Finance Python

May 31, 2021

[1]:
```
import numpy as np
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
```

```
/Users/alessandrosollazzo/opt/anaconda3/lib/python3.7/site-
packages/pandas_datareader/compat/__init__.py:7: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at
pandas.testing instead.
  from pandas.util.testing import assert_frame_equal
```

[2]:
```
import panda as pd
```

```
        ␣
 ↪---------------------------------------------------------------------------

        ModuleNotFoundError                       Traceback (most recent call␣
 ↪last)

        <ipython-input-2-f542dbfa5144> in <module>
    ----> 1 import panda as pd


        ~/opt/anaconda3/lib/python3.7/site-packages/panda/__init__.py in <module>
    ----> 1 from request import PandaRequest
          2 from models import Video, Cloud, Encoding, Profile, Notifications,␣
 ↪PandaDict
          3 from models import GroupRetriever, SingleRetriever
          4 from models import PandaError
          5 from upload_session import UploadSession


        ModuleNotFoundError: No module named 'request'
```

[3]:
```
import pandas as pd
```

[1]:
```
import pandas_datareader as web
```

```
/Users/alessandrosollazzo/opt/anaconda3/lib/python3.7/site-
packages/pandas_datareader/compat/__init__.py:7: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at
pandas.testing instead.
  from pandas.util.testing import assert_frame_equal
```

[6]:
```
PG=web.DataReader ('PG', data_source='yahoo', start='1995-1-1')
```

[7]:
```
PG.head()
```

[7]:
```
                High       Low      Open     Close     Volume  Adj Close
Date
1995-01-03  15.62500  15.43750  15.46875  15.59375  3318400.0   6.320252
1995-01-04  15.65625  15.31250  15.53125  15.46875  2218800.0   6.269589
1995-01-05  15.43750  15.21875  15.37500  15.25000  2319600.0   6.180927
1995-01-06  15.40625  15.15625  15.15625  15.28125  3438000.0   6.193593
1995-01-09  15.40625  15.18750  15.34375  15.21875  1795200.0   6.168259
```

[7]:
```
PG.tail()
```

[7]:
```
                  High         Low        Open       Close      Volume  \
Date
2020-06-15  117.279999  113.760002  114.550003  116.690002   8786000.0
2020-06-16  118.970001  116.930000  118.529999  118.129997   8165200.0
2020-06-17  119.110001  117.440002  118.389999  117.930000   6320800.0
2020-06-18  119.959999  117.370003  117.459999  119.279999   6274400.0
2020-06-19  121.820000  118.830002  120.489998  118.919998  17506200.0

             Adj Close
Date
2020-06-15  116.690002
2020-06-16  118.129997
2020-06-17  117.930000
2020-06-18  119.279999
2020-06-19  118.919998
```

[11]:
```
PG['simple_return']=(PG['Adj Close']/PG['Adj Close'].shift(1))-1
```

[13]:
```
print PG['simple_return']
```

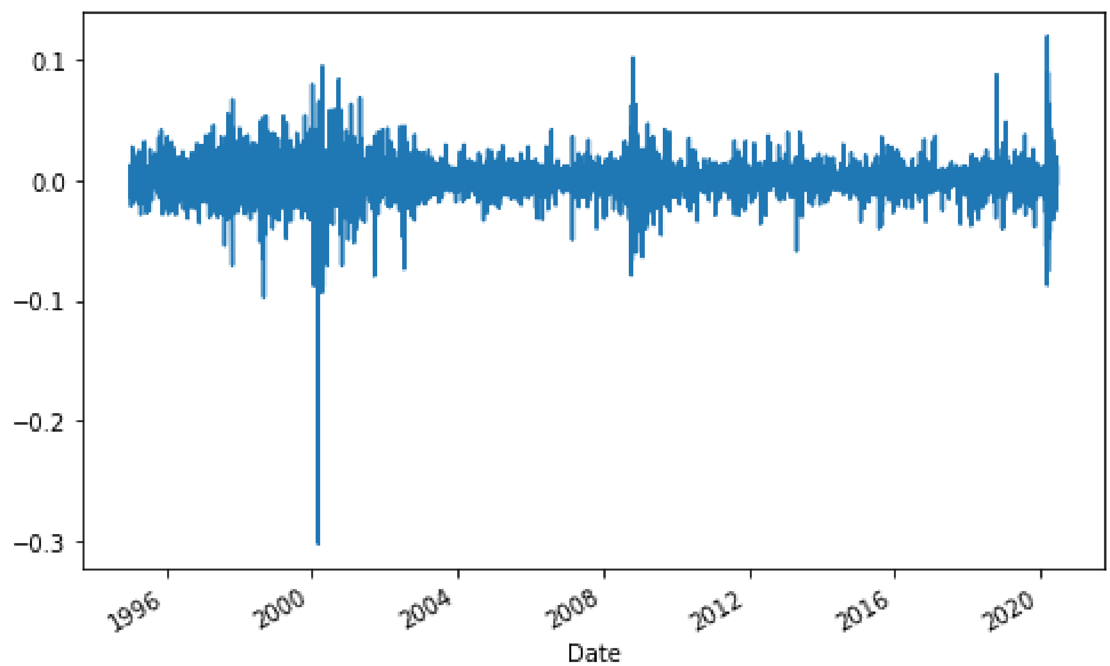
```
  File "<ipython-input-13-488f3f1a7a51>", line 1
    print PG['simple_return']
           ^

SyntaxError: Missing parentheses in call to 'print'. Did you mean
↪print(PG['simple_return'])?
```

[14]:
```
print(PG['simple_return'])
```

```
Date
1995-01-03         NaN
1995-01-04   -0.008016
1995-01-05   -0.014141
1995-01-06    0.002049
1995-01-09   -0.004090
                 …
2020-06-15    0.009254
2020-06-16    0.012340
2020-06-17   -0.001693
2020-06-18    0.011447
2020-06-19   -0.003018
Name: simple_return, Length: 6412, dtype: float64
```

[15]:
```
PG['simple_return'].plot(figsize=(8, 5))
plt.show()
```



[16]:
```
avg_returns_d=PG['simple_return'].mean()
avg_returns_d
```

[16]: 0.0005187078858490102

[17]:
```python
avg_returns_a=PG['simple_return'].mean()*250
avg_returns_a
```

[17]: 0.12967697146225254

[18]:
```python
print str(round(avg_returns_a,5)*100)+'%'
```

```
  File "<ipython-input-18-90b7bbbf47cb>", line 1
    print str(round(avg_returns_a,5)*100)+'%'
            ^
SyntaxError: invalid syntax
```

[20]:
```python
print str(round(avg_returns_a, 5)*100)+ '%'
```

```
  File "<ipython-input-20-274dad27c35b>", line 1
    print str(round(avg_returns_a, 5)*100)+ '%'
            ^
SyntaxError: invalid syntax
```

[21]:
```python
round ((avg_returns_a,2)*100)+ '%'
```

```
␣
↪---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call␣
↪last)

    <ipython-input-21-80bdac8ab425> in <module>
----> 1 round ((avg_returns_a,2)*100)+ '%'


    TypeError: type tuple doesn't define __round__ method
```

[22]:
```python
round (avg_returns_a,2)+ '%'
```

```
␣
↪---------------------------------------------------------------------------
```

```
    TypeError                                 Traceback (most recent call␣
↪last)

    <ipython-input-22-f2e2e74bb8fc> in <module>
  ----> 1 round (avg_returns_a,2)+ '%'


    TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

[24]:
```
str(round (avg_returns_a,2)*100)+ '%'
```

[24]: '13.0%'

[29]:
```
str(round (avg_returns_a,4)*100)+ '%'
```

[29]: '12.97%'

[30]:
```
PG.head()
```

[30]:
```
                High        Low       Open      Close     Volume  Adj Close  \
Date
1995-01-03  15.62500  15.43750  15.46875  15.59375  3318400.0   8.313478
1995-01-04  15.65625  15.31250  15.53125  15.46875  2218800.0   8.246841
1995-01-05  15.43750  15.21875  15.37500  15.25000  2319600.0   8.130219
1995-01-06  15.40625  15.15625  15.15625  15.28125  3438000.0   8.146880
1995-01-09  15.40625  15.18750  15.34375  15.21875  1795200.0   8.113561

            simple return  simple_return
Date
1995-01-03            NaN            NaN
1995-01-04      -0.008016      -0.008016
1995-01-05      -0.014141      -0.014141
1995-01-06       0.002049       0.002049
1995-01-09      -0.004090      -0.004090
```

[31]:
```
PG['log_return']=np.log(PG['Adj Close']/PG['Adj Close'].shift(1))
```

[32]:
```
print PG['log_return']
```

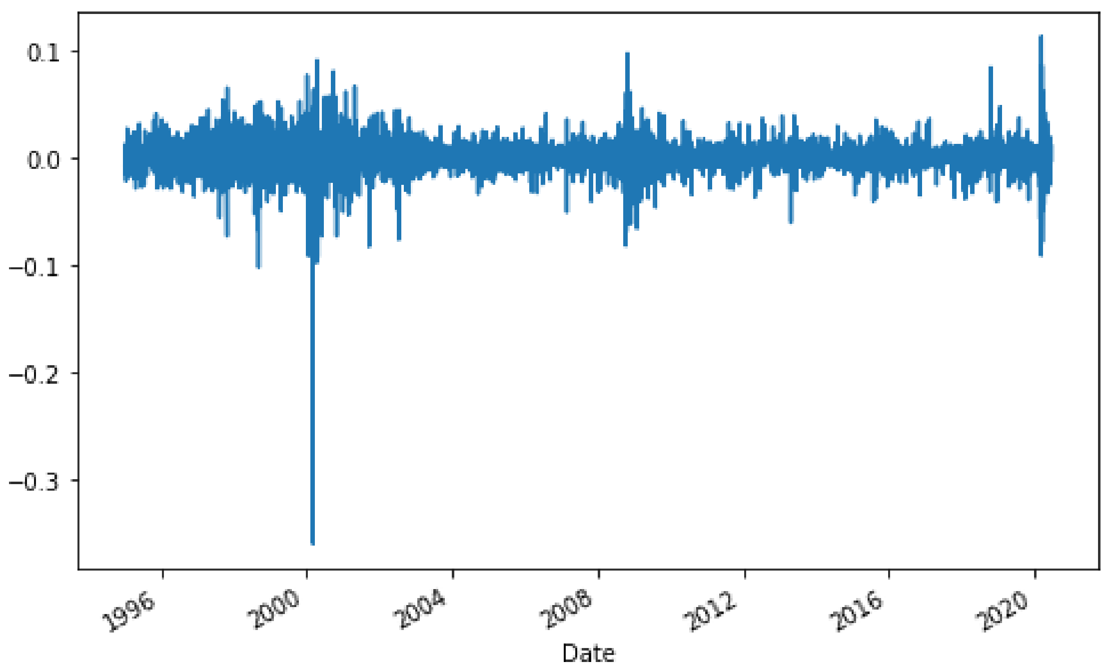
```
      File "<ipython-input-32-21a66722a3eb>", line 1
    print PG['log_return']
           ^

  SyntaxError: Missing parentheses in call to 'print'. Did you mean␣
↪print(PG['log_return'])?
```

[33]:
```
print(PG['log_return'])
```

```
Date
1995-01-03         NaN
1995-01-04   -0.008048
1995-01-05   -0.014242
1995-01-06    0.002047
1995-01-09   -0.004098
                 …
2020-06-15    0.009212
2020-06-16    0.012265
2020-06-17   -0.001694
2020-06-18    0.011382
2020-06-19   -0.003023
Name: log_return, Length: 6412, dtype: float64
```

[36]:
```
PG['log_return'].plot(figsize=(8, 5))
plt.show()
```



[37]:
```
log_return_d=PG['log_return'].mean()
log_return_d
```

[37]: 0.00041500122804857776

[38]:
```
log_return_a=PG['log_return'].mean()*250
log_return_a
```

[38]:
```
0.10375030701214444
```

[39]:
```
str(round(log_return_a, 2)*100)+'%'
```

[39]:
```
'10.0%'
```

[40]:
```
PG.head()
```

[40]:
```
                High       Low      Open     Close     Volume  Adj Close  \
Date
1995-01-03  15.62500  15.43750  15.46875  15.59375  3318400.0   8.313478
1995-01-04  15.65625  15.31250  15.53125  15.46875  2218800.0   8.246841
1995-01-05  15.43750  15.21875  15.37500  15.25000  2319600.0   8.130219
1995-01-06  15.40625  15.15625  15.15625  15.28125  3438000.0   8.146880
1995-01-09  15.40625  15.18750  15.34375  15.21875  1795200.0   8.113561

            simple return  simple_return  log_return
Date
1995-01-03            NaN            NaN         NaN
1995-01-04      -0.008016      -0.008016   -0.008048
1995-01-05      -0.014141      -0.014141   -0.014242
1995-01-06       0.002049       0.002049    0.002047
1995-01-09      -0.004090      -0.004090   -0.004098
```

[41]:
```
tickers=['PG', 'MSFT', 'F','GE']
mydata = pd.DataFrame()
for t in tickers:
    mydata[t] = web.DataReader(t, data_source='yahoo', start='1995-1-1')['Adj␣
↪Close']
```

[42]:
```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6412 entries, 1995-01-03 to 2020-06-19
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   PG      6412 non-null   float64
 1   MSFT    6412 non-null   float64
 2   F       6412 non-null   float64
 3   GE      6412 non-null   float64
dtypes: float64(4)
memory usage: 250.5 KB
```

[43]:
```
mydata.head()
```

[43]:

Date	PG	MSFT	F	GE
1995-01-03	8.313478	2.406834	3.262354	2.729230
1995-01-04	8.246841	2.424330	3.350130	2.729230
1995-01-05	8.130219	2.384342	3.320871	2.735919
1995-01-06	8.146880	2.424330	3.320871	2.722540
1995-01-09	8.113561	2.409335	3.379390	2.695783

[44]:
```
mydata.tail()
```

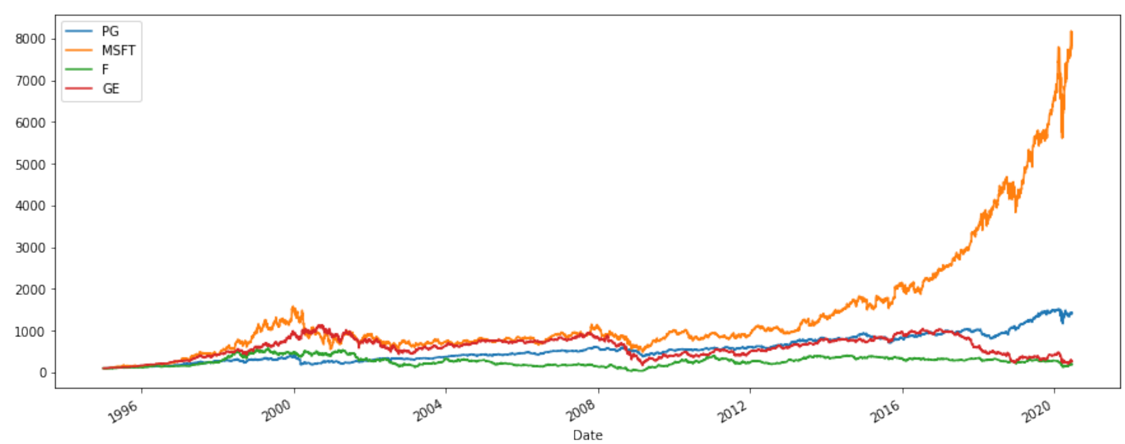
[44]:

Date	PG	MSFT	F	GE
2020-06-15	116.690002	188.940002	6.50	7.24
2020-06-16	118.129997	193.570007	6.55	7.47
2020-06-17	117.930000	194.240005	6.33	7.24
2020-06-18	119.279999	196.320007	6.33	7.28
2020-06-19	118.919998	195.149994	6.23	7.15

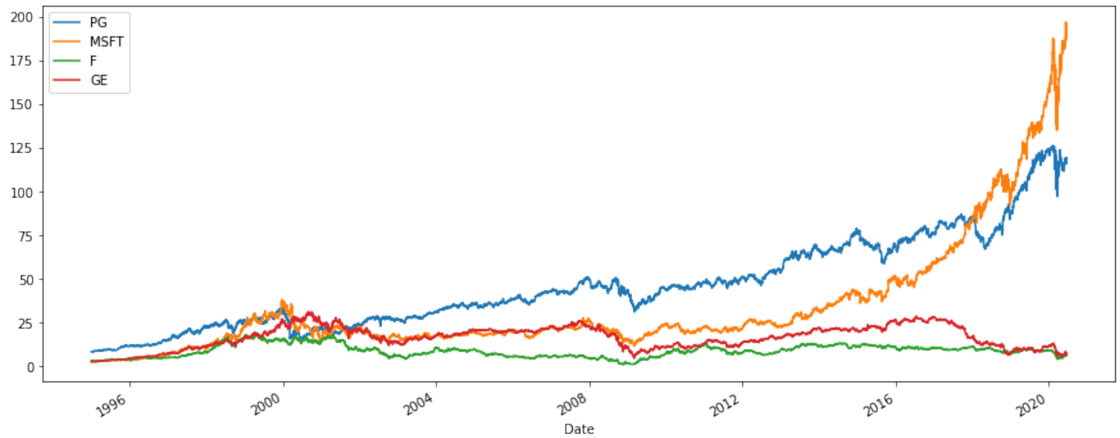
[45]:
```
mydata.iloc[0]
```

[45]:
```
PG      8.313478
MSFT    2.406834
F       3.262354
GE      2.729230
Name: 1995-01-03 00:00:00, dtype: float64
```

[46]:
```
(mydata/mydata.iloc[0]*100).plot(figsize=(15, 6));
plt.show()
```



[47]:
```
mydata.plot(figsize=(15,6))
plt.show()
```



[48]:
```
mydata.loc[1995-01-03]
```

```
  File "<ipython-input-48-a57165a38487>", line 1
    mydata.loc[1995-01-03]
                     ^
SyntaxError: invalid token
```

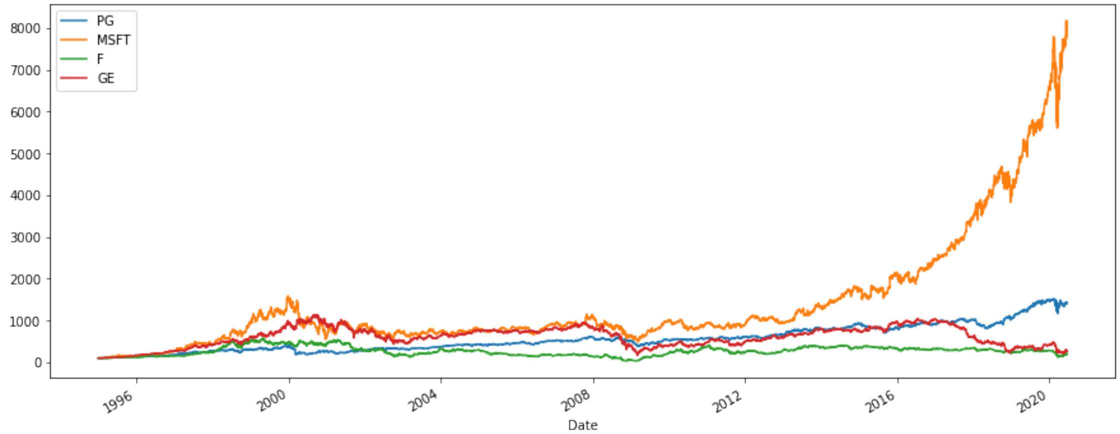
[49]:
```
mydata.loc['1995-01-03']
```

[49]:
```
PG      8.313478
MSFT    2.406834
F       3.262354
GE      2.729230
Name: 1995-01-03 00:00:00, dtype: float64
```

[50]:
```
mydata.iloc[0]
```

[50]:
```
PG      8.313478
MSFT    2.406834
F       3.262354
GE      2.729230
Name: 1995-01-03 00:00:00, dtype: float64
```

[51]:
```
(mydata/mydata.iloc[0]*100).plot(figsize=(15, 6));
plt.show()
```



[52]:
```python
returns=(mydata/mydata.shift(1))-1
```

[53]:
```python
returns.head
```

[53]:
```
<bound method NDFrame.head of                     PG      MSFT         F        GE
Date
1995-01-03        NaN       NaN       NaN       NaN
1995-01-04  -0.008016  0.007269  0.026906  0.000000
1995-01-05  -0.014141 -0.016495 -0.008734  0.002451
1995-01-06   0.002049  0.016771  0.000000 -0.004890
1995-01-09  -0.004090 -0.006185  0.017622 -0.009828
…                 …         …         …         …
2020-06-15   0.009254  0.006392  0.006192 -0.001379
2020-06-16   0.012340  0.024505  0.007692  0.031768
2020-06-17  -0.001693  0.003461 -0.033588 -0.030790
2020-06-18   0.011447  0.010708  0.000000  0.005525
2020-06-19  -0.003018 -0.005960 -0.015798 -0.017857

[6412 rows x 4 columns]>
```

[54]:
```python
returns.head()
```

[54]:
```
                  PG      MSFT         F        GE
Date
1995-01-03       NaN       NaN       NaN       NaN
1995-01-04 -0.008016  0.007269  0.026906  0.000000
1995-01-05 -0.014141 -0.016495 -0.008734  0.002451
1995-01-06  0.002049  0.016771  0.000000 -0.004890
1995-01-09 -0.004090 -0.006185  0.017622 -0.009828
```

[57]:
```python
weights = np.array([0.25, 0.25, 0.25, 0.25])
```

[58]:
```
np.dot(returns, weights)
```

[58]:
```
array([         nan,  0.00653988, -0.00922976, …, -0.01565235,
        0.0069202 , -0.01065819])
```

[59]:
```
annual_returns = returns.mean()*250
```

[60]:
```
annual_returns
```

[60]:
```
PG      0.129677
MSFT    0.221502
F       0.105312
GE      0.088126
dtype: float64
```

[61]:
```
np.dot(returns, weights)
```

[61]:
```
array([         nan,  0.00653988, -0.00922976, …, -0.01565235,
        0.0069202 , -0.01065819])
```

[63]:
```
np.dot(annual_returns, weights)
```

[63]:
```
0.1361542098903807
```

[64]:
```
pfolio_1=str(round(np.dot(annual_returns, weights),5)*100) + '%'
```

[66]:
```
print (pfolio_1)
```

```
13.614999999999998%
```

[67]:
```
weight_2=np.array([0.4,0.4,0.15,0.05])
```

[69]:
```
pfolio_2= str(round(np.dot(annual_returns, weight_2), 5)*100)+'%'
```

[71]:
```
print (pfolio_1)
print (pfolio_2)
```

```
13.614999999999998%
16.067%
```

[74]:
```
tickers = ['^GSPC','^IXIC','^GDAXI']
ind_data = pd.DataFrame()
for t in tickers:
    ind_data[t] = web.DataReader(t, data_source = 'yahoo',␣
 ↪start='1997-1-1')['Adj Close']
```

[75]:
```
ind_data.head()
```

[75]:

Date	^GSPC	^IXIC	^GDAXI
1997-01-02	737.010010	1280.699951	2820.810059
1997-01-03	748.030029	1310.680054	2863.260010
1997-01-06	747.650024	1316.400024	2890.199951
1997-01-07	753.229980	1327.729980	2876.340088
1997-01-08	748.409973	1320.349976	2904.080078

[76]:
```
ind_data.tail()
```

[76]:

Date	^GSPC	^IXIC	^GDAXI
2020-06-15	3066.590088	9726.019531	11911.349609
2020-06-16	3124.739990	9895.870117	12315.660156
2020-06-17	3113.489990	9910.530273	12382.139648
2020-06-18	3115.340088	9943.049805	12281.530273
2020-06-19	3097.739990	9946.120117	12330.759766

[77]:
```
(ind_data/ind_data.iloc[0]*100).plot(figsize = (15, 6));
plt.show()
```

[78]:
```
ind_returns=(ind_data/ind_data.shift(1))-1
ind_returns.tail()
```

[78]:

Date	^GSPC	^IXIC	^GDAXI
2020-06-15	0.008312	0.014309	-0.003174
2020-06-16	0.018962	0.017464	0.033943
2020-06-17	-0.003600	0.001481	0.005398
2020-06-18	0.000594	0.003281	-0.008125
2020-06-19	-0.005649	0.000309	0.004008

[79]:
```
annual_ind_returns=ind_returns.mean()*250
```

[80]:
```
annual_ind_returns
```

[80]:
```
^GSPC     0.080320
^IXIC     0.118697
^GDAXI    0.075806
dtype: float64
```

[81]:
```
tickers=['PG','^GSPC','^DJI']
data_2=pd.DataFrame()
for t in tickers:
    data_2[t]=web.DataReader(t,data_source='yahoo', start='2007-1-1') ['Adj␣
 ↪Close']
```

[82]:
```
data_2.tail()
```

[82]:
```
                    PG        ^GSPC           ^DJI
Date
2020-06-15  116.690002  3066.590088  25763.160156
2020-06-16  118.129997  3124.739990  26289.980469
2020-06-17  117.930000  3113.489990  26119.609375
2020-06-18  119.279999  3115.340088  26080.099609
2020-06-19  118.919998  3097.739990  25871.460938
```

[83]:
```
(data_2/data_2.iloc[0]*100).plot(figsize=(15,6));
```

[84]:
```
data_2.tail()
```

[84]:
```
                    PG        ^GSPC           ^DJI
Date
2020-06-15  116.690002  3066.590088  25763.160156
```

```
2020-06-16  118.129997  3124.739990  26289.980469
2020-06-17  117.930000  3113.489990  26119.609375
2020-06-18  119.279999  3115.340088  26080.099609
2020-06-19  118.919998  3097.739990  25871.460938
```

[20]:
```
tickers=['PG', 'BEI.DE']
prova = pd.DataFrame()
for t in tickers:
    prova[t] = web.DataReader(t, data_source='yahoo', start='2007-1-1')['Adj␣
 ↪Close']
```

[21]:
```
prova.tail()
```

[21]:
```
                    PG      BEI.DE
Date
2020-06-15  116.690002   96.379997
2020-06-16  118.129997   99.900002
2020-06-17  117.930000  100.250000
2020-06-18  119.279999   99.760002
2020-06-19  118.919998  101.250000
```

[22]:
```
prova_returns=np.log(prova/prova.shift(1))
```

[23]:
```
prova_returns
```

[23]:
```
                  PG    BEI.DE
Date
2007-01-03       NaN       NaN
2007-01-04 -0.007621  0.006544
2007-01-05 -0.008624 -0.020772
2007-01-08  0.002202  0.000202
2007-01-09 -0.002517 -0.022858
…                …         …
2020-06-15  0.009212 -0.019115
2020-06-16  0.012265  0.035871
2020-06-17 -0.001694  0.003497
2020-06-18  0.011382 -0.004900
2020-06-19 -0.003023  0.014825

[3390 rows x 2 columns]
```

[105]:
```
prova_returns['PG'].mean()
```

[105]: 0.00029915224773705256

[106]:
```
prova_returns['PG'].mean()*250
```

[106]: 0.07478806193426314

```
[107]: prova_returns['PG'].std()
```

[107]: 0.011965592497531945

```
[108]: prova_returns['PG'].std()*250**0.5
```

[108]: 0.18919262922811816

```
[113]: prova_returns['BEI.DE'].mean()
```

[113]: 0.000260295490819062

```
[111]: prova_returns['BEI.DE'].mean()*250
```

[111]: 0.0650738727047655

```
[112]: prova_returns['BEI.DE'].std()
```

[112]: 0.013801508704794914

```
[114]: prova_returns['BEI.DE'].std()*250**0.5
```

[114]: 0.2182210132689619

```
[25]: print (prova_returns['PG'].mean()*250)
      print (prova_returns['BEI.DE'].mean()*250)
```

```
0.07479513851801017
0.0650738727047655
```

```
[26]: prova_returns[['PG', 'BEI.DE']].mean()*250
```

```
[26]: PG        0.074795
      BEI.DE    0.065074
      dtype: float64
```

```
[120]: prova_returns[['PG', 'BEI.DE']].std()*250**0.5
```

```
[120]: PG        0.189193
       BEI.DE    0.218221
       dtype: float64
```

```
[123]: PG_Var = prova_returns ['PG'].var()
       PG_Var
```

[123]: 0.00014317540381699278

[124]:
```
BEI_Var = prova_returns ['BEI.DE'].var()
```

[125]:
```
BEI_Var
```

[125]: 0.00019048164252852978

[126]:
```
PG_Var_a = prova_returns ['PG'].var()*250
```

[127]:
```
PG_Var_a
```

[127]: 0.035793850954248196

[32]:
```
BEI_Var_a = prova_returns ['BEI.DE'].var()*250
```

[33]:
```
BEI_Var_a
```

[33]: 0.047620410632132446

[131]:
```
cov_matrix=prova_returns.cov()
cov_matrix
```

[131]:

	PG	BEI.DE
PG	0.000143	0.000045
BEI.DE	0.000045	0.000190

[132]:
```
cov_matrix_a=prova_returns.cov()*250
```

[133]:
```
cov_matrix_a
```

[133]:

	PG	BEI.DE
PG	0.035794	0.011371
BEI.DE	0.011371	0.047620

[134]:
```
corr_matrix=prova_returns.corr()
```

[135]:
```
corr_matrix
```

[135]:

	PG	BEI.DE
PG	1.000000	0.275006
BEI.DE	0.275006	1.000000

[136]:
```
corr_matrix_a=prova_returns.corr()*250
corr_matrix_a
```

[136]:

	PG	BEI.DE
PG	250.000000	68.751612
BEI.DE	68.751612	250.000000

[137]:
```
#Calculating Portfolio Risk
```

[139]:
```
#Equal Weighting scheme:
weights=np.array([0.5,0.5])
```

[40]:
```
#Portfolio Variance
pfolio_var=np.dot(weights.T, np.dot(prova_returns.cov()*250,weights))
pfolio_var
```

[40]: 0.026539015888108423

[141]:
```
#Portfolio Volatility
pfolio_vol=np.dot(weights.T, np.dot(prova_returns.cov()*250,weights))**0.5
pfolio_vol
```

[141]: 0.16290808293358686

[142]:
```
print (str(round(pfolio_vol, 5)*100)+ '%')
```

16.291%

[1]:
```
#Calculating Diversifable and Non-Diversifable Risk of a Portfolio
```

[27]:
```
weights= np.array([0.5,0.5])
```

[3]:
```
import pandas as pd
```

[8]:
```
import pandas_datareader as web
```

[28]:
```
weights= np.array([0.5,0.5])
```

[10]:
```
import numpy as np
```

[11]:
```
import matplotlib.pyplot as plt
```

[12]:
```
weights= np.array([0.5,0.5])
```

[13]:
```
weights[0]
```

[13]: 0.5

[14]:
```
weights[1]
```

[14]: 0.5

[30]:
```
PG_var_a=prova_returns[['PG']].var()*250
```

[31]:
```
PG_var_a
```

[31]:
```
PG    0.035794
dtype: float64
```

[35]:
```
BEI_var_a=prova_returns[['BEI.DE']].var()*250
```

[36]:
```
BEI_var_a
```

[36]:
```
BEI.DE    0.04762
dtype: float64
```

[37]:
```
float(PG_var_a)
```

[37]: 0.035793790587168625

[44]:
```
PG_var_a=prova_returns['PG'].var()*250
PG_var_a
```

[44]: 0.035793790587168625

[46]:
```
BEI_var_a=prova_returns['BEI.DE'].var()*250
BEI_var_a
```

[46]: 0.047620410632132446

[48]:
```
dr=pfolio_var-(weights[0]**2*PG_var_a)-(weights[1]**2*BEI_var_a)
```

[49]:
```
dr
```

[49]: 0.005685465583283155

[56]:
```
print (str(round(dr*100,3)) + '%')
```

```
0.569%
```

[57]:
```
n_dr_1=pfolio_var-dr
```

[58]:
```
n_dr_1
```

[58]: 0.020853550304825268

[59]:
```
import numpy as np
```

[60]:
```python
import pandas as pd
```

[61]:
```python
from scipy import stats
```

[62]:
```python
import statsmodels.api as sm
```

[63]:
```python
import matplotlib.pyplot as plt
```

[67]:
```python
data=pd.read_excel ("/Users/alessandrosollazzo/Desktop/original.xlsx")
```

[68]:
```python
data
```

[68]:
```
    House Price  House Size (sq.ft.) State  Number of Rooms  \
0       1116000                 1940    IN                8
1        860000                 1300    IN                5
2        818400                 1420    IN                6
3       1000000                 1680    IN                7
4        640000                 1270    IN                5
5       1010000                 1850    IN                7
6        600000                 1000    IN                4
7        700000                 1100   LA                 4
8       1100000                 1600   LA                 7
9        570000                 1000    NY                5
10       860000                 2150    NY                9
11      1085000                 1900    NY                9
12      1250000                 2200    NY                9
13       850000                 1100    TX                4
14       640000                  860    TX                4
15       900000                 1325    TX                6
16       730000                 1350    TX                6
17       750000                 1600    TX                6
18       650000                  950    TX                2
19       680000                 1250    TX                4

    Year of Construction
0                   2002
1                   1992
2                   1987
3                   2000
4                   1995
5                   1998
6                   2015
7                   2014
8                   2017
9                   1997
10                  1997
11                  2000
```

```
12                    2014
13                    2017
14                    1997
15                    1997
16                    2000
17                    1992
18                    1987
19                    2000
```

[70]:
```
data[['House Price','House Size (sq.ft.)']]
```

[70]:

	House Price	House Size (sq.ft.)
0	1116000	1940
1	860000	1300
2	818400	1420
3	1000000	1680
4	640000	1270
5	1010000	1850
6	600000	1000
7	700000	1100
8	1100000	1600
9	570000	1000
10	860000	2150
11	1085000	1900
12	1250000	2200
13	850000	1100
14	640000	860
15	900000	1325
16	730000	1350
17	750000	1600
18	650000	950
19	680000	1250

[71]:
```
X=data ['House Size (sq.ft.)']
Y=data['House Price']
```

[72]:
```
X
```

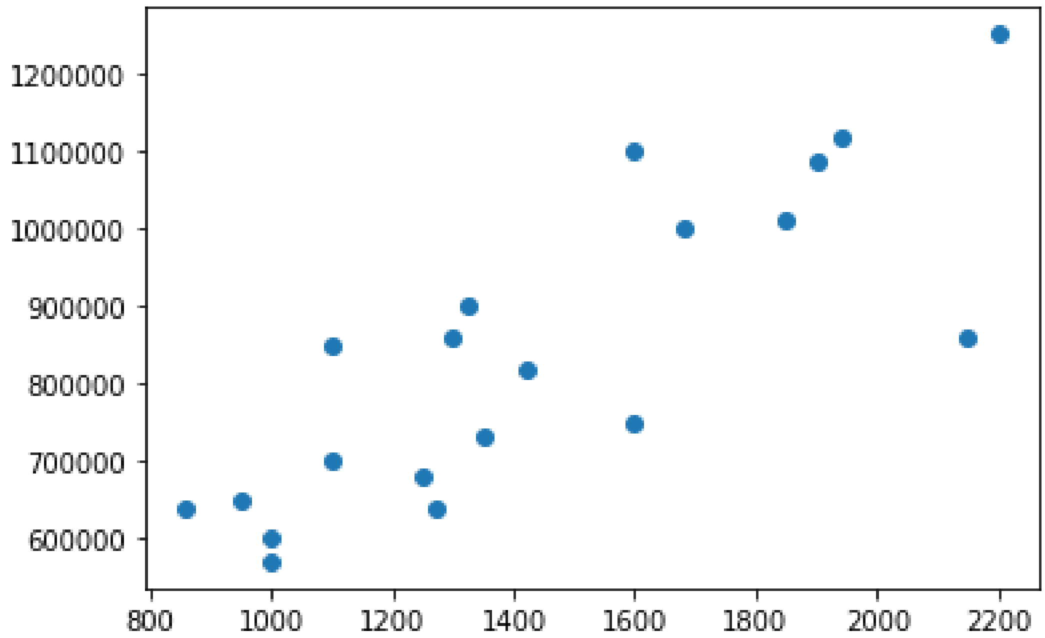
[72]:
```
0      1940
1      1300
2      1420
3      1680
4      1270
5      1850
6      1000
7      1100
8      1600
```

```
9      1000
10     2150
11     1900
12     2200
13     1100
14      860
15     1325
16     1350
17     1600
18      950
19     1250
Name: House Size (sq.ft.), dtype: int64
```

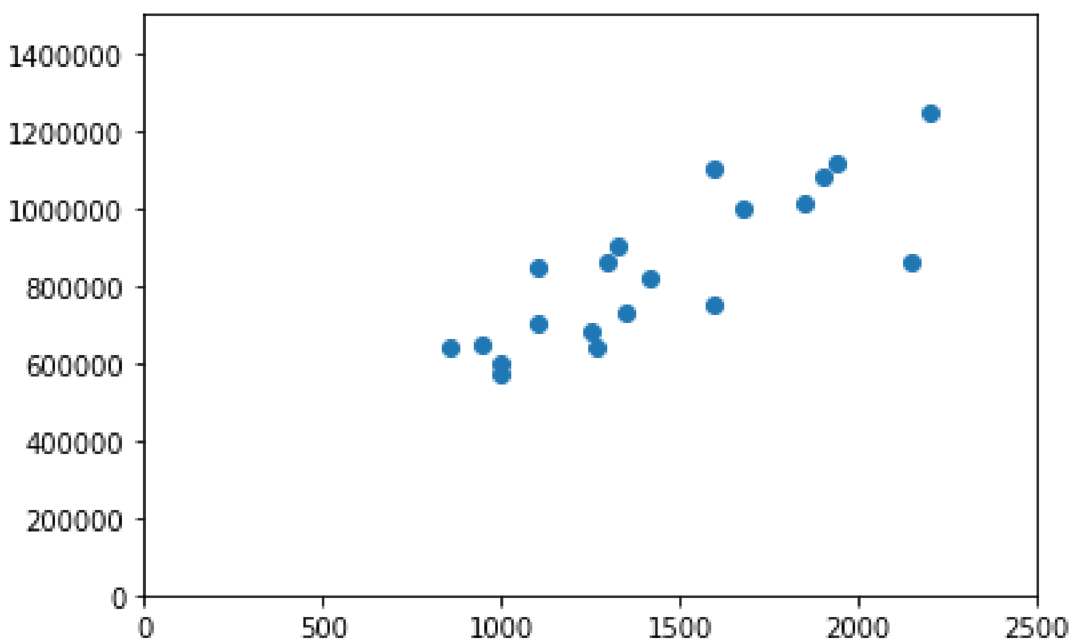
[73]:
```
Y
```

[73]:
```
0      1116000
1       860000
2       818400
3      1000000
4       640000
5      1010000
6       600000
7       700000
8      1100000
9       570000
10      860000
11     1085000
12     1250000
13      850000
14      640000
15      900000
16      730000
17      750000
18      650000
19      680000
Name: House Price, dtype: int64
```

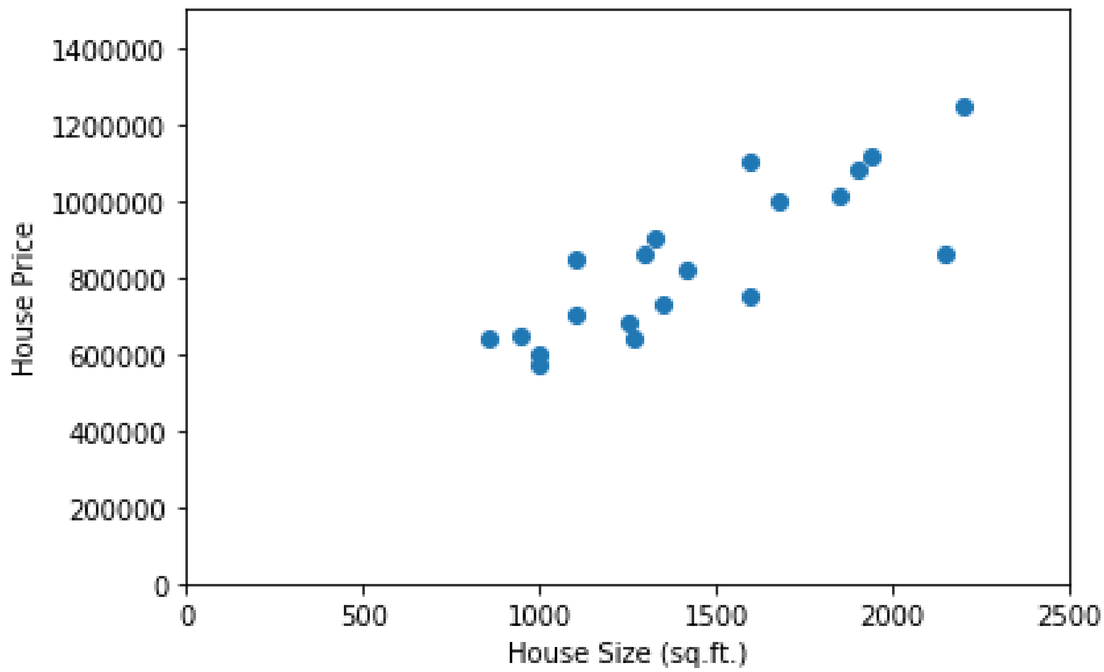
[74]:
```
plt.scatter(X,Y)
plt.show()
```



[75]:
```
plt.scatter(X,Y)
plt.axis([0, 2500, 0, 1500000])
plt.show()
```



[76]:
```
plt.scatter(X,Y)
plt.axis([0, 2500, 0, 1500000])
plt.ylabel('House Price')
plt.xlabel('House Size (sq.ft.)')
plt.show()
```



[77]:
```
X1=sm.add_constant(X)
```

[78]:
```
reg=sm.OLS(Y, X1).fit()
```

[79]:
```
reg.summary()
```

[79]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            House Price   R-squared:                       0.678
Model:                            OLS   Adj. R-squared:                  0.660
Method:                 Least Squares   F-statistic:                     37.95
Date:                Mon, 22 Jun 2020   Prob (F-statistic):           8.13e-06
Time:                        18:08:31   Log-Likelihood:                -260.43
No. Observations:                  20   AIC:                             524.9
Df Residuals:                      18   BIC:                             526.8
Df Model:                           1
Covariance Type:            nonrobust
```

```
==============================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
const               2.608e+05   9.76e+04      2.673      0.016    5.58e+04
4.66e+05
House Size (sq.ft.)  401.9163     65.243      6.160      0.000     264.846
538.987
==============================================================================
Omnibus:                        1.238   Durbin-Watson:                   1.810
Prob(Omnibus):                  0.538   Jarque-Bera (JB):                0.715
Skew:                          -0.459   Prob(JB):                        0.699
Kurtosis:                       2.884   Cond. No.                     5.66e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.66e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[80]:
```
#Expected Value of Y:
```

260800 + 402*1000

[81]:
```
260800 + 402*1000
```

[81]: 662800

[83]:
```
slope, intercept, r_value, p_value, std_error = stats.linregress(X,Y)
```

[84]:
```
slope
```

[84]: 401.91628631922595

[85]:
```
intercept
```

[85]: 260806.2360560964

[86]:
```
r_value
```

[86]: 0.8235775534696924

[87]:
```
r_value**2
```

[87]: 0.678279986579124

[88]:
```
p_value
```

[88]: 8.129642377231308e-06

[89]:
```
std_error
```

[89]: 65.24299510636492

[90]:
```
#Efficient Frontier
```

[91]:
```
import numpy as np
import pandas as pd
from pandas_datareader import data as web
import matplotlib.pyplot as plt
%matplotlib inline
```

[92]:
```
assets=['PG', '^GSPC']
pf_data=pd.DataFrame()

for a in assets:
    pf_data[a]=web.DataReader(a,data_source='yahoo', start = '2010-1-1')['Adj␣
↪Close']
```

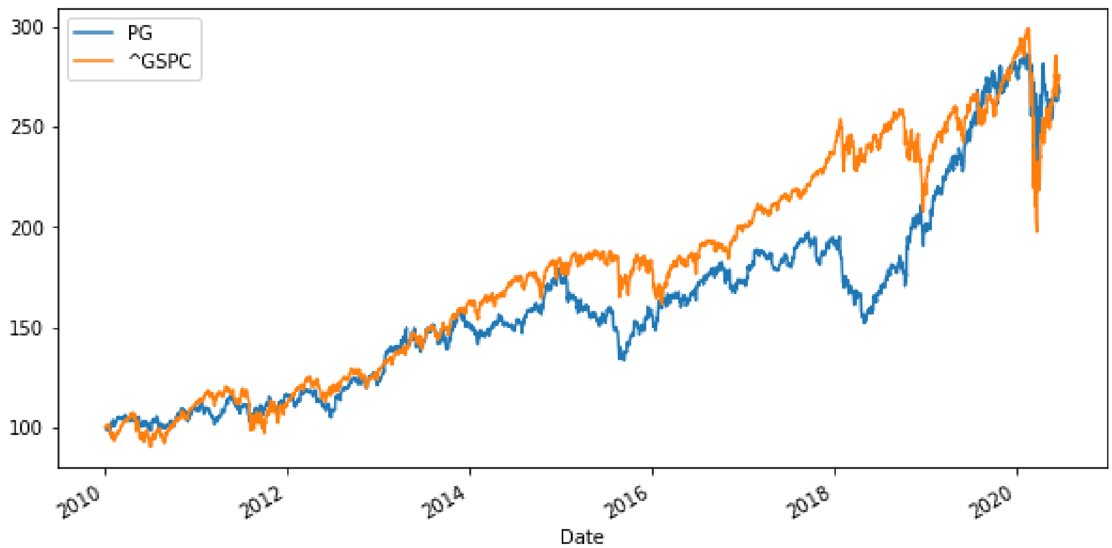
[94]:
```
pf_data.tail()
```

[94]:

Date	PG	^GSPC
2020-06-16	118.129997	3124.739990
2020-06-17	117.930000	3113.489990
2020-06-18	119.279999	3115.340088
2020-06-19	118.919998	3097.739990
2020-06-22	117.750000	3117.860107

[96]:
```
(pf_data/pf_data.iloc[0]*100).plot(figsize=(10, 5))
```

[96]: <matplotlib.axes._subplots.AxesSubplot at 0x7febc77e3210>



[ ]:

[97]:
```
log_returns=np.log(pf_data/pf_data.shift(1))
```

[98]:
```
log_returns.mean()*250
```

[98]:
```
PG       0.093312
^GSPC    0.096079
dtype: float64
```

[99]:
```
log_returns.cov()*250
```

[99]:

	PG	^GSPC
PG	0.029550	0.017978
^GSPC	0.017978	0.030639

[101]:
```
log_returns.corr()
```

[101]:

	PG	^GSPC
PG	1.000000	0.597484
^GSPC	0.597484	1.000000

[102]:
```
num_assets=len(assets)
```

[103]:
```
num_assets
```

[103]: 2

[104]:
```
arr=np.random.random(2)
arr
```

[104]: array([0.92681938, 0.56287802])

[105]:
```
weights=np.random.random(num_assets)
```

[106]:
```
weights/=np.sum(weights)
```

[107]:
```
weights
```

[107]: array([0.68965479, 0.31034521])

[108]:
```
arr[0]+arr[1]
```

[108]: 1.489697399045426

[109]:
```
weights[0]+weights[1]
```

[109]: 1.0

[110]:
```
#Expected Portfolio Return
```

[112]:
```
np.sum(weights*log_returns.mean())*250
```

[112]: 0.09417065781603497

[113]:
```
np.dot(weights.T,np.dot(log_returns.cov()*250,weights))
```

[113]: 0.024701248606204557

[114]:
```
np.sqrt(np.dot(weights.T,np.dot(log_returns.cov()*250,weights)))
```

[114]: 0.1571663087503316

[116]:
```
pfolio_returns=[]
pfolio_volatilities=[]
for x in range (1000):
    weights=np.random.random(num_assets)
    weights/=np.sum(weights)
    pfolio_returns.append(np.sum(weights*log_returns.mean())*250)
    pfolio_volatilities.append(np.sqrt(np.dot(weights.T,np.dot(log_returns.
 ↪cov()*250,weights))))

pfolio_returns=np.array(pfolio_returns)
pfolio_volatiltites=np.array(pfolio_volatilities)
```

```
pfolio_returns, pfolio_volatilities
```

[116]:
```
(array([0.09560059, 0.09457073, 0.09527279, 0.09421889, 0.09346499,
        0.09453736, 0.09482576, 0.09470435, 0.09379315, 0.09586867,
        0.09366731, 0.09361216, 0.09427773, 0.0956426 , 0.09450247,
        0.09509906, 0.09535168, 0.09509524, 0.09464568, 0.09426597,
        0.09483238, 0.09458422, 0.09340929, 0.09470454, 0.09466953,
        0.09603886, 0.09532372, 0.09352781, 0.095093  , 0.09542325,
        0.09599244, 0.09538526, 0.09394972, 0.09516095, 0.09597559,
        0.09361086, 0.09409965, 0.09588542, 0.09372767, 0.09395053,
        0.09538051, 0.09450683, 0.09343882, 0.09562786, 0.09535451,
        0.09594105, 0.09423206, 0.095123  , 0.0952852 , 0.09563569,
        0.09583193, 0.09349934, 0.09410125, 0.09443747, 0.09479604,
        0.0953479 , 0.09447449, 0.09542762, 0.09386007, 0.09490549,
        0.09488146, 0.09555966, 0.09379803, 0.09392452, 0.09486269,
        0.09497173, 0.09386709, 0.09335695, 0.09464032, 0.09371611,
        0.09576502, 0.09333946, 0.09415869, 0.09560028, 0.09381995,
        0.09569261, 0.09334741, 0.09419162, 0.09476275, 0.09449122,
        0.09467136, 0.09542289, 0.09458963, 0.09511118, 0.09497702,
        0.09367   , 0.09607735, 0.09455349, 0.09384961, 0.0941367 ,
        0.09458645, 0.09480702, 0.09511152, 0.09381743, 0.09381165,
        0.09406188, 0.09412077, 0.09516593, 0.09435748, 0.0943692 ,
        0.09459019, 0.09405192, 0.09354372, 0.09471155, 0.09506413,
        0.09468382, 0.09482836, 0.09470878, 0.09468423, 0.09334407,
        0.09462403, 0.09429817, 0.09444843, 0.09441892, 0.09444931,
        0.09373251, 0.09492435, 0.09382679, 0.09453852, 0.09347314,
        0.09442133, 0.09514623, 0.09580872, 0.09537259, 0.09362725,
        0.09332467, 0.09575327, 0.0939635 , 0.0933774 , 0.09487113,
        0.09420953, 0.09473842, 0.09522466, 0.09360245, 0.09409441,
        0.0945232 , 0.09426036, 0.09562779, 0.09484999, 0.0954985 ,
        0.09600465, 0.09580403, 0.09432985, 0.09426775, 0.09578731,
        0.09402261, 0.09354385, 0.09390005, 0.09416852, 0.09331696,
        0.09494071, 0.09336879, 0.09434579, 0.09494284, 0.09454779,
        0.09473965, 0.09528114, 0.09358425, 0.09545957, 0.09580045,
        0.09370368, 0.0942451 , 0.09348772, 0.09571325, 0.09483538,
        0.09465165, 0.09403779, 0.09371174, 0.09494763, 0.0946557 ,
        0.09473866, 0.09454763, 0.09363299, 0.09440666, 0.09554985,
        0.09372459, 0.09491579, 0.09479372, 0.09490603, 0.09448617,
        0.09426497, 0.09506834, 0.09597165, 0.09466831, 0.0948345 ,
        0.095598  , 0.09490794, 0.09467885, 0.09482733, 0.09430778,
        0.09388452, 0.0941833 , 0.09480319, 0.09482704, 0.09483166,
        0.09509057, 0.09364491, 0.09476545, 0.09548953, 0.09415353,
        0.09550527, 0.09530421, 0.09458387, 0.09354876, 0.09451161,
        0.09486543, 0.09453512, 0.09377775, 0.09437189, 0.09352002,
        0.0959562 , 0.09438938, 0.09470043, 0.09518486, 0.09553177,
        0.09359681, 0.09500778, 0.09506428, 0.09600106, 0.09566629,
        0.09464552, 0.09485042, 0.09460678, 0.09438166, 0.09481292,
```

```
0.09423803, 0.09356546, 0.09437698, 0.09435717, 0.09437869,
0.09481303, 0.09449171, 0.09430972, 0.09460732, 0.09553191,
0.09546167, 0.09473337, 0.09432789, 0.09467944, 0.09462324,
0.09526756, 0.09483637, 0.09558382, 0.09489111, 0.09398033,
0.09578202, 0.0948634 , 0.09459256, 0.09445875, 0.09514919,
0.09496428, 0.0958413 , 0.09439808, 0.09584979, 0.09486534,
0.09469276, 0.09569968, 0.09444466, 0.09380509, 0.09475962,
0.09506831, 0.09553872, 0.09564574, 0.0942259 , 0.09433833,
0.09461849, 0.09467437, 0.09570759, 0.09411808, 0.09506523,
0.09404612, 0.09471624, 0.09445415, 0.09471914, 0.09452893,
0.0951889 , 0.09419634, 0.09539235, 0.09498545, 0.09389263,
0.09482877, 0.09481596, 0.09498316, 0.09472813, 0.09435785,
0.09453682, 0.09586364, 0.09423964, 0.09498459, 0.09474926,
0.09453595, 0.09430449, 0.09413736, 0.09424614, 0.09366242,
0.09467145, 0.09434941, 0.09515218, 0.09482906, 0.09459326,
0.09449076, 0.09452294, 0.09476774, 0.0935524 , 0.0937    ,
0.09473355, 0.09485895, 0.0950404 , 0.09507539, 0.09420116,
0.09438484, 0.09419149, 0.09505918, 0.09495495, 0.09473871,
0.09482658, 0.09430784, 0.09382664, 0.09482768, 0.09424511,
0.09399444, 0.09544124, 0.09486196, 0.09522079, 0.09435505,
0.09487176, 0.09462412, 0.09602129, 0.09386954, 0.09423516,
0.09479473, 0.09515119, 0.09585169, 0.09431735, 0.09510718,
0.09499307, 0.09333132, 0.09453344, 0.09479874, 0.0936654 ,
0.09445205, 0.09470869, 0.0953035 , 0.09459969, 0.09565726,
0.09373864, 0.09470249, 0.09466665, 0.09438727, 0.0960217 ,
0.09331732, 0.09535988, 0.09435045, 0.09468354, 0.0954837 ,
0.09407763, 0.09333941, 0.09546441, 0.09417555, 0.09569718,
0.09559657, 0.09533504, 0.09481044, 0.09573911, 0.09474666,
0.09363652, 0.0933125 , 0.09530799, 0.09516655, 0.09520845,
0.09521502, 0.09528683, 0.09530614, 0.09482551, 0.09443718,
0.09511975, 0.09340659, 0.09407747, 0.09520038, 0.09469809,
0.09374609, 0.09494866, 0.09593475, 0.09377363, 0.09432521,
0.0940583 , 0.0947183 , 0.09498863, 0.09465969, 0.09473607,
0.09422115, 0.09478909, 0.09424622, 0.09423625, 0.09463608,
0.09580799, 0.0947622 , 0.0942641 , 0.09556666, 0.09517263,
0.09524058, 0.09463161, 0.09603559, 0.09528452, 0.09515566,
0.09439066, 0.09360339, 0.09476386, 0.09500229, 0.09345873,
0.09447614, 0.09529209, 0.09464621, 0.09381119, 0.09585759,
0.09541107, 0.09432023, 0.09425936, 0.09531075, 0.09399517,
0.09465208, 0.09434227, 0.09495394, 0.09364722, 0.09373139,
0.09354985, 0.09448497, 0.09556413, 0.09410239, 0.09423214,
0.0949336 , 0.09515413, 0.09463943, 0.09562285, 0.09516252,
0.0953864 , 0.09426   , 0.09437173, 0.09394125, 0.09372597,
0.09475215, 0.09485794, 0.09460287, 0.09439865, 0.09471305,
0.09443129, 0.09516738, 0.09439482, 0.0953522 , 0.09492302,
0.09379545, 0.0934463 , 0.09392678, 0.09333453, 0.0941816 ,
0.09460913, 0.09482039, 0.09427035, 0.09478563, 0.09435937,
```

0.09519456, 0.09412628, 0.09469435, 0.09513739, 0.09566686,
0.09601466, 0.09498739, 0.09459551, 0.09355367, 0.09445998,
0.0959211 , 0.09464605, 0.0941066 , 0.09545207, 0.09487247,
0.09437601, 0.09501295, 0.0957354 , 0.09507545, 0.0937278 ,
0.09384459, 0.09473935, 0.09570509, 0.09468438, 0.09345579,
0.09425882, 0.09358394, 0.09474871, 0.09510426, 0.09421865,
0.09465455, 0.09484179, 0.09494356, 0.09438001, 0.09526805,
0.09380604, 0.09501269, 0.09350883, 0.09533059, 0.0952661 ,
0.09446129, 0.09501284, 0.09521386, 0.09439604, 0.09562473,
0.09562507, 0.09399439, 0.09436619, 0.09573542, 0.09515198,
0.09379589, 0.09498064, 0.09495532, 0.09489187, 0.09528767,
0.09449975, 0.09476059, 0.09521947, 0.09549943, 0.09388002,
0.09513978, 0.09544758, 0.09488134, 0.09362149, 0.09476867,
0.09437517, 0.09516868, 0.09594557, 0.09469394, 0.09520346,
0.09440008, 0.09426617, 0.09585815, 0.09388188, 0.09446452,
0.09491664, 0.09424026, 0.09399169, 0.0940504 , 0.09463321,
0.0946626 , 0.0947724 , 0.09491585, 0.09410968, 0.09380817,
0.09341942, 0.09379388, 0.09487531, 0.09401379, 0.09604979,
0.09392684, 0.0952348 , 0.09506421, 0.09357525, 0.0952966 ,
0.09605615, 0.09451884, 0.09434897, 0.0949999 , 0.09540309,
0.09436849, 0.09485076, 0.09538262, 0.0958124 , 0.09361938,
0.0960782 , 0.09374772, 0.09471792, 0.09457256, 0.0949575 ,
0.09472836, 0.09463383, 0.09434139, 0.09399014, 0.09510898,
0.09346879, 0.09471967, 0.09475783, 0.09412186, 0.0938392 ,
0.09381303, 0.09598825, 0.0949346 , 0.09421245, 0.09452269,
0.0948356 , 0.09436601, 0.09478597, 0.09356895, 0.09583907,
0.09600038, 0.09466397, 0.09410782, 0.09454719, 0.09353789,
0.09440844, 0.0943208 , 0.09449303, 0.09546403, 0.09494764,
0.09548039, 0.09352439, 0.09381578, 0.09448078, 0.09344815,
0.09573114, 0.09451026, 0.09345234, 0.09405758, 0.09383415,
0.0942371 , 0.09441904, 0.09591245, 0.0949795 , 0.09440233,
0.0949898 , 0.09438336, 0.09465239, 0.09417972, 0.09583087,
0.09533474, 0.09383401, 0.09534959, 0.09507822, 0.09436274,
0.09547796, 0.09564209, 0.09524944, 0.0940619 , 0.09399441,
0.09407223, 0.09450007, 0.09333635, 0.095737 , 0.09404813,
0.09514148, 0.09480854, 0.09399373, 0.09494602, 0.09476307,
0.09523505, 0.09333098, 0.09358559, 0.09500519, 0.09385282,
0.09563556, 0.09545115, 0.09449327, 0.09526058, 0.09478349,
0.09454264, 0.09410757, 0.09480106, 0.09591438, 0.0950431 ,
0.09449608, 0.09451682, 0.09553864, 0.09493072, 0.09406185,
0.09586683, 0.09489576, 0.09453908, 0.09445042, 0.09535569,
0.09474332, 0.0935265 , 0.09347661, 0.09447435, 0.09438433,
0.09366904, 0.0959734 , 0.09441047, 0.09457535, 0.09515297,
0.09500921, 0.09469894, 0.09374268, 0.09430165, 0.09482994,
0.09422162, 0.09472219, 0.09458499, 0.09569888, 0.09477011,
0.09373463, 0.09601676, 0.09529177, 0.09493365, 0.09430922,
0.09369802, 0.09581612, 0.09413252, 0.09579855, 0.09480679,

0.09394086, 0.09336612, 0.09590753, 0.09347638, 0.09362007,
0.09351736, 0.09507882, 0.0947719 , 0.09508723, 0.095328 ,
0.09462764, 0.09333282, 0.09528955, 0.09479908, 0.09389553,
0.09404123, 0.09561763, 0.09514346, 0.09500447, 0.09345121,
0.09565312, 0.0941591 , 0.09431385, 0.09411787, 0.09496858,
0.09482359, 0.09470708, 0.09456809, 0.09466427, 0.09594851,
0.09566407, 0.09517247, 0.09480758, 0.0944905 , 0.09419386,
0.09470322, 0.0955674 , 0.0954521 , 0.0949665 , 0.09423822,
0.09460751, 0.09349976, 0.09525947, 0.09570911, 0.09494468,
0.09475143, 0.09504092, 0.0949844 , 0.09437444, 0.09543536,
0.09530704, 0.09450431, 0.09371684, 0.09478611, 0.09536702,
0.0946314 , 0.09589863, 0.09469075, 0.09573308, 0.09573837,
0.09557405, 0.09492036, 0.09602195, 0.09499671, 0.09460384,
0.09429762, 0.09497451, 0.09451319, 0.09359887, 0.09487834,
0.09504317, 0.09490931, 0.09597334, 0.09476458, 0.09541044,
0.09549264, 0.09490947, 0.09505447, 0.09387912, 0.09481786,
0.09509584, 0.09549198, 0.09508994, 0.09464017, 0.09511365,
0.09456563, 0.09579412, 0.09481979, 0.09468728, 0.09443181,
0.09495158, 0.0949131 , 0.09465701, 0.09488248, 0.09494685,
0.0940317 , 0.09451505, 0.09550691, 0.0942015 , 0.09471377,
0.09499095, 0.09496709, 0.09519457, 0.0946198 , 0.09508695,
0.09453319, 0.0949953 , 0.0949234 , 0.09433021, 0.09442132,
0.09523626, 0.09580136, 0.09571215, 0.09592382, 0.09527359,
0.09479259, 0.09495558, 0.09356699, 0.09475648, 0.09380929,
0.09585557, 0.09476512, 0.09419532, 0.09418946, 0.09552314,
0.09457151, 0.09438286, 0.09467776, 0.09513837, 0.0947376 ,
0.09419639, 0.09337013, 0.09526332, 0.09403149, 0.0957648 ,
0.09436453, 0.09464663, 0.09511675, 0.09590664, 0.09571982,
0.09500141, 0.09380591, 0.09379276, 0.09552072, 0.09536463,
0.09427619, 0.09467095, 0.09457514, 0.09501946, 0.09588117,
0.09511132, 0.09450913, 0.0938121 , 0.0937392 , 0.09479373,
0.09415264, 0.0948539 , 0.09415184, 0.09402967, 0.09379008,
0.09374045, 0.0940508 , 0.09364145, 0.09413502, 0.09586984,
0.0935384 , 0.09427934, 0.09450507, 0.09551084, 0.09391505,
0.09460462, 0.09466365, 0.09377122, 0.09492714, 0.09434224,
0.09492608, 0.09452838, 0.09537356, 0.09586908, 0.09456509,
0.09461049, 0.09438334, 0.09539335, 0.09527358, 0.09558602,
0.09424828, 0.09466127, 0.09479731, 0.09419141, 0.09475719,
0.09415398, 0.09452345, 0.09436317, 0.09347029, 0.0935408 ,
0.09481909, 0.09511829, 0.0958972 , 0.09492478, 0.09564188,
0.09334177, 0.0945175 , 0.09502587, 0.09485348, 0.09528745,
0.09417197, 0.09478041, 0.09482536, 0.09585636, 0.09412377,
0.09420843, 0.09453045, 0.09517943, 0.09495789, 0.09468712,
0.09600117, 0.09397961, 0.09444824, 0.0934761 , 0.0953808 ,
0.09460973, 0.09473812, 0.09352948, 0.09460341, 0.0944554 ,
0.09454999, 0.09504174, 0.09462423, 0.09549862, 0.09365078,
0.09411103, 0.09455043, 0.09494492, 0.09468671, 0.09508606,

```
       0.09401879, 0.09483731, 0.09451843, 0.09482918, 0.09501904,
       0.09513565, 0.0948302 , 0.09443516, 0.09411094, 0.09444645,
       0.09351444, 0.09405269, 0.09381317, 0.09490425, 0.09529436,
       0.09433455, 0.09466869, 0.09428987, 0.09528048, 0.09418478,
       0.09448715, 0.09463161, 0.09527947, 0.0947574 , 0.094654  ,
       0.09526113, 0.09514447, 0.09407214, 0.09502742, 0.09369491,
       0.09579551, 0.09399358, 0.09458566, 0.09473642, 0.09439584,
       0.09492432, 0.09449935, 0.09503141, 0.09525867, 0.09418878,
       0.09583859, 0.09439041, 0.09529536, 0.09581221, 0.09441708,
       0.09446682, 0.09496109, 0.09331655, 0.09403305, 0.09378814,
       0.09469069, 0.09374782, 0.09470741, 0.09539546, 0.09501738,
       0.09537538, 0.09566995, 0.09560301, 0.09394665, 0.09442977,
       0.09552369, 0.09498115, 0.09465419, 0.09507854, 0.09586955,
       0.09487834, 0.09547552, 0.09543274, 0.09493046, 0.09405685]),
 [0.16427362467208367,
  0.15503626610152327,
  0.15911785633353312,
  0.15673981089239664,
  0.16835696938828537,
  0.15509023128234403,
  0.1553742848848959,
  0.15504794843888484,
  0.16204293608073683,
  0.16986741891653565,
  0.16424777029362697,
  0.16530032022649632,
  0.15628171227736845,
  0.1650716513269347,
  0.15517090853033014,
  0.15719623120871007,
  0.16017997932808264,
  0.1571605322080828,
  0.15499792917499614,
  0.15636772376132607,
  0.1554006830721704,
  0.15502089980442404,
  0.1696049857754535,
  0.15504823514440735,
  0.1550097842470158,
  0.17400488543559073,
  0.15979025315432932,
  0.16700886677933052,
  0.15713980285645077,
  0.16124311188111262,
  0.1728337758165417,
  0.16066701110858467,
  0.15969596283185536,
```

0.15781409684606315,
0.1724163833839109,
0.1653257495894226,
0.15787709590537918,
0.1702552268470058,
0.16315556396844308,
0.15968496716728756,
0.1605969523237232,
0.1551594743097868,
0.1689374184460018,
0.1647881912252028,
0.16022021905590927,
0.17157405827562572,
0.15663130055069346,
0.15742645138520617,
0.15927730156926498,
0.1649383635678883,
0.16903234298621372,
0.16761188370301258,
0.15785991734031893,
0.15538727954125953,
0.15526670582386842,
0.16012634116006771,
0.15525354160745108,
0.16131099874259436,
0.16098494617012843,
0.15575124098101584,
0.15562411026985926,
0.16352504893272898,
0.1619630247159524,
0.16004326349196235,
0.1555329440250428,
0.15616181540145294,
0.16087872636368292,
0.17082184085114338,
0.1549968625957731,
0.16335980582194634,
0.16756587821046018,
0.17123792423006481,
0.15727918273870156,
0.16426771472002566,
0.16160963999437927,
0.16606058781745534,
0.17104821494390657,
0.15697535183167427,
0.1551674787976782,
0.15520222597864072,

0.15501117816044993,
0.1612374872005085,
0.1550157877406045,
0.15731138111666948,
0.15619833640134476,
0.16419778932000884,
0.1749997826582775,
0.15506131051604224,
0.16114508240451317,
0.15749391908625626,
0.15501872255802696,
0.15530435613617974,
0.15731469096659834,
0.16164976715644758,
0.16174236448461993,
0.15829482191894273,
0.15765543385286898,
0.15786711032852963,
0.15577101601837517,
0.15570671289900154,
0.15501529054518076,
0.15840962558704405,
0.16667770252392225,
0.1550589260391284,
0.15688058006326758,
0.1550224600847205,
0.15538454265942536,
0.15505458276197934,
0.15502287775575319,
0.17112770419465687,
0.15499722171598215,
0.15613865261382542,
0.15534477409833913,
0.1554647384683726,
0.15534146489146955,
0.16307069896662918,
0.1558591968922831,
0.161501029803449,
0.15508796353608084,
0.16817857186630122,
0.15545430076175987,
0.15766040693766073,
0.16851558317926213,
0.1604808545153521,
0.16500718642169332,
0.17159316510073183,
0.16731585248387562,

```
0.159510993463051,
0.17034138113117178,
0.15557307379973362,
0.1568189390759098,
0.1551092414402307,
0.1585274742065277,
0.1654910092254224,
0.15793333832923537,
0.1551199783373656,
0.15640976861235883,
0.16478683328699906,
0.1554752725179883,
0.1624610657345458,
0.17313881374310267,
0.16841215700443,
0.1559335058447879,
0.15635457405663536,
0.16804645955325773,
0.1587582614154094,
0.16667502368621775,
0.16039177671539626,
0.15718628103009327,
0.17177978106897632,
0.15595857643700253,
0.1705430045334414,
0.15583786270453362,
0.15597193408554813,
0.15507092081036158,
0.15511189255193983,
0.15922485082433335,
0.16585252832363778,
0.16181821015774675,
0.168333399384486,
0.16358207966115207,
0.1565272644048282,
0.16786191672947842,
0.16648089516658032,
0.15541294410491652,
0.154999806878782,
0.15857567897148556,
0.1634375808803694,
0.1560022457143455,
0.1550014963020273,
0.1551097509357179,
0.1550712087140633,
0.16489660886421217,
0.1555197645259183,
```

```
0.16335002811248617,
0.16320980513461764,
0.15580930557265266,
0.15525903467896243,
0.15575427773568662,
0.155217108323867 8,
0.15637519183095325,
0.15691740045816263,
0.17231925306626114,
0.1550089015399789,
0.1554093269698704,
0.16422530103127897,
0.1557648688739469,
0.1550175764168222,
0.15538045651161445,
0.1560743348481965,
0.1606188206599231,
0.1570501313066331,
0.15529095039464438,
0.15537930681127424,
0.155397787493007,
0.15711734296786084,
0.16466901882898996,
0.15517469325978114,
0.16231052231574541,
0.15732873121581734,
0.16257552621659405,
0.15952698509279103,
0.15502125521189983,
0.16657361947653007,
0.1551473748974961,
0.15554579915423797,
0.1550946714112782,
0.16229778987075655,
0.15569235748001953,
0.1671725501911907,
0.17194117801010386,
0.1556025000683853,
0.15504242018496794,
0.15807265220927355,
0.16303171931088933,
0.16560242411358572,
0.15642208799467938,
0.15688187264403344,
0.17304890095535705,
0.1655348577680911,
0.15499788885658125,
```

0.15547715257689224,
0.15500351053729244,
0.15564140522164757,
0.1553255935676673,
0.15658326461209998,
0.16623165360743175,
0.15566557390831876,
0.15577271011835003,
0.1556566794371757,
0.1553260002147752,
0.15520080040989326,
0.1560615544947947,
0.15500321941941314,
0.16303419867990507,
0.16185219628960043,
0.15509866731568378,
0.15594564727330726,
0.15501813147724766,
0.15499737676071657,
0.1590516634761481,
0.1554170511020216,
0.1639633577853552,
0.1556737992452945,
0.15929004059272936,
0.16793172638954,
0.1555362615210397,
0.15501326021992046,
0.15530699964834768,
0.15769098201816198,
0.15611120464114067,
0.16924319497560028,
0.15556010313188592,
0.16943561438841545,
0.15554538163236625,
0.15503250189487675,
0.1662038523125683,
0.15535911242628467,
0.1618483313212537,
0.1551593077681157,
0.15691708721031733,
0.16315330968177666,
0.16513244921994874,
0.15668160818899426,
0.15588201095404058,
0.15499857966912997,
0.15501360977295098,
0.166364906713968,

0.15768313331897005,
0.15689015054140223,
0.1584773282838126,
0.1550666447995571,
0.15532355572799653,
0.15507164967398335,
0.1551074583397189,
0.15811744340511005,
0.15693347768704594,
0.16077251865513187,
0.1562578003510075,
0.16049964827817545,
0.15538617106698874,
0.15533683144654792,
0.1562414875019709,
0.15508823499408758,
0.15576894220743404,
0.15509129265516824,
0.16975183641677583,
0.15657036366816468,
0.15625166094563658,
0.15513370006458818,
0.1550930135861281,
0.1560961218883131,
0.15748740824134685,
0.15651906478289293,
0.16433900790486353,
0.15501124714337597,
0.15581684949439287,
0.15772201094744798,
0.15538732856679807,
0.15501268870347085,
0.15520354906759776,
0.1551205628716572,
0.15518092120600185,
0.16649870511632422,
0.16364837041781807,
0.1550990380360715,
0.1555156169080299,
0.15667985644079063,
0.15697972666560853,
0.1568912695968151,
0.15562523548401952,
0.15697646147163138,
0.15683781638446911,
0.15604949341084587,
0.15510985805579122,

0.15537750628503352,
0.15607391934185685,
0.16150344172735615,
0.15538186998805975,
0.15652717117527049,
0.15910878430179165,
0.16152500793206914,
0.1555295060587299,
0.1584819531914332,
0.15578469770315506,
0.1555761267203978,
0.15499720435487047,
0.17355790682341016,
0.16084185402456486,
0.1566062805622851,
0.15526237106023608,
0.15771170574774654,
0.16947878007607028,
0.1560120532070515,
0.1572730245225707,
0.1563127563412518,
0.17143303192595402,
0.15509806433019566,
0.15527573750339496,
0.16428329959583612,
0.155331265077548,
0.15505444853528164,
0.15951749631717885,
0.15500785866225453,
0.16535726423707675,
0.16296389240296388,
0.1550452905016514,
0.15500773671747312,
0.15561296981337883,
0.1735682433599485,
0.17177090904681963,
0.16029697969164583,
0.15581090387788332,
0.1550221724790695,
0.1622134386834854,
0.15811726690390573,
0.17123895556812171,
0.16189667231447952,
0.15712093217364975,
0.16615307325305578,
0.1641986692207913,
0.1599462725312099,

0.1553165694396075,
0.16701741210217355,
0.15512760825012983,
0.16482897631470017,
0.1718881523540025,
0.15957741441025305,
0.15787371141170428,
0.15833858422581634,
0.1584145780594025,
0.15929833976427768,
0.15955263894820668,
0.15537329648051806,
0.15538843329931964,
0.15739446092388784,
0.1696667549416663,
0.15811906378322033,
0.15824648274401754,
0.1550392667046646,
0.16283484912267207,
0.15600882452051115,
0.17142225623431745,
0.16236669183091101,
0.15596233001976237,
0.15833588053843833,
0.15507018977663245,
0.15628058330357825,
0.15500348716277437,
0.15510424739095208,
0.1567209030890566,
0.15524411856728562,
0.15651843904337573,
0.15659745660532212,
0.1549964343339836,
0.16849948279233048,
0.15516603867902534,
0.15638169185008383,
0.16365096602176915,
0.15793910682468815,
0.1587179830112251,
0.15499638043854397,
0.17392132153029186,
0.1592684112593332,
0.15775839895498622,
0.15559612343828608,
0.16547247880752047,
0.1551704258960342,
0.15638076189121355,

```
0.16849483531300036,
0.15524819954292748,
0.15936691982219153,
0.15499806474114594,
0.1617497861068034,
0.16961341864785057,
0.16105554746012998,
0.15599369355330914,
0.15641732579479165,
0.15961444301930702,
0.15909957464512048,
0.15499996957088163,
0.15585855399398119,
0.15604288184791565,
0.1646251028634949,
0.1630903982123937,
0.16655111555222163,
0.15522071580244448,
0.1636054107538261,
0.1578478399455445,
0.15663062493392405,
0.15591474767986074,
0.15774233801321338,
0.1549967422907329,
0.16469269860311173,
0.15783075373520702,
0.1606838972367057,
0.15641249829133153,
0.15569316229188795,
0.15981133154852478,
0.1631853710356422,
0.1551406339761129,
0.15551096414418536,
0.15500578251381633,
0.155557343792949,
0.15506135631157092,
0.15541231768568795,
0.15788260809330187,
0.1555757891270291,
0.16018729638958074,
0.1558513375875994,
0.16200518744048226,
0.1687704461378733,
0.16001169628038375,
0.1713559448736743,
0.15706549742254375,
0.1550022957868221,
```

0.1553535331519382,
0.15633536968376308,
0.15523323611268147,
0.1557604172438974,
0.15818069742530028,
0.15759904288829793,
0.15503445610595784,
0.15757013599535283,
0.1655460878481689,
0.1733903555908024,
0.15627166518270738,
0.155010900673329,
0.1664726262899564,
0.1553026251223058,
0.1710956440911434,
0.15499802500611967,
0.1578031637038866,
0.16169758936054437,
0.15557958108732844,
0.15567059017568355,
0.15646154983780144,
0.1669397222025236,
0.15698029675473507,
0.16315327452427977,
0.16122257996936948,
0.1551112366760611,
0.16631393484717957,
0.15502304219911375,
0.16855969079923513,
0.15642141145321883,
0.16585874268968062,
0.15513239794650943,
0.15724533916565858,
0.15674177740961853,
0.15500098443733606,
0.1554397405569376,
0.15597644992664714,
0.15564986725388152,
0.15905785410814274,
0.1618328914495627,
0.15645952910144856,
0.16740950394301424,
0.15988458288231583,
0.15903320456859474,
0.15529803179079324,
0.15646066130465433,
0.15840112806730616,

```
0.15556986953293653,
0.16472841458852766,
0.16473493353960994,
0.15910946804560983,
0.15572293764850437,
0.16694024822921766,
0.15771992315678046,
0.16199802144104278,
0.15622374506367392,
0.1560519083481253,
0.15567776395862884,
0.1593092657987034,
0.15517825046916872,
0.15516181858371775,
0.15846652114243445,
0.16247680214490054,
0.16068541232103034,
0.15759442193129272,
0.16162580275733207,
0.15562353549685373,
0.1651186463990721,
0.15518348753447217,
0.15567499444646588,
0.15789647783867317,
0.17168319110383745,
0.15503394661611317,
0.1582814287134193,
0.15555056735433742,
0.15636623833017874,
0.16962627293084107,
0.16065772085927474,
0.15528682395564405,
0.1558142135117229,
0.15656549846212875,
0.15914386967639377,
0.1584272639311489,
0.15499635273294238,
0.15500514200716695,
0.1551939110005787,
0.15580965399904467,
0.15777066924429164,
0.16179848702664745,
0.16937448926655296,
0.16203100612363636,
0.15559345927656432,
0.15886640697535978,
0.1742852826572997,
```

0.16001080542099275,
0.1586482611012416,
0.15688129354393543,
0.16603350252059415,
0.1594260986032536,
0.17444916975603786,
0.15512997166825532,
0.15581939278729687,
0.156363003119326,
0.1609341759950098,
0.1557105410800308,
0.1554786550944035,
0.1606280162235162,
0.16859693114731047,
0.1651596447476362,
0.17502203589192425,
0.16280684665677106,
0.15506953349561944,
0.15503396852179005,
0.1560661627843035,
0.15508867529297574,
0.1549963561099182,
0.1558637913164136,
0.159163679089759,
0.15729028587205893,
0.1682736625409867,
0.15507258916550393,
0.15515474280742614,
0.15764424925844656,
0.16130624301921068,
0.16172022000811764,
0.17272943957500017,
0.15592083652634092,
0.15679407236342566,
0.15512112916848067,
0.155413858587024,
0.1557239544489996,
0.15523430006807795,
0.16616085515414317,
0.16919283584326164,
0.1730318273166764,
0.1550059839366769,
0.15779024061567332,
0.15507197522322747,
0.16679860796697346,
0.15551157118893658,
0.15599008476686888,

```
0.15519702128856608,
0.16189049963975513,
0.1560022989175302,
0.1621587430676681,
0.16708064085998744,
0.16167614236644562,
0.15523357044356775,
0.16872913918826674,
0.16685089083182908,
0.15515074161695253,
0.16863622222754848,
0.1583442250949307,
0.16138521068166573,
0.1565906625092562,
0.15546423719605038,
0.17089005720939113,
0.15621570660853762,
0.15553994624454529,
0.1562890041863478,
0.1556327333167877,
0.15500009049819743,
0.15708269667082353,
0.1690084837144367,
0.15994202983244804,
0.16138730416478156,
0.16015022708025034,
0.15700501314829246,
0.1557417891860268,
0.16211847123415732,
0.16506185346816848,
0.1588260279648701,
0.15829465572081994,
0.15910913920299252,
0.1581775963344517,
0.1551773767269635,
0.1713122159809925,
0.16697316496525957,
0.15845377104077865,
0.15761171338702745,
0.15530977457412684,
0.15911786682341608,
0.15599198690267177,
0.15516833371159716,
0.1586512138044842,
0.17144105034003404,
0.16582578269062206,
0.15640253302935458,
```

0.16109567378784637,
0.16493583626410782,
0.16168288284887353,
0.15519633503070174,
0.15896398910388107,
0.15522664164644187,
0.155080174862907,
0.15779294908423724,
0.15528359383341742,
0.17093583055487924,
0.15670217300600262,
0.1551883715447636,
0.15513471903135984,
0.16315192562636147,
0.15589722347708915,
0.15829525882369297,
0.1698251826429798,
0.15569840660025536,
0.15508688232982162,
0.1553373117493539,
0.16023699836958627,
0.15512000832519673,
0.16703634785120172,
0.16810281942138683,
0.15525396819043893,
0.15562776630147693,
0.1642154646418482,
0.17236245172576203,
0.1555023440898003,
0.15503058836314337,
0.15773024803544092,
0.15643295738109667,
0.15504039693463326,
0.162893798943538,
0.1561151619831466,
0.1553908672781244,
0.1567170000511899,
0.15507709990888274,
0.1550201369005476,
0.16618750494505097,
0.1551874720118794,
0.1630337563147302,
0.1734434509072911,
0.1593628136951863,
0.15591502845091526,
0.15606483816673283,
0.16368423319193798,

0.16867936677973408,
0.15753586297505842,
0.1682918179743605,
0.15530354326757925,
0.15981668151205025,
0.17060554566377062,
0.17077366166036742,
0.16810793469142823,
0.16514616763799456,
0.16722872546905923,
0.15701045513687095,
0.1551924982085941,
0.1570866860952885,
0.15984896241373045,
0.15499667460038852,
0.17139687823930458,
0.15933371705778576,
0.15527687673953716,
0.1604573998905617,
0.15853490381580354,
0.16459361201775094,
0.15763193689524999,
0.1563971177386086,
0.16866131224183406,
0.1652761524168387,
0.15727526216722396,
0.156034612061039,
0.15768534697773426,
0.1561403099856117,
0.15536581896734228,
0.15505198522494254,
0.15503970929706734,
0.1550061722467379,
0.1717544393349149,
0.16549111223824706,
0.15793737671241465,
0.15530635372147739,
0.15520431255645678,
0.15695546038110683,
0.15504632792615403,
0.16366429933031879,
0.1616979872396653,
0.15612619534721198,
0.156581717152958,
0.1550031198182704,
0.16760291440753444,
0.15895016497783332,

```
0.1663960279150984,
0.1559834958198365,
0.1551388839725932,
0.15668416003033503,
0.156250342977349,
0.1556788252255903,
0.16143220500781455,
0.1595647280656055,
0.1551660367733277,
0.16334694209640435,
0.15523475104477918,
0.16039992346966375,
0.15499638803597726,
0.1705641491368218,
0.15503009676879545,
0.16689128720919177,
0.16700200979558144,
0.1637849163762202,
0.15583578043037427,
0.17357465895188043,
0.1563393942829785,
0.1550051904557303,
0.15614242569588177,
0.1561809532768876,
0.15514348488288737,
0.16556172453361065,
0.15560848650249973,
0.15670267830462703,
0.15577253153820395,
0.17236093395728277,
0.15517235277899533,
0.1610459824439474,
0.16236255659110005,
0.1557734186079514,
0.15679752728964416,
0.16069878597859297,
0.15534393266572205,
0.1571661385343679,
0.16235154833135482,
0.15711157066084366,
0.1549968407594702,
0.1573351648333412,
0.15504304291058377,
0.16819492565568525,
0.15535123895597067,
0.15502614824678299,
0.15541015824811114,
```

0.15602758753250517,
0.15579397012880786,
0.1550021145455794,
0.15562928116609567,
0.15599729500520926,
0.1586484512453061З,
0.1551389549141118,
0.1626033898382979,
0.15688831785485877,
0.1550625252343743,
0.15629734576615456,
0.15613018177660673,
0.15818080600796425,
0.15499820066388698,
0.15708408886841524,
0.15509856559817348,
0.15632906733984162,
0.15585358820121192,
0.15593128230296774,
0.15545434752134943,
0.1586658239362608,
0.1683533450776992,
0.16645832168629365,
0.17116053034751744,
0.15912806354890277,
0.1552553553098736,
0.15605358114094647,
0.16620068248963296,
0.15515132980030139,
0.16178046307065233,
0.16956719747134688,
0.15517380647281956,
0.1569425444742615,
0.1569945786056328,
0.16288186368188032,
0.1550352832887632,
0.15563526272303055,
0.15501657030569396,
0.1575800553943749,
0.15510747969493877,
0.15693307821628444,
0.17051153280715406,
0.1589982713403347,
0.15865091782772597,
0.16756123740055778,
0.15573198143049796,
0.15499817957051132,

0.15736525957815742,
0.17075277314958048,
0.16661609724811333,
0.15637422241391336,
0.16183494067059687,
0.16204926516497492,
0.16284010087735074,
0.16036529888736245,
0.1562927780835217,
0.15501085944406853,
0.1550308431548889,
0.1565119820443461,
0.1701565509877689,
0.15731271677772474,
0.15515359392933517,
0.1617351418881574,
0.16295418103538847,
0.15525908741624705,
0.157337372531286,
0.15549267791478974,
0.15734511437557266,
0.15867280511862736,
0.1620934719612345,
0.16293241026583893,
0.15842270442155693,
0.1647347669483986,
0.1575107037281076,
0.16989451345033205,
0.16678785750344213,
0.15627013035005055,
0.15516404326435296,
0.16267043264788966,
0.16017689923824419,
0.1550047265062506,
0.15500578215120045,
0.16240714182648114,
0.1558757543126449,
0.15585873098611916,
0.15586946779147207,
0.15510863404065908,
0.16049492412641847,
0.16987698656207126,
0.15504378935367627,
0.15500164462755747,
0.15563279839431246,
0.16078747299461812,
0.1591280239806396,

0.16400384887786415,
0.15650237178011303,
0.1550043664003317,
0.15527091922936323,
0.15697720052366135,
0.15515311985146696,
0.1573244192802305,
0.15511942571805792,
0.1557394467339961,
0.1682408955638948,
0.1667381176608143,
0.1553485809212926,
0.15738023206886667,
0.17053050554449445,
0.1558617579515526,
0.16505765934819586,
0.17118262586856506,
0.15513311066278584,
0.15656240071596791,
0.15549078907724417,
0.15930646017007172,
0.15715411914359187,
0.15521731468678587,
0.1553727341281035,
0.16958533402314918,
0.15762466641088485,
0.1568283666518272,
0.15510423901679407,
0.15801296300870507,
0.15606874630911427,
0.155025979998763,
0.17305155276143533,
0.15929932908590486,
0.1553455153959388,
0.16811399123771248,
0.16060119449775234,
0.15500200509741907,
0.15510858702076893,
0.16697383340027575,
0.1550054497184025,
0.15531901237432838,
0.15506713936700225,
0.1566908574795784,
0.15499718526482473,
0.1624630695179031,
0.1645578322608359,
0.15775654320334942,

0.15506638482822982,
0.1559850440853829,
0.15502552563506453,
0.15707602327760606,
0.1588050015605344,
0.15542095374768372,
0.15513093886982915,
0.1553878307285821,
0.15650864641324747,
0.1575525594978137,
0.15539190058188365,
0.15539655947611786,
0.1577574755588921,
0.1553522800833067,
0.16729045046142765,
0.15840067715465841,
0.16171796314917364,
0.15574442474310732,
0.15939674458836284,
0.1559047789087656,
0.15500917533314695,
0.15619571513884475,
0.15921623725918735,
0.15703665022398675,
0.15521417317886574,
0.15499638040092723,
0.15920333207045528,
0.15515364380985683,
0.1550007464730745,
0.15897082907968865,
0.1576423585682393,
0.1581786634555004,
0.15657476859999567,
0.16374049283841352,
0.1682252465049379,
0.15911970401921785,
0.15501948093994208,
0.1551049883121833,
0.15557085239083931,
0.15585899130026806,
0.15517933062376663,
0.15660672657988858,
0.15894014724488228,
0.15700072648311747,
0.1691820015808929,
0.15559736384916498,
0.159409810448235,

```
0.16859278433499664,
0.15547280228732957,
0.1552789656794323,
0.15608991678702855,
0.1717898026013916,
0.15863220770681205,
0.16212531690264076,
0.15503003492037423,
0.16280507255907098,
0.155052491088839 1,
0.16081913202097195,
0.1564957786695928,
0.1605216164117663,
0.16560722432848551,
0.1643186759419461,
0.15973757029303984,
0.1554185985673121,
0.16289134983974604,
0.15622733610588802,
0.15500082835972875,
0.1570078843359922,
0.1698877608609299,
0.15560845597223608,
0.16207829617653843,
0.1613910579999036,
0.15589571286744786,
0.15835261477080084])
```

[119]:
```
portfolios=pd.DataFrame({'Return':pfolio_returns,'volatility':
 ↪pfolio_volatilities})
```

[120]:
```
portfolios.head()
```

[120]:

	Return	volatility
0	0.095601	0.164274
1	0.094571	0.155036
2	0.095273	0.159118
3	0.094219	0.156740
4	0.093465	0.168357

[121]:
```
portfolios.tail()
```

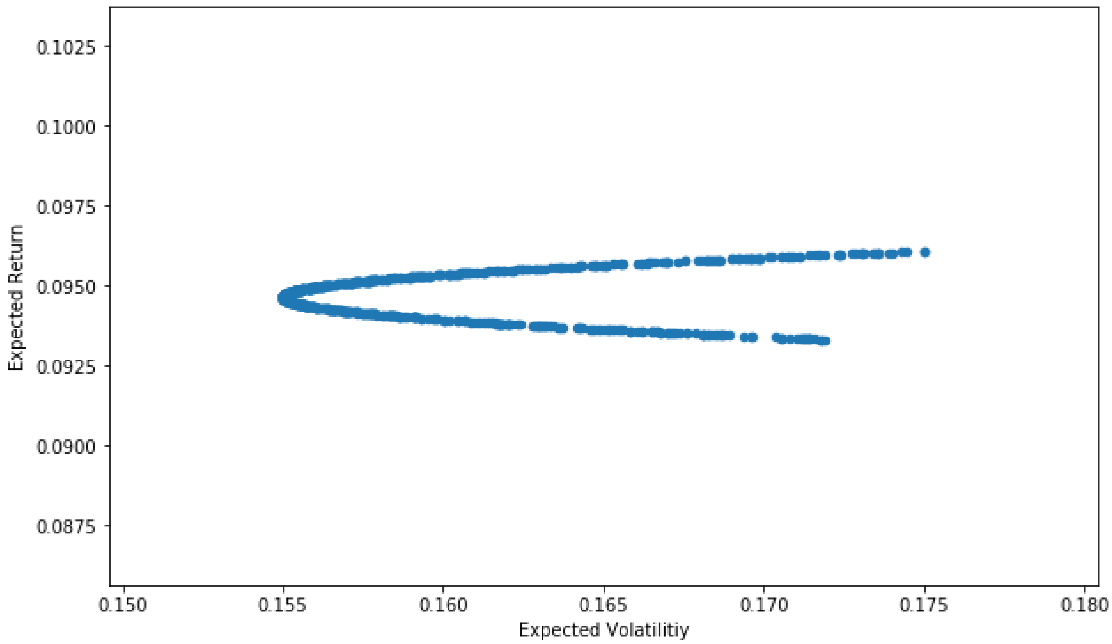
[121]:

	Return	volatility
995	0.094878	0.155608
996	0.095476	0.162078
997	0.095433	0.161391
998	0.094930	0.155896

999   0.094057   0.158353

[123]:
```
portfolios.plot(x='volatility',y='Return',kind='scatter', figsize=(10, 6));
plt.xlabel('Expected Volatilitiy')
plt.ylabel('Expected Return')
```

[123]: Text(0, 0.5, 'Expected Return')



[ ]:
```

```

[125]:
```
import numpy as np
import pandas as pd
from pandas_datareader import data as web

tickers = ['PG','^GSPC']
data=pd.DataFrame()
for t in tickers:
    data[t]=web.DataReader(t,data_source='yahoo',␣
 ↪start='2012-1-1',end='2016-12-31')['Adj Close']
```

[130]:
```
sec_returns=np.log(data/data.shift(1))
```

[131]:
```
cov=sec_returns.cov()*250
cov
```

[131]:
```
              PG     ^GSPC
PG      0.020409  0.010078
^GSPC   0.010078  0.016362
```

[132]:
```
cov_with_market=cov.iloc[0,1]
```

[133]:
```
cov_with_market
```

[133]:
```
0.010078060959412657
```

[134]:
```
market_var=sec_returns['^GSPC'].var()*250
market_var
```

[134]:
```
0.016361631002308474
```

[135]:
```
#BETA
```

[136]:
```
PG_beta=cov_with_market/market_var
```

[137]:
```
PG_beta
```

[137]:
```
0.6159569885172655
```

[138]:
```
#CAPM
```

[139]:
```
PG_er=0.025+PG_beta*0.05
PG_er
```

[139]:
```
0.05579784942586328
```

[140]:
```
#SharpeRatio
```

[141]:
```
Sharpe=(PG_er-0.025)/(sec_returns['PG'].std()*250**0.5)
Sharpe
```

[141]:
```
0.2155797835537668
```

[151]:
```
import numpy as np
import pandas as pd
from scipy import stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
data=pd.read_excel ("/Users/alessandrosollazzo/Desktop/original.xlsx")
data
```

[151]:
```
    House Price  House Size (sq.ft.) State  Number of Rooms  \
0       1116000                 1940    IN                8
```

```
1    860000                  1300    IN              5
2    818400                  1420    IN              6
3    1000000                 1680    IN              7
4    640000                  1270    IN              5
5    1010000                 1850    IN              7
6    600000                  1000    IN              4
7    700000                  1100   LA               4
8    1100000                 1600   LA               7
9    570000                  1000    NY              5
10   860000                  2150    NY              9
11   1085000                 1900    NY              9
12   1250000                 2200    NY              9
13   850000                  1100    TX              4
14   640000                   860    TX              4
15   900000                  1325    TX              6
16   730000                  1350    TX              6
17   750000                  1600    TX              6
18   650000                   950    TX              2
19   680000                  1250    TX              4

    Year of Construction
0                   2002
1                   1992
2                   1987
3                   2000
4                   1995
5                   1998
6                   2015
7                   2014
8                   2017
9                   1997
10                  1997
11                  2000
12                  2014
13                  2017
14                  1997
15                  1997
16                  2000
17                  1992
18                  1987
19                  2000
```

[158]:
```
X=data[['House Size (sq.ft.)','Number of Rooms','Year of Construction']]
Y=data['House Price']
```

[159]:
```
X1=sm.add_constant(x)
```

[160]:
```python
from scipy import stats
import statsmodels.api as sm
```

[161]:
```python
X1=sm.add_constant(x)
```

[162]:
```python
data[['House Price','House Size (sq.ft.)','Year of Construction']]
```

[162]:

	House Price	House Size (sq.ft.)	Year of Construction
0	1116000	1940	2002
1	860000	1300	1992
2	818400	1420	1987
3	1000000	1680	2000
4	640000	1270	1995
5	1010000	1850	1998
6	600000	1000	2015
7	700000	1100	2014
8	1100000	1600	2017
9	570000	1000	1997
10	860000	2150	1997
11	1085000	1900	2000
12	1250000	2200	2014
13	850000	1100	2017
14	640000	860	1997
15	900000	1325	1997
16	730000	1350	2000
17	750000	1600	1992
18	650000	950	1987
19	680000	1250	2000

[163]:
```python
X=data[['House Size (sq.ft.)','Number of Rooms','Year of Construction']]
Y=data['House Price']
```

[164]:
```python
X1=sm.add_constant(x)
```

[165]:
```python
reg=sm.OLS(Y,X1).fit()
```

```
␣
↪---------------------------------------------------------------------------

    ValueError                                Traceback (most recent call␣
↪last)

    <ipython-input-165-fb20264ce9ea> in <module>
  ----> 1 reg=sm.OLS(Y,X1).fit()
```

```
    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, missing, hasconst, **kwargs)
    857                  **kwargs):
    858         super(OLS, self).__init__(endog, exog, missing=missing,
--> 859                                   hasconst=hasconst, **kwargs)
    860         if "weights" in self._init_keys:
    861             self._init_keys.remove("weights")


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, weights, missing, hasconst,␣
↪**kwargs)
    700             weights = weights.squeeze()
    701         super(WLS, self).__init__(endog, exog, missing=missing,
--> 702                                   weights=weights,␣
↪hasconst=hasconst, **kwargs)
    703         nobs = self.exog.shape[0]
    704         weights = self.weights


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, **kwargs)
    188     """
    189     def __init__(self, endog, exog, **kwargs):
--> 190         super(RegressionModel, self).__init__(endog, exog, **kwargs)
    191         self._data_attr.extend(['pinv_wexog', 'wendog', 'wexog',␣
↪'weights'])
    192


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪__init__(self, endog, exog, **kwargs)
    234
    235     def __init__(self, endog, exog=None, **kwargs):
--> 236         super(LikelihoodModel, self).__init__(endog, exog, **kwargs)
    237         self.initialize()
    238


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪__init__(self, endog, exog, **kwargs)
     75         hasconst = kwargs.pop('hasconst', None)
     76         self.data = self._handle_data(endog, exog, missing, hasconst,
---> 77                                       **kwargs)
     78         self.k_constant = self.data.k_constant
     79         self.exog = self.data.exog
```

```
    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪_handle_data(self, endog, exog, missing, hasconst, **kwargs)
     98
     99     def _handle_data(self, endog, exog, missing, hasconst, **kwargs):
 --> 100         data = handle_data(endog, exog, missing, hasconst, **kwargs)
    101         # kwargs arrays could have changed, easier to just attach␣
↪here
    102         for key in kwargs:


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪handle_data(endog, exog, missing, hasconst, **kwargs)
    670     klass = handle_data_class_factory(endog, exog)
    671     return klass(endog, exog=exog, missing=missing,␣
↪hasconst=hasconst,
 --> 672                  **kwargs)


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪__init__(self, endog, exog, missing, hasconst, **kwargs)
     81             self.orig_endog = endog
     82             self.orig_exog = exog
 ---> 83             self.endog, self.exog = self._convert_endog_exog(endog,␣
↪exog)
     84
     85         self.const_idx = None


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪_convert_endog_exog(self, endog, exog)
    508             raise ValueError("Pandas data cast to numpy dtype of␣
↪object. "
    509                              "Check input data with np.asarray(data).
↪")
 --> 510         return super(PandasData, self)._convert_endog_exog(endog,␣
↪exog)
    511
    512     @classmethod


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪_convert_endog_exog(self, endog, exog)
    324                 xarr = xarr[:, None]
    325             if xarr.ndim != 2:
 --> 326                 raise ValueError("exog is not 1d or 2d")
    327
```

```
328         return yarr, xarr
```

```
ValueError: exog is not 1d or 2d
```

[166]:
```
import numpy as np
```

[169]:
```
import pandas as pd
```

[170]:
```
from scipy import stats
```

[171]:
```
import statsmodels.api as sm
```

[172]:
```
import matplotlib.pyplot as plt
```

[173]:
```
data=pd.read_excel ("/Users/alessandrosollazzo/Desktop/original.xlsx")
```

[174]:
```
data
```

[174]:
```
    House Price  House Size (sq.ft.) State  Number of Rooms  \
0       1116000                 1940    IN                8
1        860000                 1300    IN                5
2        818400                 1420    IN                6
3       1000000                 1680    IN                7
4        640000                 1270    IN                5
5       1010000                 1850    IN                7
6        600000                 1000    IN                4
7        700000                 1100   LA                 4
8       1100000                 1600   LA                 7
9        570000                 1000    NY                5
10       860000                 2150    NY                9
11      1085000                 1900    NY                9
12      1250000                 2200    NY                9
13       850000                 1100    TX                4
14       640000                  860    TX                4
15       900000                 1325    TX                6
16       730000                 1350    TX                6
17       750000                 1600    TX                6
18       650000                  950    TX                2
19       680000                 1250    TX                4

    Year of Construction
0                   2002
1                   1992
2                   1987
3                   2000
```

```
4                    1995
5                    1998
6                    2015
7                    2014
8                    2017
9                    1997
10                   1997
11                   2000
12                   2014
13                   2017
14                   1997
15                   1997
16                   2000
17                   1992
18                   1987
19                   2000
```

[175]:

```python
X=data[['House Size (sq.ft.)','Number of Rooms','Year of Construction']]
Y=data['House Price']
```

[176]:

```python
X
```

[176]:

	House Size (sq.ft.)	Number of Rooms	Year of Construction
0	1940	8	2002
1	1300	5	1992
2	1420	6	1987
3	1680	7	2000
4	1270	5	1995
5	1850	7	1998
6	1000	4	2015
7	1100	4	2014
8	1600	7	2017
9	1000	5	1997
10	2150	9	1997
11	1900	9	2000
12	2200	9	2014
13	1100	4	2017
14	860	4	1997
15	1325	6	1997
16	1350	6	2000
17	1600	6	1992
18	950	2	1987
19	1250	4	2000

[178]:

```python
Y
```

[178]:
```
0     1116000
1      860000
2      818400
3     1000000
4      640000
5     1010000
6      600000
7      700000
8     1100000
9      570000
10     860000
11    1085000
12    1250000
13     850000
14     640000
15     900000
16     730000
17     750000
18     650000
19     680000
Name: House Price, dtype: int64
```

[179]:
```
X1=sm.add_constant(x)
```

[180]:
```
reg=sm.OLS(Y,X1).fit()
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-180-fb20264ce9ea> in <module>
   ----> 1 reg=sm.OLS(Y,X1).fit()


       ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, missing, hasconst, **kwargs)
       857                  **kwargs):
       858         super(OLS, self).__init__(endog, exog, missing=missing,
   --> 859                                   hasconst=hasconst, **kwargs)
       860         if "weights" in self._init_keys:
       861             self._init_keys.remove("weights")
```

```
    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, weights, missing, hasconst,␣
↪**kwargs)
    700             weights = weights.squeeze()
    701         super(WLS, self).__init__(endog, exog, missing=missing,
--> 702                                   weights=weights,␣
↪hasconst=hasconst, **kwargs)
    703         nobs = self.exog.shape[0]
    704         weights = self.weights


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/regression/
↪linear_model.py in __init__(self, endog, exog, **kwargs)
    188     """
    189     def __init__(self, endog, exog, **kwargs):
--> 190         super(RegressionModel, self).__init__(endog, exog, **kwargs)
    191         self._data_attr.extend(['pinv_wexog', 'wendog', 'wexog',␣
↪'weights'])
    192


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪__init__(self, endog, exog, **kwargs)
    234
    235     def __init__(self, endog, exog=None, **kwargs):
--> 236         super(LikelihoodModel, self).__init__(endog, exog, **kwargs)
    237         self.initialize()
    238


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪__init__(self, endog, exog, **kwargs)
     75         hasconst = kwargs.pop('hasconst', None)
     76         self.data = self._handle_data(endog, exog, missing, hasconst,
---> 77                                       **kwargs)
     78         self.k_constant = self.data.k_constant
     79         self.exog = self.data.exog


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/model.py in␣
↪_handle_data(self, endog, exog, missing, hasconst, **kwargs)
     98
     99     def _handle_data(self, endog, exog, missing, hasconst, **kwargs):
--> 100         data = handle_data(endog, exog, missing, hasconst, **kwargs)
    101         # kwargs arrays could have changed, easier to just attach␣
↪here
    102         for key in kwargs:
```

```
    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪handle_data(endog, exog, missing, hasconst, **kwargs)
    670     klass = handle_data_class_factory(endog, exog)
    671     return klass(endog, exog=exog, missing=missing,␣
↪hasconst=hasconst,
--> 672                  **kwargs)


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪__init__(self, endog, exog, missing, hasconst, **kwargs)
     81             self.orig_endog = endog
     82             self.orig_exog = exog
---> 83             self.endog, self.exog = self._convert_endog_exog(endog,␣
↪exog)
     84
     85         self.const_idx = None


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪_convert_endog_exog(self, endog, exog)
    508             raise ValueError("Pandas data cast to numpy dtype of␣
↪object. "
    509                              "Check input data with np.asarray(data).
↪")
--> 510         return super(PandasData, self)._convert_endog_exog(endog,␣
↪exog)
    511
    512     @classmethod


    ~/opt/anaconda3/lib/python3.7/site-packages/statsmodels/base/data.py in␣
↪_convert_endog_exog(self, endog, exog)
    324                 xarr = xarr[:, None]
    325             if xarr.ndim != 2:
--> 326                 raise ValueError("exog is not 1d or 2d")
    327
    328         return yarr, xarr


    ValueError: exog is not 1d or 2d
```

[ ]: