Project Overview:

- For this project you will need to fetch Adj Close every company in the Russell 2000 for 2020 (Time Frame Jan 1 - current)
- We would like to see the Dips and Gains/ Daily Returns of every stock under priced under 10 dollars in ascending order.
- Repeat the same process for every company in the entire index
- Using portfolio optimization techniques find 20 companies in a portfolio that would generate a "Good" return during the current Pandemic.
- We want you to than go back 5 years and see how these companies performed pre-Covid. What is the max return we would have received during this 5 year period considering volatility.

In [ ]:

Project Approach

- Fetch all companies in Russell 2000
- Use pandas Datareader to get the Adj Close
- Calculate the Daily returns for each security
- Find the top 20 Gains and Dips for returns for the current day
- Put every security into a portfolio and optimize the portfolio with 5 years of data (2015-2019 end)
- Find the top companies the make up the index by weight and isolate them into a portfolio of their own
- Run a portfolio optimization on the portfolio using Markowitz Efficient Frontier and see what the ortfolio would have returned pre-Covid.

Acknowledgements

- Data was downloaded provided from Ben Reynolds at Secure Dividends
- https://www.suredividend.com/ (https://www.suredividend.com/)

libraries

In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas_datareader import data as web
```

In [11]:
```
russ = pd.read_excel("resources/Russle_2000_2020.xlsx", index_col='Ticker')
```

Data Overview

- Data below provides all of the data tickers and companies in the Russell 2000

In [12]:
```
russ
```

Out[12]:

Ticker	Name
AAN	Aaron's, Inc.
AAOI	Applied Optoelectronics, Inc.
AAON	AAON, Inc.
AAT	American Assets Trust, Inc.
AAWW	Atlas Air Worldwide Holdings, Inc.
...	...
ZIXI	Zix Corp.
ZUMZ	Zumiez, Inc.
ZUO	Zuora, Inc.
ZYNE	Zynerba Pharmaceuticals, Inc.
ZYXI	Zynex, Inc.

1999 rows × 1 columns

In [13]:
```
russ.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1999 entries, AAN to ZYXI
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Name    1999 non-null   object
dtypes: object(1)
memory usage: 31.2+ KB
```

Adj Closes

- Will import data already fetched previously
- Will download the Adj Close for each security for 2020 beginning 2020-1-1 to current using pandas Datareader

In [14]:
```
tickers = list(russ.index)
```

In [84]:
```
len(tickers)
```

Out[84]: 1999

In [16]:
```
data = pd.DataFrame()
```

In [17]:
```
failed = []
passed = []
for x in tickers:
    try:
        data[x] = web.DataReader(x, data_source= "yahoo", start = "2020-1-1")["Adj Close"]
        passed.append(x)
    except (IOError, KeyError):
        msg = 'Failed to read symbol: {0!r}, replacing with NaN.'
        failed.append(x)
```

Passed Tickers

- There are 1941 tickers that passed in the Yahoo data reader library
- We will use these for our analysis

In [23]:
```
len(passed)
```

Out[23]: 1941

Saving Passed Tickers

In [28]:
```
#pd.DataFrame(passed, columns=["passed"]).to_csv("resources/current/passed_tickers.csv")
```

In [ ]:
```

```

Penny Stocks

- Will consider all socks under 10 dollas a penny stocks for this project
- There are 346 penny tickers

In [46]:
```
penny_tickers = list(data[data < 10].dropna(axis = 1).columns)
```

In [49]:
```
len(penny_tickers)
```

Out[49]: 346

In [85]:
```
data[penny_tickers].head(10)
```

Out[85]:

Date	ABEO	ACER	ACOR	ACRS	ACRX	ACTG	ADMA	ADMS	ADRO	AFI	...	WTI	WTRH
2020-01-02	3.21	3.79	1.96	1.87	2.07	2.63	3.935	4.08	1.18	4.18	...	5.42	0.343
2020-01-03	2.90	3.58	2.27	1.84	2.04	2.62	3.760	4.10	1.21	4.07	...	5.80	0.369
2020-01-06	2.77	3.69	2.48	1.87	2.01	2.71	3.670	4.14	1.33	4.09	...	5.75	0.380
2020-01-07	2.57	3.71	2.44	1.85	2.07	2.71	3.760	4.16	1.34	4.08	...	5.81	0.485
2020-01-08	2.62	3.69	2.26	1.86	2.03	2.70	3.720	5.11	1.39	4.06	...	5.37	0.499
2020-01-09	2.61	3.86	2.28	1.86	1.97	2.70	4.400	5.55	1.33	4.05	...	5.29	0.418
2020-01-10	2.54	4.09	2.36	2.18	1.92	2.68	4.510	5.19	1.19	4.07	...	5.13	0.394
2020-01-13	2.36	4.00	2.42	2.11	1.91	2.73	4.350	5.50	1.15	4.01	...	5.06	0.370
2020-01-14	2.58	4.17	2.34	1.99	1.96	2.69	4.440	5.45	1.19	3.97	...	5.25	0.354
2020-01-15	2.64	4.18	2.13	1.98	1.99	2.67	4.400	5.70	1.24	4.18	...	5.25	0.378

10 rows × 346 columns

In [53]:
```
#pd.DataFrame(penny_tickers, columns=["under 10"]).to_csv("resources/current/penny_tickers.csv")
```

Checking the Daily returns for the Russell 2000

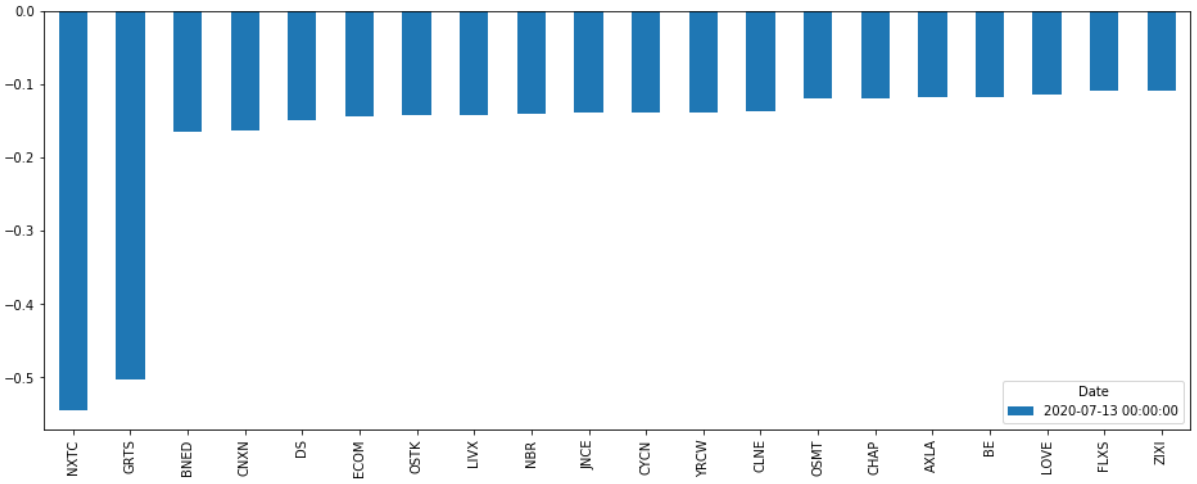
- Will check today's Date for the Dips and Spikes in the entire index
- Will find the to 20 companies that had a negative return for todays date

Dips

In [75]:
```
top_20_dips = data.pct_change()[-1:].transpose().sort_values("2020-07-13", ascending = True).dropna().head(20)
```

In [76]:
```
top_20_dips.plot(kind = "bar", figsize = (16,6))
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2a2615d0>



Spikes

In [74]:
```
top_20_Spikes = data.pct_change()[-1:].transpose().sort_values("2020-07-13", ascending = False).dropna().head(20)
```

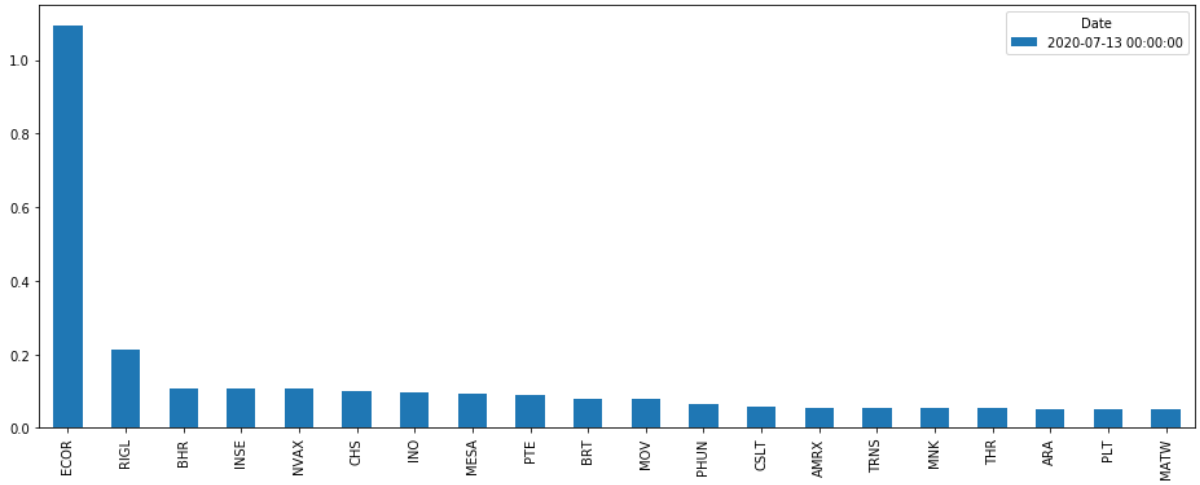
In [77]:
```
top_20_Spikes
```

Out[77]:

Date	2020-07-13
ECOR	1.094118
RIGL	0.213483
BHR	0.107438
INSE	0.106707
NVAX	0.105553
CHS	0.100775
INO	0.095808
MESA	0.093458
PTE	0.089431
BRT	0.078035
MOV	0.077228
PHUN	0.065574
CSLT	0.059524
AMRX	0.055914
TRNS	0.055372
MNK	0.055336
THR	0.052592
ARA	0.051887
PLT	0.050813
MATW	0.050489

In [80]:
```
top_20_Spikes.plot(kind = "bar", figsize = (16,6))
```

Out[80]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d6d0210>
```



In [ ]:
```
```

Optimizing the Russell 2000 as a portfolio

- Will put every company in a portfolio and see which companies would have held the mose weight during the 2020 covid 19 crisis thu far.
- Will use markowitz Portfolio theor to optimize the Russell
- Will use 2000 randomly allocated portfolios to get the efficient frontier using a combination of volatility and expected returns
- We will drop companies with missing data for the time being

In [102]:
```
russell_Returns = data.dropna(axis=1).pct_change()
```

In [103]:
```
tic = list(russell_Returns.columns)
```

In [104]:
```
n_portfolios = 2000
all_weights = np.zeros((n_portfolios, len(tic)))
all_returns = np.zeros(n_portfolios)
all_vol = np.zeros(n_portfolios)
all_sharp = np.zeros(n_portfolios)

for ind in range(n_portfolios):
    weights = np.array(np.random.random(len(tic)))
    weights = weights/weights.sum()
    all_weights[ind,:] = weights

    all_returns[ind] = np.sum(russell_Returns.mean() * weights) * 252
    all_vol[ind] = np.dot(weights.T, np.dot(russell_Returns.cov() * 252,
weights))
    all_sharp[ind] = all_returns[ind]/ all_vol[ind]
```

Plotting the Frontier

- The frontier will provide a good insight on the max return hightest sharp and the lowest volatility of the portfolios

In [ ]:

In [106]:
```
hightest_return = all_returns.argmax()
```

In [107]:
```
lowest_vol = all_vol.argmin()
```

In [108]:
```
higherst_sharp = all_sharp.argmax()
```

In [127]:
```
all_returns.max()
```

Out[127]: -0.01172690739902448

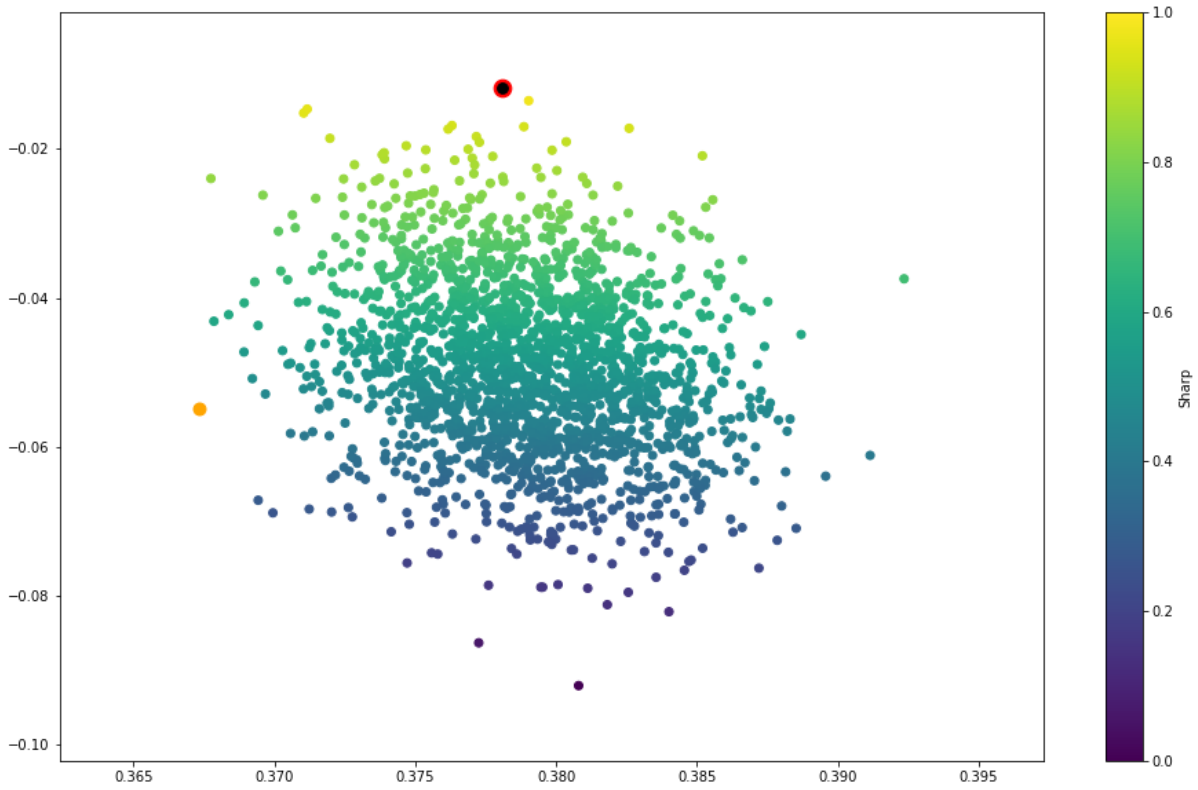
Frontier

- We see two things in the froniter
- The portfolio with the highest sharp is provides the same return as the portfolio with the highrest return
- Appears the highest return for a portfolio containing every company in the Russell is in the negative
- This is not realistic and the main purpose was to find the top 20 weighted companies in 2020
- We will fetch these below

In [126]:
```
plt.figure(figsize = (16,10))
plt.scatter(all_vol, all_returns, c=all_sharp)
plt.scatter(all_vol[hightest_return], all_returns[hightest_return], c=
"r", s = 160)
plt.scatter(all_vol[lowest_vol], all_returns[lowest_vol], c="orange", s
= 80)
plt.scatter(all_vol[higherst_sharp], all_returns[higherst_sharp], c="bla
ck", s = 60)

plt.colorbar(label = "Sharp")
```

Out[126]: <matplotlib.colorbar.Colorbar at 0x1a3c0a1c90>



Bringing the data together

- Creating a data frame for the weights returns and sharp

In [145]:
```
frontier_df = pd.DataFrame(all_returns, columns=["returns"])
```

In [146]:
```
frontier_df = pd.concat([frontier_df, pd.DataFrame(all_vol, columns=["vo
l"])], axis=1)
```

In [149]:
```
shp_df = pd.DataFrame(all_sharp, columns=["shp"])
```

In [150]:
```
frontier_df = pd.concat([frontier_df, shp_df], axis=1)
```

In [152]:
```
weights_df = pd.DataFrame(all_weights, columns=tic)
```

In [154]:
```
frontier_df = pd.concat([frontier_df, weights_df], axis=1)
```

In [156]:
```
## frontier_df.to_csv("resources/current/frontier_entier_russell.csv")
```

Lets find the portfolio with the highest returns

- Will take the top 20 compaines by weight in this portfolio and create a new portfolio with just these 20 companies

In [168]:
```
highest_ret_port = pd.DataFrame(frontier_df.sort_values("returns", ascending = False).loc[1869])
```

In [171]:
```
highest_ret_port.drop(["returns", "vol", "shp"], inplace=True)
```

Below are the top 20 weighted companies

- will create a portfolio for just these companies and see what our returns would have been for 2020

In [179]:
```
top_20_weighted = highest_ret_port.sort_values(1869, ascending = False).head(20)
```

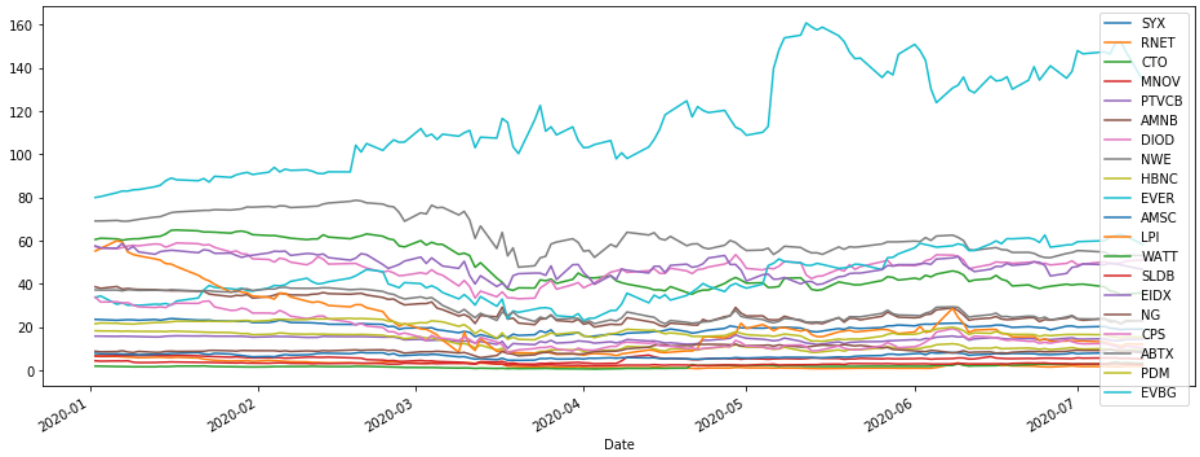
In [182]:
```
top20_tickers = list(top_20_weighted.index)
```

In [184]:
```
top_20_df = pd.DataFrame()
```

In [185]:
```
for y in top20_tickers:
    top_20_df[y] = web.DataReader(y, data_source="yahoo", start = "2020-1-1")["Adj Close"]
```

In [188]:
```
top_20_df.plot(figsize = (16,6))
```

Out[188]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3c4d6290>
```



In [189]:
```
top_20_df
```

Out[189]:

Date	SYX	RNET	CTO	MNOV	PTVCB	AMNB	DIOD	NWE	HBN
2020-01-02	23.590525	6.60	60.556934	6.68	15.847412	38.665291	57.299999	69.122437	18.33245
2020-01-03	23.487511	6.45	61.150726	6.63	15.817844	37.958714	56.639999	69.122437	18.24464
2020-01-06	23.187830	6.47	60.834038	6.80	15.729148	38.773235	56.320000	69.377502	18.20561
2020-01-07	23.337669	6.29	60.240242	6.91	15.817844	37.566174	56.840000	69.024338	18.07878
2020-01-08	23.440685	6.20	60.636105	6.87	15.788280	37.909649	57.549999	69.024338	18.19585
...	...	...	...	...	...	...	...	...	.
2020-07-07	19.520000	1.75	36.759998	5.70	14.000000	22.450001	49.299999	53.299999	9.58000
2020-07-08	19.320000	1.69	36.230000	5.78	13.710000	22.590000	49.310001	53.570000	9.46000
2020-07-09	18.820000	1.71	35.000000	5.49	13.860000	21.510000	49.790001	52.419998	9.05000
2020-07-10	19.070000	1.97	35.270000	5.60	14.290000	23.070000	50.570000	53.320000	9.48000
2020-07-13	19.080000	1.88	35.860001	5.23	14.430000	23.030001	51.070000	53.130001	9.58000

133 rows × 20 columns

In [190]:
```
top_20_returns = top_20_df.pct_change()
```

Volatility

- This portfolio has a high vol minly occurring with 2 companies
- Lets calculate the volatility below

In [198]:
```
np.sum(top_20_returns.mean() * 252)
```

Out[198]:
```
0.17715309902614762
```

In [200]:
```
top_20_returns.plot(figsize = (16,6), legend = False)
```

Out[200]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a32930d50>
```

In [201]:
```
n_portfolios = 2000
all_weights = np.zeros((n_portfolios, len(top_20_df.columns)))
all_returns = np.zeros(n_portfolios)
all_vol = np.zeros(n_portfolios)
all_sharp = np.zeros(n_portfolios)

for ind in range(n_portfolios):
    weights = np.array(np.random.random(len(top_20_df.columns)))
    weights = weights/weights.sum()
    all_weights[ind,:] = weights

    all_returns[ind] = np.sum(top_20_returns.mean() * weights) * 252
    all_vol[ind] = np.dot(weights.T, np.dot(top_20_returns.cov() * 252,
weights))
    all_sharp[ind] = all_returns[ind]/ all_vol[ind]
```

In [202]:
```
max_return = all_returns.argmax()
```

In [203]:
```
max_Sharp = all_sharp.argmax()
```

In [204]:
```
lowest_vol = all_vol.argmin()
```

Summary

- This portfolio would have yielded you 38% return year to date
- Not bad even during Covid

In [208]:
```
all_returns.max()
```

Out[208]:
```
0.38919563229425524
```

In [209]:
```
all_sharp.max()
```

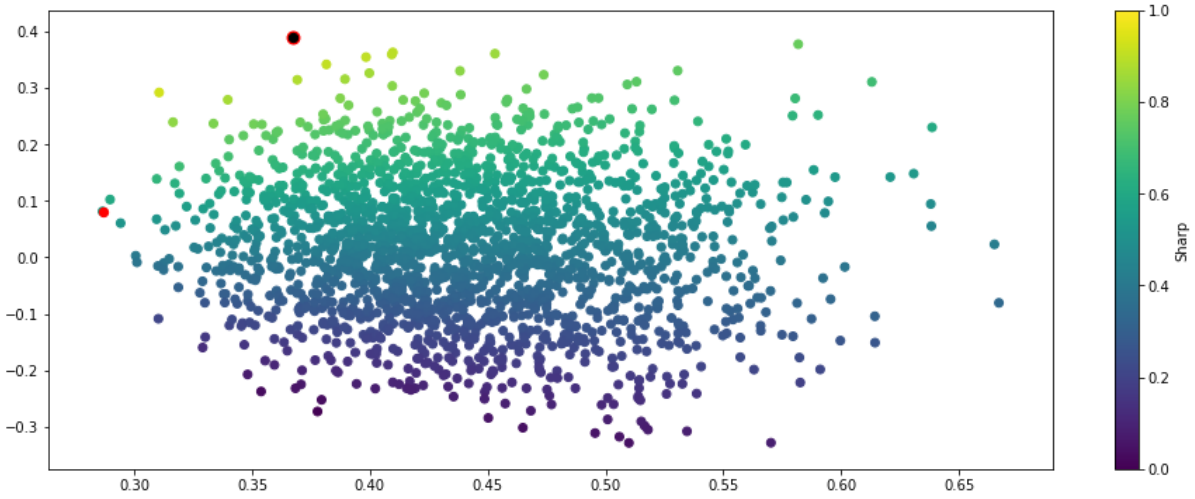
Out[209]:
```
1.0604261126275902
```

In [210]:
```
all_vol.min()
```

Out[210]:
```
0.2865265340650774
```

In [218]:
```
plt.figure(figsize=(16,6))
plt.scatter(all_vol, all_returns, c=all_sharp)
plt.scatter(all_vol[max_return], all_returns[max_return], c="r", s = 80)
plt.scatter(all_vol[max_Sharp], all_returns[max_Sharp], c="black", s = 4
0)
plt.scatter(all_vol[lowest_vol], all_returns[lowest_vol], c="r", s = 40)
plt.colorbar(label = "Sharp")
```

Out[218]:
```
<matplotlib.colorbar.Colorbar at 0x1a38170b90>
```



In [ ]:
```

```