Asset Pricing - Homework 1 - José Barretto and Daniel Deutsch

September 19, 2021

1 EA1:

By Daniel Deutsch and José Lucas Barretto

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.tsa.api as tsa
from scipy.stats import norm
```

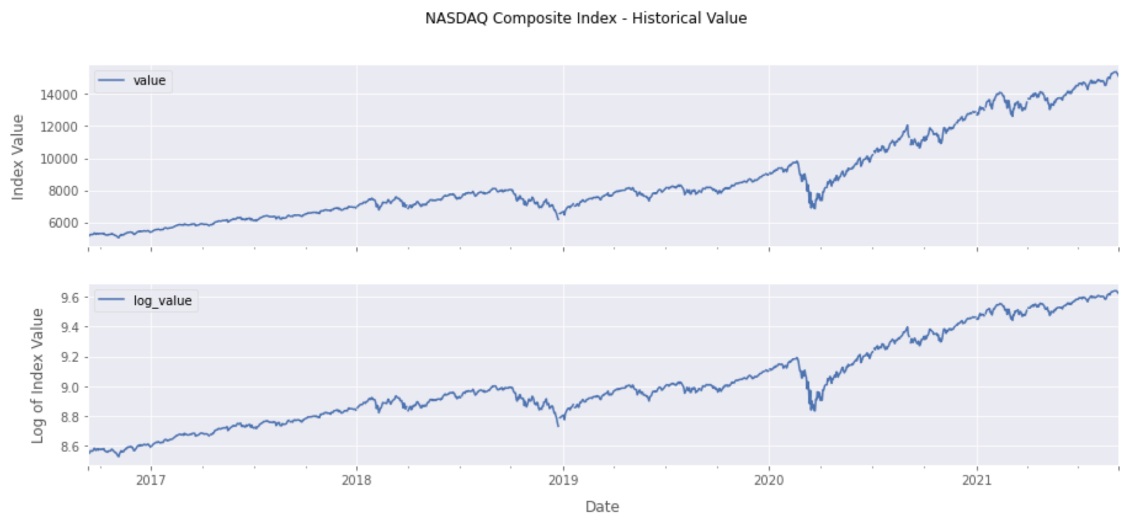
[25]:
```
# Matplotlib styles
plt.style.use('ggplot')
plt.rcParams.update({
    'figure.figsize': (15, 4),
    'axes.prop_cycle': plt.cycler(color=["#4C72B0", "#C44E52", "#55A868",␣
 ↪"#8172B2", "#CCB974", "#64B5CD"]),
    'axes.facecolor': "#EAEAF2",
    'axes.labelpad': 10,
    'axes.titlepad': 10
})
```

1.1 Load and process the dataset

We chose to work with NASDAQ's Composite Index series for the homework. Let's start by loading, processing, and visualizing the index's values and log-values.

[3]:
```
# load and process nasdaq index dataset
df = pd.read_csv("datasets/NASDAQ_Index.csv", parse_dates=[0], dtype={1: np.
 ↪float64}, na_values='.')
df.rename(columns={'DATE': 'date', 'NASDAQCOM': 'value'}, inplace=True)

# apply log to values
df['log_value'] = np.log(df['value'])
```

[4]:
```
# plot series' values and log-values
fig, ax = plt.subplots(2, 1, sharex=True, figsize=(15,6))

df.plot(x='date', y='value', ax=ax[0])
ax[0].set_ylabel("Index Value")

df.plot(x='date', y='log_value', ax=ax[1])
ax[1].set_xlabel("Date")
ax[1].set_ylabel("Log of Index Value")

fig.suptitle("NASDAQ Composite Index - Historical Value")
plt.show()
```



We can see that the data has a daily frequency, and ranges from september 2016 to september 2021. We will now divide the series into two different periods. The first period ranges from september 2016 up to the start of 2019, where the series shows a relatively controlled growing behavior. The second period, which ranges from the start of 2019 until september 2021, on the other hand, contains extremely volatile periods. This will allow us to study the unpredictable behavior of stock markets, by analysing two very different behaviors of the same asset.

[5]:
```
mask = df['date'] < '2019-01-01'
series1 = df[mask].dropna().reset_index(drop=True)
series2 = df[~mask].dropna().reset_index(drop=True)
```

1.2 Examine whether the series are martingales or not.

We will test the hypothesis:

$$H_0 : \text{The series is a martingale process.}$$

If the hypothesis is true, then:

$$\mathbf{E}[X_{t+1}|X_t, X_{t-1}, ..., X_0] = X_t \Leftrightarrow X_{t+1} = X_t + \epsilon_{t+1}$$

Where

$$\mathbf{E}[\epsilon_{t+1}|X_t, X_{t-1}, ..., X_0] = 0$$

We can rewrite the martingale hypothesis as:

$$X_{t+1} - X_t = \epsilon_{t+1}$$

Which allows us to test the RW3 hypothesis for the series of log prices:

$$H_0 : X_{t+1} - X_t = \mu + \epsilon_{t+1}$$

For that, we can follow the procedure described in Campbell, Lo and MacKinlay (1997, p.33).

[6]:
```
def test_RW3(series, q):

    ac = tsa.acf(series, nlags=len(series), fft=True, adjusted=True)
    mean = np.mean(series)

    vr = 1
    theta = 0
    for k in range(1, q):
        vr += 2*(1-k/q)*ac[k]

        num = 0
        for j in range(k+1, len(series)):
            num += (len(series)-1) * (series[j] - series[j-1] - mean)**2 *
 ↪(series[j-k] - series[j-k-1] - mean)**2

        den = 0
        for j in range(1, len(series)):
            den += (series[j] - series[j-1] - mean)**2

        delta_k = num/(den**2)
        theta += 4*((1-k/q)**2)*delta_k

    phi = np.sqrt(len(series)-1)*(vr - 1)/np.sqrt(theta)
    p_value = 2*(1 - norm().cdf(abs(phi)))

    return vr, phi, p_value
```

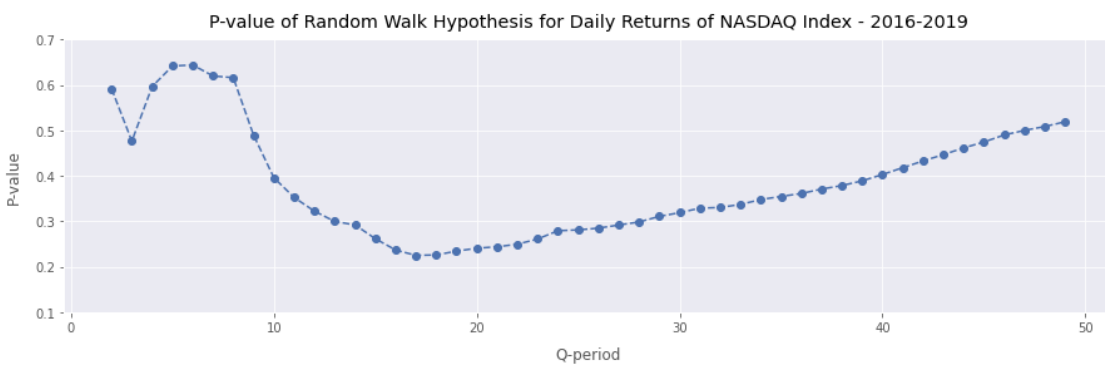
Let's run the test for multiple values of q on the log differences series, and see if the random walk hypothesis holds.

1.2.1 1st Period: September 2016 until the start of 2019

[7]:
```
# get log differences of series
log_diff1 = series1['log_value'].diff().dropna().reset_index(drop=True)
```

[8]:
```
test = {
    'q': np.arange(2,50,1),
    'VR': [],
    'T-statistic': [],
    'P-value': [],
}

for q in test['q']:
    vr, phi, p_value = test_RW3(log_diff1, q)
    test['VR'].append(vr)
    test['T-statistic'].append(phi)
    test['P-value'].append(p_value)
```

[9]:
```
plt.figure()
plt.plot(test['q'], test['P-value'], '--o')
plt.title('P-value of Random Walk Hypothesis for Daily Returns of NASDAQ Index␣
 ↪- 2016-2019')
plt.xlabel('Q-period')
plt.ylabel('P-value')
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7])
plt.show()
```



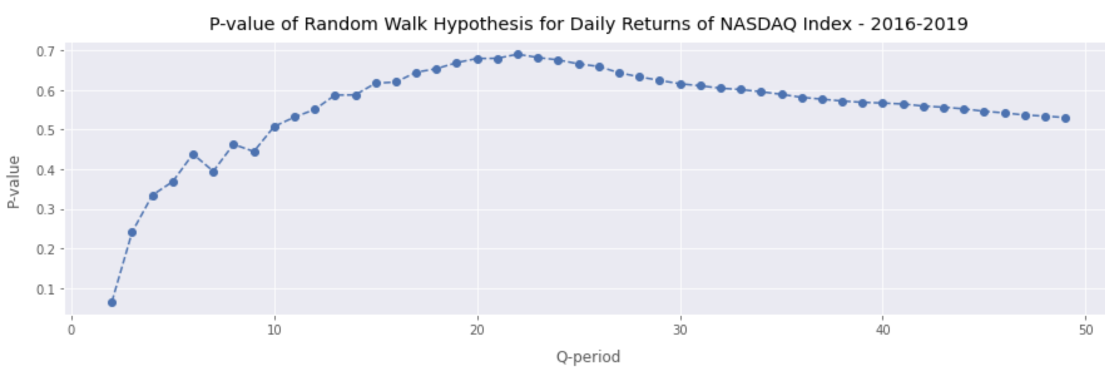
We can see that, for the analyzed period of log-values of the NASDAQ Index, the RW3 hypothesis cannot be rejected. This indicates that during this period, the market was quite competitive and efficient.

1.2.2 2nd Period: start of 2019 until September 2021

[10]:
```
# get log differences of series
log_diff2 = series2['log_value'].diff().dropna().reset_index(drop=True)
```

[11]:
```
test = {
    'q': np.arange(2,50,1),
    'VR': [],
    'T-statistic': [],
    'P-value': [],
}

for q in test['q']:
    vr, phi, p_value = test_RW3(log_diff2, q)
    test['VR'].append(vr)
    test['T-statistic'].append(phi)
    test['P-value'].append(p_value)
```

[12]:
```
plt.figure()
plt.plot(test['q'], test['P-value'], '--o')
plt.title('P-value of Random Walk Hypothesis for Daily Returns of NASDAQ Index␣
 ↪- 2016-2019')
plt.xlabel('Q-period')
plt.ylabel('P-value')
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7])
plt.show()
```



When analyzing this second period, we see that the Random Walk hypothesis can be rejected at the 10% confidence level when using q=2. This might be due to the fact that this period of the NASDAQ Index showed large variations and may have risen artificially, which negates the efficient markets hypothesis.

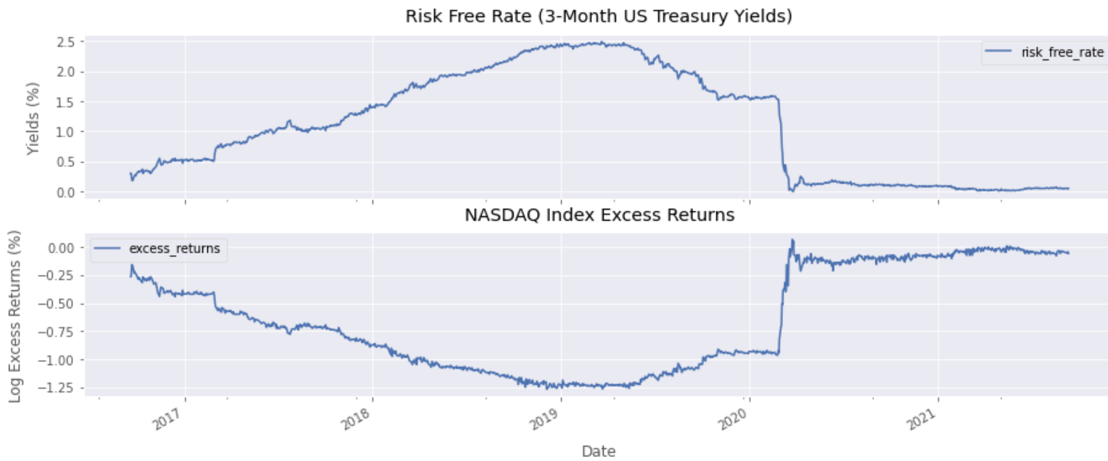
1.3 Repeating the analysis for excess returns.

1.3.1 Load and process the risk-free rate dataset

[13]:
```
# load risk free rate dataset
rf_rate = pd.read_csv("datasets/3_month_govbond_yields.csv", parse_dates=[0],␣
↪dtype={1: np.float64}, na_values='.')
rf_rate.rename(columns={'DATE': 'date', 'DGS3MO': 'risk_free_rate'},␣
↪inplace=True)
rf_rate.set_index('date', inplace=True)
```

[14]:
```
# merge datasets
df_returns = df.copy()
df_returns['returns'] = df['log_value'].diff()
df_returns = df_returns.set_index('date').merge(rf_rate, on='date',␣
↪how='inner').dropna()

# calculate excess returns
df_returns['excess_returns'] = df_returns['returns'] - np.
↪log(1+df_returns['risk_free_rate'])

# calculates first lag of excess returns
df_returns['excess_returns_lag1'] = df_returns['excess_returns'].shift()
df_returns.dropna(inplace=True)
```

[15]:
```
# plot series' values and log-values
fig, ax = plt.subplots(2, 1, sharex=True, figsize=(15,6))

df_returns.plot(y='risk_free_rate', ax=ax[0])
ax[0].set_ylabel("Yields (%)")
ax[0].set_title("Risk Free Rate (3-Month US Treasury Yields)")

df_returns.plot(y='excess_returns', ax=ax[1])
ax[1].set_xlabel("Date")
ax[1].set_ylabel("Log Excess Returns (%)")
ax[1].set_title("NASDAQ Index Excess Returns")

plt.show()
```

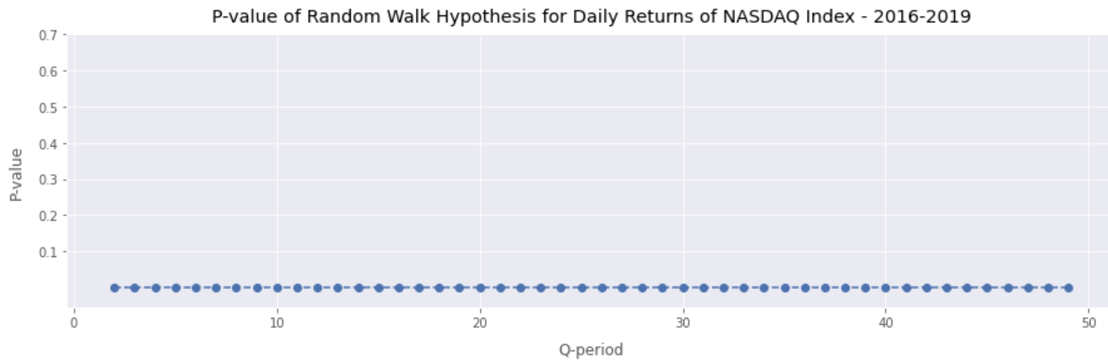


1.3.2 Analysing Period 1: 2016-2019

[16]:
```
mask = df_returns.index < '2019-01-01'
```

[17]:
```
test = {
    'q': np.arange(2,50,1),
    'VR': [],
    'T-statistic': [],
    'P-value': [],
}

for q in test['q']:
    vr, phi, p_value = test_RW3(df_returns.loc[mask, 'excess_returns'], q)
    test['VR'].append(vr)
    test['T-statistic'].append(phi)
    test['P-value'].append(p_value)
```

[18]:
```
plt.figure()
plt.plot(test['q'], test['P-value'], '--o')
plt.title('P-value of Random Walk Hypothesis for Daily Returns of NASDAQ Index␣
 ↪- 2016-2019')
plt.xlabel('Q-period')
plt.ylabel('P-value')
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7])
plt.show()
```



When analyzing the excess returns of the NASDAQ index, the Random Walk (RW3) hypothesis is rejected for all Q-periods tested. This means that this series does not behave as a martingale. Now, let's test the Strong Random Walk (RW1) hypothesis

[19]:
```
X = sm.add_constant(df_returns.loc[mask, 'excess_returns_lag1'])
model = sm.OLS(df_returns.loc[mask, 'excess_returns'], X, missing='drop')
res = model.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         excess_returns   R-squared:                       0.996
Model:                            OLS   Adj. R-squared:                  0.996
Method:                 Least Squares   F-statistic:                 1.511e+05
Date:                Sun, 19 Sep 2021   Prob (F-statistic):               0.00
Time:                        20:24:02   Log-Likelihood:                 1447.1
No. Observations:                 549   AIC:                            -2890.
Df Residuals:                     547   BIC:                            -2882.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
const                  -0.0042      0.002     -1.900      0.058      -0.008
0.000
excess_returns_lag1     0.9970      0.003    388.678      0.000       0.992
1.002
==============================================================================
Omnibus:                       31.586   Durbin-Watson:                   2.513
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               86.182
Skew:                           0.220   Prob(JB):                     1.93e-19
```

```
Kurtosis:                       4.890   Cond. No.                         5.80
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/home/josebarretto/anaconda3/envs/ftd/lib/python3.7/site-
packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of
pandas all arguments of concat except for the argument 'objs' will be keyword-
only
  x = pd.concat(x[::order], 1)
```

[20]:
```python
mu = res.params['const']
errors = res.resid
std = np.std(errors)
pi = norm().cdf(mu/std)
cj = (pi**2 + (1-pi)**2)/((2*pi)*(1-pi))
print("Estimated CJ coefficient:", cj)
```

```
Estimated CJ coefficient: 1.0744350915785825
```

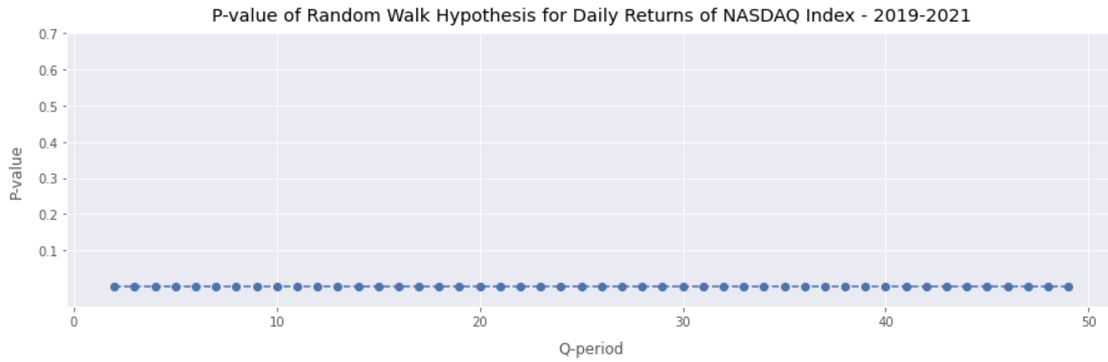
We can see that the CJ coefficient is approximately 1, which means that the RW1 hypothesis holds for this period.

1.3.3 Analysing Period 2: 2019-2021

[21]:
```python
test = {
    'q': np.arange(2,50,1),
    'VR': [],
    'T-statistic': [],
    'P-value': [],
}

for q in test['q']:
    vr, phi, p_value = test_RW3(df_returns.loc[~mask, 'excess_returns'], q)
    test['VR'].append(vr)
    test['T-statistic'].append(phi)
    test['P-value'].append(p_value)
```

[26]:
```python
plt.figure()
plt.plot(test['q'], test['P-value'], '--o')
plt.title('P-value of Random Walk Hypothesis for Daily Returns of NASDAQ Index␣
 ↪- 2019-2021')
plt.xlabel('Q-period')
plt.ylabel('P-value')
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7])
```

```
plt.show()
```



When analysing the excess returns for the second chosen period, we see the same results: we reject hypothesis RW3. Now, let's proceed to the testing of RW1.

[23]:
```
X = sm.add_constant(df_returns.loc[~mask, 'excess_returns_lag1'])
model = sm.OLS(df_returns.loc[~mask, 'excess_returns'], X, missing='drop')
res = model.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         excess_returns   R-squared:                       0.997
Model:                            OLS   Adj. R-squared:                  0.997
Method:                 Least Squares   F-statistic:                 2.178e+05
Date:                Sun, 19 Sep 2021   Prob (F-statistic):               0.00
Time:                        20:24:13   Log-Likelihood:                 1405.1
No. Observations:                 651   AIC:                            -2806.
Df Residuals:                     649   BIC:                            -2797.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
const                   0.0006      0.002      0.375      0.708      -0.002
0.004
excess_returns_lag1     0.9977      0.002    466.680      0.000       0.993
1.002
==============================================================================
Omnibus:                      371.940   Durbin-Watson:                   2.696
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            12211.647
```

```
Skew:                           1.945   Prob(JB):                         0.00
Kurtosis:                      23.858   Cond. No.                         2.61
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
/home/josebarretto/anaconda3/envs/ftd/lib/python3.7/site-
packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of
pandas all arguments of concat except for the argument 'objs' will be keyword-
only
  x = pd.concat(x[::order], 1)
```

[24]:
```python
mu = res.params['const']
errors = res.resid
std = np.std(errors)
pi = norm().cdf(mu/std)
cj = (pi**2 + (1-pi)**2)/((2*pi)*(1-pi))
print("Estimated CJ coefficient:", cj)
```

```
Estimated CJ coefficient: 1.0005609758432157
```

The same result is also verified here, since the estimate for CJ is very close to 1. For this reason, we do not reject the Strong Random Walk (RW1) hypothesis.