Financial Engineering Group 1

Yunhan (Ryanna) Liu
Haoyu (Ryan) Wang
Anqi (Angela) Zhao

09/26/2020

Content

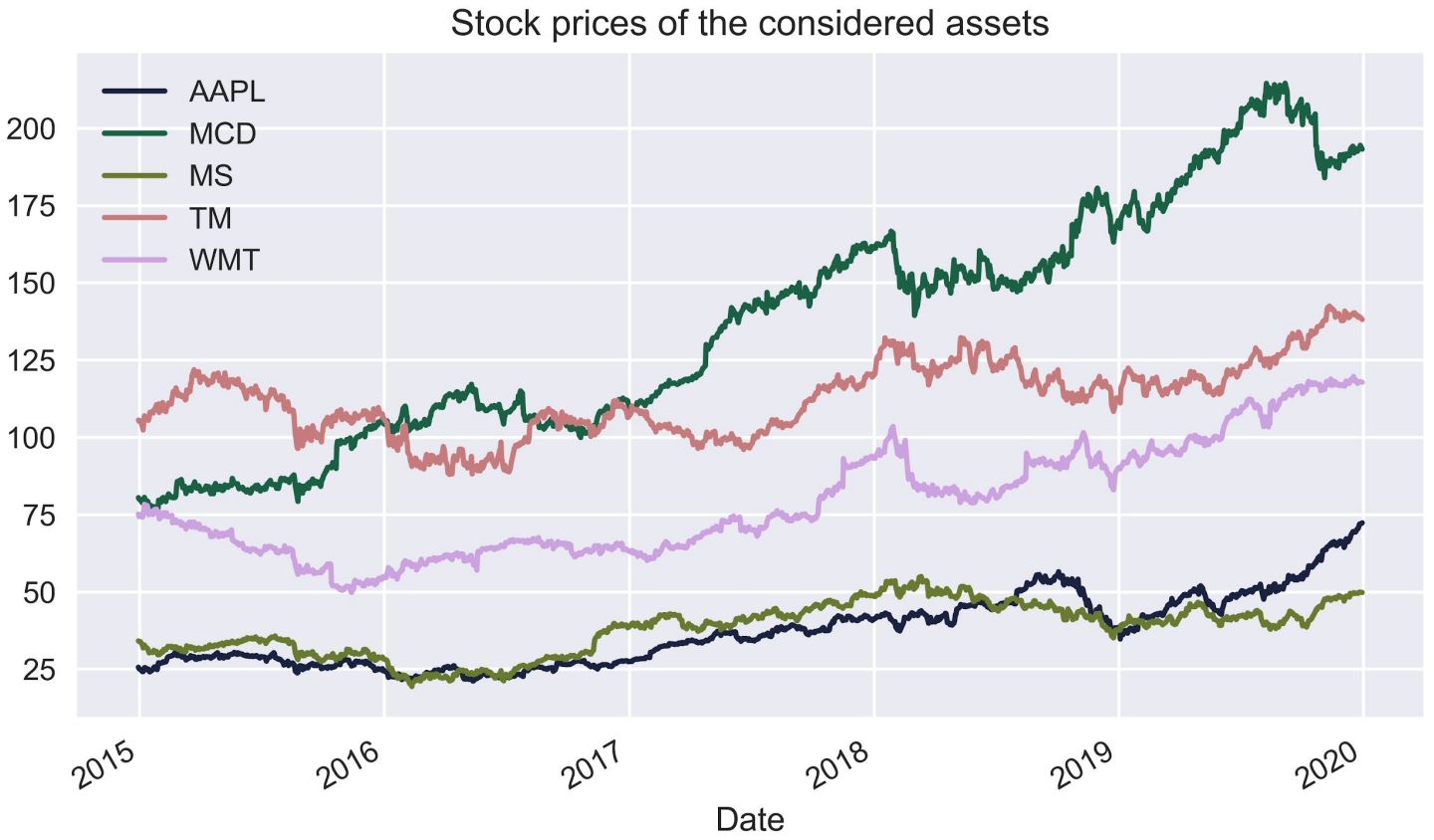
- Yahoo Finance
- Apple, McDonald, Morgan Stanley, Toyota, Walmart
- 2014/12/31 - 2019/12/31

- **Portfolio Optimization**
- **Model Fitting**
- **Prediction for Future Returns**

Part I. Portfolio Optimization

1.Evaluating the performance of a basic 1/n portfolio

Stock prices



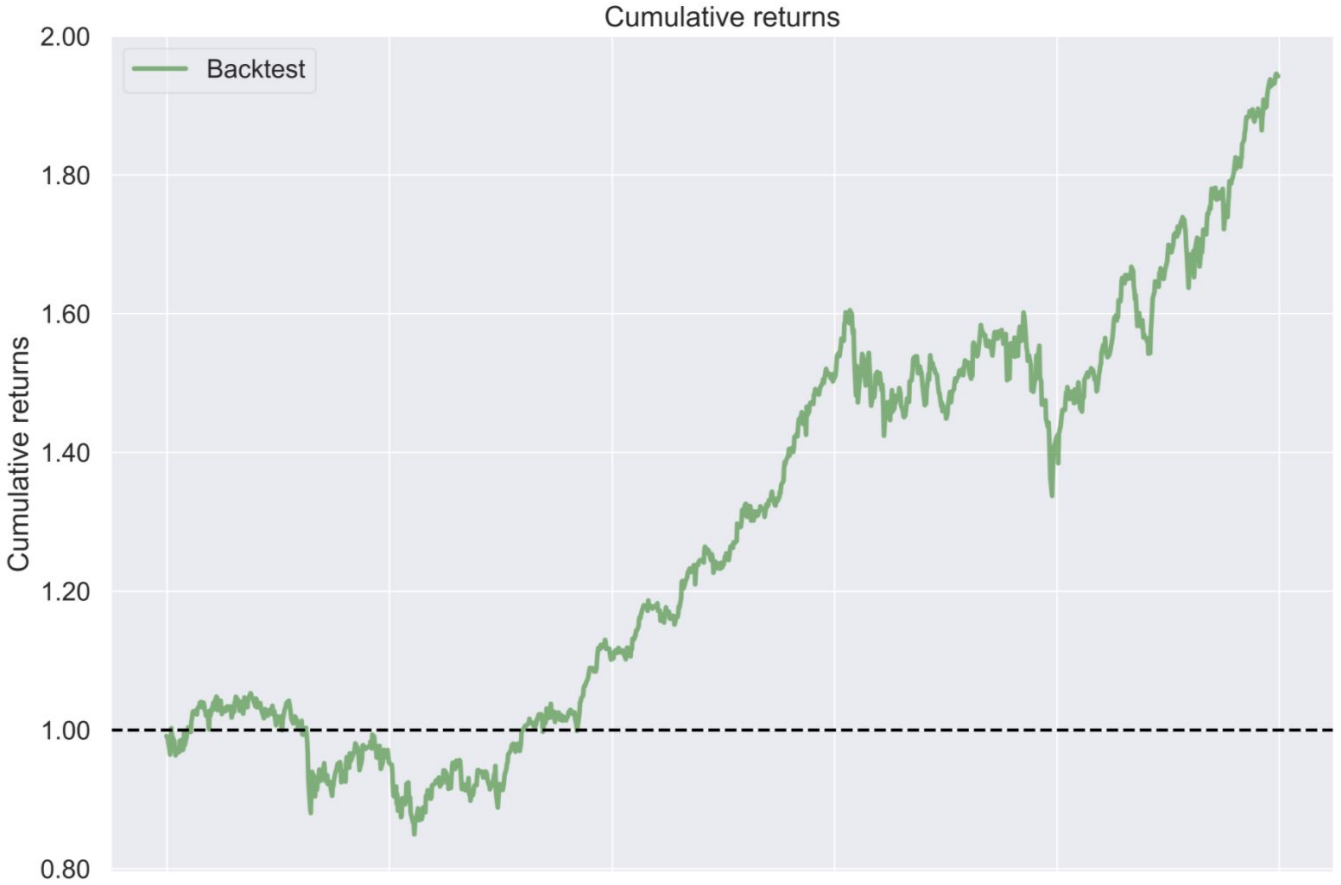
Summary

```
In [73]: pf.create_simple_tear_sheet(portfolio_returns)
```

Start date	2014-12-31
End date	2019-12-30
Total months	59

	Backtest
Annual return	14.2%
Cumulative returns	94.2%
Annual volatility	14.6%
Sharpe ratio	0.99
Calmar ratio	0.74
Stability	0.88
Max drawdown	-19.3%
Omega ratio	1.19
Sortino ratio	1.42
Skew	-0.23
Kurtosis	2.74
Tail ratio	1.00
Daily value at risk	-1.8%

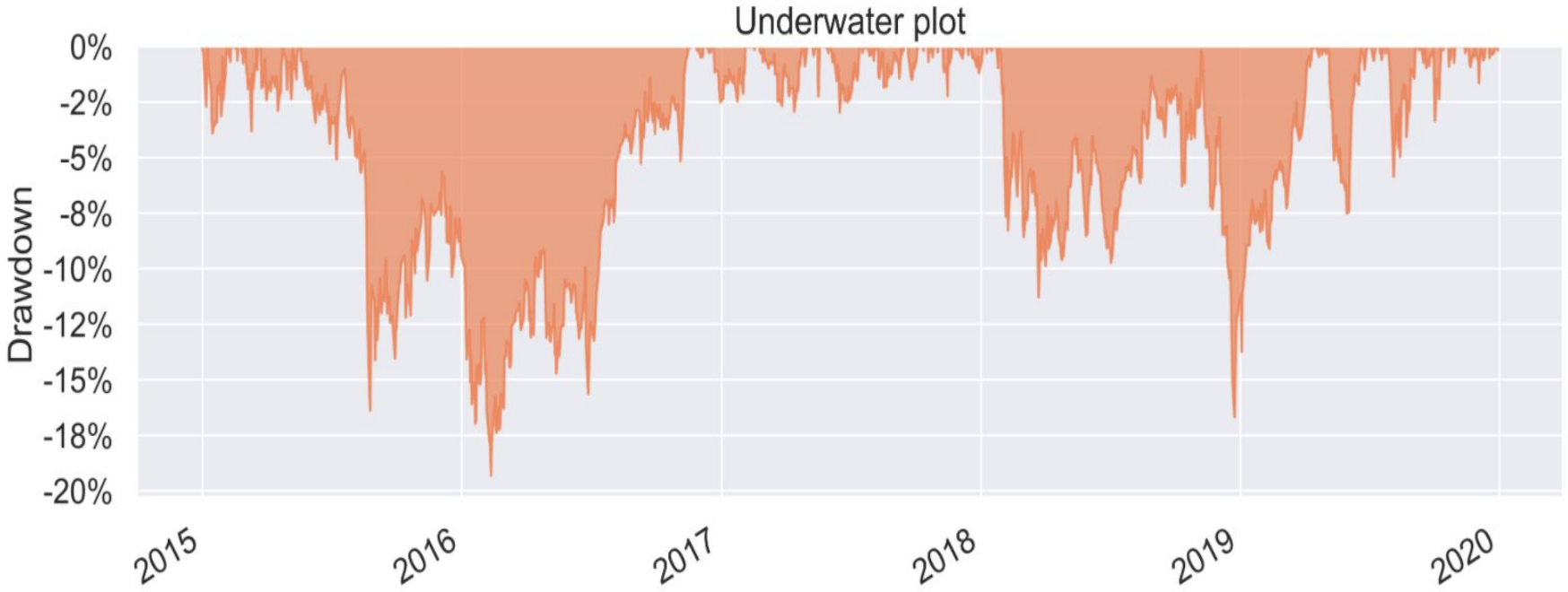
Cumulative returns



Sharpe ratio

The sharpe ratio fluctuate near 1

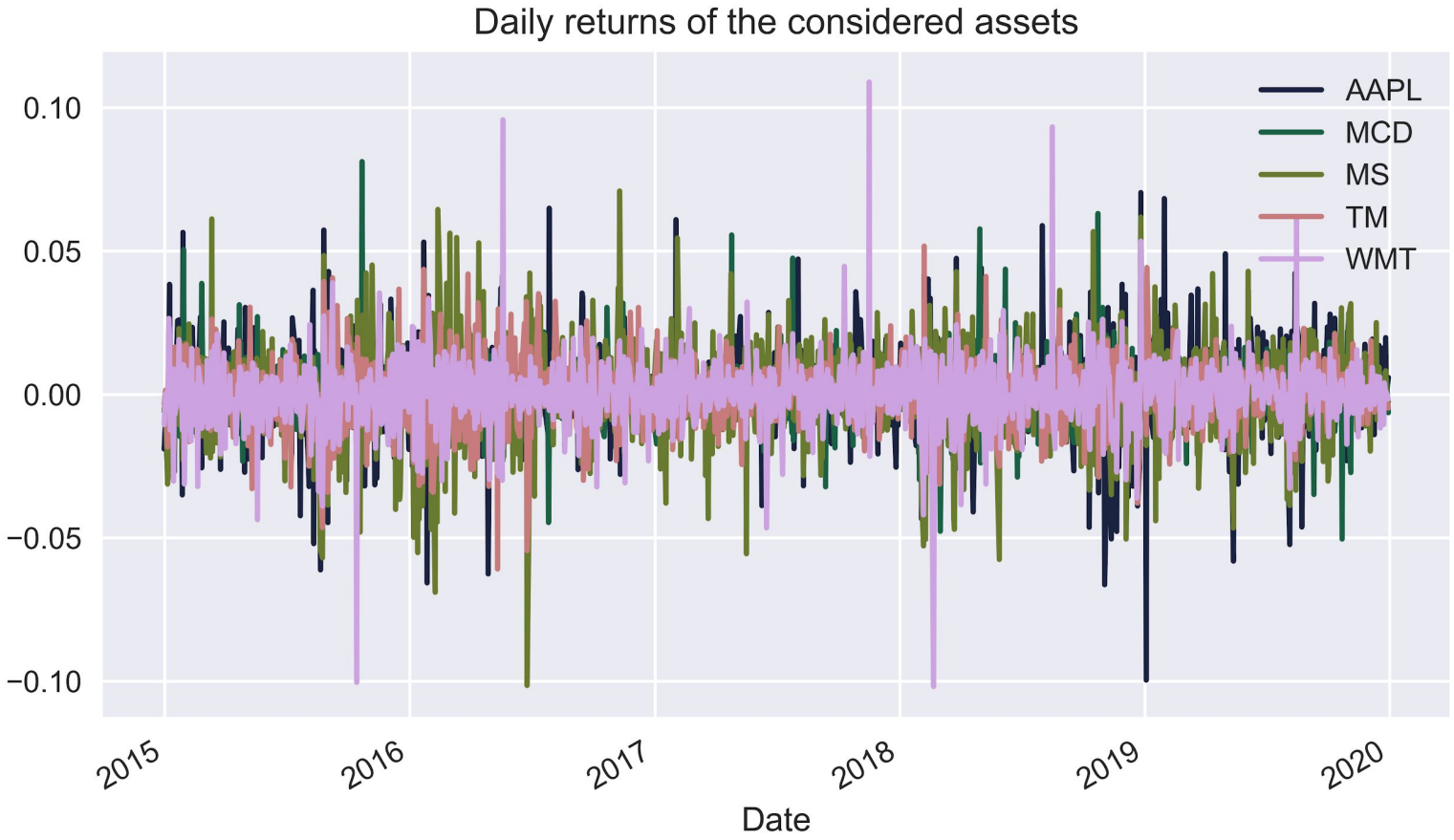
Underwater plot



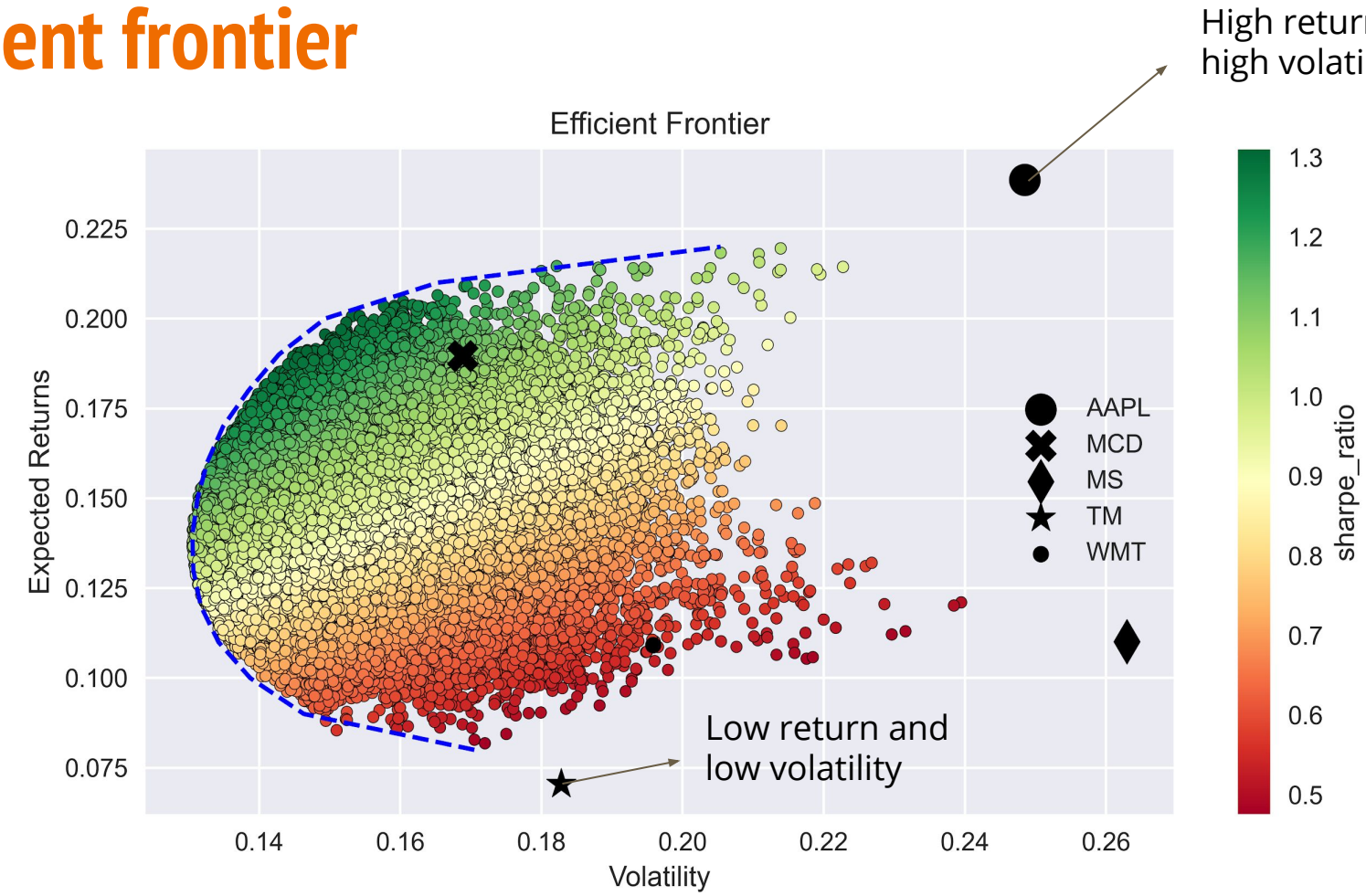
The drawdown reach -20% in 2016

2.Finding the Efficient Frontier using Monte Carlo simulations

Daily returns



Efficient frontier



Performers

In [38]:
```
print('Maximum Sharpe Ratio portfolio ----')
print('Performance')
for index, value in max_sharpe_portf.items():
    print(f'{index}: {100 * value:.2f}% ', end="", flush=True)
print('\nWeights')
for x, y in zip(RISKY_ASSETS, weights[np.argmax(portf_results_df.sharpe_ratio)]):
    print(f'{x}: {100*y:.2f}% ', end="", flush=True)
```

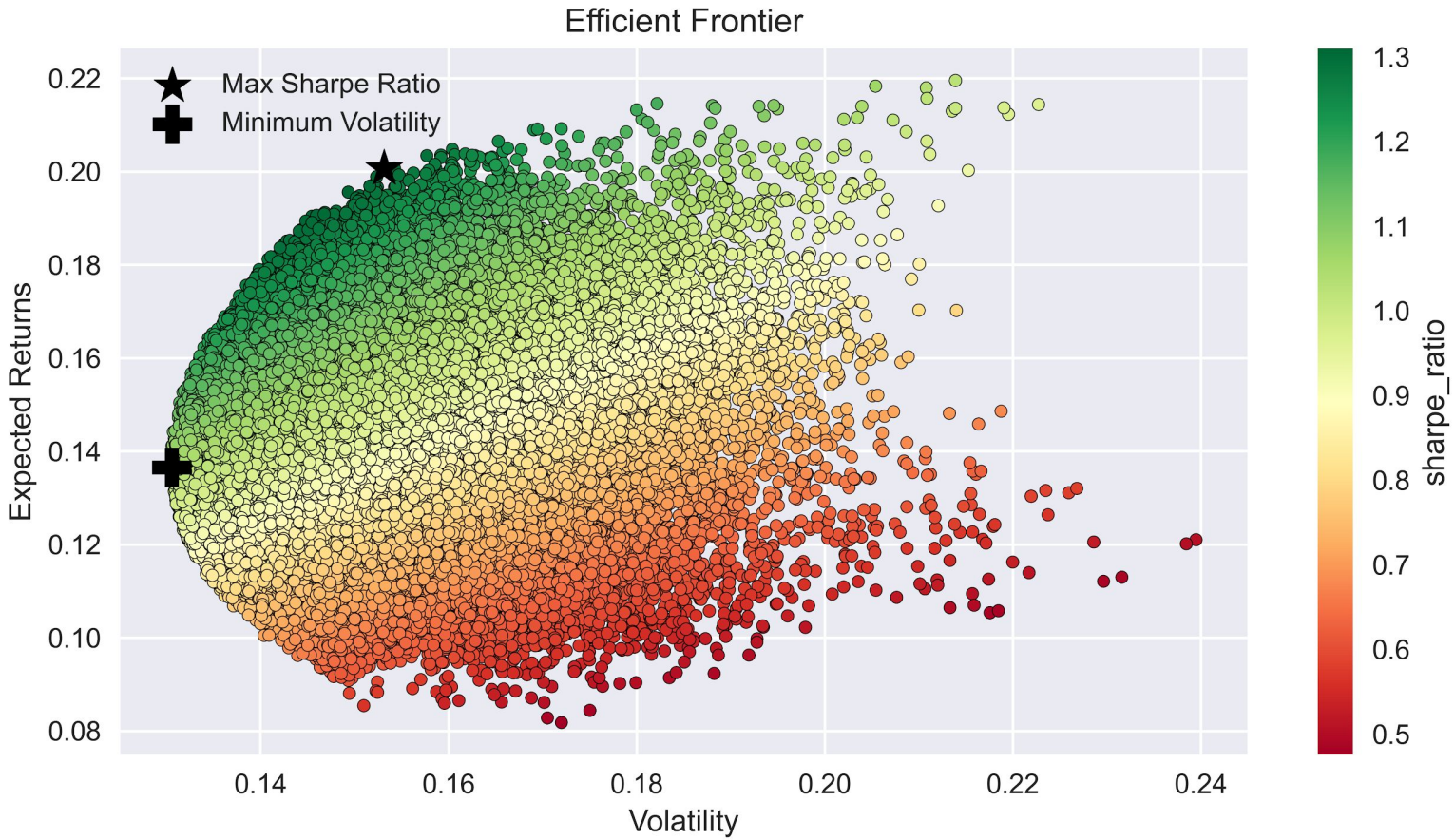
```
Maximum Sharpe Ratio portfolio ----
Performance
returns: 20.07% volatility: 15.31% sharpe_ratio: 131.03%
Weights
AAPL: 32.02% MCD: 62.81% MS: 1.39% TM: 1.27% WMT: 2.51%
```

The weights of AAPL, TM and WHT changed a lot.

In [39]:
```
print('Minimum Volatility portfolio ----')
print('Performance')
for index, value in min_vol_portf.items():
    print(f'{index}: {100 * value:.2f}% ', end="", flush=True)
print('\nWeights')
for x, y in zip(RISKY_ASSETS, weights[np.argmin(portf_results_df.volatility)]):
    print(f'{x}: {100*y:.2f}% ', end="", flush=True)
```

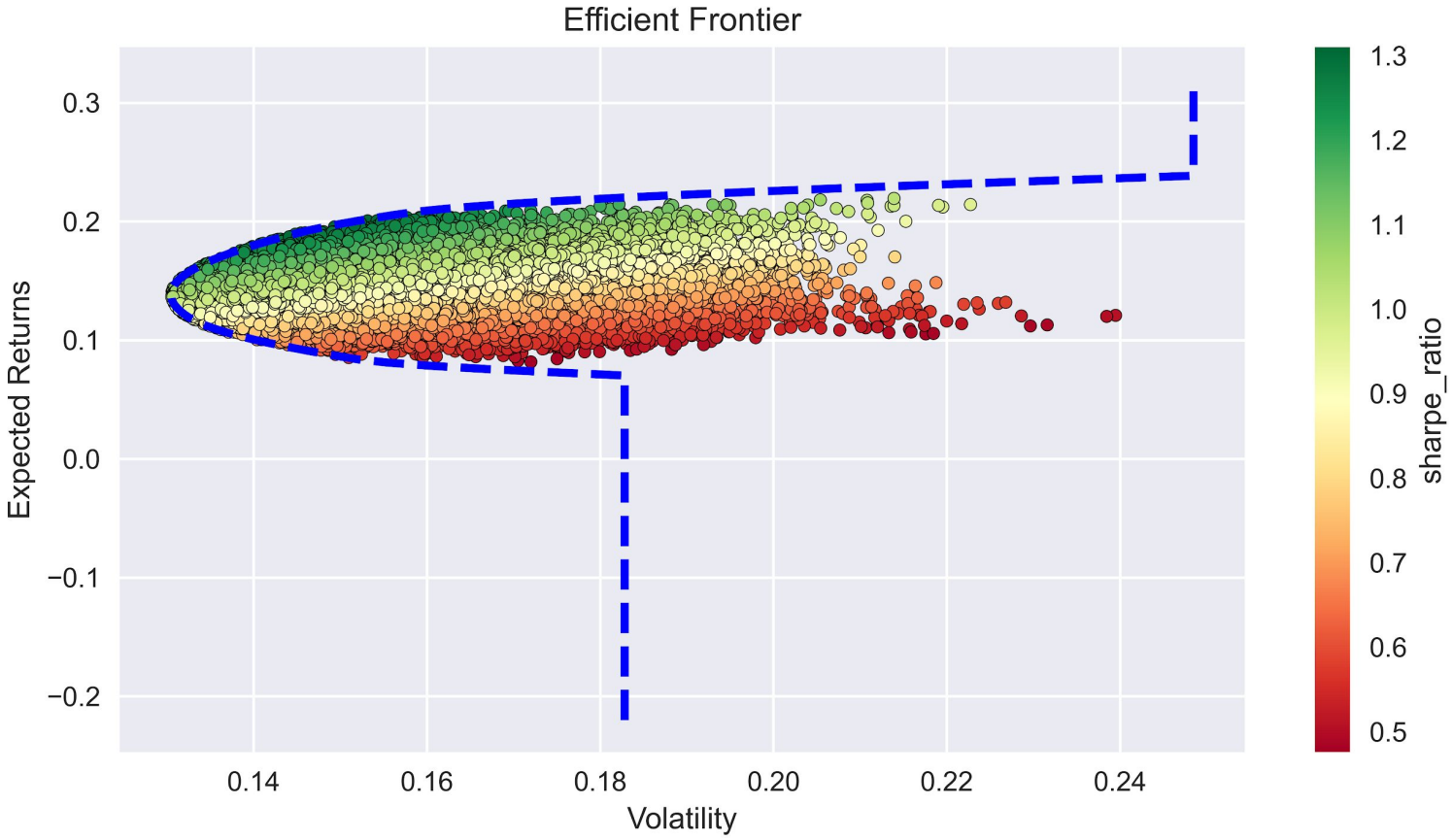
```
Minimum Volatility portfolio ----
Performance
returns: 13.66% volatility: 13.05% sharpe_ratio: 104.61%
Weights
AAPL: 7.15% MCD: 36.40% MS: 0.03% TM: 28.83% WMT: 27.58%
```

Efficient frontier



3.Finding the Efficient Frontier using optimization with scipy

Efficient frontier



Performance

Print performance summary:

In [53]:
```
print('Maximum Sharpe Ratio portfolio ----')
print('Performance')

for index, value in max_sharpe_portf.items():
    print(f'{index}: {100 * value:.2f}% ', end="", flush=True)

print('\nWeights')
for x, y in zip(RISKY_ASSETS, max_sharpe_portf_w):
    print(f'{x}: {100*y:.2f}% ', end="", flush=True)
```

```
Maximum Sharpe Ratio portfolio ----
Performance
Return: 19.91% Volatility: 15.08% Sharpe Ratio: 132.06%
Weights
AAPL: 31.65% MCD: 60.87% MS: 0.00% TM: 0.00% WMT: 7.48%
```

Performance

Print performance summary:

In [49]:
```
print('Minimum Volatility portfolio ----')
print('Performance')

for index, value in min_vol_portf.items():
    print(f'{index}: {100 * value:.2f}% ', end="", flush=True)

print('\nWeights')
for x, y in zip(RISKY_ASSETS, efficient_portfolios[min_vol_ind]['x']):
    print(f'{x}: {100*y:.2f}% ', end="", flush=True)
```

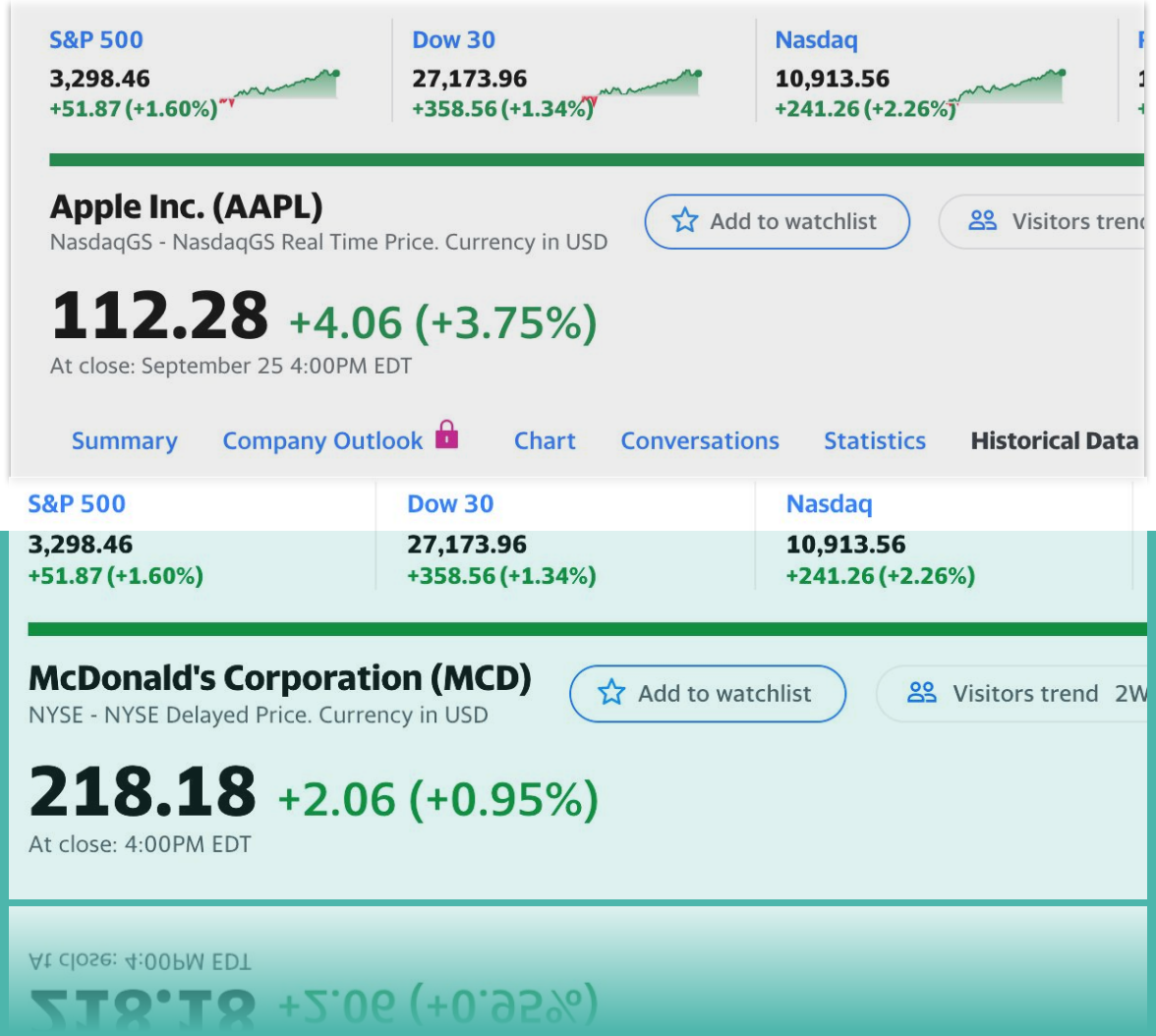
```
Minimum Volatility portfolio ----
Performance
Return: 13.82% Volatility: 13.04% Sharpe Ratio: 105.94%
Weights
AAPL: 7.23% MCD: 38.43% MS: 0.00% TM: 29.09% WMT: 25.25%
```

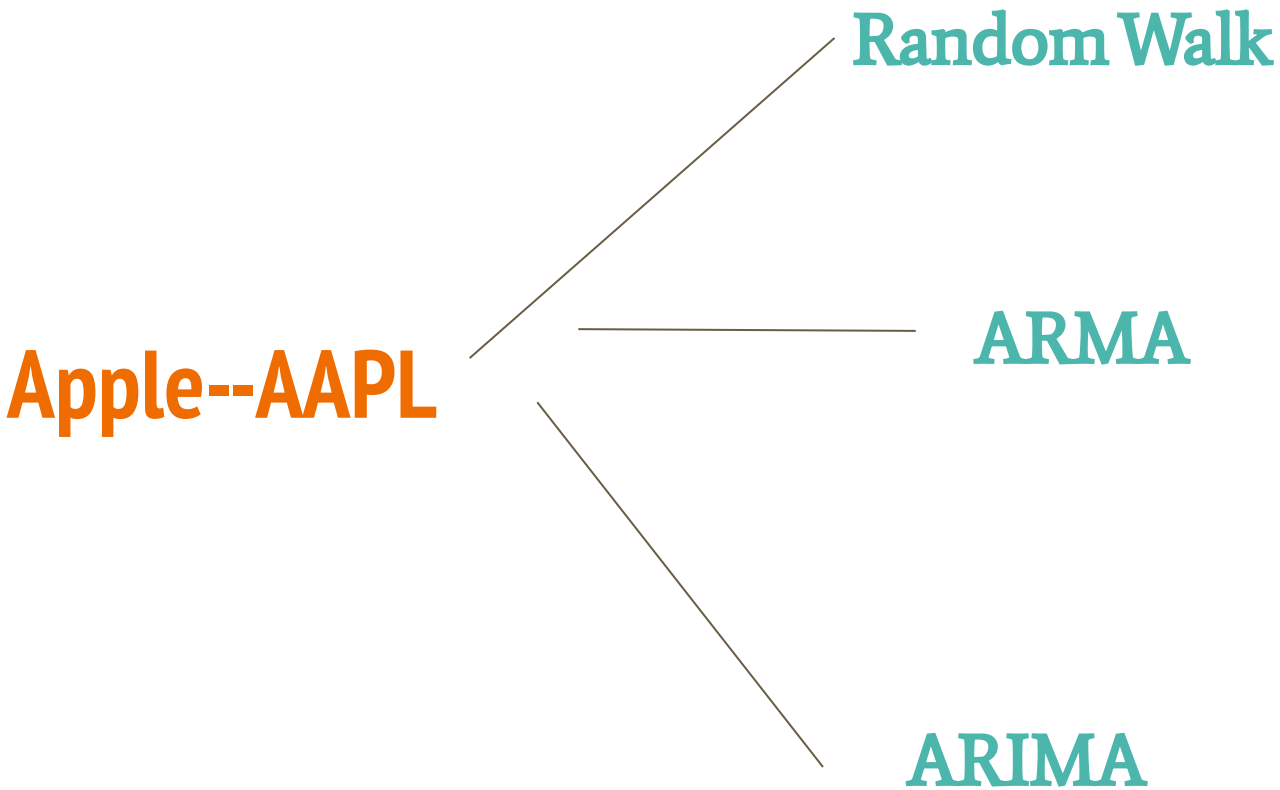
The results are similar as before

Part II. Model Fitting

APPLE

Mcdonald



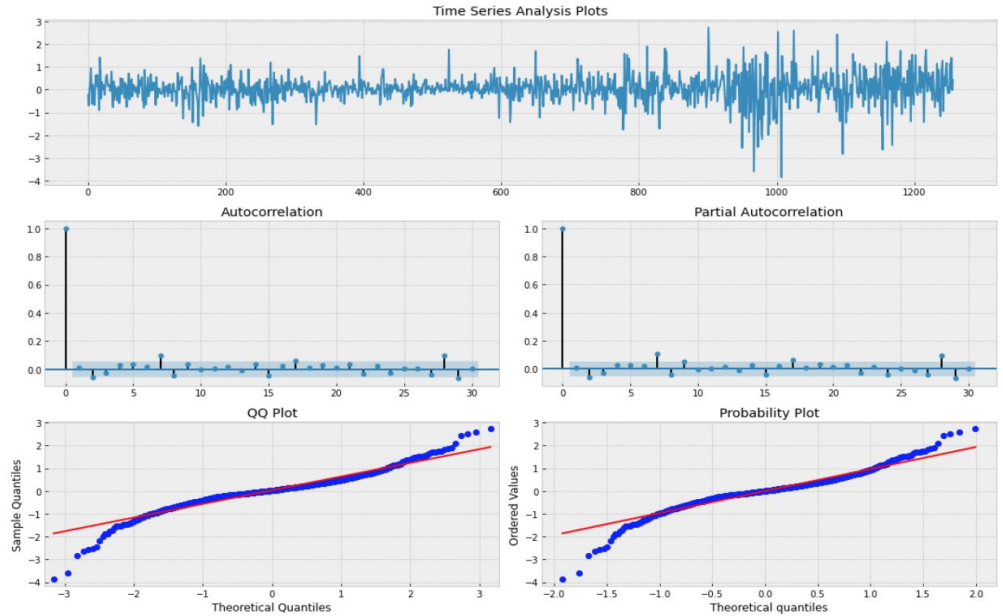

Random Walk
ARMA
Apple--AAPL
ARIMA

1. Random Walk for AAPL

The tails of the probability plot has some obvious difference between the observed value, it shows the Random walk is not well fit.

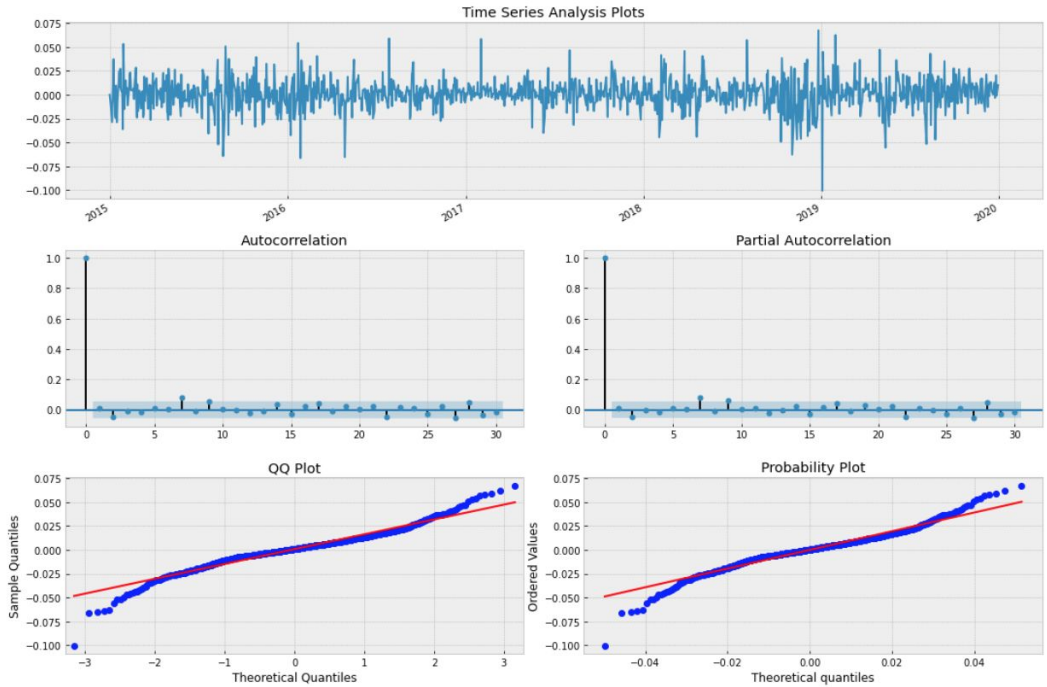
1. Difference of APPL prices--Random walk

```
_ = tsplot(np.diff(data), lags=30)
```



2. ARMA for AAPL

```
aic: -6886.67379 | order: (4, 4)
```



The best fitting model has ARMA(4, 4). Notice that there are some significant peak, especially at higher lags. This is indicative of a poor fit.

2. ARMA for AAPL

```
In [23]: sms.diagnostic.acorr_ljungbox(best_mdl.resid, lags=[20], boxpierce=False)
```

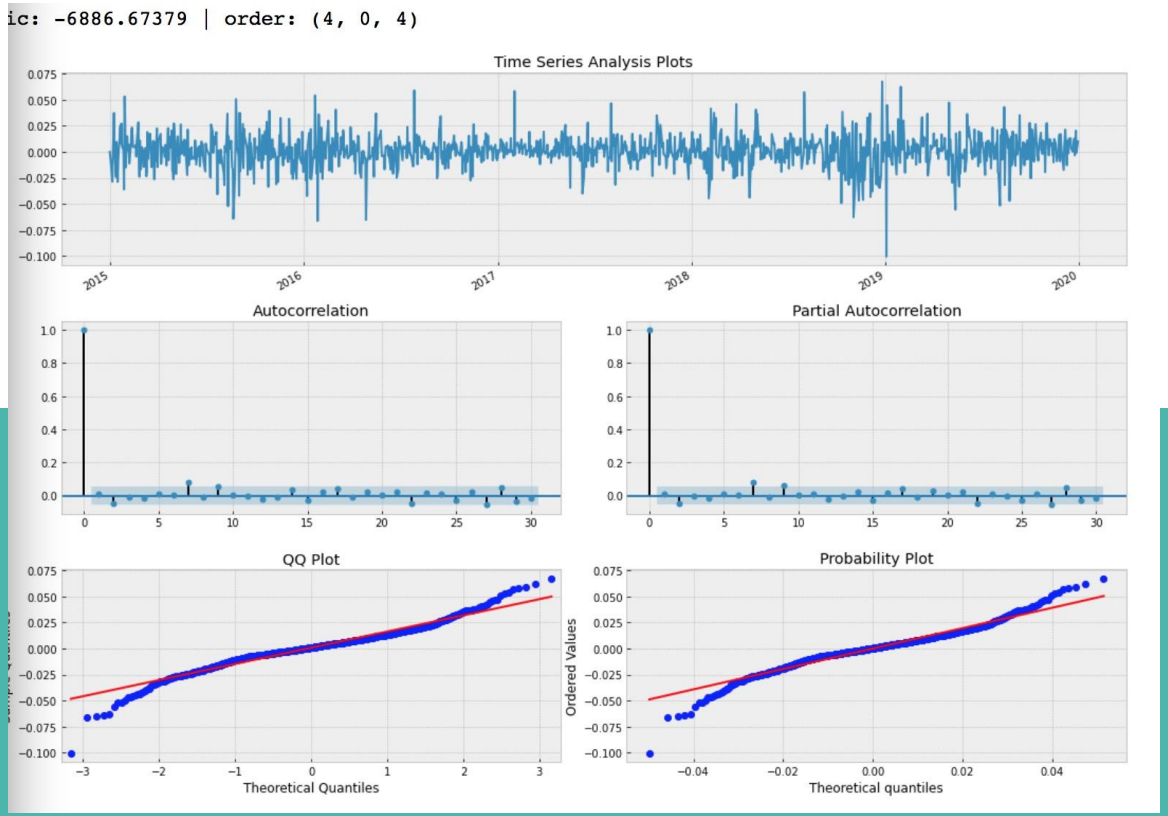
```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/stats/diagnostic.py:524: FutureWarning: The value returned will change to a single DataFrame after 0.12 is released.  Set return_df to True to use to return a DataFrame now.  Set return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```

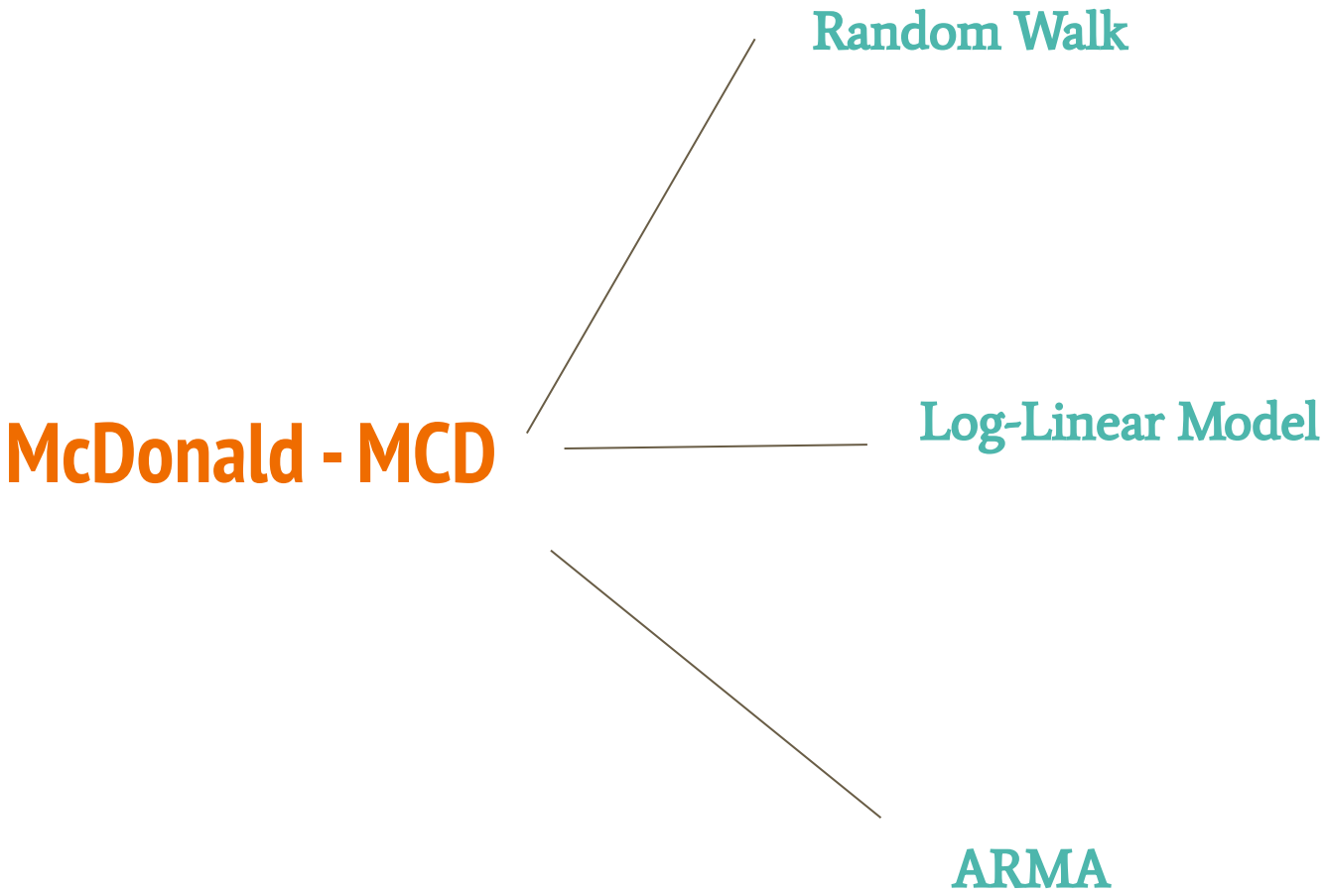
```
Out[23]: (array([22.19668582]), array([0.32993404]))
```

Let's perform a Ljung-Box test to see if we have statistical evidence for this: As we suspected, the p-value is less than 0.05 and as such we cannot say that the residuals are a realisation of discrete white noise. Hence there is additional autocorrelation in the residuals that is not explained by the fitted ARMA(4,4) model.

3. ARIMA for AAPL

Recall that we already took the first difference of log prices to calculate the stock returns. The result is essentially identical to the ARMA(4, 4) model that we fit above. Clearly this ARIMA model has not explained the conditional volatility in the series either!

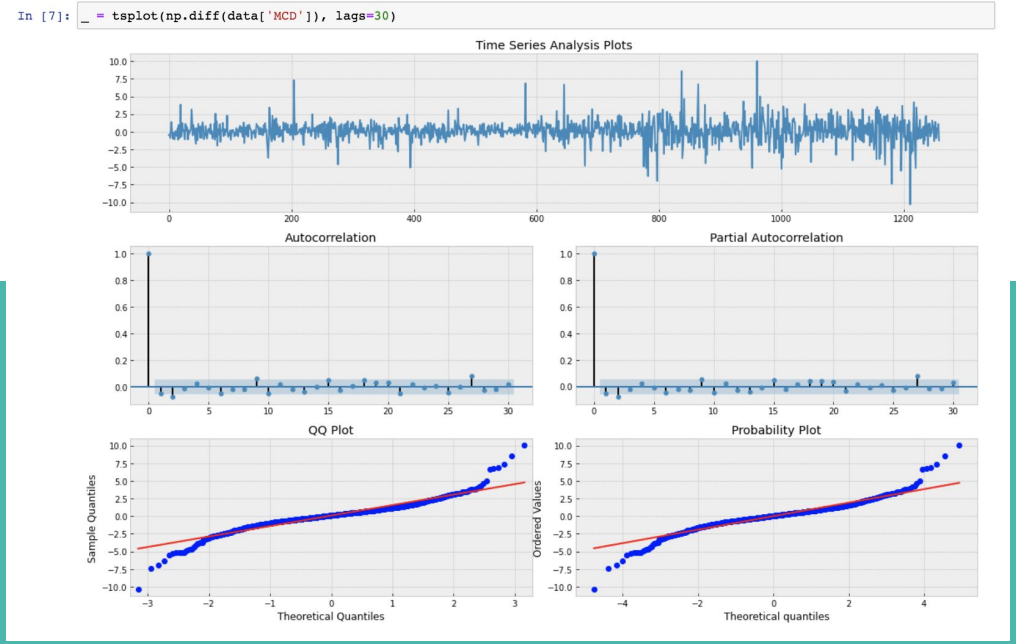



McDonald - MCD
Random Walk
Log-Linear Model
ARMA

McDonald | Random Walk

- Similar to white noise.
- Some significant serial correlation in the ACF and PACF plots
- QQ and Probability plots close to normality, but with heavy tails.

Conclusion: We have better models to explain the price of this asset.

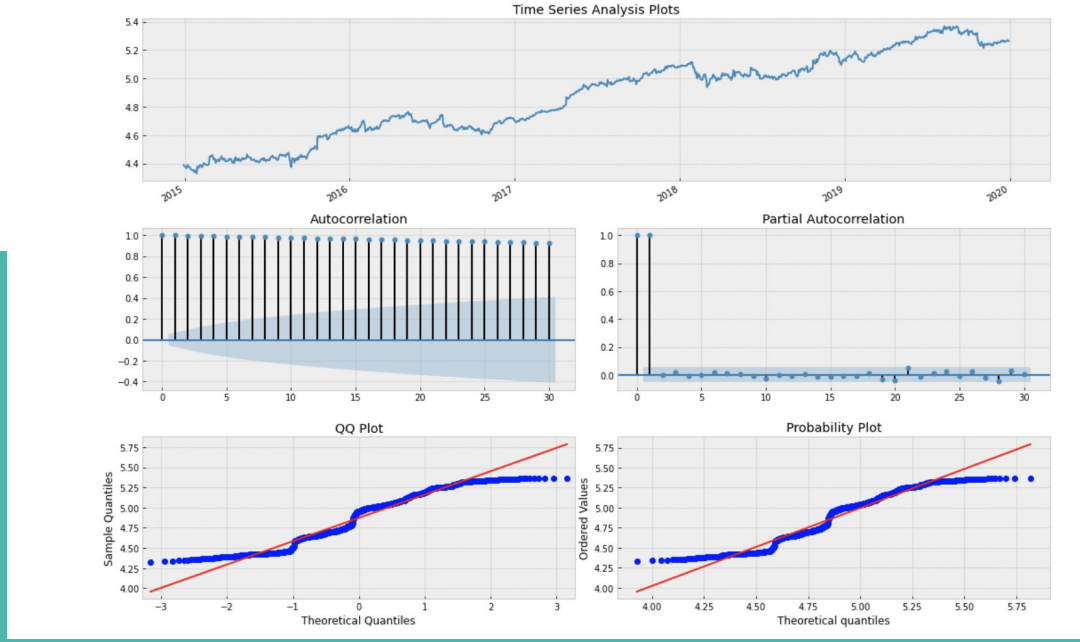


McDonald | Log-Linear Model

- Taking a log, doe not fit into a linear trend

Conclusion: log-linear model is not a good choice.

```
In [25]: _ = tsplot(np.log(data['MCD']), lags=30)
```



McDonald | ARMA

ARMA has 2 parameters: p and q

Finding the value of p and q by the Akaike Information Criterion (AIC): **p = 3, q = 2**

In [45]:
```
# pick best order by aic
# smallest aic value wins
best_aic = np.inf
best_order = None
best_mdl = None

rng = range(5)
for i in rng:
    for j in rng:
        try:
            tmp_mdl = smt.ARMA(arma32, order=(i, j)).fit(method='mle', trend='nc')
            tmp_aic = tmp_mdl.aic
            if tmp_aic < best_aic:
                best_aic = tmp_aic
                best_order = (i, j)
                best_mdl = tmp_mdl
        except: continue


print('aic: %6.5f | order: %s'%(best_aic, best_order))
```

```
aic: 14183.20899 | order: (3, 2)
```

McDonald | ARMA (cont'd)

```
In [48]: sms.diagnostic.acorr_ljungbox(best_mdl.resid, lags=[20], boxpierce=False)
Out[48]: (array([14.15889539]), array([0.82234901]))
```

Because the p-value is less than 0.05, we cannot say that the residuals are a realization of a discrete white noise.

Conclusion: There is additional autocorrelation in the residuals that is not discovered by fitting ARMA (3,2).

Part III. Prediction of Future Returns

Prediction for AAPL

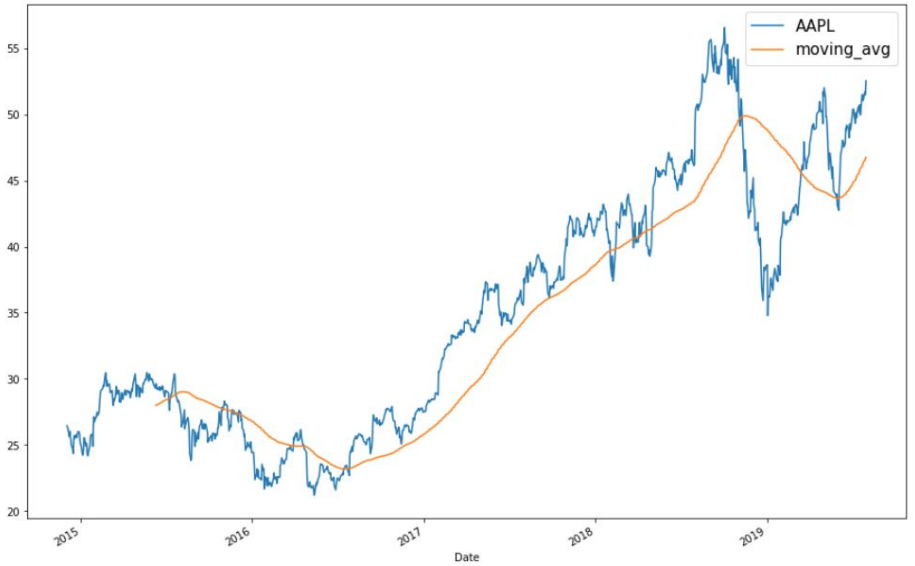
The moving average is basically looks like the trend of the original dataset but not so accurate.

Moving Average

```
close = df['Adj Close']
moving_avg = close.rolling(130).mean()
```

```
plt.figure(figsize=(15,10))
close.plot(label='AAPL')
moving_avg.plot(label='moving_avg')
plt.legend(fontsize=15)
```

```
<matplotlib.legend.Legend at 0x1222a33d0>
```



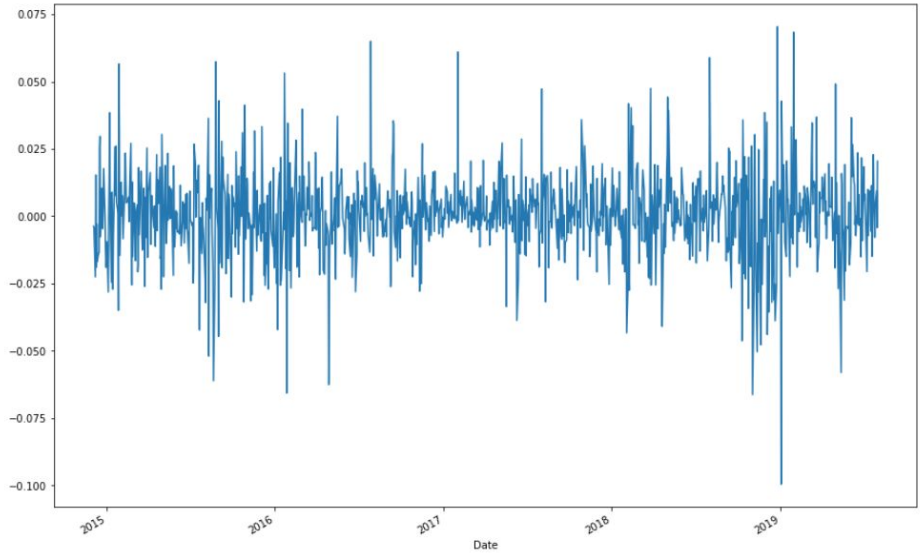
Prediction for AAPL

This is a return plot, could see the variances is not stable. We could not predict the future returns from the returns trend. It irregularly varies from day to day. So we are supposed to figure out more models to predict the future returns.

Return

In [7]:
```
rets = close / close.shift(1) - 1
rets.plot(label='return', figsize=(15,10))
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x12328fe80>



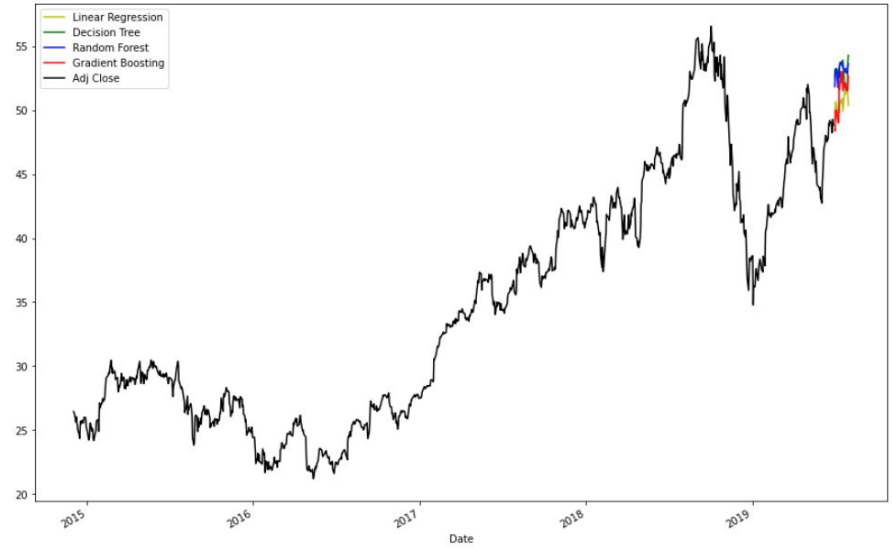
Prediction for AAPL

Prediction

```
lr_pred = pd.Series(lr.predict(X_OOS), index=df.index[-forecast_out:])
dt_pred = pd.Series(dt.predict(X_OOS), index=df.index[-forecast_out:])
rf_pred = pd.Series(rf.predict(X_OOS), index=df.index[-forecast_out:])
gb_pred = pd.Series(gb.predict(X_OOS), index=df.index[-forecast_out:])
```

```
plt.figure(figsize=(15,10))
lr_pred.plot(label='Linear Regression', c='y')
dt_pred.plot(label='Decision Tree', c='g')
rf_pred.plot(label='Random Forest', c='b')
gb_pred.plot(label='Gradient Boosting', c='r')
df['Adj Close'][:-forecast_out].plot(c='k')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x126722970>
```

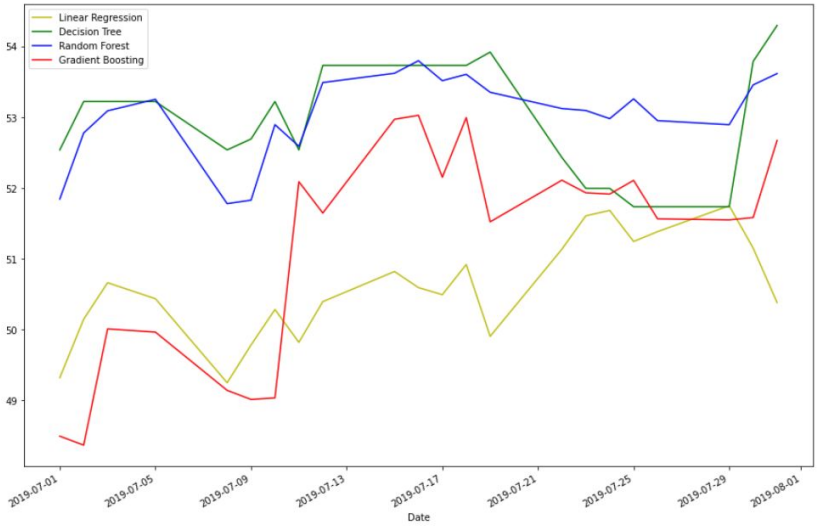


Prediction for AAPL

There are relative differences between those predictive values. So might be the next step we should compare with the original returns to see how good the models are so that we could choose the better one the predict the future returns.

In [18]:
```
plt.figure(figsize=(15,10))
lr_pred.plot(label='Linear Regression', c='y')
dt_pred.plot(label='Decision Tree', c='g')
rf_pred.plot(label='Random Forest', c='b')
gb_pred.plot(label='Gradient Boosting', c='r')
plt.legend()
```

Out[18]: <matplotlib.legend.Legend at 0x1231d3b50>



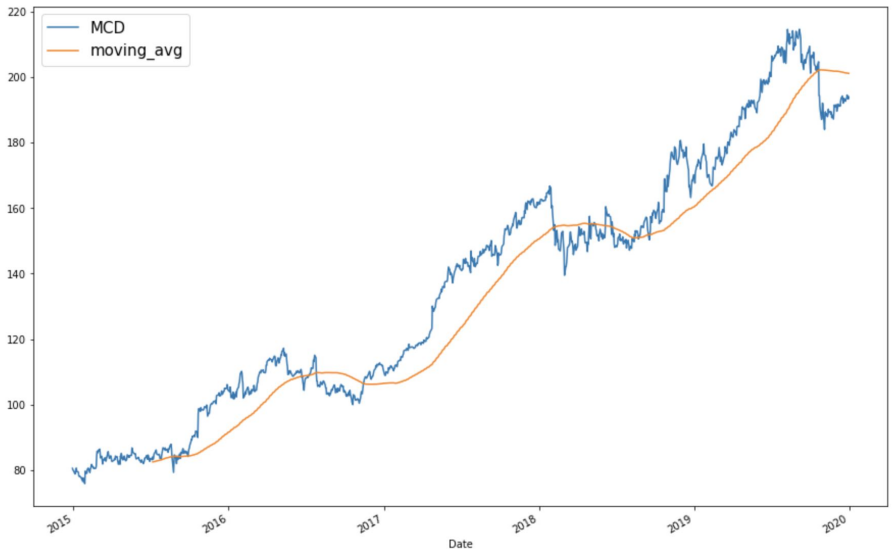
McDonald | Prediction of Future Returns

Moving Average plot: a direct visualization to see the trend of the asset prices.

Conclusion: The price will go upward in the long run, but during some times, it might encounter a small decrease in the prices.

```
In [4]: plt.figure(figsize=(15,10))
        close.plot(label='MCD')
        moving_avg.plot(label='moving_avg')
        plt.legend(fontsize=15)

Out[4]: <matplotlib.legend.Legend at 0x1227ba1f0>
```

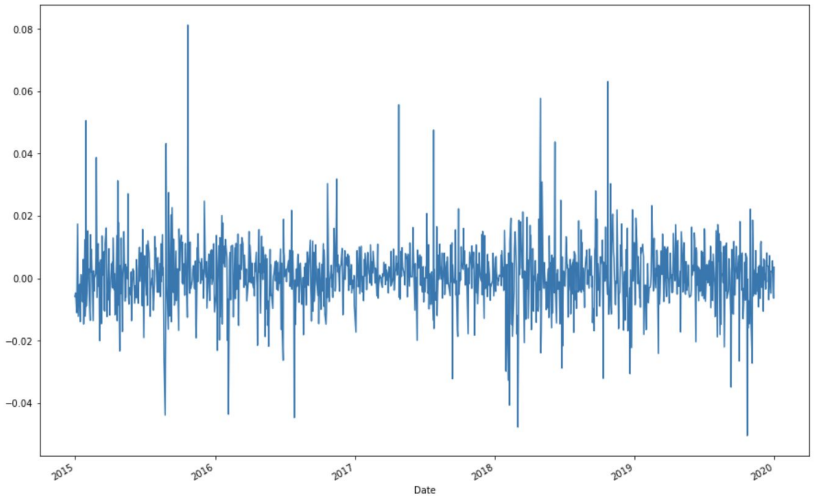


McDonald | Prediction of Future Returns (cont'd)

Return plot: Variance is unstable so the plot changes quickly and erratically.

```
In [5]: rets = close / close.shift(1) - 1
        rets.plot(label='return', figsize=(15,10))
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1192619a0>
```

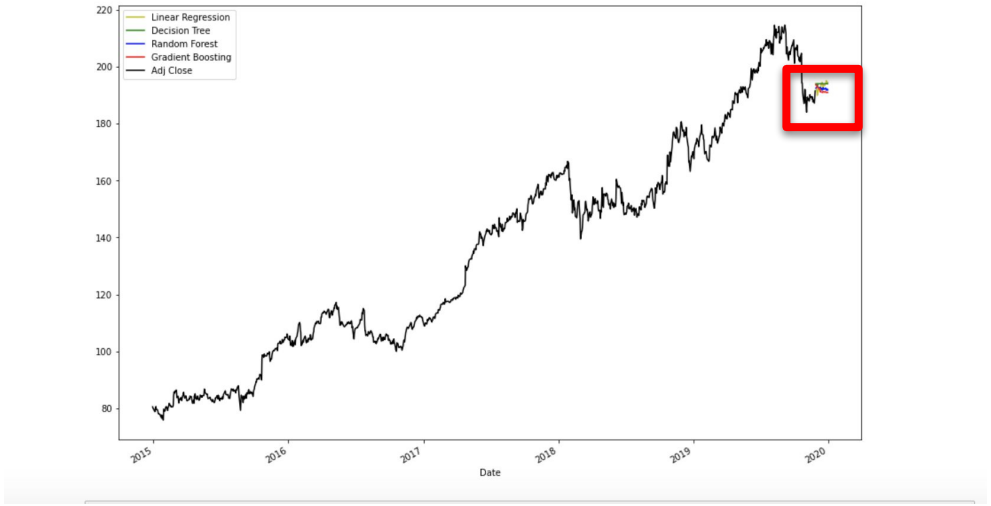


McDonald | Prediction of Future Returns (cont'd)

- Using sklearn for prediction
- Exact asset price from 2015-2020
- Predicative trends from four models at the tail of the curve

```
In [15]: plt.figure(figsize=(15,10))
         lr_pred.plot(label='Linear Regression', c='y')
         dt_pred.plot(label='Decision Tree', c='g')
         rf_pred.plot(label='Random Forest', c='b')
         gb_pred.plot(label='Gradient Boosting', c='r')
         df['Adj Close'][:-forecast_out].plot(c='k')
         plt.legend()

Out[15]: <matplotlib.legend.Legend at 0x11c8fd580>
```

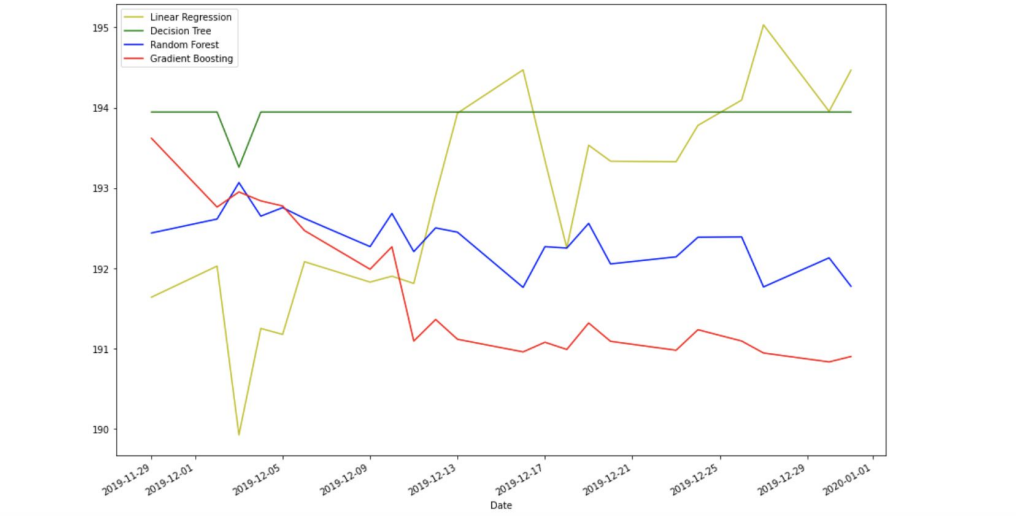


McDonald | Prediction of Future Returns (cont'd)

Comparing the exact value and predicted value to choose the best prediction.

```
In [16]: plt.figure(figsize=(15,10))
         lr_pred.plot(label='Linear Regression', c='y')
         dt_pred.plot(label='Decision Tree', c='g')
         rf_pred.plot(label='Random Forest', c='b')
         gb_pred.plot(label='Gradient Boosting', c='r')
         plt.legend()

Out[16]: <matplotlib.legend.Legend at 0x11c95ebb0>
```



Thanks for your watching and listening!