loss_ESG_r(lists,dfs)[0],marker='o',label='all')
    for i in range(1,len(loss_ESG_r(lists,dfs))):
        string = ','.join(str(elm) for elm in lists[i-1])
        plt.scatter(r,loss_ESG_r(lists,dfs)[i],marker='o',label='no '+string)
    plt.title('Increase in risk of ESG portfolio vs Markowitz')
    plt.xlabel('risk')
    plt.ylabel('expected loss')
    plt.legend()
    plt.show()
```

[41]:
```
plot_loss_ESG_s([['XLE'],['XLE','GDX']],dfs)
```




[42]:
```
plot_loss_ESG_r([['XLE'],['XLE','GDX']],dfs)
```




As we can see, we have very low loss in expected return of the ESG portfolio compared to the Markowitz portfolio without investment in the XLE ETF.

## References

[Sus23] Morningstar Sustainalytics. *ESG Risk Ratings Methodology*. 2023. URL: https://connect.sustainalytics.com/esg-risk-ratings-methodology (visited on 06/25/2024).