University of New South Wales, School of Economics
Financial Econometrics
Tutorial 3 solutions

Question 1. Consider the AR(1) model

$$y_t = \alpha + b_1 y_{t-1} + \varepsilon_t \text{ where } \varepsilon_t \sim WN(0, \sigma^2).$$

(a) Calculate unconditional $E(y_t), \mathrm{var}(y_t)$ and $\mathrm{corr}(y_t, y_{t-i})$ for $i = 1, 2$.

$$E(y_t) = \alpha + b_1 E(y_{t-1}) + E(\varepsilon_t) = \alpha + b_1 E(y_{t-1}) + 0$$

Impose stationarity: $E(y_t) = E(y_{t-1})$ (because for non-stationary time-series unconditional mean does not exist).

$$E(y_t) = \frac{\alpha}{1 - b_1}$$

In a similar way

$$\mathrm{var}(y_t) = 0 + b_1^2 \,\mathrm{var}(y_{t-1}) + \sigma^2$$

$$\mathrm{var}(y_t) = \frac{\sigma^2}{1 - b_1^2}$$

$$\mathrm{corr}(y_t, y_{t-i}) = \mathrm{cov}(y_t, y_{t-i}) / \ \mathrm{var}(y_t)$$

We are going to use the fact that covariance is a linear operator in each of its arguments and assume stationarity

Proof of linearity (just as illustration), *x,y,z* –random variables; *a,b* - constants:

$$\mathrm{cov}(a + bx + y, z) = E[(a + bx + y - E(a + bx + y))(z - E(z))] =$$
$$= E[(a - E(a))(z - E(z)) + (bx - E(bx))(z - E(z)) + (y - E(y))(z - E(z))] =$$
$$= 0 + bE[(x - E(x))(z - E(z))] + E[(y - E(y))(z - E(z))] = b\,\mathrm{cov}(x, z) + \mathrm{cov}(y, z)$$

$$\mathrm{cov}(y_t, y_{t-1}) = \mathrm{cov}(\alpha + b_1 y_{t-1} + \varepsilon_t, y_{t-1}) = \mathrm{cov}(\alpha, y_{t-1}) + \mathrm{cov}(b_1 y_{t-1}, y_{t-1}) + \mathrm{cov}(\varepsilon_t, y_{t-1}) =$$
$$= 0 + b_1 \,\mathrm{var}(y_t) + 0$$

The last zero follows because of the assumption of white noise for ε_t. This implies that ε_t uncorrelated with all past ε_{t-i} and hence past y_{t-i}. Note that all future y_{t+i} are correlated with ε_t. The shock lives in the AR model infinitely.

Also note that you could derive the expression for covariance using the basic definition of the covariance. The expression would be more tedious, but the result would be the same.

$$\mathrm{corr}(y_t, y_{t-1}) = b_1$$

$$\mathrm{cov}(y_t, y_{t-1}) = \mathrm{cov}(\alpha + b_1(\alpha + b_1 y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t, y_{t-2}) =$$
$$= \mathrm{cov}(\alpha, y_{t-2}) + \mathrm{cov}(b_1(\alpha + b_1 y_{t-2} + \varepsilon_{t-1}), y_{t-2}) + \mathrm{cov}(\varepsilon_t, y_{t-2}) =$$
$$= 0 + 0 + b_1^2\ \mathrm{var}(y_{t-2}) + 0 + 0$$

$$\mathrm{corr}(y_t, y_{t-2}) = b_1^2$$

(b) What is the (optimal) forecast of y_{t+i}, for $i = 1, 2$ on the basis of time t information?

$$E(y_{t+1} \mid \Omega_t) = \alpha + b_1 y_t$$
$$E(y_{t+2} \mid \Omega_t) = \alpha + b_1(\alpha + b_1 y_t)$$

(c) Calculate conditional variance $\mathrm{var}(y_{t+1} \mid \Omega_t)$ and form confidence interval for forecast.

$\mathrm{var}(y_{t+1} \mid \Omega_t) = \sigma^2$ as $\alpha + b_1 y_t$ is a constant under Ω_t

CI: $\alpha + b_1 y_t \pm s\sigma$, where s depends on the confidence level s=1.96 at 95% confidence.

(d) Is y_t a white noise process?

y_t is white noise when $b_1 = 0$, but typically it is not a white noise process.

(e) When y_t is a covariance stationary process?

y_t is covariance stationary $|b_1| < 1$

(f) Think about an economic example where AR(1) is relevant?

Many economic aggregates like unemployment or GDP are AR(1).

Question 2. Suppose that a researcher estimated the lag 1 autocorrelation coefficient using a series of T=100 observations, and found it to be equal to 0.15. Is the autocorrelation coefficient significantly different from 0? Specify the null hypothesis, the alternative, test statistics, null distribution and decision criterion.

In case of two sided test (typically used for autocorrelations)

Ho: $\rho_1 = 0$. Ha: $\rho_1 \neq 0$. Null dist. - $N(0, \frac{1}{T})$, Test stat = $\frac{0.15}{1/\sqrt{100}} = 1.5$

Need to specify significance level, say, typically 5%. Decision rule: -1.96 < Test stat< 1.96
Decision: fail to reject the null: the estimate of the autocorrelation coefficient is not significantly different from 0.

Question 4.

Let $f_{t+h|t}$ be the forecast based on Ω_t. Namely, $f_{t+h|t}$ is a function of elements in Ω_t. Which $f_{t+h|t}$ minimises the mean square forecast error (MSFE)?

$$MSFE = E[(y_{t+h} - f_{t+h|t})^2 | \Omega_t].$$

Conditional mean $E[y_{t+h}|\Omega_t]$ is the best forecast in terms of the MSFE criterion

*Formal proof

Option 1. Explicitly write down the definition of the expectation in terms of the integral

$$MSFE = E[(y_{t+h} - f_{t+h|t})^2 | \Omega_t] = \int y_{t+h}^2 g(y_{t+h} | \Omega_t) dy_{t+h} - 2\int y_{t+h} f_{t+h|t} g(y_{t+h} | \Omega_t) dy_{t+h} + \int f_{t+h|t}^2 g(y_{t+h} | \Omega_t) dy_{t+h} =$$

$$= \int y_{t+h}^2 g(y_{t+h} | \Omega_t) dy_{t+h} - 2 f_{t+h|t} \int y_{t+h} g(y_{t+h} | \Omega_t) dy_{t+h} + f_{t+h|t}^2 \int g(y_{t+h} | \Omega_t) dy_{t+h} =$$

$$= \int y_{t+h}^2 g(y_{t+h} | \Omega_t) dy_{t+h} - 2 f_{t+h|t} E(y_{t+h} | \Omega_t) + f_{t+h|t}^2$$

Note we took out f outside of integral because it is not a function of y_{t+h}. We also used the fact that proper (conditional) density g integrates toward one and the definition of the conditional expectation.

FOC:

$$\frac{\partial MSFE}{\partial f_{t+h|t}} = -2E(y_{t+h} | \Omega_t) + 2 f_{t+h|t} \equiv 0 \to f_{t+h|t}^* = E(y_{t+h} | \Omega_t)$$

SOC:

$$\frac{\partial^2 MSFE}{\partial f_{t+h|t}^2} = 2 > 0 \to \text{true minimum}$$

Option 2. Subtract and add the correct answer in the squared term.

$$MSFE = E[(y_{t+h} - f_{t+h|t})^2 \mid \Omega_t] = E[(y_{t+h} - E(y_{t+h} \mid \Omega_t) + E(y_{t+h} \mid \Omega_t) - f_{t+h|t})^2 \mid \Omega_t] =$$
$$= E[(y_{t+h} - E(y_{t+h} \mid \Omega_t))^2 \mid \Omega_t] + 2E[(y_{t+h} - E(y_{t+h} \mid \Omega_t))(E(y_{t+h} \mid \Omega_t) - f_{t+h|t}) \mid \Omega_t] +$$
$$+E[(E(y_{t+h} \mid \Omega_t) - f_{t+h|t})^2 \mid \Omega_t]$$

The first term of the last equality above is not a function of f and the third term if quadratic in f. Hence, if we show that the second term is zero we have the proof.

$$E[(y_{t+h} - E(y_{t+h} \mid \Omega_t))(E(y_{t+h} \mid \Omega_t) - f_{t+h|t})) \mid \Omega_t] = (E(y_{t+h} \mid \Omega_t) - f_{t+h|t})E[(y_{t+h} - E(y_{t+h} \mid \Omega_t)) \mid \Omega_t] = 0$$

$E(y_{t+h} \mid \Omega_t) - f_{t+h|t}$ is just a constant under Ω_t and can be taken outside of the conditional expectation. Then, we just apply the expectation operator sequentially.

QUESTION 3:

Consider the OLS estimator in the AR(1) model:

$$y_t = \alpha + b_1 y_{t-1} + \epsilon_t$$

To find the OLS estimator, we minimize the sum of squared residuals:

$$SSR = \sum_{t=1}^{T} (y_t - \alpha - b_1 y_{t-1})^2 \tag{1}$$

$$\frac{\partial SSR}{\partial \hat{\alpha}} = -2\sum_{t}^{T} (y_t - \hat{\alpha} - \hat{b}_1 y_{t-1}) = 0 \tag{2}$$

$$\hat{\alpha} = \frac{\sum_t y_t - \hat{b}_1 \sum_t y_{t-1}}{T} \tag{3}$$

$$\frac{\partial SSR}{\partial \hat{b}_1} = -2\sum_{t}^{T} y_{t-1}(y_t - \hat{\alpha} - \hat{b}_1 y_{t-1}) = 0 \tag{4}$$

$$\sum_{t}^{T} y_{t-1} y_t - \hat{\alpha} \sum_t y_{t-1} - \hat{b}_1 \sum_t y_{t-1}^2 = 0 \tag{5}$$

After substituting $\hat{\alpha}$ and solving for $\hat{b}_1$

$$\begin{aligned}
\hat{b}_1 &= \frac{\sum_t y_{t-1} y_t - \frac{1}{T} \sum y_t \sum y_{t-1}}{\sum y_{t-1}^2 - \frac{1}{T}(\sum y_{t-1})^2} && (6)\\
&= \frac{\frac{1}{T}\sum_t y_{t-1} y_t - \frac{1}{T^2} \sum y_t \sum y_{t-1}}{\frac{1}{T}\sum y_{t-1}^2 - \frac{1}{T^2}(\sum y_{t-1})^2} && (7)\\
&\to \frac{E(y_{t-1} y_t) - E^2(y_t)}{E(y_t^2) - E^2(y_t)} && (8)
\end{aligned}$$

Now let us check the properties of this estimator in terms of bias and consistency.

$$\hat{b}_1 = \frac{\sum_t y_{t-1}(\alpha + b_1 y_{t-1} + \epsilon_t) - \frac{1}{T}\sum(\alpha + b_1 y_{t-1} + \epsilon_t)\sum y_{t-1}}{\sum y_{t-1}^2 - \frac{1}{T}(\sum y_{t-1})^2} \tag{9}$$

$$\hat{b}_1 = b_1 + \frac{\sum_{t=2}^{T} y_{t-1}\epsilon_t - \frac{1}{T}\cdot\sum_{t=2}^{T}\epsilon_t \sum_{t=2}^{T} y_{t-1}}{\sum_{t=2}^{T} y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^{T} y_{t-1})^2} \tag{10}$$

Taking unconditional expectation:

$$\begin{aligned}
E\left(\hat{b}_1\right) &= b_1 + E\left[\frac{\sum_{t=2}^{T} y_{t-1}\epsilon_t - \frac{1}{T}\cdot\sum_{t=2}^{T}\epsilon_t \sum_{t=2}^{T} y_{t-1}}{\sum_{t=2}^{T} y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^{T} y_{t-1})^2}\right] && (11)\\
&= b_1 + E\left[\frac{\sum_{t=2}^{T} y_{t-1}\epsilon_t}{\sum_{t=2}^{T} y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^{T} y_{t-1})^2}\right] - \frac{1}{T}E\left[\frac{(\sum_{t=2}^{T}\epsilon_t)(\sum_{t=2}^{T} y_{t-1})}{\sum_{t=2}^{T} y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^{T} y_{t-1})^2}\right]
\end{aligned}$$

Now, even with the assumption that $E(\epsilon_t|y_{t-1}) = 0$, we cannot write that $E\left((\sum_{t=2}^T \epsilon_t)(\sum_{t=2}^T y_{t-1})\right) = 0$, because ϵ_t is independent of y_{t-1} BUT ϵ_t is not independent of y_t, $E(y_t\epsilon_t) = E(\epsilon_t^2) \neq 0$. We cannot separate the product of the two sums and use the fact that $E(E(\epsilon_t|y_{t-1}) = 0$.

$$E\left((\sum_{t=2}^T \epsilon_t)(\sum_{t=2}^T y_{t-1})\right) \neq E\left(\sum_{t=2}^T E(\epsilon_t|y_{t-1})(\sum_{t=2}^T y_{t-1})\right)$$

For the first expectation, we have the same issue, even if ϵ_t and y_{t-1} are independent, ϵ_t is not independent from the numerator $\sum_{t=2}^T y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^T y_{t-1})^2$, therefore, we cannot separate the two expressions in the fraction:

$$\begin{aligned} E\left[\frac{\sum_{t=2}^T y_{t-1}\epsilon_t}{\sum_{t=2}^T y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^T y_{t-1})^2}\right] &= E\left[\left(\frac{\sum_{t=2}^T y_{t-1}}{\sum_{t=2}^T y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^T y_{t-1})^2}\right)\epsilon_t\right] \quad (12) \\ &\neq E\left[\left(\frac{\sum_{t=2}^T y_{t-1}}{\sum_{t=2}^T y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^T y_{t-1})^2}\right)E(\epsilon_t|y_{t-1})\right] \end{aligned}$$

OLS is BIASED in the AR(1) or any AR(p) model.

In this question, to show asymptotic normality, we do not need unbiased estimator. We can use the LLN and CLT.

The LLN gives us the result that OLS is consistent. The key to using the LLN is to write everything in terms of averages, so every sum term is divided by T:

$$\begin{aligned} \text{plim}\left(\hat{b}_1\right) &= b_1 + \text{plim}\left[\frac{\sum_{t=2}^T y_{t-1}\epsilon_t - \frac{1}{T}\cdot\sum_{t=2}^T \epsilon_t \sum_{t=2}^T y_{t-1}}{\sum_{t=2}^T y_{t-1}^2 - \frac{1}{T}(\sum_{t=2}^T y_{t-1})^2}\right] \quad (13) \\ &= b_1 + \text{plim}\left[\frac{\sum_{t=2}^T y_{t-1}\epsilon_t/T}{\sum_{t=2}^T y_{t-1}^2/T - \frac{1}{T^2}(\sum_{t=2}^T y_{t-1})^2}\right] \quad (14) \\ &- \text{plim}\left[\frac{(\sum_{t=2}^T \epsilon_t/T)(\sum_{t=2}^T y_{t-1}/T)}{\sum_{t=2}^T y_{t-1}^2/T - \frac{1}{T^2}(\sum_{t=2}^T y_{t-1})^2}\right] \end{aligned}$$

By the LLN, each average converges to the expected value of its argument:

$$\begin{aligned} \text{plim}\sum_{t=2}^T \epsilon_t/T &= E(\epsilon_T) = 0, \quad (15) \\ \text{plim}\sum_{t=2}^T y_{t-1}\epsilon_t/T &= E(y_{t-1}\epsilon_t) = E(y_{t-1}E(\epsilon_t|y_{t-1})) = 0 \quad (16) \end{aligned}$$

We get the result: plim $\hat{b}_1 = b_1$ which is equal to zero under the null hypothesis in the question of the tutorial. To show asymptotic normality, we use the CLT:

$$\begin{aligned}
\text{plim } \sqrt{T}\hat{b}_1 &= \left[\frac{\text{plim } \left(\sum_{t=2}^{T} y_{t-1}\epsilon_t/\sqrt{T} \right)}{\text{plim } \left(\sum_{t=2}^{T} y_{t-1}^2/T - \frac{1}{T^2}(\sum_{t=2}^{T} y_{t-1})^2 \right)} \right] \\
&- \left[\frac{(\text{ plim } \sum_{t=2}^{T} \epsilon_t/\sqrt{T})(\text{ plim } \sum_{t=2}^{T} y_{t-1}/T)}{\text{plim } \left(\sum_{t=2}^{T} y_{t-1}^2/T - \frac{1}{T^2}(\sum_{t=2}^{T} y_{t-1})^2 \right)} \right]
\end{aligned} \tag{17}$$

For simplicity, let us call $Q = \text{ plim } \left(\sum_{t=2}^{T} y_{t-1}^2/T - \frac{1}{T^2}(\sum_{t=2}^{T} y_{t-1})^2 \right)$ and assume that it exists. Which will because $Q = V(y_{t-1})$ and stationarity it should be finite. We also assume that $\mu = \text{ plim } \sum_{t=2}^{T} y_{t-1}^2/T = E(y_{t-1})$ exists and finite (under stationarity).

$$\begin{aligned}
\text{plim } \sqrt{T}\hat{b}_1 &= \left[\frac{\text{plim } \left(\sum_{t=2}^{T} y_{t-1}\epsilon_t/\sqrt{T} \right)}{Q} \right] \\
&- \left[\frac{(\text{ plim } \sum_{t=2}^{T} \epsilon_t/\sqrt{T})E(y_{t-1})}{Q} \right]
\end{aligned} \tag{18}$$

By CLT, we know that if Z_t is $N(0, \sigma^2)$, then $\frac{\sum_t Z_t}{\sqrt{T}}$ is approximately normal $N(0, \sigma^2)$. Therefore:

$$\sum_{t=2}^{T} y_{t-1}\epsilon_t/\sqrt{T} \sim N\left(0, \sigma^2 \text{plim } \frac{\sum_t y_{t-1}^2}{T}\right) \tag{19}$$

$$\sum_{t=2}^{T} \epsilon_t/\sqrt{T} \sim N(0, \sigma^2) \tag{20}$$

Using properties of normally distributed random variables: for any constant c, and X that is $N(a, b)$, cX is $N(a, c^2 b)$.

$$\sqrt{T}\hat{b}_1 \sim \frac{1}{Q} N\left(0, \sigma^2 \text{ plim} \frac{\sum_t y_{t-1}^2}{T}\right) - \frac{E(y_{t-1})}{Q} N(0, \sigma^2) \tag{21}$$

$$\sim N(0, \sigma^2 V) \tag{22}$$

$$V = \left(\text{plim} \frac{\sum_t y_{t-1}^2}{T} - E(y_{t-1})^2 \right) / Q^2 = 1/Q = 1/V(y_t) \tag{23}$$

$$V(y_t) = \sigma^2/(1 - b_1^2) \tag{24}$$

$$V = (1 - b_1^2)/\sigma^2 \tag{25}$$

Therefore, $\sqrt{T}\hat{b}_1 \sim N(0, (1 - b_1^2))$. Under the null hypothesis of $b_1 = 0$, we have $\sqrt{T}\hat{b}_1 \sim N(0, 1)$, which means that we can approximate the distribution of $\hat{b}_1$ with $N(0, 1/T)$.

CAPMGEGO

June 24, 2021

1 Calculate the daily log returns of T-Bill, gold, GE stock and market

[92]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[93]:
```
data = pd.read_csv("C:\\Users\\rluck\\OneDrive\\capm.csv", header=[4])
data
```

[93]:

	DATE	Gold	S&P500	Rf	GE
0	12/08/1975	166.05	87.12	6.40	0.9218
1	13/08/1975	163.50	85.97	6.45	0.9036
2	14/08/1975	163.50	85.60	6.45	0.9036
3	15/08/1975	163.50	86.36	6.42	0.9244
4	18/08/1975	163.50	86.20	6.42	0.9348
…	…	…	…	…	…
10432	6/08/2015	1090.15	2083.56	0.04	26.0800
10433	7/08/2015	1096.85	2077.57	0.06	25.7900
10434	10/08/2015	1103.35	2104.18	0.12	26.2400
10435	11/08/2015	1109.67	2084.07	0.10	25.7100
10436	12/08/2015	1123.85	2086.05	0.10	25.8600

[10437 rows x 5 columns]

#Computing log returns: R_gold = 100*ln(P_g/P_g-1)

Rf = 100/360*ln(1+rf)

[94]:
```
data['R_gold']=100*np.log(data['Gold']/data['Gold'].shift(1))
data['R_f']= 100/360*np.log(1+data['Rf']/100)
data['R_GE']= 100*np.log(data['GE']/data['GE'].shift(1))
data['R_m']= 100*np.log(data['S&P500']/data['S&P500'].shift(1))
print(data.head())
```

```
         DATE    Gold  S&P500    Rf      GE    R_gold       R_f      R_GE  \
0  12/08/1975  166.05   87.12  6.40  0.9218       NaN  0.017232       NaN
1  13/08/1975  163.50   85.97  6.45  0.9036 -1.547596  0.017363 -1.994150
2  14/08/1975  163.50   85.60  6.45  0.9036  0.000000  0.017363  0.000000
```

```
3  15/08/1975  163.50   86.36  6.42  0.9244  0.000000  0.017284  2.275809
4  18/08/1975  163.50   86.20  6.42  0.9348  0.000000  0.017284  1.118772

        R_m
0       NaN
1 -1.328808
2 -0.431312
3  0.883932
4 -0.185443
```

2 Calculating excess returns for gold and GE

[95]:

```python
data['R_p']= data['R_m']- data['R_f']
data['R_ge']= data['R_GE']-data['R_f']
data['R_go']= data['R_gold']-data['R_f']
data
```

[95]:

```
             DATE     Gold   S&P500    Rf       GE    R_gold       R_f  \
0      12/08/1975   166.05    87.12  6.40   0.9216       NaN  0.017232
1      13/08/1975   163.50    85.97  6.45   0.9036 -1.547596  0.017363
2      14/08/1975   163.50    85.60  6.45   0.9036  0.000000  0.017363
3      15/08/1975   163.50    86.36  6.42   0.9244  0.000000  0.017284
4      18/08/1975   163.50    86.20  6.42   0.9348  0.000000  0.017284
…             …        …        …     …        …         …         …
10432   6/08/2015  1090.15  2083.56  0.04  26.0300  0.415484  0.000111
10433   7/08/2015  1096.85  2077.57  0.06  25.7900  0.612713  0.000167
10434  10/08/2015  1103.35  2104.18  0.12  26.2400  0.590857  0.000333
10435  11/08/2015  1109.67  2084.07  0.10  25.7100  0.571167  0.000278
10436  12/08/2015  1123.85  2086.05  0.10  25.8600  1.269762  0.000278

           R_GE       R_m       R_p      R_ge      R_go
0           NaN       NaN       NaN       NaN       NaN
1     -1.994150 -1.328808 -1.346171 -2.011512 -1.564958
2      0.000000 -0.431312 -0.448674 -0.017363 -0.017363
3      2.275809  0.883932  0.866648  2.258525 -0.017284
4      1.118772 -0.185443 -0.202727  1.101488 -0.017284
…           …         …         …         …         …
10432 -0.268560 -0.778318 -0.778429 -0.268671  0.415373
10433 -0.926290 -0.287903 -0.288069 -0.926457  0.612547
10434  1.729814  1.272690  1.272357  1.729481  0.590524
10435 -2.040494 -0.960313 -0.960591 -2.040772  0.570889
10436  0.581735  0.094961  0.094684  0.581458  1.269484

[10437 rows x 12 columns]
```

3 Data : Remove N/A

[96]:
```
data = data.dropna(subset=["R_p"])
data.to_csv("C:\\Users\\rluck\\OneDrive\\capm1.csv")
data.head()
```

[96]:
```
         DATE   Gold  S&P500    Rf      GE    R_gold       R_f      R_GE  \
1  13/08/1975  163.5   85.97  6.45  0.9036 -1.547596  0.017363 -1.994150
2  14/08/1975  163.5   85.60  6.45  0.9036  0.000000  0.017363  0.000000
3  15/08/1975  163.5   86.36  6.42  0.9244  0.000000  0.017284  2.275809
4  18/08/1975  163.5   86.20  6.42  0.9348  0.000000  0.017284  1.118772
5  19/08/1975  163.5   84.95  6.47  0.9218  0.000000  0.017415 -1.400432

        R_m       R_p      R_ge      R_go
1 -1.328808 -1.346171 -2.011512 -1.564958
2 -0.431312 -0.448674 -0.017363 -0.017363
3  0.883932  0.866648  2.258525 -0.017284
4 -0.185443 -0.202727  1.101488 -0.017284
5 -1.460733 -1.478148 -1.417847 -0.017415
```


[97]:
```
!pip install sklearn
!pip install statsmodels
```



```
Requirement already satisfied: sklearn in c:\users\rluck\anaconda3\lib\site-
packages (0.0)
Requirement already satisfied: scikit-learn in
c:\users\rluck\anaconda3\lib\site-packages (from sklearn) (0.24.1)
Requirement already satisfied: scipy>=0.19.1 in
c:\users\rluck\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.6.2)
Requirement already satisfied: joblib>=0.11 in
c:\users\rluck\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.0.1)
Requirement already satisfied: numpy>=1.13.3 in
c:\users\rluck\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.20.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\rluck\anaconda3\lib\site-packages (from scikit-learn->sklearn) (2.1.0)
Requirement already satisfied: statsmodels in c:\users\rluck\anaconda3\lib\site-
packages (0.12.2)
Requirement already satisfied: numpy>=1.15 in c:\users\rluck\anaconda3\lib\site-
packages (from statsmodels) (1.20.1)
Requirement already satisfied: scipy>=1.1 in c:\users\rluck\anaconda3\lib\site-
packages (from statsmodels) (1.6.2)
Requirement already satisfied: pandas>=0.21 in
c:\users\rluck\anaconda3\lib\site-packages (from statsmodels) (1.2.4)
Requirement already satisfied: patsy>=0.5 in c:\users\rluck\anaconda3\lib\site-
packages (from statsmodels) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\users\rluck\anaconda3\lib\site-packages (from pandas>=0.21->statsmodels)
(2.8.1)
```

```
Requirement already satisfied: pytz>=2017.3 in
c:\users\rluck\anaconda3\lib\site-packages (from pandas>=0.21->statsmodels)
(2021.1)
Requirement already satisfied: six in c:\users\rluck\anaconda3\lib\site-packages
(from patsy>=0.5->statsmodels) (1.15.0)
```

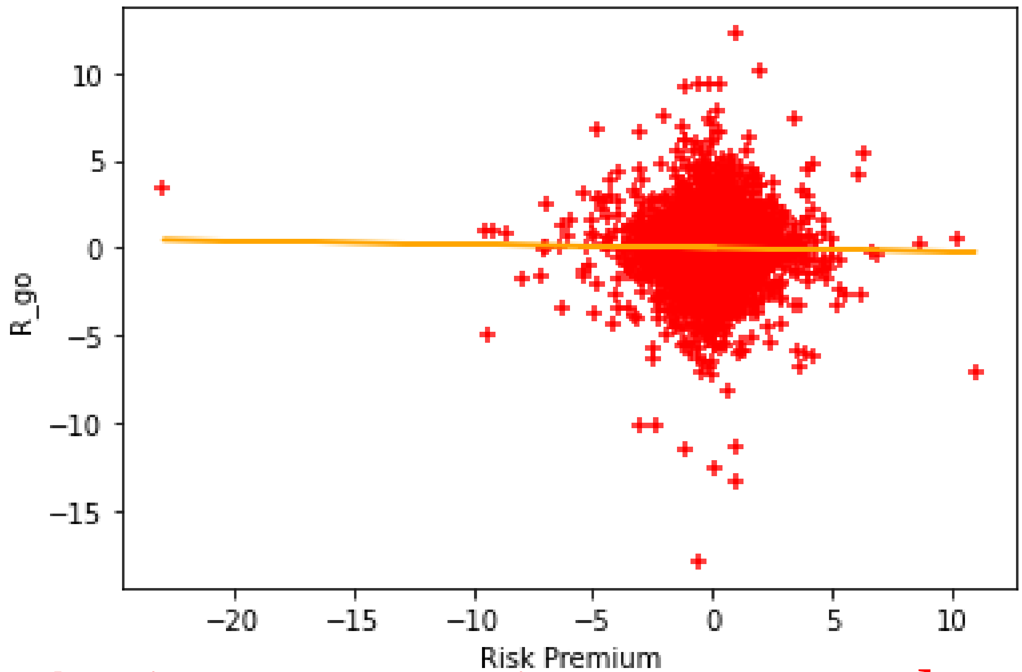
[98]:
```
%matplotlib inline
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn import linear_model
import matplotlib.pyplot as plt
```

4 I. Plotting Gold excess returns with market excess returns

[99]:
```
#Regressing excess returns on gold (R_g-Rf) over risk-free rate against the␣
↪excess market return (Rp=Rm-rf)
reg = linear_model.LinearRegression()
X =data[['R_p']].dropna()
y1 =data['R_go'].dropna()
reg.fit(X,y1)
predictions =reg.predict(X)
```


[100]:
```
plt.xlabel('Risk Premium')
plt.ylabel('R_go')
plt.scatter(data.R_p,data.R_gold,color='red',marker='+')
plt.plot(data.R_p,reg.predict(data[['R_p']]), color='orange')
```


[100]:
```
[<matplotlib.lines.Line2D at 0x18397b4e220>]
```



[101]:
```
#model with intercept
X= sm.add_constant(X)
model = sm.OLS(y1,X).fit()
predictions = model.predict(X)
j= (model.summary())
print(j)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   R_go   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     3.181
Date:                Thu, 24 Jun 2021   Prob (F-statistic):             0.0745
Time:                        13:42:13   Log-Likelihood:                -16959.
No. Observations:               10436   AIC:                         3.392e+04
Df Residuals:                   10434   BIC:                         3.394e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0057      0.012      0.473      0.637      -0.018       0.029
R_p           -0.0201      0.011     -1.784      0.075      -0.042       0.002
==============================================================================
Omnibus:                     2812.110   Durbin-Watson:                   2.071
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           111926.422
Skew:                          -0.573   Prob(JB):                         0.00
Kurtosis:                      19.003   Cond. No.                         1.07
==============================================================================

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

DW-stats of 2.071 is close to 2.0, implying that there is no serial correlation.

Yet, the p-value of the beta coefficient indicates that it is slightly significant at 7.5% significance level and the R-squared is very low, explaining low explanatory power of the model.

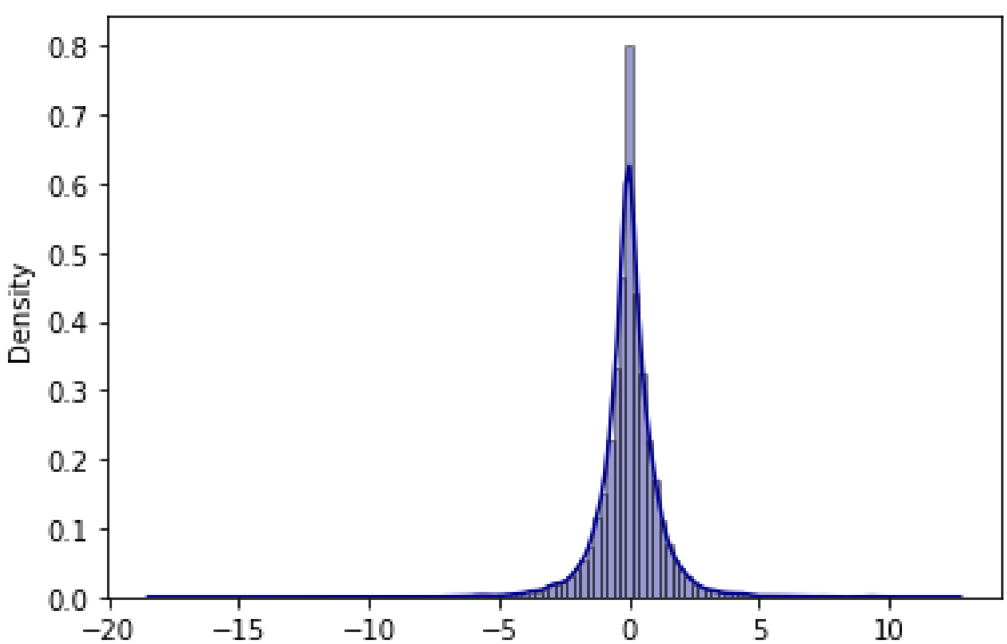
5 Residuals plot for gold

[102]:
```python
residuals_go = model.resid
import seaborn as sns
sns.distplot(residuals_go,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)
```


[102]: <AxesSubplot:ylabel='Density'>



```
[103]: from scipy import stats
       JB_go= stats.jarque_bera(residuals_go)
       JB_go
```

```
[103]: Jarque_beraResult(statistic=111926.42195044507, pvalue=0.0)
```

The plot and JB test (p-value <0.05) rejects the null hypothesis of normality. It is clearly a non-normal distribution.

6 Cusum Test for Gold

```
[104]: # endog = data.R_go
       Rp = data.R_p
       endog = data.R_go
       exog = sm.add_constant(Rp)
       mod = sm.RecursiveLS(endog,exog)
       res_1 = mod.fit()
       fig = res_1.plot_cusum(figsize=(10,6));
```



```
C:\Users\rluck\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:578: ValueWarning: An unsupported
index was provided and will be ignored when e.g. forecasting.
  warnings.warn('An unsupported index was provided and will be'
```


Cusum test of stability for gold shows high periods of instability during the early part of the graph (namely before 1980s).Then, the beta stabilises.

7 White Test of Heteroskedasticity for Gold

[105]:
```
from statsmodels.stats.diagnostic import het_white
from statsmodels.compat import lzip
from patsy import dmatrices
```

[106]:
```
expr = 'y1 ~ X'
y1, X = dmatrices(expr, data, return_type='dataframe')
olsr_results = smf.ols(expr, data).fit()
keys = ['Lagrange Multiplier statistic:', 'LM test\'s p-value:', 'F-statistic:
 ↪', 'F-test\'s p-value:']
results = het_white(olsr_results.resid, X)
lzip(keys, results)
```

[106]:
```
[('Lagrange Multiplier statistic:', 36.866651106570274),
 ("LM test's p-value:", 9.874348003656595e-09),
 ('F-statistic:', 18.49335699862814),
 ("F-test's p-value:", 9.608158442586967e-09)]
```


LM test statistic is 36.87 and the corresponding p-value is 0

F-stats = 18.49 and the corresponding p-value is 0

Since the p-value of the both LM and F-stats is less than 0.05, we reject the null hypothesis that there is no heteroskedasticity in the residuals. It infers that the heteroskedasticity exists and the standard errors need to be corrected.

8 Breusch-Godfrey LM test for Gold

[107]:
```
import statsmodels.stats.diagnostic as dg
print (dg.acorr_breusch_godfrey(model, nlags= 2))
```

```
(14.058774886495657, 0.0008854740175917412, 7.036171882380294,
0.0008836668869260258)
```

T-statistic of Chi-squared is 14.0588 and the corresponding p-value is 0.0009

F-statistic is 7.0362 and the corresponding p-value is 0.0009

Since p-value is less than 0.05, we reject the null hypothesis, thus inferring there is some autocorrelation at order less than or equal to 2.0

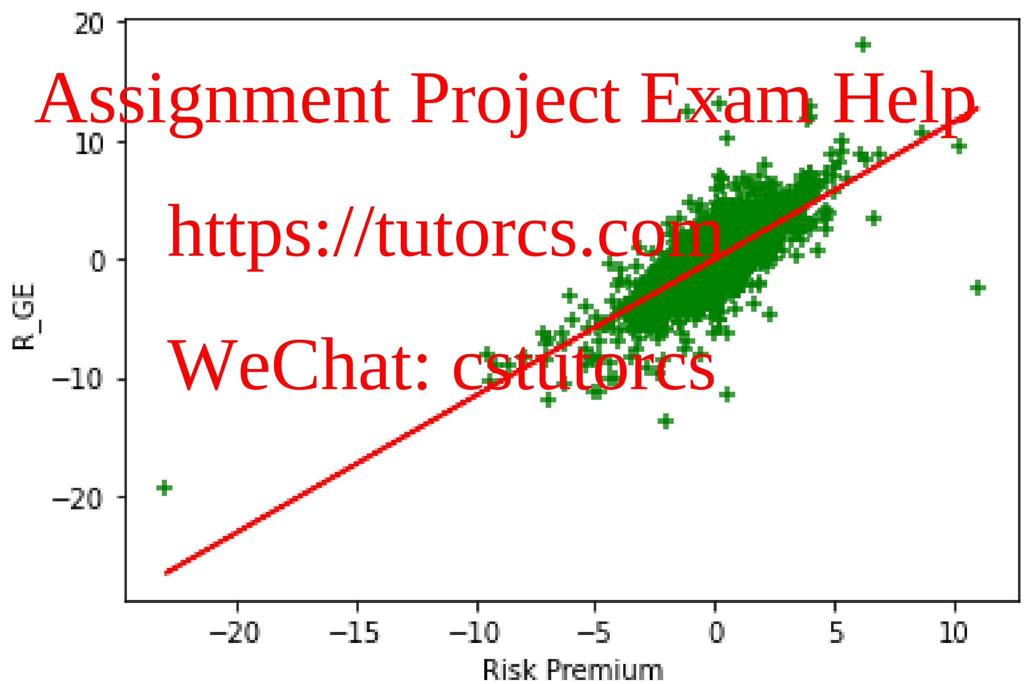
9 II. Plotting GE excess returns with market excess returns

[108]:
```
%matplotlib inline
reg = linear_model.LinearRegression()
X =data[['R_p']]
y =data['R_ge']
reg.fit(X,y)
```

[108]:
```
LinearRegression()
```

[109]:
```
plt.xlabel('Risk Premium')
plt.ylabel('R_GE')
plt.scatter(data.R_p,data.R_GE,color='green',marker='+')
plt.plot(data.R_p,reg.predict(data[['R_p']]), color='red')
```

[109]:
```
[<matplotlib.lines.Line2D at 0x18397269b20>]
```



10 Regressing GE excess return with market excess return

[110]:
```
#model with intercept
X =sm.add_constant(X)
model_1 = sm.OLS(y,X).fit()
predictions = model_1.predict(X)
j= (model_1.summary())
```

```python
print(j)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   R_ge   R-squared:                       0.564
Model:                            OLS   Adj. R-squared:                  0.564
Method:                 Least Squares   F-statistic:                 1.351e+04
Date:                Thu, 24 Jun 2021   Prob (F-statistic):               0.00
Time:                        13:42:14   Log-Likelihood:                -15682.
No. Observations:               10436   AIC:                         3.137e+04
Df Residuals:                   10434   BIC:                         3.138e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0012      0.011     -0.114      0.909      -0.022       0.020
R_p            1.1569      0.010    116.224      0.000       1.137       1.176
==============================================================================
Omnibus:                     2325.123   Durbin-Watson:                   1.995
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           109234.319
Skew:                           0.109   Prob(JB):                         0.00
Kurtosis:                      18.848   Cond. No.                         1.07
==============================================================================

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


DW-stats of 1.995 is close to 2.0, implying that there is no serial correlation.

Since p-value of the beta coefficient is less than 0.05, we reject the null hypothesis that beta is zero.

The CAPM equation for GE can be written as follows:

$R_ge = 1.1569 * R_p + Rf$

where R_ge is the return from GE stock, $R_p = Rm - Rf$ is the market risk premium and Rf is the risk free rate of return

If we want to replicate the returns from GE, we can rearrange the above equation:

$R_ge = 1.1569 * Rm + (1 - 1.1569) * Rf$

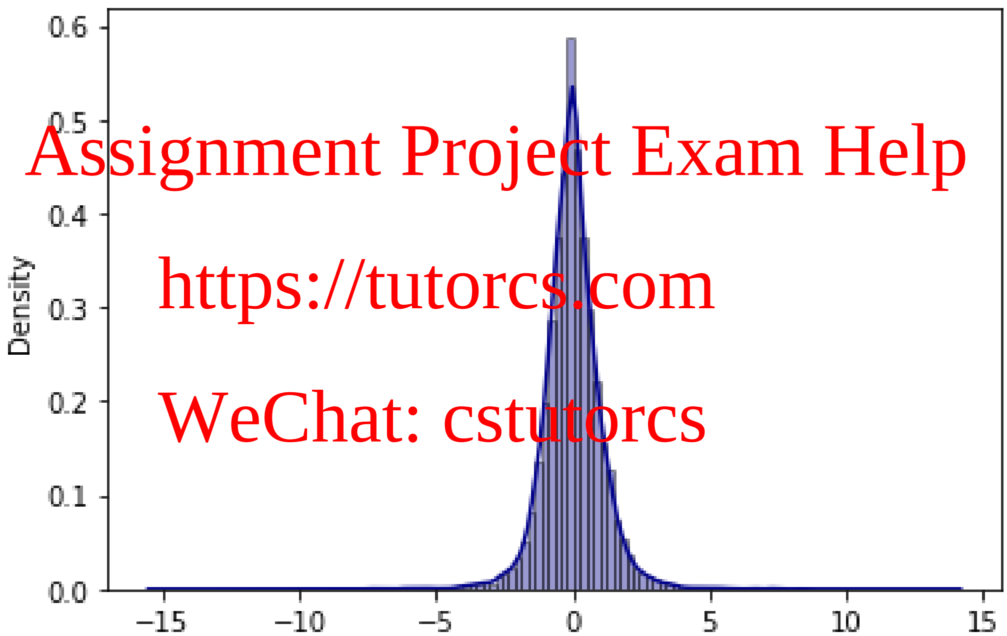
$\Rightarrow$ We can buy 1.1569 of market portfolio (i.e: S&P500 index fund) and then short 0.1569 T-Bill.

11 Residual Plots for GE

[111]:
```python
residuals = model_1.resid
import seaborn as sns
sns.distplot(residuals,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)
```

[111]:
```
<AxesSubplot:ylabel='Density'>
```



[112]:
```python
from scipy import stats
JB_GE= stats.jarque_bera(residuals)
JB_GE
```

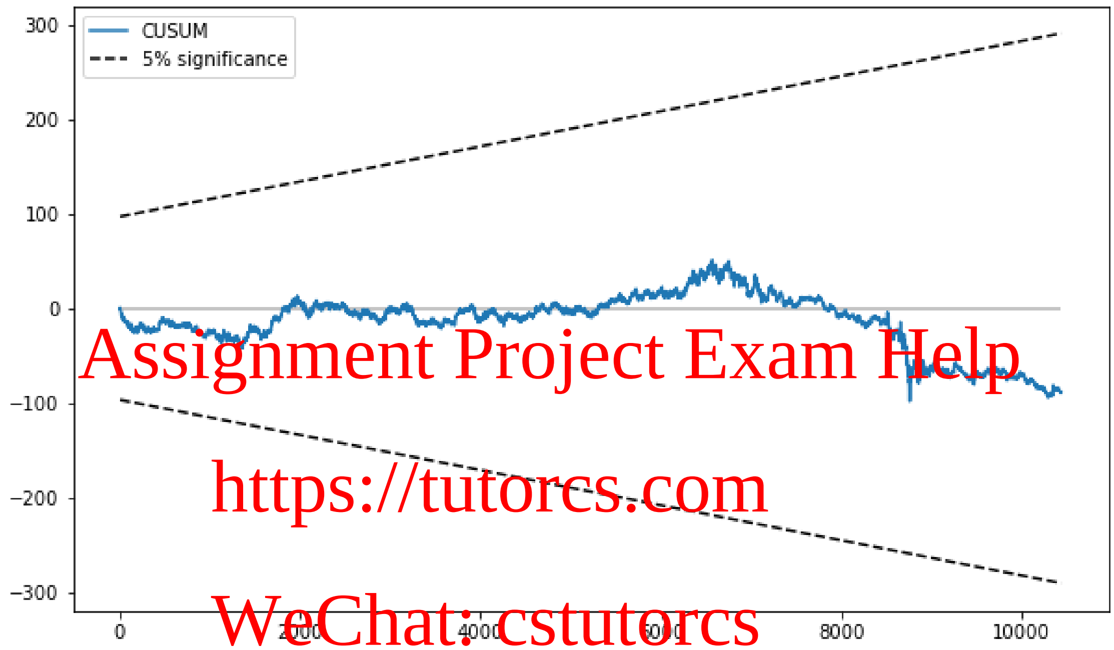
[112]:
```
Jarque_beraResult(statistic=109234.31887176927, pvalue=0.0)
```

The plot and JB test (p-value <0.05) rejects the null hypothesis of normality. It is clearly a non-normal distribution.

[113]:
```python
endog = data.R_ge
Rp = data.R_p
```

```
exog = sm.add_constant(Rp)
mod = sm.RecursiveLS(endog,exog)
res_1 = mod.fit()
fig = res_1.plot_cusum(figsize=(10,6));
```

```
C:\Users\rluck\anaconda3\lib\site-
packages\statsmodels\tsa\base\tsa_model.py:578: ValueWarning: An unsupported
index was provided and will be ignored when e.g. forecasting.
  warnings.warn('An unsupported index was provided and will be'
```



Cusum test of stability for GE shows stability of beta as it is within the 5% significance level band.

12 White Test of Heteroskedasticity for GE

[114]:
```
expr = 'y ~ X'
y, X = dmatrices(expr, data, return_type='dataframe')
olsr_results = smf.ols(expr, data).fit()
keys = ['Lagrange Multiplier statistic:', 'LM test\'s p-value:', 'F-statistic:
 ↪', 'F-test\'s p-value:']
results = het_white(olsr_results.resid, X)
lzip(keys, results)
```

[114]:
```
[('Lagrange Multiplier statistic:', 600.7119211665138),
 ("LM test's p-value:", 3.606315445696676e-131),
 ('F-statistic:', 318.60924780728743),
```

```
("F-test's p-value:", 4.9073934718673876e-135)]
```

LM test statistic is 600.72 and the corresponding p-value is 0

F-stats = 318.61 and the corresponding p-value is 0

Since the p-value of the both LM and F-stats is less than 0.05, we reject the null hypothesis that there is no heteroskedasticity in the residuals. It infers that the heteroskedasticity exists and the standard errors need to be corrected.

[115]:
```
print (dg.acorr_breusch_godfrey(model_1, nlags= 2))
```

```
(5.174836714176367, 0.07521396525512013, 2.587709781212114, 0.07524031416320724)
```

T-statistic of Chi-squared = 5.1748 and the corresponding p-value = 0.075.

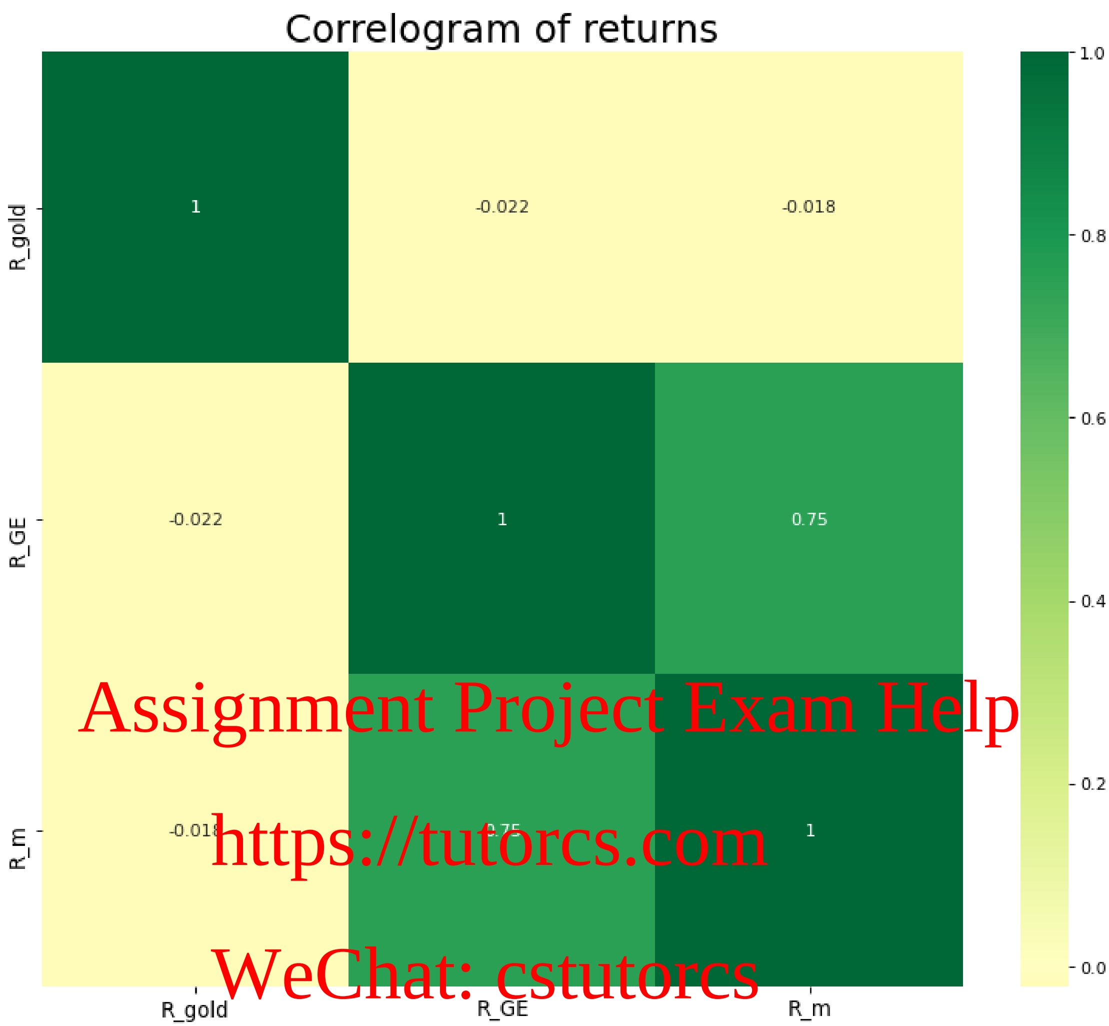
F-statistics = 2.5877 and the corresponding p-value = 0.075

Since p-value exceeds 0.05, we fail to reject the null hypothesis, thus inferring there is no autocorrelation at order less than or equal to 2.0

[ ]:
```

```


13 Extra: Correlation matrix between returns of gold, GE and market

[116]:
```
import seaborn as sns
import pandas as pd
# Import Dataset
data = pd.read_csv("C:\\Users\\iluck\\OneDrive\\capm1.csv",␣
 ↪usecols=['R_gold','R_m', 'R_GE'])

# Plot
plt.figure(figsize=(12,10), dpi= 80)
sns.heatmap(data.corr(), xticklabels=data.corr().columns, yticklabels=data.
 ↪corr().columns, cmap='RdYlGn', center=0, annot=True)

# Decorations
plt.title('Correlogram of returns', fontsize=22)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```


Correlogram of returns
R_gold
R_GE
R_m
1
-0.022
-0.018
-0.022
1
0.75
-0.018
0.75
1
1.0
0.8
0.6
0.4
0.2
0.0


```
[ ]:
```