Tut1-read

une 10, 2021

1 Import Pac

[24]:
```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```


2 Reading data from Excel and Data Cleaning

[25]:
```
#Reading Excel file
dt=pd.read_excel("C:\\Users\\rluck\\OneDrive\\UNSW\\Financial Econometrics-S2␣
↪2021\\ASX200-SE-Indexes.xlsx")
#Start with 28/7/1995
dta=dt.iloc[0:]
dta.head()
```


[25]:

	Date	ASX200	SSE
0	28/7/1995	2087.8	721.506
1	31/7/1995	2083.3	721.007
2	1995-01-08 00:00:00	2087.6	741.821
3	1995-02-08 00:00:00	2107.0	724.914
4	1995-03-08 00:00:00	2108.2	728.542

[26]:
```
#Renaming Columns
dta_cols =['Date','ASX','SSE']
dta.columns= dta_cols
dta.head()
dta.tail()
```

[26]:

	Date	ASX	SSE
5214	23/7/2015	5590.3	4320.844
5215	24/7/2015	5566.1	4265.340
5216	27/7/2015	5589.9	3903.456
5217	28/7/2015	5584.7	3836.990
5218	29/7/2015	5624.2	3969.366

[27]:
```
#Save the file to hard drive under new name and re-read the new data file (not␣
↪necessary but helpful if you want to keep a record)
```

[28]:
```
dta.to_excel('C:\\Users\\rluck\\OneDrive\\UNSW\\Financial Econometrics-S2␣
↪2021\\ASX200-SE
dat=pd.read_excel                    \\OneDrive\\UNSW\\Financial Econometrics-S2␣
↪2021\\ASX200-SE
```

3 Computing

Daily Returns can be following formula:

$$R = ln(P_t/P_{t-1})$$

To express it in % we multiply the above by 100.

In Python, we can use the following data['R'] = 100.nplog(data['P']/data['P'].shift(1).dropna()

NB: We add shift(1) to show the lag in price and then drop N/A by using dropna() at the end

[29]:
```
#ASX200 Stock Index Returns
dat['R_a'] = 100*np.log(dat['ASX']/dat['ASX'].shift(1)).dropna()
#SSE Index returns
dat['R_sse'] =100*np.log(dat['SSE']/dat['SSE'].shift(1)).dropna()
data=dat.dropna()
data.head()
```

[29]:

	Unnamed: 0	Date	ASX	SSE	R_a	R_sse
1	1	31/7/1995	2083.3	721.007	-0.215771	-0.069185
2	2	1995-01-08 00:00:00	2087.6	741.821	0.206191	2.845913
3	3	1995-02-08 00:00:00	2107.0	724.914	0.925005	-2.305495
4	4	1995-03-08 00:00:00	2108.2	728.542	0.056937	0.499225
5	5	1995-04-08 00:00:00	2119.9	735.500	0.553441	0.950526

4 Descriptive Statistics

[30]:
```
stats.describe(data['R_a'])
```

[30]:
```
DescribeResult(nobs=5218, minmax=(-8.704293656938496, 5.724441325766271),
mean=0.018991334529736698, variance=0.9267577439903348,
skewness=-0.4858255708371836, kurtosis=6.327508480057597)
```

[31]:
```
stats.describe(data['R_sse'])
```

[31]:
```
DescribeResult(nobs=5218, minmax=(-10.44676691132011, 9.48095053904682),
mean=0.03267575640106423, variance=2.7784345990611756,
skewness=-0.2945797862973655, kurtosis=5.3319081854664265)
```

Kurtosis results in Python will have deducted 3, implying excess Kurtosis is already given. Both SSE and ASX have negatively skewed stock returns distribution and excess kurtosis. Although SSE has higher volatility, it has a higher mean daily return over the sample period.

5 Jarque-Ber

[32]:
```
JB_ASX= stats.jar                    '])
JB_ASX
```

[32]: Jarque_beraResult 449516528, pvalue=0.0)

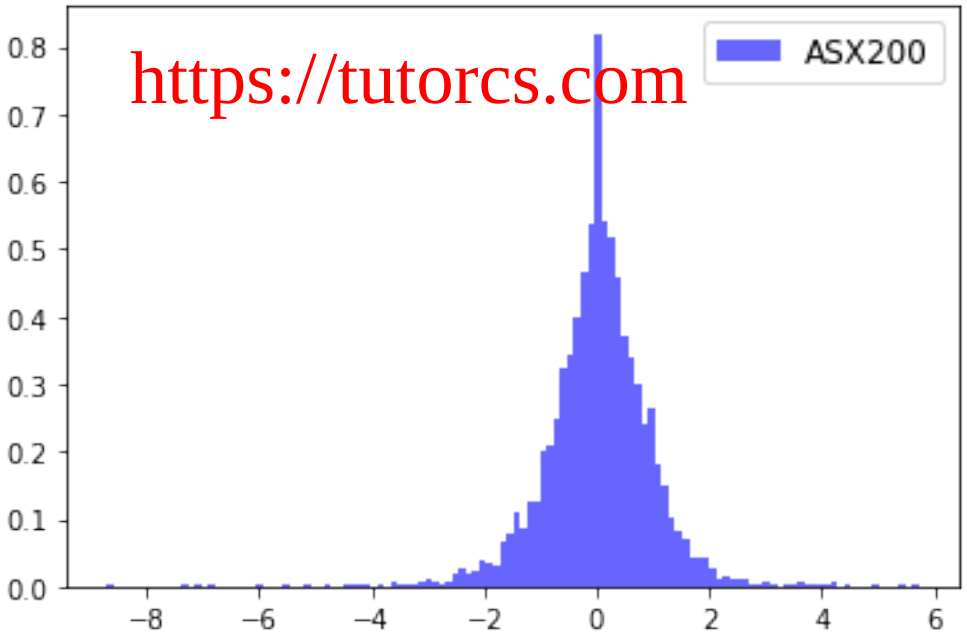
[33]:
```
JB_SSE= stats.jarque_bera(data['R_sse'])
JB_SSE
```

[33]: Jarque_beraResult(statistic=6256.458943807005, pvalue=0.0)

Interpretation: Since pvalue_JB <0.05, we can reject the null hypothesis of normality. We can infer that there is non-normal distribution.

6 Histogram

[75]:
```
#Plot histogram for ASX200
plt.hist(data['R_a'],bins=120,label='ASX200', density=True, alpha=0.6,␣
↪color='b')
plt.legend(loc='best', fontsize='large')
plt.show()
```

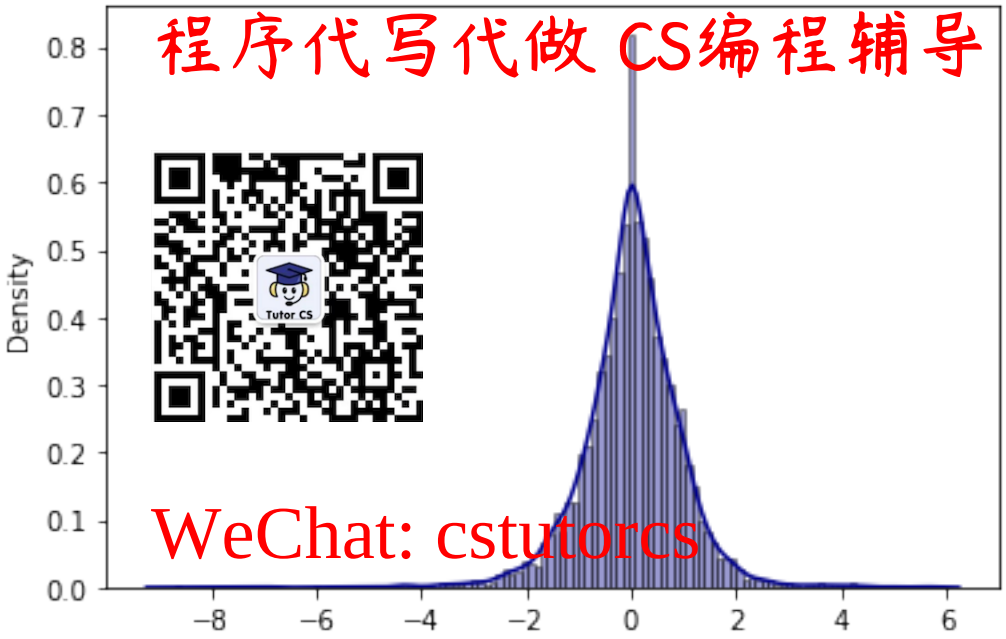



[77]:
```
#Plot histogram for SSE
plt.hist(data['R_sse'],bins=120,label='SSE', density=True, alpha=0.6, color='r')
plt.legend(loc='b                rge')
plt.show()
```

[79]:
```
import seaborn as sns
sns.distplot(data['R_a'],hist=True, kde=True, bins=int(120), color=␣
↪'darkblue',hist_kws={'edgecolor':'black'})
```

```
C:\Users\rluck\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[79]:
```
<AxesSubplot:xlabel='R_a', ylabel='Density'>
```



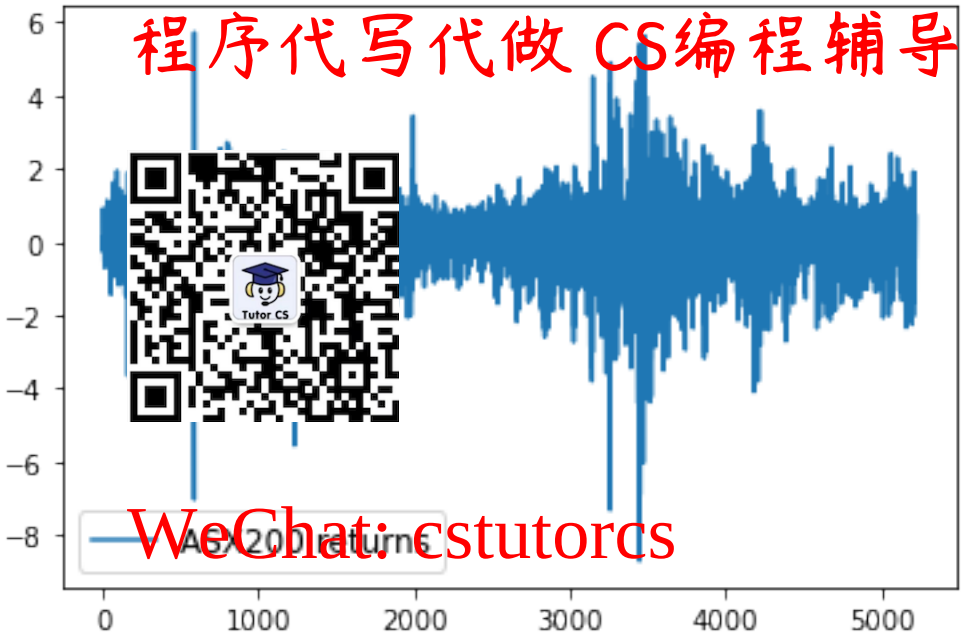
[81]:
```
sns.distplot(data['R_sse'],hist=True, kde=True, bins=int(120), color= 'red',␣
↪hist_kws={'edgecolor':'orange'})
```

```
C:\Users\rluck\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

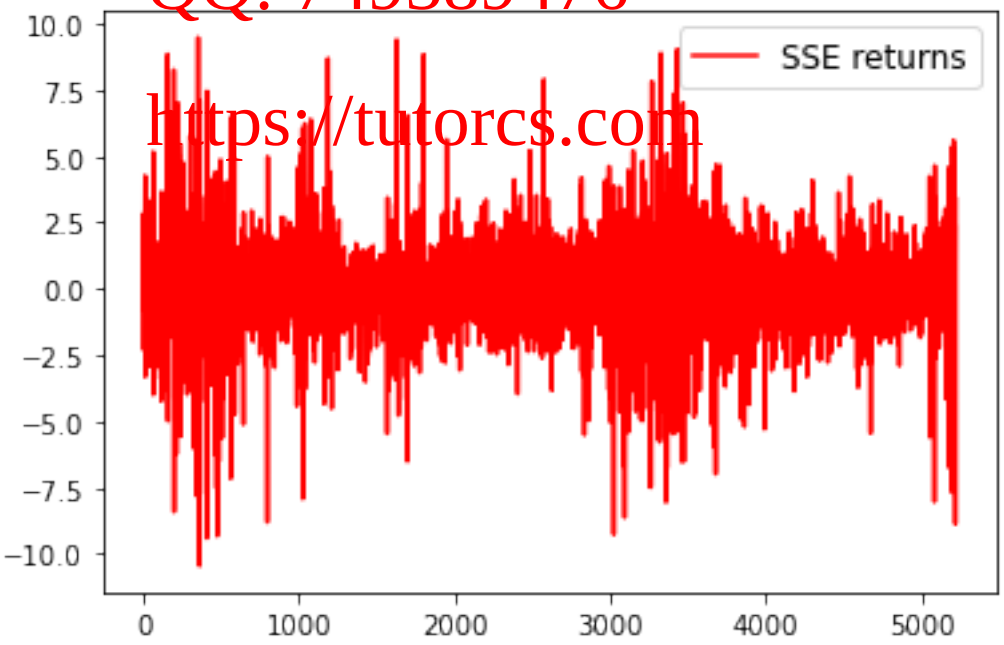
[81]:
```
<AxesSubplot:xlabel='R_sse', ylabel='Density'>
```

7 Plot

[36]:
```
#Plotting the ASX200 returns series
plt.plot(data.R_a,label='ASX200 returns')
plt.legend(loc='best', fontsize='large')
plt.show()
```



[37]:
```
#Plotting SSE returns series
plt.plot(data.R_sse,label='SSE returns',color='red')
plt.legend(loc='best', fontsize='large')
plt.show()
```



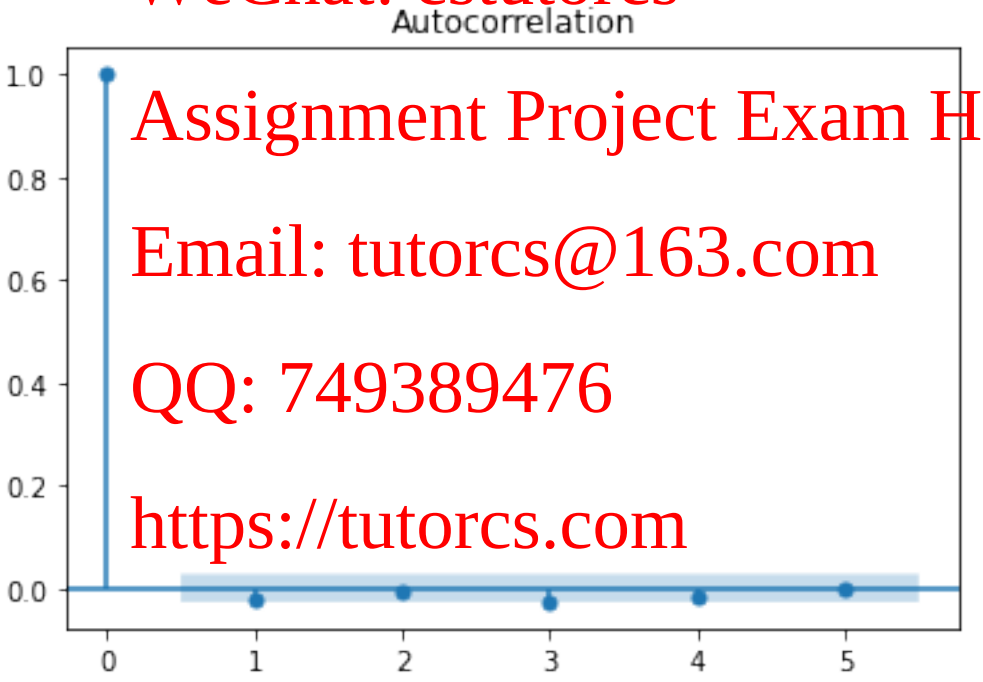
[38]:
```
#Computing correlation
from scipy.stats import pearsonr
Correlation=pearsonr(data.R_a,data.R_sse)
Correlation
```

[38]: (0.17300419339789[illegible]30516e-36)

The Pearson's correla[illegible] return series of ASX 200 and SSE is 0.17

8 Autocorrela[illegible]raph

[83]:
```
from statsmodels.graphics import tsaplots
fig =tsaplots.plot_acf(data['R_a'],lags=5)
plt.show()
```



[84]:
```
fig =tsaplots.plot_acf(data['R_sse'],lags=5)
plt.show()
```

9 Autocorrelation coefficients at multiple lags

[41]:
```
import statsmodels.api as sm
sm.tsa.acf(data['R_a'])
```

```
C:\Users\rluck\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:657:
FutureWarning: The default number of lags is changing from 40 tomin(int(10 *
np.log10(nobs)), nobs - 1) after 0.12is released. Set the number of lags to an
integer to  silence this warning.
  warnings.warn(
C:\Users\rluck\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:667:
FutureWarning: fft=True will become the default after the release of the 0.12
release of statsmodels. To suppress this warning, explicitly set fft=False.
  warnings.warn(
```

[41]:
```
array([ 1.00000000e+00, -2.00819036e-02, -7.45621736e-03, -2.59732010e-02,
       -1.59731301e-02,  6.21230060e-04, -1.76336711e-02, -1.55468191e-02,
       -1.50533801e-02,  2.42828613e-02,  5.09979156e-03, -1.20443357e-02,
       -7.90596922e-03,  1.16266121e-02, -9.44871407e-04, -1.14682419e-03,
       -8.27634852e-03,  1.25941056e-02, -2.52951690e-02,  8.15456959e-03,
       -4.90550380e-03,  1.19753088e-02, -1.13968485e-02,  4.71589237e-03,
       -2.09316328e-02,  2.11629759e-02, -8.17081382e-03,  7.62250565e-03,
        2.73483195e-02, -5.20350748e-03,  3.44330214e-03, -6.66124582e-03,
       -1.02522505e-02,  1.77346608e-02, -1.95879331e-02, -1.29511264e-02,
```

```
-1.43985714e-02,  1.29777824e-02,  7.13354214e-03, -7.65878735e-03,
 2.65162260e-02])
```

[42]:
```
sm.tsa.acf(data['R_sse'])
```

[42]:
```
array([ 1.0000000[illegible]e-03, -1.76342144e-02,  5.79484537e-02,
        4.2834380[illegible]e-03, -3.01447017e-02,  1.15330435e-03,
       -4.8872128[illegible]e-02,  4.85124876e-03, -1.93491880e-02,
        2.6723802[illegible]e-02, -4.08790279e-03,  6.22617857e-02,
        1.5207054[illegible]e-02,  6.66707910e-04, -3.56191982e-02,
        3.8364599[illegible]e-02,  2.52851825e-02, -1.23808014e-02,
       -1.9517479[illegible]e-02,  1.28303299e-02, -3.31447306e-03,
       -1.65438605e-02,  6.25110694e-02,  2.99159141e-03, -4.30915903e-03,
        1.16498707e-02,  8.18669819e-03,  1.35146257e-02,  5.33048366e-04,
        1.83304643e-02,  2.87361987e-03, -1.86447798e-02,  2.46929314e-02,
        4.33252016e-02])
```

[ ]: