EXERCISE CLASS 8 - Multivariate SPC

(Chapter 11, Montgomery)

EXERCISE 1

A boatyard manager decided to measure the paint width (mm) and the corrosion speed (mm/year) to determine when the boat needs to be re-painted. They selected ten boats on which the measures of controlled parameters were performed in three predefined locations. The experiment lasted for two months and the relative collected data are in the file `ESE8_ex1.csv`.

Design an appropriate control chart and verify if the 9th sample is IC or OOC.

SOLUTION

> Let's start by importing the required libraries and loading the data.

In [ ]:

```
# Import the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats

# Import the dataset
data = pd.read_csv('ESE8_ex1.csv')

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

Out[ ]:

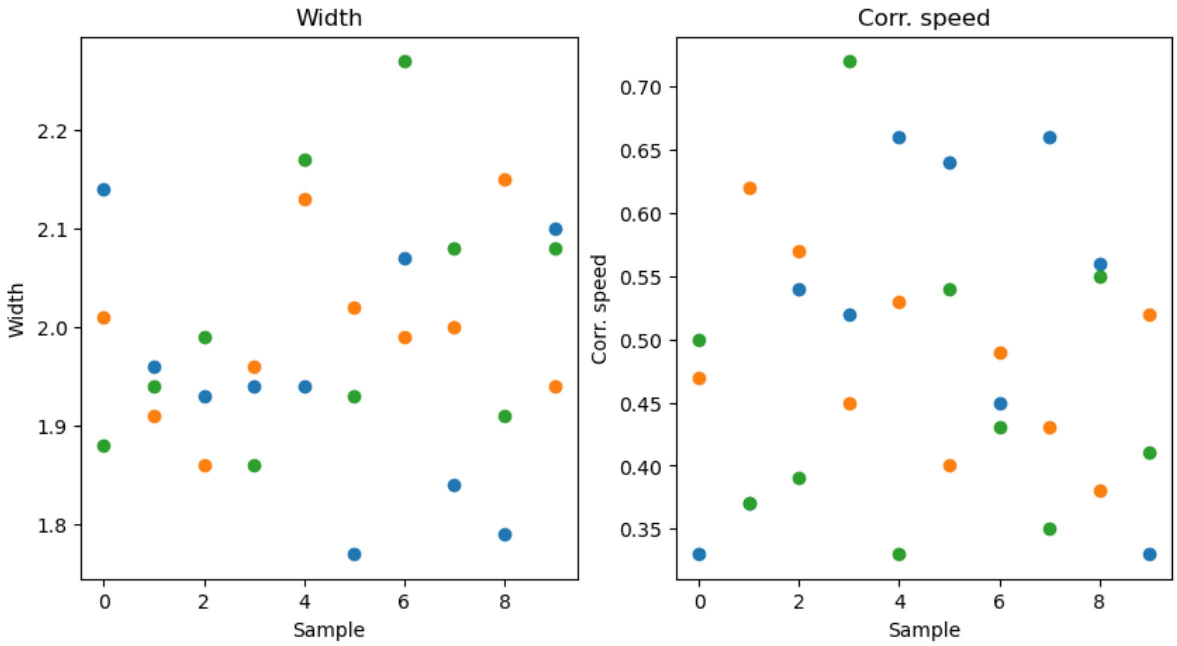
	Boat	Width1	Width2	Width3	Corr.speed1	Corr.speed2	Corr.speed3
0	1	2.14	2.01	1.88	0.33	0.47	0.50
1	2	1.96	1.91	1.94	0.37	0.62	0.37
2	3	1.93	1.86	1.99	0.54	0.57	0.39
3	4	1.94	1.96	1.86	0.52	0.45	0.72
4	5	1.94	2.13	2.17	0.66	0.53	0.33

> First of all we check the Marginal Normality of the stacked values of `Width` and `Corr.speed`

In [ ]:

```
# Extract the stacked array
Width = data[['Width1', 'Width2', 'Width3']]
Corr_speed = data[['Corr.speed1', 'Corr.speed2', 'Corr.speed3']]
```

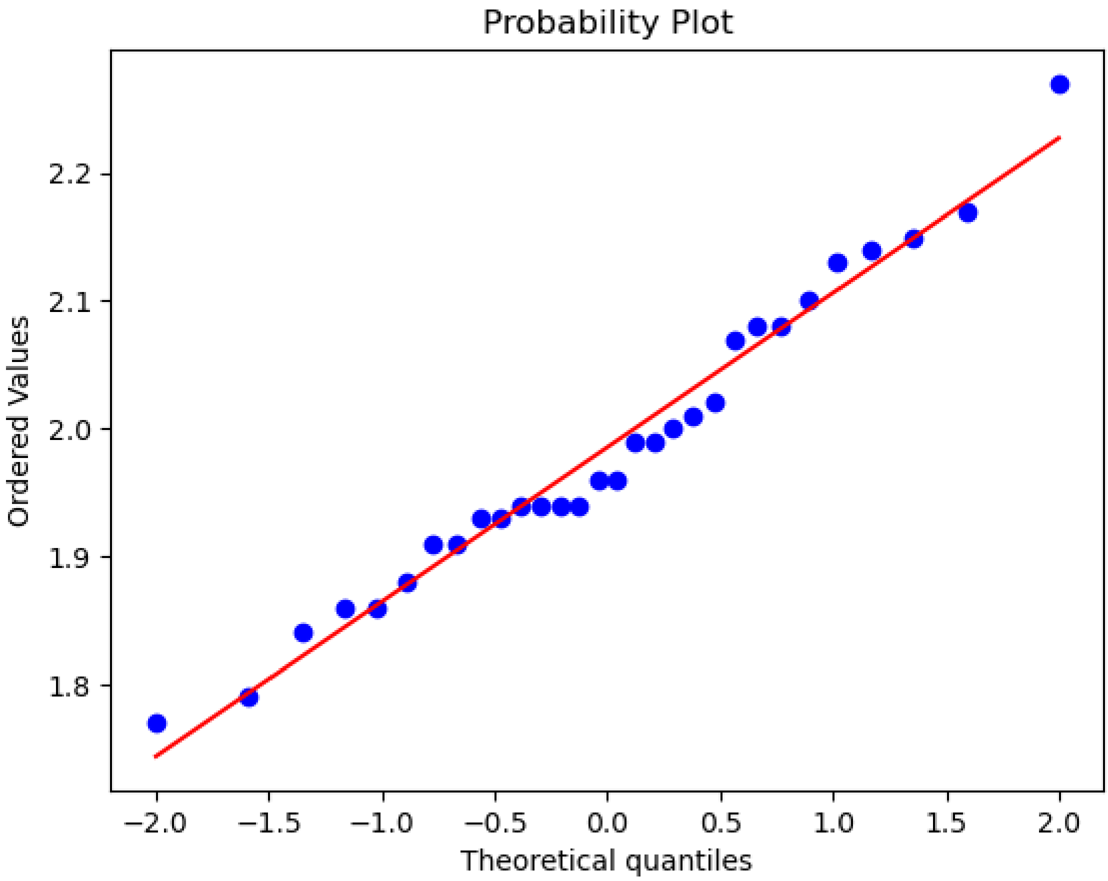
```
# Plot the data
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].plot(Width, 'o')
ax[0].set_xlabel('Sample')
ax[0].set_ylabel('Width')
ax[0].set_title('Width')
ax[1].plot(Corr_speed, 'o')
ax[1].set_xlabel('Sample')
ax[1].set_ylabel('Corr. speed')
ax[1].set_title('Corr. speed')
plt.show()
```



In [ ]:
```
# Perform the Shapiro-Wilk test on the Width
_, pval1_SW = stats.shapiro(Width.stack())
print('Shapiro-Wilk test on Width p-value = %.3f' % pval1_SW)

# Plot the qqplot
stats.probplot(Width.stack(), dist="norm", plot=plt)
plt.show()
```

```
Shapiro-Wilk test on Width p-value = 0.704
```

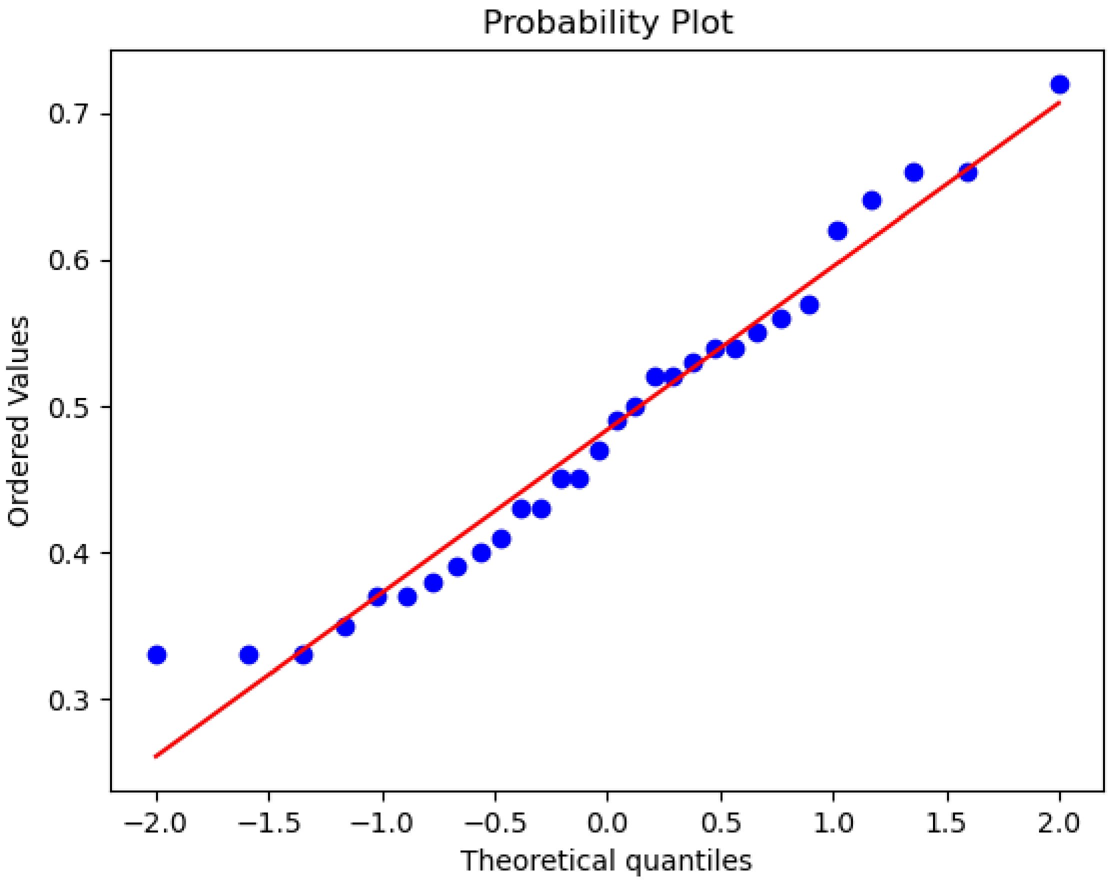


In [ ]:

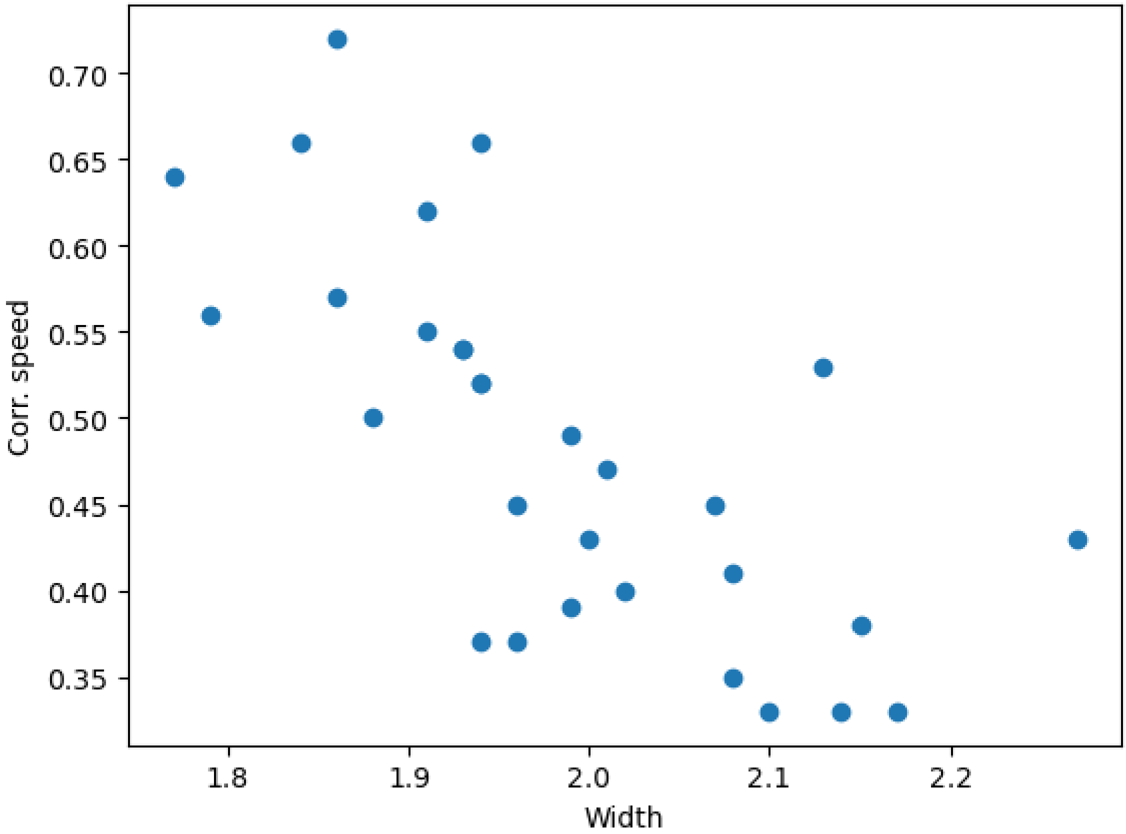
```
# Perform the Shapiro-Wilk test on the Corr.speed
_, pval2_SW = stats.shapiro(Corr_speed.stack())
print('Shapiro-Wilk test on Corr.speed p-value = %.3f' % pval2_SW)

# Plot the qqplot
stats.probplot(Corr_speed.stack(), dist="norm", plot=plt)
plt.show()
```

```
Shapiro-Wilk test on Corr.speed p-value = 0.249
```



In [ ]:
```
# Plot the data
plt.scatter(Width, Corr_speed)
plt.xlabel('Width')
plt.ylabel('Corr. speed')
plt.show()
```



> Remember the Hotelling's CC:

Control chart for the mean: Hotelling's T2

$$T_i^2 = n \cdot \left(\underline{\bar{X}}_i - \underline{\bar{\bar{X}}}\right)^T \cdot \underline{S}^{-1} \cdot \left(\underline{\bar{X}}_i - \underline{\bar{\bar{X}}}\right) \qquad i=1,\ldots,m$$

$$\underline{S} = \begin{pmatrix} \bar{S}_1 & \overline{S_{12}} \\ \overline{S_{12}} & \bar{S}_2 \end{pmatrix}$$

(here: p=2)

> Assuming that the data follow a binomial distribution, let's estimate the grand mean vector **X** and the variance/covariance matrix **S**

> Compute the grand mean vector, $\underline{\bar{\bar{\mathbf{X}}}}$.

In [ ]:
```
# Create a new dataframe to store the sample mean
sample_mean = pd.DataFrame()
sample_mean['width'] = Width.mean(axis=1)
sample_mean['corr_speed'] = Corr_speed.mean(axis=1)

# Calculate the grand mean
Xbarbar = sample_mean.mean()
print(Xbarbar)
```

```
width         1.985333
corr_speed    0.483667
dtype: float64
```

> Now compute the variance/covariance matrix, $\underline{\mathbf{S}}$.

In [ ]:
```
# Create a new dataframe to store the stacked data
data_stack = pd.DataFrame()
data_stack[['sample', 'width']] = Width.transpose().melt()
data_stack['corr_speed'] = Corr_speed.transpose().melt()['value']

data_stack.head(9)
```

Out[ ]:

	sample	width	corr_speed
0	0	2.14	0.33
1	0	2.01	0.47
2	0	1.88	0.50
3	1	1.96	0.37
4	1	1.91	0.62
5	1	1.94	0.37
6	2	1.93	0.54
7	2	1.86	0.57
8	2	1.99	0.39

In [ ]:

```python
# Compute the variance and covariance matrix of each group (sample)
cov_matrix = data_stack.groupby('sample').cov()

cov_matrix.head(8)
```

Out[ ]:

		width	corr_speed
sample			
0	**width**	0.016900	-0.011050
	corr_speed	-0.011050	0.008233
1	**width**	0.000633	-0.003333
	corr_speed	-0.003333	0.020833
2	**width**	0.004233	-0.005750
	corr_speed	-0.005750	0.009300
3	**width**	0.002800	-0.007400
	corr_speed	-0.007400	0.019633

In [ ]:

```python
# Compute the mean covariance matrix
S = cov_matrix.groupby(level=1).mean()

print(S)
```

```
               width  corr_speed
corr_speed -0.010970    0.014633
width       0.013263   -0.010970
```

> Attention! The indeces are now in alphabetic order. We need to reorder them in the order of the variables to get the correct variance/covariance matrix.

In [ ]:

```python
# Reorder the indeces of S to match the order of the columns
# get the names of the columns
cols = S.columns.tolist()

S = S.reindex(columns=cols, index=cols)

print(S)
```

```
              width  corr_speed
width        0.013263  -0.010970
corr_speed -0.010970   0.014633
```

p = number of random variables = 2

m = number of samples = 10

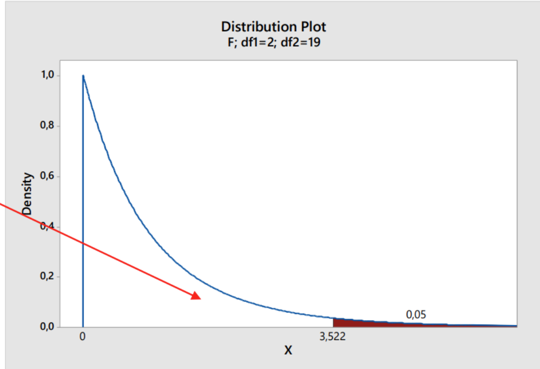
n = number of replicates (sample size) = 3

$\alpha = 0.05$ (this is our assumption; the Type I error is our choice)

$$T_i^2 = n \cdot \left(\bar{\underline{X}}_i - \bar{\bar{\underline{X}}}\right)^T \cdot \underline{S}^{-1} \cdot \left(\bar{\underline{X}}_i - \bar{\bar{\underline{X}}}\right) \approx \frac{p(m-1)(n-1)}{m(n-1)-(p-1)} F\left(p, mn-m+1-p\right)$$

$$UCL = \frac{p(m-1)(n-1)}{m(n-1)-(p-1)} F_\alpha\left(\underbrace{p}_{2}, \underbrace{mn-m+1-p}_{19}\right)$$

$$UCL = \frac{36}{19} F_{0.05}(2,19) = 6.67$$




```
In [ ]: p = 2              # number of random variables
        m = 10             # number of samples
        n = 3              # number of replicates (sample size)
        alpha = 0.05       # significance level

        UCL = (p * (m-1) * (n-1)) / (m * (n-1) - (p-1)) * stats.f.ppf(1-alpha, p, m*n - m 

        print('UCL = %.3f' % UCL)
```

```
UCL = 6.673
```

> Let's evaluate if the 9th sample is in control
>
> Remember that the sample is IC if
> T^2_i < UCL

```
In [ ]: # Calculate the Hotelling T2 statistic for the 9th sample
        index = 8
        S_inv = np.linalg.inv(S)
        T2 = n * (sample_mean.iloc[index]-Xbarbar).transpose().dot(S_inv).dot(sample_mean.

        print('\nThe Hotelling T2 statistic for the sample number %d is: %.3f' % (index + 
```

```
The Hotelling T2 statistic for the sample number 9 is: 0.424
```

> The 9th sample in therefore in control

> We can extend this analysis to the other samples.

```
In [ ]: # Calculate the Hotelling T2 statistic for all the samples
T2 = []
for i in range(m):
    T2.append(n * (sample_mean.iloc[i]-Xbarbar).transpose().dot(S_inv).dot(sample_

# Plot the Hotelling T2 statistic
plt.plot(T2, 'o-')
plt.plot([0, m], [UCL, UCL], 'r-')
plt.plot([0, m], [np.median(T2), np.median(T2)], 'g-')
plt.xlabel('Sample')
plt.ylabel('Hotelling T2')
plt.show()
```