STAT 6910: HW 5

David Angeles

```
## Warning: package 'ggplot2' was built under R version 3.4.4

## 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 codes 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
```

```
n_1 <- nrow(Post_grams); n_1
```

```
## [1] 30
```

```
v_1 <- length(unique(Post_grams$Code)); v_1
```

```
## [1] 6
```

```
rs_1 <- tapply(rep(1,n_1) , Post_grams$Code, sum); rs_1
```

```
## 1 2 3 4 5 6
## 5 5 5 5 5 5
```

```
r_i.vector_1 <- rep(rs_1,time =rs_1); r_i.vector_1
```

```
## 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6 6 6
## 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 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
```

```
anova(Post_weight_model)[2,"Mean Sq"]
```

```
## [1] 7.266667
```

```
weights.fitted <- fitted(Post_weight_model); weights.fitted
```

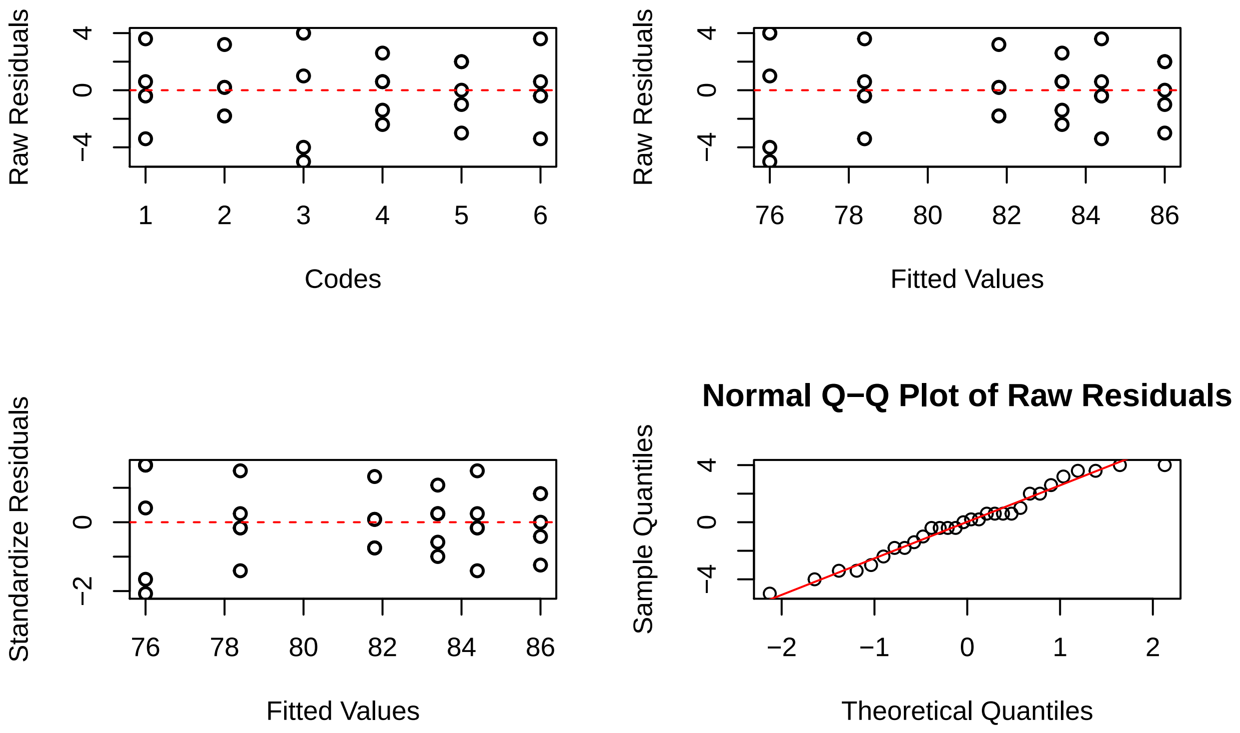
```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 84.4 84.4 84.4 84.4 84.4 81.8 81.8 81.8 81.8 81.8 76.0 76.0 76.0 76.0 76.0
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
## 83.4 83.4 83.4 83.4 83.4 86.0 86.0 86.0 86.0 86.0 78.4 78.4 78.4 78.4 78.4
```

```
weights.raw.resid <- resid(Post_weight_model); weights.raw.resid
```

```
##             1             2             3             4             5
## -3.400000e+00  3.600000e+00  6.000000e-01 -4.000000e-01 -4.000000e-01
##             6             7             8             9            10
##  3.200000e+00 -1.800000e+00  2.000000e-01 -1.800000e+00  2.000000e-01
##            11            12            13            14            15
## -5.000000e+00  1.000000e+00 -4.000000e+00  4.000000e+00  4.000000e+00
##            16            17            18            19            20
##  6.000000e-01  6.000000e-01 -1.400000e+00 -2.400000e+00  2.600000e+00
##            21            22            23            24            25
## -3.000000e+00  2.000000e+00 -1.000000e+00 -1.110223e-16  2.000000e+00
##            26            27            28            29            30
## -4.000000e-01 -3.400000e+00 -4.000000e-01  6.000000e-01  3.600000e+00
```

```
#MSE <- anova(Post_weight_model)[2,"Mean Sq"]; MSE
#std.residuals <- weights.raw.resid/(sqrt(MSE * (1-1/r_i.vector_1))); std.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,
     xlab = "Codes", 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,
     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,
     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")
```



Assuption (a): The error have mean 0:

Due to the formulation of the One Way ANOVA Model, the residuals always sum up to 0.

Assuption (b): The error have constant variance:

By plotting the residuals against the fitted values of hte treatment levels we saw no big difference in the pattern of the spread within the groups. So we feel comfortable that the equal variance assumption is approximately satisfied.

Since the the samples sizes of each group is equal, we can see that the the expected result is strengthens the assumption.

Assuption (c): The error are nrmally distributed:

From the qq-plot above we see that the data is fairly straight and no outliers are apparent. Therefore the normality assumption is reasonable.

Assuption (d): The error are independent:

Since we do not have any information on the

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:

$$1 = \text{store brand, unstirred } 2 = \text{store brand, stirred}$$
$$3 = \text{national brand, unstirred } 4 = \text{national brand, stirred}$$
$$5{=}\text{gourmet brand,unstirred } 6{=}\text{gourmet brand,stirred}$$

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).

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

spaghetti.sauce.data <- as.data.frame(spaghetti.sauce.data)
spaghetti.sauce.data$trtmt <- as.factor(spaghetti.sauce.data$trtmt)



spaghetti.model <- aov(weight ~ trtmt , data = spaghetti.sauce.data)
anova(spaghetti.model)
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## trtmt      5 7976.4 1595.29  98.678 2.548e-09 ***
## Residuals 12  194.0   16.17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Get fitted values from the model
spaghetti.fitted <- fitted(spaghetti.model); spaghetti.fitted
```

```
##        1        2        3        4        5        6        7        8
## 17.00000 65.66667 23.00000 17.00000 23.00000 15.33333 58.00000 15.66667
##        9       10       11       12       13       14       15       16
## 15.66667 15.33333 58.00000 65.66667 23.00000 15.66667 17.00000 15.33333
##       17       18
## 65.66667 58.00000
```

```
# Raw Residuals
spaghetti.raw.residuals <- resid(spaghetti.model); spaghetti.raw.residuals
```

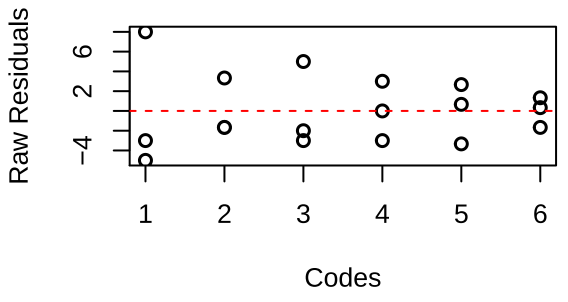
```
##             1             2             3             4             5
## -3.000000e+00  3.333333e+00  3.000000e+00 -2.000000e+00 -3.000000e+00
##             6             7             8             9            10
## -3.333333e+00 -3.000000e+00 -1.666667e+00  3.333333e-01  6.666667e-01
##            11            12            13            14            15
##  8.000000e+00 -1.666667e+00 -6.217249e-15  1.333333e+00  5.000000e+00
##            16            17            18
##  2.666667e+00 -1.666667e+00 -5.000000e+00
```

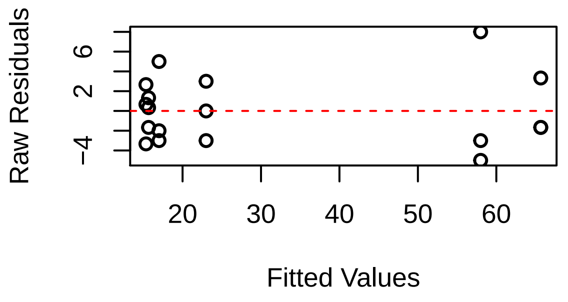
```
# Standardize residuals
spaghetti.stand.residuals <- rstandard(spaghetti.model); spaghetti.stand.residuals
```

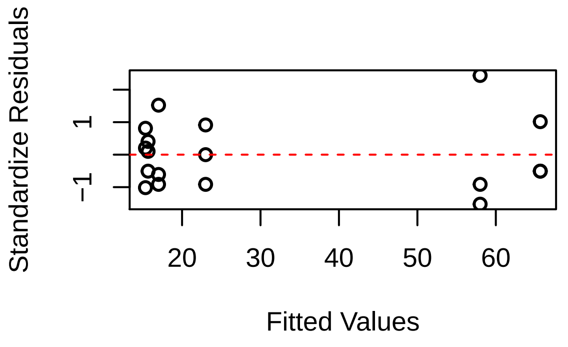
```
##          1          2          3          4          5          6
## -0.9138115  1.0153462  0.9138115 -0.6092077 -0.9138115 -1.0153462
##          7          8          9         10         11         12
## -0.9138115 -0.5076731  0.1015346  0.2030692  2.4368308 -0.5076731
##         13         14         15         16         17         18
##  0.0000000  0.4061385  1.5230192  0.8122769 -0.5076731 -1.5230192
```

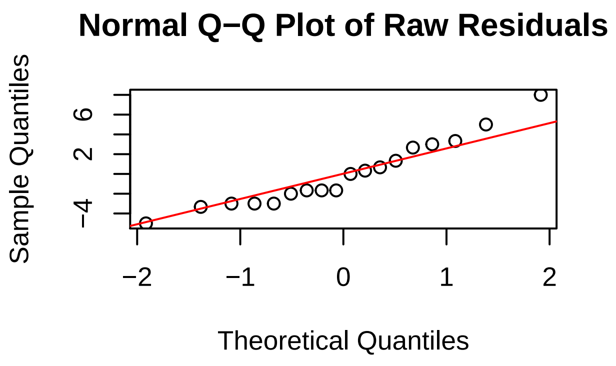
```
par(mfrow = c(2,2))
#Raw Residuals vs. Frying/ Grilling fat content
plot(as.numeric(spaghetti.sauce.data$trtmt) , spaghetti.raw.residuals,
     xlab = "Codes", ylab = "Raw Residuals", xaxt = "n",
     lwd = 1.5)
axis(1, at = 1:6, labels = c("1","2","3","4","5","6"))
abline(h=0, col = "red", lty = 2)

#Raw Residuals vs. Fitted Values
plot(spaghetti.fitted , spaghetti.raw.residuals,
     xlab = "Fitted Values", ylab = "Raw Residuals",
```

```r
     lwd = 1.5)
abline(h=0, col = "red", lty = 2)

#Standardized Residuals vs. Fitted Values
plot(spaghetti.fitted , spaghetti.stand.residuals,
     xlab = "Fitted Values", ylab = "Standardize Residuals",
     lwd = 1.5)
abline(h=0, col = "red", lty = 2)

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









(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 since we want a simultaneous confidence intervals for the six preplanned contrasts. Therefore,

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

A simultaius confidence interval for $\tau_1 - \tau_3$ is $(66.07661, 15.92339)$. A simultaius confidence interval for $\tau_3 - \tau_5$ is $(17.18098, -13.84765)$.

The work is shown in the code below.

```
spaghetti.fitted
```

```
##        1        2        3        4        5        6        7        8
## 17.00000 65.66667 23.00000 17.00000 23.00000 15.33333 58.00000 15.66667
##        9       10       11       12       13       14       15       16
## 15.66667 15.33333 58.00000 65.66667 23.00000 15.66667 17.00000 15.33333
##       17       18
## 65.66667 58.00000
```

```
t_1 <- spaghetti.fitted[7]; t_1 # tau_1
```

```
##  7
## 58
```

```
t_2 <- spaghetti.fitted[2];t_2 # tau_2
```

```
##        2
## 65.66667
```

```
t_3 <- spaghetti.fitted[1]; t_3 # tau_3
```

```
##  1
## 17
```

```
t_4 <- spaghetti.fitted[3];t_4# tau_4
```

```
##  3
## 23
```

```
t_5 <- spaghetti.fitted[6];t_5 # tau_5
```

```
##        6
## 15.33333
```

```
t_6 <- spaghetti.fitted[8];t_6 # tau_6
```

```
##        8
## 15.66667
```

```
#1st treatment variance
var_1 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==1]))^2; var_1
```

```
## [1] 49
```

```
#2nd treatment variance
var_2 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==2]))^2; var_2
```

```
## [1] 8.333333
```

```
#3rd treatment variance
var_3 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==3]))^2; var_3
```

```
## [1] 19
```

```
#4th treatment variance
var_4 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==4]))^2; var_4
```

```
## [1] 9
```

```
#5th treatment variance
var_5 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==5]))^2; var_5
```

```
## [1] 9.333333
```

```
#6th treatment variance
var_6 <- (sd(spaghetti.sauce.data$weight[spaghetti.sauce.data$trtmt ==6]))^2; var_6
```

```
## [1] 2.333333
```

```
# Confidence Interval for tau_i - tau_j
CI_Satterthwaite <- function(t1,t2,variance_1,variance_2,r,v){
 #t1= 9.33; t2= 9.03;variance_1 = 2.95; variance_2 = 1.29; r= 10; v=4

# Numerator for Degree of Freedom
num_1 <- sum(variance_1/r,variance_2/r)^2
# Denominator for Degree of Freedom
den_1 <- sum((variance_1/r)^2/(r-1), (variance_2/r)^2/(r-1))
#Degrees of freedom
deg.fr_tau1_tau2 <- num_1/ den_1
#Standard error
SE_1 <- sqrt(sum(variance_1/r,variance_2/r))
# w_T
w_1 <- qtukey(.95,v,deg.fr_tau1_tau2)/ sqrt(2)
# tau_i - tau_j
t1_min_t2 <- t1 - t2
#return confidence interval
return(CI_t1_min_t2 <-c(t1_min_t2 + w_1 * SE_1,t1_min_t2 - w_1 * SE_1))
}


CI_tau1_min_tau2 <- CI_Satterthwaite(t_1,t_2,var_1,var_2,3,6);CI_tau1_min_tau2
```

```
##         7         7
##  19.63595 -34.96928
```

```
CI_tau3_min_tau4 <- CI_Satterthwaite(t_3,t_4,var_3,var_4,3,6);CI_tau3_min_tau4
```

```
##          1          1
##   9.516045 -21.516045
```

```
CI_tau5_min_tau6 <- CI_Satterthwaite(t_5,t_6,var_5,var_6,3,6);CI_tau5_min_tau6
```

```
##         6         6
##  11.04009 -11.70676
```

```
CI_tau1_min_tau5 <- CI_Satterthwaite(t_1,t_5,var_1,var_5,3,6);CI_tau1_min_tau5
```

```
##        7        7
## 69.58676 15.74657
```

```
CI_tau1_min_tau3 <- CI_Satterthwaite(t_1,t_3,var_1,var_3,3,6);CI_tau1_min_tau3
```

```
##        7        7
## 66.07661 15.92339
```

```
CI_tau3_min_tau5 <- CI_Satterthwaite(t_3,t_5,var_3,var_5,3,6);CI_tau3_min_tau5
```

```
##         1         1
##  17.18098 -13.84765
```