MaltHouse Dashboard Tutorial

Nicole Yanxin Ye

August 2020

Set up: Install R

To run R scripts in Power BI, you need to install R on your local machine. You can download and install R for free from the CRAN Repository. (https://cran.r-project.org/bin/windows/base/)

Create Power BI visuals using R

After you've installed R, Power BI Desktop enables it automatically. If the **Enable script visuals** window appears, select **Enable**.

1. Select the R Visual icon in the Visualization pane to add an R visual.
2. In the Values section of the Visualization pane, drag fields from the Fields pane that you want to consume in your R script, just as you would with any other Power BI Desktop visual.
3. Now you can use the data you selected to create a plot. Click on the R Visual you created, a R script editor window will appear below. Some supporting R script binding code has been generated automatically by Power BI. Don't change it.

The data you dragged is stored as a dataframe in R. The default name of the dataframe is "dataset". When you want to extract one column, for example, barley_protein, it is **dataset$barley_protein** in R.

Start coding below the line

```
# Paste or type your script code here:
```

For more details refer to PowerBI doc: **Run R scripts in Power BI Desktop**. https://docs.microsoft.com/en-us/power-bi/connect-data/desktop-r-scripts.

How the Filter Card works?

There are several filter cards on the canvas. They control what data goes into the R visuals. If you filter one variable, only filtered data will go into the R visual. If you clear all filters, all data will go into the R visual.

Plots in R

Set specifications

Most barley valieties share one general specification. But ABI4 and ABI5 have different specifications. Store the info into a dataframe and name it specs, for future use.

```
specs=data.frame("name" = c("general","ABI4", "ABI5"),
                 "protein_lower" = c(9.5, 10, 11),
                 "protein_upper" = c(12.5, 12.5, 13),
                 "don_upper" = c(1000, 1000, 1000),
                 "RT_upper"= c(5,5,5),
                 "CG_lower"= c(95,95,95),
                 "c1_lower" = c(75,75,75)
                )
specs
```

```
##      name protein_lower protein_upper don_upper RT_upper CG_lower c1_lower
## 1 general           9.5          12.5      1000        5       95       75
## 2    ABI4          10.0          12.5      1000        5       95       75
## 3    ABI5          11.0          13.0      1000        5       95       75
```

Here we implement a feature that when users filter barley valieties in Power BI, if certain variety was selected, e.g. ABI4 or ABI5, the specifications will change accordingly.

```
if ( unique(dataset$name)=="ABI4"){
    protein_lower = specs$protein_lower[specs['name']=="ABI4"]
    protein_upper = specs$protein_upper[specs['name']=="ABI4"]
    don_upper = specs$don_upper[specs['name']=="ABI4"]
    rt_upper =specs$RT_upper[specs['name']=="ABI4"]
    cg_lower = specs$CG_lower[specs['name']=="ABI4"]
    c1_lower = specs$c1_lower[specs['name']=="ABI4"]
  } else {
    if (unique(dataset$name)=="ABI5" ){
    protein_lower = specs$protein_lower[specs['name']=="ABI5"]
    protein_upper = specs$protein_upper[specs['name']=="ABI5"]
    don_upper = specs$don_upper[specs['name']=="ABI5"]
    rt_upper =specs$RT_upper[specs['name']=="ABI5"]
    cg_lower = specs$CG_lower[specs['name']=="ABI5"]
    c1_lower = specs$c1_lower[specs['name']=="ABI5"]
    } else{
    protein_lower = specs$protein_lower[specs['name']=="general"]
    protein_upper = specs$protein_upper[specs['name']=="general"]
    don_upper = specs$don_upper[specs['name']=="general"]
    rt_upper =specs$RT_upper[specs['name']=="general"]
    cg_lower = specs$CG_lower[specs['name']=="general"]
    c1_lower = specs$c1_lower[specs['name']=="general"]
    }
}
```

```
## Warning in if (unique(dataset$name) == "ABI4") {: the condition has length > 1
## and only the first element will be used
```

```
## Warning in if (unique(dataset$name) == "ABI5") {: the condition has length > 1
## and only the first element will be used
```

If you run into warning: "the condition has length > 1 and only the first element will be used", ignore it.

Keep in mind that the two chuncks of code above are needed in **every** R visual in PowerBI, otherwise you will lose this feature.

1. Histogram and Q-Q Plot

In natural and social sciences, most real-valued valiables are normally distributed, e.g., seed size, people's height, human IQ. Normal distribution is important, because a random valiable whose distribution is unknown will converge to normal distribution as the number of sample increases, according to central limit theorem.

Therefore, we assume **the quality of barley is also normally distributed**. We use histogram and Q-Q plot to analyze five major quality measurements: protein, DON, CG, RT, Classification1.

What is Q-Q plot?

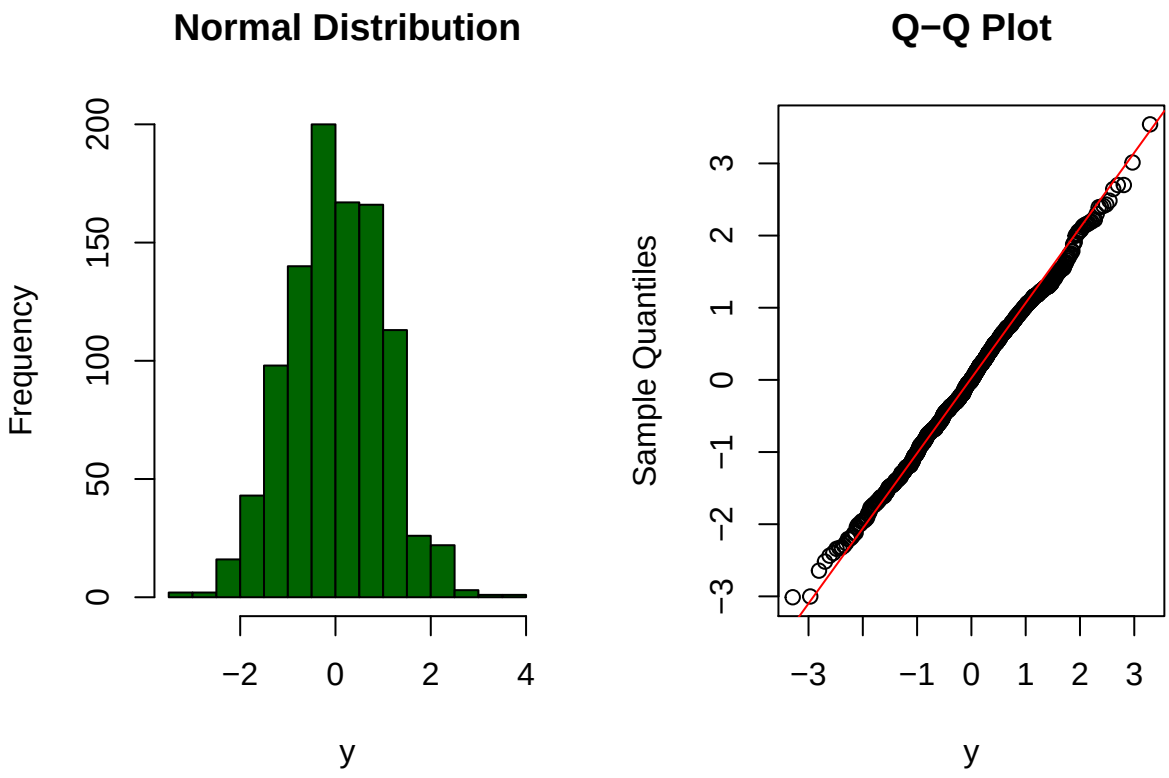
The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from normal distribution. We assume the quality of barley is Normally distributed, so we use a Normal Q-Q plot to check this assumption.

It's just a visual check, not an air-tight proof, so it is somewhat subjective. But it allows us to see at-a-glance if our assumption is plausible, and if not, how the assumption is violated, and what data points contribute to the violation.

The red line is qqline. It adds a line to a "theoretical" normal distribution.

Let's create a sample of 1000 numbers which are normally distributed, and check what an ideal Q-Q plot looks like.

```
set.seed(10)
par(mfcol=c(1,2))
y <- rnorm(1000)
hist(y, main = "Normal Distribution", col = "darkgreen",xlab = "y")
qqnorm(y, main ="Q-Q Plot", xlab = "y")
qqline(y, col = "red")
```



The points on Q-Q plot follow a strongly linear pattern, which suggests that the data is normally distributed. The histogram also strengthens our assumption.

Create Q-Q plot in R

Some old versions of R may require installing package to draw Q-Q plot. If you run into problems like that at the first time you run the script, uncomment the two lines below.

```
#install.packages("qqplotr",repos = "http://cran.us.r-project.org")
#library(qqplotr)
```

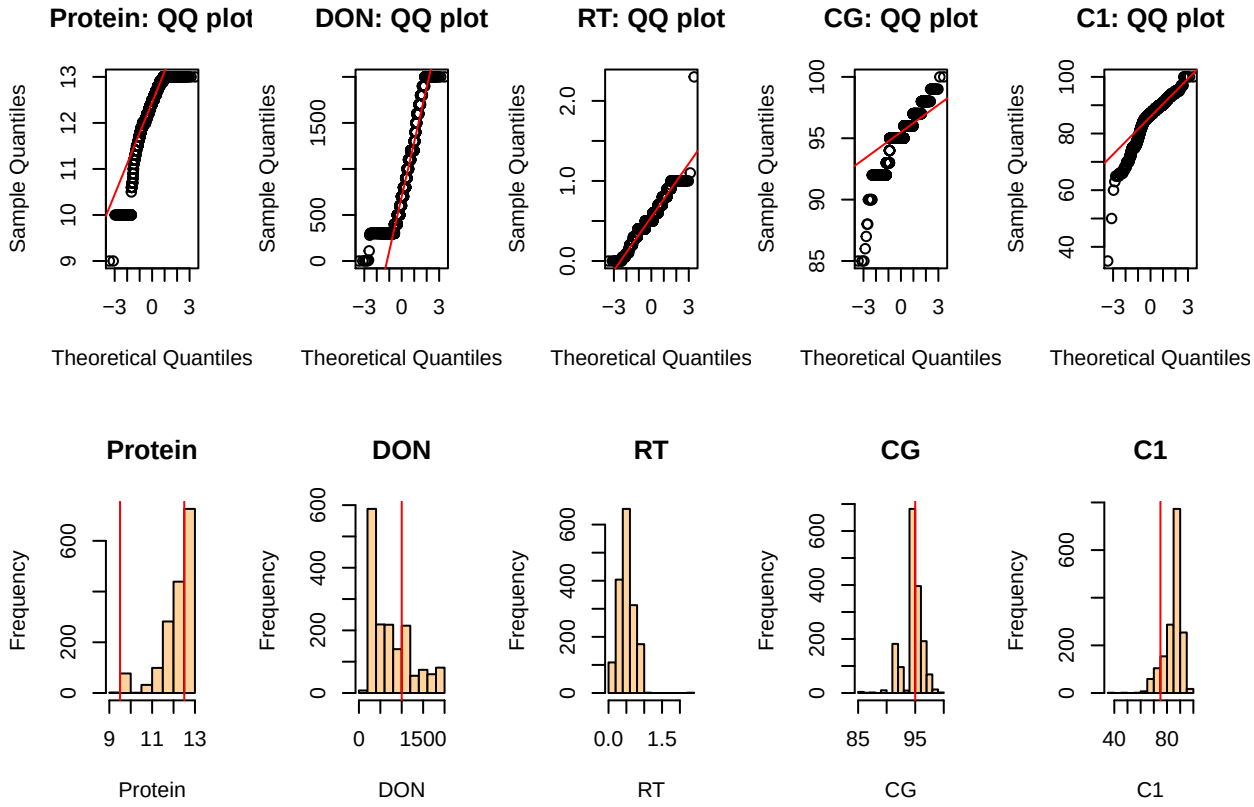
We use a function to draw QQ-plot.

The red lines are specifications.

```
qq <- function(val, val_name, lower_bound = NULL, upper_bound = NULL) {
  qqnorm(val, main = paste0(val_name, ": QQ plot"))
  qqline(val, col = "red")
  hist(val, xlab = val_name, main = val_name,col = "burlywood1")
  abline(v=lower_bound, col="red")
  abline(v=upper_bound, col="red")
}
```

Draw Q-Q plot and histograms of five major quality measurements, in a matrix of 2rows x 5cols plots.

```
par(mfcol=c(2,5))
qq(dataset$barley_protein, "Protein", lower_bound = protein_lower, upper_bound = protein_upper)
qq(dataset$barley_don, "DON", upper_bound = don_upper)
qq(dataset$barley_total_residue, "RT", upper_bound = rt_upper)
qq(dataset$barley_germination, "CG", lower_bound = cg_lower)
qq(dataset$barley_classification1, "C1", lower_bound = c1_lower)
```



2. BoxPlot

What is Box Plot?

A box plot is to visually show the distribution of numerical data and skewness through displaying the data quartiles.

Box plots show the five-number summary of a set of data (excluding outliers): the minimum , 25% quartile (Q1), median, 75% quartile (Q3), and maximum.

Interquartile range (IQR): 25th to the 75th percentile.

The blue line is called “whiskers”. The length of whiskers is usually set to 1.5*IQR.

In a box plot:

“Maximum”: Q3 + 1.5*IQR

“Minimum”: Q1 -1.5*IQR

Data points that outside the range of minimum and maxmum are defined as “outliers”.

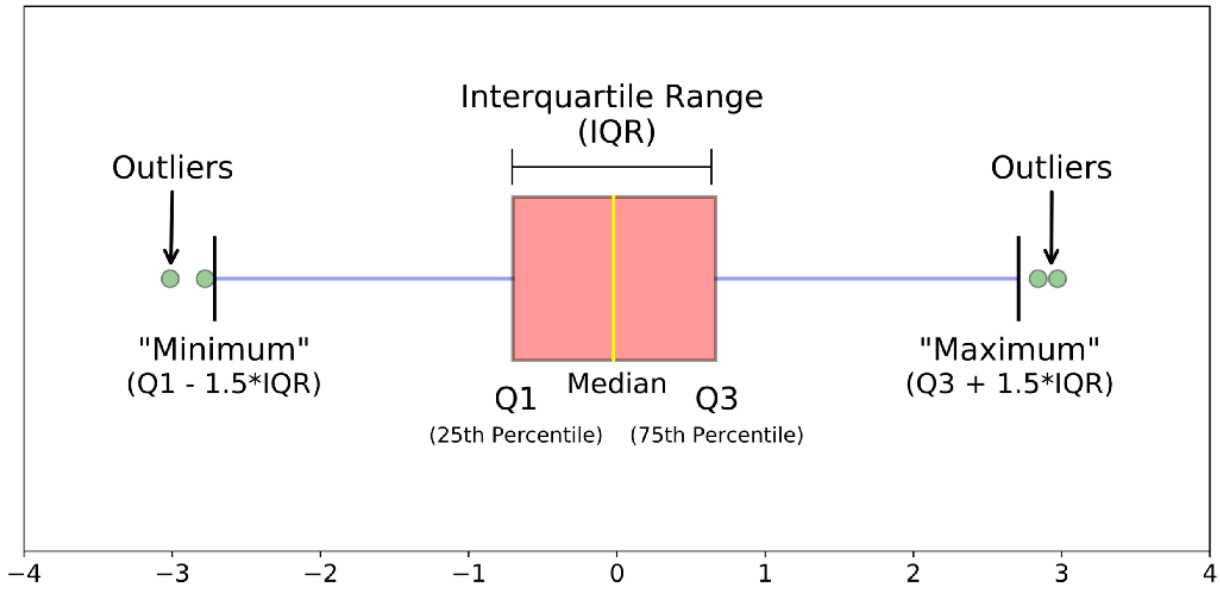


Figure 1: Box Plot

Create Box Plot in R

This set of graphs show the boxplot of quality measurements, and the boxplot if all outliers were removed.

The parameter 'range' is for setting the length of whisker. The default setting is 1.5. If too many outliers are detected , you can manully change the value to a larger number.

Red line: specifications.

Green line: mean. The number on top of the green line is the value of mean.

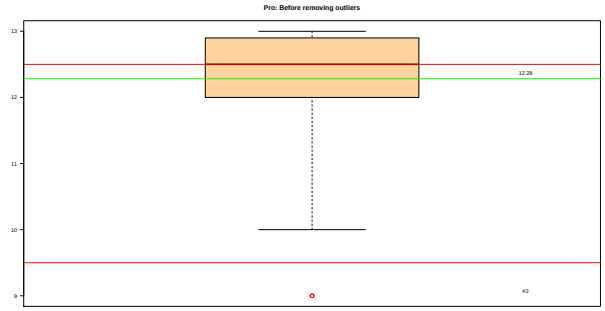
The number on bottom right of the graph with "#": the number of outliers detected.

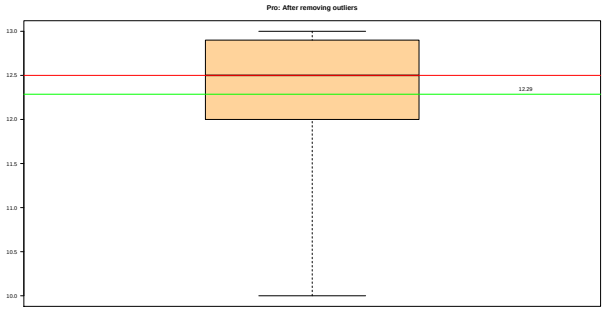
```
# Remove outlier
rm_outlier <- function(val, range = 1.5) {
  Q = quantile(val, probs=c(.25, .75), na.rm = FALSE)
  iqr <- IQR(val)
  up <-  Q[2]+range*iqr # "Maximum" of Boxplot
  low <- Q[1]-range*iqr # "Minimum" of Boxplot
  val_after = val[val>=low & val<=up]
  return (val_after)
}
# Function to draw boxplot
bp <- function(val, val_name, range = 1.5, lower_bound=NULL, upper_bound=NULL) {
  bp = boxplot(val,outcol = "red", outcex = 1.5,range=range
      , main =paste0(val_name, ": Before removing outliers")
      , las = 2, col = "burlywood1")
  abline(h = mean(val), col = 'green')
  abline(h = upper_bound, col = 'red')
  abline(h = lower_bound, col = 'red')
  text(x = 1.4, y = mean(val), ylim = c(min(val), max(val)),
       paste0(round(mean(val),2)), pos = 3, srt = 0)
  ## remove outlier
```

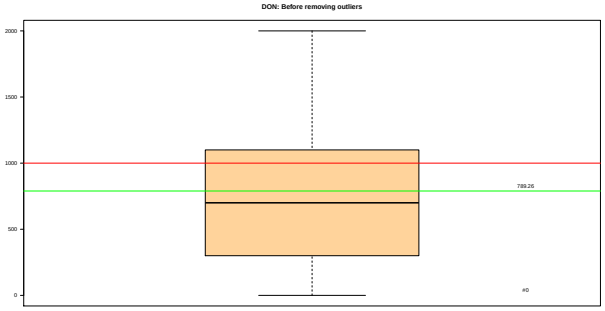
```
  val_rm = rm_outlier(val, range = range)
  ## count the number of outliers
  text(x = 1.4, y = min(val), pos = 3,paste0('#',length(val)-length(val_rm)))
  ## plot box plot after outliers were removed
    bp = boxplot(val_rm, outcol = "red", outcex = 1.5,range=range
  , main =paste0(val_name, ": After removing outliers")
  , las =2, col = "burlywood1")

  abline(h = mean(val_rm), col = 'green')
  abline(h = upper_bound, col = 'red')
  abline(h = lower_bound, col = 'red')
  text(x = 1.4, y = mean(val_rm), paste0(round(mean(val_rm),2)), pos = 3, srt = 0)
}
```

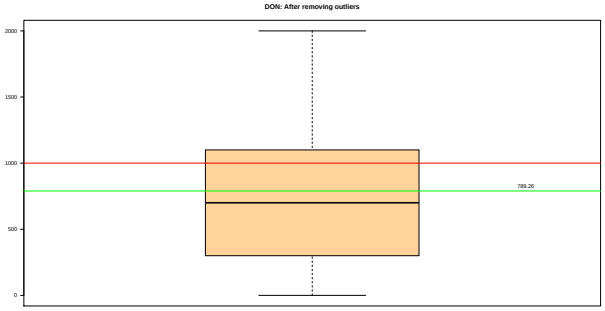
Draw boxplot of quality measurements in a matrix of 5rows x 2cols plots.

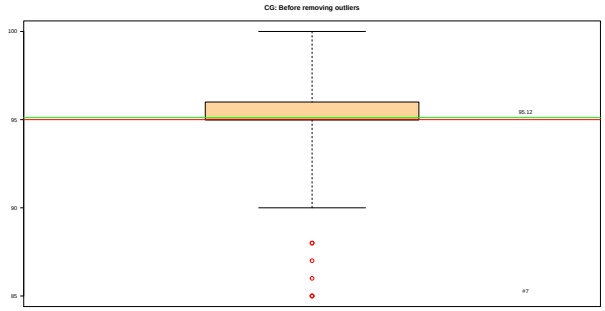
```
## Draw pictures
par(mfrow=c(5,2))
bp(dataset$barley_protein, 'Pro', range=3, lower_bound = protein_lower, upper_bound = protein_upper)
bp(dataset$barley_don, 'DON', upper_bound = don_upper)
bp(dataset$barley_germination,range= 5 ,"CG", lower_bound=cg_lower)
bp(dataset$barley_total_residue,range= 3 ,"RT", upper_bound=rt_upper)
bp(dataset$barley_classification1,range= 3 ,"C1",lower_bound = c1_lower )
```

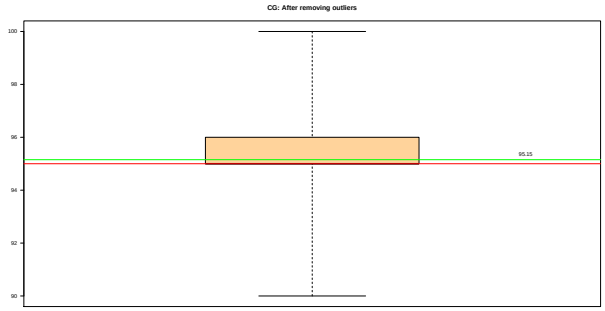

Pro: Before removing outliers
12.28
#2

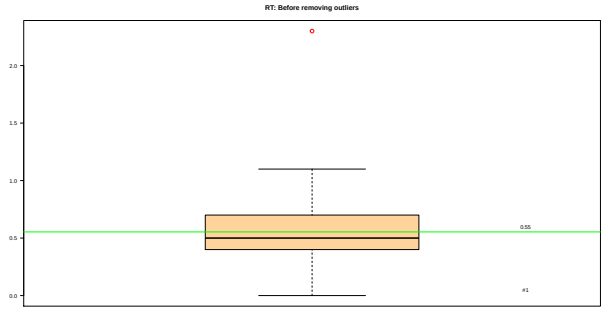

Pro: After removing outliers
12.29

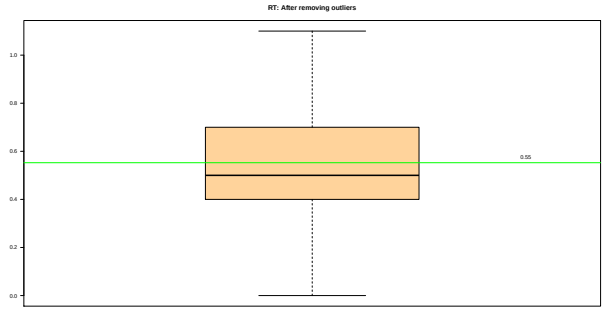

DON: Before removing outliers
789.26
#0

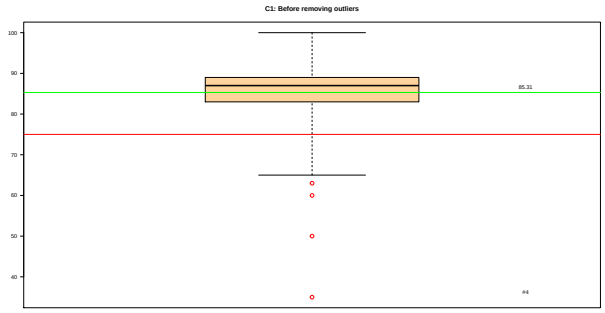

DON: After removing outliers
789.26

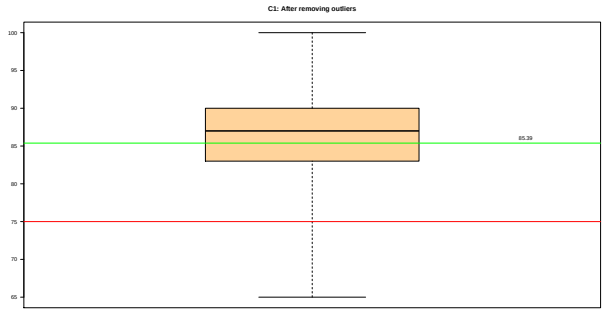

CG: Before removing outliers
95.12
#7


CG: After removing outliers
95.15


RT: Before removing outliers
0.55
#1


RT: After removing outliers
0.55


C1: Before removing outliers
85.31
#4


C1: After removing outliers
85.39

3. Control Chart

Control chart shows the chronological order of data. The first occurance in each date was labeled in x-axis.

Green dots: Data within specifications.

Red dots: Data outside specifications.

The red lines, as always, are specification limits.

The blue lines are the six-sigma line of variables.

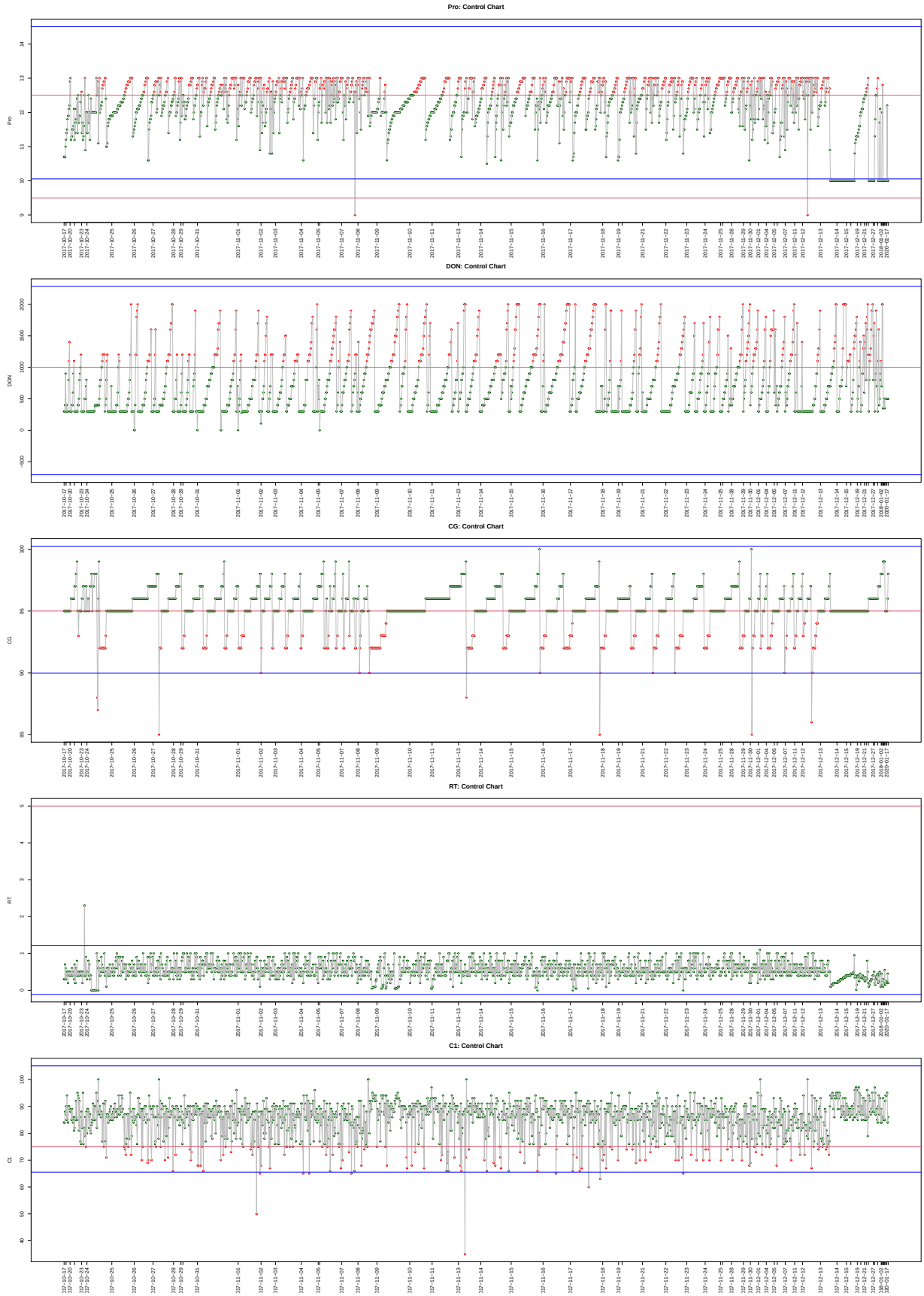
Draw Control Chart in R

There are four major steps to draw the control chart. 1. Sort data by receipt time. 2. Label the first occurance of each date. 3. Set color for dots. 4. Add specification and six-sigma lines.

```
control_chart <- function(val, val_name, lower_bound=NULL, upper_bound=NULL) {
## set dot color
df_sort$col="darkgreen"
df_sort$col[val>upper_bound]="red"
df_sort$col[val<lower_bound]="red"
## set 6sigma line
sigma = sqrt(var(val))
c1 = mean(val)-3*sigma
c2 = mean(val)+3*sigma
## set x-axis label to date
xlabel = unique(substr(df_sort$receipt_date,1,10))
xat = match(xlabel, substr(df_sort$receipt_date,1,10))
## draw line chart
plot(val, cex = 1, type="p", pch = 20,ylim= c(min(val,lower_bound,c1),max(val, upper_bound, c2)),
     col = df_sort$col, main =paste0(val_name, ": Control Chart"),
     ylab= val_name, xlab = "", xaxt = 'n')
axis(1, at=xat,labels = xlabel , las=2)
lines(val,col = "grey")
# draw spcifics
abline(h = upper_bound, col = 2)
abline(h = lower_bound, col = 2)
abline(h = c1, col = 'blue')
abline(h = c2, col = 'blue')
}
df_sort = dataset[order(dataset$receipt_date),]
```

Draw control graph in a matrix of 1rows x 5cols plots.

```
par(mfcol=c(5,1))
control_chart(dataset$barley_protein, "Pro", lower_bound=protein_lower, upper_bound=protein_upper)
control_chart(df_sort$barley_don, "DON",upper_bound = don_upper)
control_chart(dataset$barley_germination,"CG" ,lower_bound=cg_lower)
control_chart(dataset$barley_total_residue, "RT", upper_bound=rt_upper)
control_chart(dataset$barley_classification1, "C1", lower_bound=c1_lower)
```


Pro: Control Chart
Pro
DON: Control Chart
DON
CG: Control Chart
CG
RT: Control Chart
RT
C1: Control Chart
C1

Reference

For sample dataset, code and PowerBI file see https://github.com/Yanxin-Ye/Barley-Dashboard.