EXERCISE 3

The data of `ESE06_ex3.csv` report the daily changes of the General Motors Co. closing prices since September 4, 1998 to November 27, 1998.

1. Design a suitable quality control tool by assuming the existence of an assignable cause for the OOC observations if any.
2. Determine if the values reported in the array `new_obs` are IC (use the previously designed control chart point 1). `new_obs = np.array((1.327, 1.594, 0.716, 1.767, 0.915, 2.524, 0.563, 2.053))`

In [ ]:
```
# Import the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats
import qda

# Import the dataset
data = pd.read_csv('ESE06_ex3.csv')

# Inspect the dataset
data.head()
```

Out[ ]:

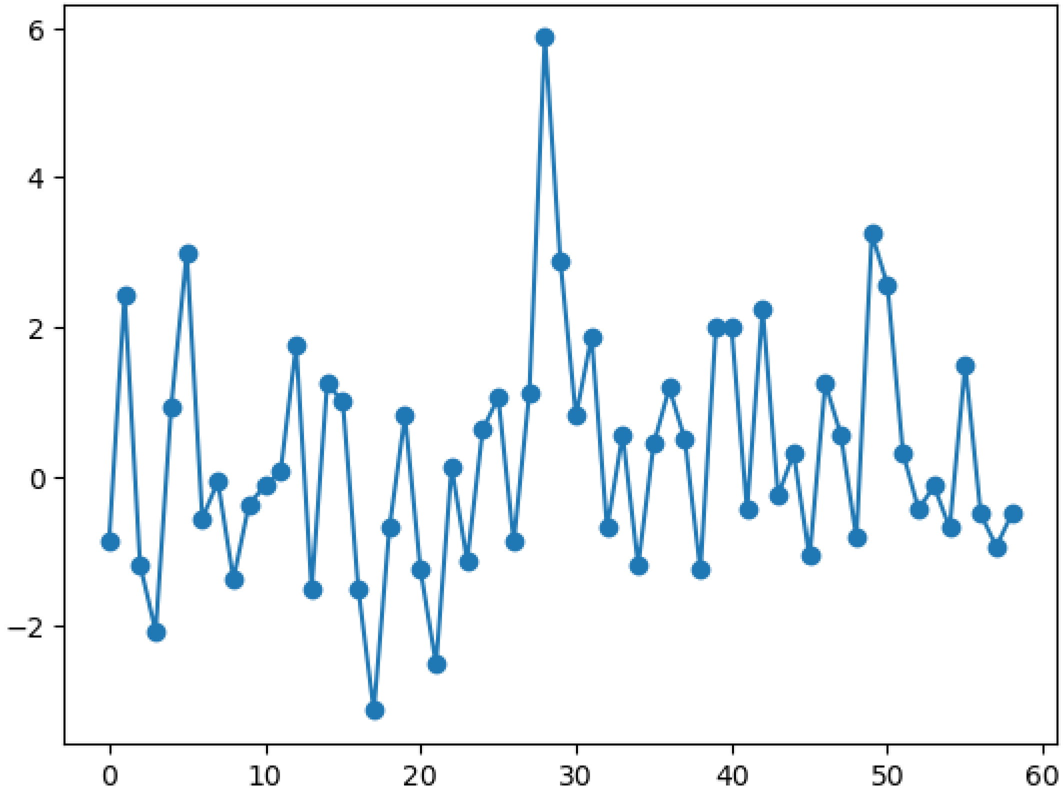
	GM
0	-0.875
1	2.437
2	-1.187
3	-2.063
4	0.938

Point 1

Design a suitable quality control tool (by using run rules too) by assuming the existence of an assignable cause for the OOC observations (if any).

> ### Solution
>
> Let's plot the data first.

In [ ]:
```
# Plot the data
plt.plot(data, 'o-')
plt.show()
```



> Looks like there's one point with a value much higher than the others. But let's test all assumptions first.

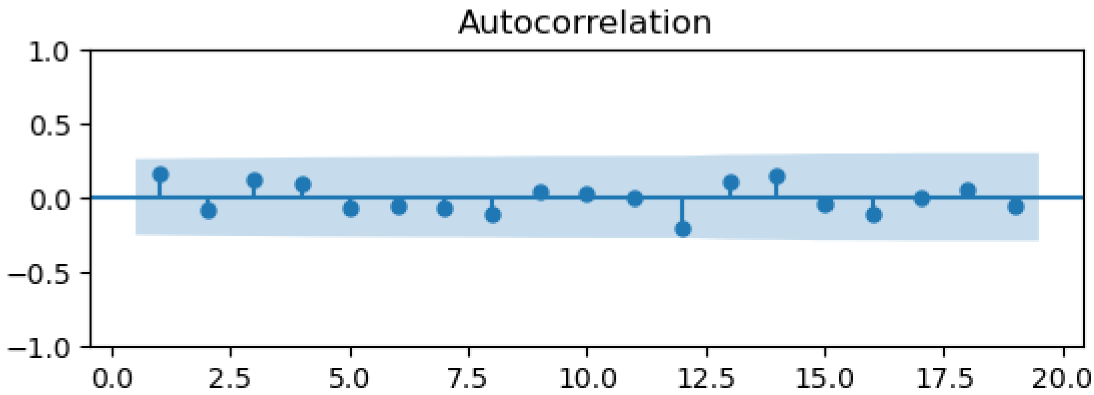
> Perform the runs test to check if the data are random. Use the `runstest_1samp` function from the `statsmodels` package.

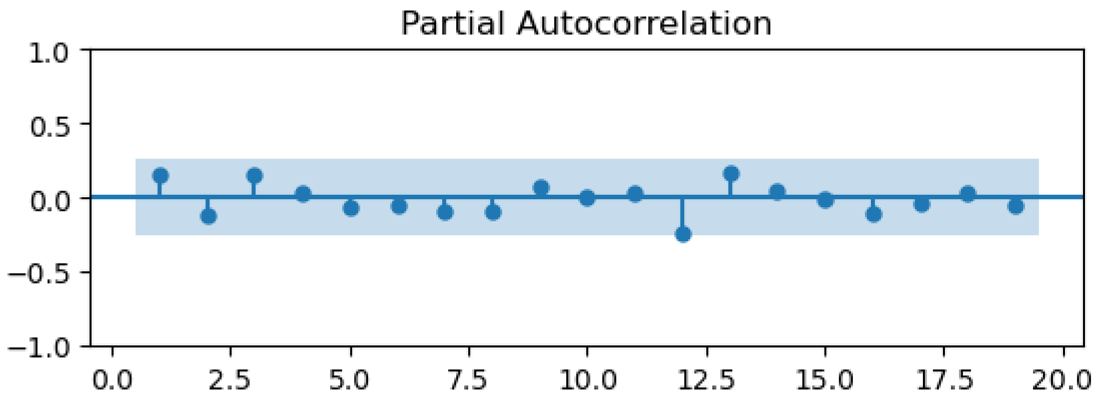
In [ ]:
```
# Verify if the data are random with runs test
# Import the necessary libraries for the runs test
from statsmodels.sandbox.stats.runs import runstest_1samp

_, pval_runs = runstest_1samp(data['GM'], correction=False)
print('Runs test p-value = {:.3f}'.format(pval_runs))
```

```
Runs test p-value = 0.879
```

> Plot the autocorrelation and partial autocorrelation functions of the data. Use the `plot_acf` and `plot_pacf` functions from the `statsmodels` package.

In [ ]:
```
# Plot the acf and pacf using the statsmodels library
import statsmodels.graphics.tsaplots as sgt

fig, ax = plt.subplots(2, 1)
sgt.plot_acf(data['GM'], lags = int(len(data)/3), zero=False, ax=ax[0])
fig.subplots_adjust(hspace=0.5)
sgt.plot_pacf(data['GM'], lags = int(len(data)/3), zero=False, ax=ax[1], method =
plt.show()
```





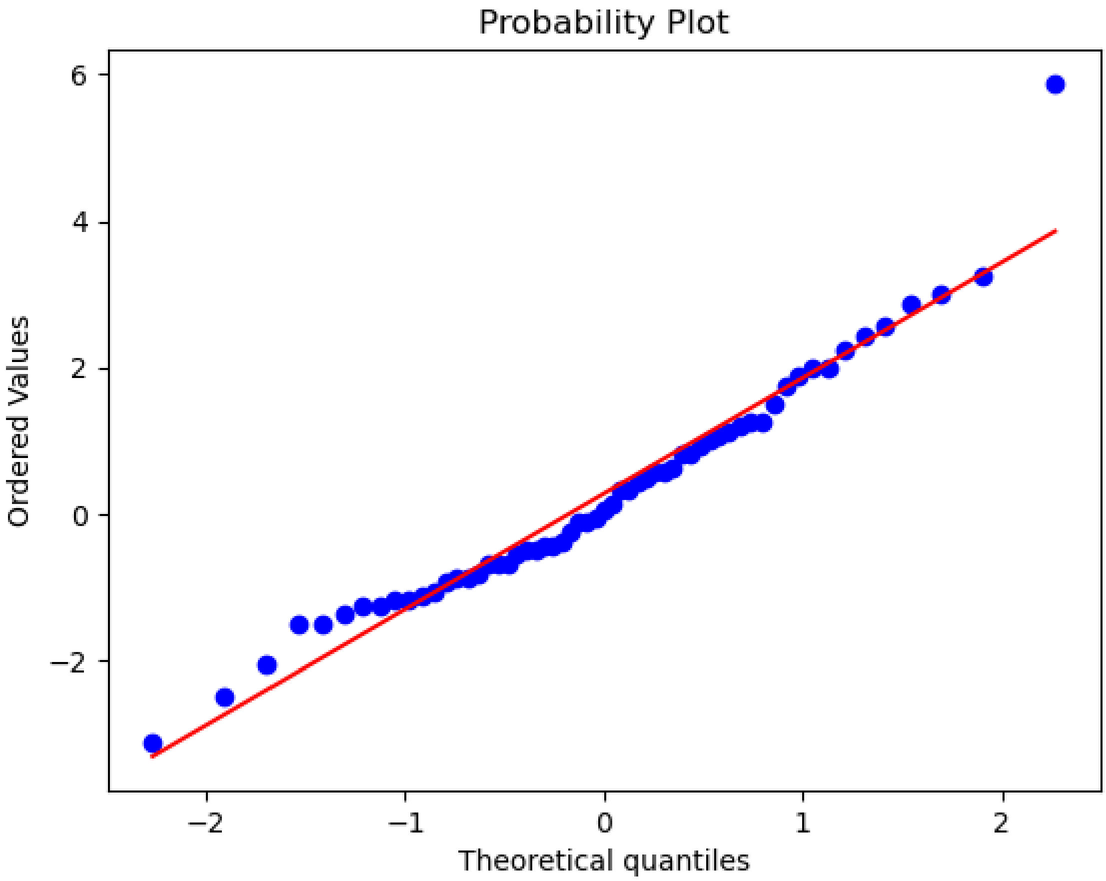
> From the results of the autocorrelation and the runs tests, there is no statistical evidence to assume non randomness of the process.

> Now let's verify the normality assumption.

In [ ]:

```
# Perform the Shapiro-Wilk test
_, pval_SW = stats.shapiro(data['GM'])
print('Shapiro-Wilk test p-value = %.3f' % pval_SW)

# Plot the qqplot
stats.probplot(data['GM'], dist="norm", plot=plt)
plt.show()
```

```
Shapiro-Wilk test p-value = 0.068
```



> We cannot reject the null hypothesis that the data are normally distributed with confidence 95%. However, one point deserves attention, as it is responsible for borderline normality.

> Let's go ahead with the design of the I-MR control chart.
>
> Remember, the computation of moving ranges MR:
>
> 1. Compute the differences between consecutive observations (lag = 1): $D_i = X_{i+1} - X_i$.
> 2. Compute the absolute values of the differences: $MR = |D_i|$.

In [ ]:
```
# Compute the moving ranges using the diff function
data['MR'] = data['GM'].diff().abs()

# Print out descriptive statistics of MR and time
data.describe()
```

Out[ ]:

	GM	MR
count	59.000000	58.000000
mean	0.275424	1.743603
std	1.581153	1.091674
min	-3.125000	0.000000
25%	-0.844000	0.749500
50%	0.062000	1.718500
75%	1.156500	2.484500
max	5.875000	4.750000

> Now let's make the control chart for the mean of the moving ranges.
>
> Remember the formulas for the control limits.
>
> **I chart**:
>
> - $UCL = \bar{x} + 3\left(\frac{\bar{MR}}{d_2}\right)$
> - $CL = \bar{x}$
> - $LCL = \bar{x} - 3\left(\frac{\bar{MR}}{d_2}\right)$
>
> **MR chart**:
>
> - $UCL = D_4\bar{MR}$
> - $CL = \bar{MR}$
> - $LCL = 0$

Factors for contructing variable control charts

Observations in Sample, n	Chart for Averages: Factors for Control Limits			Chart for Standard Deviations: Factors for Center Line		Factors for Control Limits				Chart for Ranges: Factors for Center Line		Factors for Control Limits				
	A	A_2	A_3	c_4	$1/c_4$	B_3	B_4	B_5	B_6	d_2	$1/d_2$	d_3	D_1	D_2	D_3	D_4
2	2.121	1.880	2.659	0.7979	1.2533	0	3.267	0	2.606	1.128	0.8865	0.853	0	3.686	0	3.267
3	1.732	1.023	1.954	0.8862	1.1284	0	2.568	0	2.276	1.693	0.5907	0.888	0	4.358	0	2.574
4	1.500	0.729	1.628	0.9213	1.0854	0	2.266	0	2.088	2.059	0.4857	0.880	0	4.698	0	2.282
5	1.342	0.577	1.427	0.9400	1.0638	0	2.089	0	1.964	2.326	0.4299	0.864	0	4.918	0	2.114
6	1.225	0.483	1.287	0.9515	1.0510	0.030	1.970	0.029	1.874	2.534	0.3946	0.848	0	5.078	0	2.004
7	1.134	0.419	1.182	0.9594	1.0423	0.118	1.882	0.113	1.806	2.704	0.3698	0.833	0.204	5.204	0.076	1.924
8	1.061	0.373	1.099	0.9650	1.0363	0.185	1.815	0.179	1.751	2.847	0.3512	0.820	0.388	5.306	0.136	1.864
9	1.000	0.337	1.032	0.9693	1.0317	0.239	1.761	0.232	1.707	2.970	0.3367	0.808	0.547	5.393	0.184	1.816
10	0.949	0.308	0.975	0.9727	1.0281	0.284	1.716	0.276	1.669	3.078	0.3249	0.797	0.687	5.469	0.223	1.777
11	0.905	0.285	0.927	0.9754	1.0252	0.321	1.679	0.313	1.637	3.173	0.3152	0.787	0.811	5.535	0.256	1.744
12	0.866	0.266	0.886	0.9776	1.0229	0.354	1.646	0.346	1.610	3.258	0.3069	0.778	0.922	5.594	0.283	1.717
13	0.832	0.249	0.850	0.9794	1.0210	0.382	1.618	0.374	1.585	3.336	0.2998	0.770	1.025	5.647	0.307	1.693
14	0.802	0.235	0.817	0.9810	1.0194	0.406	1.594	0.399	1.563	3.407	0.2935	0.763	1.118	5.696	0.328	1.672
15	0.775	0.223	0.789	0.9823	1.0180	0.428	1.572	0.421	1.544	3.472	0.2880	0.756	1.203	5.741	0.347	1.653
16	0.750	0.212	0.763	0.9835	1.0168	0.448	1.552	0.440	1.526	3.532	0.2831	0.750	1.282	5.782	0.363	1.637
17	0.728	0.203	0.739	0.9845	1.0157	0.466	1.534	0.458	1.511	3.588	0.2787	0.744	1.356	5.820	0.378	1.622
18	0.707	0.194	0.718	0.9854	1.0148	0.482	1.518	0.475	1.496	3.640	0.2747	0.739	1.424	5.856	0.391	1.608
19	0.688	0.187	0.698	0.9862	1.0140	0.497	1.503	0.490	1.483	3.689	0.2711	0.734	1.487	5.891	0.403	1.597
20	0.671	0.180	0.680	0.9869	1.0133	0.510	1.490	0.504	1.470	3.735	0.2677	0.729	1.549	5.921	0.415	1.585
21	0.655	0.173	0.663	0.9876	1.0126	0.523	1.477	0.516	1.459	3.778	0.2647	0.724	1.605	5.951	0.425	1.575
22	0.640	0.167	0.647	0.9882	1.0119	0.534	1.466	0.528	1.448	3.819	0.2618	0.720	1.659	5.979	0.434	1.566
23	0.626	0.162	0.633	0.9887	1.0114	0.545	1.455	0.539	1.438	3.858	0.2592	0.716	1.710	6.006	0.443	1.557
24	0.612	0.157	0.619	0.9892	1.0109	0.555	1.445	0.549	1.429	3.895	0.2567	0.712	1.759	6.031	0.451	1.548
25	0.600	0.153	0.606	0.9896	1.0105	0.565	1.435	0.559	1.420	3.931	0.2544	0.708	1.806	6.056	0.459	1.541

For $n > 25$.

In [ ]:

```
# Define the control limits
d2 = 1.128
```

```
D4 = 3.267

# make a copy of the data
df = data.copy()
# change the name of the column time to I
df.rename(columns={'GM':'I'}, inplace=True)

# Print the first 5 rows of the new dataframe
df.head()
```

Out[ ]:

	I	MR
0	-0.875	NaN
1	2.437	3.312
2	-1.187	3.624
3	-2.063	0.876
4	0.938	3.001

In [ ]:

```
# Create columns for the upper and lower control limits
df['I_UCL'] = df['I'].mean() + (3*df['MR'].mean()/d2)
df['I_CL'] = df['I'].mean()
df['I_LCL'] = df['I'].mean() - (3*df['MR'].mean()/d2)
df['MR_UCL'] = D4 * df['MR'].mean()
df['MR_CL'] = df['MR'].mean()
df['MR_LCL'] = 0

# Print the first 5 rows of the new dataframe
df.head()
```

Out[ ]:

	I	MR	I_UCL	I_CL	I_LCL	MR_UCL	MR_CL	MR_LCL
0	-0.875	NaN	4.912667	0.275424	-4.361819	5.696352	1.743603	0
1	2.437	3.312	4.912667	0.275424	-4.361819	5.696352	1.743603	0
2	-1.187	3.624	4.912667	0.275424	-4.361819	5.696352	1.743603	0
3	-2.063	0.876	4.912667	0.275424	-4.361819	5.696352	1.743603	0
4	0.938	3.001	4.912667	0.275424	-4.361819	5.696352	1.743603	0

In [ ]:

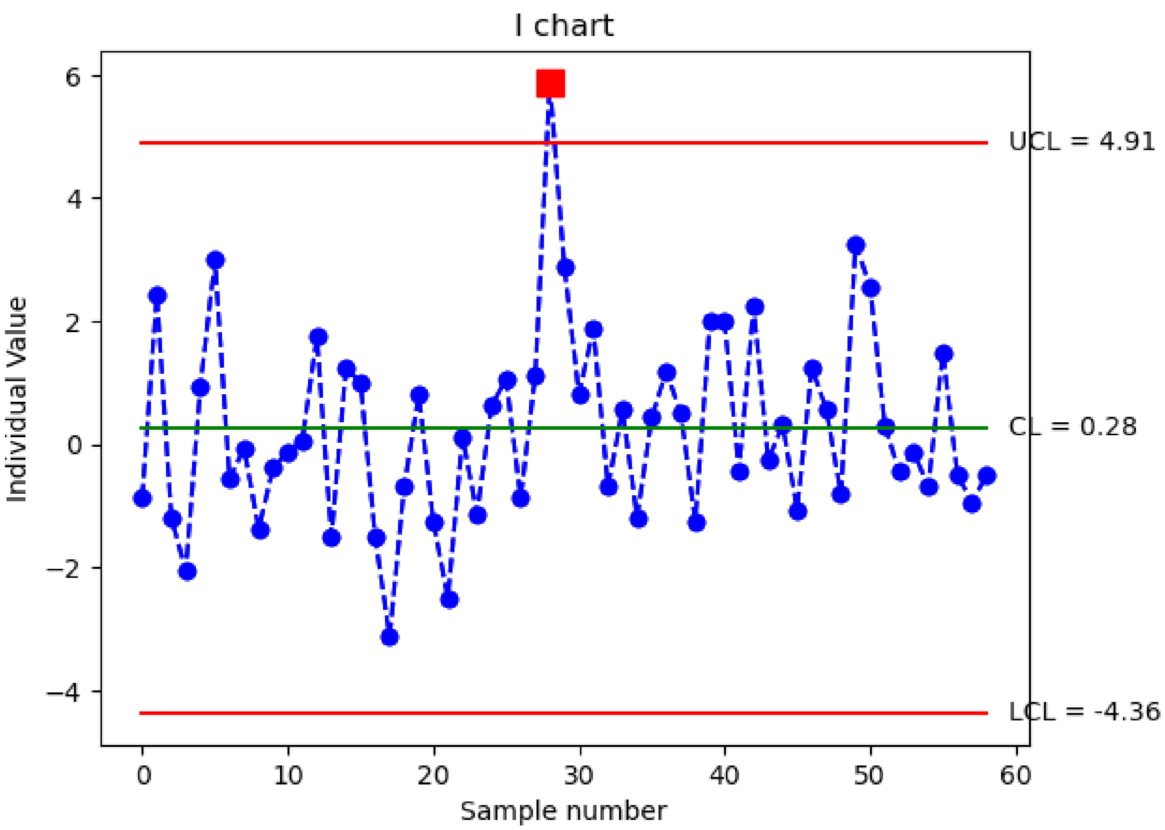
```
# Define columns for possible violations of the control limits
df['I_TEST1'] = np.where((df['I'] > df['I_UCL']) |
                (df['I'] < df['I_LCL']), df['I'], np.nan)
df['MR_TEST1'] = np.where((df['MR'] > df['MR_UCL']) |
                (df['MR'] < df['MR_LCL']), df['MR'], np.nan)

# Print the first 5 rows of the new dataframe
df.head()
```

Out[ ]:

	I	MR	I_UCL	I_CL	I_LCL	MR_UCL	MR_CL	MR_LCL	I_TEST1	MR_TEST1
0	-0.875	NaN	4.912667	0.275424	-4.361819	5.696352	1.743603	0	NaN	NaN
1	2.437	3.312	4.912667	0.275424	-4.361819	5.696352	1.743603	0	NaN	NaN
2	-1.187	3.624	4.912667	0.275424	-4.361819	5.696352	1.743603	0	NaN	NaN
3	-2.063	0.876	4.912667	0.275424	-4.361819	5.696352	1.743603	0	NaN	NaN
4	0.938	3.001	4.912667	0.275424	-4.361819	5.696352	1.743603	0	NaN	NaN

In [ ]:

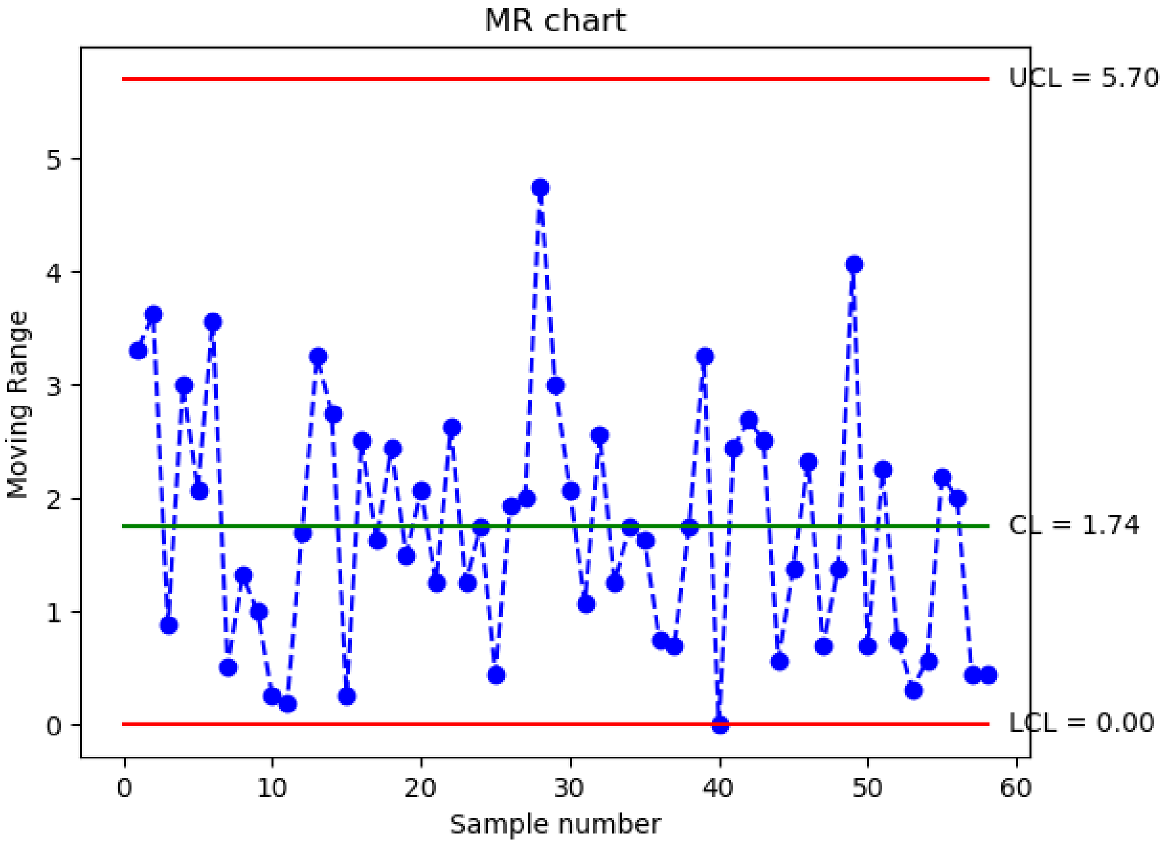
```
# Plot the I chart
plt.title('I chart')
plt.plot(df['I'], color='b', linestyle='--', marker='o')
plt.plot(df['I'], color='b', linestyle='--', marker='o')
plt.plot(df['I_UCL'], color='r')
plt.plot(df['I_CL'], color='g')
plt.plot(df['I_LCL'], color='r')
plt.ylabel('Individual Value')
plt.xlabel('Sample number')
# add the values of the control limits on the right side of the plot
plt.text(len(df)+.5, df['I_UCL'].iloc[0], 'UCL = {:.2f}'.format(df['I_UCL'].iloc[0
plt.text(len(df)+.5, df['I_CL'].iloc[0], 'CL = {:.2f}'.format(df['I_CL'].iloc[0]),
plt.text(len(df)+.5, df['I_LCL'].iloc[0], 'LCL = {:.2f}'.format(df['I_LCL'].iloc[0
# highlight the points that violate the alarm rules
plt.plot(df['I_TEST1'], linestyle='none', marker='s', color='r', markersize=10)
plt.show()
```



In [ ]:

```
plt.title('MR chart')
plt.plot(df['MR'], color='b', linestyle='--', marker='o')
plt.plot(df['MR_UCL'], color='r')
plt.plot(df['MR_CL'], color='g')
plt.plot(df['MR_LCL'], color='r')
plt.ylabel('Moving Range')
```

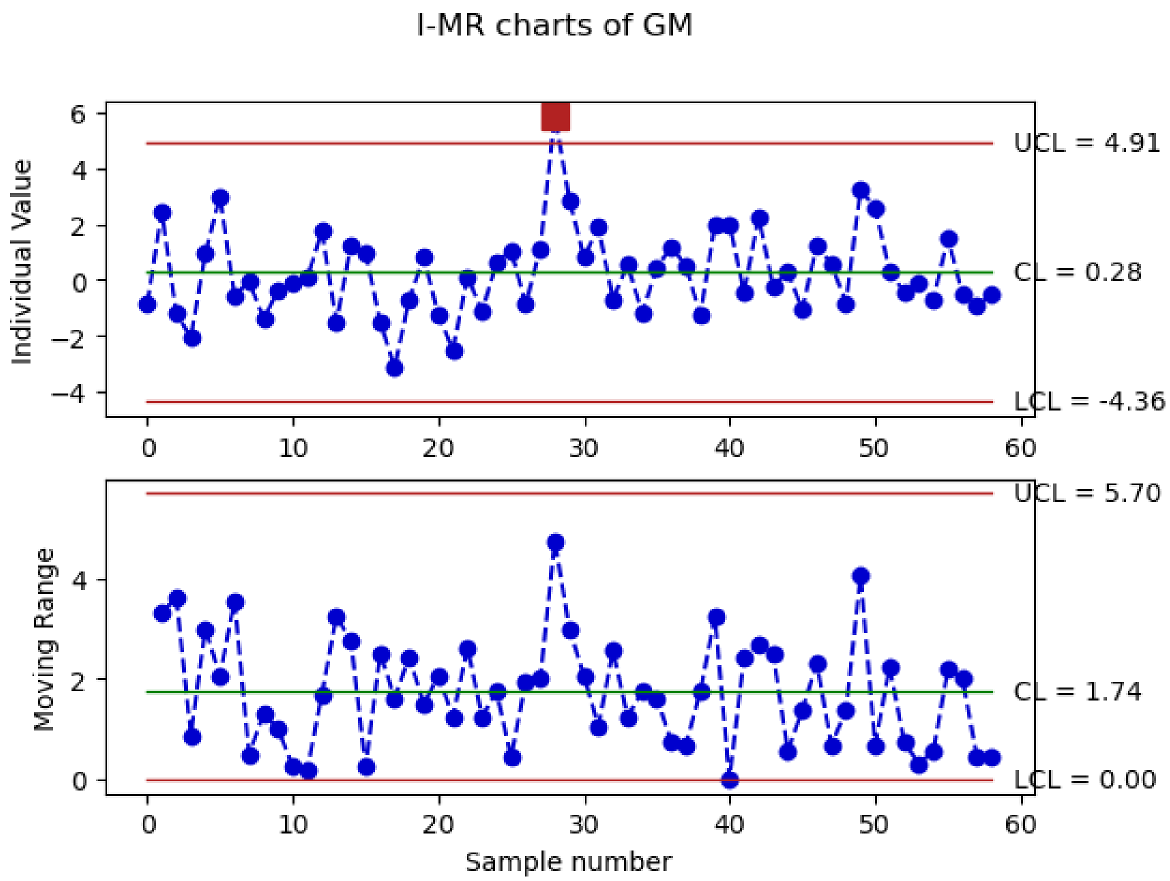
```
plt.xlabel('Sample number')
# add the values of the control limits on the right side of the plot
plt.text(len(df)+.5, df['MR_UCL'].iloc[0], 'UCL = {:.2f}'.format(df['MR_UCL'].iloc
plt.text(len(df)+.5, df['MR_CL'].iloc[0], 'CL = {:.2f}'.format(df['MR_CL'].iloc[0]
plt.text(len(df)+.5, df['MR_LCL'].iloc[0], 'LCL = {:.2f}'.format(df['MR_LCL'].iloc
# highlight the points that violate the alarm rules
plt.plot(df['MR_TEST1'], linestyle='none', marker='s', color='r', markersize=10)
plt.show()
```



> There is one point outside the control limits.

> You can also design the control chart using the `IMR` function in `qda.ControlCharts` package.

In [ ]:
```
data_IMR = qda.ControlCharts.IMR(data, 'GM')
```



> Let's find the index of the OOC point.

```
In [ ]: # Find the index of the I_TEST1 column different from NaN
        OOC_idx = np.where(data_IMR['I_TEST1'].notnull())[0]
        # Print the index of the OOC points
        print('The index of the OOC point is: {}'.format(OOC_idx))
```

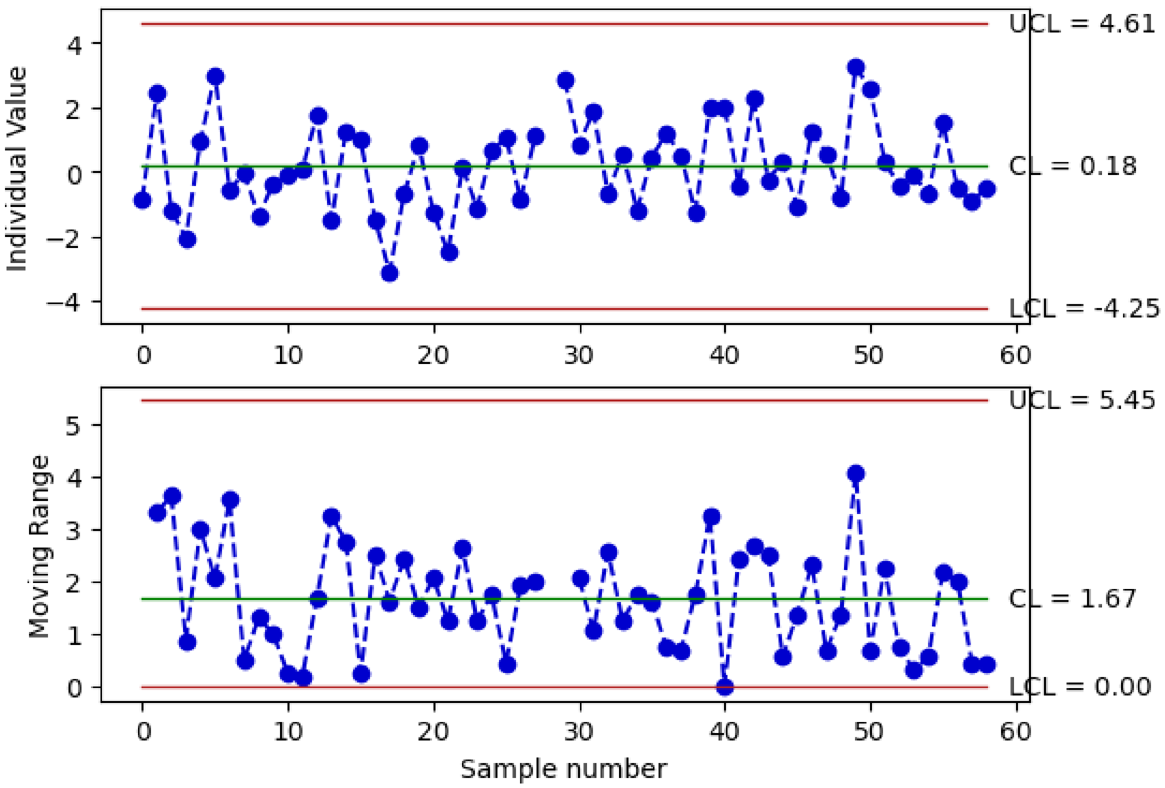
```
The index of the OOC point is: [28]
```

Index 28 (i.e. **the 29th** observation) is the one that is out of control.

> Assume we found an assignable cause for the OOC point, we have to remove it from the data.

```
In [ ]: # make a copy of the data
        data_2 = data.copy()
        # replace the OOC point with NaN
        data_2['GM'].iloc[OOC_idx] = np.nan

        # Plot the new control chart
        data_IMR_2 = qda.ControlCharts.IMR(data_2, 'GM')
```

I-MR charts of GM



Point 2

Determine if the values reported in the array `new_obs` are IC (use the previously designed control chart point 1).

In [ ]:
```python
# Import the dataset
data = pd.read_csv('ESE06_ex3.csv')
# replace the OOC point with NaN
data['GM'].iloc[OOC_idx] = np.nan

# Define the array with the new observations
new_obs = np.array((1.327, 1.594, 0.716, 1.767, 0.915, 2.524, 0.563, 2.053))

# Add the new observations to the dataset
new_data = data.append(pd.DataFrame(new_obs, columns=['GM']), ignore_index=True)

new_data.tail(10)
```

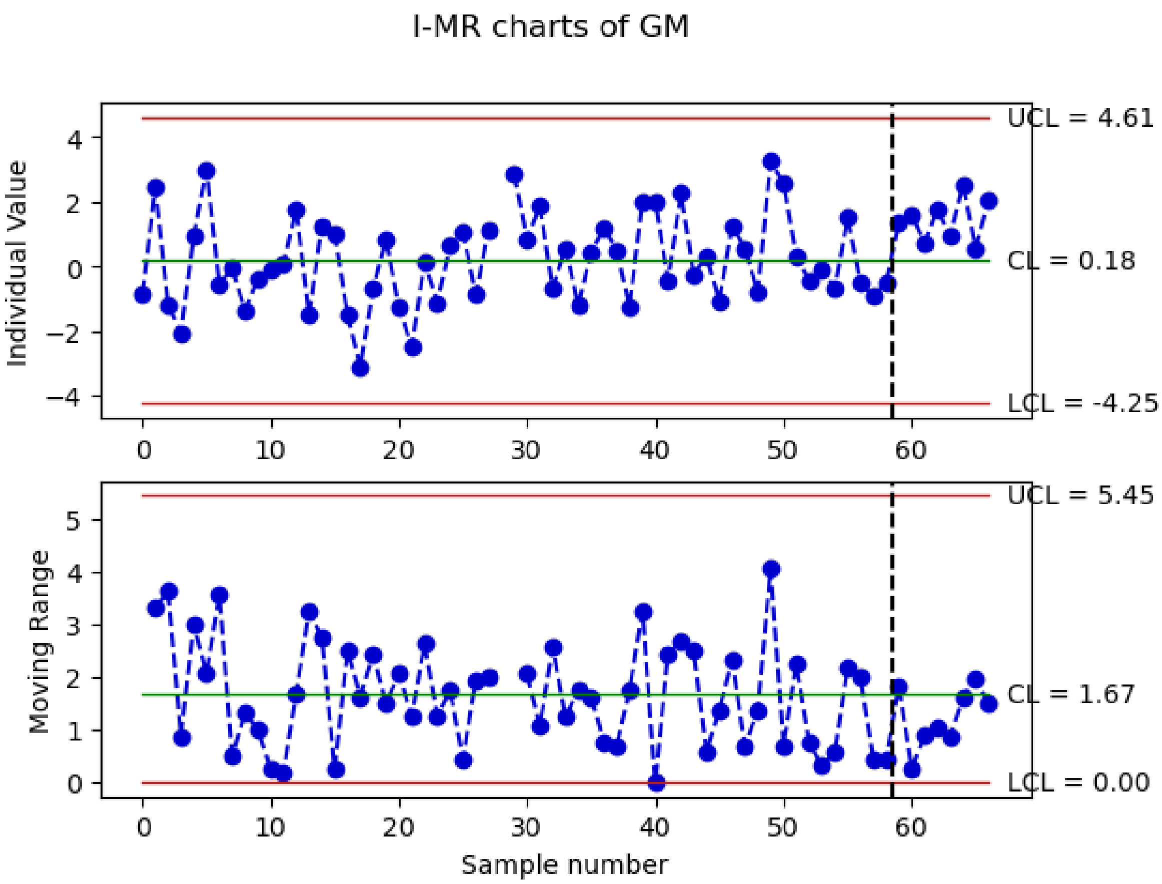
Out[ ]:

	GM
57	-0.937
58	-0.500
59	1.327
60	1.594
61	0.716
62	1.767
63	0.915
64	2.524
65	0.563
66	2.053

Use the parameter `subset_size` to specify the number of observations to be used for the computation of the control limits (Phase 1 data).

In [ ]:

```
new_data_IMR = qda.ControlCharts.IMR(new_data, 'GM', subset_size=len(data))
```



The new data (Phase II) are in-control