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 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
```

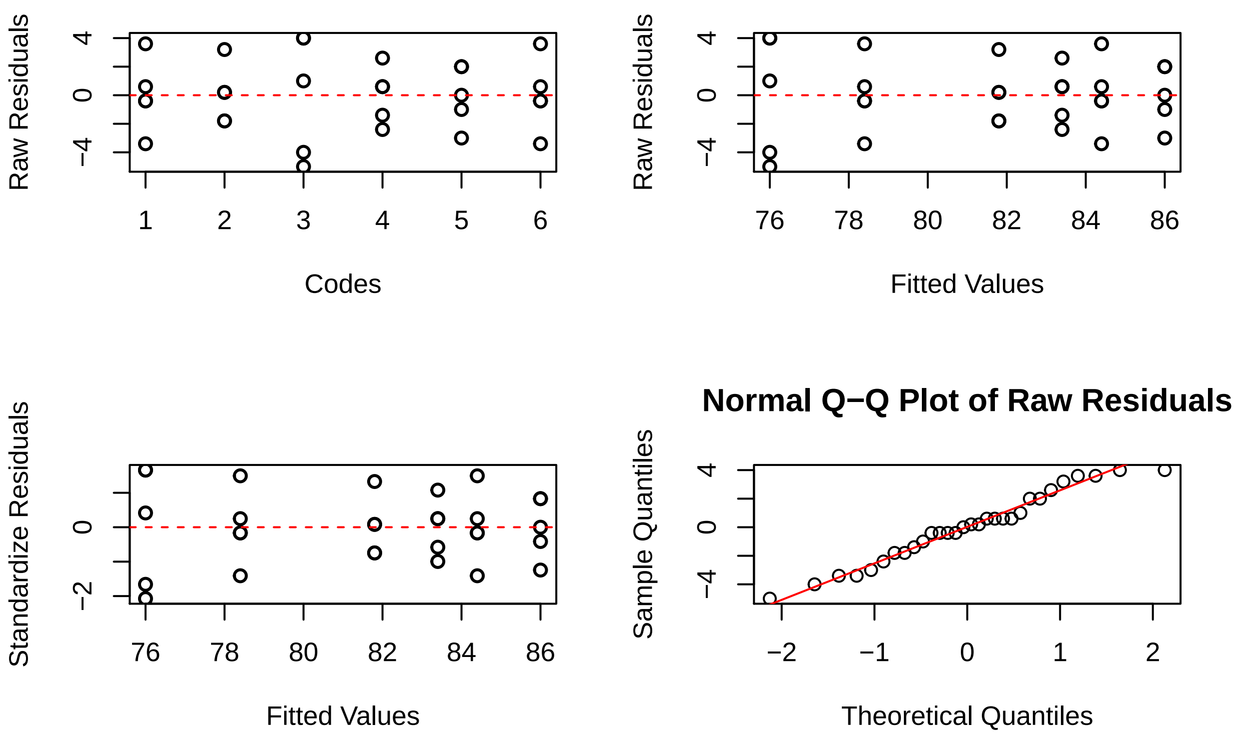
```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 84.4 84.4 84.4 84.4 84.4 81.8 81.8 81.8 81.8 81.8 76.0 76.0 76.0 76.0 76.0
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
## 83.4 83.4 83.4 83.4 83.4 86.0 86.0 86.0 86.0 86.0 78.4 78.4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

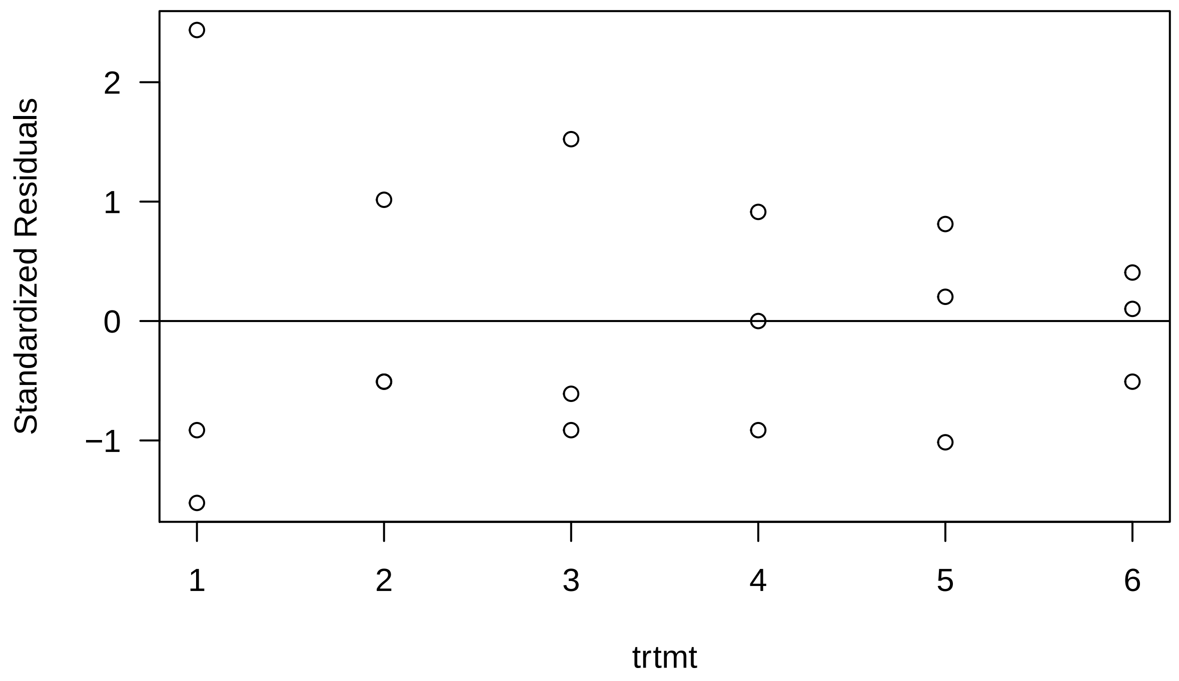
Part of the data collected is shown in Table 5.22. There are three observations per treatment, and the response variable is the weight (in grams) of sauce that flowed through a colander in a given period of time. A thicker sauce would give rise to smaller weights.

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

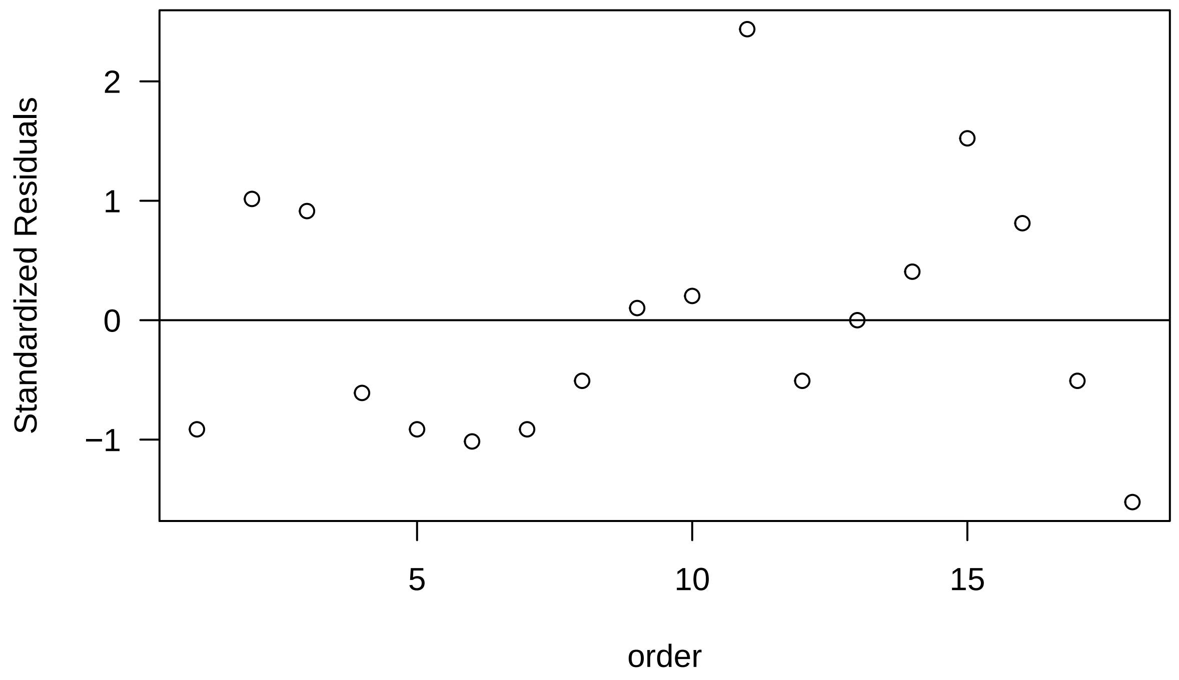
```
spaghetti.data = read.table("~/Desktop/Stats 6910/HW_4_and_5/spaghetti.sauce.txt", heade

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
```

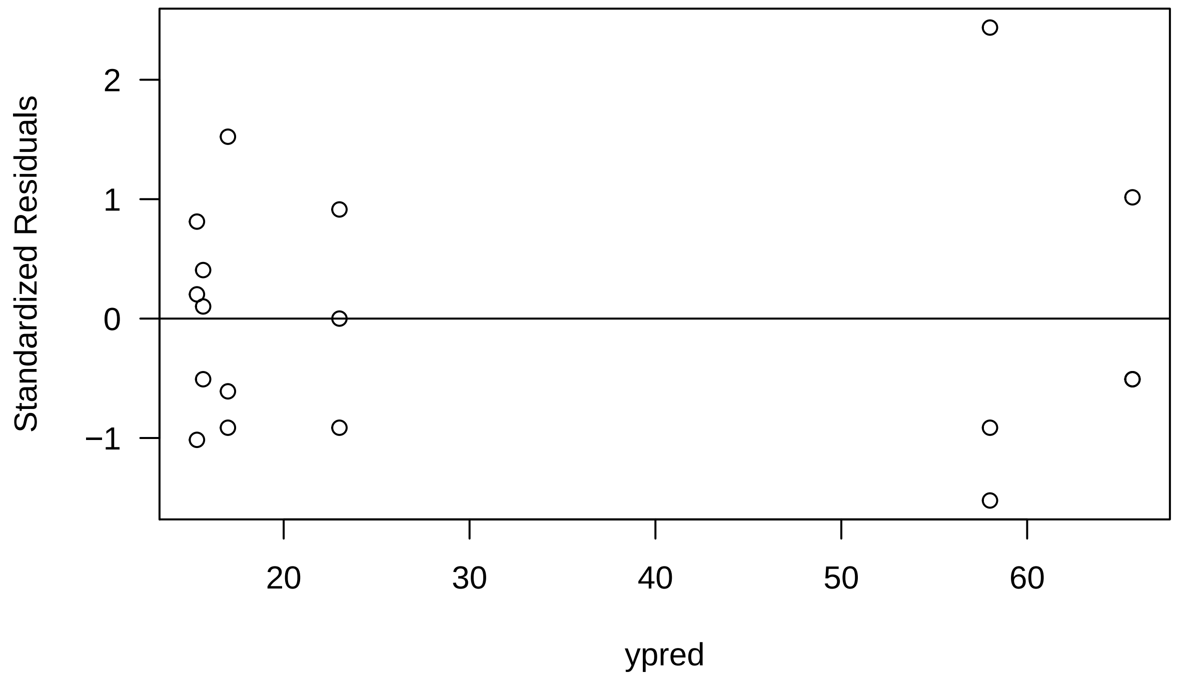
```
# Generate residual plots


plot(z ~ trtmt, data=spaghetti.data, ylab="Standardized Residuals", las=1)
abline(h=0)  # Horizontal line at zero
```



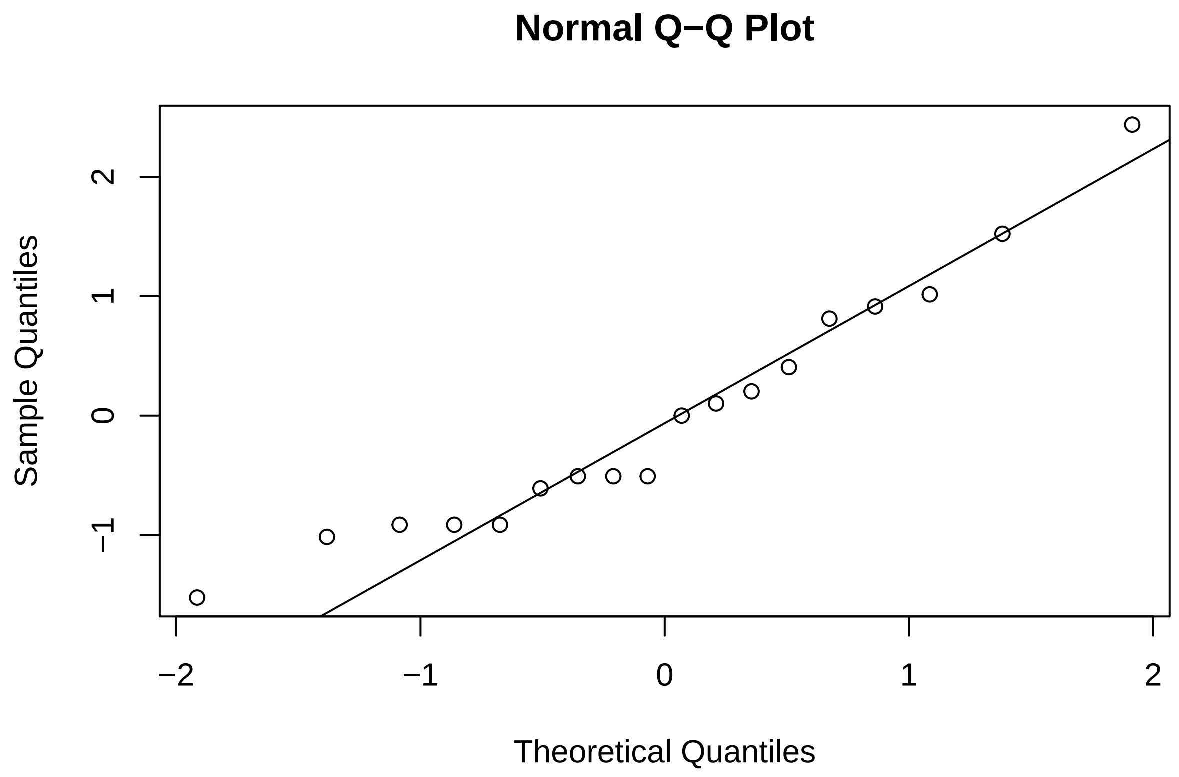
```
plot(z ~ order, data=spaghetti.data, ylab="Standardized Residuals", las=1)
abline(h=0)
```



```r
plot(z ~ ypred, data=spaghetti.data, ylab="Standardized Residuals", las=1)
abline(h=0)
```



```r
qqnorm(spaghetti.data$z)
# Line through 1st and 3rd quantile points
qqline(spaghetti.data$z)
```



```r
# Compute sample means and variances and their natural logs by trtmt
MeanWeight = by(spaghetti.data$weight, spaghetti.data$trtmt, mean) # Sample means
VarWeight = by(spaghetti.data$weight, spaghetti.data$trtmt, var) # Sample variances
LnMean = log(MeanWeight)  # Column of ln sample means
LnVar = log(VarWeight)  # Column of ln sample variances
Trtmt = c(1:6)  # Column of trtmt levels
stats = cbind(Trtmt, MeanWeight, VarWeight, LnMean, LnVar) # Column bind
stats  # Display the stats data
```

```
##   Trtmt MeanWeight VarWeight   LnMean     LnVar
## 1     1   58.00000 49.000000 4.060443 3.8918203
## 2     2   65.66667  8.333333 4.184591 2.1202635
## 3     3   17.00000 19.000000 2.833213 2.9444390
## 4     4   23.00000  9.000000 3.135494 2.1972246
## 5     5   15.33333  9.333333 2.730029 2.2335922
## 6     6   15.66667  2.333333 2.751535 0.8472979
```

(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.

```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
#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)) }



colnames(CI_ti_minus_tj) <- c("Left","Right")
CI_ti_minus_tj
```

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