One-way output

```
############################One-way ANOVA#############################
#page 734, In a study of the effectiveness of different rust inhibitors, four brands (A, B, C,D)
#were tested. 40 experimental units were randomly assigned to the four brands, with 10
#units assigned to each brand. A portion of the results after exposing the experimental
#units to severe weather conditions is given in coded form in Table 17.2. The higher the coded
#value, the more effective is the rust inhibitor.


#install packages

# install.packages("lmtest") #levene test
# install.packages("car") #BF test, boxcox transformation
# install.packages("nortest") #normality test
# install.packages("plyr") #Tools for splitting, applying and combining data
# install.packages("ggplot2")
# install.packages(multcompView)  #tukey comparison
# install.packages(lsmeans)  #tukey comparison
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(nortest)
library(plyr)
library(ggplot2)
library(multcompView)
library(lsmeans)
```

```
## The 'lsmeans' package is being deprecated.
## Users are encouraged to switch to 'emmeans'.
## See help('transition') for more information, including how
## to convert 'lsmeans' objects and scripts to work with 'emmeans'.
```

```
rust<- read.table(file="~/Desktop/jenn/teaching/stat445545/data/CH17TA02.txt",sep="",col.names=c("resul
nt<-nrow(rust)
nt
```

```
## [1] 40
```

```
attach(rust)
rust$brand<-factor(rust$brand)  #brand is a categorical variable
rust
```

```
##    result brand obs
## 1    43.9     1   1
```

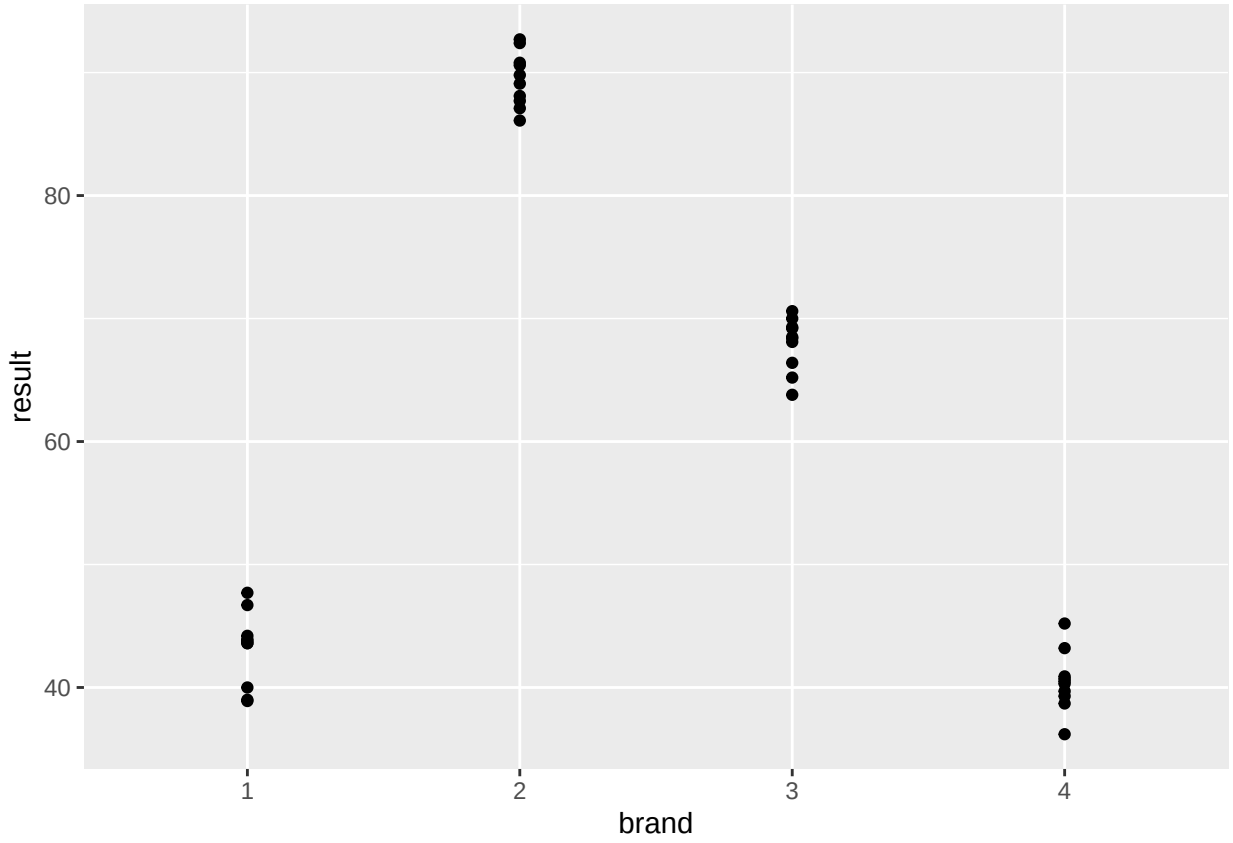
```
## 2    39.0     1   2
## 3    46.7     1   3
## 4    43.8     1   4
## 5    44.2     1   5
## 6    47.7     1   6
## 7    43.6     1   7
## 8    38.9     1   8
## 9    43.6     1   9
## 10   40.0     1  10
## 11   89.8     2   1
## 12   87.1     2   2
## 13   92.7     2   3
## 14   90.6     2   4
## 15   87.7     2   5
## 16   92.4     2   6
## 17   86.1     2   7
## 18   88.1     2   8
## 19   90.8     2   9
## 20   89.1     2  10
## 21   68.4     3   1
## 22   69.3     3   2
## 23   68.5     3   3
## 24   66.4     3   4
## 25   70.0     3   5
## 26   68.1     3   6
## 27   70.6     3   7
## 28   65.2     3   8
## 29   63.8     3   9
## 30   69.2     3  10
## 31   36.2     4   1
## 32   45.2     4   2
## 33   40.7     4   3
## 34   40.5     4   4
## 35   39.3     4   5
## 36   40.3     4   6
## 37   43.2     4   7
## 38   38.7     4   8
## 39   40.9     4   9
## 40   39.7     4  10
```

```r
#Summary statistics
# Calculate the mean, sd, n, and se for the four brands

# The plyr package is an advanced way to apply a function to subsets of data
#   "Tools for splitting, applying and combining data"
library(plyr)
# ddply "dd" means the input and output are both data.frames
rust.summary <- ddply(rust,
                      "brand",
                      function(X) {
                        data.frame( m = mean(X$result),
                                    s = sd(X$result),
                                    n = length(X$result)
                        )
```

```
                    }
)
rust.summary
```

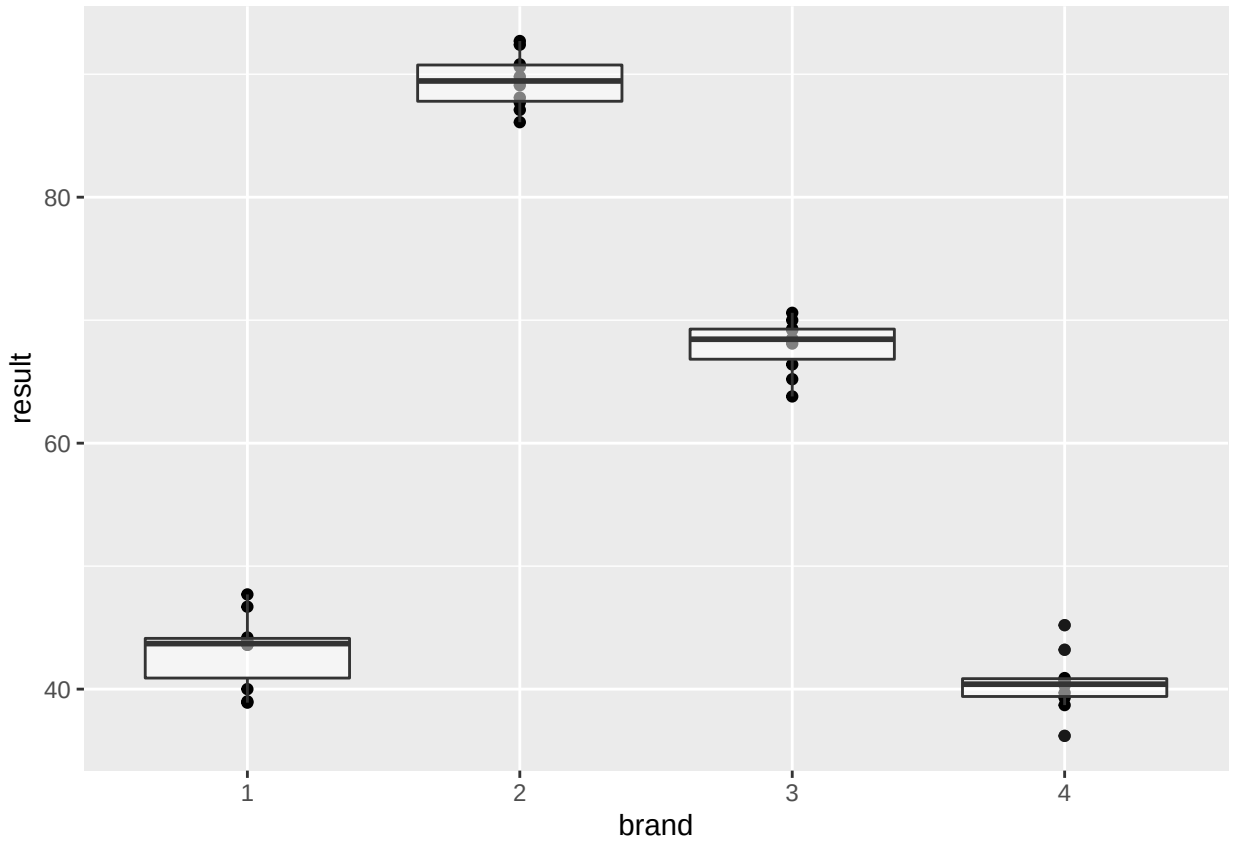
```
##   brand     m        s  n
## 1     1 43.14 3.000074 10
## 2     2 89.44 2.218207 10
## 3     3 67.95 2.168589 10
## 4     4 40.47 2.436322 10
```

```
# Plot the data using ggplot
library(ggplot2)
#### basic plot
#Geom: is the "type" of plot
#Aesthetics: shape, colour, size, alpha

p <- ggplot(rust, aes(x =brand, y = result))
p <- p + geom_point() # add a plot layer with points
print(p)
```




```
##adding titles and axis names
print(p+ggtitle("Plot of rust inhibitor brand data")
      +labs(y="Result after exposure (result)",x="Brands (brand)" ))
```


```
# boxplot
p <- p + geom_boxplot(alpha = 0.5)
print(p)
```




```
# points may lie on top of each other,reduce the
#opacity to 1/2 to indicate when points overlap.

##overall F test
oneway.test(result ~ brand, var.equal=TRUE)
```

```
##
##  One-way analysis of means
##
## data:  result and brand
## F = 866.12, num df = 3, denom df = 36, p-value < 2.2e-16
```

```
myfit = aov(result ~ brand, data=rust)
summary(myfit) # display Type I ANOVA table
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## brand        3  15953    5318   866.1 <2e-16 ***
## Residuals   36    221       6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
myfitlm<-summary.lm(myfit)
myfitlm
```

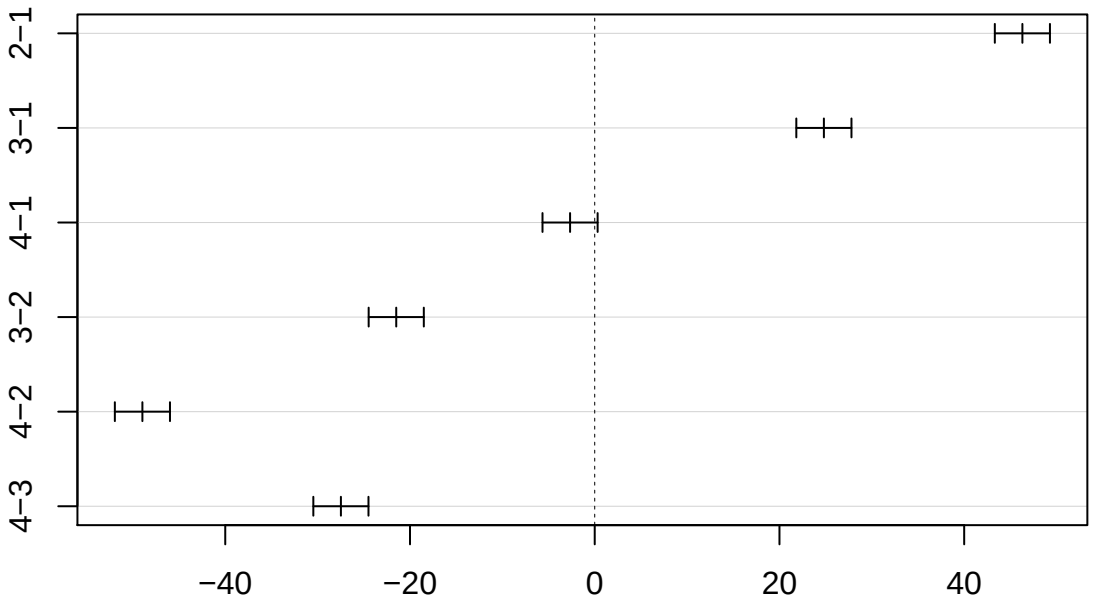
```
##
## Call:
## aov(formula = result ~ brand, data = rust)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -4.270 -1.597  0.395  1.275  4.730
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  43.1400     0.7836  55.056   <2e-16 ***
## brand2       46.3000     1.1081  41.782   <2e-16 ***
## brand3       24.8100     1.1081  22.389   <2e-16 ***
## brand4       -2.6700     1.1081  -2.409   0.0212 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.478 on 36 degrees of freedom
## Multiple R-squared:  0.9863, Adjusted R-squared:  0.9852
## F-statistic: 866.1 on 3 and 36 DF,  p-value: < 2.2e-16
```

```r
#You can look at the results from aov() in terms of the
#linear regression that was carried out, i.e. you can see the parameters that were estimated.

#Tukey's multiple comparison
myfit2<-TukeyHSD(myfit,conf.level=.95)
myfit2
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = result ~ brand, data = rust)
##
## $brand
##       diff        lwr         upr     p adj
## 2-1  46.30  43.315536  49.2844635 0.0000000
## 3-1  24.81  21.825536  27.7944635 0.0000000
## 4-1  -2.67  -5.654464   0.3144635 0.0933303
## 3-2 -21.49 -24.474464 -18.5055365 0.0000000
## 4-2 -48.97 -51.954464 -45.9855365 0.0000000
## 4-3 -27.48 -30.464464 -24.4955365 0.0000000
```

```r
plot(myfit2, sub="Tukey Honest Significant Differences")
```



Tukey Honest Significant Differences

```
comp1<-lsmeans(myfit, "brand",adjust="tukey")
cld(comp1, alpha=.05,Letters=letters)
```

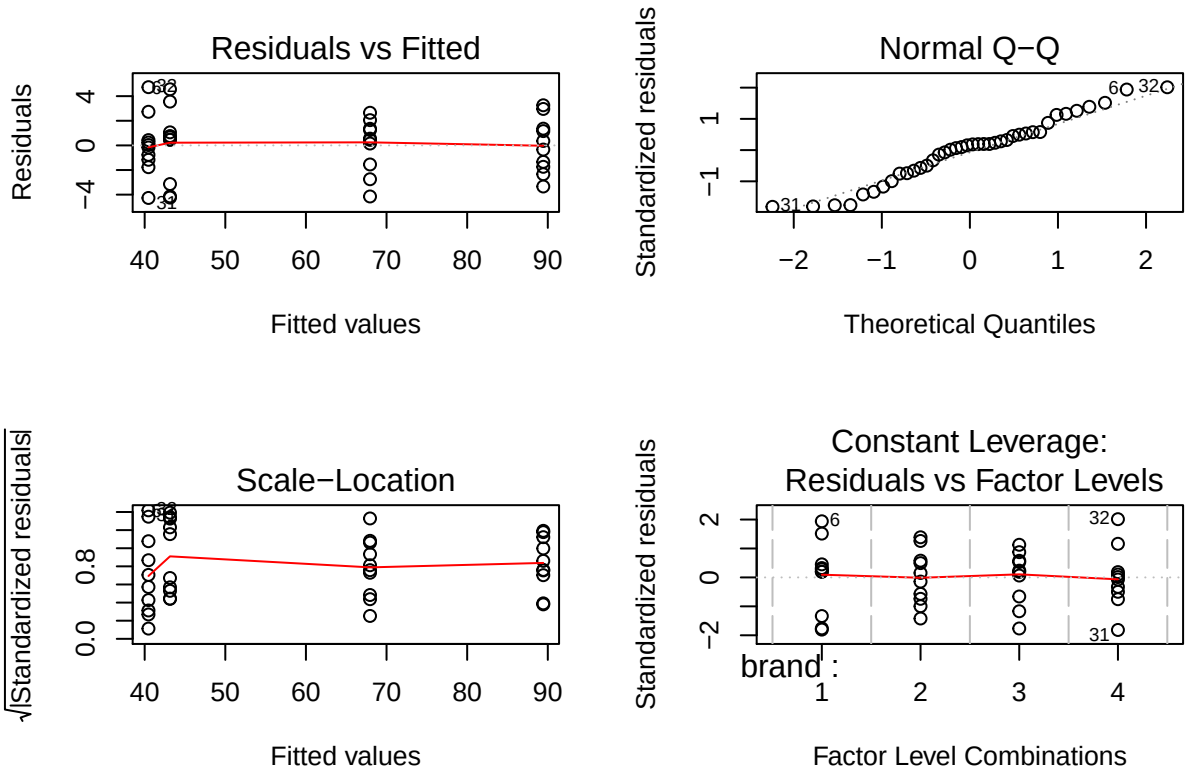
```
##  brand lsmean        SE df lower.CL upper.CL .group
##  4      40.47 0.7835709 36 38.88084 42.05916  a
##  1      43.14 0.7835709 36 41.55084 44.72916  a
##  3      67.95 0.7835709 36 66.36084 69.53916   b
##  2      89.44 0.7835709 36 87.85084 91.02916    c
##
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 4 estimates
## significance level used: alpha = 0.05
```

```
#Bonferroni multiple comparison

pairwise.t.test(result, brand,
                pool.sd = TRUE, p.adjust.method = "bonf")
```

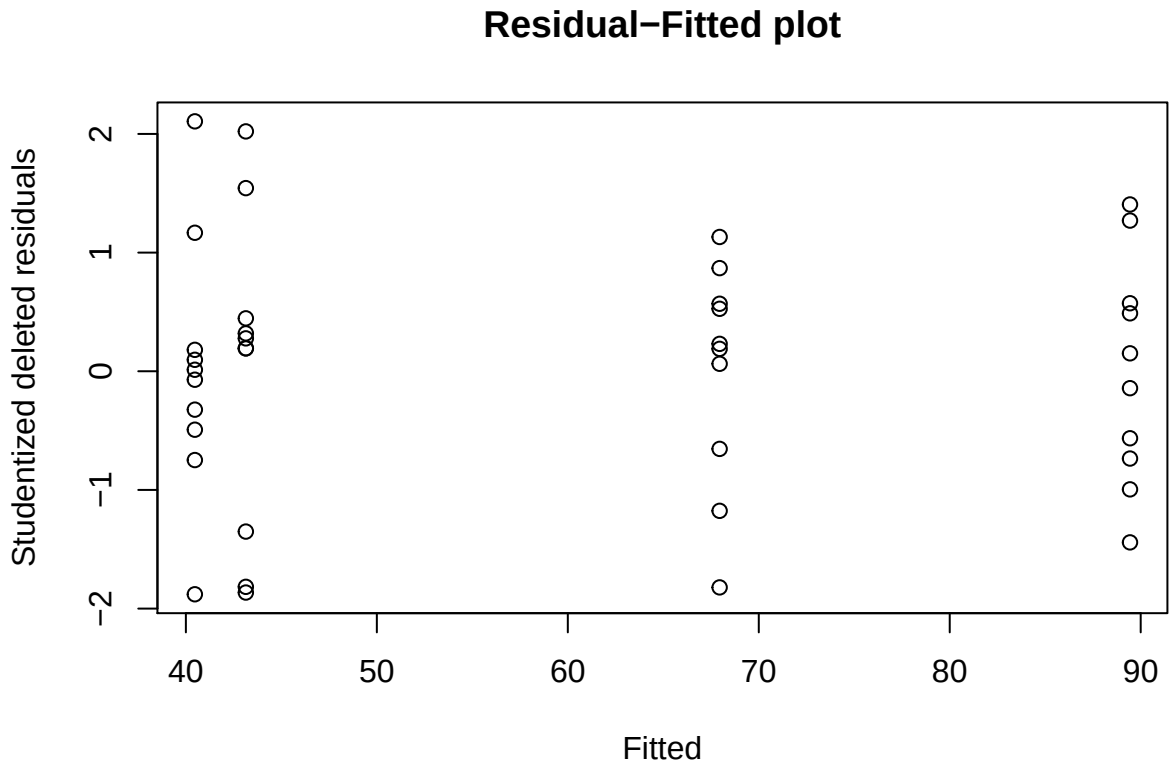
```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  result and brand
##
##   1      2      3
## 2 <2e-16 -      -
## 3 <2e-16 <2e-16 -
## 4 0.13   <2e-16 <2e-16
##
```

```
## P value adjustment method: bonferroni
```

```
#Diagnostic plots provide checks for heteroscedasticity, normality, and influential observerations.
#Standardized residuals, also called internally studentized residuals
par(mfrow=c(2,2)) # optional layout
plot(myfit) # diagnostic plots
```




```
##plot studentized deleted residual vs fitted values
par(mfrow=c(1,1))
plot(myfit$fit,rstudent(myfit),xlab="Fitted",
     ylab="Studentized deleted residuals",main="Residual-Fitted plot")
```




```
##check outliers
rstudent<-rstudent(myfit)
bcritical<- qt(1-0.05/(2*nt), nt-4-1)
outliers <- which(abs(rstudent) > bcritical)
outliers
```

```
## named integer(0)
```

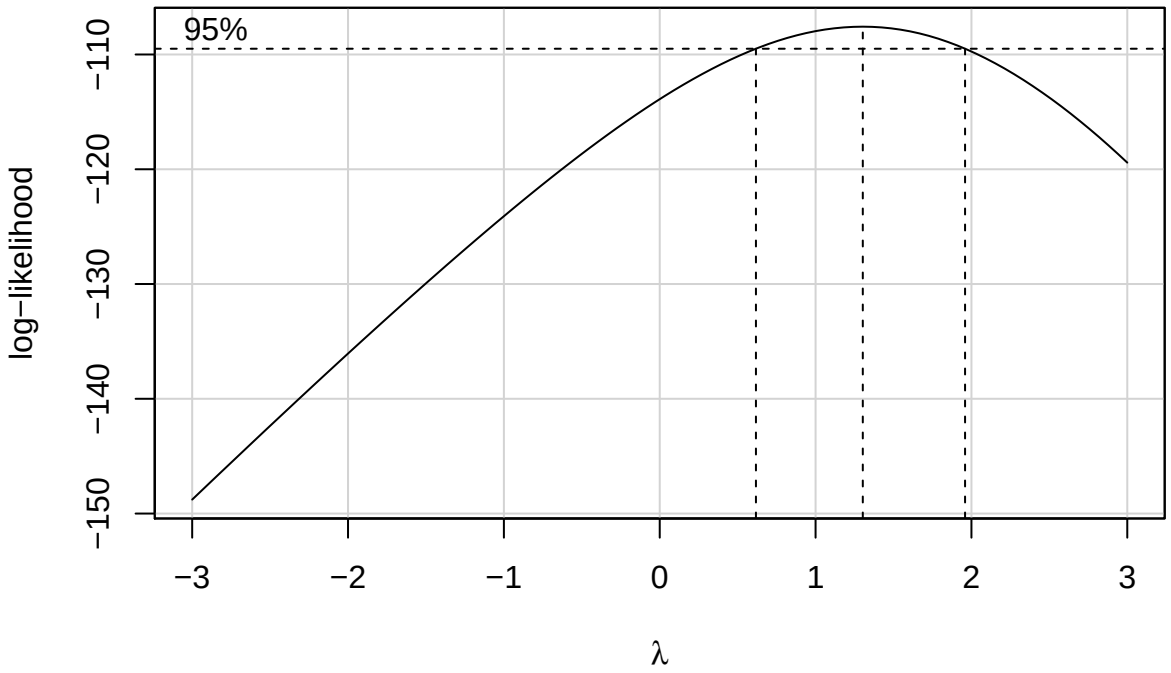
```
##Brown-Forsythe Test (modified levene test) for constancy of the error variance, reject constant varia
#p-value of the test is less than 0.05
leveneTest(result~brand, data=rust)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  3  0.2262 0.8775
##       36
```

```
##perform normality tests
#Shapiro-Wilk test is based approximately also on the coefficient of correlation between the ordered
#residuals and their expected values under normality, reject normality assumption if p-value of the tes
shapiro.test(myfit$resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  myfit$resid
## W = 0.96703, p-value = 0.2887
```

```
#box-cox transformation
boxCox(myfit, lambda = seq(-3, 3, length = 10))
```




```
##\lambda =1 is within the 95\% CI, therefore, no transformation is needed
```