Quality - Individual Range Chart.

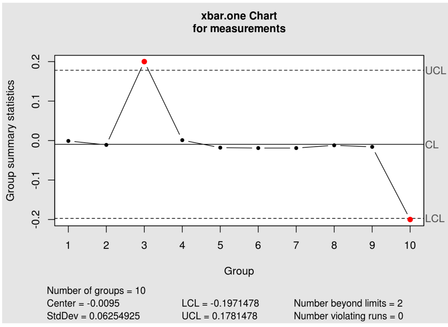
Carlos Kassab

2018, November 9

```
# Loading needed libraries
# R quality control library
suppressWarnings( suppressMessages( library( qcc ) ) )
# One of the R nice charts library
suppressWarnings( suppressMessages( library( dygraphs ) ) )

measurements = c( -0.001, -0.011, .2, 0.001, -0.018, -0.019, -0.019, -0.012, -0.016, -.2 )

# Using qcc library to display Individual Range Chart.
iRangeChart = qcc( measurements, type = "xbar.one", add.stats = TRUE, plot = TRUE )
```




```
# Getting qcc library values:
qccCenterLimit = round( iRangeChart$center, 8 )
qccStandardDeviation = round( iRangeChart$std.dev, 8 )
qccUpperControlLimit = round( iRangeChart$limits[1,2], 8 )
qccLowerControlLimit = round( iRangeChart$limits[1,1], 8 )
qccBeyondLimits = iRangeChart$violations$beyond.limits # Positions in our data to be red in chart.


#############################################################################
#                          Getting values manually.                         #
#############################################################################

# mean = average = sum(1..n) / n
ourCenterLimit = mean( measurements ) # CL

# By the formula:
# MovingRanges = absoluteValue( measurements[i+1] - measurements[i] )
# i = Measurement number
measurementsMovingRanges = abs ( measurements[1:length(measurements)-1] -
                                 measurements[2:length(measurements)] )

movingRangeCenterLimit = mean( measurementsMovingRanges )

# According to the formula:
# myStandardDeviation = MovingRangeMean / d2, where d2 = 1.128 for Individual Range Chart
ourStandardDeviation = movingRangeCenterLimit / 1.128

# According to the formula:
#  UCL = SamplesMean + ( 3 * ( MovingRangeMean / d2 ) )
ourUpperControlLimit = ourCenterLimit + ( 3 * ( movingRangeCenterLimit / 1.128 ) ) # USL

# According to the formula:
# LCL = SamplesMean - ( 3 * ( MovingRangeMean / d2 ) )
ourLowerControlLimit = ourCenterLimit - ( 3 * ( movingRangeCenterLimit / 1.128 ) ) # LCL

# Getting our out of limit positions in our data
ourBeyondLimits = c( which( measurements > ourUpperControlLimit )
                   , which( measurements < ourLowerControlLimit ) )


# Just to prove our calculations are the same, we do this comparison.

if( round( ourCenterLimit, 8 ) == qccCenterLimit ) {
  print( "Center Limit is The Same." )
} else cat( "Center Limit is DIFFERENT." )
```

```
> [1] "Center Limit is The Same."
```

```
if( round( ourStandardDeviation, 8 ) == qccStandardDeviation ) {
  cat( "Standard Deviation is The Same." )
} else cat( "Standard Deviation is DIFFERENT." )
```

```
> Standard Deviation is The Same.
```

```
if( round( ourUpperControlLimit, 8 ) == qccUpperControlLimit ) {
  cat( "Upper Control Limit is The Same." )
} else cat( "Upper Control Limit is DIFFERENT." )
```

```
> Upper Control Limit is The Same.
```

```
if( round( ourLowerControlLimit, 8 ) == qccLowerControlLimit ) {
  cat( "Lower Control Limit is The Same." )
} else cat( "Lower Control Limit is DIFFERENT." )
```

```
> Lower Control Limit is The Same.
```

```
if( identical( ourBeyondLimits, qccBeyondLimits ) ) {
  cat( "Beyond Limit Points Are The Same." )
} else cat( "Beyond Limit Points Are DIFFERENT." )
```

```
> Beyond Limit Points Are The Same.
```

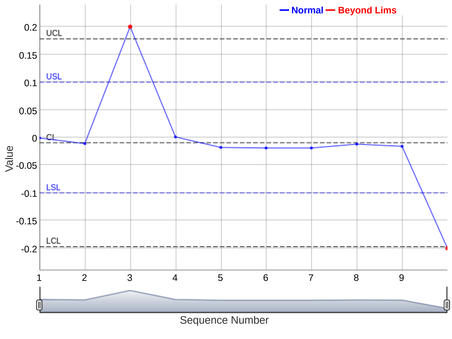
```
#############################################################################
#                  Creating a nice chart using dygraphs.                    #
#############################################################################

# Getting our dataframe as dygraphs needs it, note BeyondLimits is initialized to NA
measurementsData = data.frame( Sequence = seq( 1, length( measurements ), 1 )
                             , Values = measurements, BeyondLimits = NA )

# Now setting values to beyond limit points
measurementsData$BeyondLimits[ourBeyondLimits] = measurementsData[ourBeyondLimits,2]

# You can also set Specification limits, Lower Spec Lim and Upper Spec Lim
ourLSL = -0.1
ourUSL = 0.1

dygraph( measurementsData, main = NULL
       , xlab = "Sequence Number", ylab = "Value" ) %>%
  dySeries( name = "Values", label = "Normal", drawPoints = TRUE, pointShape = "dot"
          , color = "blue", pointSize = 2 ) %>%
  dySeries( name = "BeyondLimits", label = "Beyond Lims", drawPoints = TRUE, pointShape = "dot"
          , color = "red", pointSize = 3 ) %>%
  dyLimit( ourUpperControlLimit, color = "black"
         , label = "UCL", labelLoc = "left" ) %>%
  dyLimit( ourCenterLimit, color = "black"
         , label = "CL", labelLoc = "left" ) %>%
  dyLimit( ourLowerControlLimit, color = "black"
         , label = "LCL", labelLoc = "left" ) %>%
  dyLimit( ourUSL, color = "blue", label = "USL", labelLoc = "left" ) %>%
  dyLimit( ourLSL, color = "blue", label = "LSL", labelLoc = "left" ) %>%
  dyRangeSelector()
```



Number of measurements = 10

Center = -0.0095 **LCL** = -0.197148 **Number beyond limits** = 2
StdDev = 0.062549 **UCL** = 0.178148 **Number violating runs** = NA

Some Theory.

d2 is a value from constants table, which is 1.128 for Individual Range Chart calculations. The number 3 is a constant and typical value used in statistical control charts. This values are the same used by qcc R library.

This is a good site to go for information about the constant values: https://andrewmilivojevich.com/xbar-and-r-chart/

An interesting link on how to interpret this qcc chart: https://www.spcforexcel.com/knowledge/control-charts-basics/interpreting-control-charts

Note: I am not related to any of the above sites, I just put them as information.

Violating Runs

By the theory: Violating runs are points out of control, or points you should be careful with.

qcc library has a parameter called run.length as a way to control the way violating runs are calculated. just set the value before calling qcc library by running qcc.options function in this way:

qcc.options(run.length = 5) # As I saw in my tests, 7 is the default value for qcc library

You can see the qcc library sources in github how they are calculating violating runs.

In my case, it was not needed to go further with the violating runs calculation, that is why they are not mentioned in my calculations.

Note: I am not a specialist in quality control, I am just showing how I did to calculate the values in R and how to create a nice chart.