STAT 6910: HW 5

David Angeles

```
## Warning: package 'emmeans' was built under R version 3.4.4
```

```
## NOTE: As of emmeans versions > 1.2.3,
##       The 'cld' function will be deprecated in favor of 'CLD'.
##       You may use 'cld' only if you have package:multcomp attached.
```

Problem 1

Check the assumptions on the one-way analysis of variance model (3.3.1) for the meat cooking experiment, which was introduced in Exercise 14 of Chap.3. The data were given in Table 3.14. (the order of collection of observations is not available).

The data displayed below is represented by the levels 1, 2, 3, 4, 5, and 6 which denote the frying fat content at 10%, 15%, and 20% and the grilling fat content at 10%, 15% and 20% respectively for the post-cooking weight data (in grams) for the meat cooking experiment.

```
colnames(Post_grams) <- c( "Weight", "Code")
Post_grams <- data.frame(Post_grams)
Post_grams$Code <- factor(Post_grams$Code)
summary(Post_grams)
```

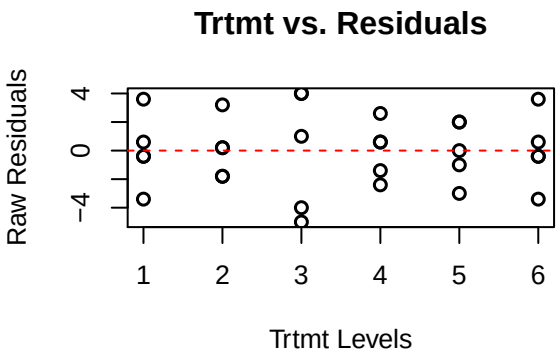
```
##      Weight      Code
##  Min.   :71.00   1:5
##  1st Qu.:80.00   2:5
##  Median :82.00   3:5
##  Mean   :81.67   4:5
##  3rd Qu.:84.75   5:5
##  Max.   :88.00   6:5
```

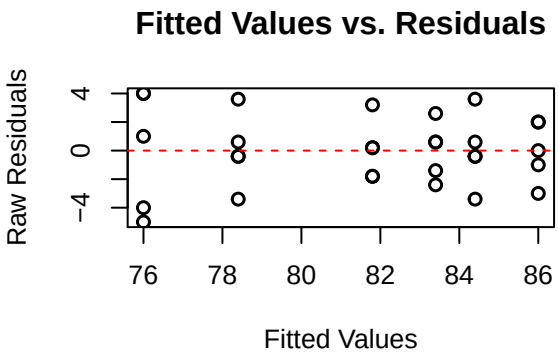
```
Post_weight_model <- aov(Weight ~ Code , data = Post_grams)
anova(Post_weight_model)
```

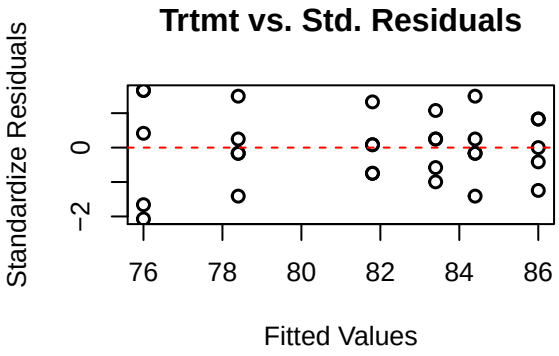
```
## Analysis of Variance Table
##
## Response: Weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Code       5 360.27  72.053  9.9156 3.075e-05 ***
## Residuals 24 174.40   7.267
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

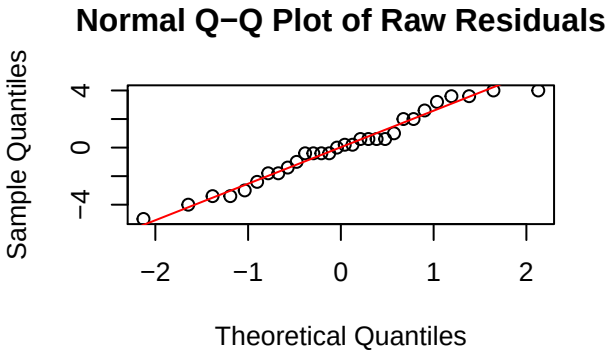
```r
# Fitted Values
weights.fitted <- fitted(Post_weight_model)
# Raw Residuals
weights.raw.resid <- resid(Post_weight_model)
# Standardized Residuals
std.residuals <- rstandard(Post_weight_model)
```

```r
par(mfrow = c(2,2))
#Raw Residuals vs. Frying/ Grilling fat content
plot(as.numeric(Post_grams$Code) , weights.raw.resid,
     main= "Trtmt vs. Residuals", xlab = "Trtmt Levels",
     ylab = "Raw Residuals", xaxt = "n",
     lwd = 1.5)
axis(1, at = 1:6, labels = c("1","2","3","4","5","6"))
abline(h=0, col = "red", lty = 2)

#Raw Residuals vs. Fitted Values
plot(weights.fitted , weights.raw.resid,
     main= "Fitted Values vs. Residuals", xlab = "Fitted Values",
     ylab = "Raw Residuals",
     lwd = 1.5)
abline(h=0, col = "red", lty = 2)

#Standardized Residuals vs. Fitted Values
plot(weights.fitted , std.residuals,
     main= "Trtmt vs. Std. Residuals", xlab = "Fitted Values",
     ylab = "Standardize Residuals",
     lwd = 1.5)
abline(h=0, col = "red", lty = 2)

#Norma Probability Plot
qqnorm(weights.raw.resid, main= "Normal Q-Q Plot of Raw Residuals")
qqline(weights.raw.resid, col = "red")
```









Assumption (a): The error have mean 0:

Due to the formulation of the One Way ANOVA Model, the residuals always sum up to 0. Hence, the assumption cannot be checked.

Assumption (b): The error have constant variance:

```
vars<- tapply(Post_grams$Weight,Post_grams$Code, var)
max(vars)/min(vars)
```

```
## [1] 4.868421
```

Using the rule of thumb based on within group variances we can see that

$$\frac{\max_i\{S_i^2\}}{\min_i\{S_i^2\}} = \frac{18.5}{3.8} = 4.87.$$

However, by plotting the standardized residuals against the fitted values and the treatment levels we saw no big difference in the pattern of the spread within the groups. Therefore, the $\frac{\max_i\{S_i^2\}}{\min_i\{S_i^2\}}$ result can simply be due to the fact that the sample sizes of each group is only 4. So we feel comfortable that the equal variance assumption is approximately satisfied.

Assumption (c): The error are normally distributed:

From the qq-plot above we see that the data is fairly straight. Furthermore, all the standardized residuals are $|z_i| \leq 2.074$, so we don't have any apparent outliers. Therefore the normality assumption is reasonable.

```r
max(abs(std.residuals))
```

```
## [1] 2.073755
```

Assumption (d): The errors are independent:

Since we do not have any information on how the data was collected, we can't verify the Independence assumption.

Problem 2

The spaghetti sauce experiment was run to compare the thicknesses of three particular brands of spaghetti sauce, both when stirred and unstirred. The six treatments were:

1 = store brand, unstirred 2 = store brand, stirred
3 = national brand, unstirred 4 = national brand, stirred
5=gourmet brand,unstirred 6=gourmet brand,stirred

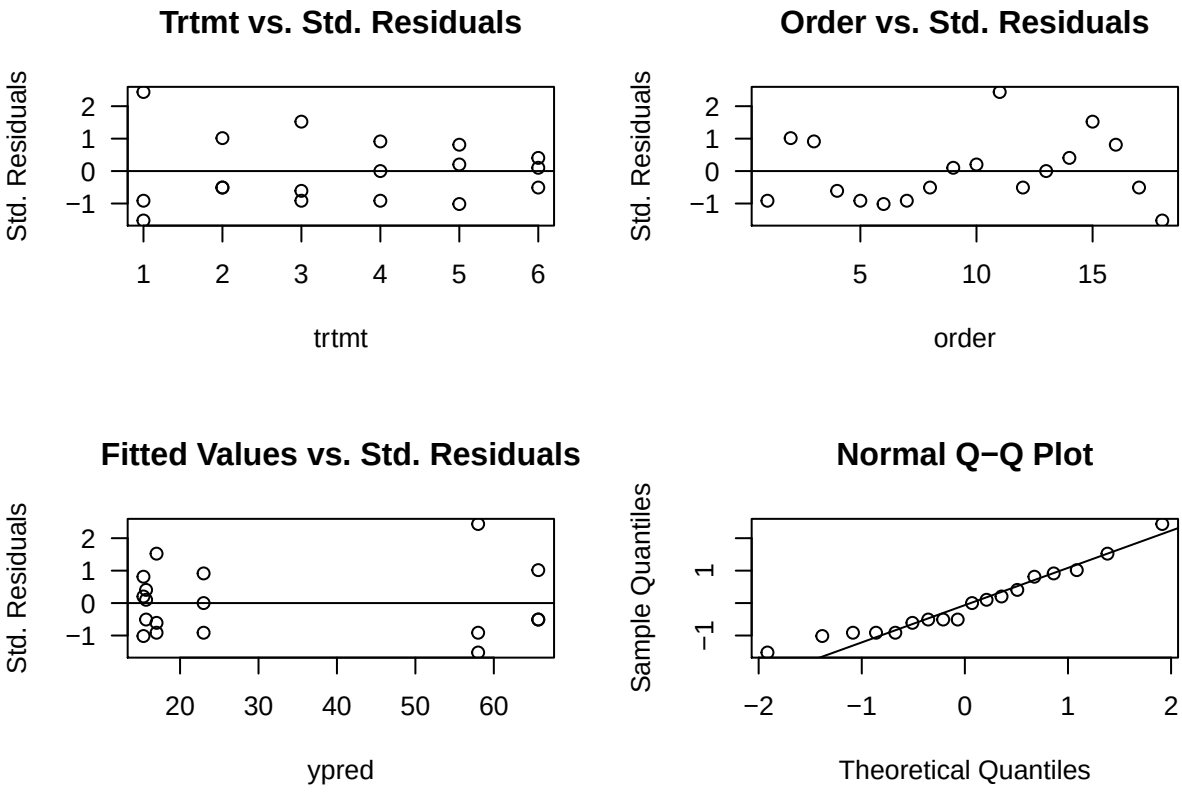
Part of the data collected is shown in Table 5.22. There are three observations per treatment, and the response variable is the weight (in grams) of sauce that flowed through a colander in a given period of time. A thicker sauce would give rise to smaller weights.

(a) Check the assumptions on the one-way analysis of variance model (3.3.1).

```r
spaghetti.data = read.table("~/Desktop/Stats 6910/HW_4_and_5/spaghetti.sauce.txt", heade

spaghetti.model = aov(weight ~ factor(trtmt), spaghetti.data)
# Compute predicted values, residuals, standardized residuals, normal scores
spaghetti.data = within(spaghetti.data, {
  # Compute predicted, residual, and standardized residual values
  ypred = fitted(spaghetti.model)
  e = resid(spaghetti.model)
  z = rstandard(spaghetti.model)})
# Display first 10 lines of mung.data, 4 digits per variable
print(head(spaghetti.data, 10), digits=4)
```

```
##    order trtmt weight brand stir       z       e ypred
## 1      1     3     14     2    1 -0.9138 -3.0000 17.00
## 2      2     2     69     1    2  1.0153  3.3333 65.67
## 3      3     4     26     2    2  0.9138  3.0000 23.00
## 4      4     3     15     2    1 -0.6092 -2.0000 17.00
## 5      5     4     20     2    2 -0.9138 -3.0000 23.00
## 6      6     5     12     3    1 -1.0153 -3.3333 15.33
## 7      7     1     55     1    1 -0.9138 -3.0000 58.00
## 8      8     6     14     3    2 -0.5077 -1.6667 15.67
## 9      9     6     16     3    2  0.1015  0.3333 15.67
## 10    10     5     16     3    1  0.2031  0.6667 15.33
```

```r
# Generate residual plots

par(mfrow = c(2,2))
plot(z ~ trtmt, data=spaghetti.data,
      main= "Trtmt vs. Std. Residuals",
     ylab="Std. Residuals", las=1)
abline(h=0)  # Horizontal line at zero
plot(z ~ order, data=spaghetti.data,
     main= "Order vs. Std. Residuals",
     ylab="Std. Residuals", las=1)
abline(h=0)
plot(z ~ ypred, data=spaghetti.data,
      main= "Fitted Values vs. Std. Residuals",
     ylab="Std. Residuals", las=1)
abline(h=0)
qqnorm(spaghetti.data$z)
# Line through 1st and 3rd quantile points
qqline(spaghetti.data$z)
```



Assumption (a): The error have mean 0:

Due to the formulation of the One Way ANOVA Model, the residuals always sum up to 0. Hence, the assumption cannot be checked.

Assumption (b): The error have constant variance:

```
Y_i_var <- tapply(spaghetti.data$weight, spaghetti.data$trtmt, var);
max(Y_i_var)/min(Y_i_var)
```

```
## [1] 21
```

Using the rule of thumb based on within group variances we can see that

$$\frac{\max_i\{S_i^2\}}{\min_i\{S_i^2\}} = \frac{49}{2.33} = 21.$$

Furthermore, by plotting the standardized residuals against the fitted values we can see that the spread of the standardized residuals increases as the fitted values increase. Hence, with these two results we feel comfortable to say that the equal variance assumption is not satisfied.

Assumption (c): The error are normally distributed:

From the qq-plot above we see that the data is fairly straight. Furthermore, all the standardized residuals are $|z_i| \leq 2.437$, so we don't have any apparent outliers. Therefore, the normality assumption is reasonable.

```
max(abs(spaghetti.data$z))
```

```
## [1] 2.436831
```

Assumption (d): The errors are independent:

From the plot of Order vs. Residuals we can see that there isn't a pattern as time increases. Therefore we feel comfortable that the errors are approximately satisfies.

(b) Use Satterthwaite's method to obtain simultaneous confidence intervals for the six preplanned contrasts

$$\tau_1 - \tau_2, \tau_3 - \tau_4, \tau_5 - \tau_6, \tau_1 - \tau_5, \tau_1 - \tau_3, \tau_3 - \tau_5,$$

Select an overall confidence level of at least 95%.

We will use Tukey's method of multiple comparison since we want a simultaneous confidence intervals for the preplanned contrasts. Although Tukey's method gives confidence intervals for all pairwise comparison, we will only display the ones requested. Therefore,

A simultaneous confidence interval for $\tau_1 - \tau_2$ is $(-34.96928, 19.63595)$.
A simultaneous confidence interval for $\tau_3 - \tau_4$ is $(-21.516045, 9.516045)$.
A simultaneous confidence interval for $\tau_5 - \tau_6$ is $(-11.70676, 11.04009)$.
A simultaneous confidence interval for $\tau_1 - \tau_5$ is $(15.74657, 69.58676)$.
A simultaneous confidence interval for $\tau_1 - \tau_3$ is $(15.92339, 66.07661)$.
A simultaneous confidence interval for $\tau_3 - \tau_5$ is $(-13.84765, 17.18098)$.

The work is shown in the code below.

```r
# Fitted values
Y_i_hat <- tapply(spaghetti.data$weight, spaghetti.data$trtmt, mean);
# Sample Variance
Y_i_var <- tapply(spaghetti.data$weight, spaghetti.data$trtmt, var);


# Confidence Interval for tau_i - tau_j
CI_Satterthwaite <- function(ti,tj,variance_i,variance_j,r,v){
# Numerator for Degree of Freedom
numerator <- sum(variance_i/r,variance_j/r)^2
# Denominator for Degree of Freedom
denominator <- sum((variance_i/r)^2/(r-1), (variance_j/r)^2/(r-1))
#Degrees of freedom
deg.fr_taui_tauj <- numerator/ denominator
print(deg.fr_taui_tauj)
#Standard error
SE <- sqrt(sum(variance_i/r,variance_j/r))
# w_T
w_T <- qtukey(.95,v,deg.fr_taui_tauj)/ sqrt(2)
# tau_i - tau_j
ti_minus_tj <- ti - tj
#return confidence interval
return(CI_ti_minus_tj <-c(ti_minus_tj - w_T * SE,ti_minus_tj + w_T * SE))
}

t_i <- c(1,3,5,1,1,3)
t_j <- c(2,4,6,5,3,5)
CI_ti_minus_tj <- NULL

for (i in 1:6){
CI_ti_minus_tj<-rbind(CI_ti_minus_tj,
CI_Satterthwaite(Y_i_hat[t_i[i]],Y_i_hat[t_j[i]],
                 Y_i_var[t_i[i]],Y_i_var[t_j[i]],3,6)) }
```

```
## [1] 2.66115
## [1] 3.547511
## [1] 2.941176
## [1] 2.73523
## [1] 3.348298
## [1] 3.582941
```

```r
colnames(CI_ti_minus_tj) <- c("Lower","Upper")
CI_ti_minus_tj
```

```
##          Lower     Upper
```

```
## [1,] -34.96928 19.635948
## [2,] -21.51604  9.516045
## [3,] -11.70676 11.040092
## [4,]  15.74657 69.586762
## [5,]  15.92339 66.076609
## [6,] -13.84765 17.180981
```