Statistical Process Control (SPC)

SPC is a methodology used in quality control and process improvement to monitor, control, and analyze the variation and stability of a process. It involves using statistical techniques to understand and manage the process performance and ensure that it operates within specified quality limits.

The key objective of SPC is to identify and reduce process variation, which can lead to defects or deviations from desired outcomes. By monitoring and analyzing the data from the process, SPC helps organizations identify any patterns or trends that indicate a process is out of control or deviating from its normal performance.

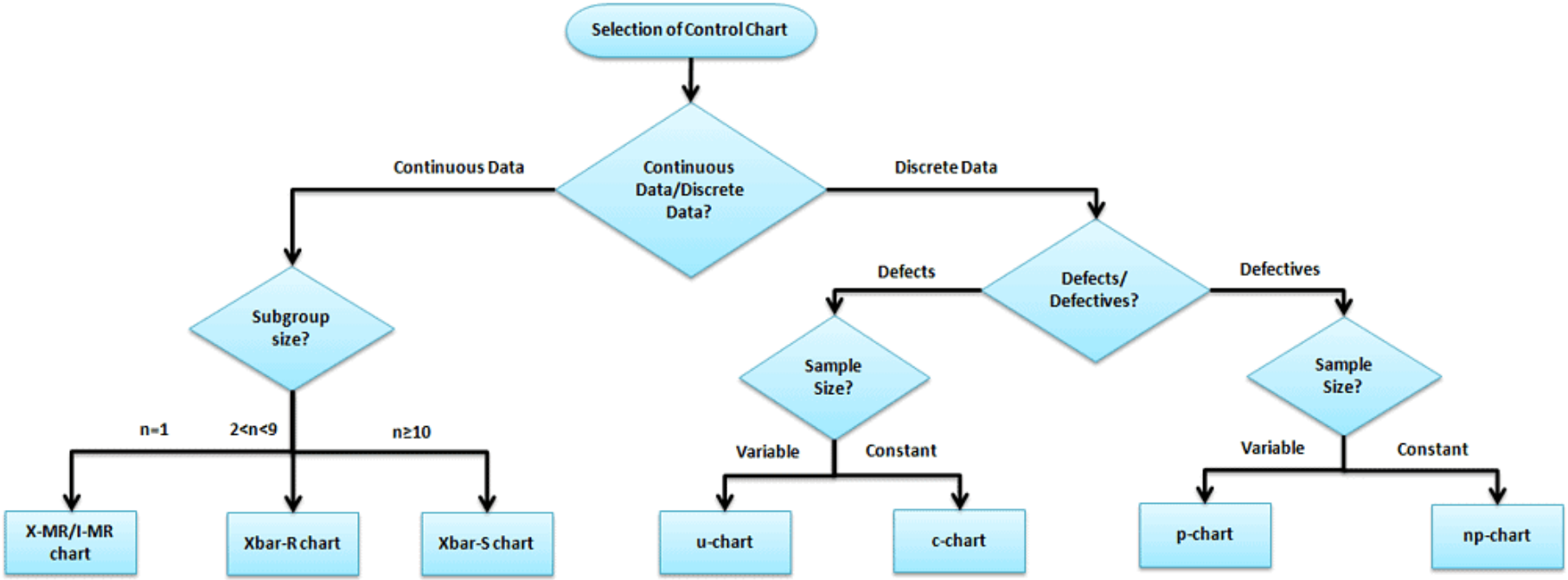
The main components of **Statistical Process Control:**

1. **Control Charts:** Control charts are the primary tool used in SPC. They provide a visual representation of process data over time, allowing practitioners to monitor the stability of a process. Control charts typically plot data points against control limits, which are calculated based on the inherent variation in the process.
2. **Process Variation:** SPC recognizes that every process has inherent variation due to various factors, including measurement error, material variation, and environmental factors. SPC aims to distinguish between common cause variation (inherent to the process) and special cause variation (due to assignable factors or exceptional events). By identifying and understanding special causes, the process can be improved and made more consistent.
3. **Data Collection and Analysis:** SPC relies on the collection and analysis of process data. Data is collected at regular intervals and plotted on control charts to determine if the process is in control or out of control. Data analysis techniques, such as calculating process capability indices (e.g., Cp and Cpk), are used to assess how well the process meets specifications and customer requirements.
4. **Process Improvement:** SPC provides insights into process performance and guides process improvement efforts. When a process is found to be out of control or exhibiting excessive variation, SPC helps identify the causes so that appropriate corrective or preventive actions can be taken. By reducing process variation, SPC aims to improve quality, increase efficiency, and reduce costs.

SPC is widely used in industries such as manufacturing, healthcare, finance, and service sectors to ensure consistent and reliable process performance. It enables organizations to make data-driven decisions, maintain quality standards, and continuously improve their processes. In fact, SPC can be applied to any process, as long as we're able to messure and collect the relevant process values.

The selection of an appropriate control chart is very important in control chart mapping. The main reason for this is statistical validity: Each control chart type is based on certain statistical assumptions and models. For example, an Individual-Moving Range (I-MR) chart is appropriate for individual

measurements over time, while an X-bar and R chart is used when multiple data points are collected at the same time.



Reference : sixsigmastudyguide.com

To identify special cause variation the following control chart zones are identified:

- **Center line:** corresponds to the mean of the measure being analyzed
- **Zone A:** corresponds to the zone from two sigma to three sigma above/below the center line
- **Zone B:** corresponds to the zone from one sigma to two sigma above/below the center line
- **Zone C:** corresponds to the area from the center line to one sigma above/below the center line
- **UCL (Upper Control Limit):** corresponds to the maximum tolerance above the mean
- **LCL (Lower Control Limit):** corresponds to the maximum tolerance below the mean

The following rules are available:

#1 Beyond limits: one or more points are beyond the control limits

#2 Zone A: 2 out of 3 consecutive points in Zone A or beyond

#3 Zone B: 4 out of 5 consecutive points in Zone B or beyond

#4 Zone C: 7 or more consecutive points on one side of the average (in Zone C or beyond)

#5 Trend: 7 consecutive points trending up or trending down

#6 Mixture: 8 consecutive points with no points in Zone C

#7 Stratification: 15 consecutive points in Zone C

#8 Over-control: 14 consecutive points alternating up and down

The following functionality is currently provided under the current distribution:

- X-MR control chart
- I-MR control chart
- Xbar-R control chart
- Xbar-S control chart
- u control chart
- c control chart
- p control chart
- np control chart
- multi vari chart
- process capability chart and statistics
- 8 possible rules to identify special cause variation
- recognize stages automatically or set stages manually
- test for a normal distribution for the control charts with continuous data
- show control chart zones optional
- all control charts support calculated data extract (to visualize in a dashboard)
- stable indicator for all control charts
- capable indicator for the process capability chart

The Python package can be downloaded at : https://pypi.org/project/sixsigmaspc/

Authors : Junior Marte Garcia and Marcel van Velzen

Publication date: 30 May 2023

Import statements

In [52]:
```
import numpy as np
from SPC import Rule01
from SPC import Rule02
from SPC import Rule03
from SPC import Rule04
from SPC import Rule05
from SPC import Rule06
from SPC import Rule07
from SPC import Rule08
from SPC import XmRControlChart
from SPC import ImRControlChart
from SPC import XbarRControlChart
from SPC import XbarSControlChart
from SPC import UControlChart
from SPC import CControlChart
from SPC import PControlChart
from SPC import NPControlChart
from SPC import MultiVariChart
from SPC import ProcessCapabilityChart
```

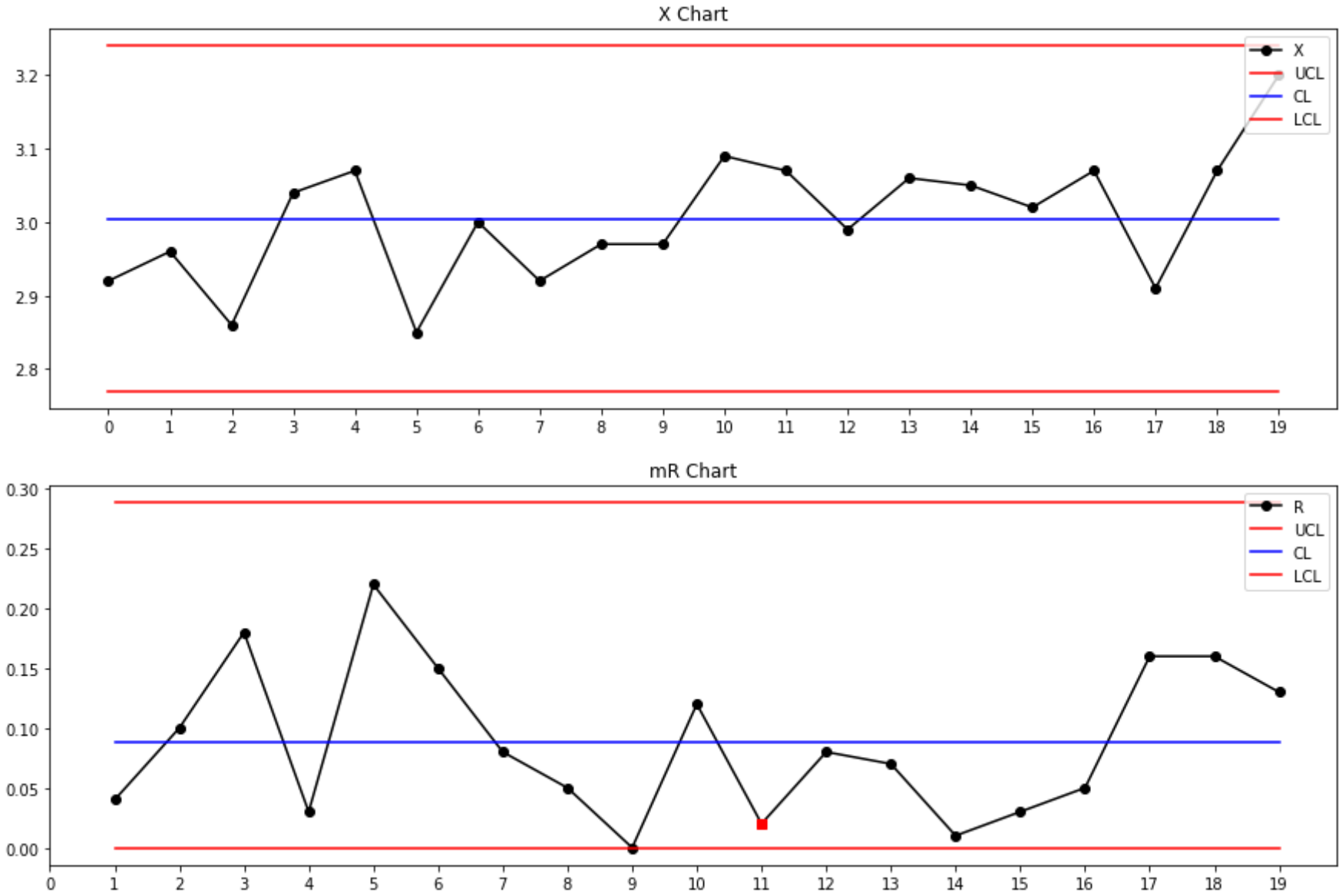
X-MR chart

An **individuals and moving range (X-MR) chart** is a pair of control charts for processes with a subgroup size of one. Used to determine if a process is stable and predictable, it creates a picture of how the system changes over time. The individual (X) chart displays individual measurements.

Reference: https://sixsigmastudyguide.com/xmr-charts/

In [64]:
```
data = np.array([2.92, 2.96, 2.86, 3.04, 3.07, 2.85, 3.00, 2.92, 2.97, 2.97, 3.09, 3.07, 2.99, 3.06, 3.05, 3.02, 3.07, 2.91, 3.07, 3.
chart = XmRControlChart(data=data)
chart.limits=False # True displays chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
normally_distributed=chart.normally_distributed(data=chart.value_X, significance_level=0.05)
```

```
print("normally_distributed={0}".format(normally_distributed))
print("stable={0}".format(chart.stable()))
```



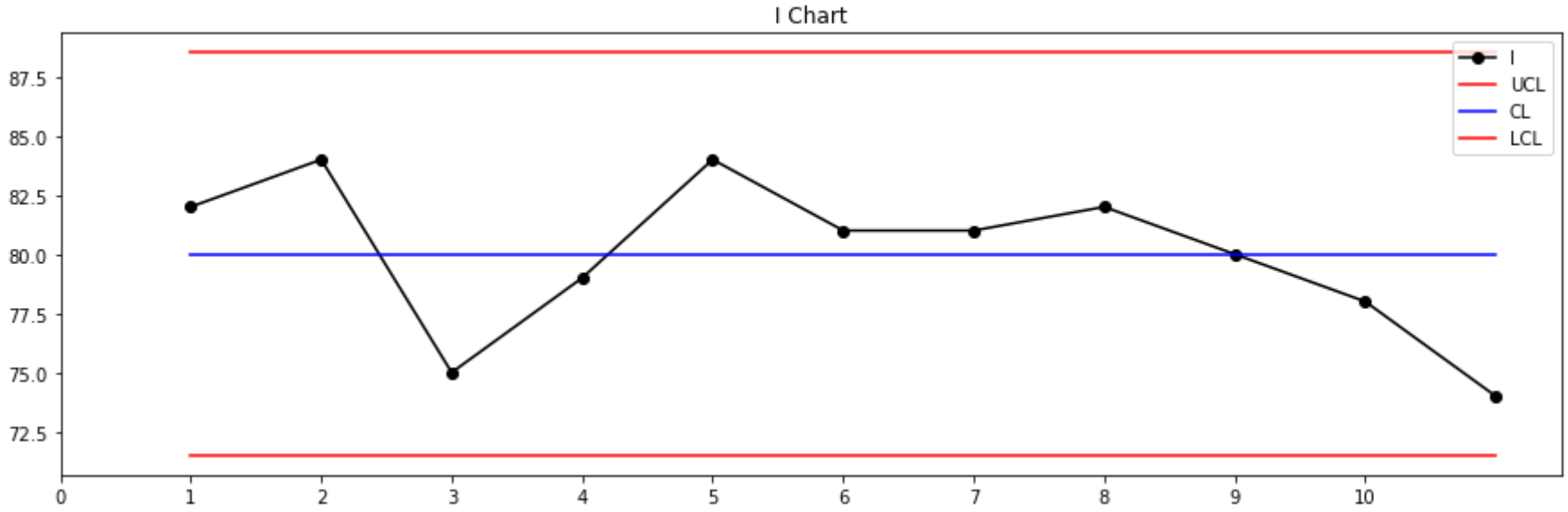

```
normally_distributed=True
stable=False
```

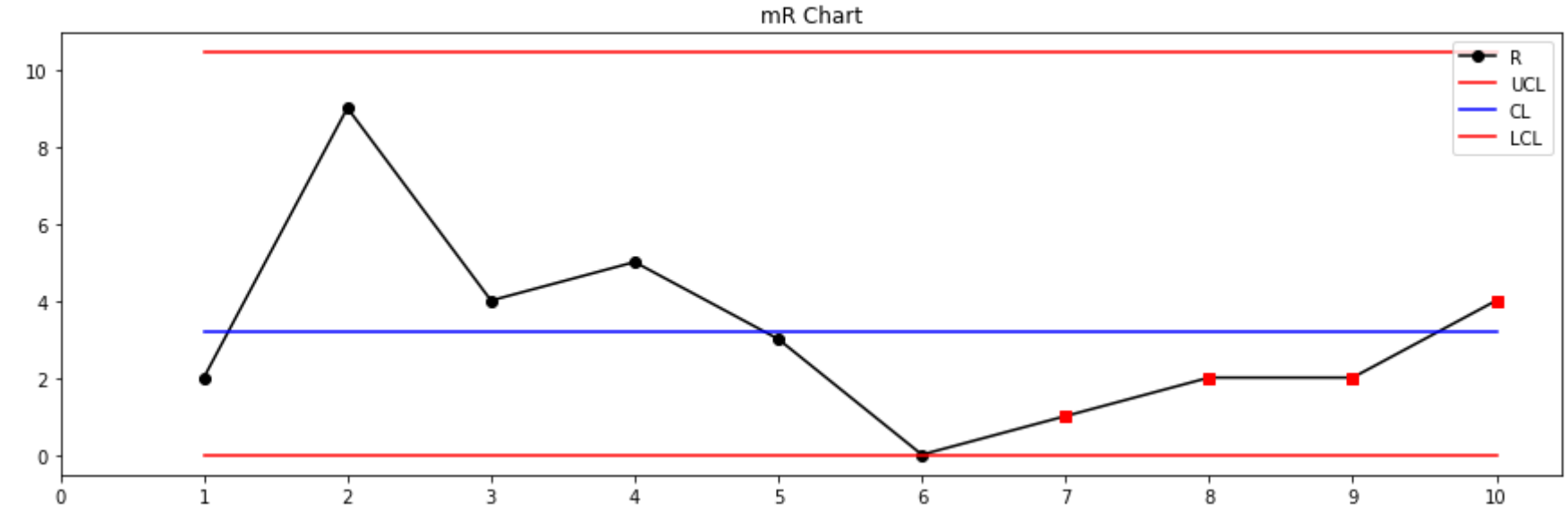
I-MR chart

An **Individual moving range (I-MR) chart** is used when data is continuous and not collected in subgroups. An I-MR chart provides process variation over time in graphical method. I-MR chart is basically two separate charts – Individuals (I) chart and Moving Range (MR) chart, the combination of two charts provides the complete picture of process behavior.

I-Chart: Individual chart displays the individual data points and monitors mean and shifts in the process when the data points collected at regular intervals of time. This chart will help to identify the common and assignable causes in the process, if any. **MR Chart:** While Individual chart monitors the process mean, the Moving Range chart monitors the process variation when the data points collected at regular intervals of time. In other words the moving range chart tracks the absolute difference of each measurement to its previous measurement. **Reference:** **https://sixsigmastudyguide.com/i-mr-chart/**

In [54]:
```
data = np.array([82, 84, 75, 79, 84, 81, 81, 82, 80, 78, 74])
chart = ImRControlChart(data=data)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
normally_distributed=chart.normally_distributed(data=chart.value_I, significance_level=0.05)
print("normally_distributed={0}".format(normally_distributed))
print("stable={0}".format(chart.stable()))
```






```
normally_distributed=True
stable=False
```

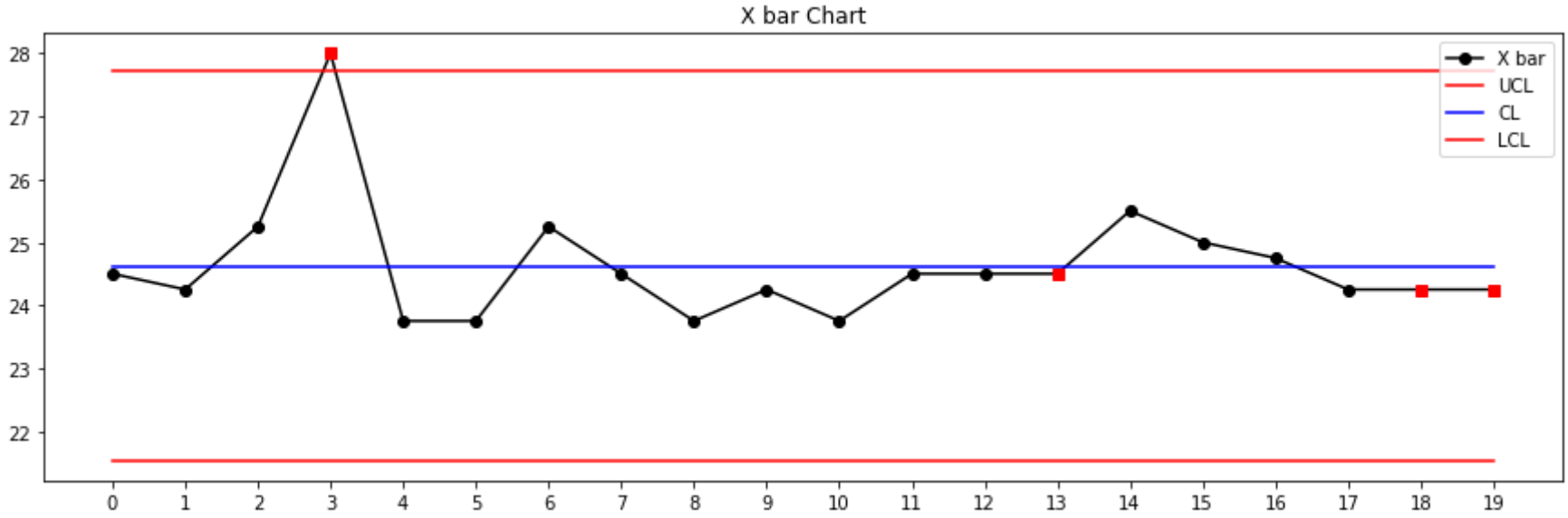
Xbar-R chart

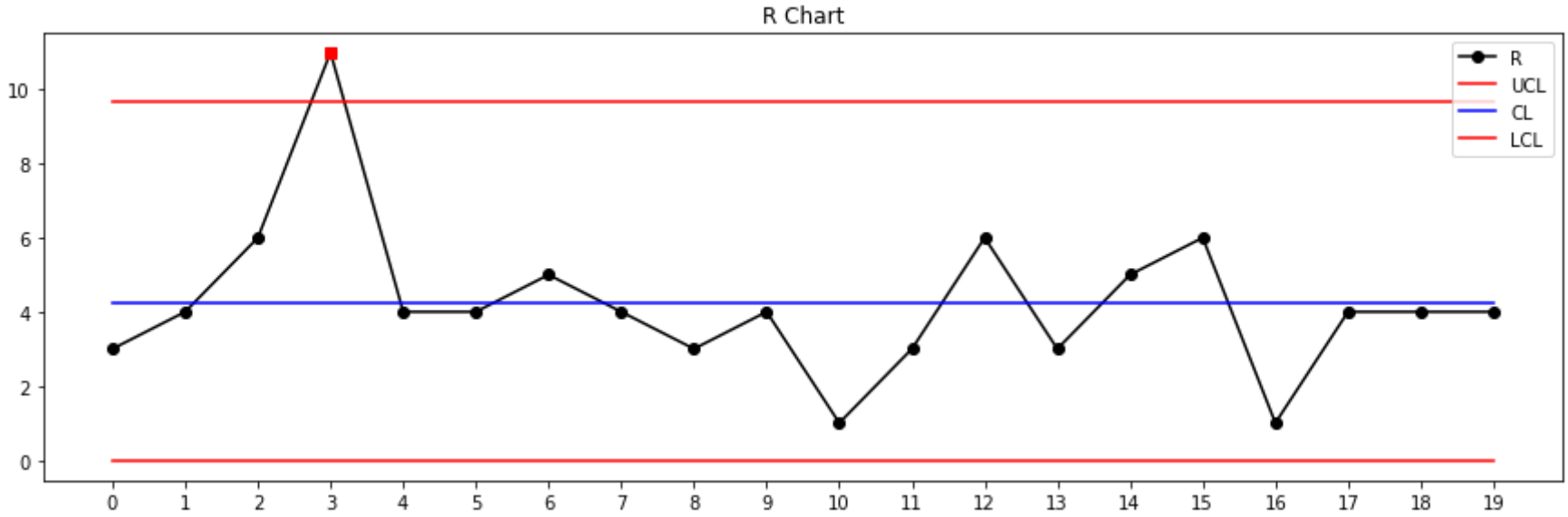
The **x-bar and R-chart** are control charts used to monitor the mean and variation of a process based on samples taken in a given time.

X-bar chart: The mean or average change in process over time from subgroup values. The control limits on the X-Bar brings the sample's mean and center into consideration. **R-chart:** The range of the process over the time from subgroups values. This monitors the spread of the process over the time. **Reference: https://sixsigmastudyguide.com/x-bar-r-control-charts/**

In [55]:

```
data = np.array([[23, 25, 24, 26],
                 [22, 26, 24, 25],
                 [28, 28, 22, 23],
                 [25, 25, 26, 36],
                 [22, 22, 25, 26],
                 [26, 24, 23, 22],
                 [29, 24, 24, 24],
                 [26, 25, 25, 22],
                 [22, 25, 24, 24],
                 [25, 22, 26, 24],
                 [24, 24, 24, 23],
                 [24, 25, 26, 23],
                 [22, 28, 22, 26],
                 [23, 24, 25, 26],
                 [24, 25, 29, 24],
                 [24, 22, 28, 26],
                 [24, 25, 25, 25],
                 [22, 24, 25, 26],
                 [26, 25, 22, 24],
                 [26, 22, 24, 25]])
chart = XbarRControlChart(data=data)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
normally_distributed=chart.normally_distributed(data=chart.value_X, significance_level=0.05)
print("normally_distributed={0}".format(normally_distributed))
print("stable={0}".format(chart.stable()))
```



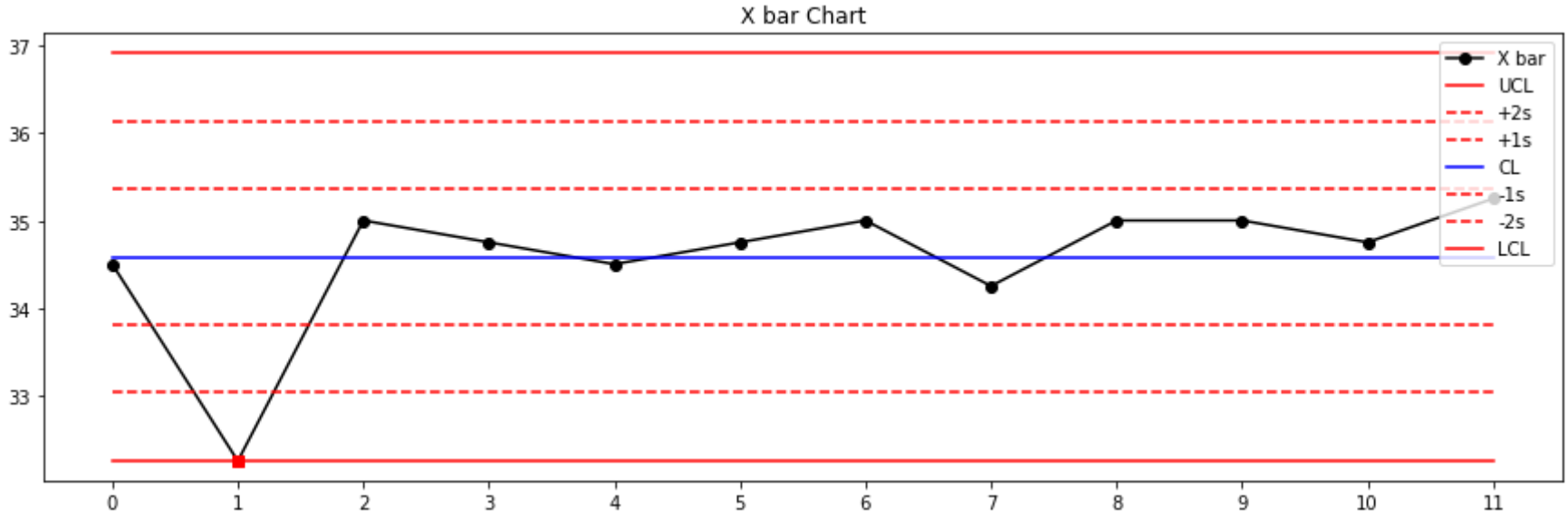


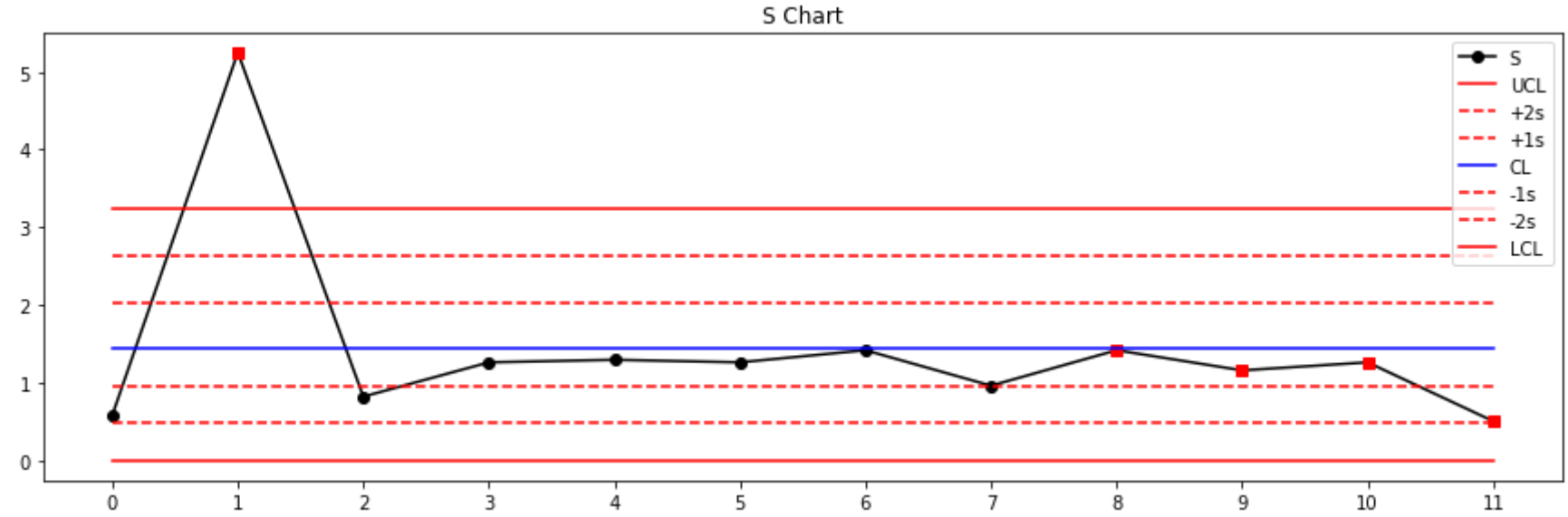

```
normally_distributed=False
stable=False
```

Xbar-S chart

X Bar S charts are control charts to examine the process mean and standard deviation over the time. **X-bar chart:** The mean or average change in process over time from subgroup values. The control limits on the X-Bar brings the sample's mean and center into consideration. **S-chart:** The standard deviation of the process over the time from subgroups values. This monitors the process standard deviation (as approximated by the sample moving range). **Reference: https://sixsigmastudyguide.com/x-bar-s-chart/**

In [56]:
```
data = np.array([[34,35,35,34],
                 [37,35,25,32],
                 [36,35,35,34],
                 [33,36,35,35],
                 [36,33,34,35],
                 [33,36,35,35],
                 [36,36,35,33],
                 [33,35,35,34],
                 [36,36,35,33],
                 [34,36,36,34],
                 [36,35,33,35],
                 [35,35,35,36]])
chart = XbarSControlChart(data=data)
chart.limits=True # Do display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
normally_distributed=chart.normally_distributed(data=chart.value_X, significance_level=0.05)
print("normally_distributed={0}".format(normally_distributed))
print("stable={0}".format(chart.stable()))
```





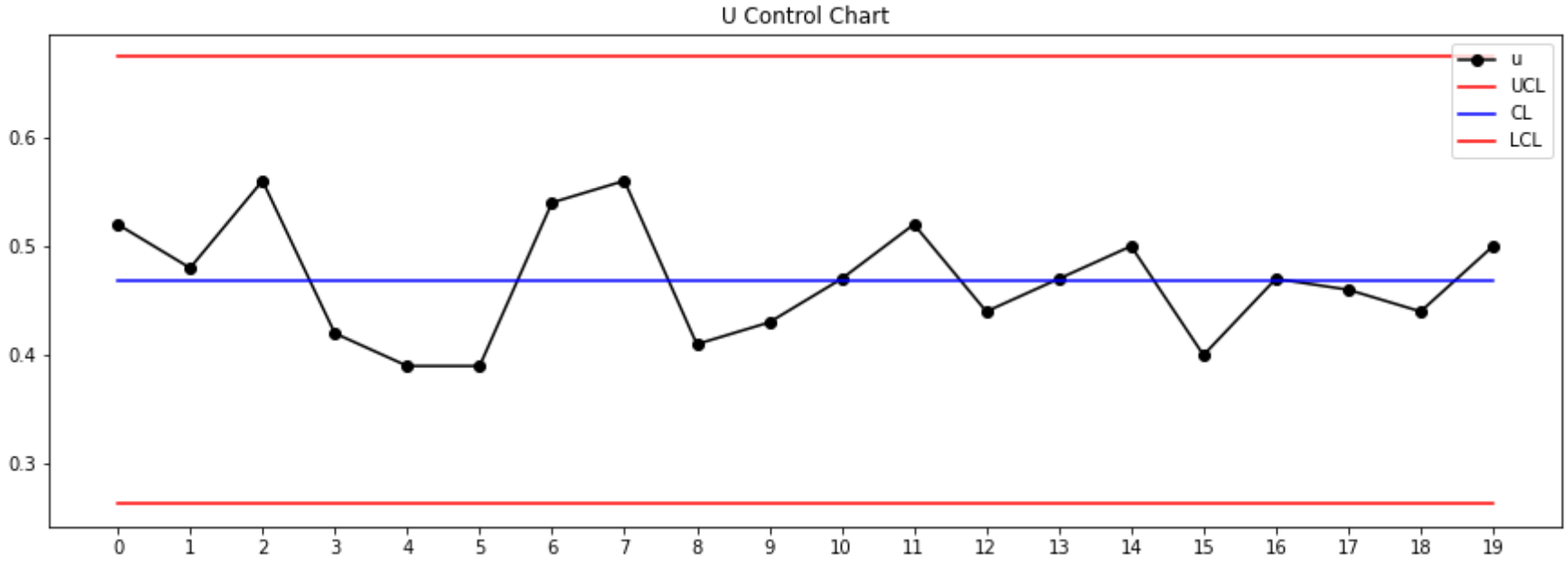

```
normally_distributed=False
stable=False
```

u chart

Attribute chart: **u chart** is also known as the control chart for defects per unit chart.

Reference: https://sixsigmastudyguide.com/attribute-chart-u-chart/

In [57]:
```
c = np.array([52,48,56,42,39,39,54,56,41,43,47,52,44,47,50,40,47,46,44,50])
n = np.array([100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100])
chart = UControlChart(c=c, n=n)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
print("stable={0}".format(chart.stable()))
```



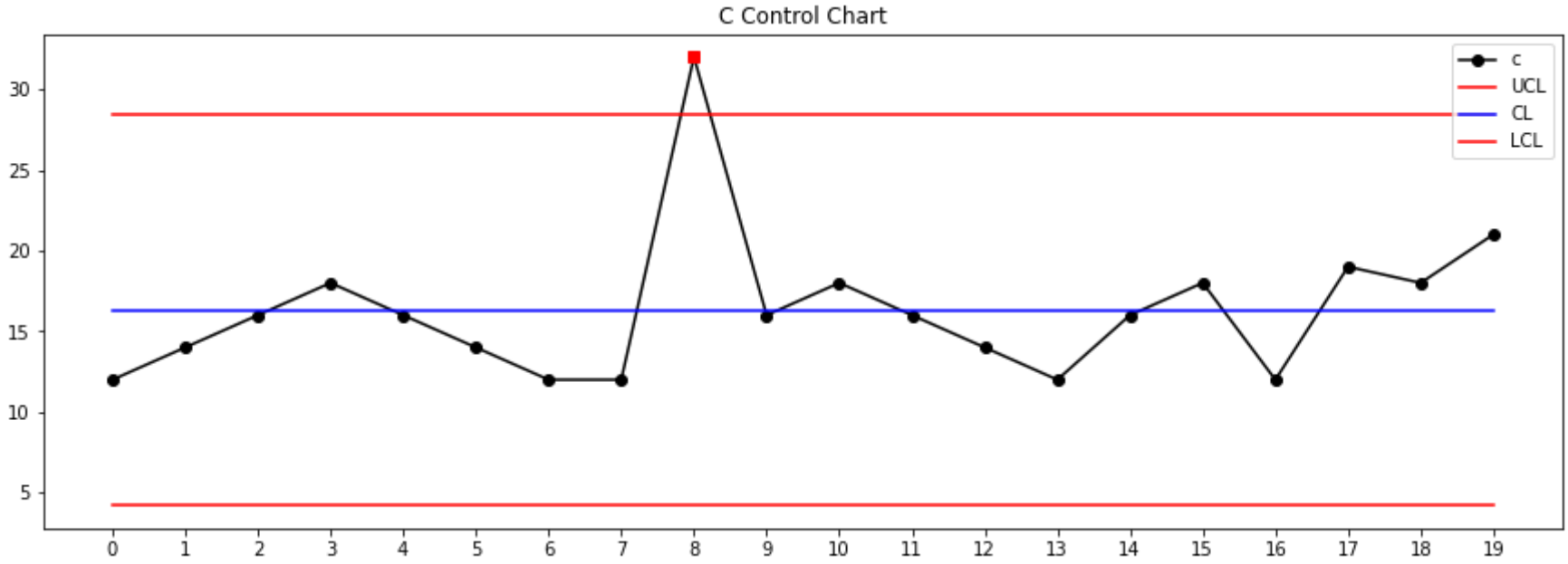

```
stable=True
```

c chart

Attribute chart: **c chart** is also known as the control chart for defects (counting of the number of defects). It is generally used to monitor the number of defects in constant size units.

Reference: https://sixsigmastudyguide.com/attribute-chart-c-chart/

In [63]:
```
c = np.array([12,14,16,18,16,14,12,12,32,16,18,16,14,12,16,18,12,19,18,21])
n = np.array([500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500])
chart = CControlChart(c=c, n=n)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
print("stable={0}".format(chart.stable()))
```



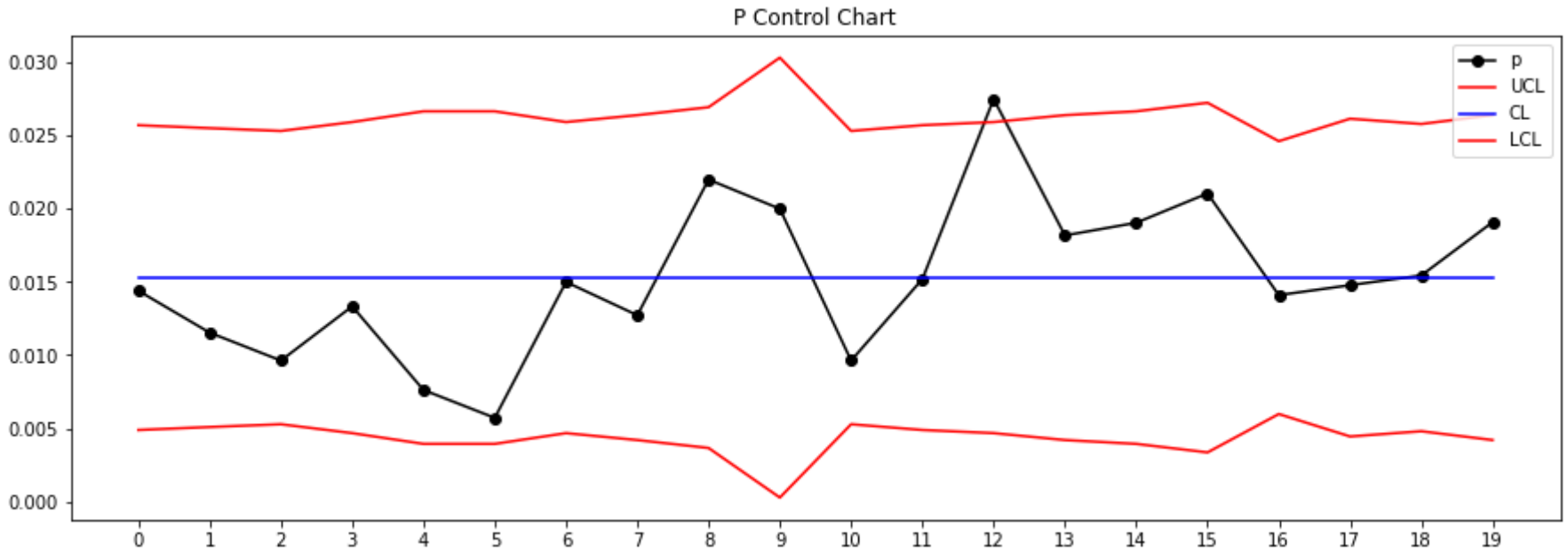

```
stable=False
```

p chart

Attribute charts: p chart is also known as the control chart for proportions.

Reference: https://sixsigmastudyguide.com/p-attribute-charts/

In [59]:
```
c = np.array([18,15,13,16,8,6,18,14,22,12,13,19,33,20,20,20,22,17,19,21])
n = np.array([1250,1300,1350,1200,1050,1050,1200,1100,1000,600,1350,1250,1200,1100,1050,950,1560,1150,1230,1100])
chart = PControlChart(c=c, n=n)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
print("stable={0}".format(chart.stable()))
```



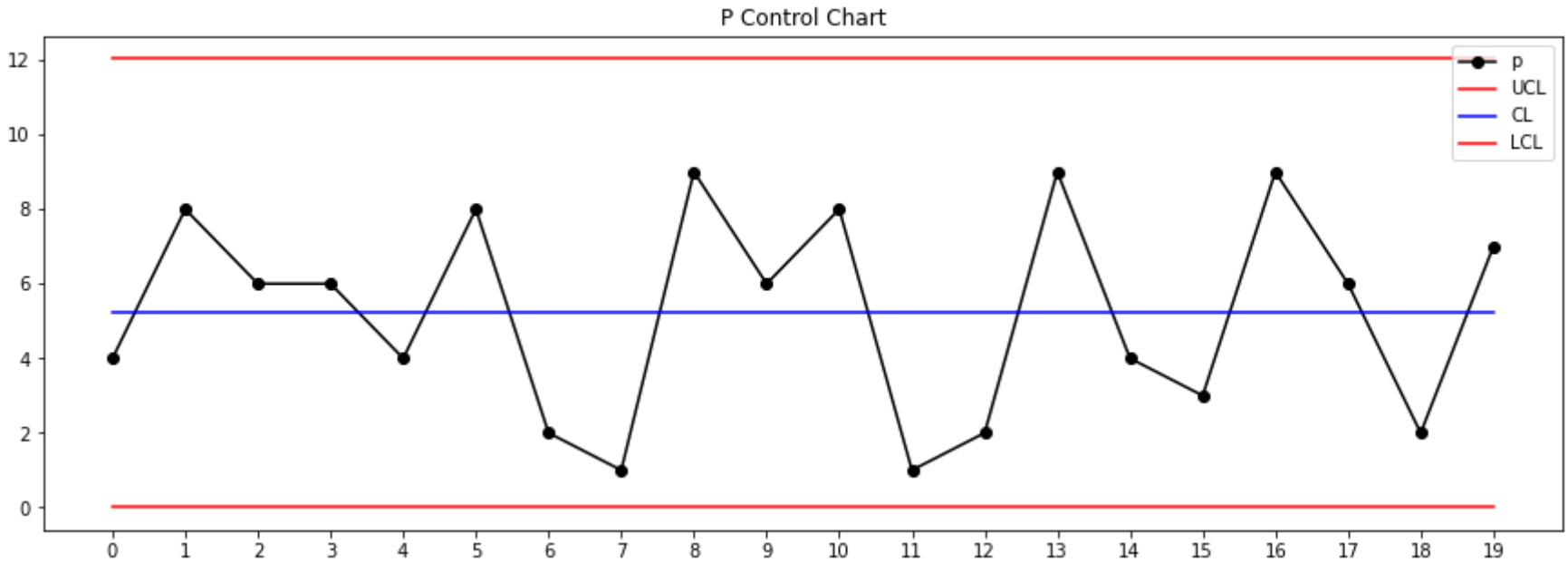

```
stable=False
```

np chart

Attribute chart: **np chart** is also known as the control chart for defectives (d-chart).

Reference: https://sixsigmastudyguide.com/attribute-chart-np-chart/

In [60]:
```
c = np.array([4,8,6,6,4,8,2,1,9,6,8,1,2,9,4,3,9,6,2,7])
n = np.array([200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200])
chart = NPControlChart(c=c, n=n)
chart.limits=False # Don't display chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.plot()
print("stable={0}".format(chart.stable()))
```

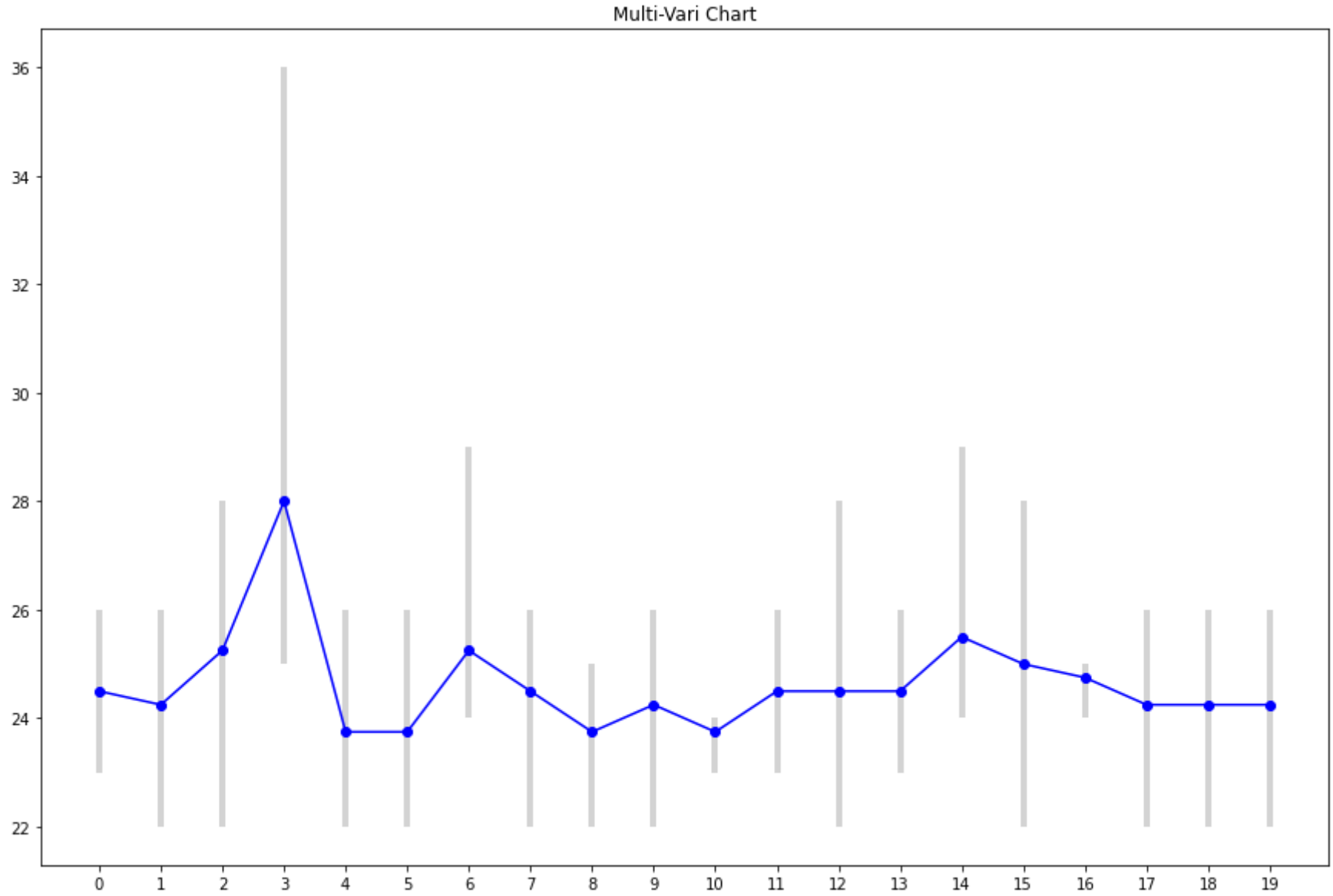



```
stable=True
```

Multi-vari chart

Multi-vari charts help you to identify sources of variation. **Reference: https://sixsigmastudyguide.com/multi-vari-study-multi-vari-charts/**

In [61]:
```
data = np.array([[23, 25, 24, 26],
                 [22, 26, 24, 25],
                 [28, 28, 22, 23],
                 [25, 25, 26, 36],
                 [22, 22, 25, 26],
                 [26, 24, 23, 22],
                 [29, 24, 24, 24],
                 [26, 25, 25, 22],
                 [22, 25, 24, 24],
                 [25, 22, 26, 24],
                 [24, 24, 24, 23],
                 [24, 25, 26, 23],
                 [22, 28, 22, 26],
                 [23, 24, 25, 26],
                 [24, 25, 29, 24],
                 [24, 22, 28, 26],
                 [24, 25, 25, 25],
                 [22, 24, 25, 26],
                 [26, 25, 22, 24],
                 [26, 22, 24, 25]])
chart = MultiVariChart(data=data)
chart.plot()
```



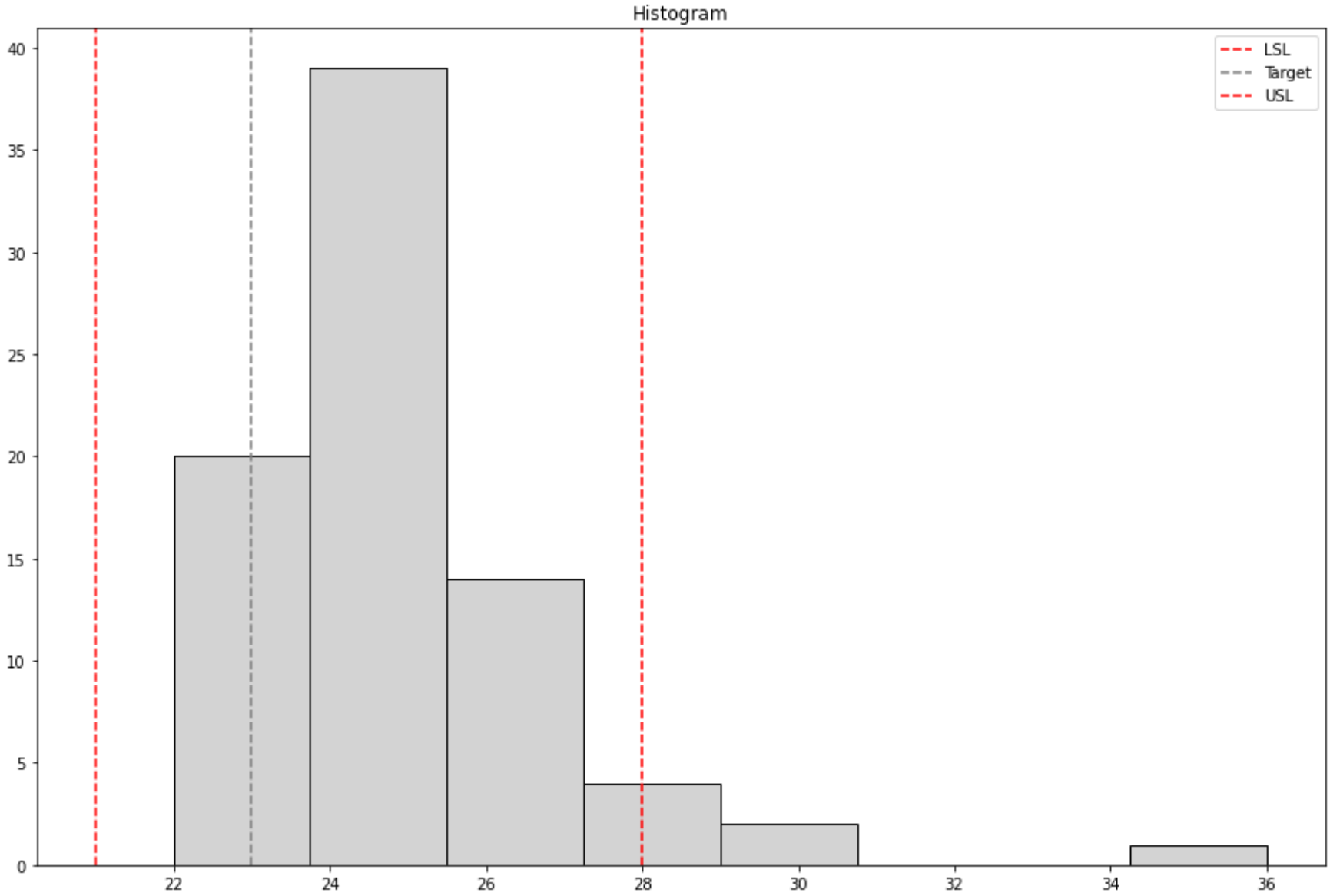
Process Capability chart

A **process capability chart** uses both the process variability and the specification limits to determine whether the process is "capable".

Reference: https://sixsigmastudyguide.com/process-capability-cp-cpk/

In [78]:

```
data = np.array([[23, 25, 24, 26],
                 [22, 26, 24, 25],
                 [28, 28, 22, 23],
                 [25, 25, 26, 36],
                 [22, 22, 25, 26],
                 [26, 24, 23, 22],
                 [29, 24, 24, 24],
                 [26, 25, 25, 22],
                 [22, 25, 24, 24],
                 [25, 22, 26, 24],
                 [24, 24, 24, 23],
                 [24, 25, 26, 23],
                 [22, 28, 22, 26],
                 [23, 24, 25, 26],
                 [24, 25, 29, 24],
                 [24, 22, 28, 26],
                 [24, 25, 25, 25],
                 [22, 24, 25, 26],
                 [26, 25, 22, 24],
                 [26, 22, 24, 25]])
data_flat = np.concatenate(data).ravel().tolist() # Flatten the data.
capability = ProcessCapabilityChart(data=data_flat, target=23, LSL=21, USL=28)
capability.plot(bins=8)
print("Cp={0}".format(capability.Cp))
print("Cpk={0}".format(capability.Cpk))
print("samples={0}".format(capability.num_samples))
print("mean={0}".format(capability.sample_mean))
print("stdev={0}".format(capability.sample_std))
print("max={0}".format(capability.sample_max))
print("min={0}".format(capability.sample_min))
print("mean={0}".format(capability.sample_median))
print("pct_below_LSL={0}".format(capability.pct_below_LSL))
print("pct_above_USL={0}".format(capability.pct_above_USL))
print("reject_rate={0}".format(capability.reject_rate))
print("capable={0}".format(capability.capable(target_cpk=1.33)))
```



Cp=0.5478833121624307
Cpk=0.5250036920220678
samples=80
mean=24.625
stdev=2.1428420734852893
max=36
min=22
mean=24.0

```
pct_below_LSL=0.0
pct_above_USL=3.75
reject_rate=3.75
capable=False
```

Calculated data extract (to visualize in a dashboard)

All control charts supports this functionality, below an example using the XmRControlchart.

In [88]:
```
data = np.array([2.92, 2.96, 2.86, 3.04, 3.07, 2.85, 3.00, 2.92, 2.97, 2.97, 3.09, 3.07, 2.99, 3.06, 3.05, 3.02, 3.07, 2.91, 3.07, 3.
chart = XmRControlChart(data=data)
chart.limits=False # True displays chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
df1 = chart.data(0) # 0=X
print(df1)
df2 = chart.data(1) # 1=mR
print(df2)
```

```
    value       UCL       +2s       +1s      CL       -1s       -2s       LCL  \
0    2.92  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
1    2.96  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
2    2.86  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
3    3.04  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
4    3.07  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
5    2.85  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
6    3.00  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
7    2.92  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
8    2.97  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
9    2.97  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
10   3.09  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
11   3.07  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
12   2.99  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
13   3.06  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
14   3.05  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
15   3.02  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
16   3.07  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
17   2.91  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
```

```
18   3.07  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338
19   3.20  3.239662  3.161275  3.082887  3.0045  2.926113  2.847725  2.769338

    SIGNAL    R01    R02    R03    R04    R05    R06    R07    R08
0    False  False  False  False  False  False  False  False  False
1    False  False  False  False  False  False  False  False  False
2    False  False  False  False  False  False  False  False  False
3    False  False  False  False  False  False  False  False  False
4    False  False  False  False  False  False  False  False  False
5    False  False  False  False  False  False  False  False  False
6    False  False  False  False  False  False  False  False  False
7    False  False  False  False  False  False  False  False  False
8    False  False  False  False  False  False  False  False  False
9    False  False  False  False  False  False  False  False  False
10   False  False  False  False  False  False  False  False  False
11   False  False  False  False  False  False  False  False  False
12   False  False  False  False  False  False  False  False  False
13   False  False  False  False  False  False  False  False  False
14   False  False  False  False  False  False  False  False  False
15   False  False  False  False  False  False  False  False  False
16   False  False  False  False  False  False  False  False  False
17   False  False  False  False  False  False  False  False  False
18   False  False  False  False  False  False  False  False  False
19   False  False  False  False  False  False  False  False  False
    value       UCL       +2s       +1s        CL       -1s       -2s  LCL  \
0    0.04  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
1    0.10  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
2    0.18  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
3    0.03  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
4    0.22  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
5    0.15  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
6    0.08  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
7    0.05  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
8    0.00  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
9    0.12  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
10   0.02  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
11   0.08  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
12   0.07  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
13   0.01  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
14   0.03  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
15   0.05  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
16   0.16  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
17   0.16  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0
18   0.13  0.288872  0.222055  0.155238  0.088421  0.058947  0.029474  0.0

    SIGNAL    R01    R02    R03    R04    R05    R06    R07    R08
0    False  False  False  False  False  False  False  False  False
```

```
1    False  False  False  False  False  False  False  False  False
2    False  False  False  False  False  False  False  False  False
3    False  False  False  False  False  False  False  False  False
4    False  False  False  False  False  False  False  False  False
5    False  False  False  False  False  False  False  False  False
6    False  False  False  False  False  False  False  False  False
7    False  False  False  False  False  False  False  False  False
8    False  False  False  False  False  False  False  False  False
9    False  False  False  False  False  False  False  False  False
10    True  False   True  False  False  False  False  False  False
11   False  False  False  False  False  False  False  False  False
12   False  False  False  False  False  False  False  False  False
13   False  False  False  False  False  False  False  False  False
14   False  False  False  False  False  False  False  False  False
15   False  False  False  False  False  False  False  False  False
16   False  False  False  False  False  False  False  False  False
17   False  False  False  False  False  False  False  False  False
18   False  False  False  False  False  False  False  False  False
```

Recognize stages automatically

All control charts supports this functionality, below an example using the XmRControlchart.

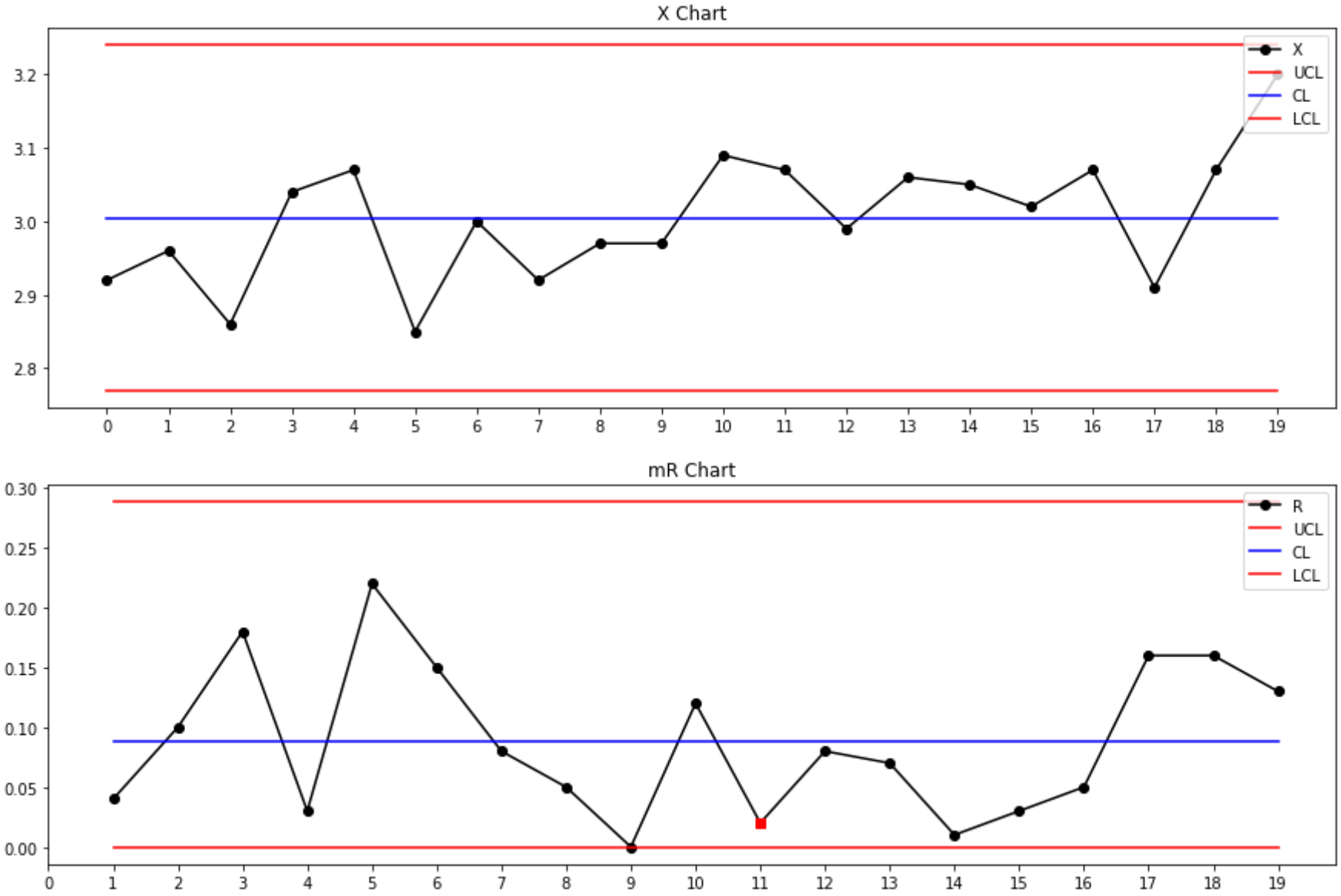
Dynamic programming model L2 (Euclidean Distance) is used to determine the different stages and the knee (elbow) locator the find the optimal number of stages.

Keep in mind that the data used doesn't justify stages, this is just for demo purpose.

In [90]:
```
data = np.array([2.92, 2.96, 2.86, 3.04, 3.07, 2.85, 3.00, 2.92, 2.97, 2.97, 3.09, 3.07, 2.99, 3.06, 3.05, 3.02, 3.07, 2.91, 3.07, 3.
chart = XmRControlChart(data=data)
chart.limits=False # True displays chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
stages=chart.stages(data=chart.value_X, max_stages=4)
if stages is not None:
    chart.split(stages)
else:
```

```
    print('no stages found')
chart.plot()
```

```
no stages found
```

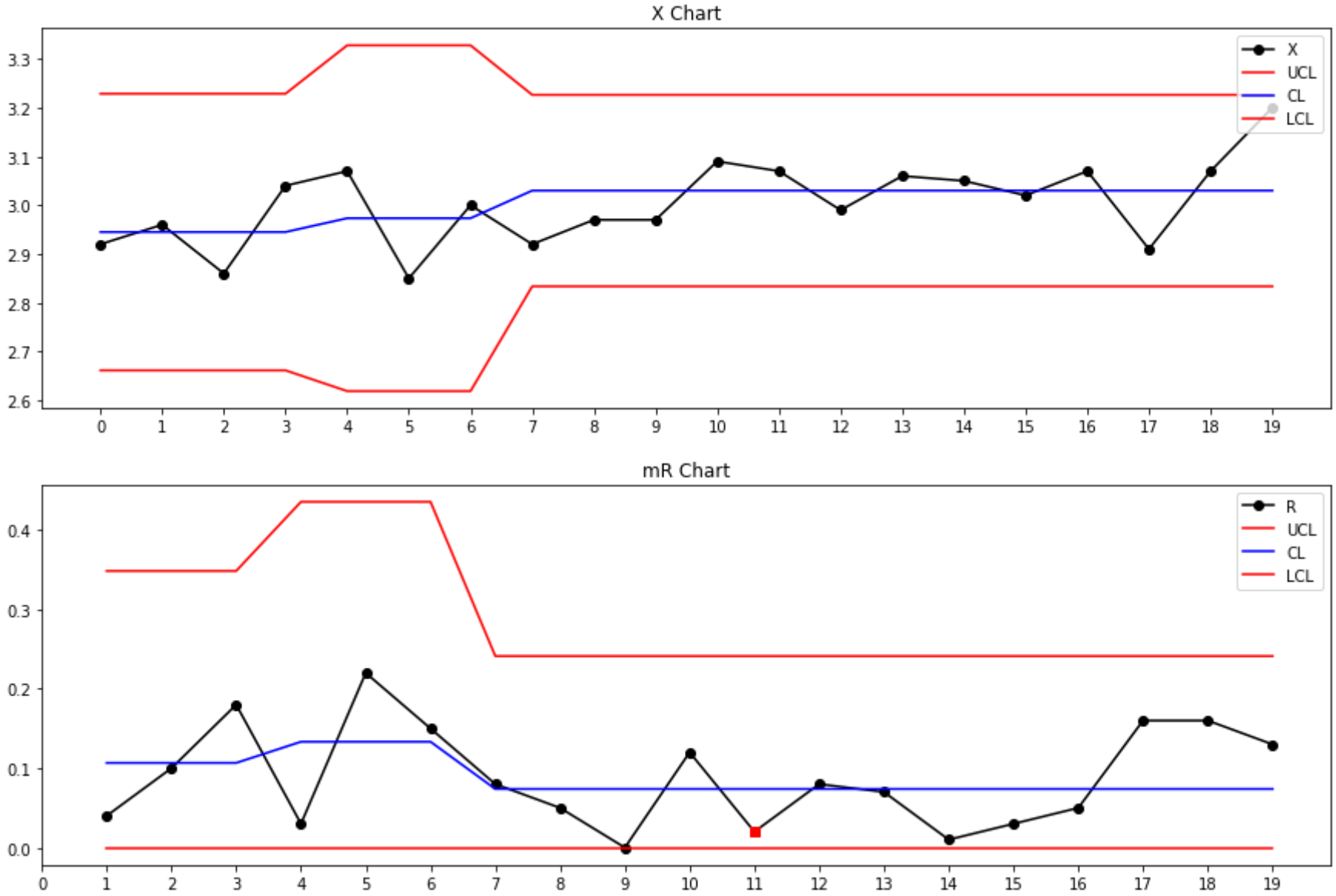


Set stages manually

All control charts supports this functionality, below an example using the XmRControlchart.

Keep in mind that the data used doesn't justify stages, this is just for demo purpose.

In [86]:
```
data = np.array([2.92, 2.96, 2.86, 3.04, 3.07, 2.85, 3.00, 2.92, 2.97, 2.97, 3.09, 3.07, 2.99, 3.06, 3.05, 3.02, 3.07, 2.91, 3.07, 3.
chart = XmRControlChart(data=data)
chart.limits=False # True displays chart zones.
chart.append_rule(Rule01())
chart.append_rule(Rule02())
chart.append_rule(Rule03())
chart.append_rule(Rule04())
chart.append_rule(Rule05())
chart.append_rule(Rule06())
chart.append_rule(Rule07())
chart.append_rule(Rule08())
chart.split([4, 7])
chart.plot()
```


X Chart
X
UCL
CL
LCL
3.3
3.2
3.1
3.0
2.9
2.8
2.7
2.6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
mR Chart
R
UCL
CL
LCL
0.4
0.3
0.2
0.1
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19