Replicated 2^5 Complete Factorial Design to decide significant factors affecting taste of Chai

Abhijeet Bhardwaj

April 30, 2021

1 Summary

This report is a summary of a replicated 2^5 factorial experiment. The experiment is conducted to decide the factors that affect the taste of chai (Indian milk tea). The factors studied in the experiment include the presence of ginger (factor A), spice (factor B), amount of sugar (factor C), material of utensil (D) and simmer time (E). The experiment is conducted on four different days, making day a block variable. Scores from three judges (AD, AG, KJ) are analyzed by combining them and also individually. A difference in scores of two judges is also studied to gain more insights. The tasting order for each judge is randomized separately. The data is analyzed in stages and detailed analysis is provided through code included in the appendix. It is observed that only the factor sugar (C) turns out to be the significant for Judge KJ, AD while the factor sugar (C) (with and without transformation) and simmer time (E) (with transformation) turns out to be significant for judge AG and also on average. It is also observed that no factor turns out to be significant for the analysis on difference in scores, which suggest that the significant factors are indeed specific to each individual.

2 Introduction

For street vendors in India, selling chai has been a very prevalent business and source of earning money. Chai is a beverage in which processed tea leaves are first boiled in water and then milk is added to it. After adding the milk, the mixture is again brought to a boil and is kept to simmer for some time. Many ingredients along with the tea leaves can be added in water while preparing chai, which enhance the taste of chai. In this experiment we would like to study the effects of various factors over taste of chai. The factors that we consider for our study are shown in Table 1. The reason behind choosing each of them are as follows:

- **Ginger (A)**: Adding Ginger to the tea has been traditionally considered effective in increasing the medicinal benefits of chai. However apart from its medicinal benefits we would also like to study how does it affect its taste.
- **Spice (B)**: Some people prefer to drink chai as a spice beverage termed as masala chai in India. However it would be interesting to see if this is generally preferred.
- **Sugar (C)**: Chai is a sweet drink and is normally prepared with sugar unlike brewed coffee. However, the amount of sugar may change the taste of chai for each individual differently.
- **Utensil (D)**: Heating/ boiling chai is preferred so that all the extract from the tea leaves is absorbed in water. Different kind of utensils have different heat conductivity. Thus it would be reasonable to assume that Aluminum utensil would conduct more heat than Steal utensil based on its higher conductivity and hence we could study its interaction in making chai more tasty.

- **Simmer Time (E)**: To prepare chai first the mizture is boiled in water without milk. After the chai mixture has come to a boil milk is added to it and the mixture is boiled again. However to have better taste the chai is left to simmer after the milk mixture is brought to a boil. Thus higher simmer times could intuitively have better taste for the chai.

Table 1: Factors studied in the analysis and their levels

A	B	C	D	E
Ginger	Spice	Sugar	Utensil	Simmer Time
low= 0 gms; high = 5 gms	low=0 gms; high =0. 5 gms	low=1 tbs; high =2 tbs	low = Aluminium utensil; high =Steel utensil	low = 3.5 mins; high = 7.0 mins

3 Steps to perform and some pre-analysis

To perform the scoring, I invited three judges (denoted by AD, AG and KJ). I wished to conduct a replicated 2^5 factorial experiment to have better judgment of the variance in the model to generate confidence intervals. Thus, a total of 64 samples were prepared. As chai is acidic by nature, it was not possible to taste all the chai at once or even in two occasions. Thus I have to block each of my replicate. I prepared 16 samples each day on four different days. The judges were asked to provide a score between 1-10, where decimal values of 0.5 were allowed to minimize ties. The tasting order for chai was randomized for each judge seperately by picking a chit from bowl. I also, randomized my preparation order for chai so that other random factors like burner (stove) fatigue may be compensated for.

For each run I measured the same precise amount of water and milk by having the same marking over the glass. I used the same brand milk every time which was taken out of the refrigerator for each run. I also measured the exact amount of ginger and spice mix using a food weighing machine and used the same table spoon while adding sugar in the chai. Preparing chai involved 3 steps. First, we bring the mixture and water to a boil which approximately takes around 4 minutes, then we add milk to the mixture and bring it to a boil again which additionally takes around 3 minutes. After having the second boil, the mixture is left to simmer. Thus, the time that I was going to take to prepare each sample, was dependent on the amount of simmer time that I choose. Thus, I did a small pre analysis to decide the levels for simmer time. I asked one of the judges to try two different sets of chai (anonymously) prepared at two simmer levels (rest of the factors being same). The first set of chai was simmered at 5 mins and 10 mins levels, while the second set was simmered at 3.5 mins, 7 mins. For both the sets the judge always preferred the one at higher simmer level. Thus to reduce the time for each run, I choose the levels of simmer time to be 3.5 mins and 7 mins. Finally, it took me around 4 hours each day to prepare the 16 chai samples and an additional hour for scoring and cleaning. I thus spent a total of around 20 hours to perform the complete experiment. The cost of the ingredients were $3.19 \times 4 = 12.76$ \$ for milk, 6\$ for ginger, 3\$ for spice mix.

I perform a replicated 2^5 complete factorial design. For the first replicate, I choose my block variable to be the fifth factor interaction term ABCDE and thus I can not have an estimate of ABCDE from the first replicate. However, in the second replicate I use the variable ABDE as a block, so that ABCDE can be estimated from the second replicate, and, ABDE gets estimated from the first replicate.

After performing the experiment for Day 1, I have scores for the first block of first replicate. The scores for Day 1 and Day 2 along with tasting order are given in Table 2 where the scores and tasting orders are arranged by block variable ABCDE. I analyze the average scores given by three judges by treating it as a 2^{5-1} fractional factorial in-order to gain initial insights about the estimates of various main effects. I would denote the term 'factor interaction' by 'fi'. The generator in this case is I=ABCDE, thus main effects are represented as E=ABCD $\cdots$ while 2 fi's are represented as AB = CDE $\cdots$ and thus all the 3 fi's are confounded with the 2 fi's while the main effects are confounded with 4 fi's. However, we can assume the 3 fi's and 4 fi's to be zero to estimate the main effects and the 2 fi's. Also, I perform the Daniel plot analysis with Loh correction Loh (1992) to identify the significant

effects. As this is a preliminary analysis I have included the plots and estimated coefficients in Appendix A.1. It turns out that only the factor sugar (C) is significant from the Daniels plot analysis. Although not significant, but other factors to look up-to were factors D, E and interaction term CE.

Table 2: Scores from three judges and their run orders for first replicate. The table headings like J_AD would imply the scores from Judge AD and the headings like O_AD is the order in which the judge AD taste each sample. Please note the design matrix is not arranged in standard order but is grouped by Days/block variable ABCDE to have a better understanding of the run order for each day.

A	**B**	**C**	**D**	**E**	**Block (ABCDE)**	**Run day**	**J_AD**	**O_AD**	**J_AG**	**O_AG**	**J_KJ**	**O_KJ**
-1	-1	-1	-1	-1	-1	DAY1	6	11	4	13	4	10
1	1	-1	-1	-1	-1	DAY1	6	12	3.5	6	1.5	7
1	-1	1	-1	-1	-1	DAY1	6.5	2	7.5	5	8	11
-1	1	1	-1	-1	-1	DAY1	7	14	6.5	12	8.5	1
1	-1	-1	1	-1	-1	DAY1	4	10	3	3	1	12
-1	1	-1	1	-1	-1	DAY1	4	3	3	14	2	4
-1	-1	1	1	-1	-1	DAY1	8	16	6.5	11	4	3
1	1	1	1	-1	-1	DAY1	6	8	6	1	8.5	15
1	-1	-1	-1	1	-1	DAY1	3	5	6	16	5	5
-1	1	-1	-1	1	-1	DAY1	7	15	3	7	5.5	14
-1	-1	1	-1	1	-1	DAY1	7	6	7	19	5	8
1	1	1	-1	1	-1	DAY1	8	7	7	10	8.5	16
-1	-1	-1	1	1	-1	DAY1	5.5	4	4	2	2	9
1	1	-1	1	1	-1	DAY1	5	13	5.5	8	7.5	13
1	-1	1	1	1	-1	DAY1	7	1	8	4	6	6
-1	1	1	1	1	-1	DAY1	7	9	7.5	15	4.5	2
1	-1	-1	-1	-1	1	DAY2	7	4	2	11	4.5	3
-1	1	-1	-1	-1	1	DAY2	7.5	3	5.5	14	4	1
-1	-1	1	-1	-1	1	DAY2	7	1	2	12	1.5	10
1	1	1	-1	-1	1	DAY2	7	12	5	8	7.5	12
-1	-1	-1	1	-1	1	DAY2	7	8	4	3	3.5	8
1	1	-1	1	-1	1	DAY2	5.5	16	3	4	3	9
1	-1	1	1	-1	1	DAY2	9	10	6	7	7	15
-1	1	1	1	-1	1	DAY2	7.5	2	6	2	7.5	4
-1	-1	-1	-1	1	1	DAY2	7.5	13	6.5	16	8	16
1	1	-1	-1	1	1	DAY2	5.5	7	2.5	13	7	5
1	-1	1	-1	1	1	DAY2	8	11	6.5	9	6.5	11
-1	1	1	-1	1	1	DAY2	6.5	15	6	5	7.5	13
1	-1	-1	1	1	1	DAY2	6	6	4	1	4	2
-1	1	-1	1	1	1	DAY2	6.5	9	4	6	5.5	12
-1	-1	1	1	1	1	DAY2	5.5	14	5	15	7	7
1	1	1	1	1	1	DAY2	8.5	5	7	10	8	6

4 Analyzing the data

4.1 Un-replicated analysis using first replicate

After performing the experiment for the second block of the first replicate, I analyzed the combined data as un-replicated complete 2^5 factorial design. As we have scores from three different judges, I analyze the data from the three judges by averaging them, and also individually. I also analyze obtained, as difference between scores from judge AG and AD and refer to it as difference score. Please note that the table and figures shown in the main report are for the analyses performed using averaged score as response, while the readers are directed to suitable appendix sections for the analyses of individual scores and the difference scores. Apart from the block variable ABCDE, all other factors were estimable from the un-replicated analysis. The full model is shown in equation 1. The estimates of the effects for all the factors with average scores (excluding block variable ABCDE) are shown in Table 3. The estimates of effects for all factors of judge AD,AG, KJ and difference scores are is shown in Appendix

A.3.1.2, A.4.1.2, A.5.1.2, A.6.1.2.

$$y = \mu + \frac{A}{2}X_1 + \frac{B}{2}X_2 + \frac{C}{2}X_3 + \frac{D}{2}X_4 + \frac{E}{2}X_5 + \frac{AB}{2}X_1X_2 + \frac{BC}{2}X_2X_3 + \frac{CD}{2}X_3X_4 + \cdots + \frac{ABC}{2}X_1X_2X_3 + \frac{BCD}{2}X_2X_3X_4 + \cdots + \frac{ABCD}{2}X_1X_2X_3X_4 + \frac{BCDE}{2}X_2X_3X_4X_5 + \cdots + \frac{ABCDE}{2}X_1X_2X_3X_4X_5 + \epsilon$$

$$\epsilon \sim N_{i,i,d}(0, \sigma) \quad (1)$$

4.1.1 Testing Significance

To find the significant effects from the unreplicated data, we use the simultaneous Bonferroni Confidence Interval (CI). I choose a 90% Bonferroni CI to check which variable are significant (for which zero is not included in the upper and lower bounds). I first use the 4 fi's to estimate the variance. Note that I cannot use 5 fi because it is confounded with block effect. Thus I have have 5 d.o.f to estimate variance, while there 31-1-5 terms to be estimated the confidence interval threshold is given by $\sigma_{4fi} \times t_{5,0.1/2/25}$. For zero to not be included in the CI, the absolute values of the factors estimate should be greater than the interval threshold. For average scores, the threshold value is 1.787 and only the main factor C turns out to be significant while for individual scores by Judge AD, AG and KJ none of the factor turn up to be significant (Appendix section A.3.1.2, A.4.1.2, A.5.1.2, A.6.1.2). We thus relax the threshold by assuming that even 3fi's are also zero and use them to estimate variance. Now the interval threshold is given by $\sigma_{3,4fi} \times t_{15,0.1/2/25}$ and for model with average score as response variable its value is 1.1179. Using this we again find that factor C is significant, when average scores is used as response, also for each of the three judges individually we get factor C to be significant (Appendix section A.3.1.2, A.4.1.2, A.5.1.2). However, for difference scores as response, we have no significant factors (Appendix Section A.6.1.2).

Table 3: Estimates of effects after the first replicate for analysis with average score

Factor	Estimate	Factor	Estimate	Factor	Estimate	Factor	Estimate	Factor	Estimate	Factor	Estimate
A	0.229	**A:B**	-0.104	**B:D**	0.0833	**A:B:C**	-0.54167	**A:D:E**	0.395833	**A:B:C:D**	0.0625
B	0.396	**A:C**	0.833	**B:E**	0.04167	**A:B:D**	0.333333	**B:C:D**	-0.39583	**A:B:C:E**	0.354167
C	2	**A:D**	0.25	**C:D**	0.521	**A:B:E**	0.5	**B:C:E**	-0.14583	**A:B:D:E**	-0.6875
D	-0.292	**A:E**	0.167	**C:E**	-0.396	**A:C:D**	-0.39583	**B:D:E**	0.520833	**A:C:D:E**	-0.04167
E	0.75	**B:C**	0.333	**D:E**	-0.104	**A:C:E**	-0.10417	**C:D:E**	-0.33333	**B:C:D:E**	-0.16667

As the number of significant factors turns out to be pretty low for both the confidence interval thresholds, we analysis the data again using Daniels plot with Loh correction Loh (1992). We use the code to select the best set of labels which give minimum median values of the estimates (as can be seen in code of Appendix section A.2.1.3, A.3.1.3, A.4.1.3, A.5.1.3). We show the Q-Q plot for the selected labels when response variable is average scores in Figure 1. It can be seen that factor C is surely an outlier with a value of -2 (Appendix section A.2.1.3). To confirm the same analytically, we find the estimate values that lie outside the interval formed by $1.5 \times IQR$ (Interquartile Range) and also follow the steps recommended in Loh (1992) to get the final significant estimates. Using the analysis we find out that the factor C is significant when we have average scores as response variable. For models with individual scores by judge AD and judge AG the significant factor after Daniel plot analysis is factor C (sugar) while for model with scores by judge KJ there is no significant factor after the Daniel plot analysis. To get a better confirmation for the same, I also performed Dong's analysis to find out the significant factors. As shown in Appendix section A.2.1.4, A.3.1.4, A.4.1.4, A.5.1.4 we get the same significant factors which we got from the Daniel plot analysis. Thus for the model with average scores as response, as well as for models with score from judge AG and AD as response we get factor C (sugar) as the significant factor while for model having scores from judge KJ as response we do not get any significant effect. For model having difference score (score of judge AG - score of judge AD) we find that no factor turns out to be significant using either Daniels plot analysis or Dongs method (Appendix section A.6.1.4).

The diagnostic plots for the reduced model $y = C/2X_3 + \epsilon$ for the average score as response is shown in Figure 2a. The diagnostic plots suggests that the assumptions of the linear regression model (normality, constant variance)are

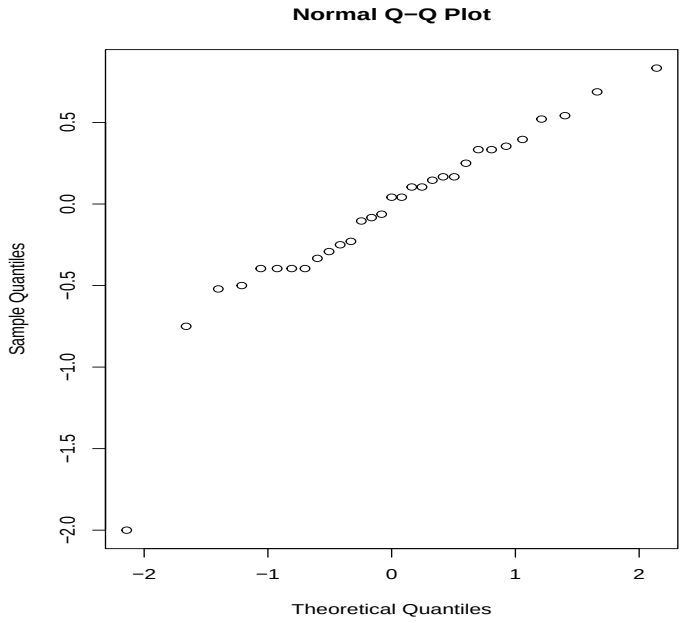


Figure 1: Daniel Plot with Loh (1992) corrections.

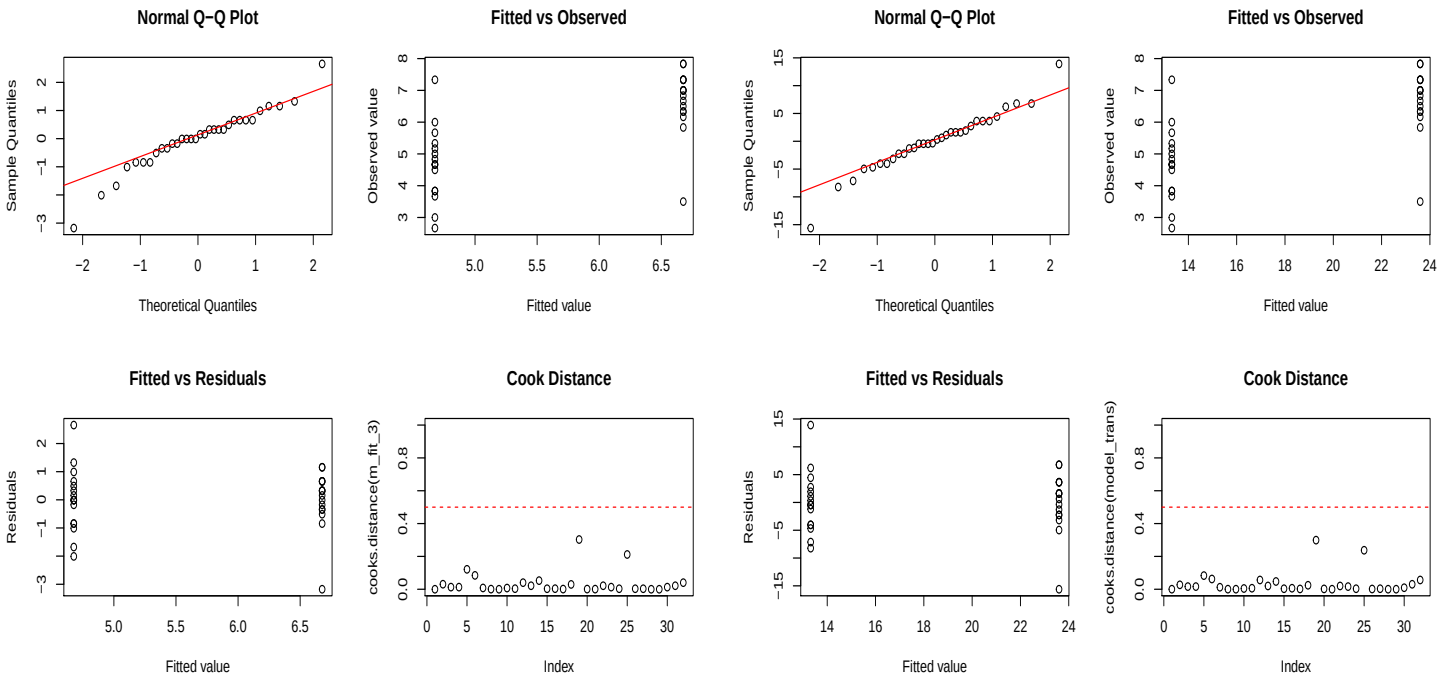
met quite well for the reduced model. Also, there are no outliers present in the data-set. However, we apply box-cox transformation to see if the assumptions can be improved a bit more (for example the residual versus fit plot shows that for high values of sugar (C) ($X_3 = 1$) the residuals are compactly tied with each other while this is not the case for lower levels of sugar (C)). Box-Cox transformation works well when the model is additive in nature, also unless the main factor are significant having their interaction to be significant would be weird. Thus we input the linear additive model in box-cox transformation as $box - cox\{y = X_1 + X_2 + X + 3 + X_4 + X_5 + block\}$ here *block* is the 5^{th} fi term. The assumptions of the model are improved a bit (Figure 2b) as the QQ plot shrinks a bit more towards the straight line and also the residuals ease a bit on the high levels of factor C in the residual versus fitted plot. However, no other factor become significant for the model having average score as the response variable. However, it must be pointed out that, after transformation, the model having scores from judge KJ as response , (which initially had no significant factors), shows significance for factor C (sugar). This is verified by the *adjusted* p-values as shown in Appendix A.5.1.5. The final summary of significant factors for each model with different response variables is shown in Table 4.

Table 4: Significant factors for each model after analysis with un-replicated data

Model #	Response Variable	Significant factors without transformation	Significant factors after transformation
1	**Average score**	C (sugar)	C (sugar)
2	**Judge AD score**	C (sugar)	C (sugar)
3	**Judge AG score**	C (sugar)	C (sugar)
4	**Judge KJ score**	–	C (sugar)
5	**Difference score (AG-AD)**	–	Not performed

4.2 Lessons learnt after analyzing first replicate

After completing the analysis for the first replicate, I observed that the factor sugar was overpowering every other factor present in the experiment. Even-though, decimal values were allowed for scoring the taste of chai, I found out that there were large number of ties present in the scores for a single day run. This lead to small amount of variance in the response variable, which was easily explained by the factor C. On consulting with the judges, they informed me that, tie in scores occurred if they were really not able to get the difference ,or if they gave that



(a) Diagnostic Plots for reduced model having average scores as response

(b) Diagnostic Plots for reduced model after Box-Cox transformation having average scores as response

Figure 2: Diagnostic Plots for reduced model with average score as response variable for un-replicated data

score earlier, and during the course of the experiment the memory of the initial taste got fade away. These problems would naturally arise if we have large block size, especially for a tasting experiment. Apart from having small block size, I also learned that having ranks as a response variable, for my samples would have helped me a lot to over come the issue of small variance in the model. However, instead of changing my design for the experiment, I decided to complete the second replicate with the same design (due to the constraints of time as well) and better estimate the confidence intervals.

4.3 Analysis using second replicate

For the second replicate, I used the 4^{th} fi ABDE as the block variable, for two days. The response for each judge and their tasting orders for the second replicate is shown in Table 5, which are grouped by the block factor ABDE. The reason to choose a different block factor was to get some estimates for the 5^{th} fi ABCDE without confounding. The estimated effects for all the factors (excpet ABDE) in the second replicate are shown in Table 6 for the model having average score as response variable. For the estimates of effects for judges AD, AG, KJ and Difference scores please refer to the Appendix section (A.3.2.1, A.4.2.1, A.4.2.1, A.6.2.1). The final estimates for the effects of all the factors are given by the average of the two estimates in each run (apart from the block variables, which only have single estimates from each run). The final estimates for all the factors are given in the Table 7.

4.3.1 Testing significance of factors

Having two different replicates allows us to have sufficient of d.o.f to estimate standard deviation (σ) of the residuals. I calculated the standard deviation using two different methods. In the first method, I used the first principles, where each complete replicate gave me a complete block to estimate the effects $A_i, i \in \{1, 2\}$ and thus provided their average $\bar{A} = \frac{A_1+A_2}{2}$, which helped to give an estimate of variance for effects as $\hat{var}(\hat{A}_i) = \frac{\sum_{i=1}^{2}(A_i-\bar{A})^2}{2-1}$. I thus have 29 such estimates of variance (the factors ABCDE and ABDE do not provide me estimate of variance as they are blocked), averaging them which give me the estimate of variance of each effect $\sigma_{effects} = \frac{\sum_{i=1}^{29} \hat{var}(\hat{effect}_i)}{29} =$

Table 5: Scores from three judges and their run orders for second replicate. The table headings like J_AD would imply the scores from Judge AD and the headings like O_AD is the order in which the judge AD taste each sample. Please note the design matrix is not arranged in standard order but is grouped by Days/block variable ABDE to have a better understanding of the run order for each day.

A	B	C	D	E	Block (ABDE)	Run day	J_AD	O_AD	J_AG	O_AG	J_KJ	O_KJ
1	-1	-1	-1	-1	-1	DAY3	4	7	4	7	1	2
-1	1	-1	-1	-1	-1	DAY3	6	12	5.5	8	1.5	9
1	-1	1	-1	-1	-1	DAY3	7.5	2	6.5	14	4	13
-1	1	1	-1	-1	-1	DAY3	8	1	7	6	4.5	10
-1	-1	-1	1	-1	-1	DAY3	5	10	4.5	12	3	8
1	1	-1	1	-1	-1	DAY3	3.5	14	6	3	1	14
-1	-1	1	1	-1	-1	DAY3	8.5	6	7	9	3	7
1	1	1	1	-1	-1	DAY3	6.5	8	6.5	5	5.5	15
-1	-1	-1	-1	1	-1	DAY3	5.5	16	5.5	4	1.5	11
1	1	-1	-1	1	-1	DAY3	5	3	5	15	4.5	4
-1	-1	1	-1	1	-1	DAY3	8	15	8	1	7.5	16
1	1	1	-1	1	-1	DAY3	7.5	9	8	13	8	1
1	-1	-1	1	1	-1	DAY3	5	13	5	2	1	5
-1	1	-1	1	1	-1	DAY3	6	4	5	11	2.5	3
1	-1	1	1	1	-1	DAY3	7	5	7	16	7	6
-1	1	1	1	1	-1	DAY3	7.5	11	7	10	2.5	12
-1	-1	-1	-1	-1	1	DAY4	3	5	2	10	5	1
1	1	-1	-1	-1	1	DAY4	2	9	1	15	6	2
-1	-1	1	-1	-1	1	DAY4	9	13	7	12	7	4
1	1	1	-1	-1	1	DAY4	7	3	6	6	8	8
1	-1	-1	1	-1	1	DAY4	7.5	11	6	3	6.5	12
-1	1	-1	1	-1	1	DAY4	4	7	4	5	3	3
1	-1	1	1	-1	1	DAY4	8.5	16	6.5	7	8.5	10
-1	1	1	1	-1	1	DAY4	7	1	5	13	7	14
1	-1	-1	-1	1	1	DAY4	9	14	1.5	11	4	6
-1	1	-1	-1	1	1	DAY4	5	8	2.5	16	6	15
1	-1	1	-1	1	1	DAY4	6	12	7.5	2	9.5	9
-1	1	1	-1	1	1	DAY4	8	2	9	1	5	16
-1	-1	-1	1	1	1	DAY4	2	15	5.5	4	4	7
1	1	-1	1	1	1	DAY4	5	6	1.5	14	6	13
-1	-1	1	1	1	1	DAY4	7	10	7	8	6	5
1	1	1	1	1	1	DAY4	8	4	8	9	8.5	11

$\frac{4\sigma^2}{2^5}$. The confidence interval for each factor can then be constructed using some multiplier (M) as $\{\bar{A}+/-M\times\sqrt{\frac{\frac{4\sigma^2}{2^5}}{2}}\}$. For the second method of estimating stadard deviation for each effect, I fit the full linear model having all interactions included along with a block variable, which I define to be a single categorical variable having 4 levels for each day. Then I use the anova function in R to get the estimate for σ^2 using which we get the estimate for standard deviation for each effect as $\sqrt{\frac{4\times\sigma^2}{64}}$. The standard deviation calculated for each effect using both the methods match exactly upto 5 decimal places thus confirming the accuracy of the methods.

I construct Bonferroni as well as Studentized maximum modulus Confidence Intervals (CI) at 90% confidence level. The Bonferroni multiplier is given by $t_{\nu,0.1/2/g}$ where g is the number of effects to estimate and ν is the degree of freedom. In our case the value of $g = 31$ as we have 31 effects to estimate, while the value for $\nu = 64-31-3-1 = 29$ as we have 3 d.o.f for block factors and 1 intercept term and 31 effects. Similarly the Studentized Maximum Modulus (SMM) coefficient is calculated as $M_{g,\nu}^{(0.1)}$ for 90% confidence level. We use the table for $\alpha = 0.1$ from Stoline & Ury (1979) and interpolate for our value of g and ν by using the four corner values (3.156,3.251,3.104,3.196) as shown in the code in Appendix section A.2.2.2. Confidence interval is constructed using estimates of standard deviation from both the methods and using both the multipliers as shown in Appendix section A.2.2.2, A.3.2.2, A.4.2.2, A.5.2.2 and A.6.2.2. The Bonferroni and SMM confidence interval for the model having average of scores as response variable is given in Table 8. By observing the confidence interval it turns out that only the factor C is significant for all the models, except for the model with difference score as response, for which we have no significant effect. It should be mentioned that block factor also turns out to be significant as can be seen in the confidence intervals depicted

Table 6: Estimates of effects after the second replicate for analysis with average score as response variable

Factors	**Estimate**	**Factors**	**Estimate**	**Factors**	**Estimate**	Factors	**Estimate**	Factors	**Estiamtes**	Factors	**Estimate**
A	0.2917	**A:B**	-0.0625	**B:D**	-0.2917	**A:B:C**	0.4583	**A:D:E**	-0.3542	**A:B:C:D**	-0.0625
B	-0.1875	**A:C**	0.1458	**B:E**	0.3542	**A:B:D**	-0.2500	**B:C:D**	0.1875	**A:B:C:E**	-0.0417
C	2.8542	**A:D**	0.4792	**C:D**	-0.3750	**A:B:E**	0.3542	**B:C:E**	-0.1667	**A:C:D:E**	0.3333
D	-0.1042	**A:E**	0.1667	**C:E**	0.1042	**A:C:D**	0.1667	**B:D:E**	0.4583	**B:C:D:E**	-0.2292
E	0.4583	**B:C**	-0.0417	**D:E**	-0.5625	**A:C:E**	0.1875	**C:D:E**	0.2500	**A:B:C:D:E**	0.2292

Table 7: Final Estimates of effects after the combining the replicates from each run for analysis with average score as response variable. Note that the estimate of effect for factor ABCDE is from the second replicate while that for factor ABDE is from first replicate

Factors	**Estimate**	**Factors**	**Estimate**	**Factors**	**Estimate**	**Factors**	**Estimate**	**Factors**	**Estimate**
A	0.2604	**A:D**	0.3646	**D:E**	-0.3333	**B:C:D**	-0.1042	**A:C:D:E**	0.1458
B	0.1042	**A:E**	0.1667	**A:B:C**	-0.0417	**B:C:E**	-0.1563	**B:C:D:E**	-0.1979
C	2.4271	**B:C**	0.1458	**A:B:D**	0.0417	**B:D:E**	0.4896	**A:B:C:D:E**	0.2292
D	-0.1979	**B:D**	-0.1042	**A:B:E**	0.4271	**C:D:E**	-0.0417		
E	0.6042	**B:E**	0.1979	**A:C:D**	-0.1146	**A:B:C:D**	0.0000		
A:B	-0.0833	**C:D**	0.0729	**A:C:E**	0.0417	**A:B:C:E**	0.1563		
A:C	0.4896	**C:E**	-0.1458	**A:D:E**	0.0208	**A:B:D:E**	-0.6875		

in appendix A.2.2.2. However, we do not include the block variable in the reduced model for prediction. The diagnostic plots for the reduced model with response variable as average scores is shown in Figure 3. Diagnositc plots for reduced models having scores of judge AD, AG, KJ are shown in Appendix A.3.2.3, A.4.2.3, A.5.2.3. For difference score model we do not have a reduced model as there are no significant effects. The diagnostic plots shows that the assumption for linear regression models are met approximately for all models apart from the model having scores from judge AD as response variable. For this model the residual versus fitted values have a higher spread over lower levels of C while they have lower spread over higher levels of C, also the trend in Q-Q plot deviates from straight line on both the edges. The plot for Cooks distance reveals that there is no outlier in the data.

4.3.2 Performing Transformation

In order to check if the assumption of the linear additive models can be improved, I performed a Box-Cox transformation with additive blocking to check if after the transformation any new factors turns out to be significant or else the transformation leads to better model assumptions. The optimal value of λ, for the model with average score as response variable, turns out to be 1.298; while for the model with scores from judge AD as response is 1.798; while for model with response from judge AG is 1.645; and for model with response from judge KJ is 0.99. Thus we do not transform the model with response from judge KJ as well as the model with difference score as response (as it have negative values and no significant effect without transformation). The code and analysis is provided in appendix sections A.2.2.4, A.3.2.4, A.4.2.4, A.5.2.4. After transformation, the significance of variables are again analyzed for each model using the adjusted p-values and also the simultaneous confidence interval formed by Studentized Maximum Modulus at a confidence level of 90%. The analysis is shown in the appendix section A.2.2.4, A.3.2.4 and A.4.2.4 for models with average scores, scores by Judge AD and scores by judge AG as response respectively. For model with average scores as response variable, factor E (simmer time) also turns out to be weakly significant using the SMM confidence interval. Also, the block factor which was significant previously now become insignificant for the model with average scores as response. For the SMM CI, the value of $g = 5$ and the degree of freedom $\nu = 64 - 5 - 3 - 1 = 55$. We thus interpolate between the values (2.183,2.470,2.160,2.439) to get the multiplier $M^{0.1}_{5,55}$. For model with scores from judge AG as response variable, we find that the significance of factor E is much stronger along with significance of factor C. The confidence intervals for transformed models are shown in Table 9 for models with average score as response as well as for model with scores from judge AG (excluding the block variables). For model with scores from judge AD as response variable, there are no new significant factors but the diagnostic plot suggests an improvement in model fit as can be seen in plots provided in appendix section

h[]

Table 8: Studentized maximum modulus Confidence Interval and Simultaneous Bonferroni Confidence Intervals for the final estimates using both replicates for model with average score as response

Factors	SMM CI	BONF. CI	Factors	SMM CI	BONF. CI	Factors	SMM CI	BONF. CI
A	{-0.4953, 1.0162}	{-0.5107, 1.0315}	**A:E**	{-0.5891, 0.9224}	{-0.6045, 0.9378}	**B:C:E**	{-0.912, 0.5995}	{-0.9274, 0.6149}
B	{-0.6516, 0.8599}	{-0.667, 0.8753}	**B:E**	{-0.5578, 0.9537}	{-0.5732, 0.969}	**A:D:E**	{-0.7349, 0.7766}	{-0.7503, 0.792}
C	**{1.6713, 3.1828}**	**{1.656, 3.1982}**	**C:E**	{-0.9016, 0.6099}	{-0.917, 0.6253}	**B:D:E**	{-0.2662, 1.2453}	{-0.2815, 1.2607}
D	{-0.9537, 0.5578}	{-0.969, 0.5732}	**D:E**	{-1.0891, 0.4224}	{-1.1045, 0.4378}	**C:D:E**	{-0.7974, 0.7141}	{-0.8128, 0.7295}
E	{-0.1516, 1.3599}	{-0.167, 1.3753}	**A:B:C**	{-0.7974, 0.7141}	{-0.8128, 0.7295}	**A:B:C:D**	{-0.7557, 0.7557}	{-0.7711, 0.7711}
A:B	{-0.8391, 0.6724}	{-0.8545, 0.6878}	**A:B:D**	{-0.7141, 0.7974}	{-0.7295, 0.8128}	**A:B:C:E**	{-0.5995, 0.912}	{-0.6149, 0.9274}
A:C	{-0.2662, 1.2453}	{-0.2815, 1.2607}	**A:C:D**	{-0.8703, 0.6412}	{-0.8857, 0.6565}	**A:B:D:E**	{-1.4432, 0.0682}	{-1.4586, 0.0836}
B:C	{-0.6099, 0.9016}	{-0.6253, 0.917}	**B:C:D**	{-0.8599, 0.6516}	{-0.8753, 0.667}	**A:C:D:E**	{-0.6099, 0.9016}	{-0.6253, 0.917}
A:D	{-0.3912, 1.1203}	{-0.4065, 1.1357}	**A:B:E**	{-0.3287, 1.1828}	{-0.344, 1.1982}	**B:C:D:E**	{-0.9537, 0.5578}	{-0.969, 0.5732}
B:D	{-0.8599, 0.6516}	{-0.8753, 0.667}	**A:C:E**	{-0.7141, 0.7974}	{-0.7295, 0.8128}	**A:B:C:D:E**	{-0.5266, 0.9849}	{-0.542, 1.0003}
C:D	{-0.6828, 0.8287}	{-0.6982, 0.844}						

A.3.2.5. For models having scores from each judge (AD, AG) as response variable we find that block variable shows some significance. The final diagnostic plots for transformed and reduced model having average score as response variable is shown in Figure 4. The diagnostic plots adhere to the assumptions of the linear regression models and also show no presence of outliers.

Table 9: Estimates and Studentized Maximum modulus Confidence Intervals after transformation for models with average score as response variable and score from judge AG as response variable

Factors	Model (Avg Response)		Model (Score Judge AG)	
	Estimate	**SMM CI**	**Estimate**	**SMM CI**
A	0.619859	{-0.5757, 1.8154}	-0.16519	{-3.448, 3.1176}
B	0.209493	{-0.986, 1.405}	-0.55742	{-3.8402, 2.7254}
C	5.235061	**{4.0395, 6.4306}**	12.44405	**{9.1612, 15.7269}**
D	-0.43401	{-1.6295, 0.7615}	0.442395	{-2.8404, 3.7252}
E	1.258884	**{0.0633, 2.4544}**	3.906783	**{0.624, 7.1896}**

4.3.3 Trend Analysis

Finally to conclude we also perform a trend analysis. There are chances that the tasting experiment might have some temporal trends in them. This may arise because of short term taste memory of the judges. To analyze the same we plot the residuals obtained from the final reduced model with the order in which each runs were conducted on different days as shown in Figure 5 for model with scores from judge AG as response variable (plots for other judges are shown in Appendix section A.3.2.5 and A.4.2.5 for models with response variables from judge AD and KJ respectively). We also check the auto-correlation functions plot to see the presence of short term trends in scoring provided by judges on each day. However the plots reveal that there are no significant temporal trends in the scores provided by the judges.

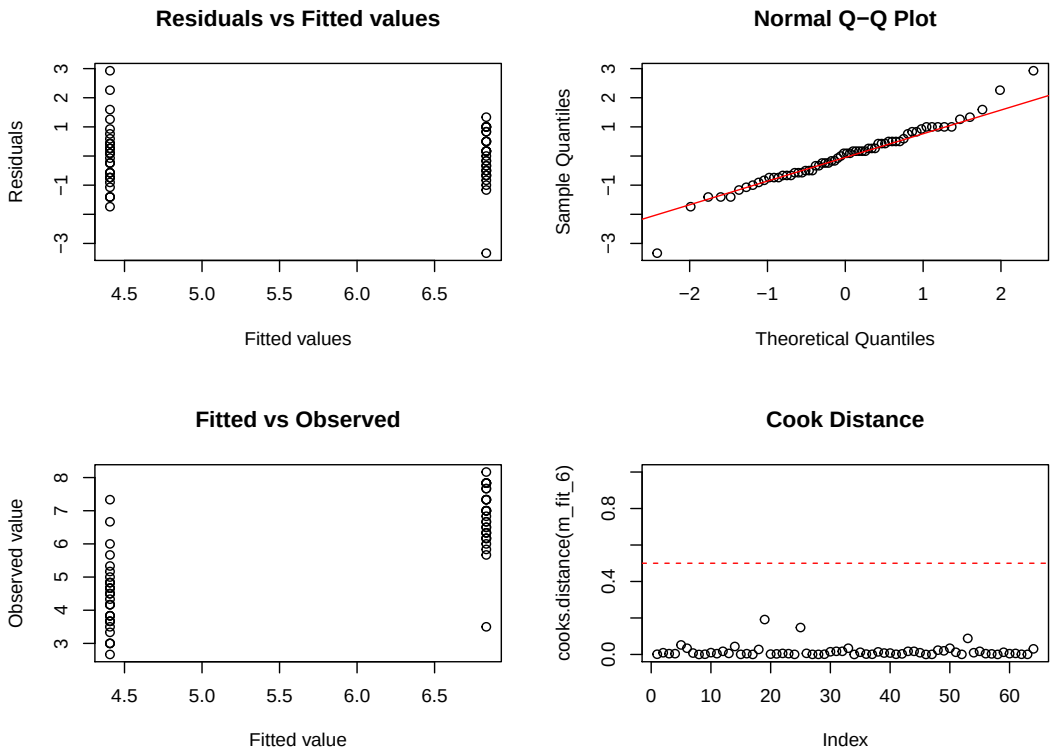


Figure 3: Diagnostic plots for reduced model with average score as response variable

4.3.4 Final Model

Finally we present the significant factors for each of our models in Table 10. It should be noted that, I included block variables while transforming the data to see if blocking turns out to be significant for the transformed models or not. For the average model we do not observe significance of blocking, however for the models having scores from each individual judges as response we find that some levels of blocking variable are significant even after transformation. For model with judge KJ as response variable, we find that block is significant along with factor C suggesting an improvement in scoring guidelines for the experiment. We report the final models without including the block variables in equations 2, 3, 4, 5, 6 for prediction. The final model for each type of response variables are:

Table 10: Significant factors for each model after analysis with Replicated data

Model #	Response Variable	Significant factors without transformation	Significant factors after transformation
1	**Average score**	C (sugar)	C (sugar), E (simmer time)
2	**Judge AD score**	C (sugar)	C (sugar)
3	**Judge AG score**	C (sugar)	C (sugar), E (simmer time)
4	**Judge KJ score**	C (sugar)	Not performed
5	**Difference score (AG-AD)**	–	Not performed

- For model with average score as response variable:

$$y_{average} = (19.09 + 5.235X_3 + 1.259X_5)^{\frac{1}{1.298}} \tag{2}$$

- For model having scores from judge AD as response variable:

$$y_{J_{AD}} = (58.414 + 15.211X_3)^{\frac{1}{1.798}} \tag{3}$$

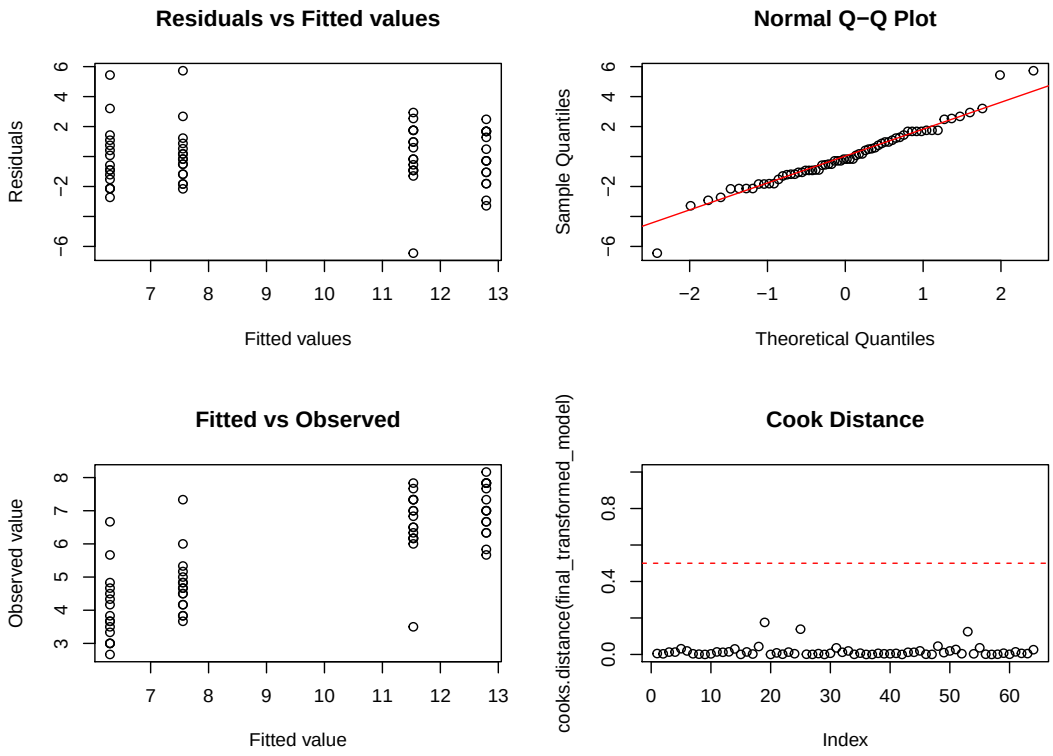


Figure 4: Diagnostic plots for reduced model with average score as response variable

- For model having scores from judge AG as response variable:

$$y_{J_{AG}} = (33.5 + 12.44X_3 + 3.9X_5)^{\frac{1}{1.645}} \tag{4}$$

- For model having scores from judge KJ as response variable:

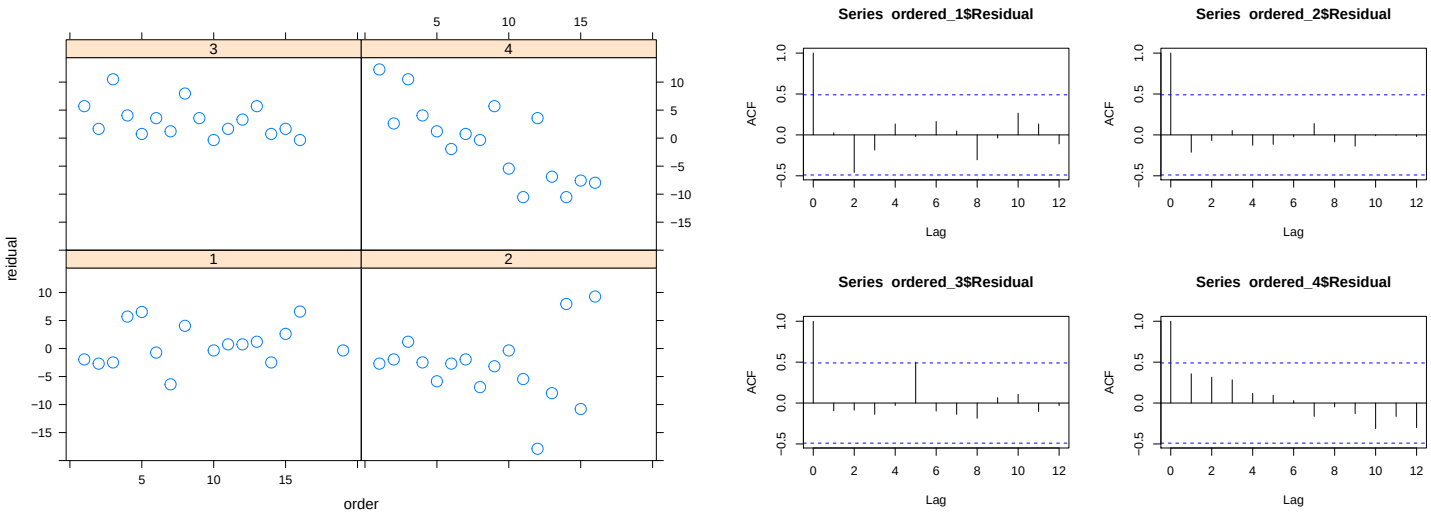
$$y_{J_{KJ}} = 10.36 + 2.58X_3 \tag{5}$$

- For model having difference scores (scores from judge AG- scores from judge AK) as response variable:

$$y_{diff} = -0.3125 \tag{6}$$

5 Conclusion

After analyzing all the models it can be concluded that factor C which is higher levels of sugar surely affects the taste of chai for all judges as well as on average. For scores of judge AG and also for average scores, simmer time (factor E) is also significance for the taste of chai. For the experiment conducted using difference in scores from judge AG and AD we found out that none of the variables are significant, suggesting that the main factors which turn up to be significant for individual judges are indeed specific to the judges. Other factors do not seem to be significant in the experiment. This is because of the low variance observed in scores from the judges. One possible reason for the same could be that, for people with sweet tooth all other factors seems to underplay when compared with sugar. This is also observed, when we do not have salt, or have low salt in food, the taste of all other spices seems to be bland if adequate salt is not present. Also, intuitively simmering chai for a longer time should be important as one can get the extracts from all ingredients properly dissolved with larger simmer time. The experiment conducted had limited stratification in terms of demographics of the judges like their age, medical history etc.,. Significance of other factors would also change when such stratification in data is large, which would



(a) Residual versus Run order for each day

(b) Autocorrelations between runs of each day

Figure 5: Trend analysis for reduced and transformed model having scores of judge a AG as response

help to make a better global comment.

6 Lessons learnt

I have three important takeaways from the experiment. The first most important lesson I learned was that, analyzing the data to get early estimates of the main effects is important. As seen in the report, analyzing only the data obtained from the first day, using 2^{5-1} fractional factorial design, gave us a good indication about the significant main effects, which persisted even as we gathered more data. This puts an emphasis on the advantage of performing a complete block run as a preliminary analysis to see if there are changes necessary for the design of experiment. The second very important takeaway that I have from the experiment is that, while performing a tasting experiment blocks of smaller sizes (≤ 8) should be preferred. This is because the tasting memory of the experimenter fades up as they taste increasing number of samples, this cause them to provide scores with low variability. The third major takeaway is that, one can prefer to use ranking of samples as response variable rather than discrete point scores. There would be some issues while using ranks variable for time blocked experiments, as the memory of the scores would get faded with each day. However, even than it will help us to generate larger variability in data as opposed to the discrete scores.

References

Loh, Wei-Yin. 1992. Identification of active contrasts in unreplicated factorial experiments. *Computational statistics & data analysis*, **14**(2), 135–148.

Stoline, Michael R, & Ury, Hans K. 1979. Tables of the studentized maximum modulus distribution and an application to multiple comparisons among means. *Technometrics*, **21**(1), 87–93.

ANALYSIS_FRACTIONAL_FACTORIAL

Abhijeet Bhardwaj

4/28/2021

APPENDIX CODE

Section A.1. Analysis of day 1 run using average score as 2^{5-1} design

```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3

subset_ = files_1[files_1$Block_ABCDE == -1, ]
A = subset_$A
B = subset_$B
C = subset_$C
D = subset_$D
E = subset_$E
block = subset_$Block_ABCDE
avg_score = subset_$J_AVG
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^2,data=subset_dat_)
2*summary(m_fit_1)$coefficients
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.10416667        NaN     NaN      NaN
## A            0.39583333        NaN     NaN      NaN
## B            0.43750000        NaN     NaN      NaN
## C            2.68750000        NaN     NaN      NaN
## D           -0.64583333        NaN     NaN      NaN
## E            0.68750000        NaN     NaN      NaN
## A:B          0.22916667        NaN     NaN      NaN
## A:C          0.31250000        NaN     NaN      NaN
## A:D          0.39583333        NaN     NaN      NaN
## A:E          0.56250000        NaN     NaN      NaN
## B:C         -0.06250000        NaN     NaN      NaN
## B:D          0.18750000        NaN     NaN      NaN
## B:E          0.43750000        NaN     NaN      NaN
## C:D          0.02083333        NaN     NaN      NaN
## C:E         -0.72916667        NaN     NaN      NaN
## D:E          0.43750000        NaN     NaN      NaN
```

```r
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

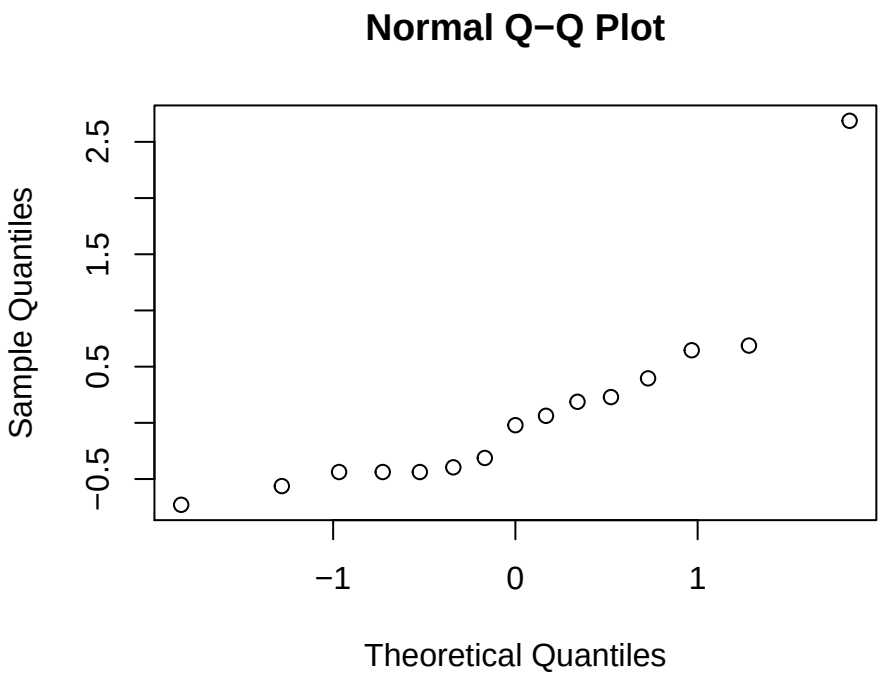
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^2,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 6
```

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

```r
A_final = perm[1,6] * A
B_final = perm[2,6] * B
C_final = perm[3,6] * C
D_final = perm[4,6] * D
E_final = perm[5,6] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^2,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -0.3406948 -0.9674216  1.8339146  0.9674216  1.2815516  0.5244005
##  [7] -0.1678940  0.7279133 -1.2815516  0.1678940  0.3406948 -0.7279133
## [13]  0.0000000 -1.8339146 -0.5244005
##
## $y
##         A_final         B_final         C_final         D_final         E_final
##     -0.39583333     -0.43750000      2.68750000      0.64583333      0.68750000
## A_final:B_final A_final:C_final A_final:D_final A_final:E_final B_final:C_final
##      0.22916667     -0.31250000      0.39583333     -0.56250000      0.06250000
## B_final:D_final B_final:E_final C_final:D_final C_final:E_final D_final:E_final
##      0.18750000     -0.43750000     -0.02083333     -0.72916667     -0.43750000
```

```
# Step2: Fit a lm ------------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier ------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # outliers::: SIGNIFICANT c_final
```

```
## C_final
##       3
```

```
x_inlier = qq_x[which(abs(qq_y) <= CC)]
y_inlier = qq_y[which(abs(qq_y) <= CC)]
```

```r
# Step4: Fit another lm ---------------------------------------------------
fit2 = lm(y_inlier ~ x_inlier)
beta2 = fit2$coefficients[2]
beta1 / beta2 # 1.186
```

```
##     qq_x
## 1.479717
```

```r
# From the table in the paper,
# n = 31, alpha = 0.05 -> threshold = 1.108
# So we have the significance here

m = length(x_inlier)
y_mean = mean(y_inlier)
x_mean = mean(x_inlier)
n_prime = round(31/4,1)
sigma_user = sqrt(sum((fit2$residuals)^2)/fit2$df.residual)

LHS = abs(y_inlier - y_mean + beta2 * (x_inlier - x_mean))
RHS = n_prime * sqrt(qf(p = 0.975,df1 = n_prime,df2 = m-2)) * sigma_user *
  sqrt(1 + 1/m + (x_inlier - x_mean)^2 / (var(x_inlier) * (m+1) ))
final_idx = which(LHS <= RHS)
D_ = max(abs(y_inlier[final_idx]))
c(D_, CC)
```

```
##                 75%
## 0.7291667 1.5000000
```

```r
final_threshold = max(CC,D_)
print(paste0("after daniels method :", names(which(abs(qq_y) > final_threshold)))) # again only C_final
```

```
## [1] "after daniels method :C_final"
```

ANALYSIS_AVERAGE

Abhijeet Bhardwaj

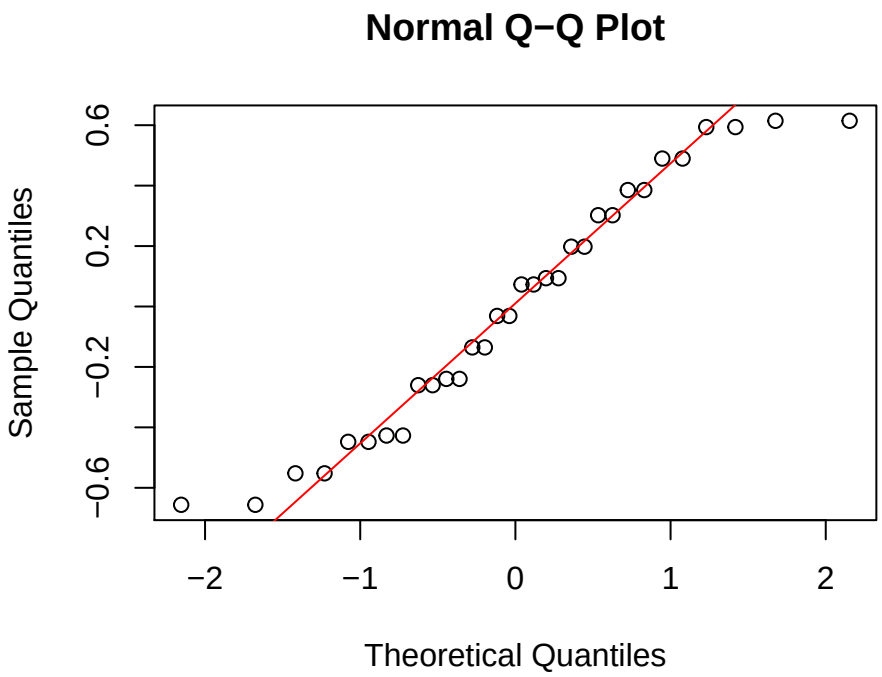
4/27/2021

SECTION A.2: Analysis for scores by averaging

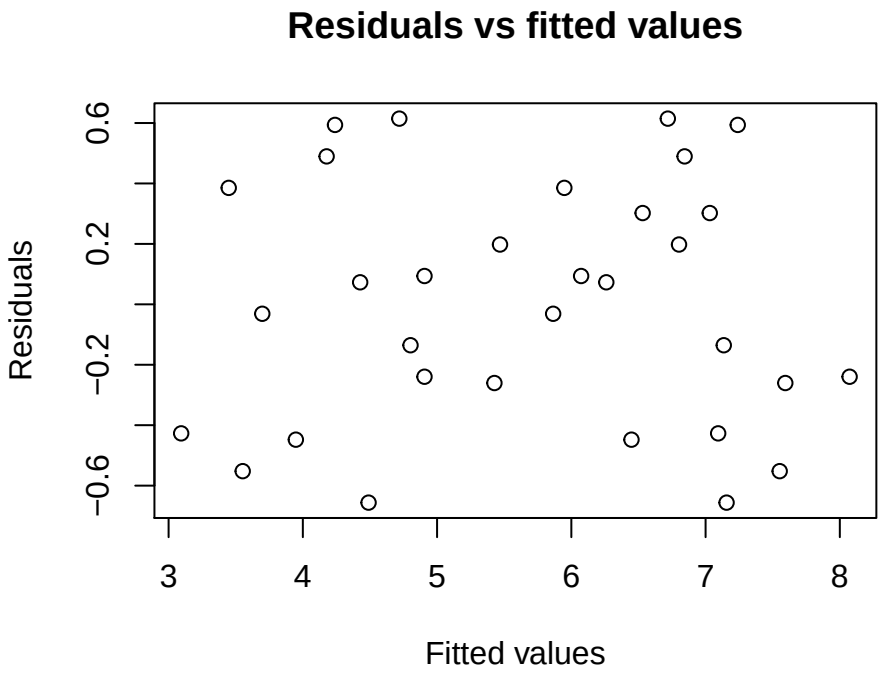
SECTION A.2.1 :Analysis of first run

#SECTION A.2.1.1 DATA VISUALIZATION

```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3
A = files_1$A
B = files_1$B
C = files_1$C
D = files_1$D
E = files_1$E
block = files_1$Block_ABCDE
avg_score = files_1$J_AVG
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^3+as.factor(block),data=subset_dat_)
resid = m_fit_1$resid
fitted = m_fit_1$fit
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs fitted values")
```



SECTION A.2.1.2 ANALYZING SIGNIFICANCE USING CONFIDENCE INTERVALS

```
# ASSUMING 4 FI's TO BE ZERO but not ABCDE as it is a block effect for sigma
#fitting complete model to get estimates of 4fis
m_fit_2 = lm(y~.^5,data=subset_dat_)
summary_fullmodel_ = summary(m_fit_2)
2*summary_fullmodel_$coefficients
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.35416667        NaN     NaN      NaN
## A            0.22916667        NaN     NaN      NaN
## B            0.39583333        NaN     NaN      NaN
## C            2.00000000        NaN     NaN      NaN
## D           -0.29166667        NaN     NaN      NaN
## E            0.75000000        NaN     NaN      NaN
## A:B         -0.10416667        NaN     NaN      NaN
## A:C          0.83333333        NaN     NaN      NaN
## A:D          0.25000000        NaN     NaN      NaN
## A:E          0.16666667        NaN     NaN      NaN
## B:C          0.33333333        NaN     NaN      NaN
## B:D          0.08333333        NaN     NaN      NaN
## B:E          0.04166667        NaN     NaN      NaN
## C:D          0.52083333        NaN     NaN      NaN
## C:E         -0.39583333        NaN     NaN      NaN
## D:E         -0.10416667        NaN     NaN      NaN
## A:B:C       -0.54166667        NaN     NaN      NaN
## A:B:D        0.33333333        NaN     NaN      NaN
## A:B:E        0.50000000        NaN     NaN      NaN
## A:C:D       -0.39583333        NaN     NaN      NaN
## A:C:E       -0.10416667        NaN     NaN      NaN
## A:D:E        0.39583333        NaN     NaN      NaN
## B:C:D       -0.39583333        NaN     NaN      NaN
## B:C:E       -0.14583333        NaN     NaN      NaN
## B:D:E        0.52083333        NaN     NaN      NaN
## C:D:E       -0.33333333        NaN     NaN      NaN
## A:B:C:D      0.06250000        NaN     NaN      NaN
## A:B:C:E      0.35416667        NaN     NaN      NaN
## A:B:D:E     -0.68750000        NaN     NaN      NaN
## A:C:D:E     -0.04166667        NaN     NaN      NaN
## B:C:D:E     -0.16666667        NaN     NaN      NaN
## A:B:C:D:E    0.25000000        NaN     NaN      NaN
```

```
coeffs_4fis = 2*summary_fullmodel_$coefficients[27:31,1]
sd_4 = sqrt(sum(coeffs_4fis^2)/5)
threshold_4 = sd_4*qt(0.1/(2*25),5,lower.tail = FALSE)
all_coeffs_ = 2*summary_fullmodel_$coefficients[,1]
eff = as.numeric(all_coeffs_)
which(abs(eff)>threshold_4) # intercept and C turns out significant
```

```
## [1] 1 4
```

```
# ASSUMING 3 and 4 FI's TO BE ZERO but not ABCDE as it is a block effect for estimating sigma
#fitting complete model to get estimates of 4fis
coeffs_34fis = 2*summary_fullmodel_$coefficients[17:31,1]
```

```r
var_34 = sum(coeffs_34fis^2)/15
sd_34 = sqrt(var_34)
threshold_34 = sd_4*qt(0.1/(2*15),15,lower.tail = FALSE)
threshold_34
```

```
## [1] 1.1179
```

```r
which(abs(eff)>threshold_34) # intercept and C turns out significant
```

```
## [1] 1 4
```

SECTION A.2.1.3 ANALYZING SIGNIFICANCE USING DANIEL METHOD

```r
## APPLY DANIEL METHOD
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

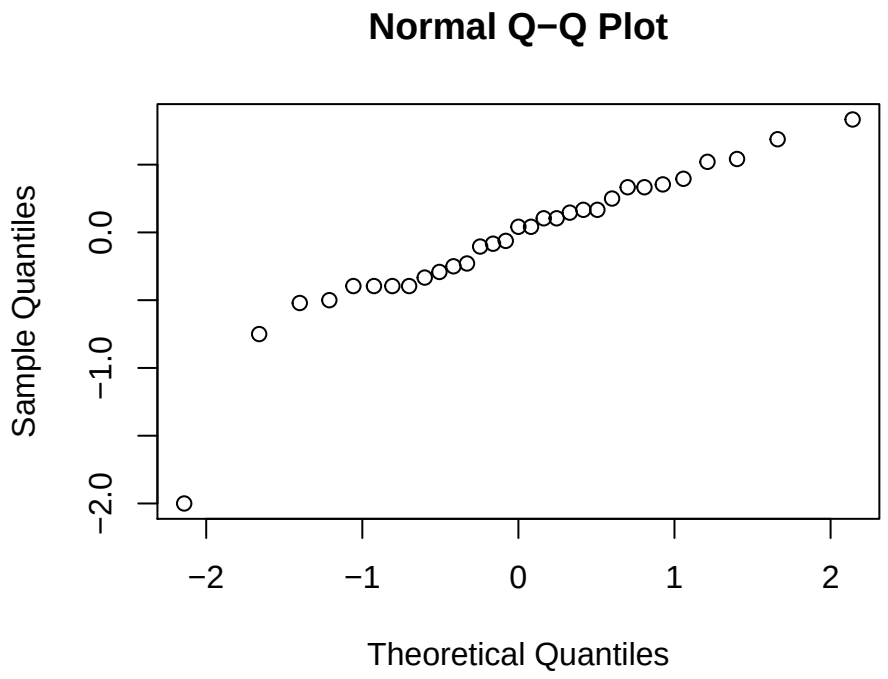
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^5,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 3
```

```
## [1] 3
```

```r
A_final = perm[1,3] * A
B_final = perm[2,3] * B
C_final = perm[3,3] * C
D_final = perm[4,3] * D
E_final = perm[5,3] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^5,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -0.32929135 -1.05741423 -2.14119812 -0.50593365 -1.66069761 -0.24500622
##  [7]  2.14119812 -0.41598722  0.41598722  0.70009021 -0.16242937  0.00000000
## [13] -1.40074506 -0.92524456  0.24500622  1.40074506  0.80754104 -1.21123213
## [19] -0.80754104  0.16242937  1.05741423 -0.70009021  0.32929135  1.21123213
## [25] -0.60017878 -0.08094729  0.92524456  1.66069761  0.08094729  0.50593365
## [31]  0.60017878
##
## $y
##                                A_final                                B_final
##                             -0.22916667                            -0.39583333
##                                C_final                                D_final
##                             -2.00000000                            -0.29166667
##                                E_final                       A_final:B_final
##                             -0.75000000                            -0.10416667
##                       A_final:C_final                       A_final:D_final
##                              0.83333333                            -0.25000000
##                       A_final:E_final                       B_final:C_final
##                              0.16666667                             0.33333333
##                       B_final:D_final                       B_final:E_final
```

```
##                            -0.08333333                            0.04166667
##                         C_final:D_final                         C_final:E_final
##                            -0.52083333                            -0.39583333
##                         D_final:E_final                  A_final:B_final:C_final
##                             0.10416667                             0.54166667
##                  A_final:B_final:D_final                  A_final:B_final:E_final
##                             0.33333333                            -0.50000000
##                  A_final:C_final:D_final                  A_final:C_final:E_final
##                            -0.39583333                             0.10416667
##                  A_final:D_final:E_final                  B_final:C_final:D_final
##                             0.39583333                            -0.39583333
##                  B_final:C_final:E_final                  B_final:D_final:E_final
##                             0.14583333                             0.52083333
##                  C_final:D_final:E_final         A_final:B_final:C_final:D_final
##                            -0.33333333                            -0.06250000
##          A_final:B_final:C_final:E_final         A_final:B_final:D_final:E_final
##                             0.35416667                             0.68750000
##          A_final:C_final:D_final:E_final         B_final:C_final:D_final:E_final
##                             0.04166667                             0.16666667
## A_final:B_final:C_final:D_final:E_final
##                             0.25000000
```

```
# Step2: Fit a lm ------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier ------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # outliers::: SIGNIFICANT c_final
```

```
## C_final
##       3
```

```
x_inlier = qq_x[which(abs(qq_y) <= CC)]
y_inlier = qq_y[which(abs(qq_y) <= CC)]
# Step4: Fit another lm ----------------------------------------------------
fit2 = lm(y_inlier ~ x_inlier)
beta2 = fit2$coefficients[2]
beta1 / beta2 # 1.186
```

```
##     qq_x
## 1.186383
```

```
# From the table in the paper,
# n = 31, alpha = 0.05 -> threshold = 1.108
# So we have the significance here

m = length(x_inlier)
y_mean = mean(y_inlier)
x_mean = mean(x_inlier)
n_prime = round(31/4,1)
sigma_user = sqrt(sum((fit2$residuals)^2)/fit2$df.residual)
```

```
LHS = abs(y_inlier - y_mean + beta2 * (x_inlier - x_mean))
RHS = n_prime * sqrt(qf(p = 0.975,df1 = n_prime,df2 = m-2)) * sigma_user *
  sqrt(1 + 1/m + (x_inlier - x_mean)^2 / (var(x_inlier) * (m+1) ))
final_idx = which(LHS <= RHS)
D_ = max(abs(y_inlier[final_idx]))
c(D_, CC)
```

```
##                 75%
## 0.3333333 1.3125000
```

```
final_threshold = max(CC,D_)
print(paste0("after daniels method :", names(which(abs(qq_y) > final_threshold)))) # again only C_final
```

```
## [1] "after daniels method :C_final"
```

```
################################################################################
```

SECTION A.2.1.4 ANALYZING SIGNIFICANCE USING DONG's METHOD

```
#ANALYSIS BY DONGS METHOD
coeff=coef(m_fit_2)*2
theta1=abs(na.omit(coeff[-c(1,31)]))
s0=1.5*median(theta1)
m1=sum(theta1<=2.5*s0)
s1=sqrt(sum(theta1[theta1<=2.5*s0]^2)/m1)
m2=sum(theta1<=2.5*s1)
s2=sqrt(sum(theta1[theta1<=2.5*s1]^2)/m2)
g=length(theta1)
gamma=0.5*(1-(1-0.05)^(1/g))
print(paste0("after DONG's method significant effect :", names(theta1[theta1>s2*qt(gamma,m2,lower.tail =
```

```
## [1] "after DONG's method significant effect :C"
```

```
################################################
```

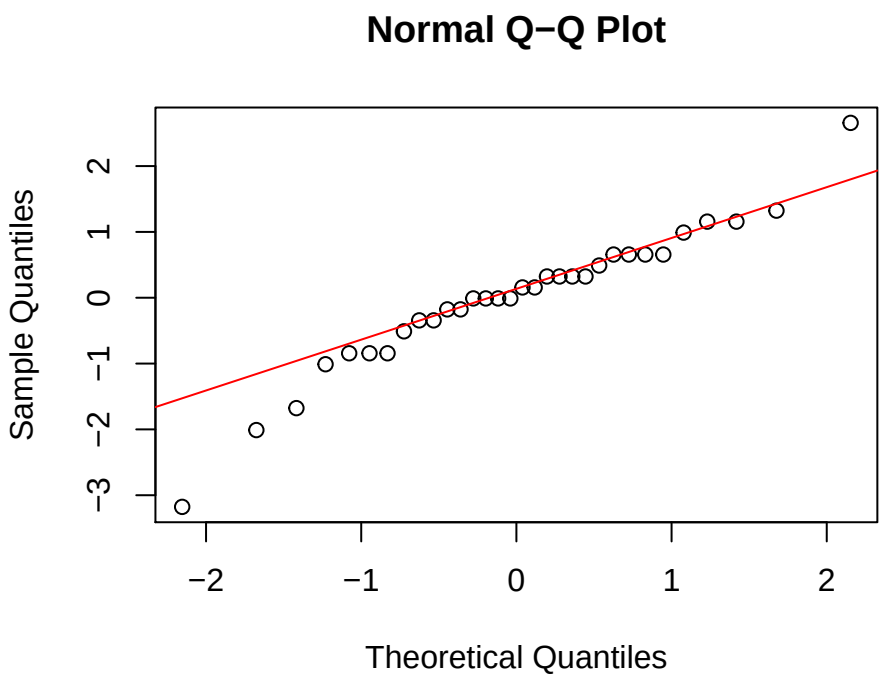
SECTION A.2.1.5 MODEL FITTING WITH SIGNIFICANT FACTOR FOR UN REPLICATED EXPERIMENT

```
# Model with c only analysis
m_fit_3 = lm(y~C,data=subset_dat_)
summary(m_fit_3)
```

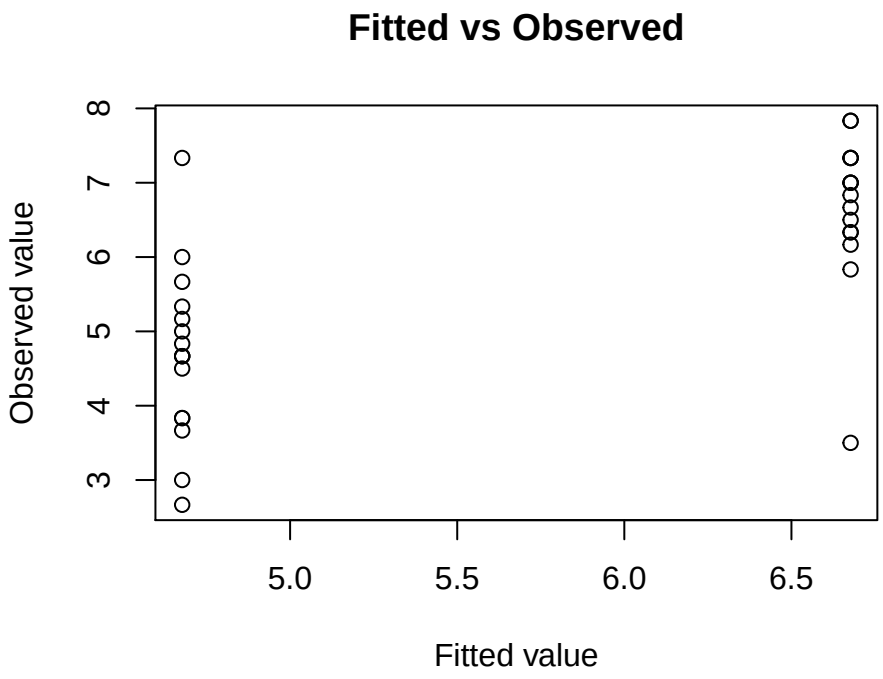
```
##
## Call:
## lm(formula = y ~ C, data = subset_dat_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1771 -0.3854  0.0729  0.6562  2.6563
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   5.6771     0.1924  29.500  < 2e-16 ***
## C             1.0000     0.1924   5.196 1.34e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.089 on 30 degrees of freedom
## Multiple R-squared:  0.4737, Adjusted R-squared:  0.4562
## F-statistic:    27 on 1 and 30 DF,  p-value: 1.338e-05
```

```
resid = m_fit_3$resid
fitted = m_fit_3$fit

qqnorm(resid)
qqline(resid,col="red")
```



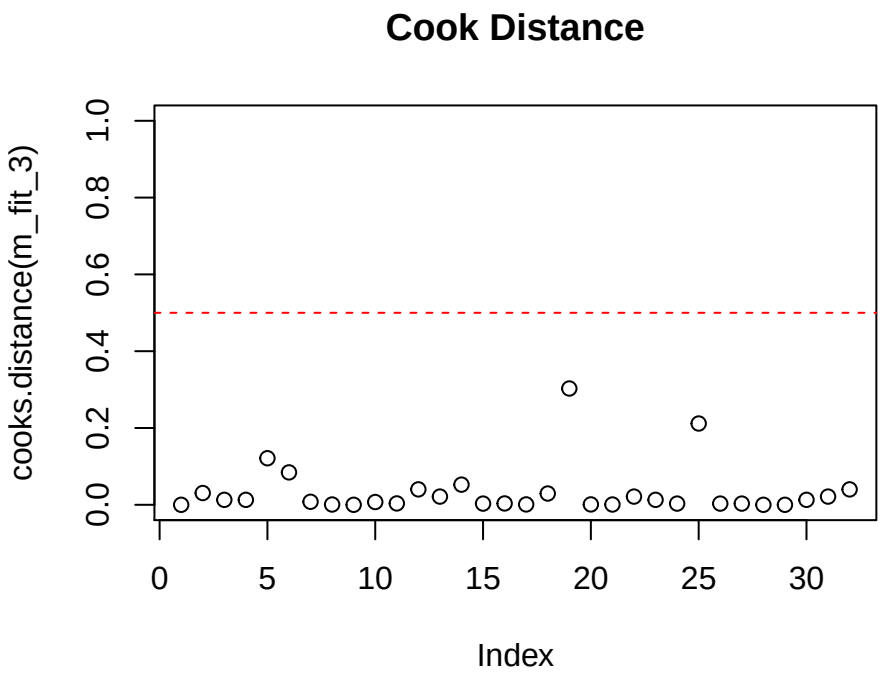

```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
bartlett.test(y~C) # vIOLATED CONSTANT VARIANCE ASSUMPTION
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  y by C
## Bartlett's K-squared = 0.2117, df = 1, p-value = 0.6454
```

```
# checking for outliers in the final model
plot(cooks.distance(m_fit_3), ylim =c(0,1) ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.2.1.6 APPLYING TRANSFORMATION TO SEE IF MORE SIGNIFICANT FACTORS CAN BE EXTRACTED

```
######################
y0=subset_dat_$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+block)
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda
```

```
## [1] 1.658418
```

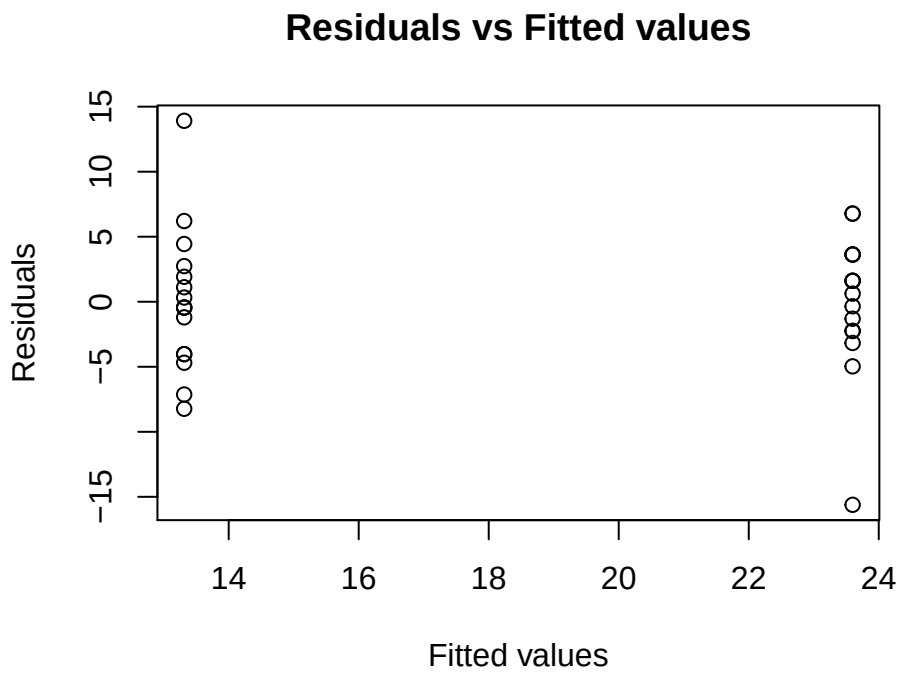
```
y_t=y0^lambda
#data_subs_bc_t=data.frame(y_t= y_t,A=A,B=B,C=C,D=D,E=E)
#model_trans=lm(y_t~ .^2,data_subs_bc_t)
model_trans=lm(y_t~ A+B+C+D+E+block)
summary(model_trans)
```

```
##
## Call:
## lm(formula = y_t ~ A + B + C + D + E + block)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3212  -3.2011  -0.3122   3.2880  12.7138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.4567     0.9353  19.734  < 2e-16 ***
## A             0.7266     0.9353   0.777   0.4445
## B             1.0052     0.9353   1.075   0.2928
## C             5.1413     0.9353   5.497 1.04e-05 ***
## D            -0.6707     0.9353  -0.717   0.4799
## E             1.7456     0.9353   1.866   0.0738 .
## block         0.5151     0.9353   0.551   0.5867
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.291 on 25 degrees of freedom
## Multiple R-squared:  0.592,  Adjusted R-squared:  0.4941
## F-statistic: 6.046 on 6 and 25 DF,  p-value: 0.0005158
```

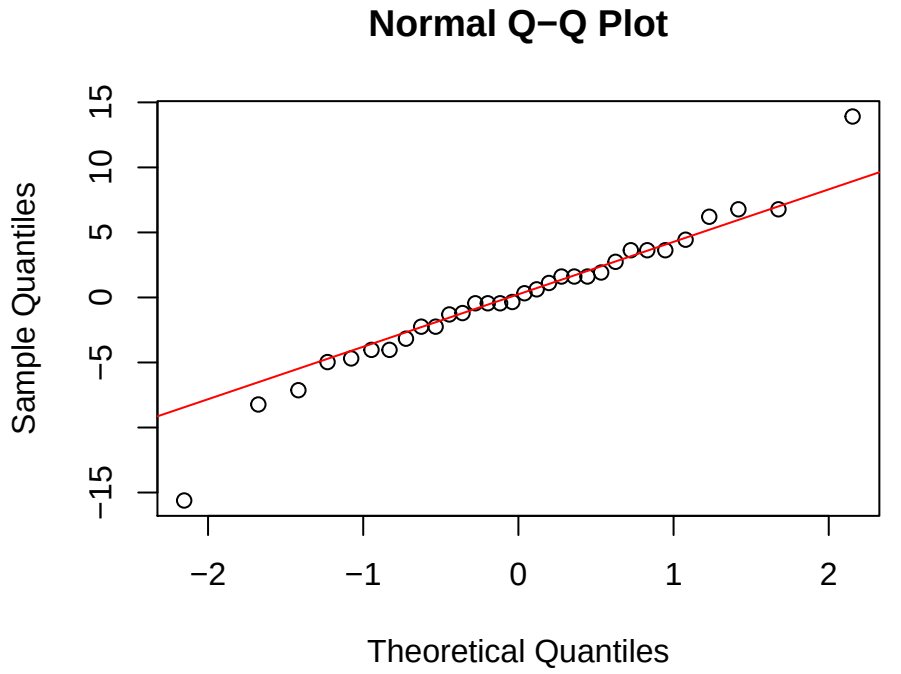
```
print(round(p.adjust(summary(model_trans)$coefficients[,4]),4))
```

```
## (Intercept)           A           B           C           D           E
##      0.0000      1.0000      1.0000      0.0001      1.0000      0.3688
##       block
##      1.0000
```

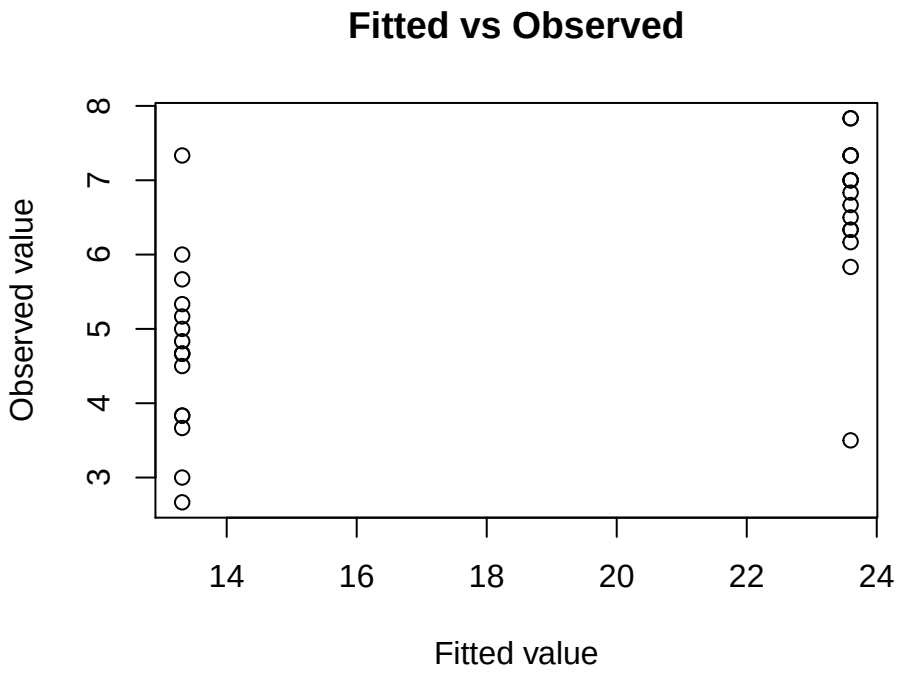
```
# ONLY C SIGNIFICANT AFTER TRANSFORMATION
model_trans=lm(y_t~ C)
resid = model_trans$resid
fitted = model_trans$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



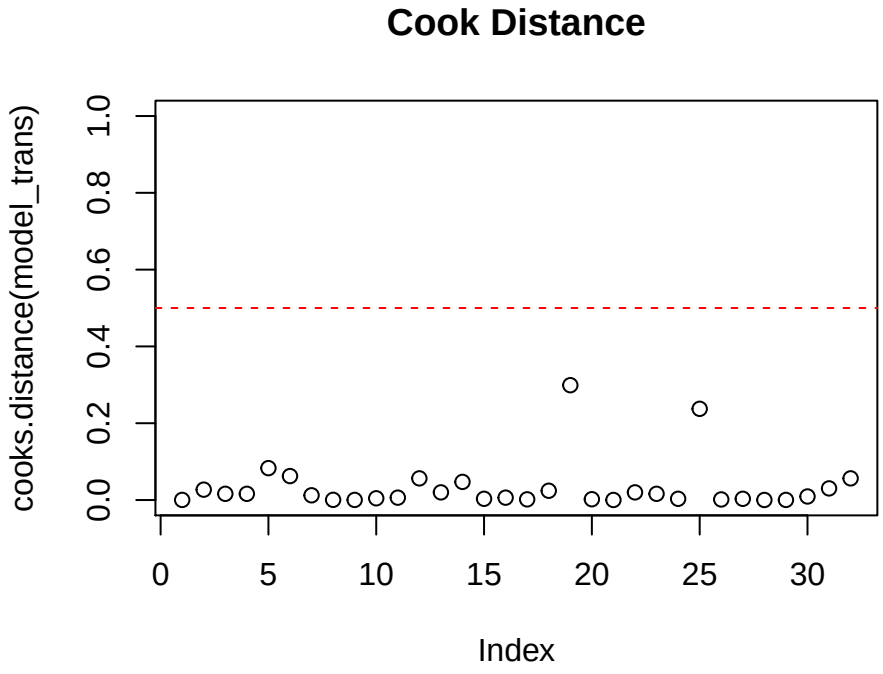

```
qqnorm(resid)
qqline(resid,col="red")
```



```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
plot(cooks.distance(model_trans), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.2.2 :Analysis of second run

SECTION A.2.2.1 :Estimates from second run

```
# COMBINING THE SECOND COMPLETE REPLICATE

files_2 = read.csv('Run_2_with_scores.csv')
files_2$J_AVG = (files_2$J_AD+files_2$J_AG+files_2$J_KJ)/3
A2 = files_2$A
B2 = files_2$B
C2 = files_2$C
D2 = files_2$D
E2 = files_2$E
avg_score2 = files_2$J_AVG
subset_dat_2 = data.frame(y = avg_score2,A=A2,B=B2,C=C2,D=D2,E=E2)
m_fit_4 = lm(y~.^5,data=subset_dat_2)
summary_fullmodel_2 = summary(m_fit_4)
2*summary_fullmodel_2$coefficients
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.12500000        NaN     NaN      NaN
## A            0.29166667        NaN     NaN      NaN
## B           -0.18750000        NaN     NaN      NaN
## C            2.85416667        NaN     NaN      NaN
## D           -0.10416667        NaN     NaN      NaN
## E            0.45833333        NaN     NaN      NaN
## A:B         -0.06250000        NaN     NaN      NaN
## A:C          0.14583333        NaN     NaN      NaN
## A:D          0.47916667        NaN     NaN      NaN
## A:E          0.16666667        NaN     NaN      NaN
## B:C         -0.04166667        NaN     NaN      NaN
## B:D         -0.29166667        NaN     NaN      NaN
## B:E          0.35416667        NaN     NaN      NaN
## C:D         -0.37500000        NaN     NaN      NaN
## C:E          0.10416667        NaN     NaN      NaN
## D:E         -0.56250000        NaN     NaN      NaN
## A:B:C        0.45833333        NaN     NaN      NaN
## A:B:D       -0.25000000        NaN     NaN      NaN
## A:B:E        0.35416667        NaN     NaN      NaN
## A:C:D        0.16666667        NaN     NaN      NaN
## A:C:E        0.18750000        NaN     NaN      NaN
## A:D:E       -0.35416667        NaN     NaN      NaN
## B:C:D        0.18750000        NaN     NaN      NaN
## B:C:E       -0.16666667        NaN     NaN      NaN
## B:D:E        0.45833333        NaN     NaN      NaN
## C:D:E        0.25000000        NaN     NaN      NaN
## A:B:C:D     -0.06250000        NaN     NaN      NaN
## A:B:C:E     -0.04166667        NaN     NaN      NaN
## A:B:D:E      0.45833333        NaN     NaN      NaN
## A:C:D:E      0.33333333        NaN     NaN      NaN
## B:C:D:E     -0.22916667        NaN     NaN      NaN
## A:B:C:D:E    0.22916667        NaN     NaN      NaN
```

SECTION A.2.2.2 :Calculating Confidence Intervals by combining the replicates

```
# CACULATION OF VARIANCE BY FIRST PRINCIPLES
coeffs_1st_rep = 2*summary_fullmodel_$coefficients[-1,1]
coeffs_2st_rep = 2*summary_fullmodel_2$coefficients[-1,1]
avg_coeff_ = (coeffs_1st_rep+coeffs_2st_rep)/2
diff_1 = (coeffs_1st_rep-avg_coeff_)^2
diff_2 = (coeffs_2st_rep-avg_coeff_)^2

diff_1_ = diff_1[-c(31,28)]
diff_2_ = diff_2[-c(31,28)]

var_ = (diff_1_+diff_2_)/(2-1)
var_hat_ = sum(var_)/29 #estimate of 4sigma^2/2^k) \hat{var(Ai)}
multiplier1 = qt(0.1/2/31,df = 64-31-3-1,lower.tail = F)
boundary_pt =   multiplier1*sqrt(var_hat_/2)
print(paste0("standard deviation using first principle: ", sqrt(var_hat_/2)))
```

```
## [1] "standard deviation using first principle: 0.240144894101114"
```

```
avg_coeff_up = avg_coeff_+boundary_pt
avg_coeff_dwn = avg_coeff_-boundary_pt
CI_interval = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn, Up_=

######################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval
```

```
##           variable_      estimate       Lwr_       Up_
## A                 A  2.604167e-01 -0.5107071 1.0315404
## B                 B  1.041667e-01 -0.6669571 0.8752904
## C                 C  2.427083e+00  1.6559596 3.1982071
## D                 D -1.979167e-01 -0.9690404 0.5732071
## E                 E  6.041667e-01 -0.1669571 1.3752904
## A:B             A:B -8.333333e-02 -0.8544571 0.6877904
## A:C             A:C  4.895833e-01 -0.2815404 1.2607071
## A:D             A:D  3.645833e-01 -0.4065404 1.1357071
## A:E             A:E  1.666667e-01 -0.6044571 0.9377904
## B:C             B:C  1.458333e-01 -0.6252904 0.9169571
## B:D             B:D -1.041667e-01 -0.8752904 0.6669571
## B:E             B:E  1.979167e-01 -0.5732071 0.9690404
## C:D             C:D  7.291667e-02 -0.6982071 0.8440404
## C:E             C:E -1.458333e-01 -0.9169571 0.6252904
## D:E             D:E -3.333333e-01 -1.1044571 0.4377904
## A:B:C         A:B:C -4.166667e-02 -0.8127904 0.7294571
## A:B:D         A:B:D  4.166667e-02 -0.7294571 0.8127904
## A:B:E         A:B:E  4.270833e-01 -0.3440404 1.1982071
## A:C:D         A:C:D -1.145833e-01 -0.8857071 0.6565404
## A:C:E         A:C:E  4.166667e-02 -0.7294571 0.8127904
## A:D:E         A:D:E  2.083333e-02 -0.7502904 0.7919571
## B:C:D         B:C:D -1.041667e-01 -0.8752904 0.6669571
## B:C:E         B:C:E -1.562500e-01 -0.9273737 0.6148737
## B:D:E         B:D:E  4.895833e-01 -0.2815404 1.2607071
```

```
## C:D:E         C:D:E -4.166667e-02 -0.8127904 0.7294571
## A:B:C:D     A:B:C:D -2.983724e-16 -0.7711237 0.7711237
## A:B:C:E     A:B:C:E  1.562500e-01 -0.6148737 0.9273737
## A:B:D:E     A:B:D:E -1.145833e-01 -0.8857071 0.6565404
## A:C:D:E     A:C:D:E  1.458333e-01 -0.6252904 0.9169571
## B:C:D:E     B:C:D:E -1.979167e-01 -0.9690404 0.5732071
## A:B:C:D:E A:B:C:D:E  2.395833e-01 -0.5315404 1.0107071
```

```
# USING STUDENTIZED MAX MODULUS
smm_user <- function(colnum, rownum, m){
  r1 = weighted.mean(m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r2 = 1 / weighted.mean(1 / m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r3 = weighted.mean(m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r4 = 1 / weighted.mean(1 / m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  rl = mean(r1,r2)
  rr = mean(r3,r4)
  c1 = weighted.mean(c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  c2 = 1 / weighted.mean(1 / c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  return(round(mean(c1,c2),3))
}

#g=31 dof = 64-31-3-1 #
(multiplier2 = smm_user(colnum = c(24,29,30),
                        rownum = c(28,31,36),
                        m = matrix(c(3.156,3.251,3.104,3.196),2,2))) # g =31, v = 64-31-3-1
```

```
## [1] 3.147
```

```
boundary_pt2 =   multiplier2*sqrt(var_hat_/2)
avg_coeff_up2 = avg_coeff_+boundary_pt2
avg_coeff_dwn2 = avg_coeff_-boundary_pt2
CI_interval2 = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn2, Up

#####################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval2
```

```
##           variable_      estimate       Lwr_       Up_
## A                 A  2.604167e-01 -0.4953193 1.0161526
## B                 B  1.041667e-01 -0.6515693 0.8599026
## C                 C  2.427083e+00  1.6713474 3.1828193
## D                 D -1.979167e-01 -0.9536526 0.5578193
## E                 E  6.041667e-01 -0.1515693 1.3599026
## A:B             A:B -8.333333e-02 -0.8390693 0.6724026
## A:C             A:C  4.895833e-01 -0.2661526 1.2453193
## A:D             A:D  3.645833e-01 -0.3911526 1.1203193
## A:E             A:E  1.666667e-01 -0.5890693 0.9224026
## B:C             B:C  1.458333e-01 -0.6099026 0.9015693
## B:D             B:D -1.041667e-01 -0.8599026 0.6515693
## B:E             B:E  1.979167e-01 -0.5578193 0.9536526
## C:D             C:D  7.291667e-02 -0.6828193 0.8286526
## C:E             C:E -1.458333e-01 -0.9015693 0.6099026
## D:E             D:E -3.333333e-01 -1.0890693 0.4224026
## A:B:C         A:B:C -4.166667e-02 -0.7974026 0.7140693
## A:B:D         A:B:D  4.166667e-02 -0.7140693 0.7974026
```

```
## A:B:E         A:B:E  4.270833e-01 -0.3286526 1.1828193
## A:C:D         A:C:D -1.145833e-01 -0.8703193 0.6411526
## A:C:E         A:C:E  4.166667e-02 -0.7140693 0.7974026
## A:D:E         A:D:E  2.083333e-02 -0.7349026 0.7765693
## B:C:D         B:C:D -1.041667e-01 -0.8599026 0.6515693
## B:C:E         B:C:E -1.562500e-01 -0.9119860 0.5994860
## B:D:E         B:D:E  4.895833e-01 -0.2661526 1.2453193
## C:D:E         C:D:E -4.166667e-02 -0.7974026 0.7140693
## A:B:C:D     A:B:C:D -2.983724e-16 -0.7557360 0.7557360
## A:B:C:E     A:B:C:E  1.562500e-01 -0.5994860 0.9119860
## A:B:D:E     A:B:D:E -1.145833e-01 -0.8703193 0.6411526
## A:C:D:E     A:C:D:E  1.458333e-01 -0.6099026 0.9015693
## B:C:D:E     B:C:D:E -1.979167e-01 -0.9536526 0.5578193
## A:B:C:D:E A:B:C:D:E  2.395833e-01 -0.5161526 0.9953193
```

```
# calculating variance with anova
# Model with c only analysis
subset_dat_$block =files_1$Block_ABCDE
subset_dat_2$block =files_2$ABED
append_data_ = rbind(subset_dat_,subset_dat_2)
block_variable_ = c(rep(1,16),rep(2,16),rep(3,16),rep(4,16))
append_data_$final_block = block_variable_

A3 = append_data_$A
B3 = append_data_$B
C3 = append_data_$C
D3 = append_data_$D
E3 = append_data_$E
Y3 = append_data_$y
full_data = data.frame(y = Y3,A=A3,B=B3,C=C3,D=D3,E=E3)

m_fit_5 = lm(y~A*B*C*D*E+as.factor(block_variable_),data=full_data)
#summary(m_fit_3)
#anova(m_fit_3)
anova(m_fit_5)
```

```
## Analysis of Variance Table
##
## Response: y
##                            Df Sum Sq Mean Sq  F value    Pr(>F)
## A                           1  1.085   1.085   1.1760   0.28711
## B                           1  0.174   0.174   0.1882   0.66767
## C                           1 94.252  94.252 102.1463 5.184e-11 ***
## D                           1  0.627   0.627   0.6792   0.41658
## E                           1  5.840   5.840   6.3295   0.01767 *
## as.factor(block_variable_)  3  2.391   0.797   0.8636   0.47106
## A:B                         1  0.111   0.111   0.1204   0.73109
## A:C                         1  3.835   3.835   4.1563   0.05069 .
## B:C                         1  0.340   0.340   0.3688   0.54840
## A:D                         1  2.127   2.127   2.3049   0.13980
## B:D                         1  0.174   0.174   0.1882   0.66767
## C:D                         1  0.085   0.085   0.0922   0.76357
## A:E                         1  0.444   0.444   0.4817   0.49319
## B:E                         1  0.627   0.627   0.6792   0.41658
## C:E                         1  0.340   0.340   0.3688   0.54840
```

```
## D:E                         1  1.778   1.778   1.9267   0.17569
## A:B:C                       1  0.028   0.028   0.0301   0.86346
## A:B:D                       1  0.028   0.028   0.0301   0.86346
## A:C:D                       1  0.210   0.210   0.2277   0.63684
## B:C:D                       1  0.174   0.174   0.1882   0.66767
## A:B:E                       1  2.918   2.918   3.1628   0.08582 .
## A:C:E                       1  0.028   0.028   0.0301   0.86346
## B:C:E                       1  0.391   0.391   0.4233   0.52040
## A:D:E                       1  0.007   0.007   0.0075   0.93146
## B:D:E                       1  3.835   3.835   4.1563   0.05069 .
## C:D:E                       1  0.028   0.028   0.0301   0.86346
## A:B:C:D                     1  0.000   0.000   0.0000   1.00000
## A:B:C:E                     1  0.391   0.391   0.4233   0.52040
## A:B:D:E                     1  3.781   3.781   4.0980   0.05223 .
## A:C:D:E                     1  0.340   0.340   0.3688   0.54840
## B:C:D:E                     1  0.627   0.627   0.6792   0.41658
## A:B:C:D:E                   1  0.420   0.420   0.4553   0.50516
## Residuals                  29 26.759   0.923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(sigma2 = anova(m_fit_5)[33,3])
```

```
## [1] 0.9227131
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.2401449
```

```
print(paste0("standard deviation using ANOVA: ", sd_user))
```

```
## [1] "standard deviation using ANOVA: 0.240144894101114"
```

```
(effects_user = 2 * m_fit_5$coefficients)
```

```
##                  (Intercept)                            A
##                1.133333e+01                2.604167e-01
##                            B                            C
##                1.041667e-01                2.427083e+00
##                            D                            E
##               -1.979167e-01                6.041667e-01
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                4.166667e-02               -1.354167e+00
## as.factor(block_variable_)4                          A:B
##                9.375000e-01               -8.333333e-02
##                          A:C                          B:C
##                4.895833e-01                1.458333e-01
##                          A:D                          B:D
##                3.645833e-01               -1.041667e-01
##                          C:D                          A:E
##                7.291667e-02                1.666667e-01
##                          B:E                          C:E
##                1.979167e-01               -1.458333e-01
##                          D:E                        A:B:C
##               -3.333333e-01               -4.166667e-02
##                        A:B:D                        A:C:D
##                4.166667e-02               -1.145833e-01
```

```
##                       B:C:D                       A:B:E
##               -1.041667e-01                4.270833e-01
##                       A:C:E                       B:C:E
##                4.166667e-02               -1.562500e-01
##                       A:D:E                       B:D:E
##                2.083333e-02                4.895833e-01
##                       C:D:E                     A:B:C:D
##               -4.166667e-02                3.882727e-16
##                     A:B:C:E                     A:B:D:E
##                1.562500e-01               -6.875000e-01
##                     A:C:D:E                     B:C:D:E
##                1.458333e-01               -1.979167e-01
##                   A:B:C:D:E
##                2.291667e-01
```

```
(multiplier1 = qt(0.1/2/31,df = 29,lower.tail = F))
```

```
## [1] 3.211077
```

```
####################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)

sim_ci = cbind(effects_user - multiplier2 * sd_user, effects_user + multiplier2 * sd_user)
colnames(sim_ci) <- c('Lower Bound','Upper Bound')
sim_ci
```

```
##                              Lower Bound Upper Bound
## (Intercept)                   10.5775974 12.08906932
## A                             -0.4953193  1.01615265
## B                             -0.6515693  0.85990265
## C                              1.6713474  3.18281932
## D                             -0.9536526  0.55781932
## E                             -0.1515693  1.35990265
## as.factor(block_variable_)2   -0.7140693  0.79740265
## as.factor(block_variable_)3   -2.1099026 -0.59843068
## as.factor(block_variable_)4    0.1817640  1.69323598
## A:B                           -0.8390693  0.67240265
## A:C                           -0.2661526  1.24531932
## B:C                           -0.6099026  0.90156932
## A:D                           -0.3911526  1.12031932
## B:D                           -0.8599026  0.65156932
## C:D                           -0.6828193  0.82865265
## A:E                           -0.5890693  0.92240265
## B:E                           -0.5578193  0.95365265
## C:E                           -0.9015693  0.60990265
## D:E                           -1.0890693  0.42240265
## A:B:C                         -0.7974026  0.71406932
## A:B:D                         -0.7140693  0.79740265
## A:C:D                         -0.8703193  0.64115265
## B:C:D                         -0.8599026  0.65156932
## A:B:E                         -0.3286526  1.18281932
## A:C:E                         -0.7140693  0.79740265
## B:C:E                         -0.9119860  0.59948598
## A:D:E                         -0.7349026  0.77656932
## B:D:E                         -0.2661526  1.24531932
```

```
## C:D:E                        -0.7974026  0.71406932
## A:B:C:D                      -0.7557360  0.75573598
## A:B:C:E                      -0.5994860  0.91198598
## A:B:D:E                      -1.4432360  0.06823598
## A:C:D:E                      -0.6099026  0.90156932
## B:C:D:E                      -0.9536526  0.55781932
## A:B:C:D:E                    -0.5265693  0.98490265
```

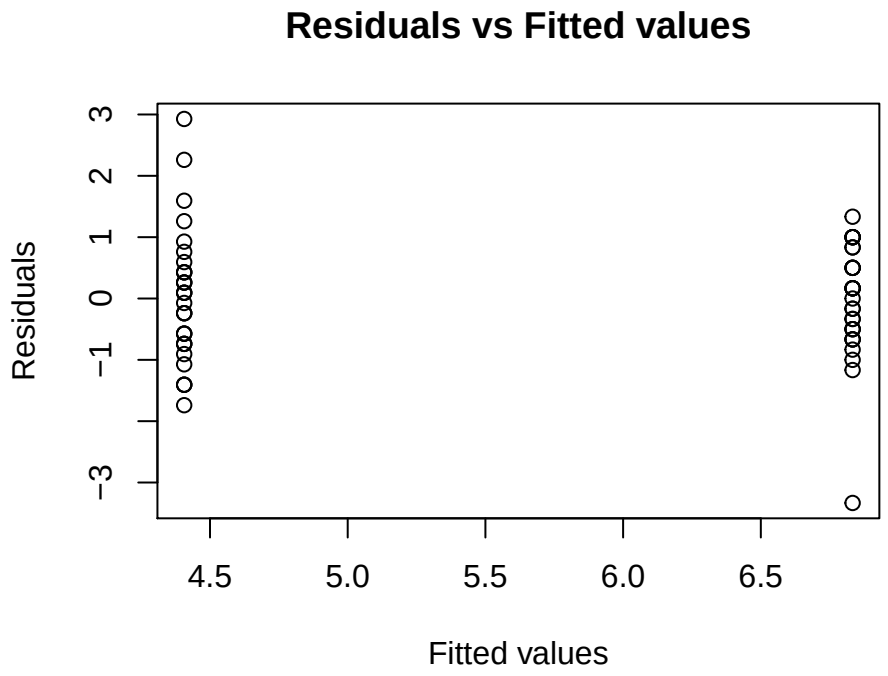
```
sim_ci2 = cbind(effects_user - multiplier1 * sd_user, effects_user + multiplier1 * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
#################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                  10.5622096 12.10445708
## A                            -0.5107071  1.03154042
## B                            -0.6669571  0.87529042
## C                             1.6559596  3.19820708
## D                            -0.9690404  0.57320708
## E                            -0.1669571  1.37529042
## as.factor(block_variable_)2  -0.7294571  0.81279042
## as.factor(block_variable_)3  -2.1252904 -0.58304292
## as.factor(block_variable_)4   0.1663763  1.70862375
## A:B                          -0.8544571  0.68779042
## A:C                          -0.2815404  1.26070708
## B:C                          -0.6252904  0.91695708
## A:D                          -0.4065404  1.13570708
## B:D                          -0.8752904  0.66695708
## C:D                          -0.6982071  0.84404042
## A:E                          -0.6044571  0.93779042
## B:E                          -0.5732071  0.96904042
## C:E                          -0.9169571  0.62529042
## D:E                          -1.1044571  0.43779042
## A:B:C                        -0.8127904  0.72945708
## A:B:D                        -0.7294571  0.81279042
## A:C:D                        -0.8857071  0.65654042
## B:C:D                        -0.8752904  0.66695708
## A:B:E                        -0.3440404  1.19820708
## A:C:E                        -0.7294571  0.81279042
## B:C:E                        -0.9273737  0.61487375
## A:D:E                        -0.7502904  0.79195708
## B:D:E                        -0.2815404  1.26070708
## C:D:E                        -0.8127904  0.72945708
## A:B:C:D                      -0.7711237  0.77112375
## A:B:C:E                      -0.6148737  0.92737375
## A:B:D:E                      -1.4586237  0.08362375
## A:C:D:E                      -0.6252904  0.91695708
## B:C:D:E                      -0.9690404  0.57320708
## A:B:C:D:E                    -0.5419571  1.00029042
```

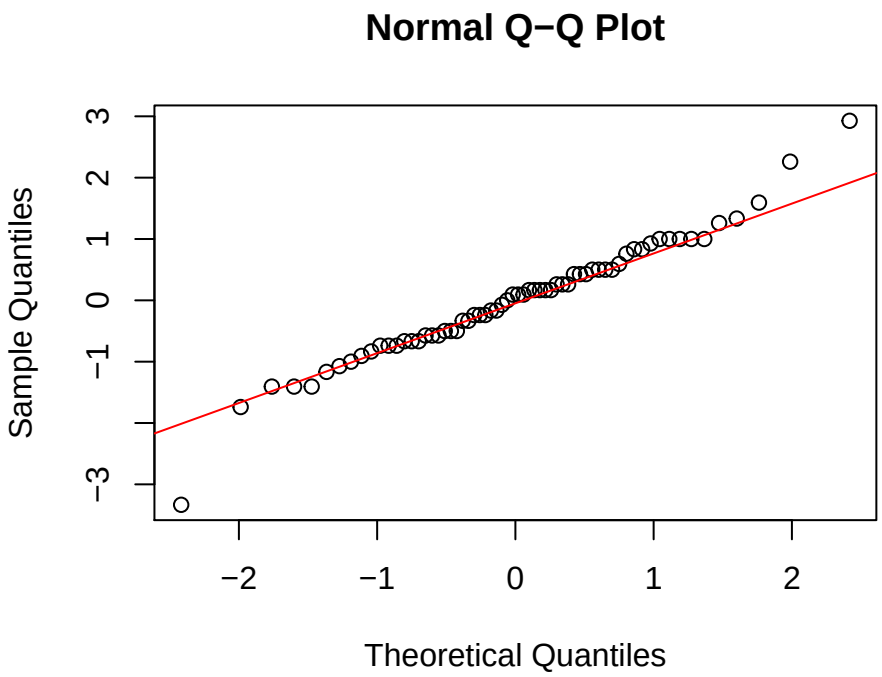
```
######################
```

SECTION A.2.2.3 :FITTING THE MODEL AND DIAGNOSTIC PLOTS

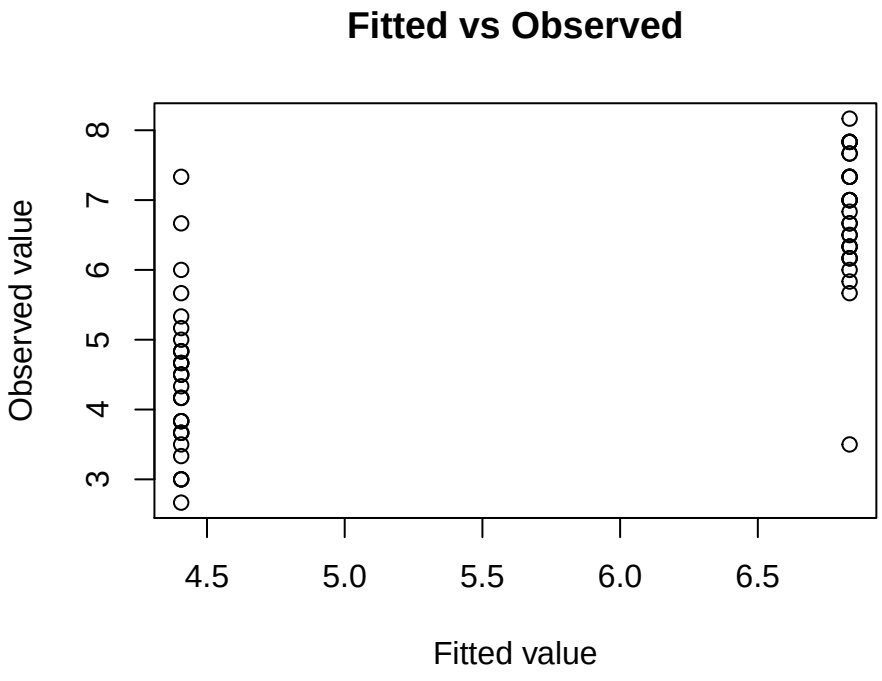
```
m_fit_6 = lm(y~C ,data=full_data)
resid = m_fit_6$resid
fitted = m_fit_6$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



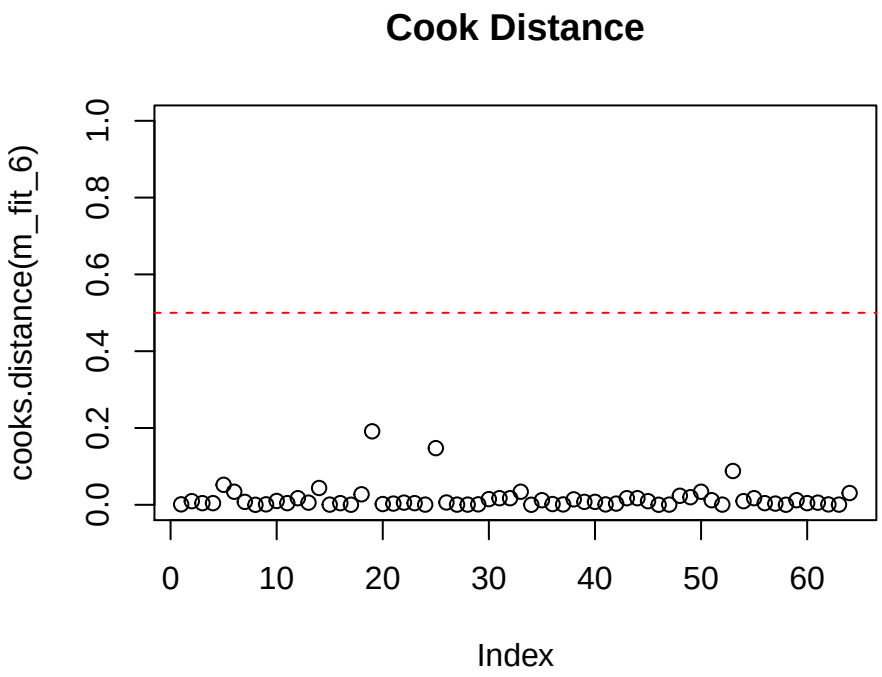

```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
plot(cooks.distance(m_fit_6), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.2.2.4 : STUDYING THE EFFECT OF TRANSFORMATIONS

```
A= full_data$A
B= full_data$B
C= full_data$C
D= full_data$D
E= full_data$E
y = full_data$y

y0=full_data$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+as.factor(block_variable_))
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda
```

```
## [1] 1.298172
```

```r
y_t=y0^lambda
model_trans2=lm(y_t~ A+B+C+D+E+as.factor(block_variable_))
anova(model_trans2)
```

```
## Analysis of Variance Table
##
## Response: y_t
##                            Df Sum Sq Mean Sq  F value    Pr(>F)
## A                           1   6.15    6.15   1.4883   0.22768
## B                           1   0.70    0.70   0.1700   0.68171
## C                           1 438.49  438.49 106.1591 1.892e-14 ***
## D                           1   3.01    3.01   0.7296   0.39671
## E                           1  25.36   25.36   6.1388   0.01632 *
## as.factor(block_variable_)  3  11.37    3.79   0.9172   0.43868
## Residuals                  55 227.18    4.13
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
print(round(p.adjust(summary(model_trans2)$coefficients[,4], method = c("holm")),4)) # AT ALPHA = 0.1 L
```

```
##                 (Intercept)                           A
##                      0.0000                      1.0000
##                           B                           C
##                      1.0000                      0.0000
##                           D                           E
##                      1.0000                      0.1143
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                      1.0000                      1.0000
## as.factor(block_variable_)4
##                      1.0000
```

```r
# After box cox with block variable then finally giving significant variable shouud we also take
# g = 5, dof = 64-5-3-1
(multiplier_trans = smm_user(colnum = c(40,55,60),
                      rownum = c(3,5,6),
                      m = matrix(c(2.183,2.470,2.160,2.439),2,2))) # g = 5, v = 64-5-3-1
```

```
## [1] 2.353
```

```r
(sigma2 = anova(model_trans2)[7,3])
```

```
## [1] 4.130532
```

```r
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.5080928
```

```r
(effects_user = 2 * model_trans2$coefficients)
```

```
##                 (Intercept)                           A
##                  18.8168710                  0.6198589
##                           B                           C
##                   0.2094925                  5.2350605
##                           D                           E
##                  -0.4340073                  1.2588842
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                   0.9775908                 -0.9852544
## as.factor(block_variable_)4
```

```
##                   1.1019785
```

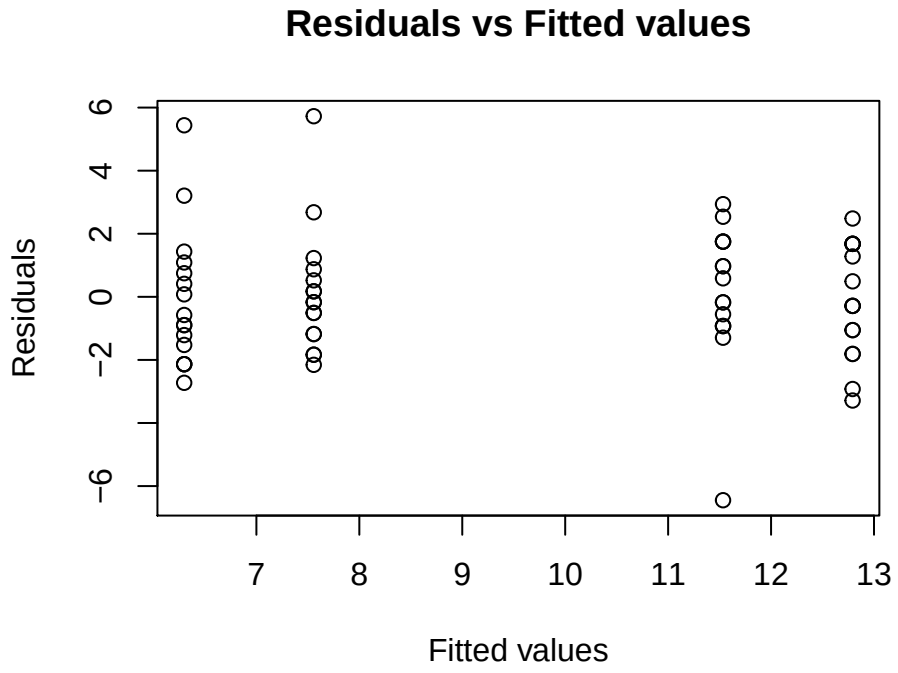
```
sim_ci2 = cbind(effects_user - multiplier_trans * sd_user, effects_user + multiplier_trans * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')

#############################################################
## STUDENTIZED MAXIMUM MODULUS CI FOR TRANSFORMED DATA
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                 17.62132872  20.0124133
## A                           -0.57568341   1.8154012
## B                           -0.98604983   1.4050348
## C                            4.03951819   6.4306028
## D                           -1.62954958   0.7615350
## E                            0.06334188   2.4544265
## as.factor(block_variable_)2 -0.21795149   2.1731331
## as.factor(block_variable_)3 -2.18079674   0.2102879
## as.factor(block_variable_)4 -0.09356384   2.2975208
```

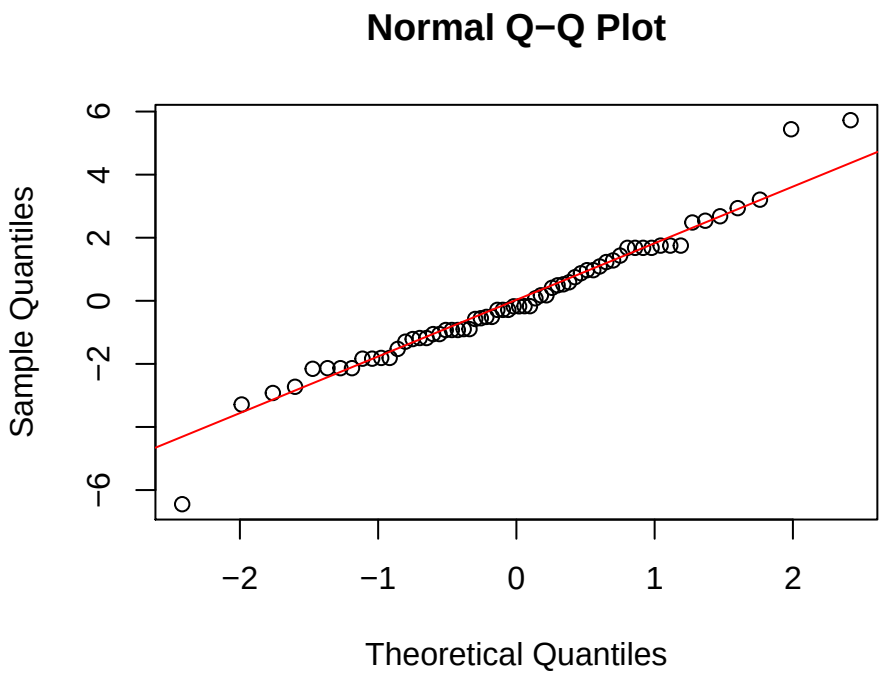
```
final_transformed_model = lm(y_t~C+E)
2 * final_transformed_model$coefficients
```

```
## (Intercept)           C           E
##   19.090450    5.235061    1.258884
```

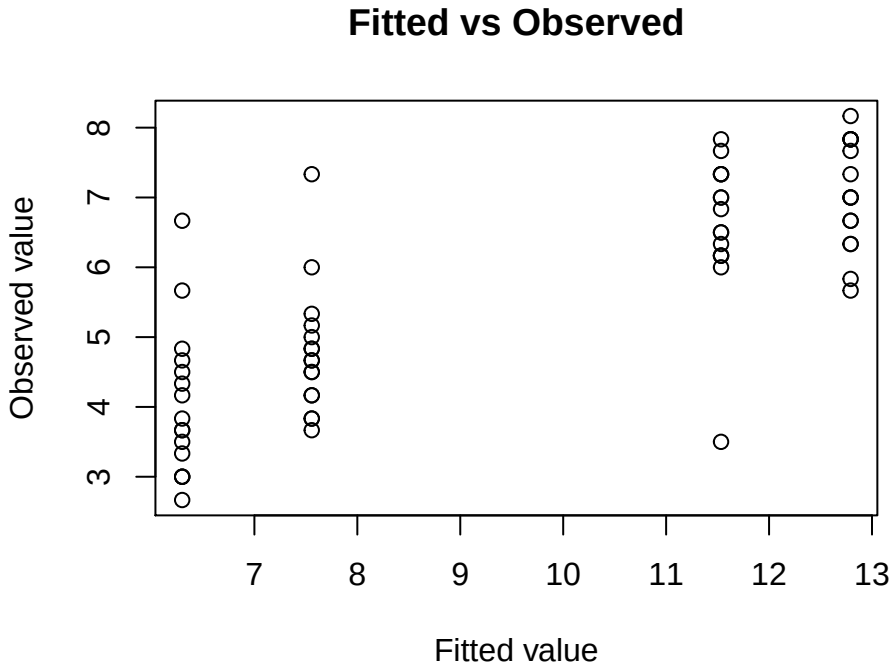
```
resid = final_transformed_model$resid
fitted = final_transformed_model$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



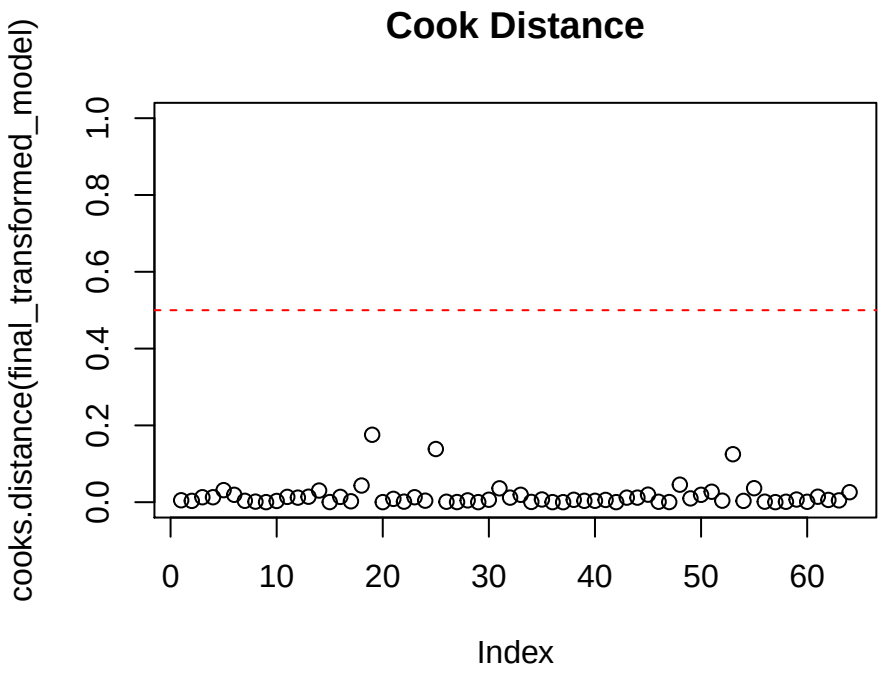
```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```



```
plot(cooks.distance(final_transformed_model), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



ANALYSIS_JUDGE_AD

Abhijeet Bhardwaj

4/28/2021

SECTION A.3: ANALYSIS OF SCORES BY JUDGE AD

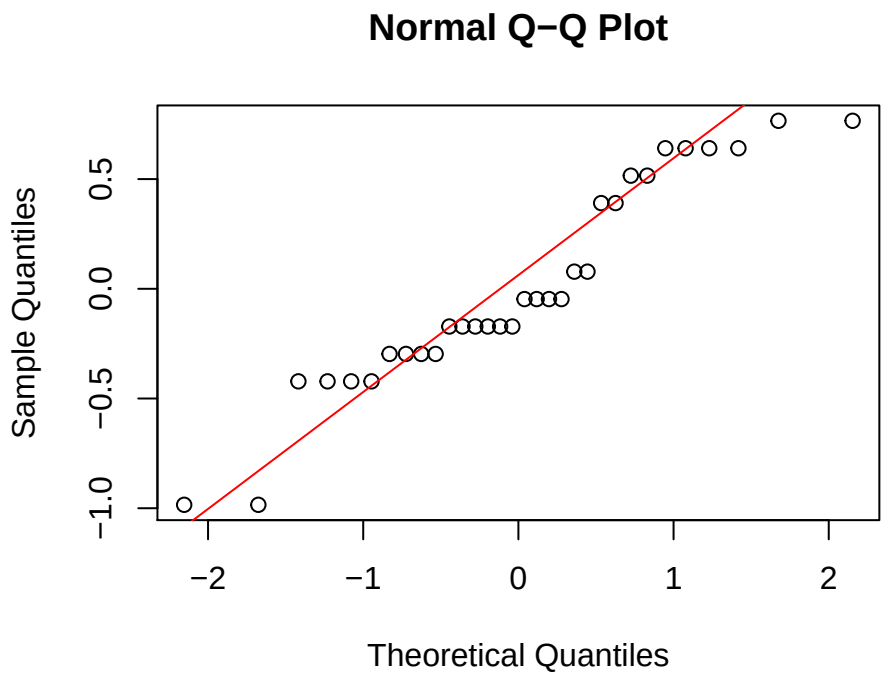
SECTION A.3.1 :Analysis of first run

#SECTION A.3.1.1 DATA VISUALIZATION

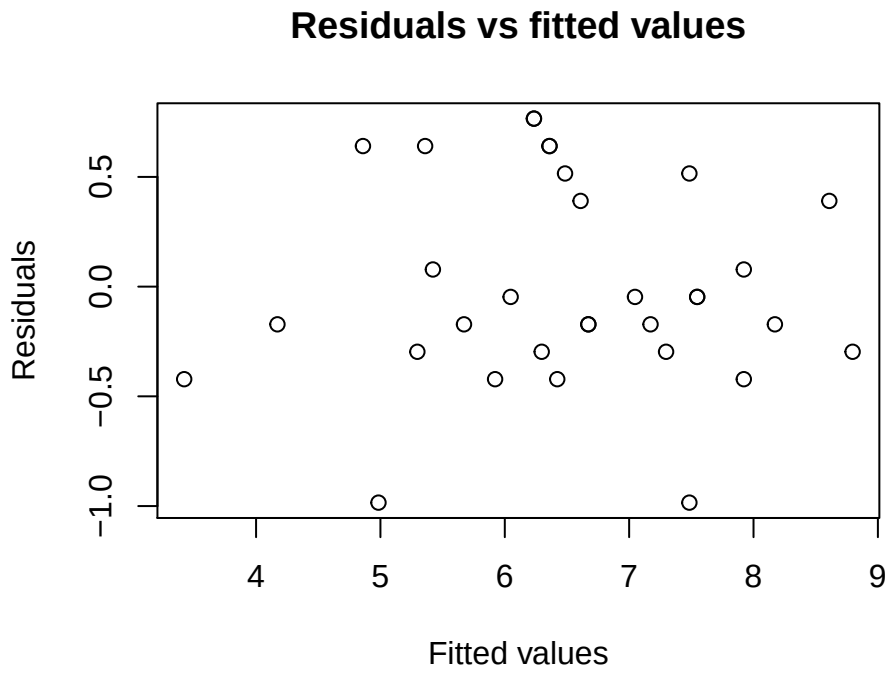
```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
#files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3
A = files_1$A
B = files_1$B
C = files_1$C
D = files_1$D
E = files_1$E
block = files_1$Block_ABCDE
avg_score = files_1$J_AD
order_scoring = files_1$O_AD
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^3+block,data=subset_dat_)
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=6) to new 7 after EncodeVars() -- should no longer happen!
```

```
resid = m_fit_1$resid
fitted = m_fit_1$fit
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs fitted values")
```



SECTION A.3.1.2 ANALYZING SIGNIFICANCE USING CONFIDENCE INTERVALS

```
# ASSUMING 4 FI's TO BE ZERO but not ABCDE as it is a block effect for sigma
#fitting complete model to get estimates of 4fis
m_fit_2 = lm(y~.^5,data=subset_dat_)
summary_fullmodel_ = summary(m_fit_2)
2*summary_fullmodel_$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.03125        NaN     NaN      NaN
## A           -0.28125        NaN     NaN      NaN
## B            0.03125        NaN     NaN      NaN
## C            1.40625        NaN     NaN      NaN
## D           -0.28125        NaN     NaN      NaN
## E           -0.09375        NaN     NaN      NaN
## A:B          0.09375        NaN     NaN      NaN
## A:C          0.84375        NaN     NaN      NaN
## A:D          0.28125        NaN     NaN      NaN
## A:E          0.09375        NaN     NaN      NaN
## B:C         -0.09375        NaN     NaN      NaN
## B:D         -0.28125        NaN     NaN      NaN
## B:E          0.53125        NaN     NaN      NaN
## C:D          0.46875        NaN     NaN      NaN
## C:E          0.03125        NaN     NaN      NaN
## D:E          0.09375        NaN     NaN      NaN
## A:B:C       -0.28125        NaN     NaN      NaN
## A:B:D       -0.09375        NaN     NaN      NaN
## A:B:E        0.09375        NaN     NaN      NaN
## A:C:D       -0.21875        NaN     NaN      NaN
## A:C:E        0.71875        NaN     NaN      NaN
## A:D:E        0.40625        NaN     NaN      NaN
## B:C:D        0.21875        NaN     NaN      NaN
## B:C:E        0.15625        NaN     NaN      NaN
## B:D:E        0.46875        NaN     NaN      NaN
## C:D:E       -0.65625        NaN     NaN      NaN
## A:B:C:D     -0.34375        NaN     NaN      NaN
## A:B:C:E      0.21875        NaN     NaN      NaN
## A:B:D:E     -0.59375        NaN     NaN      NaN
## A:C:D:E     -0.34375        NaN     NaN      NaN
## B:C:D:E      0.46875        NaN     NaN      NaN
## A:B:C:D:E    0.90625        NaN     NaN      NaN
```

```
coeffs_4fis = 2*summary_fullmodel_$coefficients[27:31,1]
sd_4 = sqrt(sum(coeffs_4fis^2)/5)
threshold_4 = sd_4*qt(0.1/(2*25),5,lower.tail = FALSE)
all_coeffs_ = 2*summary_fullmodel_$coefficients[,1]
eff = as.numeric(all_coeffs_)
which(abs(eff)>threshold_4) # only intercept
```

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

```
# ASSUMING 3 and 4 FI's TO BE ZERO but not ABCDE as it is a block effect for estimating sigma
#fitting complete model to get estimates of 4fis
coeffs_34fis = 2*summary_fullmodel_$coefficients[17:31,1]
```

```r
var_34 = sum(coeffs_34fis^2)/15
sd_34 = sqrt(var_34)
threshold_34 = sd_4*qt(0.1/(2*15),15,lower.tail = FALSE)
threshold_34
```

```
## [1] 1.301854
```

```r
which(abs(eff)>threshold_34) # C significant
```

```
## [1] 1 4
```

SECTION A.3.1.3 ANALYZING SIGNIFICANCE USING DANIEL METHOD

```r
## APPLY DANIEL METHOD
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

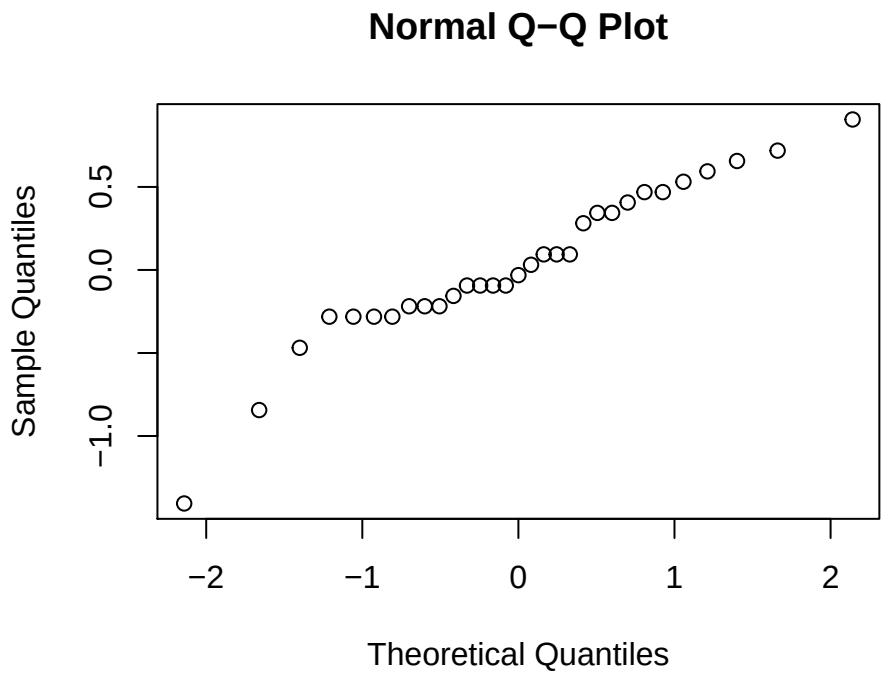
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^5,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 17
```

```
## [1] 17
```

```r
A_final = perm[1,17] * A
B_final = perm[2,17] * B
C_final = perm[3,17] * C
D_final = perm[4,17] * D
E_final = perm[5,17] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^5,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -0.92524456  0.00000000 -2.14119812  0.41598722  0.24500622 -0.32929135
##  [7] -1.66069761 -0.80754104 -0.08094729 -0.24500622 -1.21123213  1.05741423
## [13]  0.92524456  0.08094729  0.32929135 -1.05741423 -0.16242937  0.16242937
## [19] -0.60017878  1.66069761  0.70009021 -0.50593365 -0.41598722 -1.40074506
## [25]  1.40074506  0.50593365 -0.70009021  1.21123213  0.60017878  0.80754104
## [31]  2.14119812
##
## $y
##                                A_final                                B_final
##                               -0.28125                               -0.03125
##                                C_final                                D_final
##                               -1.40625                                0.28125
##                                E_final                       A_final:B_final
##                                0.09375                               -0.09375
##                       A_final:C_final                       A_final:D_final
##                               -0.84375                               -0.28125
##                       A_final:E_final                       B_final:C_final
##                               -0.09375                               -0.09375
##                       B_final:D_final                       B_final:E_final
```

```
##                                -0.28125                                 0.53125
##                          C_final:D_final                          C_final:E_final
##                                 0.46875                                 0.03125
##                          D_final:E_final                  A_final:B_final:C_final
##                                 0.09375                                -0.28125
##                  A_final:B_final:D_final                  A_final:B_final:E_final
##                                -0.09375                                 0.09375
##                  A_final:C_final:D_final                  A_final:C_final:E_final
##                                -0.21875                                 0.71875
##                  A_final:D_final:E_final                  B_final:C_final:D_final
##                                 0.40625                                -0.21875
##                  B_final:C_final:E_final                  B_final:D_final:E_final
##                                -0.15625                                -0.46875
##                  C_final:D_final:E_final          A_final:B_final:C_final:D_final
##                                 0.65625                                 0.34375
##          A_final:B_final:C_final:E_final          A_final:B_final:D_final:E_final
##                                -0.21875                                 0.59375
##          A_final:C_final:D_final:E_final          B_final:C_final:D_final:E_final
##                                 0.34375                                 0.46875
## A_final:B_final:C_final:D_final:E_final
##                                 0.90625
```

```
# Step2: Fit a lm ------------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier --------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # outliers::C_final
```

```
## C_final
##       3
```

```
x_inlier = qq_x[which(abs(qq_y) <= CC)]
y_inlier = qq_y[which(abs(qq_y) <= CC)]
# Step4: Fit another lm ------------------------------------------------------
fit2 = lm(y_inlier ~ x_inlier)
beta2 = fit2$coefficients[2]
beta1 / beta2 # 1.186
```

```
##     qq_x
## 1.094271
```

```
# From the table in the paper,
# n = 31, alpha = 0.05 -> threshold = 1.108
# So we have the significance here

m = length(x_inlier)
y_mean = mean(y_inlier)
x_mean = mean(x_inlier)
n_prime = round(31/4,1)
sigma_user = sqrt(sum((fit2$residuals)^2)/fit2$df.residual)
```

```
LHS = abs(y_inlier - y_mean + beta2 * (x_inlier - x_mean))
RHS = n_prime * sqrt(qf(p = 0.975,df1 = n_prime,df2 = m-2)) * sigma_user *
  sqrt(1 + 1/m + (x_inlier - x_mean)^2 / (var(x_inlier) * (m+1) ))
final_idx = which(LHS <= RHS)
D_ = max(abs(y_inlier[final_idx]))
c(D_, CC)
```

```
##             75%
## 0.59375 1.18750
```

```
final_threshold = max(CC,D_)
print(paste0("after daniels method :", names(which(abs(qq_y) > final_threshold)))) # again only C_final
```

```
## [1] "after daniels method :C_final"
```

```
################################################################################
```

SECTION A.3.1.4 ANALYZING SIGNIFICANCE USING DONG's METHOD

```
#ANALYSIS BY DONGS METHOD
coeff=coef(m_fit_2)*2
theta1=abs(na.omit(coeff[-c(1,31)]))
s0=1.5*median(theta1)
m1=sum(theta1<=2.5*s0)
s1=sqrt(sum(theta1[theta1<=2.5*s0]^2)/m1)
m2=sum(theta1<=2.5*s1)
s2=sqrt(sum(theta1[theta1<=2.5*s1]^2)/m2)
g=length(theta1)
gamma=0.5*(1-(1-0.05)^(1/g))
print(paste0("after DONG's method significant effect :", names(theta1[theta1>s2*qt(gamma,m2,lower.tail =
```

```
## [1] "after DONG's method significant effect :C"
```

```
################################################
```

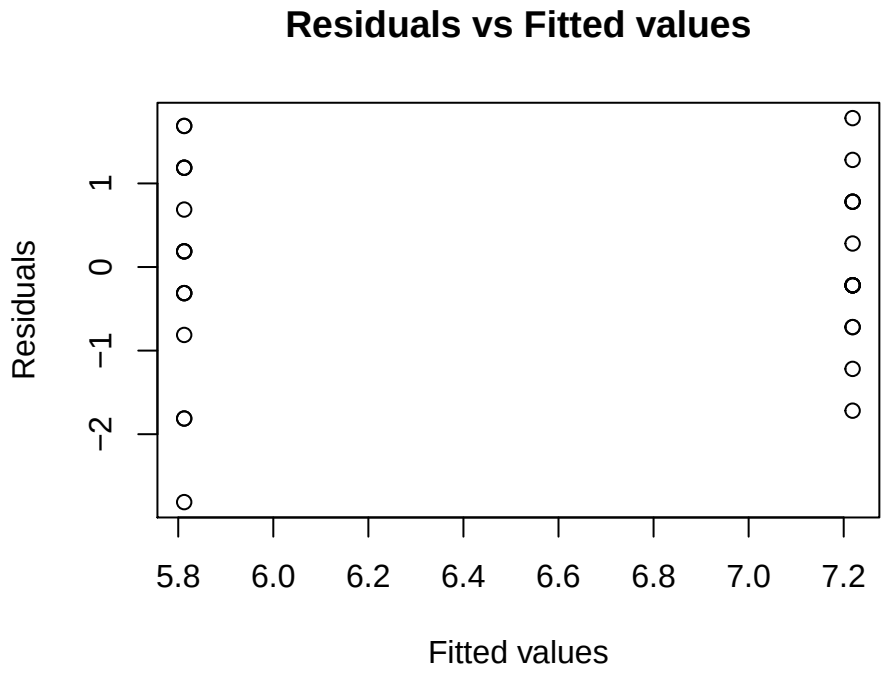
SECTION A.3.1.5 MODEL FITTING WITH SIGNIFICANT FACTOR FOR UN REPLICATED EXPERIMENT

```
# Model with c only analysis
m_fit_3 = lm(y~C,data=subset_dat_)
summary(m_fit_3)
```

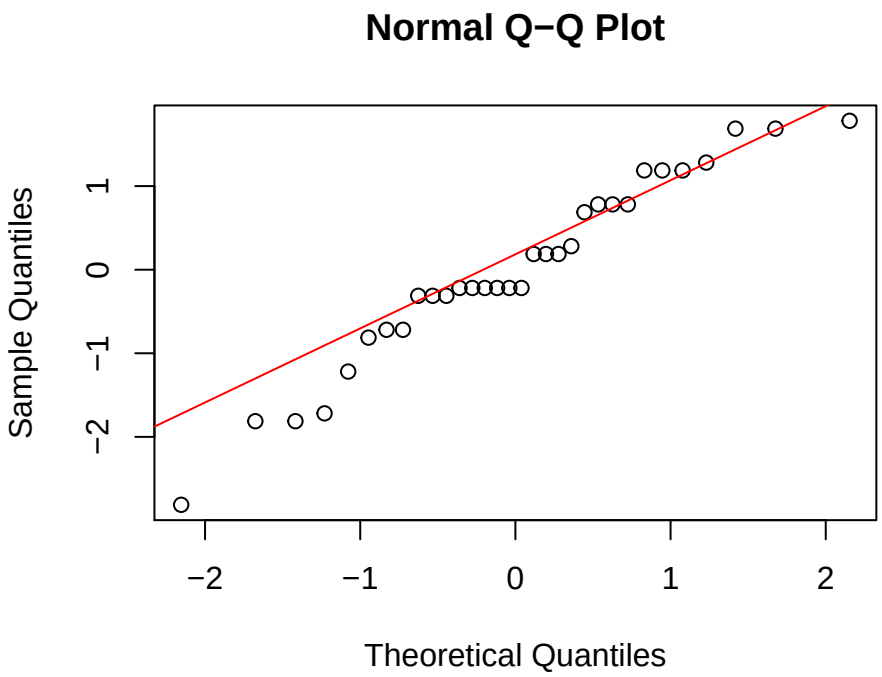
```
##
## Call:
## lm(formula = y ~ C, data = subset_dat_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8125 -0.4141 -0.2188  0.7812  1.7812
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   6.5156     0.2001  32.569  < 2e-16 ***
## C             0.7031     0.2001   3.515  0.00142 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.132 on 30 degrees of freedom
## Multiple R-squared:  0.2917, Adjusted R-squared:  0.268
## F-statistic: 12.35 on 1 and 30 DF,  p-value: 0.001421
```

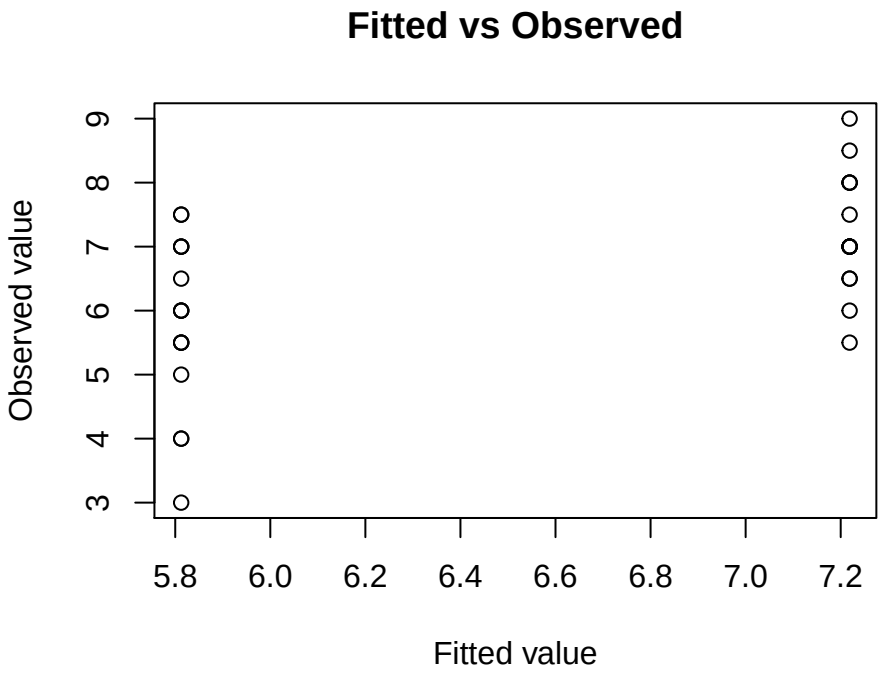
```
resid = m_fit_3$resid
fitted = m_fit_3$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



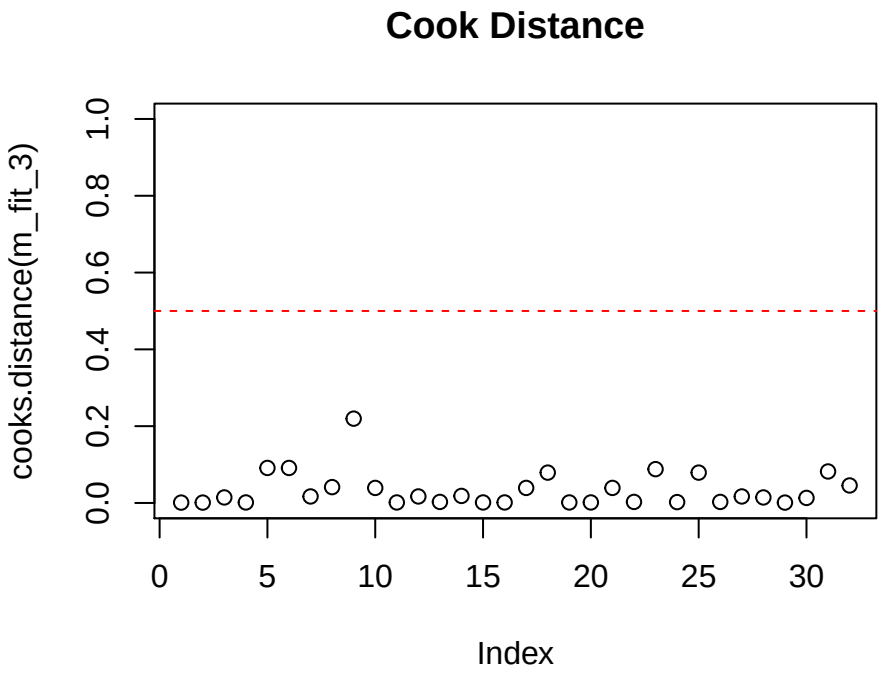

```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
# checking for outliers in the final model
plot(cooks.distance(m_fit_3), ylim =c(0,1),main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.3.1.6 APPLYING TRANSFORMATION TO SEE IF MORE SIGNIFICANT FACTORS CAN BE EXTRACTED

```
#####################
y0=subset_dat_$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+block)
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda
```

```
## [1] 1.778564
```

```
y_t=y0^lambda
#data_subs_bc_t=data.frame(y_t= y_t,A=A,B=B,C=C,D=D,E=E)
#model_trans=lm(y_t~ .^2,data_subs_bc_t)
model_trans=lm(y_t~ A+B+C+D+E+block)
```

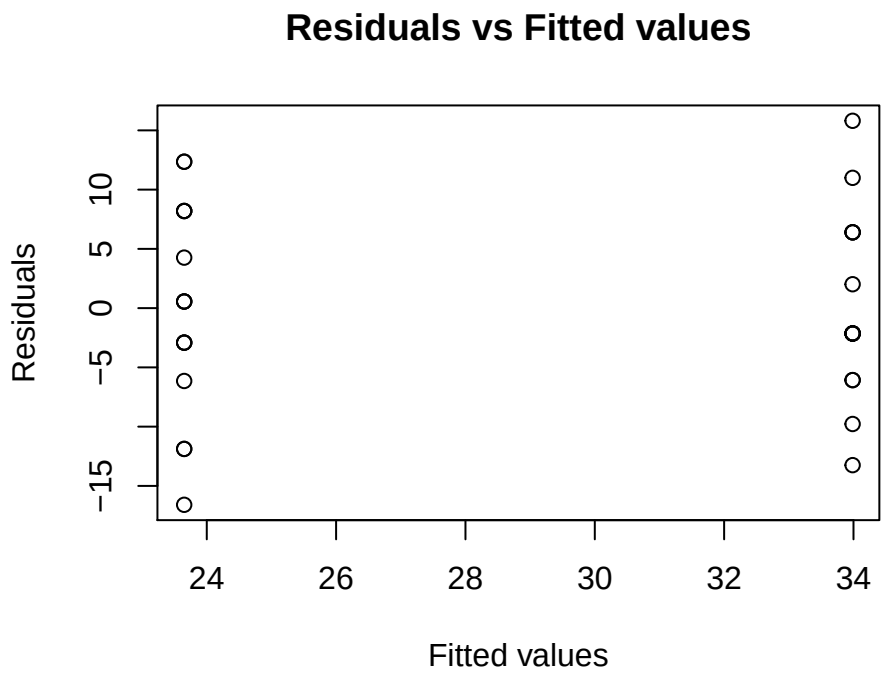
```r
summary(model_trans)
```

```
##
## Call:
## lm(formula = y_t ~ A + B + C + D + E + block)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.0460  -5.2980   0.2151   4.4606  13.8245
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.81862    1.42934  20.162  < 2e-16 ***
## A           -0.72891    1.42934  -0.510  0.61455
## B           -0.08365    1.42934  -0.059  0.95380
## C            5.16797    1.42934   3.616  0.00132 **
## D           -0.92493    1.42934  -0.647  0.52347
## E           -0.32152    1.42934  -0.225  0.82385
## block        3.23209    1.42934   2.261  0.03270 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.086 on 25 degrees of freedom
## Multiple R-squared:  0.4308, Adjusted R-squared:  0.2942
## F-statistic: 3.153 on 6 and 25 DF,  p-value: 0.01925
```

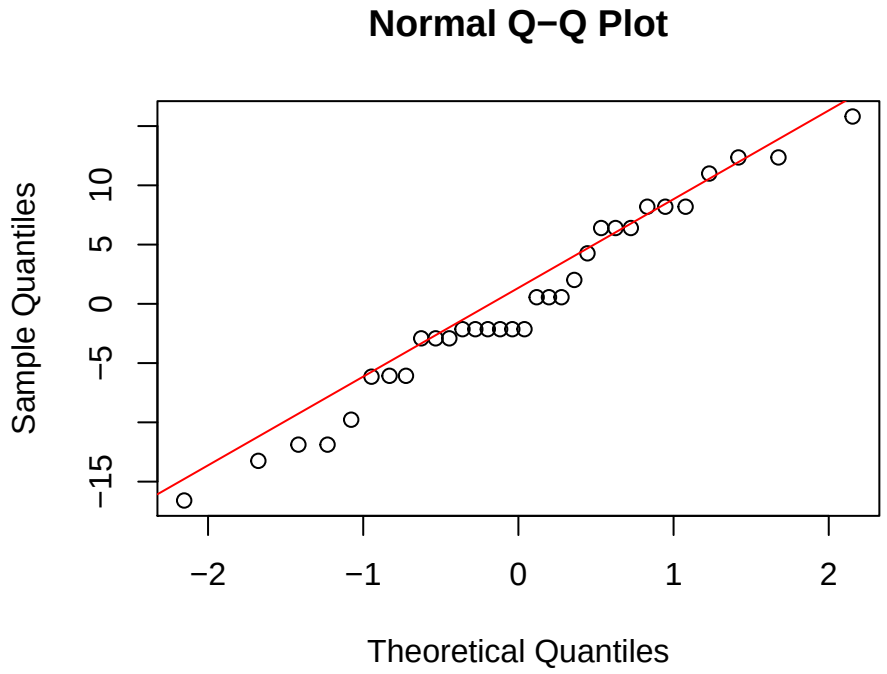
```r
print(round(p.adjust(summary(model_trans)$coefficients[,4]),4))
```

```
## (Intercept)           A           B           C           D           E
##      0.0000      1.0000      1.0000      0.0079      1.0000      1.0000
##       block
##      0.1635
```

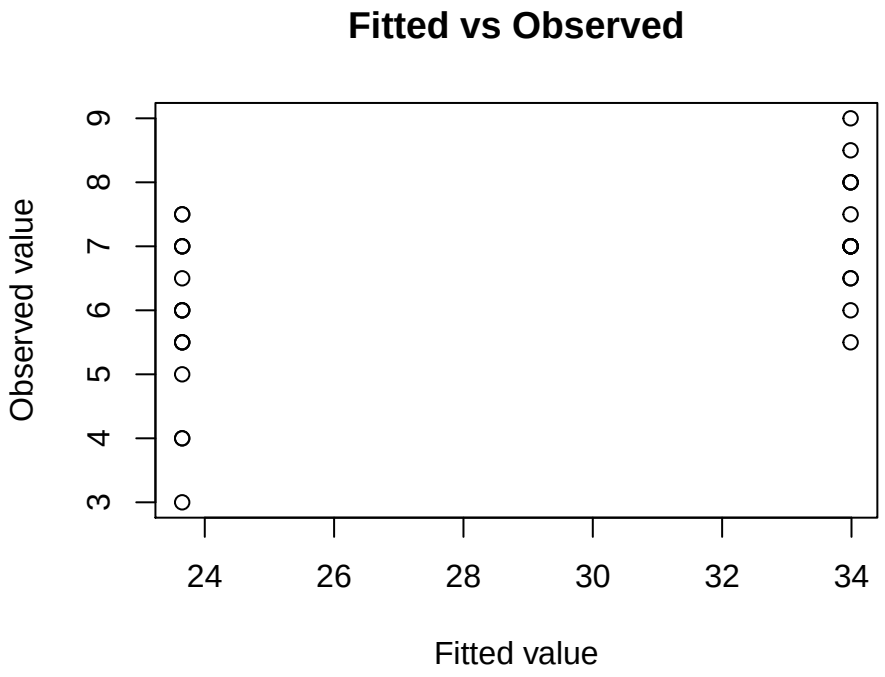
```r
model_trans=lm(y_t~C) #AS ONLY C IS SIGNIFICANT
resid = model_trans$resid
fitted = model_trans$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



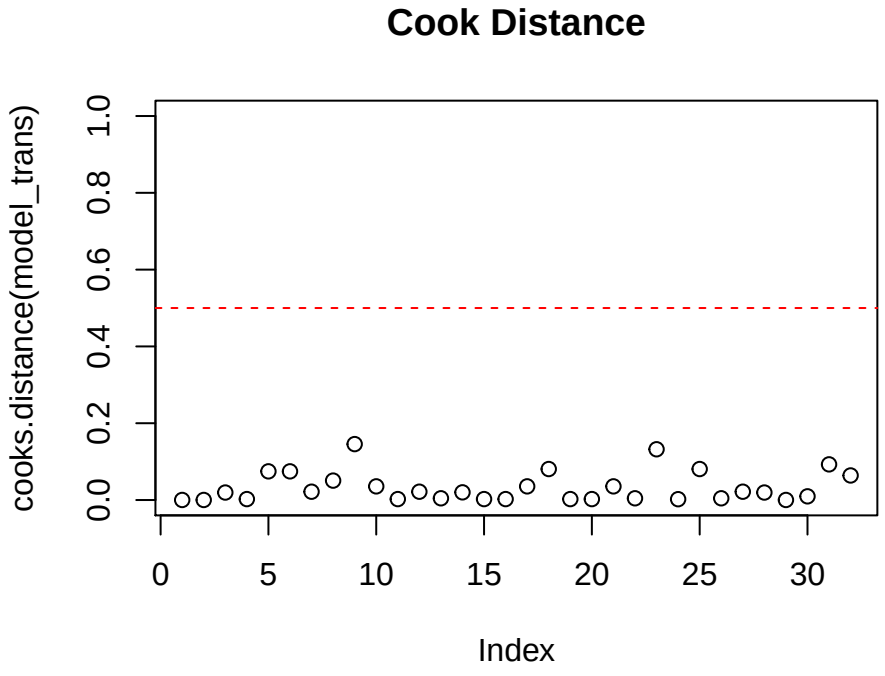

```
qqnorm(resid)
qqline(resid,col="red")
```



```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
plot(cooks.distance(model_trans), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.3.2 :Analysis of Second run and combined two runs for Judge AD

SECTION A.3.2.1 :Estimates from second run

```
# COMBINING THE SECOND COMPLETE REPLICATE

files_2 = read.csv('Run_2_with_scores.csv')
#files_2$J_AVG = (files_2$J_AD+files_2$J_AG+files_2$J_KJ)/3
A2 = files_2$A
B2 = files_2$B
C2 = files_2$C
D2 = files_2$D
E2 = files_2$E
avg_score2 = files_2$J_AD
subset_dat_2 = data.frame(y = avg_score2,A=A2,B=B2,C=C2,D=D2,E=E2)
m_fit_4 = lm(y~.^5,data=subset_dat_2)
summary_fullmodel_2 = summary(m_fit_4)
2*summary_fullmodel_2$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.40625        NaN     NaN      NaN
## A           -0.03125        NaN     NaN      NaN
## B           -0.40625        NaN     NaN      NaN
## C            2.71875        NaN     NaN      NaN
## D           -0.15625        NaN     NaN      NaN
## E            0.28125        NaN     NaN      NaN
## A:B         -0.84375        NaN     NaN      NaN
## A:C         -0.59375        NaN     NaN      NaN
## A:D          0.53125        NaN     NaN      NaN
## A:E          0.46875        NaN     NaN      NaN
## B:C          0.15625        NaN     NaN      NaN
## B:D          0.03125        NaN     NaN      NaN
## B:E          0.71875        NaN     NaN      NaN
## C:D          0.03125        NaN     NaN      NaN
## C:E         -0.65625        NaN     NaN      NaN
## D:E         -0.65625        NaN     NaN      NaN
## A:B:C        1.09375        NaN     NaN      NaN
## A:B:D       -0.03125        NaN     NaN      NaN
## A:B:E        0.15625        NaN     NaN      NaN
## A:C:D        0.09375        NaN     NaN      NaN
## A:C:E       -0.34375        NaN     NaN      NaN
## A:D:E       -0.34375        NaN     NaN      NaN
## B:C:D       -0.28125        NaN     NaN      NaN
## B:C:E        0.28125        NaN     NaN      NaN
## B:D:E        1.03125        NaN     NaN      NaN
## C:D:E        0.78125        NaN     NaN      NaN
## A:B:C:D     -0.21875        NaN     NaN      NaN
## A:B:C:E      0.09375        NaN     NaN      NaN
## A:B:D:E     -0.15625        NaN     NaN      NaN
## A:C:D:E      0.46875        NaN     NaN      NaN
## B:C:D:E     -0.78125        NaN     NaN      NaN
## A:B:C:D:E    0.15625        NaN     NaN      NaN
```

SECTION A.3.2.2 :Calculating Confidence Intervals by combining the replicates

```
# CACULATION OF VARIANCE BY FIRST PRINCIPLES
coeffs_1st_rep = 2*summary_fullmodel_$coefficients[-1,1]
coeffs_2st_rep = 2*summary_fullmodel_2$coefficients[-1,1]
avg_coeff_ = (coeffs_1st_rep+coeffs_2st_rep)/2
diff_1 = (coeffs_1st_rep-avg_coeff_)^2
diff_2 = (coeffs_2st_rep-avg_coeff_)^2
diff_1_ = diff_1[-c(31,28)]
diff_2_ = diff_2[-c(31,28)]

var_ = (diff_1_+diff_2_)/(2-1)
var_hat_ = sum(var_)/29 #estimate of 4sigma^2/2^k) \hat{var(Ai)}
multiplier1 = qt(0.1/2/31,df = 64-31-3-1,lower.tail = F)
boundary_pt =   multiplier1*sqrt(var_hat_/2)
print(paste0("standard deviation using first principle: ", sqrt(var_hat_/2)))
```

```
## [1] "standard deviation using first principle: 0.363092188706945"
```

```
avg_coeff_up = avg_coeff_+boundary_pt
avg_coeff_dwn = avg_coeff_-boundary_pt
CI_interval = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn, Up_=

#####################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval
```

```
##           variable_ estimate      Lwr_      Up_
## A                 A -0.15625 -1.322167 1.009667
## B                 B -0.18750 -1.353417 0.978417
## C                 C  2.06250  0.896583 3.228417
## D                 D -0.21875 -1.384667 0.947167
## E                 E  0.09375 -1.072167 1.259667
## A:B             A:B -0.37500 -1.540917 0.790917
## A:C             A:C  0.12500 -1.040917 1.290917
## A:D             A:D  0.40625 -0.759667 1.572167
## A:E             A:E  0.28125 -0.884667 1.447167
## B:C             B:C  0.03125 -1.134667 1.197167
## B:D             B:D -0.12500 -1.290917 1.040917
## B:E             B:E  0.62500 -0.540917 1.790917
## C:D             C:D  0.25000 -0.915917 1.415917
## C:E             C:E -0.31250 -1.478417 0.853417
## D:E             D:E -0.28125 -1.447167 0.884667
## A:B:C         A:B:C  0.40625 -0.759667 1.572167
## A:B:D         A:B:D -0.06250 -1.228417 1.103417
## A:B:E         A:B:E  0.12500 -1.040917 1.290917
## A:C:D         A:C:D -0.06250 -1.228417 1.103417
## A:C:E         A:C:E  0.18750 -0.978417 1.353417
## A:D:E         A:D:E  0.03125 -1.134667 1.197167
## B:C:D         B:C:D -0.03125 -1.197167 1.134667
## B:C:E         B:C:E  0.21875 -0.947167 1.384667
## B:D:E         B:D:E  0.75000 -0.415917 1.915917
## C:D:E         C:D:E  0.06250 -1.103417 1.228417
```

```
## A:B:C:D     A:B:C:D -0.28125 -1.447167 0.884667
## A:B:C:E     A:B:C:E  0.15625 -1.009667 1.322167
## A:B:D:E     A:B:D:E -0.37500 -1.540917 0.790917
## A:C:D:E     A:C:D:E  0.06250 -1.103417 1.228417
## B:C:D:E     B:C:D:E -0.15625 -1.322167 1.009667
## A:B:C:D:E A:B:C:D:E  0.53125 -0.634667 1.697167
```

```
# USING STUDENTIZED MAX MODULUS
smm_user <- function(colnum, rownum, m){
  r1 = weighted.mean(m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r2 = 1 / weighted.mean(1 / m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r3 = weighted.mean(m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r4 = 1 / weighted.mean(1 / m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  rl = mean(r1,r2)
  rr = mean(r3,r4)
  c1 = weighted.mean(c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  c2 = 1 / weighted.mean(1 / c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  return(round(mean(c1,c2),3))
}

#g=31 dof = 64-31-3-1 #
(multiplier2 = smm_user(colnum = c(24,29,30),
                        rownum = c(28,31,36),
                        m = matrix(c(3.156,3.251,3.104,3.196),2,2))) # g =31, v = 64-31-3-1
```

```
## [1] 3.147
```

```
boundary_pt2 =   multiplier2*sqrt(var_hat_/2)
avg_coeff_up2 = avg_coeff_+boundary_pt2
avg_coeff_dwn2 = avg_coeff_-boundary_pt2
CI_interval2 = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn2, Up
#################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval2
```

```
##           variable_ estimate       Lwr_       Up_
## A                 A -0.15625 -1.2989011 0.9864011
## B                 B -0.18750 -1.3301511 0.9551511
## C                 C  2.06250  0.9198489 3.2051511
## D                 D -0.21875 -1.3614011 0.9239011
## E                 E  0.09375 -1.0489011 1.2364011
## A:B             A:B -0.37500 -1.5176511 0.7676511
## A:C             A:C  0.12500 -1.0176511 1.2676511
## A:D             A:D  0.40625 -0.7364011 1.5489011
## A:E             A:E  0.28125 -0.8614011 1.4239011
## B:C             B:C  0.03125 -1.1114011 1.1739011
## B:D             B:D -0.12500 -1.2676511 1.0176511
## B:E             B:E  0.62500 -0.5176511 1.7676511
## C:D             C:D  0.25000 -0.8926511 1.3926511
## C:E             C:E -0.31250 -1.4551511 0.8301511
## D:E             D:E -0.28125 -1.4239011 0.8614011
## A:B:C         A:B:C  0.40625 -0.7364011 1.5489011
## A:B:D         A:B:D -0.06250 -1.2051511 1.0801511
## A:B:E         A:B:E  0.12500 -1.0176511 1.2676511
## A:C:D         A:C:D -0.06250 -1.2051511 1.0801511
```

```
## A:C:E         A:C:E  0.18750 -0.9551511 1.3301511
## A:D:E         A:D:E  0.03125 -1.1114011 1.1739011
## B:C:D         B:C:D -0.03125 -1.1739011 1.1114011
## B:C:E         B:C:E  0.21875 -0.9239011 1.3614011
## B:D:E         B:D:E  0.75000 -0.3926511 1.8926511
## C:D:E         C:D:E  0.06250 -1.0801511 1.2051511
## A:B:C:D     A:B:C:D -0.28125 -1.4239011 0.8614011
## A:B:C:E     A:B:C:E  0.15625 -0.9864011 1.2989011
## A:B:D:E     A:B:D:E -0.37500 -1.5176511 0.7676511
## A:C:D:E     A:C:D:E  0.06250 -1.0801511 1.2051511
## B:C:D:E     B:C:D:E -0.15625 -1.2989011 0.9864011
## A:B:C:D:E A:B:C:D:E  0.53125 -0.6114011 1.6739011
```

```
# calculating variance with anova
# Model with c only analysis
subset_dat_$block =files_1$Block_ABCDE
subset_dat_2$block =files_2$ABED
append_data_ = rbind(subset_dat_,subset_dat_2)
block_variable_ = c(rep(1,16),rep(2,16),rep(3,16),rep(4,16))
append_data_$final_block = block_variable_

A3 = append_data_$A
B3 = append_data_$B
C3 = append_data_$C
D3 = append_data_$D
E3 = append_data_$E
Y3 = append_data_$y
full_data = data.frame(y = Y3,A=A3,B=B3,C=C3,D=D3,E=E3)

m_fit_5 = lm(y~A*B*C*D*E+as.factor(block_variable_),data=full_data)
#summary(m_fit_3)
anova(m_fit_5)
```

```
## Analysis of Variance Table
##
## Response: y
##                             Df Sum Sq Mean Sq F value    Pr(>F)
## A                            1  0.391   0.391  0.1852   0.67014
## B                            1  0.563   0.563  0.2667   0.60949
## C                            1 68.063  68.063 32.2667 3.841e-06 ***
## D                            1  0.766   0.766  0.3630   0.55155
## E                            1  0.141   0.141  0.0667   0.79808
## as.factor(block_variable_)   3  8.328   2.776  1.3160   0.28813
## A:B                          1  2.250   2.250  1.0667   0.31024
## A:C                          1  0.250   0.250  0.1185   0.73313
## B:C                          1  0.016   0.016  0.0074   0.93201
## A:D                          1  2.641   2.641  1.2519   0.27238
## B:D                          1  0.250   0.250  0.1185   0.73313
## C:D                          1  1.000   1.000  0.4741   0.49659
## A:E                          1  1.266   1.266  0.6000   0.44485
## B:E                          1  6.250   6.250  2.9630   0.09584 .
## C:E                          1  1.563   1.563  0.7407   0.39648
## D:E                          1  1.266   1.266  0.6000   0.44485
## A:B:C                        1  2.641   2.641  1.2519   0.27238
## A:B:D                        1  0.062   0.062  0.0296   0.86453
```

```
## A:C:D                      1  0.062   0.062  0.0296   0.86453
## B:C:D                      1  0.016   0.016  0.0074   0.93201
## A:B:E                      1  0.250   0.250  0.1185   0.73313
## A:C:E                      1  0.563   0.563  0.2667   0.60949
## B:C:E                      1  0.766   0.766  0.3630   0.55155
## A:D:E                      1  0.016   0.016  0.0074   0.93201
## B:D:E                      1  9.000   9.000  4.2667   0.04791 *
## C:D:E                      1  0.063   0.063  0.0296   0.86453
## A:B:C:D                    1  1.266   1.266  0.6000   0.44485
## A:B:C:E                    1  0.391   0.391  0.1852   0.67014
## A:B:D:E                    1  2.820   2.820  1.3370   0.25699
## A:C:D:E                    1  0.062   0.062  0.0296   0.86453
## B:C:D:E                    1  0.391   0.391  0.1852   0.67014
## A:B:C:D:E                  1  0.195   0.195  0.0926   0.76308
## Residuals                 29 61.172   2.109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(sigma2 = anova(m_fit_5)[33,3])
```

```
## [1] 2.109375
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.3630922
```

```
print(paste0("standard deviation using ANOVA: ", sd_user))
```

```
## [1] "standard deviation using ANOVA: 0.363092188706945"
```

```
(effects_user = 2 * m_fit_5$coefficients)
```

```
##                  (Intercept)                            A
##                     12.28125                      -0.15625
##                            B                            C
##                     -0.18750                       2.06250
##                            D                            E
##                     -0.21875                       0.09375
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                      1.50000                      -0.31250
## as.factor(block_variable_)4                          A:B
##                      0.56250                      -0.37500
##                          A:C                          B:C
##                      0.12500                       0.03125
##                          A:D                          B:D
##                      0.40625                      -0.12500
##                          C:D                          A:E
##                      0.25000                       0.28125
##                          B:E                          C:E
##                      0.62500                      -0.31250
##                          D:E                        A:B:C
##                     -0.28125                       0.40625
##                        A:B:D                        A:C:D
##                     -0.06250                      -0.06250
##                        B:C:D                        A:B:E
##                     -0.03125                       0.12500
##                        A:C:E                        B:C:E
```

```
##                     0.18750                     0.21875
##                       A:D:E                       B:D:E
##                     0.03125                     0.75000
##                       C:D:E                       A:B:C:D
##                     0.06250                    -0.28125
##                     A:B:C:E                       A:B:D:E
##                     0.15625                    -0.59375
##                     A:C:D:E                       B:C:D:E
##                     0.06250                    -0.15625
##                   A:B:C:D:E
##                     0.15625
```

```
(multiplier1 = qt(0.1/2/31,df = 64-31-1-1,lower.tail = F))
```

```
## [1] 3.192655
```

```
#################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)

sim_ci = cbind(effects_user - multiplier2 * sd_user, effects_user + multiplier2 * sd_user)
colnames(sim_ci) <- c('Lower Bound','Upper Bound')
sim_ci
```

```
##                              Lower Bound Upper Bound
## (Intercept)                   11.1385989  13.4239011
## A                             -1.2989011   0.9864011
## B                             -1.3301511   0.9551511
## C                              0.9198489   3.2051511
## D                             -1.3614011   0.9239011
## E                             -1.0489011   1.2364011
## as.factor(block_variable_)2    0.3573489   2.6426511
## as.factor(block_variable_)3   -1.4551511   0.8301511
## as.factor(block_variable_)4   -0.5801511   1.7051511
## A:B                           -1.5176511   0.7676511
## A:C                           -1.0176511   1.2676511
## B:C                           -1.1114011   1.1739011
## A:D                           -0.7364011   1.5489011
## B:D                           -1.2676511   1.0176511
## C:D                           -0.8926511   1.3926511
## A:E                           -0.8614011   1.4239011
## B:E                           -0.5176511   1.7676511
## C:E                           -1.4551511   0.8301511
## D:E                           -1.4239011   0.8614011
## A:B:C                         -0.7364011   1.5489011
## A:B:D                         -1.2051511   1.0801511
## A:C:D                         -1.2051511   1.0801511
## B:C:D                         -1.1739011   1.1114011
## A:B:E                         -1.0176511   1.2676511
## A:C:E                         -0.9551511   1.3301511
## B:C:E                         -0.9239011   1.3614011
## A:D:E                         -1.1114011   1.1739011
## B:D:E                         -0.3926511   1.8926511
## C:D:E                         -1.0801511   1.2051511
## A:B:C:D                       -1.4239011   0.8614011
## A:B:C:E                       -0.9864011   1.2989011
```

```
## A:B:D:E                      -1.7364011   0.5489011
## A:C:D:E                      -1.0801511   1.2051511
## B:C:D:E                      -1.2989011   0.9864011
## A:B:C:D:E                    -0.9864011   1.2989011
```

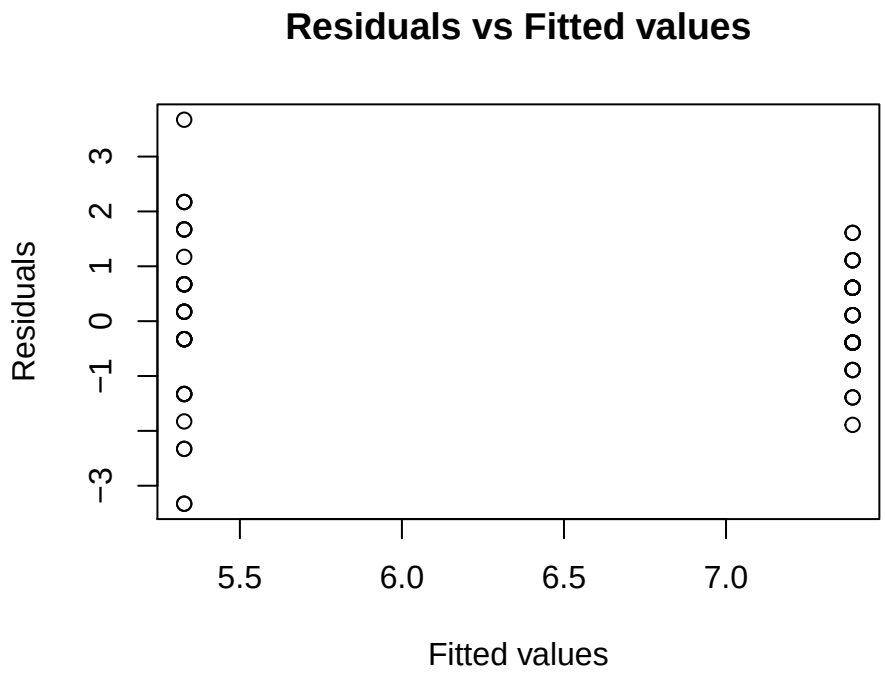
```
#####################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci2 = cbind(effects_user - multiplier1 * sd_user, effects_user + multiplier1 * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                  11.1220217  13.4404783
## A                            -1.3154783   1.0029783
## B                            -1.3467283   0.9717283
## C                             0.9032717   3.2217283
## D                            -1.3779783   0.9404783
## E                            -1.0654783   1.2529783
## as.factor(block_variable_)2   0.3407717   2.6592283
## as.factor(block_variable_)3  -1.4717283   0.8467283
## as.factor(block_variable_)4  -0.5967283   1.7217283
## A:B                          -1.5342283   0.7842283
## A:C                          -1.0342283   1.2842283
## B:C                          -1.1279783   1.1904783
## A:D                          -0.7529783   1.5654783
## B:D                          -1.2842283   1.0342283
## C:D                          -0.9092283   1.4092283
## A:E                          -0.8779783   1.4404783
## B:E                          -0.5342283   1.7842283
## C:E                          -1.4717283   0.8467283
## D:E                          -1.4404783   0.8779783
## A:B:C                        -0.7529783   1.5654783
## A:B:D                        -1.2217283   1.0967283
## A:C:D                        -1.2217283   1.0967283
## B:C:D                        -1.1904783   1.1279783
## A:B:E                        -1.0342283   1.2842283
## A:C:E                        -0.9717283   1.3467283
## B:C:E                        -0.9404783   1.3779783
## A:D:E                        -1.1279783   1.1904783
## B:D:E                        -0.4092283   1.9092283
## C:D:E                        -1.0967283   1.2217283
## A:B:C:D                      -1.4404783   0.8779783
## A:B:C:E                      -1.0029783   1.3154783
## A:B:D:E                      -1.7529783   0.5654783
## A:C:D:E                      -1.0967283   1.2217283
## B:C:D:E                      -1.3154783   1.0029783
## A:B:C:D:E                    -1.0029783   1.3154783
```

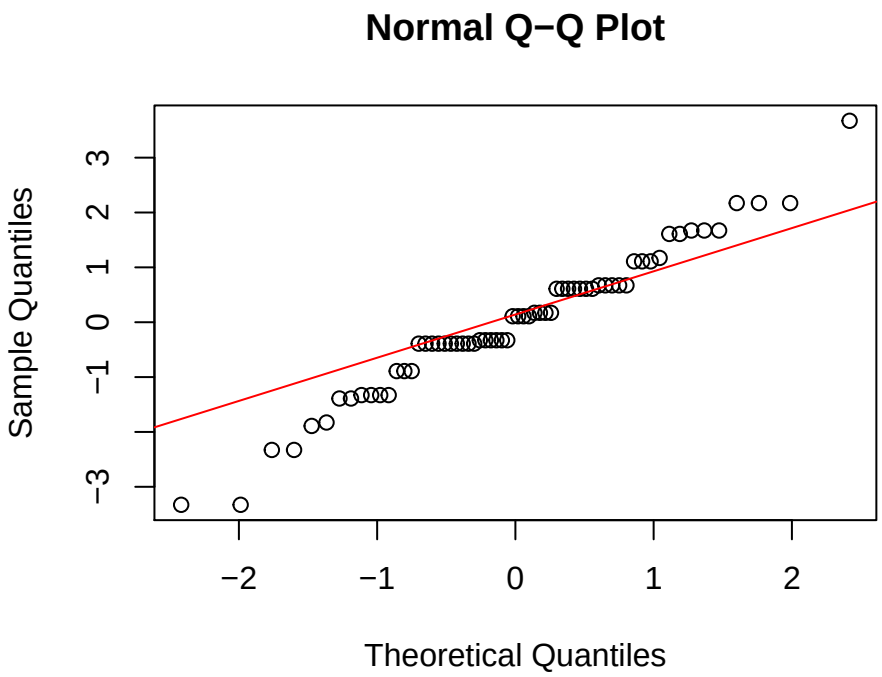
```
######################
```

SECTION A.3.2.3 :FITTING THE MODEL AND DIAGNOSTIC PLOTS

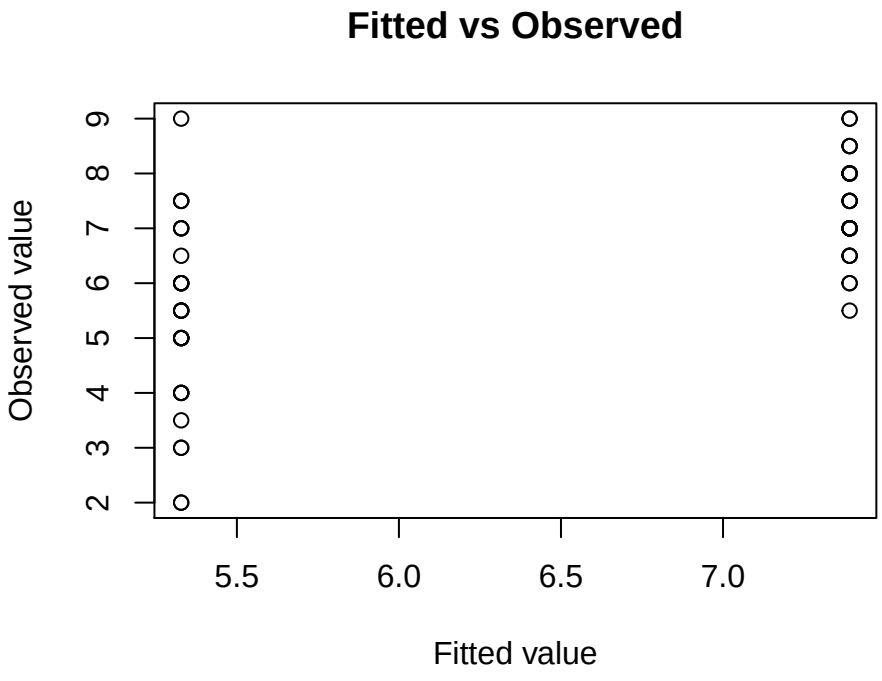
```
m_fit_6=lm(y~C,data=full_data)
summary_fullmodel_2 = summary(m_fit_6)

resid = m_fit_6$resid
fitted = m_fit_6$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```




```
plot.new()
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
group = as.factor(c(rep(c(1,2),8),rep(c(3,4),8)))
plot(cooks.distance(m_fit_6), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```

SECTION A.3.2.4 : STUDYING THE EFFECT OF TRANSFORMATIONS

```
#####################
A= full_data$A
B= full_data$B
C= full_data$C
D= full_data$D
E= full_data$E
y = full_data$y

y0=full_data$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+as.factor(block_variable_))
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
```

```
lambda
```

```
## [1] 1.798058
```

```
y_t=y0^lambda
model_trans2=lm(y_t~ A+B+C+D+E+as.factor(block_variable_))
anova(model_trans2)
```

```
## Analysis of Variance Table
##
## Response: y_t
##                             Df Sum Sq Mean Sq F value    Pr(>F)
## A                            1   14.6    14.6  0.1575    0.6930
## B                            1   63.3    63.3  0.6818    0.4125
## C                            1 3702.3  3702.3 39.8822 4.959e-08 ***
## D                            1   49.3    49.3  0.5315    0.4691
## E                            1    1.7     1.7  0.0181    0.8933
## as.factor(block_variable_)  3  400.3   133.4  1.4375    0.2418
## Residuals                   55 5105.7    92.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(round(p.adjust(summary(model_trans2)$coefficients[,4]),4))
```

```
##                 (Intercept)                           A
##                      0.0000                      1.0000
##                           B                           C
##                      1.0000                      0.0000
##                           D                           E
##                      1.0000                      1.0000
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                      0.3632                      1.0000
## as.factor(block_variable_)4
##                      1.0000
```

```
# g = 5, dof = 64-5-3-1
(multiplier_trans = smm_user(colnum = c(40,55,60),
                      rownum = c(3,5,6),
                      m = matrix(c(2.183,2.470,2.160,2.439),2,2))) # g = 5, v = 64-5-3-1
```

```
## [1] 2.353
```

```
(sigma2 = anova(model_trans2)[7,3])
```

```
## [1] 92.83051
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 2.408715
```

```
(effects_user = 2 * model_trans2$coefficients)
```

```
##                 (Intercept)                           A
##                  53.0695704                  -0.9560587
##                           B                           C
##                  -1.9888495                  15.2116004
##                           D                           E
##                  -1.7560344                   0.3244540
## as.factor(block_variable_)2 as.factor(block_variable_)3
```

```
##                   13.5386884                    3.5238738
## as.factor(block_variable_)4
##                    4.3161536
```

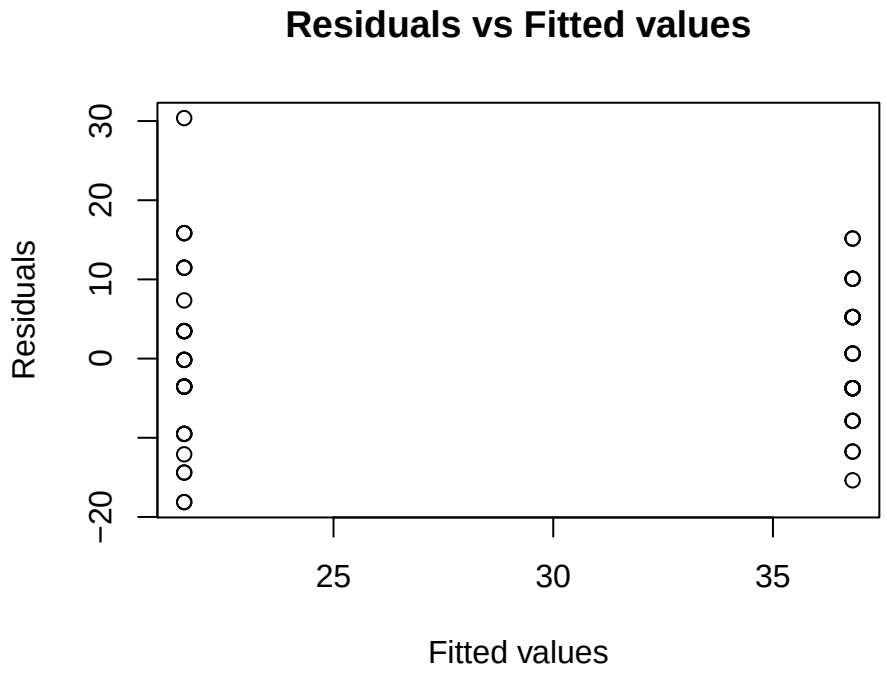
```
sim_ci2 = cbind(effects_user - multiplier_trans * sd_user, effects_user + multiplier_trans * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
############################################################
## STUDENTIZED MAXIMUM MODULUS CI FOR TRANSFORMED DATA
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                   47.401864   58.737276
## A                             -6.623765    4.711647
## B                             -7.656556    3.678856
## C                              9.543894   20.879306
## D                             -7.423740    3.911672
## E                             -5.343252    5.992160
## as.factor(block_variable_)2    7.870982   19.206394
## as.factor(block_variable_)3   -2.143832    9.191580
## as.factor(block_variable_)4   -1.351552    9.983860
```

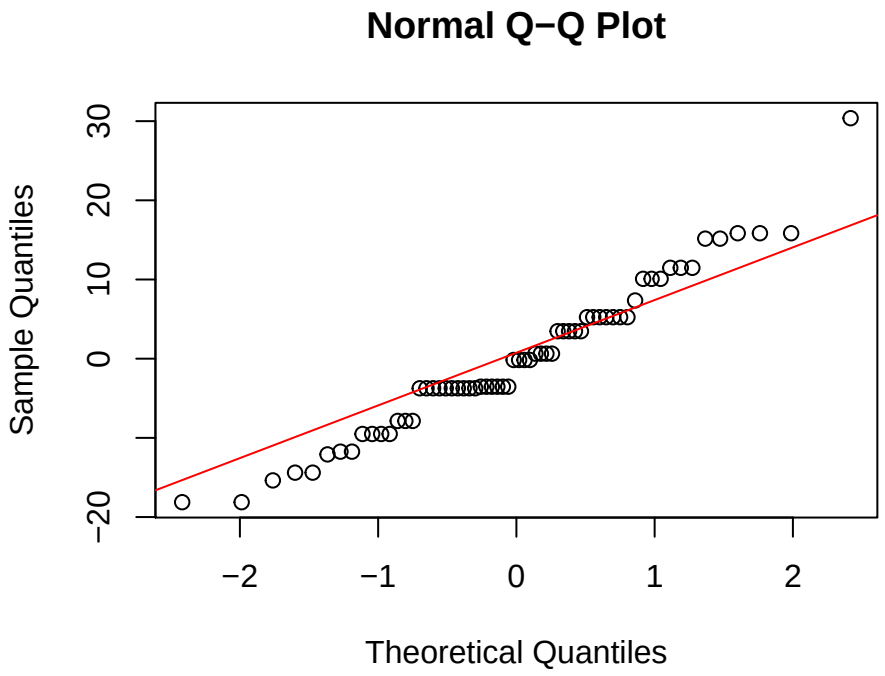
```
final_transformed_model = lm(y_t~C)
2 * final_transformed_model$coefficients
```

```
## (Intercept)           C
##    58.41425    15.21160
```

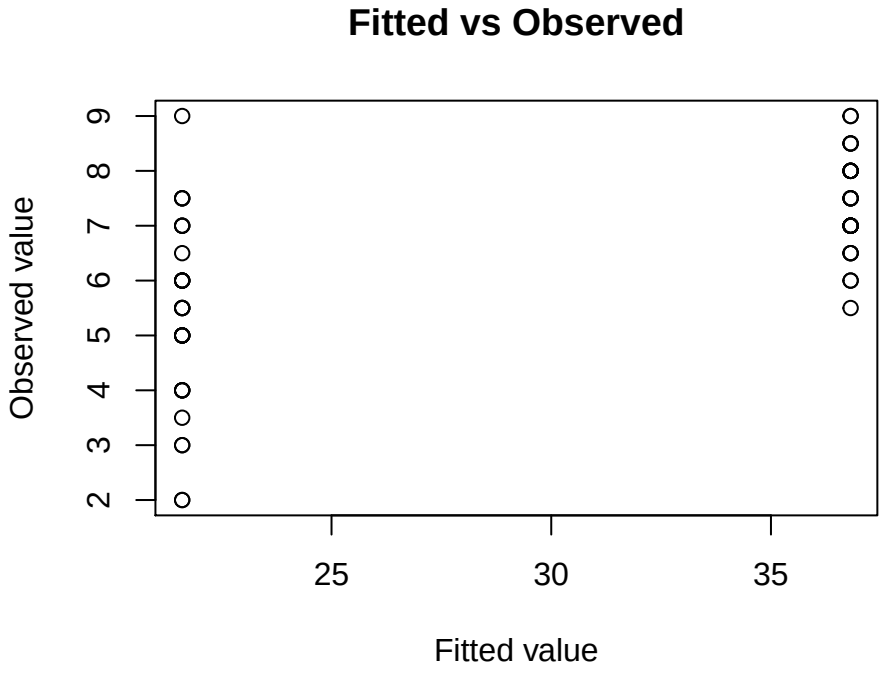
```
resid = final_transformed_model$resid
fitted = final_transformed_model$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



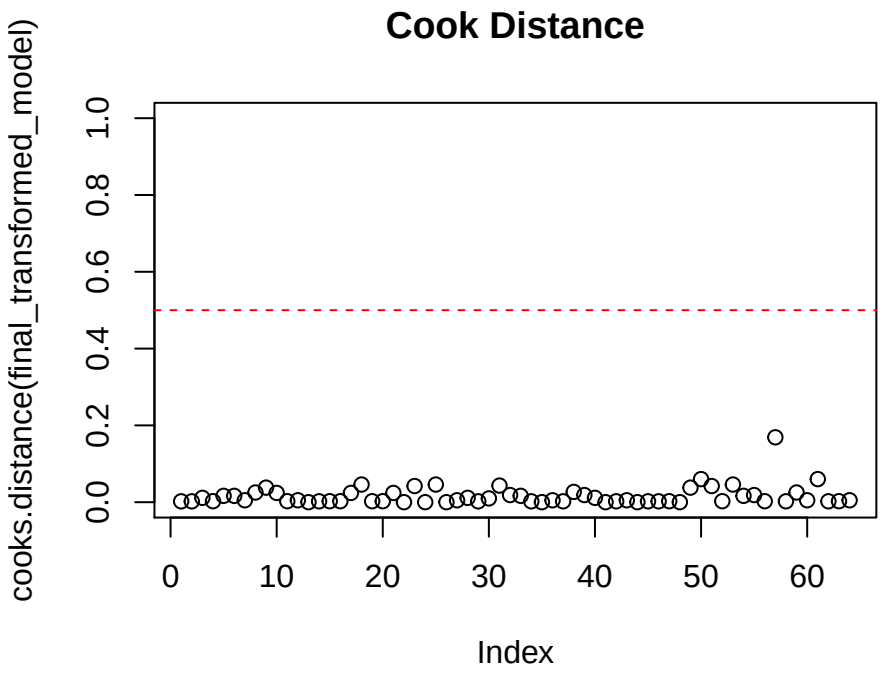
```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```



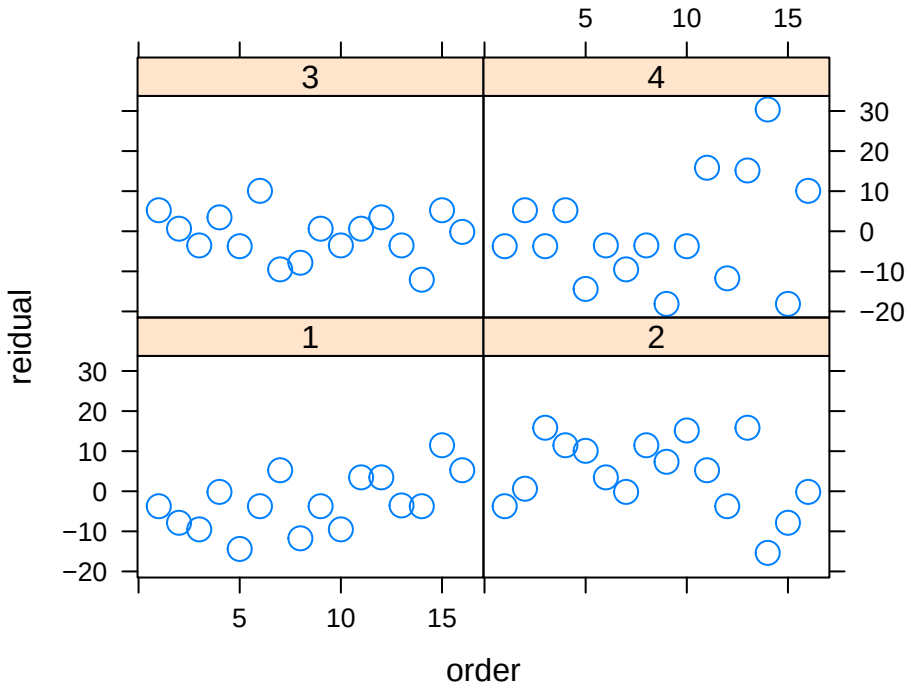
```
plot(cooks.distance(final_transformed_model), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



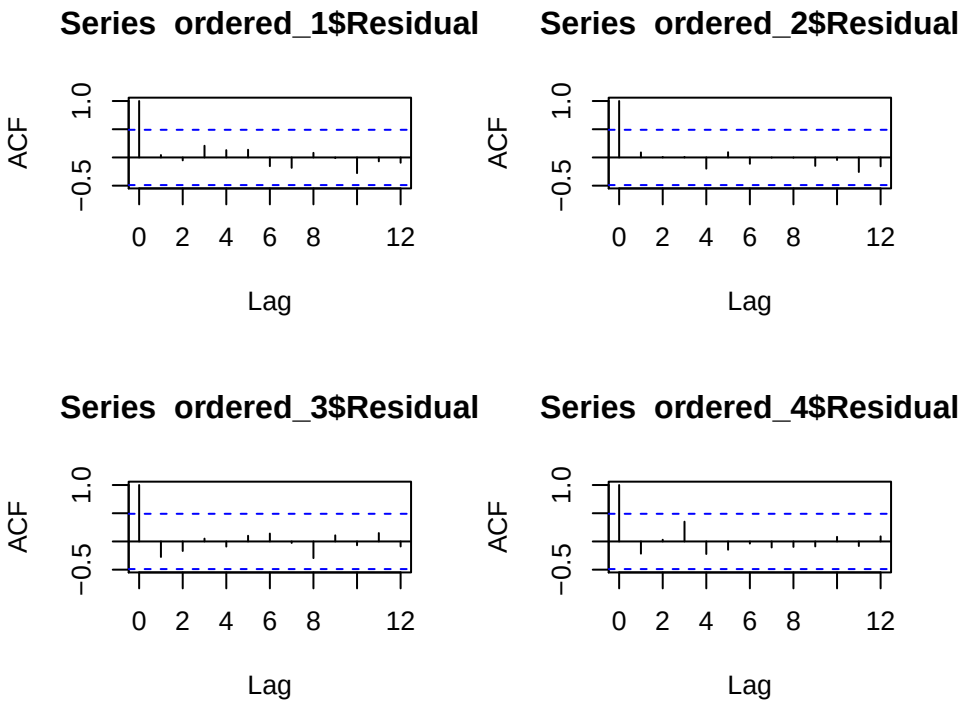

```
################ PLOT TO SEE IF THERE IS TREND IN RESIDUALS ################
```

SECTION A.3.2.5 : STUDYING IF THERE IS ANY TREND

```
order_= c(files_1$O_AD, files_2$O_AD)
ord_data_ = data.frame(order = order_, reidual =resid, days= block_variable_)


library(lattice)
xyplot(reidual ~ order | as.factor(block_variable_), data = ord_data_,
       auto.key = list(corner = c(0, .98)), cex = 1.5)
```




```
ordered1 = ord_data_[ord_data_$days==1,]
colnames(ordered1) <- c("Ordering", "Residual","Days")
ordered_1 = ordered1[order(ordered1$Ordering),]

ordered2 = ord_data_[ord_data_$days==2,]
colnames(ordered2) <- c("Ordering", "Residual","Days")
ordered_2 = ordered2[order(ordered2$Ordering),]

ordered3 = ord_data_[ord_data_$days==3,]
colnames(ordered3) <- c("Ordering", "Residual","Days")
ordered_3 = ordered3[order(ordered3$Ordering),]

ordered4 = ord_data_[ord_data_$days==4,]
colnames(ordered4) <- c("Ordering", "Residual","Days")
ordered_4 = ordered4[order(ordered4$Ordering),]

par(mfrow=c(2,2))
acf(ordered_1$Residual)
acf(ordered_2$Residual)
acf(ordered_3$Residual)
acf(ordered_4$Residual)
```


Series ordered_1$Residual
Series ordered_2$Residual
Series ordered_3$Residual
Series ordered_4$Residual
ACF
1.0
−0.5
0 2 4 6 8 12
Lag

ANALYSIS_JUDGE_AG

Abhijeet Bhardwaj

4/27/2021

SECTION A.4: ANALYSIS OF SCORES BY JUDGE AG

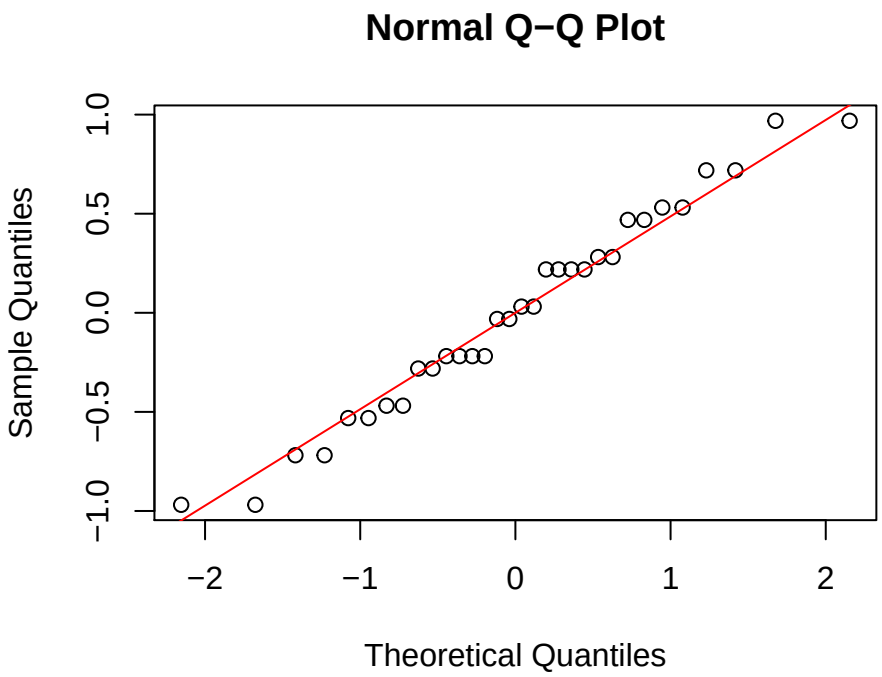
SECTION A.4.1 :Analysis of first run

SECTION A.4.1.1 DATA VISUALIZATION

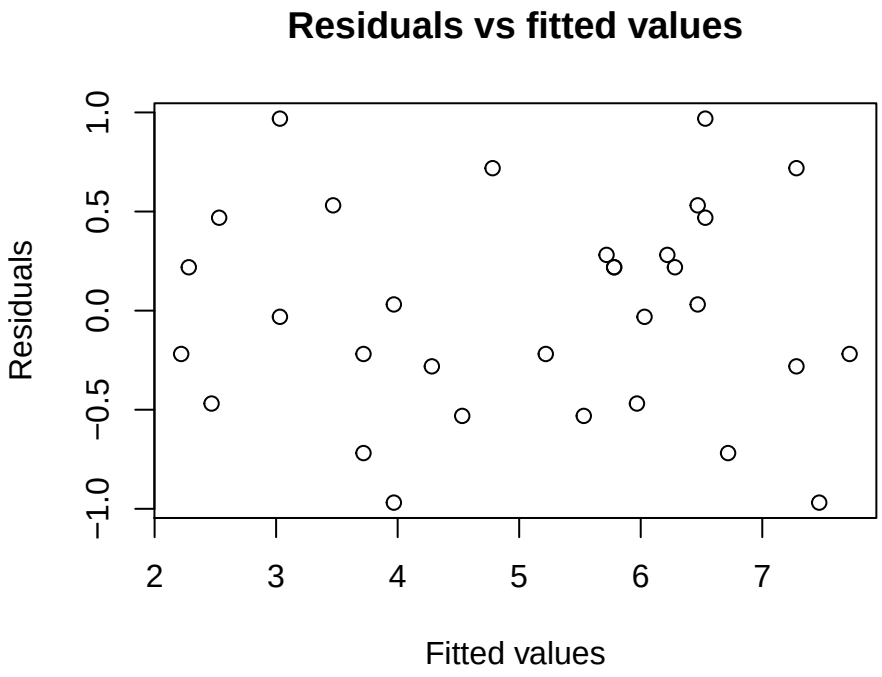
```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
#files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3
A = files_1$A
B = files_1$B
C = files_1$C
D = files_1$D
E = files_1$E
block = files_1$Block_ABCDE
avg_score = files_1$J_AG
order_scoring = files_1$O_AG
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^3+block,data=subset_dat_)
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=6) to new 7 after EncodeVars() -- should no longer happen!
```

```
resid = m_fit_1$resid
fitted = m_fit_1$fit
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs fitted values")
```



SECTION A.4.1.2 ANALYZING SIGNIFICANCE USING CONFIDENCE INTERVALS

```
# ASSUMING 4 FI's TO BE ZERO but not ABCDE as it is a block effect for sigma
#fitting complete model to get estimates of 4fis
m_fit_2 = lm(y~.^5,data=subset_dat_)
summary_fullmodel_ = summary(m_fit_2)
2*summary_fullmodel_$coefficients
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.018750e+01        NaN     NaN      NaN
## A            1.250000e-01        NaN     NaN      NaN
## B           -6.250000e-02        NaN     NaN      NaN
## C            2.250000e+00        NaN     NaN      NaN
## D            1.250000e-01        NaN     NaN      NaN
## E            1.000000e+00        NaN     NaN      NaN
## A:B         -3.750000e-01        NaN     NaN      NaN
## A:C          6.875000e-01        NaN     NaN      NaN
## A:D          1.875000e-01        NaN     NaN      NaN
## A:E          3.125000e-01        NaN     NaN      NaN
## B:C          3.750000e-01        NaN     NaN      NaN
## B:D          2.500000e-01        NaN     NaN      NaN
## B:E         -5.000000e-01        NaN     NaN      NaN
## C:D          4.375000e-01        NaN     NaN      NaN
## C:E          6.250000e-02        NaN     NaN      NaN
## D:E         -6.250000e-02        NaN     NaN      NaN
## A:B:C       -6.875000e-01        NaN     NaN      NaN
## A:B:D        3.125000e-01        NaN     NaN      NaN
## A:B:E        3.125000e-01        NaN     NaN      NaN
## A:C:D       -5.000000e-01        NaN     NaN      NaN
## A:C:E       -3.750000e-01        NaN     NaN      NaN
## A:D:E        3.750000e-01        NaN     NaN      NaN
## B:C:D       -3.125000e-01        NaN     NaN      NaN
## B:C:E        4.375000e-01        NaN     NaN      NaN
## B:D:E        1.062500e+00        NaN     NaN      NaN
## C:D:E       -2.500000e-01        NaN     NaN      NaN
## A:B:C:D     -2.207943e-16        NaN     NaN      NaN
## A:B:C:E      2.500000e-01        NaN     NaN      NaN
## A:B:D:E     -7.500000e-01        NaN     NaN      NaN
## A:C:D:E      4.375000e-01        NaN     NaN      NaN
## B:C:D:E     -5.000000e-01        NaN     NaN      NaN
## A:B:C:D:E   -8.125000e-01        NaN     NaN      NaN
```

```
coeffs_4fis = 2*summary_fullmodel_$coefficients[27:31,1]
sd_4 = sqrt(sum(coeffs_4fis^2)/5)
threshold_4 = sd_4*qt(0.1/(2*25),5,lower.tail = FALSE)
all_coeffs_ = 2*summary_fullmodel_$coefficients[,1]
eff = as.numeric(all_coeffs_)
which(abs(eff)>threshold_4) # only intercept
```

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

```
# ASSUMING 3 and 4 FI's TO BE ZERO but not ABCDE as it is a block effect for estimating sigma
#fitting complete model to get estimates of 4fis
coeffs_34fis = 2*summary_fullmodel_$coefficients[17:31,1]
```

```r
var_34 = sum(coeffs_34fis^2)/15
sd_34 = sqrt(var_34)
threshold_34 = sd_4*qt(0.1/(2*15),15,lower.tail = FALSE)
threshold_34
```

```
## [1] 1.452693
```

```r
which(abs(eff)>threshold_34) # INTERCEPT AND C
```

```
## [1] 1 4
```

SECTION A.4.1.3 ANALYZING SIGNIFICANCE USING DANIEL METHOD

```r
## APPLY DANIEL METHOD
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

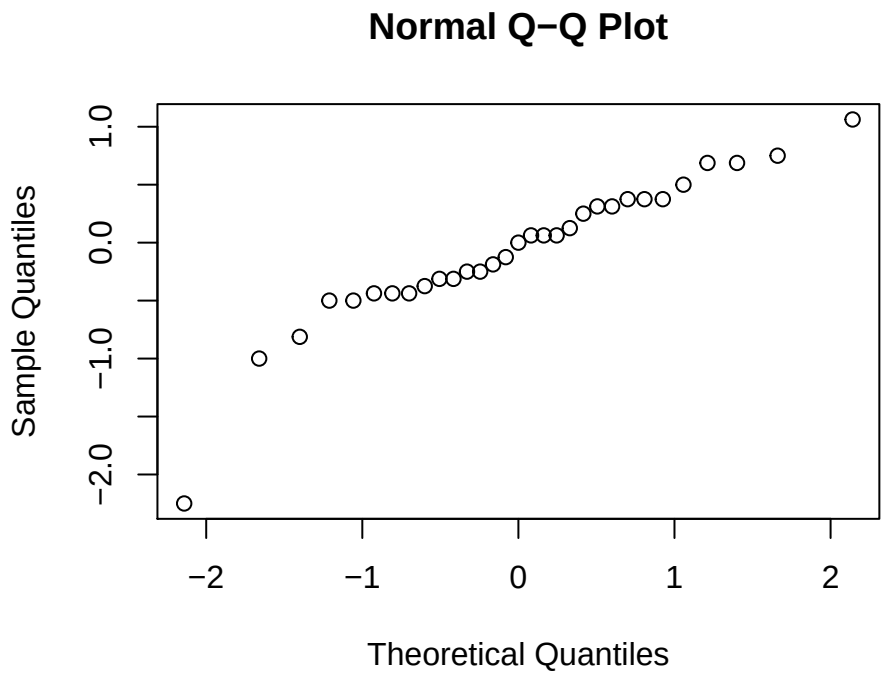
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^5,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 3
```

```
## [1] 3
```

```r
A_final = perm[1,3] * A
B_final = perm[2,3] * B
C_final = perm[3,3] * C
D_final = perm[4,3] * D
E_final = perm[5,3] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^5,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -0.08094729  0.16242937 -2.14119812  0.32929135 -1.66069761 -0.60017878
##  [7]  1.21123213 -0.16242937  0.50593365  0.70009021 -0.24500622 -1.21123213
## [13] -0.80754104  0.08094729  0.24500622  1.40074506  0.60017878 -0.50593365
## [19] -1.05741423  0.80754104  0.92524456 -0.41598722 -0.92524456  2.14119812
## [25] -0.32929135  0.00000000  0.41598722  1.66069761 -0.70009021  1.05741423
## [31] -1.40074506
##
## $y
##                              A_final                              B_final
##                        -1.250000e-01                         6.250000e-02
##                              C_final                              D_final
##                        -2.250000e+00                         1.250000e-01
##                              E_final                        A_final:B_final
##                        -1.000000e+00                        -3.750000e-01
##                       A_final:C_final                       A_final:D_final
##                         6.875000e-01                        -1.875000e-01
##                       A_final:E_final                       B_final:C_final
##                         3.125000e-01                         3.750000e-01
##                       B_final:D_final                       B_final:E_final
```

```
##                           -2.500000e-01                           -5.000000e-01
##                         C_final:D_final                         C_final:E_final
##                           -4.375000e-01                            6.250000e-02
##                         D_final:E_final                  A_final:B_final:C_final
##                            6.250000e-02                            6.875000e-01
##                  A_final:B_final:D_final                  A_final:B_final:E_final
##                            3.125000e-01                           -3.125000e-01
##                  A_final:C_final:D_final                  A_final:C_final:E_final
##                           -5.000000e-01                            3.750000e-01
##                  A_final:D_final:E_final                  B_final:C_final:D_final
##                            3.750000e-01                           -3.125000e-01
##                  B_final:C_final:E_final                  B_final:D_final:E_final
##                           -4.375000e-01                            1.062500e+00
##                  C_final:D_final:E_final         A_final:B_final:C_final:D_final
##                           -2.500000e-01                            2.207943e-16
##          A_final:B_final:C_final:E_final         A_final:B_final:D_final:E_final
##                            2.500000e-01                            7.500000e-01
##          A_final:C_final:D_final:E_final         B_final:C_final:D_final:E_final
##                           -4.375000e-01                            5.000000e-01
## A_final:B_final:C_final:D_final:E_final
##                           -8.125000e-01
```

```
# Step2: Fit a lm ------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier ------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # outliers::C_final
```

```
## C_final
##       3
```

```
x_inlier = qq_x[which(abs(qq_y) <= CC)]
y_inlier = qq_y[which(abs(qq_y) <= CC)]
# Step4: Fit another lm ----------------------------------------------------
fit2 = lm(y_inlier ~ x_inlier)
beta2 = fit2$coefficients[2]
beta1 / beta2 # 1.186
```

```
##     qq_x
## 1.153422
```

```
# From the table in the paper,
# n = 31, alpha = 0.05 -> threshold = 1.108
# So we have the significance here

m = length(x_inlier)
y_mean = mean(y_inlier)
x_mean = mean(x_inlier)
n_prime = round(31/4,1)
sigma_user = sqrt(sum((fit2$residuals)^2)/fit2$df.residual)
```

```
LHS = abs(y_inlier - y_mean + beta2 * (x_inlier - x_mean))
RHS = n_prime * sqrt(qf(p = 0.975,df1 = n_prime,df2 = m-2)) * sigma_user *
  sqrt(1 + 1/m + (x_inlier - x_mean)^2 / (var(x_inlier) * (m+1) ))
final_idx = which(LHS <= RHS)
D_ = max(abs(y_inlier[final_idx]))
c(D_, CC)
```

```
##          75%
## 0.4375 1.5000
```

```
final_threshold = max(CC,D_)
print(paste0("after daniels method :", names(which(abs(qq_y) > final_threshold)))) # again only C_final
```

```
## [1] "after daniels method :C_final"
```

```
##############################################################################
```

SECTION A.4.1.4 ANALYZING SIGNIFICANCE USING DONG's METHOD

```
#ANALYSIS BY DONGS METHOD
coeff=coef(m_fit_2)*2
theta1=abs(na.omit(coeff[-c(1,31)]))
s0=1.5*median(theta1)
m1=sum(theta1<=2.5*s0)
s1=sqrt(sum(theta1[theta1<=2.5*s0]^2)/m1)
m2=sum(theta1<=2.5*s1)
s2=sqrt(sum(theta1[theta1<=2.5*s1]^2)/m2)
g=length(theta1)
gamma=0.5*(1-(1-0.05)^(1/g))
print(paste0("after DONG's method significant effect :", names(theta1[theta1>s2*qt(gamma,m2,lower.tail =
```

```
## [1] "after DONG's method significant effect :C"
```

```
##############################################
```

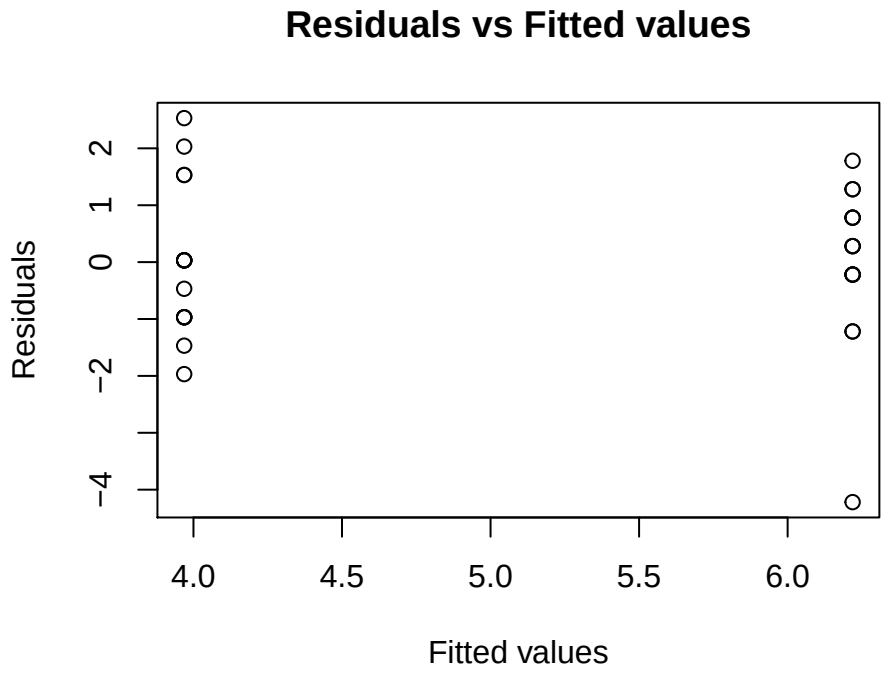
SECTION A.4.1.5 MODEL FITTING WITH SIGNIFICANT FACTOR FOR UN REPLICATED EXPERIMENT

```
# Model with c only analysis
m_fit_3 = lm(y~C,data=subset_dat_)
summary(m_fit_3)
```

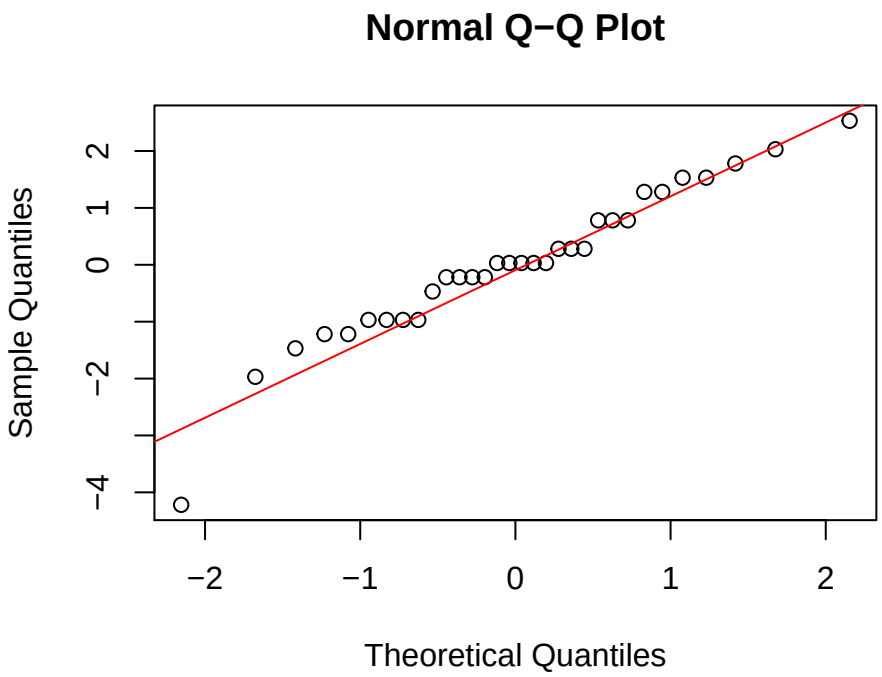
```
##
## Call:
## lm(formula = y ~ C, data = subset_dat_)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2188 -0.9687  0.0313  0.7812  2.5313
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   5.0937     0.2387  21.336  < 2e-16 ***
## C             1.1250     0.2387   4.712 5.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.351 on 30 degrees of freedom
## Multiple R-squared:  0.4253, Adjusted R-squared:  0.4062
## F-statistic:  22.2 on 1 and 30 DF,  p-value: 5.246e-05
```

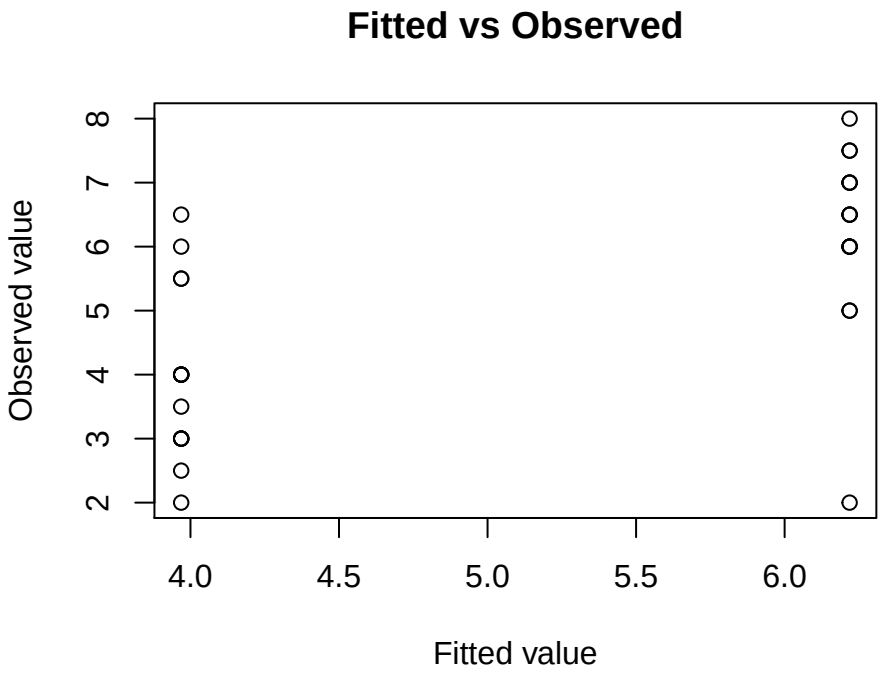
```
resid = m_fit_3$resid
fitted = m_fit_3$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



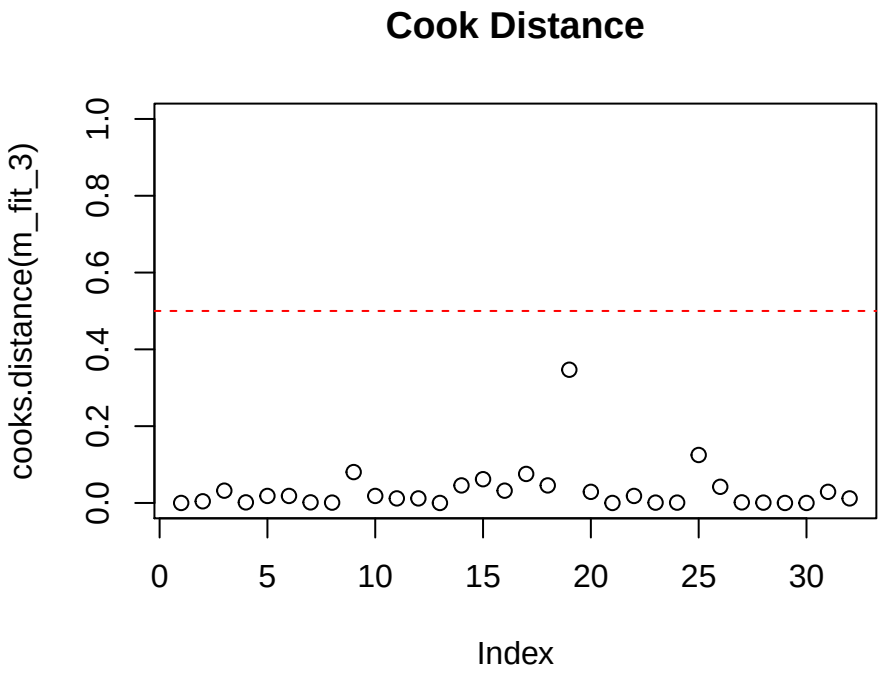

```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
# checking for outliers in the final model
plot(cooks.distance(m_fit_3), ylim =c(0,1),main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```




```
####################
```

SECTION A.4.1.6 APPLYING TRANSFORMATION TO SEE IF MORE SIGNIFICANT FACTORS CAN BE EXTRACTED

```
y0=subset_dat_$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+block)
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda
```

```
## [1] 1.589455
```

```
y_t=y0^lambda
#data_subs_bc_t=data.frame(y_t= y_t,A=A,B=B,C=C,D=D,E=E)
#model_trans=lm(y_t~ .^2,data_subs_bc_t)
```

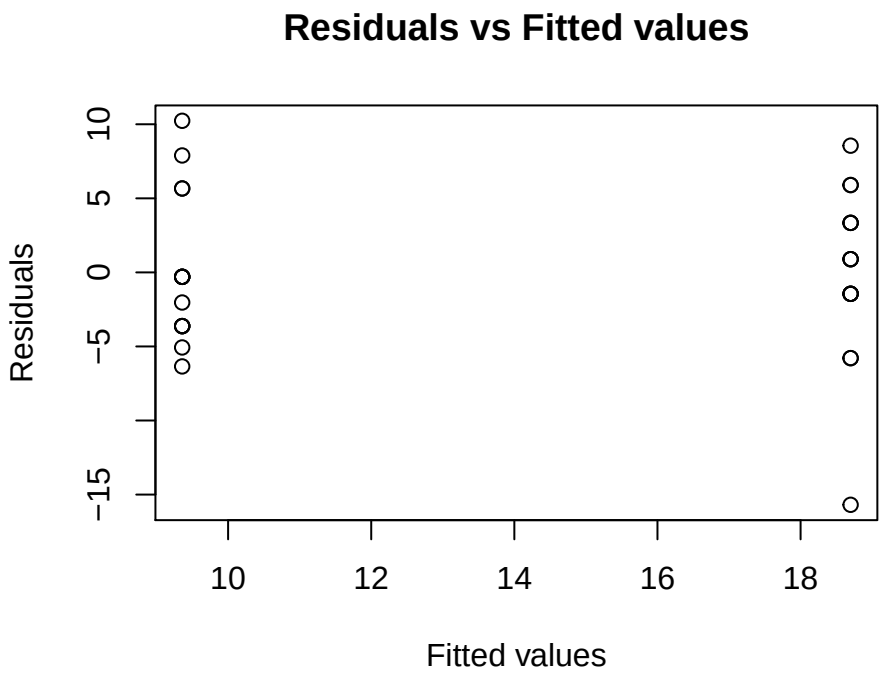
```
model_trans=lm(y_t~ A+B+C+D+E+block)
summary(model_trans)
```

```
##
## Call:
## lm(formula = y_t ~ A + B + C + D + E + block)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6779  -2.2619  -0.1314   2.6541  10.2050
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  14.0286     0.8827  15.892 1.41e-14 ***
## A             0.3690     0.8827   0.418   0.6795
## B            -0.2251     0.8827  -0.255   0.8008
## C             4.6707     0.8827   5.291 1.76e-05 ***
## D             0.1337     0.8827   0.151   0.8808
## E             2.0208     0.8827   2.289   0.0308 *
## block        -1.7137     0.8827  -1.941   0.0636 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.994 on 25 degrees of freedom
## Multiple R-squared:  0.5985, Adjusted R-squared:  0.5021
## F-statistic: 6.211 on 6 and 25 DF,  p-value: 0.0004303
```

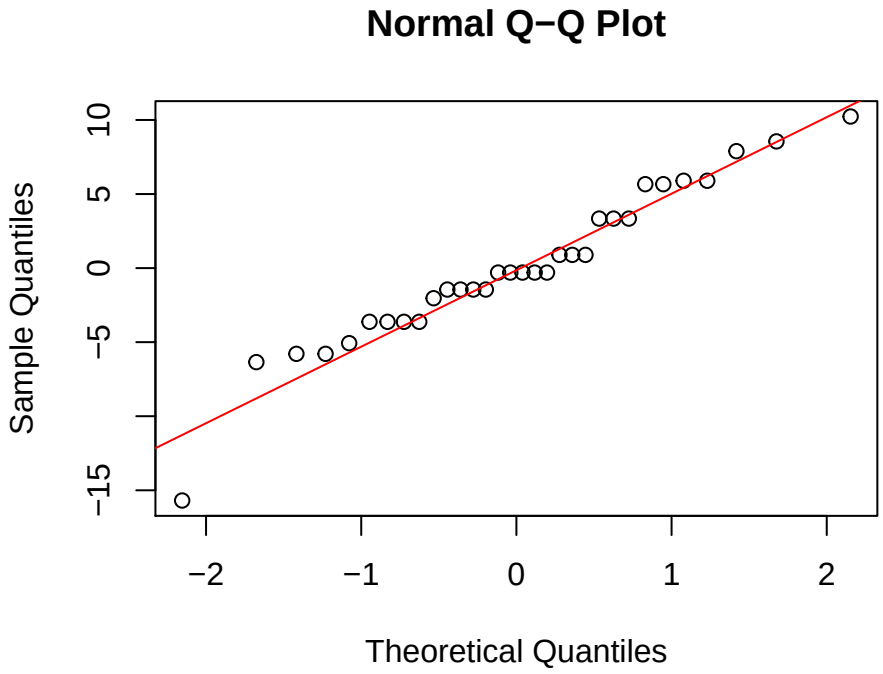
```
print(round(p.adjust(summary(model_trans)$coefficients[,4]),4))
```

```
## (Intercept)           A           B           C           D           E
##      0.0000      1.0000      1.0000      0.0001      1.0000      0.1539
##       block
##      0.2543
```

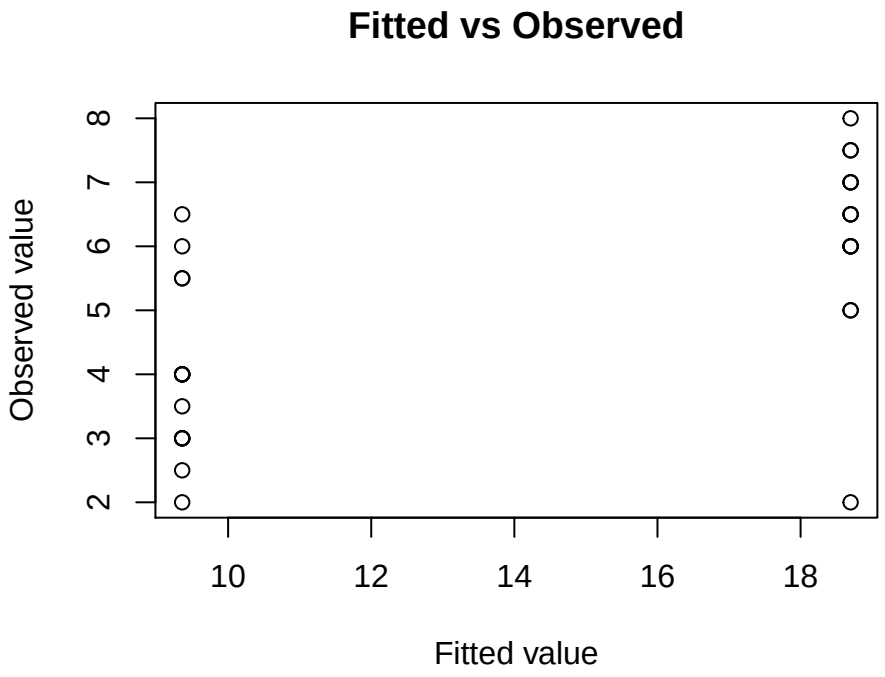
```
model_trans=lm(y_t~C) #AS ONLY C IS SIGNIFICANT
resid = model_trans$resid
fitted = model_trans$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



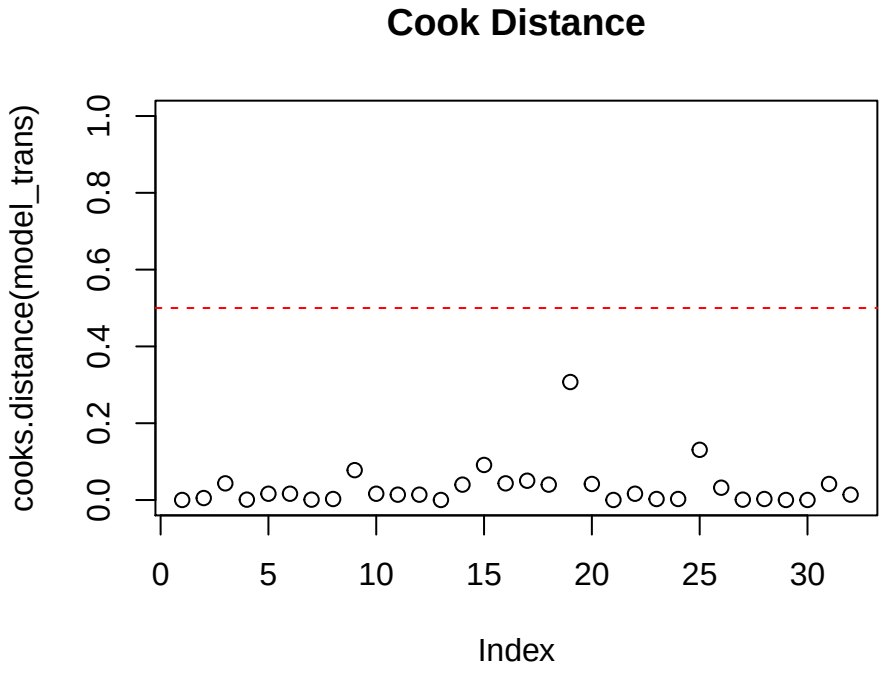

```
qqnorm(resid)
qqline(resid,col="red")
```



```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
plot(cooks.distance(model_trans), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.4.2

:Analysis of second run # SECTION A.4.2.1 :Estimates from second run

```
# COMBINING THE SECOND COMPLETE REPLICATE

files_2 = read.csv('Run_2_with_scores.csv')
#files_2$J_AVG = (files_2$J_AD+files_2$J_AG+files_2$J_KJ)/3
A2 = files_2$A
B2 = files_2$B
C2 = files_2$C
D2 = files_2$D
E2 = files_2$E
avg_score2 = files_2$J_AG
subset_dat_2 = data.frame(y = avg_score2,A=A2,B=B2,C=C2,D=D2,E=E2)
m_fit_4 = lm(y~.^5,data=subset_dat_2)
summary_fullmodel_2 = summary(m_fit_4)
2*summary_fullmodel_2$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.09375        NaN     NaN      NaN
## A           -0.34375        NaN     NaN      NaN
## B           -0.21875        NaN     NaN      NaN
## C            3.03125        NaN     NaN      NaN
## D            0.34375        NaN     NaN      NaN
## E            0.53125        NaN     NaN      NaN
## A:B         -0.03125        NaN     NaN      NaN
## A:C          0.21875        NaN     NaN      NaN
## A:D          0.53125        NaN     NaN      NaN
## A:E         -0.40625        NaN     NaN      NaN
## B:C          0.21875        NaN     NaN      NaN
## B:D         -0.46875        NaN     NaN      NaN
## B:E          0.09375        NaN     NaN      NaN
## C:D         -0.96875        NaN     NaN      NaN
## C:E          0.71875        NaN     NaN      NaN
## D:E         -0.46875        NaN     NaN      NaN
## A:B:C        0.28125        NaN     NaN      NaN
## A:B:D        0.09375        NaN     NaN      NaN
## A:B:E        0.53125        NaN     NaN      NaN
## A:C:D        0.09375        NaN     NaN      NaN
## A:C:E        0.40625        NaN     NaN      NaN
## A:D:E       -0.53125        NaN     NaN      NaN
## B:C:D        0.21875        NaN     NaN      NaN
## B:C:E        0.53125        NaN     NaN      NaN
## B:D:E       -0.15625        NaN     NaN      NaN
## C:D:E        0.21875        NaN     NaN      NaN
## A:B:C:D      0.40625        NaN     NaN      NaN
## A:B:C:E     -0.65625        NaN     NaN      NaN
## A:B:D:E     -1.09375        NaN     NaN      NaN
## A:C:D:E      0.53125        NaN     NaN      NaN
## B:C:D:E      0.28125        NaN     NaN      NaN
## A:B:C:D:E    0.96875        NaN     NaN      NaN
```

SECTION A.4.2.2 :Calculating Confidence Intervals by combining the replicates

```
# CACULATION OF VARIANCE BY FIRST PRINCIPLES
coeffs_1st_rep = 2*summary_fullmodel_$coefficients[-1,1]
coeffs_2st_rep = 2*summary_fullmodel_2$coefficients[-1,1]
avg_coeff_ = (coeffs_1st_rep+coeffs_2st_rep)/2
diff_1 = (coeffs_1st_rep-avg_coeff_)^2
diff_2 = (coeffs_2st_rep-avg_coeff_)^2

diff_1_ = diff_1[-c(31,28)]
diff_2_ = diff_2[-c(31,28)]

var_ = (diff_1_+diff_2_)/(2-1)
var_hat_ = sum(var_)/29 #estimate of 4sigma^2/2^k) \hat{var(Ai)}
multiplier1 = qt(0.1/2/31,df = 64-31-3-1,lower.tail = F)
boundary_pt =   multiplier1*sqrt(var_hat_/2)
print(paste0("standard deviation using first principle: ", sqrt(var_hat_/2)))
```

```
## [1] "standard deviation using first principle: 0.321385005343253"
```

```
avg_coeff_up = avg_coeff_+boundary_pt
avg_coeff_dwn = avg_coeff_-boundary_pt
CI_interval = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn, Up_=
################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval
```

```
##           variable_  estimate      Lwr_      Up_
## A                 A -0.109375 -1.141367 0.922617
## B                 B -0.140625 -1.172617 0.891367
## C                 C  2.640625  1.608633 3.672617
## D                 D  0.234375 -0.797617 1.266367
## E                 E  0.765625 -0.266367 1.797617
## A:B             A:B -0.203125 -1.235117 0.828867
## A:C             A:C  0.453125 -0.578867 1.485117
## A:D             A:D  0.359375 -0.672617 1.391367
## A:E             A:E -0.046875 -1.078867 0.985117
## B:C             B:C  0.296875 -0.735117 1.328867
## B:D             B:D -0.109375 -1.141367 0.922617
## B:E             B:E -0.203125 -1.235117 0.828867
## C:D             C:D -0.265625 -1.297617 0.766367
## C:E             C:E  0.390625 -0.641367 1.422617
## D:E             D:E -0.265625 -1.297617 0.766367
## A:B:C         A:B:C -0.203125 -1.235117 0.828867
## A:B:D         A:B:D  0.203125 -0.828867 1.235117
## A:B:E         A:B:E  0.421875 -0.610117 1.453867
## A:C:D         A:C:D -0.203125 -1.235117 0.828867
## A:C:E         A:C:E  0.015625 -1.016367 1.047617
## A:D:E         A:D:E -0.078125 -1.110117 0.953867
## B:C:D         B:C:D -0.046875 -1.078867 0.985117
## B:C:E         B:C:E  0.484375 -0.547617 1.516367
## B:D:E         B:D:E  0.453125 -0.578867 1.485117
## C:D:E         C:D:E -0.015625 -1.047617 1.016367
```

```
## A:B:C:D     A:B:C:D  0.203125 -0.828867 1.235117
## A:B:C:E     A:B:C:E -0.203125 -1.235117 0.828867
## A:B:D:E     A:B:D:E -0.921875 -1.953867 0.110117
## A:C:D:E     A:C:D:E  0.484375 -0.547617 1.516367
## B:C:D:E     B:C:D:E -0.109375 -1.141367 0.922617
## A:B:C:D:E A:B:C:D:E  0.078125 -0.953867 1.110117
```

```
# USING STUDENTIZED MAX MODULUS
smm_user <- function(colnum, rownum, m){
  r1 = weighted.mean(m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r2 = 1 / weighted.mean(1 / m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r3 = weighted.mean(m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r4 = 1 / weighted.mean(1 / m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  rl = mean(r1,r2)
  rr = mean(r3,r4)
  c1 = weighted.mean(c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  c2 = 1 / weighted.mean(1 / c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  return(round(mean(c1,c2),3))
}

#g=31 dof = 64-31-3-1 #
(multiplier2 = smm_user(colnum = c(24,29,30),
                        rownum = c(28,31,36),
                        m = matrix(c(3.156,3.251,3.104,3.196),2,2))) # g =31, v = 64-31-3-1
```

```
## [1] 3.147
```

```
boundary_pt2 =   multiplier2*sqrt(var_hat_/2)
avg_coeff_up2 = avg_coeff_+boundary_pt2
avg_coeff_dwn2 = avg_coeff_-boundary_pt2
CI_interval2 = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn2, Up
#################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval2
```

```
##           variable_  estimate       Lwr_        Up_
## A                 A -0.109375 -1.1207736 0.90202361
## B                 B -0.140625 -1.1520236 0.87077361
## C                 C  2.640625  1.6292264 3.65202361
## D                 D  0.234375 -0.7770236 1.24577361
## E                 E  0.765625 -0.2457736 1.77702361
## A:B             A:B -0.203125 -1.2145236 0.80827361
## A:C             A:C  0.453125 -0.5582736 1.46452361
## A:D             A:D  0.359375 -0.6520236 1.37077361
## A:E             A:E -0.046875 -1.0582736 0.96452361
## B:C             B:C  0.296875 -0.7145236 1.30827361
## B:D             B:D -0.109375 -1.1207736 0.90202361
## B:E             B:E -0.203125 -1.2145236 0.80827361
## C:D             C:D -0.265625 -1.2770236 0.74577361
## C:E             C:E  0.390625 -0.6207736 1.40202361
## D:E             D:E -0.265625 -1.2770236 0.74577361
## A:B:C         A:B:C -0.203125 -1.2145236 0.80827361
## A:B:D         A:B:D  0.203125 -0.8082736 1.21452361
## A:B:E         A:B:E  0.421875 -0.5895236 1.43327361
## A:C:D         A:C:D -0.203125 -1.2145236 0.80827361
```

```
## A:C:E         A:C:E  0.015625 -0.9957736 1.02702361
## A:D:E         A:D:E -0.078125 -1.0895236 0.93327361
## B:C:D         B:C:D -0.046875 -1.0582736 0.96452361
## B:C:E         B:C:E  0.484375 -0.5270236 1.49577361
## B:D:E         B:D:E  0.453125 -0.5582736 1.46452361
## C:D:E         C:D:E -0.015625 -1.0270236 0.99577361
## A:B:C:D     A:B:C:D  0.203125 -0.8082736 1.21452361
## A:B:C:E     A:B:C:E -0.203125 -1.2145236 0.80827361
## A:B:D:E     A:B:D:E -0.921875 -1.9332736 0.08952361
## A:C:D:E     A:C:D:E  0.484375 -0.5270236 1.49577361
## B:C:D:E     B:C:D:E -0.109375 -1.1207736 0.90202361
## A:B:C:D:E A:B:C:D:E  0.078125 -0.9332736 1.08952361
```

```
# calculating variance with anova
# Model with c only analysis
subset_dat_$block =files_1$Block_ABCDE
subset_dat_2$block =files_2$ABED
append_data_ = rbind(subset_dat_,subset_dat_2)
block_variable_ = c(rep(1,16),rep(2,16),rep(3,16),rep(4,16))
append_data_$final_block = block_variable_

A3 = append_data_$A
B3 = append_data_$B
C3 = append_data_$C
D3 = append_data_$D
E3 = append_data_$E
Y3 = append_data_$y
full_data = data.frame(y = Y3,A=A3,B=B3,C=C3,D=D3,E=E3)

m_fit_5 = lm(y~A*B*C*D*E+as.factor(block_variable_),data=full_data)
#summary(m_fit_3)
anova(m_fit_5)
```

```
## Analysis of Variance Table
##
## Response: y
##                             Df  Sum Sq Mean Sq F value    Pr(>F)
## A                            1   0.191   0.191  0.1158   0.73607
## B                            1   0.316   0.316  0.1915   0.66495
## C                            1 111.566 111.566 67.5091 4.648e-09 ***
## D                            1   0.879   0.879  0.5318   0.47169
## E                            1   9.379   9.379  5.6752   0.02398 *
## as.factor(block_variable_)   3  18.137   6.046  3.6582   0.02375 *
## A:B                          1   0.660   0.660  0.3995   0.53232
## A:C                          1   3.285   3.285  1.9879   0.16920
## B:C                          1   1.410   1.410  0.8533   0.36325
## A:D                          1   2.066   2.066  1.2504   0.27266
## B:D                          1   0.191   0.191  0.1158   0.73607
## C:D                          1   1.129   1.129  0.6831   0.41527
## A:E                          1   0.035   0.035  0.0213   0.88505
## B:E                          1   0.660   0.660  0.3995   0.53232
## C:E                          1   2.441   2.441  1.4773   0.23400
## D:E                          1   1.129   1.129  0.6831   0.41527
## A:B:C                        1   0.660   0.660  0.3995   0.53232
## A:B:D                        1   0.660   0.660  0.3995   0.53232
```

```
## A:C:D                      1   0.660   0.660  0.3995   0.53232
## B:C:D                      1   0.035   0.035  0.0213   0.88505
## A:B:E                      1   2.848   2.848  1.7231   0.19959
## A:C:E                      1   0.004   0.004  0.0024   0.96156
## B:C:E                      1   3.754   3.754  2.2715   0.14259
## A:D:E                      1   0.098   0.098  0.0591   0.80965
## B:D:E                      1   3.285   3.285  1.9879   0.16920
## C:D:E                      1   0.004   0.004  0.0024   0.96156
## A:B:C:D                    1   0.660   0.660  0.3995   0.53232
## A:B:C:E                    1   0.660   0.660  0.3995   0.53232
## A:B:D:E                    1   4.500   4.500  2.7230   0.10971
## A:C:D:E                    1   3.754   3.754  2.2715   0.14259
## B:C:D:E                    1   0.191   0.191  0.1158   0.73607
## A:B:C:D:E                  1   7.508   7.508  4.5430   0.04165 *
## Residuals                 29  47.926   1.653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(sigma2 = anova(m_fit_5)[33,3])
```

```
## [1] 1.652613
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.321385
```

```
print(paste0("standard deviation using ANOVA: ", sd_user))
```

```
## [1] "standard deviation using ANOVA: 0.321385005343253"
```

```
(effects_user = 2 * m_fit_5$coefficients)
```

```
##                  (Intercept)                            A
##                    11.968750                   -0.109375
##                            B                            C
##                    -0.140625                    2.640625
##                            D                            E
##                     0.234375                    0.765625
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                    -3.562500                   -0.531250
## as.factor(block_variable_)4                          A:B
##                    -1.218750                   -0.203125
##                          A:C                          B:C
##                     0.453125                    0.296875
##                          A:D                          B:D
##                     0.359375                   -0.109375
##                          C:D                          A:E
##                    -0.265625                   -0.046875
##                          B:E                          C:E
##                    -0.203125                    0.390625
##                          D:E                        A:B:C
##                    -0.265625                   -0.203125
##                        A:B:D                        A:C:D
##                     0.203125                   -0.203125
##                        B:C:D                        A:B:E
##                    -0.046875                    0.421875
##                        A:C:E                        B:C:E
```

```
##                     0.015625                     0.484375
##                        A:D:E                        B:D:E
##                    -0.078125                     0.453125
##                        C:D:E                      A:B:C:D
##                    -0.015625                     0.203125
##                      A:B:C:E                      A:B:D:E
##                    -0.203125                    -0.750000
##                      A:C:D:E                      B:C:D:E
##                     0.484375                    -0.109375
##                    A:B:C:D:E
##                     0.968750
```

```
(multiplier1 = qt(0.1/2/31,df = 64-31-1-1,lower.tail = F))
```

```
## [1] 3.192655
```

```
################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)

sim_ci = cbind(effects_user - multiplier2 * sd_user, effects_user + multiplier2 * sd_user)
colnames(sim_ci) <- c('Lower Bound','Upper Bound')
sim_ci
```

```
##                               Lower Bound Upper Bound
## (Intercept)                  10.95735139  12.9801486
## A                            -1.12077361   0.9020236
## B                            -1.15202361   0.8707736
## C                             1.62922639   3.6520236
## D                            -0.77702361   1.2457736
## E                            -0.24577361   1.7770236
## as.factor(block_variable_)2 -4.57389861  -2.5511014
## as.factor(block_variable_)3 -1.54264861   0.4801486
## as.factor(block_variable_)4 -2.23014861  -0.2073514
## A:B                          -1.21452361   0.8082736
## A:C                          -0.55827361   1.4645236
## B:C                          -0.71452361   1.3082736
## A:D                          -0.65202361   1.3707736
## B:D                          -1.12077361   0.9020236
## C:D                          -1.27702361   0.7457736
## A:E                          -1.05827361   0.9645236
## B:E                          -1.21452361   0.8082736
## C:E                          -0.62077361   1.4020236
## D:E                          -1.27702361   0.7457736
## A:B:C                        -1.21452361   0.8082736
## A:B:D                        -0.80827361   1.2145236
## A:C:D                        -1.21452361   0.8082736
## B:C:D                        -1.05827361   0.9645236
## A:B:E                        -0.58952361   1.4332736
## A:C:E                        -0.99577361   1.0270236
## B:C:E                        -0.52702361   1.4957736
## A:D:E                        -1.08952361   0.9332736
## B:D:E                        -0.55827361   1.4645236
## C:D:E                        -1.02702361   0.9957736
## A:B:C:D                      -0.80827361   1.2145236
## A:B:C:E                      -1.21452361   0.8082736
```

```
## A:B:D:E                     -1.76139861   0.2613986
## A:C:D:E                     -0.52702361   1.4957736
## B:C:D:E                     -1.12077361   0.9020236
## A:B:C:D:E                   -0.04264861   1.9801486
```

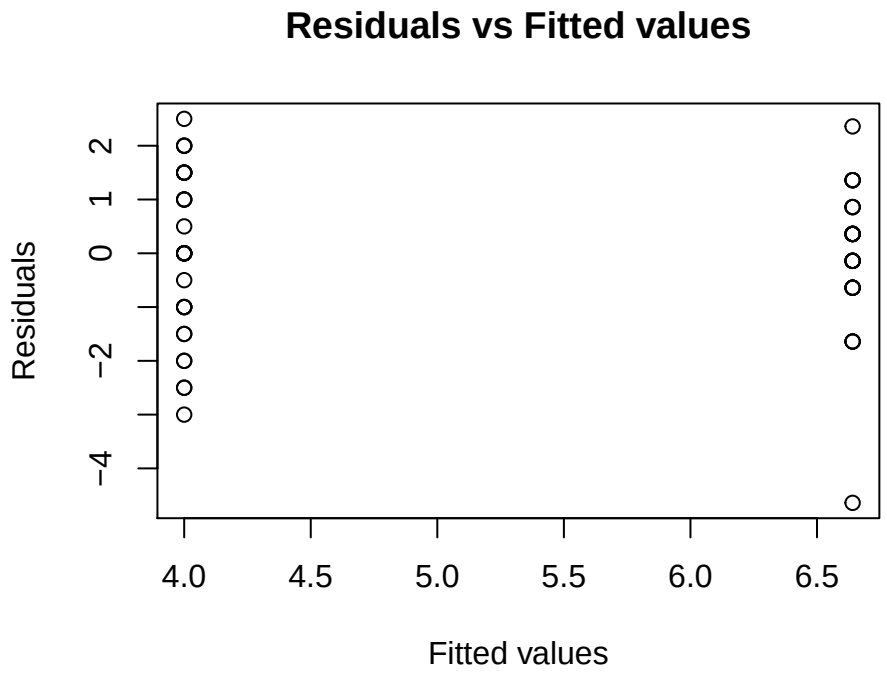
```
#####################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci2 = cbind(effects_user - multiplier1 * sd_user, effects_user + multiplier1 * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                 10.94267842  12.9948216
## A                           -1.13544658   0.9166966
## B                           -1.16669658   0.8854466
## C                            1.61455342   3.6666966
## D                           -0.79169658   1.2604466
## E                           -0.26044658   1.7916966
## as.factor(block_variable_)2 -4.58857158  -2.5364284
## as.factor(block_variable_)3 -1.55732158   0.4948216
## as.factor(block_variable_)4 -2.24482158  -0.1926784
## A:B                         -1.22919658   0.8229466
## A:C                         -0.57294658   1.4791966
## B:C                         -0.72919658   1.3229466
## A:D                         -0.66669658   1.3854466
## B:D                         -1.13544658   0.9166966
## C:D                         -1.29169658   0.7604466
## A:E                         -1.07294658   0.9791966
## B:E                         -1.22919658   0.8229466
## C:E                         -0.63544658   1.4166966
## D:E                         -1.29169658   0.7604466
## A:B:C                       -1.22919658   0.8229466
## A:B:D                       -0.82294658   1.2291966
## A:C:D                       -1.22919658   0.8229466
## B:C:D                       -1.07294658   0.9791966
## A:B:E                       -0.60419658   1.4479466
## A:C:E                       -1.01044658   1.0416966
## B:C:E                       -0.54169658   1.5104466
## A:D:E                       -1.10419658   0.9479466
## B:D:E                       -0.57294658   1.4791966
## C:D:E                       -1.04169658   1.0104466
## A:B:C:D                     -0.82294658   1.2291966
## A:B:C:E                     -1.22919658   0.8229466
## A:B:D:E                     -1.77607158   0.2760716
## A:C:D:E                     -0.54169658   1.5104466
## B:C:D:E                     -1.13544658   0.9166966
## A:B:C:D:E                   -0.05732158   1.9948216
```

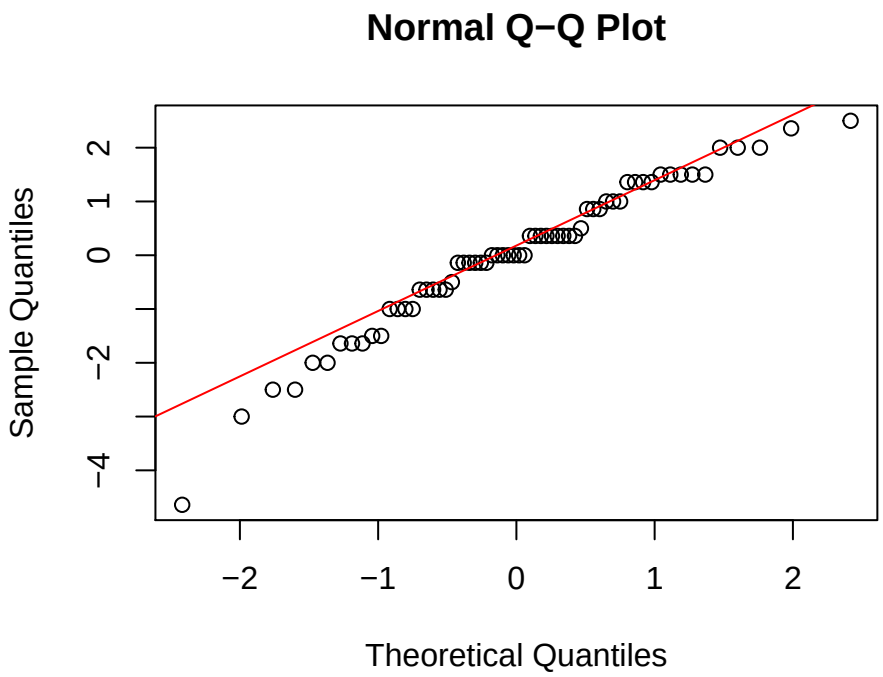
```
######################
```

SECTION A.4.2.3 :FITTING THE MODEL AND DIAGNOSTIC PLOTS

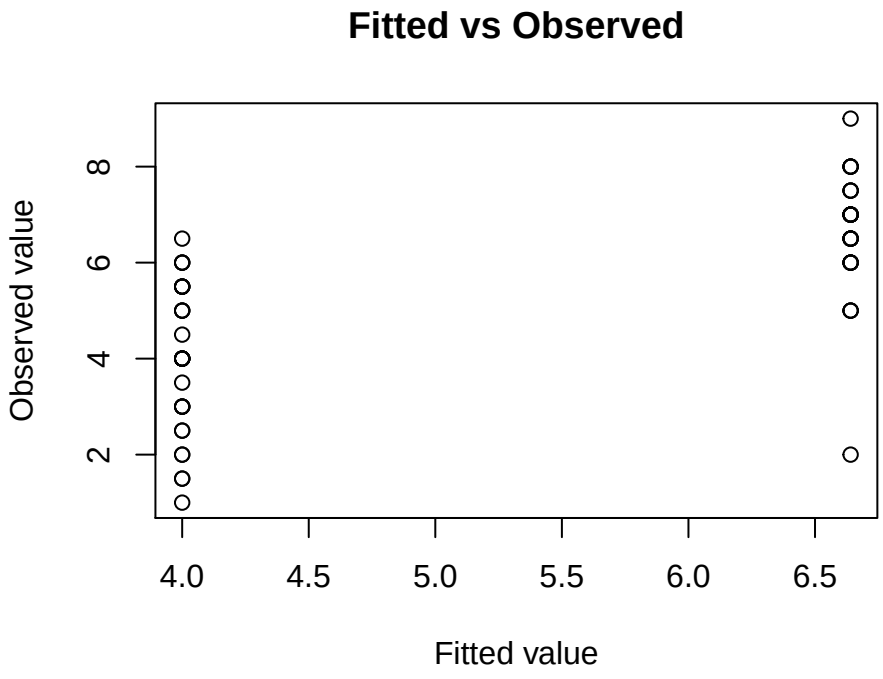
```
m_fit_6=lm(y~C,data=full_data)

resid = m_fit_6$resid
fitted = m_fit_6$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```




```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
group = as.factor(c(rep(c(1,2),8),rep(c(3,4),8)))
plot(cooks.distance(m_fit_6), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```


```
####################
```

SECTION A.4.2.4 : STUDYING THE EFFECT OF TRANSFORMATIONS

```
A= full_data$A
B= full_data$B
C= full_data$C
D= full_data$D
E= full_data$E
y = full_data$y

y0=full_data$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+as.factor(block_variable_))
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
```

```
lambda=op$minimum
lambda
```

```
## [1] 1.645621
```

```
y_t=y0^lambda
model_trans2=lm(y_t~ A+B+C+D+E+as.factor(block_variable_))
anova(model_trans2)
```

```
## Analysis of Variance Table
##
## Response: y_t
##                             Df  Sum Sq Mean Sq F value    Pr(>F)
## A                            1    0.44    0.44  0.0140  0.906178
## B                            1    4.97    4.97  0.1596  0.691044
## C                            1 2477.67 2477.67 79.5563 2.844e-12 ***
## D                            1    3.13    3.13  0.1005  0.752374
## E                            1  244.21  244.21  7.8413  0.007032 **
## as.factor(block_variable_)  3  345.60  115.20  3.6990  0.016955 *
## Residuals                   55 1712.90   31.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(round(p.adjust(summary(model_trans2)$coefficients[,4]),4))
```

```
##                 (Intercept)                           A
##                      0.0000                      1.0000
##                           B                           C
##                      1.0000                      0.0000
##                           D                           E
##                      1.0000                      0.0492
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                      0.3176                      1.0000
## as.factor(block_variable_)4
##                      1.0000
```

```
# g = 5, dof = 64-5-3-1
(multiplier_trans = smm_user(colnum = c(40,55,60),
                      rownum = c(3,5,6),
                      m = matrix(c(2.183,2.470,2.160,2.439),2,2))) # g = 5, v = 64-5-3-1
```

```
## [1] 2.353
```

```
(sigma2 = anova(model_trans2)[7,3])
```

```
## [1] 31.14363
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 1.395162
```

```
(effects_user = 2 * model_trans2$coefficients)
```

```
##                 (Intercept)                           A
##                  34.8550570                  -0.1651948
##                           B                           C
##                  -0.5574212                  12.4440509
##                           D                           E
```

```
##                   0.4423945                   3.9067833
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                  -7.8060762                   5.0642902
## as.factor(block_variable_)4
##                  -2.6738549
```

```
sim_ci2 = cbind(effects_user - multiplier_trans * sd_user, effects_user + multiplier_trans * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
#############################################################
## STUDENTIZED MAXIMUM MODULUS CI FOR TRANSFORMED DATA
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                  31.5722409  38.1378730
## A                            -3.4480108   3.1176213
## B                            -3.8402373   2.7253948
## C                             9.1612348  15.7268669
## D                            -2.8404215   3.7252106
## E                             0.6239672   7.1895993
## as.factor(block_variable_)2 -11.0888922  -4.5232601
## as.factor(block_variable_)3   1.7814742   8.3471063
## as.factor(block_variable_)4  -5.9566710   0.6089611
```

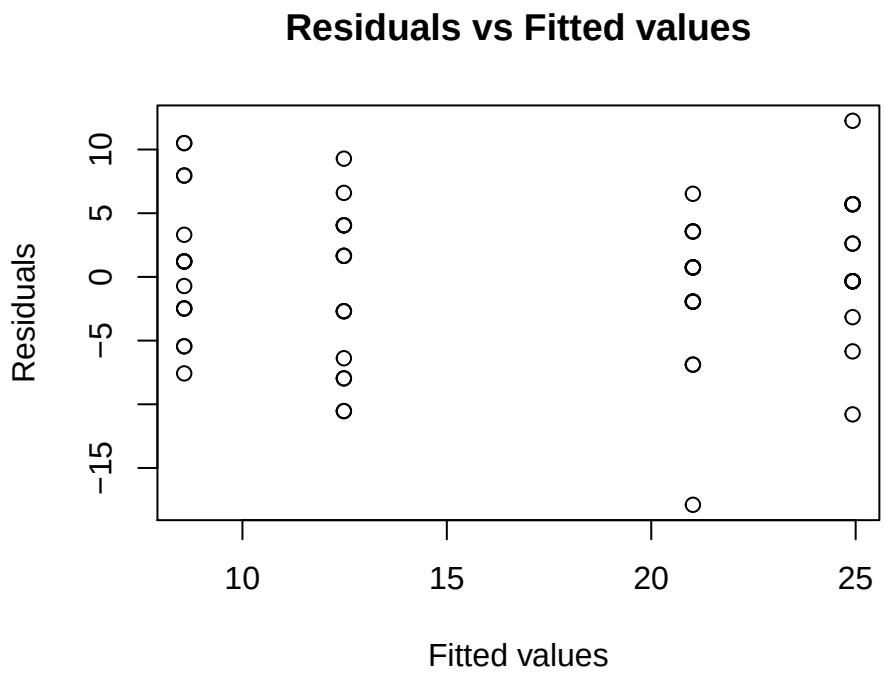
```
# as C and E are significant at 0.1 Confidence levels
final_transformed_model = lm(y_t~C+E)
2 * final_transformed_model$coefficients
```

```
## (Intercept)           C           E
##   33.501147   12.444051    3.906783
```

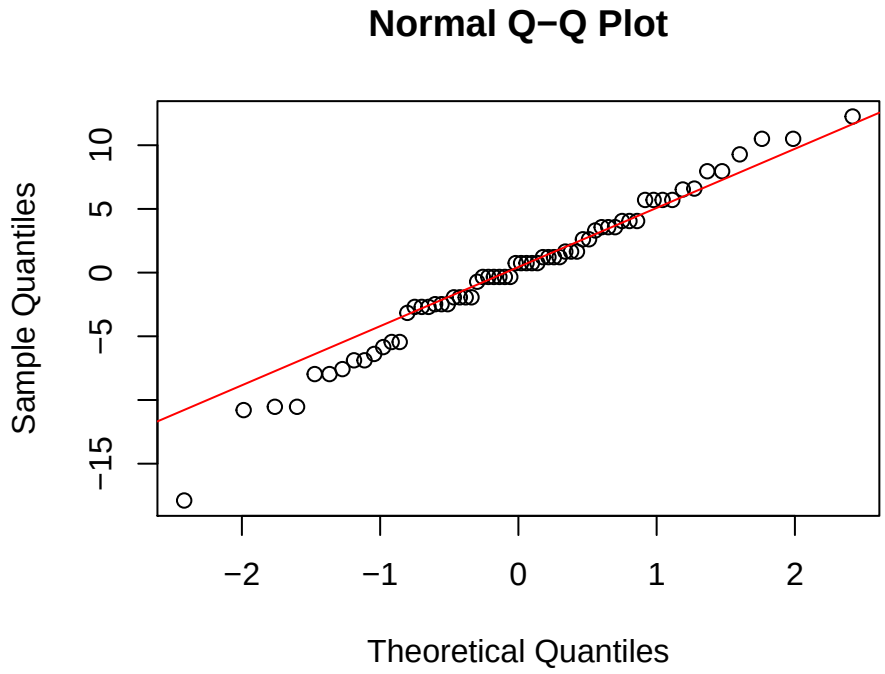
```
summary(final_transformed_model)
```

```
##
## Call:
## lm(formula = y_t ~ C + E)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.8904  -2.6924   0.7452   3.5681  12.2550
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  16.7506     0.7276  23.020  < 2e-16 ***
## C             6.2220     0.7276   8.551 4.99e-12 ***
## E             1.9534     0.7276   2.685  0.00934 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.821 on 61 degrees of freedom
## Multiple R-squared:  0.5684, Adjusted R-squared:  0.5542
## F-statistic: 40.16 on 2 and 61 DF,  p-value: 7.428e-12
```

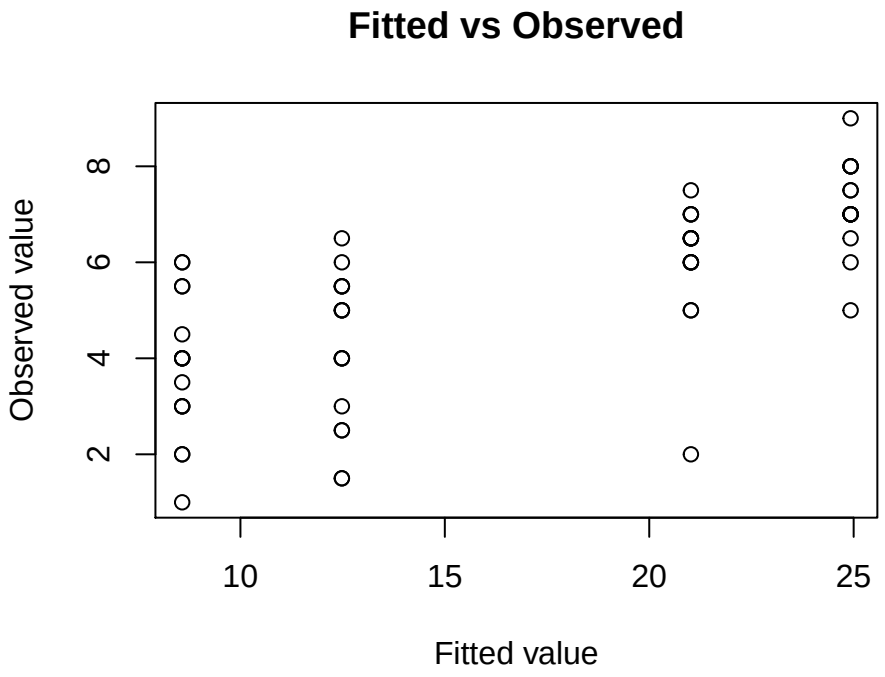
```
resid = final_transformed_model$resid
fitted = final_transformed_model$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



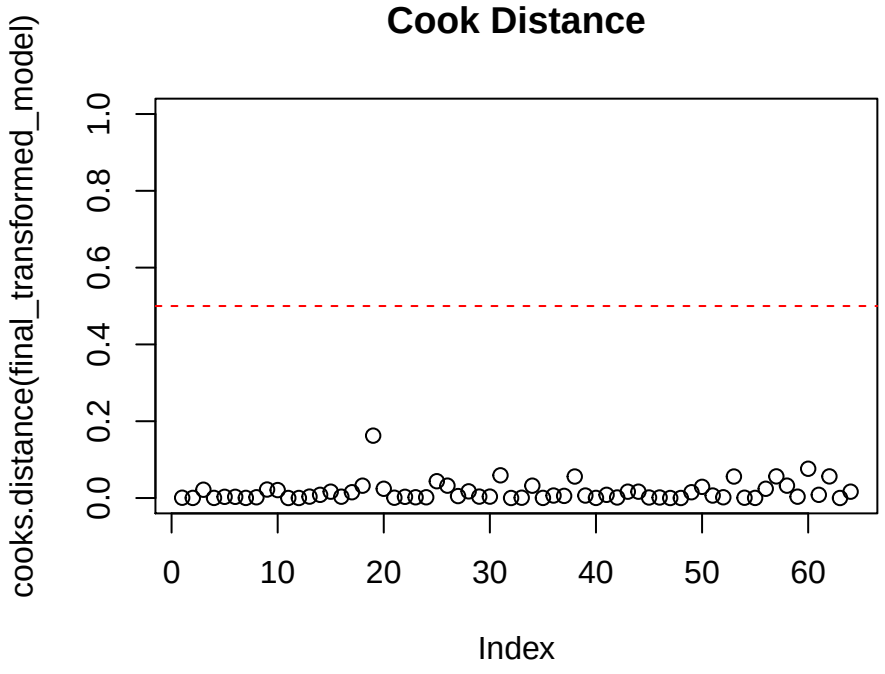

```
qqnorm(resid)
qqline(resid,col="red")
```



```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```

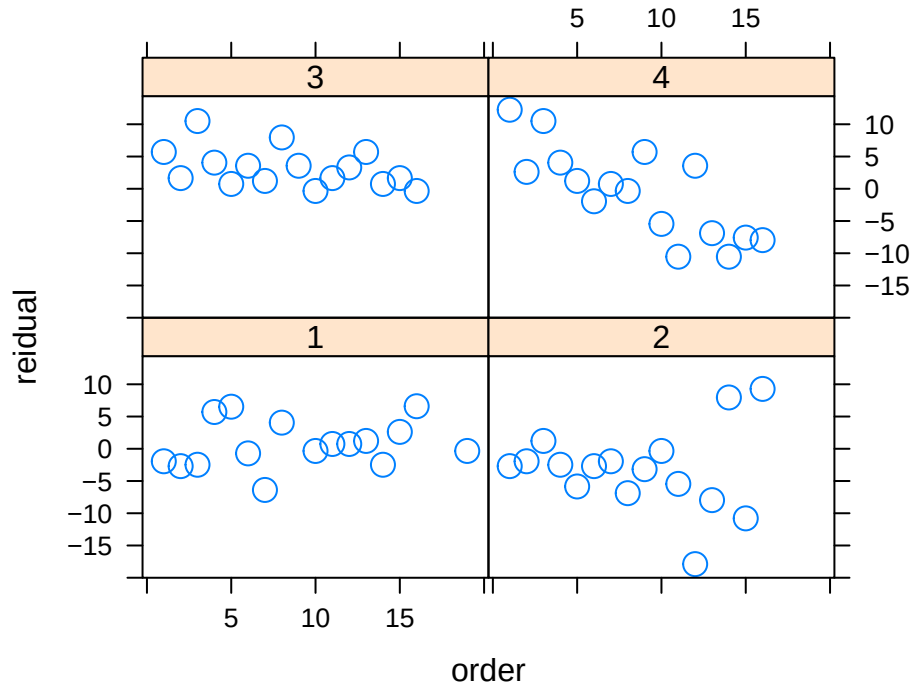



```
plot(cooks.distance(final_transformed_model), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



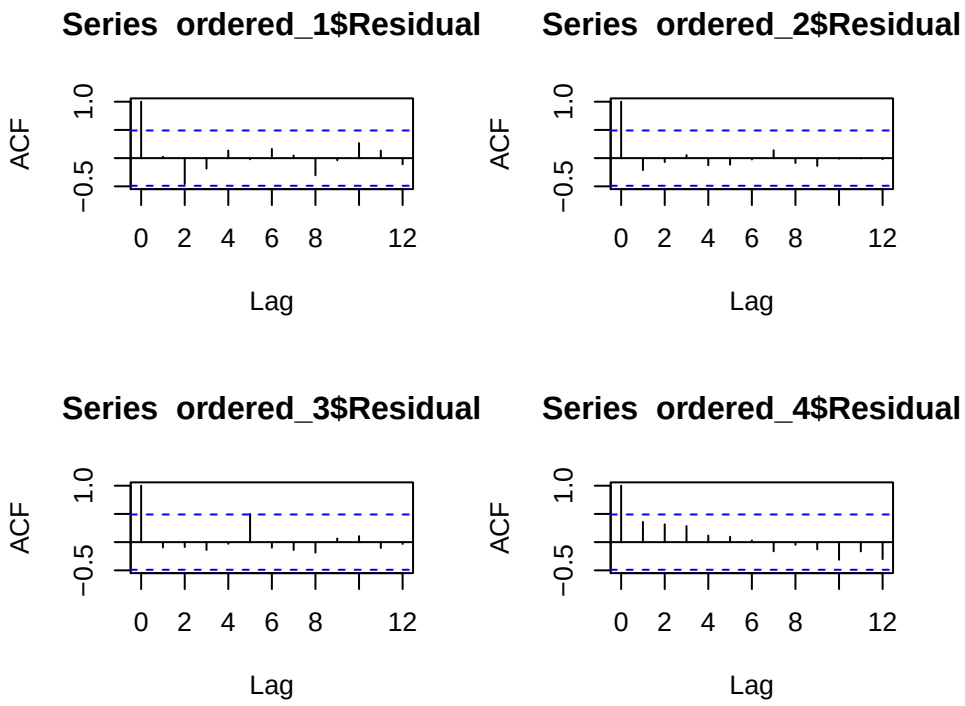
SECTION A.4.2.5 : STUDYING IF THERE IS ANY TREND

```
############### PLOT TO SEE IF THERE IS TREND IN RESIDUALS ###############

order_= c(files_1$O_AG, files_2$O_AG)
ord_data_ = data.frame(order = order_, reidual =resid, days= block_variable_)


library(lattice)
xyplot(reidual ~ order | as.factor(block_variable_), data = ord_data_,
       auto.key = list(corner = c(0, .98)), cex = 1.5)
```




```
ordered1 = ord_data_[ord_data_$days==1,]
colnames(ordered1) <- c("Ordering", "Residual","Days")
ordered_1 = ordered1[order(ordered1$Ordering),]

ordered2 = ord_data_[ord_data_$days==2,]
colnames(ordered2) <- c("Ordering", "Residual","Days")
ordered_2 = ordered2[order(ordered2$Ordering),]

ordered3 = ord_data_[ord_data_$days==3,]
colnames(ordered3) <- c("Ordering", "Residual","Days")
ordered_3 = ordered3[order(ordered3$Ordering),]

ordered4 = ord_data_[ord_data_$days==4,]
colnames(ordered4) <- c("Ordering", "Residual","Days")
ordered_4 = ordered4[order(ordered4$Ordering),]

par(mfrow=c(2,2))
```

```
acf(ordered_1$Residual)
acf(ordered_2$Residual)
acf(ordered_3$Residual)
acf(ordered_4$Residual)
```



ANALYSIS_JUDGE_KJ

Abhijeet Bhardwaj

4/28/2021

SECTION A.5: ANALYSIS OF SCORES BY JUDGE KJ

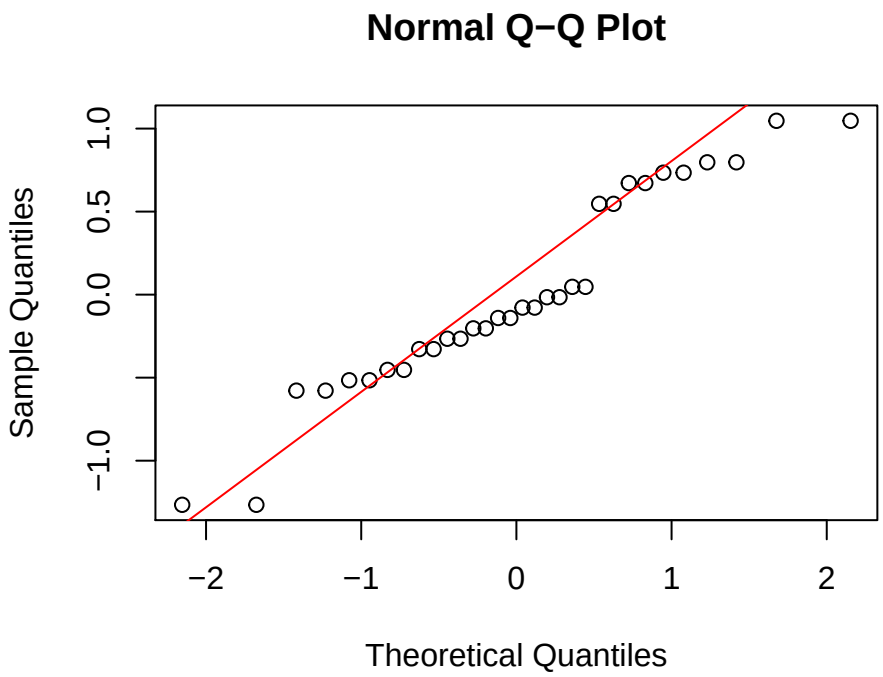
SECTION A.5.1 :Analysis of first run by judge KJ

#SECTION A.5.1.1 DATA VISUALIZATION

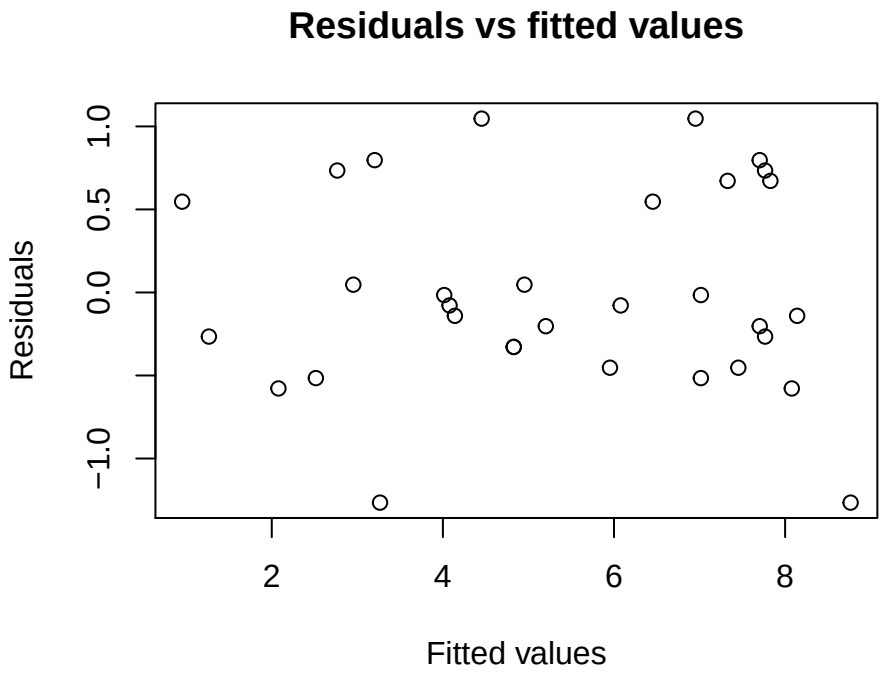
```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
#files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3
A = files_1$A
B = files_1$B
C = files_1$C
D = files_1$D
E = files_1$E
block = files_1$Block_ABCDE
avg_score = files_1$J_KJ
order_scoring = files_1$O_KJ
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^3+block,data=subset_dat_)
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=6) to new 7 after EncodeVars() -- should no longer happen!
```

```
resid = m_fit_1$resid
fitted = m_fit_1$fit
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs fitted values")
```



SECTION A.5.1.2 ANALYZING SIGNIFICANCE USING CONFIDENCE INTERVALS

```
# ASSUMING 4 FI's TO BE ZERO but not ABCDE as it is a block effect for sigma
#fitting complete model to get estimates of 4fis
m_fit_2 = lm(y~.^5,data=subset_dat_)
summary_fullmodel_ = summary(m_fit_2)
2*summary_fullmodel_$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.84375        NaN     NaN      NaN
## A            0.84375        NaN     NaN      NaN
## B            1.21875        NaN     NaN      NaN
## C            2.34375        NaN     NaN      NaN
## D           -0.71875        NaN     NaN      NaN
## E            1.34375        NaN     NaN      NaN
## A:B         -0.03125        NaN     NaN      NaN
## A:C          0.96875        NaN     NaN      NaN
## A:D          0.28125        NaN     NaN      NaN
## A:E          0.09375        NaN     NaN      NaN
## B:C          0.71875        NaN     NaN      NaN
## B:D          0.28125        NaN     NaN      NaN
## B:E          0.09375        NaN     NaN      NaN
## C:D          0.65625        NaN     NaN      NaN
## C:E         -1.28125        NaN     NaN      NaN
## D:E         -0.34375        NaN     NaN      NaN
## A:B:C       -0.65625        NaN     NaN      NaN
## A:B:D        0.78125        NaN     NaN      NaN
## A:B:E        1.09375        NaN     NaN      NaN
## A:C:D       -0.46875        NaN     NaN      NaN
## A:C:E       -0.65625        NaN     NaN      NaN
## A:D:E        0.40625        NaN     NaN      NaN
## B:C:D       -1.09375        NaN     NaN      NaN
## B:C:E       -1.03125        NaN     NaN      NaN
## B:D:E        0.03125        NaN     NaN      NaN
## C:D:E       -0.09375        NaN     NaN      NaN
## A:B:C:D      0.53125        NaN     NaN      NaN
## A:B:C:E      0.59375        NaN     NaN      NaN
## A:B:D:E     -0.71875        NaN     NaN      NaN
## A:C:D:E     -0.21875        NaN     NaN      NaN
## B:C:D:E     -0.46875        NaN     NaN      NaN
## A:B:C:D:E    0.65625        NaN     NaN      NaN
```

```
coeffs_4fis = 2*summary_fullmodel_$coefficients[27:31,1]
sd_4 = sqrt(sum(coeffs_4fis^2)/5)
threshold_4 = sd_4*qt(0.1/(2*25),5,lower.tail = FALSE)
all_coeffs_ = 2*summary_fullmodel_$coefficients[,1]
eff = as.numeric(all_coeffs_)
which(abs(eff)>threshold_4) # none
```

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

```
# ASSUMING 3 and 4 FI's TO BE ZERO but not ABCDE as it is a block effect for estimating sigma
#fitting complete model to get estimates of 4fis
coeffs_34fis = 2*summary_fullmodel_$coefficients[17:31,1]
```

```
var_34 = sum(coeffs_34fis^2)/15
sd_34 = sqrt(var_34)
threshold_34 = sd_4*qt(0.1/(2*15),15,lower.tail = FALSE)
threshold_34
```

```
## [1] 1.675696
```

```
which(abs(eff)>threshold_34) # none significant
```

```
## [1] 1 4
```

SECTION A.5.1.3 ANALYZING SIGNIFICANCE USING DANIEL METHOD

```
## APPLY DANIEL METHOD
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

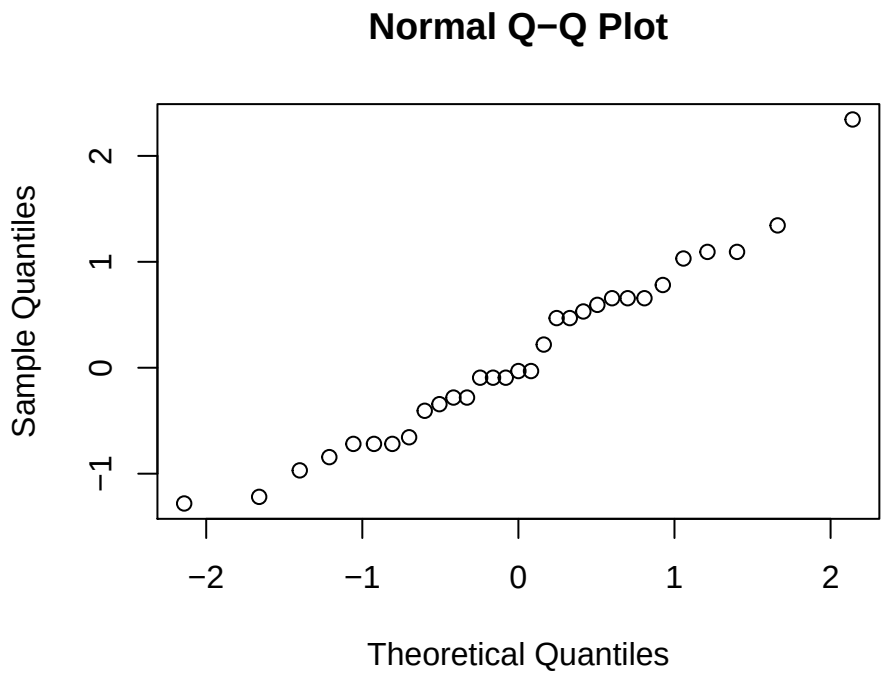
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^5,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 8
```

```
## [1] 8
```

```r
A_final = perm[1,8] * A
B_final = perm[2,8] * B
C_final = perm[3,8] * C
D_final = perm[4,8] * D
E_final = perm[5,8] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^5,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -1.21123213 -1.66069761  2.14119812 -0.80754104  1.66069761  0.00000000
##  [7] -1.40074506 -0.41598722 -0.08094729 -0.92524456 -0.32929135 -0.24500622
## [13]  0.60017878 -2.14119812 -0.50593365 -0.70009021  0.92524456  1.40074506
## [19]  0.24500622  0.70009021 -0.60017878  1.21123213  1.05741423  0.08094729
## [25] -0.16242937  0.41598722  0.50593365 -1.05741423  0.16242937  0.32929135
## [31]  0.80754104
##
## $y
##                                A_final                                B_final
##                               -0.84375                               -1.21875
##                                C_final                                D_final
##                                2.34375                               -0.71875
##                                E_final                      A_final:B_final
##                                1.34375                               -0.03125
##                      A_final:C_final                      A_final:D_final
##                               -0.96875                               -0.28125
##                      A_final:E_final                      B_final:C_final
##                               -0.09375                               -0.71875
##                      B_final:D_final                      B_final:E_final
```

```
##                                -0.28125                                -0.09375
##                          C_final:D_final                          C_final:E_final
##                                 0.65625                                 -1.28125
##                          D_final:E_final                  A_final:B_final:C_final
##                                -0.34375                                -0.65625
##                  A_final:B_final:D_final                  A_final:B_final:E_final
##                                 0.78125                                 1.09375
##                  A_final:C_final:D_final                  A_final:C_final:E_final
##                                 0.46875                                 0.65625
##                  A_final:D_final:E_final                  B_final:C_final:D_final
##                                -0.40625                                 1.09375
##                  B_final:C_final:E_final                  B_final:D_final:E_final
##                                 1.03125                                -0.03125
##                  C_final:D_final:E_final          A_final:B_final:C_final:D_final
##                                -0.09375                                 0.53125
##          A_final:B_final:C_final:E_final          A_final:B_final:D_final:E_final
##                                 0.59375                                -0.71875
##          A_final:C_final:D_final:E_final          B_final:C_final:D_final:E_final
##                                 0.21875                                 0.46875
## A_final:B_final:C_final:D_final:E_final
##                                 0.65625
```

```
# Step2: Fit a lm ------------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier ------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # outliers:::named integer(0) NONE integer
```

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

```
##############################################################################
```

SECTION A.5.1.4 ANALYZING SIGNIFICANCE USING DONG's METHOD

```
#ANALYSIS BY DONGS METHOD
coeff=coef(m_fit_2)*2
theta1=abs(na.omit(coeff[-c(1,31)]))
s0=1.5*median(theta1)
m1=sum(theta1<=2.5*s0)
s1=sqrt(sum(theta1[theta1<=2.5*s0]^2)/m1)
m2=sum(theta1<=2.5*s1)
s2=sqrt(sum(theta1[theta1<=2.5*s1]^2)/m2)
g=length(theta1)
gamma=0.5*(1-(1-0.05)^(1/g))
print(paste0("after DONG's method significant effect :", names(theta1[theta1>s2*qt(gamma,m2,lower.tail =
```

```
## [1] "after DONG's method significant effect :"
```

```
################################################
# none of the factors are significant
```

SECTION A.5.1.5 APPLYING TRANSFORMATION TO SEE IF MORE SIGNIFICANT FACTORS CAN BE EXTRACTED

```
y0=subset_dat_$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+block)
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda #  1.184087
```

```
## [1] 1.184087
```

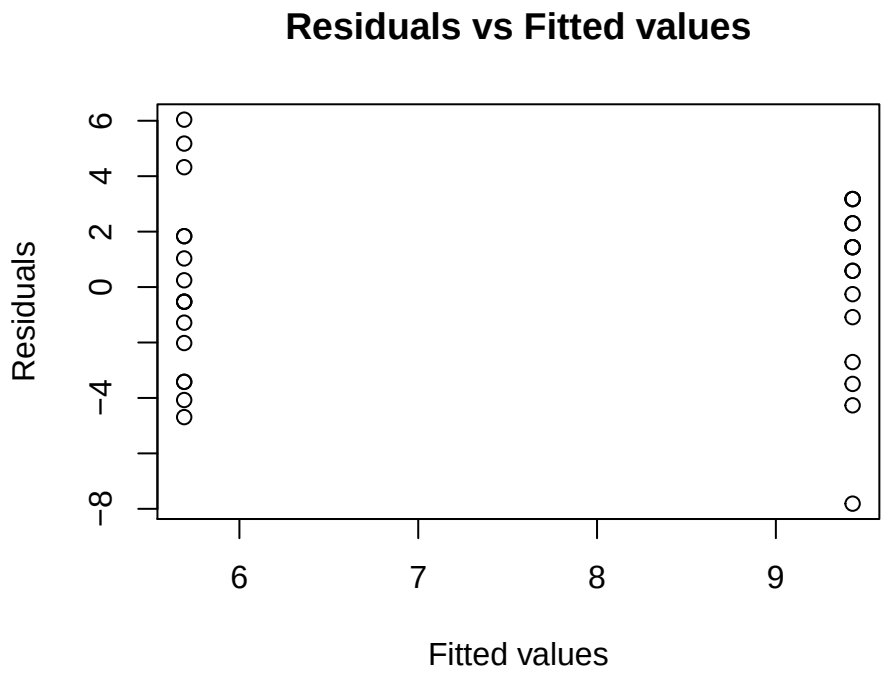
```
y_t=y0^lambda
data_subs_bc_t=data.frame(y_t= y_t,A=A,B=B,C=C,D=D,E=E)
model_trans=lm(y_t~ A+B+C+D+E+block)
summary(model_trans)
```

```
##
## Call:
## lm(formula = y_t ~ A + B + C + D + E + block)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.173 -1.653 -0.027  1.615  5.647
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.5602     0.5106  14.807 7.03e-14 ***
## A             0.6940     0.5106   1.359  0.18620
## B             0.9864     0.5106   1.932  0.06478 .
## C             1.8690     0.5106   3.661  0.00118 **
## D            -0.5689     0.5106  -1.114  0.27576
## E             1.0164     0.5106   1.991  0.05755 .
## block         0.4881     0.5106   0.956  0.34828
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.888 on 25 degrees of freedom
## Multiple R-squared:  0.501,  Adjusted R-squared:  0.3812
## F-statistic: 4.183 on 6 and 25 DF,  p-value: 0.004791
```

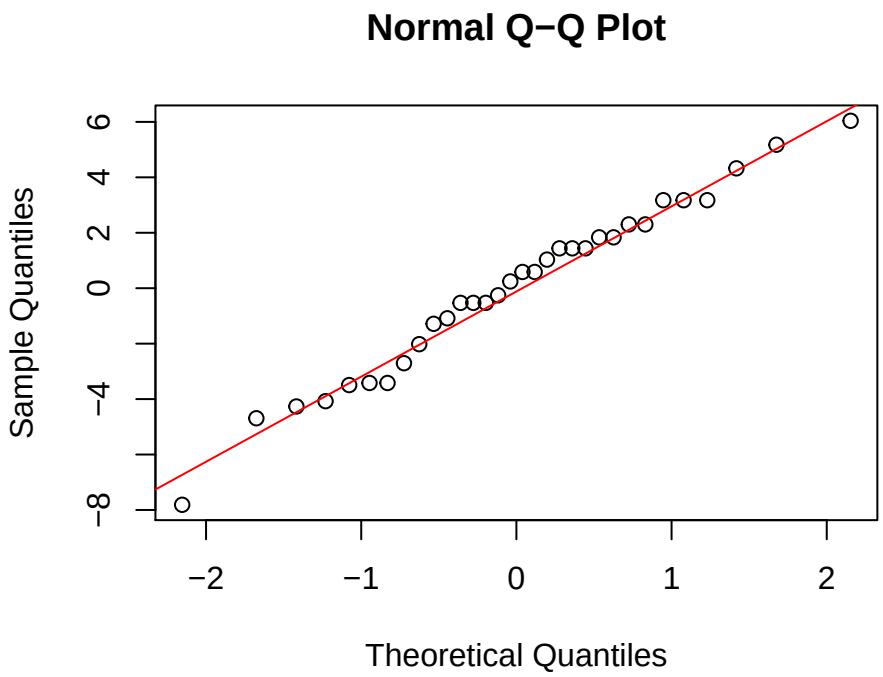
```
print(round(p.adjust(summary(model_trans)$coefficients[,4]),4))
```

```
## (Intercept)           A           B           C           D           E
##      0.0000      0.5586      0.2877      0.0071      0.5586      0.2877
##       block
##      0.5586
```

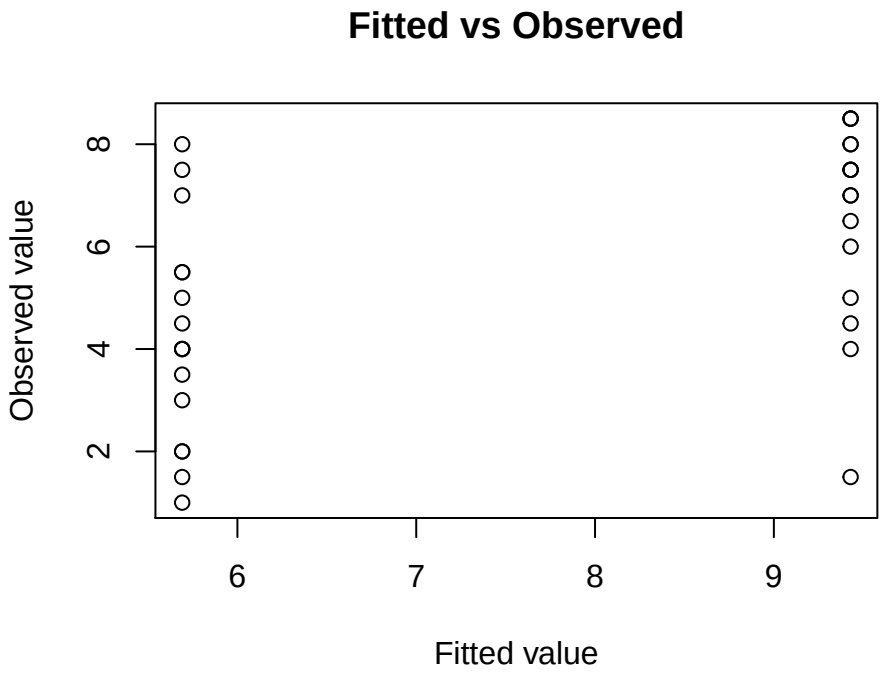
```
model_trans=lm(y_t~ C)
resid = model_trans$resid
fitted = model_trans$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```




```
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(fitted,subset_dat_$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
plot(cooks.distance(model_trans), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```

SECTION A.5.2 :Analysis of Second run and combined two runs for Judge KJ

SECTION A.5.2.1 :Estimates from second run

```
# COMBINING THE SECOND COMPLETE REPLICATE

files_2 = read.csv('Run_2_with_scores.csv')
#files_2$J_AVG = (files_2$J_AD+files_2$J_AG+files_2$J_KJ)/3
A2 = files_2$A
B2 = files_2$B
C2 = files_2$C
D2 = files_2$D
E2 = files_2$E
avg_score2 = files_2$J_KJ
subset_dat_2 = data.frame(y = avg_score2,A=A2,B=B2,C=C2,D=D2,E=E2)
m_fit_4 = lm(y~.^5,data=subset_dat_2)
summary_fullmodel_2 = summary(m_fit_4)
2*summary_fullmodel_2$coefficients
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.875000e+00        NaN     NaN      NaN
## A            1.250000e+00        NaN     NaN      NaN
## B            6.250000e-02        NaN     NaN      NaN
## C            2.812500e+00        NaN     NaN      NaN
## D           -5.000000e-01        NaN     NaN      NaN
## E            5.625000e-01        NaN     NaN      NaN
```

```
## A:B          6.875000e-01        NaN     NaN      NaN
## A:C          8.125000e-01        NaN     NaN      NaN
## A:D          3.750000e-01        NaN     NaN      NaN
## A:E          4.375000e-01        NaN     NaN      NaN
## B:C         -5.000000e-01        NaN     NaN      NaN
## B:D         -4.375000e-01        NaN     NaN      NaN
## B:E          2.500000e-01        NaN     NaN      NaN
## C:D         -1.875000e-01        NaN     NaN      NaN
## C:E          2.500000e-01        NaN     NaN      NaN
## D:E         -5.625000e-01        NaN     NaN      NaN
## A:B:C        8.218453e-17        NaN     NaN      NaN
## A:B:D       -8.125000e-01        NaN     NaN      NaN
## A:B:E        3.750000e-01        NaN     NaN      NaN
## A:C:D        3.125000e-01        NaN     NaN      NaN
## A:C:E        5.000000e-01        NaN     NaN      NaN
## A:D:E       -1.875000e-01        NaN     NaN      NaN
## B:C:D        6.250000e-01        NaN     NaN      NaN
## B:C:E       -1.312500e+00        NaN     NaN      NaN
## B:D:E        5.000000e-01        NaN     NaN      NaN
## C:D:E       -2.500000e-01        NaN     NaN      NaN
## A:B:C:D     -3.750000e-01        NaN     NaN      NaN
## A:B:C:E      4.375000e-01        NaN     NaN      NaN
## A:B:D:E      2.625000e+00        NaN     NaN      NaN
## A:C:D:E      7.212612e-16        NaN     NaN      NaN
## B:C:D:E     -1.875000e-01        NaN     NaN      NaN
## A:B:C:D:E   -4.375000e-01        NaN     NaN      NaN
```

SECTION A.5.2.2 :Calculating Confidence Intervals by combining the replicates

```
# CACULATION OF VARIANCE BY FIRST PRINCIPLES
coeffs_1st_rep = 2*summary_fullmodel_$coefficients[-1,1]
coeffs_2st_rep = 2*summary_fullmodel_2$coefficients[-1,1]
avg_coeff_ = (coeffs_1st_rep+coeffs_2st_rep)/2
diff_1 = (coeffs_1st_rep-avg_coeff_)^2
diff_2 = (coeffs_2st_rep-avg_coeff_)^2
diff_1_ = diff_1[-c(31,28)]
diff_2_ = diff_2[-c(31,28)]

var_ = (diff_1_+diff_2_)/(2-1)
var_hat_ = sum(var_)/29 #estimate of 4sigma^2/2^k) \hat{var(Ai)}
multiplier1 = qt(0.1/2/31,df = 64-31-3-1,lower.tail = F)
boundary_pt =   multiplier1*sqrt(var_hat_/2)
print(paste0("standard deviation using first principle: ", sqrt(var_hat_/2)))
```

```
## [1] "standard deviation using first principle: 0.396273811560199"
```

```
avg_coeff_up = avg_coeff_+boundary_pt
avg_coeff_dwn = avg_coeff_-boundary_pt
CI_interval = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn, Up_=

#################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
```

```
CI_interval
```

```
##           variable_  estimate       Lwr_       Up_
## A                 A  1.046875 -0.2255907 2.3193407
## B                 B  0.640625 -0.6318407 1.9130907
## C                 C  2.578125  1.3056593 3.8505907
## D                 D -0.609375 -1.8818407 0.6630907
## E                 E  0.953125 -0.3193407 2.2255907
## A:B             A:B  0.328125 -0.9443407 1.6005907
## A:C             A:C  0.890625 -0.3818407 2.1630907
## A:D             A:D  0.328125 -0.9443407 1.6005907
## A:E             A:E  0.265625 -1.0068407 1.5380907
## B:C             B:C  0.109375 -1.1630907 1.3818407
## B:D             B:D -0.078125 -1.3505907 1.1943407
## B:E             B:E  0.171875 -1.1005907 1.4443407
## C:D             C:D  0.234375 -1.0380907 1.5068407
## C:E             C:E -0.515625 -1.7880907 0.7568407
## D:E             D:E -0.453125 -1.7255907 0.8193407
## A:B:C         A:B:C -0.328125 -1.6005907 0.9443407
## A:B:D         A:B:D -0.015625 -1.2880907 1.2568407
## A:B:E         A:B:E  0.734375 -0.5380907 2.0068407
## A:C:D         A:C:D -0.078125 -1.3505907 1.1943407
## A:C:E         A:C:E -0.078125 -1.3505907 1.1943407
## A:D:E         A:D:E  0.109375 -1.1630907 1.3818407
## B:C:D         B:C:D -0.234375 -1.5068407 1.0380907
## B:C:E         B:C:E -1.171875 -2.4443407 0.1005907
## B:D:E         B:D:E  0.265625 -1.0068407 1.5380907
## C:D:E         C:D:E -0.171875 -1.4443407 1.1005907
## A:B:C:D     A:B:C:D  0.078125 -1.1943407 1.3505907
## A:B:C:E     A:B:C:E  0.515625 -0.7568407 1.7880907
## A:B:D:E     A:B:D:E  0.953125 -0.3193407 2.2255907
## A:C:D:E     A:C:D:E -0.109375 -1.3818407 1.1630907
## B:C:D:E     B:C:D:E -0.328125 -1.6005907 0.9443407
## A:B:C:D:E A:B:C:D:E  0.109375 -1.1630907 1.3818407
```

```
# USING STUDENTIZED MAX MODULUS
smm_user <- function(colnum, rownum, m){
  r1 = weighted.mean(m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r2 = 1 / weighted.mean(1 / m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r3 = weighted.mean(m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r4 = 1 / weighted.mean(1 / m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  rl = mean(r1,r2)
  rr = mean(r3,r4)
  c1 = weighted.mean(c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  c2 = 1 / weighted.mean(1 / c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  return(round(mean(c1,c2),3))
}

#g=31 dof = 64-31-3-1 #
(multiplier2 = smm_user(colnum = c(24,29,30),
                        rownum = c(28,31,36),
                        m = matrix(c(3.156,3.251,3.104,3.196),2,2))) # g =31, v = 64-31-3-1
```

```
## [1] 3.147
```

```
boundary_pt2 =   multiplier2*sqrt(var_hat_/2)
avg_coeff_up2 = avg_coeff_+boundary_pt2
avg_coeff_dwn2 = avg_coeff_-boundary_pt2
CI_interval2 = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn2, Up
#################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval2
```

```
##           variable_  estimate       Lwr_        Up_
## A                 A  1.046875 -0.2001987 2.29394868
## B                 B  0.640625 -0.6064487 1.88769868
## C                 C  2.578125  1.3310513 3.82519868
## D                 D -0.609375 -1.8564487 0.63769868
## E                 E  0.953125 -0.2939487 2.20019868
## A:B             A:B  0.328125 -0.9189487 1.57519868
## A:C             A:C  0.890625 -0.3564487 2.13769868
## A:D             A:D  0.328125 -0.9189487 1.57519868
## A:E             A:E  0.265625 -0.9814487 1.51269868
## B:C             B:C  0.109375 -1.1376987 1.35644868
## B:D             B:D -0.078125 -1.3251987 1.16894868
## B:E             B:E  0.171875 -1.0751987 1.41894868
## C:D             C:D  0.234375 -1.0126987 1.48144868
## C:E             C:E -0.515625 -1.7626987 0.73144868
## D:E             D:E -0.453125 -1.7001987 0.79394868
## A:B:C         A:B:C -0.328125 -1.5751987 0.91894868
## A:B:D         A:B:D -0.015625 -1.2626987 1.23144868
## A:B:E         A:B:E  0.734375 -0.5126987 1.98144868
## A:C:D         A:C:D -0.078125 -1.3251987 1.16894868
## A:C:E         A:C:E -0.078125 -1.3251987 1.16894868
## A:D:E         A:D:E  0.109375 -1.1376987 1.35644868
## B:C:D         B:C:D -0.234375 -1.4814487 1.01269868
## B:C:E         B:C:E -1.171875 -2.4189487 0.07519868
## B:D:E         B:D:E  0.265625 -0.9814487 1.51269868
## C:D:E         C:D:E -0.171875 -1.4189487 1.07519868
## A:B:C:D     A:B:C:D  0.078125 -1.1689487 1.32519868
## A:B:C:E     A:B:C:E  0.515625 -0.7314487 1.76269868
## A:B:D:E     A:B:D:E  0.953125 -0.2939487 2.20019868
## A:C:D:E     A:C:D:E -0.109375 -1.3564487 1.13769868
## B:C:D:E     B:C:D:E -0.328125 -1.5751987 0.91894868
## A:B:C:D:E A:B:C:D:E  0.109375 -1.1376987 1.35644868
```

```
# calculating variance with anova
# Model with c only analysis
subset_dat_$block =files_1$Block_ABCDE
subset_dat_2$block =files_2$ABED
append_data_ = rbind(subset_dat_,subset_dat_2)
block_variable_ = c(rep(1,16),rep(2,16),rep(3,16),rep(4,16))
append_data_$final_block = block_variable_

A3 = append_data_$A
B3 = append_data_$B
C3 = append_data_$C
D3 = append_data_$D
E3 = append_data_$E
```

```r
Y3 = append_data_$y
full_data = data.frame(y = Y3,A=A3,B=B3,C=C3,D=D3,E=E3)

m_fit_5 = lm(y~A*B*C*D*E+as.factor(block_variable_),data=full_data)
#summary(m_fit_3)
anova(m_fit_5)
```

```
## Analysis of Variance Table
##
## Response: y
##                             Df  Sum Sq Mean Sq F value    Pr(>F)
## A                            1  17.535  17.535  6.9791 0.0131510 *
## B                            1   6.566   6.566  2.6135 0.1167878
## C                            1 106.348 106.348 42.3270 4.016e-07 ***
## D                            1   5.941   5.941  2.3647 0.1349474
## E                            1  14.535  14.535  5.7851 0.0227681 *
## as.factor(block_variable_)   3  62.324  20.775  8.2685 0.0003975 ***
## A:B                          1   1.723   1.723  0.6856 0.4144184
## A:C                          1  12.691  12.691  5.0513 0.0323841 *
## B:C                          1   0.191   0.191  0.0762 0.7844988
## A:D                          1   1.723   1.723  0.6856 0.4144184
## B:D                          1   0.098   0.098  0.0389 0.8450862
## C:D                          1   0.879   0.879  0.3498 0.5588035
## A:E                          1   1.129   1.129  0.4493 0.5079655
## B:E                          1   0.473   0.473  0.1881 0.6676956
## C:E                          1   4.254   4.254  1.6931 0.2034391
## D:E                          1   3.285   3.285  1.3075 0.2621976
## A:B:C                        1   1.723   1.723  0.6856 0.4144184
## A:B:D                        1   0.004   0.004  0.0016 0.9688179
## A:C:D                        1   0.098   0.098  0.0389 0.8450862
## B:C:D                        1   0.879   0.879  0.3498 0.5588035
## A:B:E                        1   8.629   8.629  3.4344 0.0740538 .
## A:C:E                        1   0.098   0.098  0.0389 0.8450862
## B:C:E                        1  21.973  21.973  8.7452 0.0061167 **
## A:D:E                        1   0.191   0.191  0.0762 0.7844988
## B:D:E                        1   1.129   1.129  0.4493 0.5079655
## C:D:E                        1   0.473   0.473  0.1881 0.6676956
## A:B:C:D                      1   0.098   0.098  0.0389 0.8450862
## A:B:C:E                      1   4.254   4.254  1.6931 0.2034391
## A:B:D:E                      1   4.133   4.133  1.6449 0.2098139
## A:C:D:E                      1   0.191   0.191  0.0762 0.7844988
## B:C:D:E                      1   1.723   1.723  0.6856 0.4144184
## A:B:C:D:E                    1   1.531   1.531  0.6094 0.4413226
## Residuals                   29  72.863   2.513
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
(sigma2 = anova(m_fit_5)[33,3])
```

```
## [1] 2.512527
```

```r
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.3962738
```

```r
print(paste0("standard deviation using ANOVA: ", sd_user))
```

```
## [1] "standard deviation using ANOVA: 0.396273811560199"
```

```r
(effects_user = 2 * m_fit_5$coefficients)
```

```
##                  (Intercept)                            A
##                     9.750000                     1.046875
##                            B                            C
##                     0.640625                     2.578125
##                            D                            E
##                    -0.609375                     0.953125
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                     2.187500                    -3.218750
## as.factor(block_variable_)4                          A:B
##                     3.468750                     0.328125
##                          A:C                          B:C
##                     0.890625                     0.109375
##                          A:D                          B:D
##                     0.328125                    -0.078125
##                          C:D                          A:E
##                     0.234375                     0.265625
##                          B:E                          C:E
##                     0.171875                    -0.515625
##                          D:E                        A:B:C
##                    -0.453125                    -0.328125
##                        A:B:D                        A:C:D
##                    -0.015625                    -0.078125
##                        B:C:D                        A:B:E
##                    -0.234375                     0.734375
##                        A:C:E                        B:C:E
##                    -0.078125                    -1.171875
##                        A:D:E                        B:D:E
##                     0.109375                     0.265625
##                        C:D:E                      A:B:C:D
##                    -0.171875                     0.078125
##                      A:B:C:E                      A:B:D:E
##                     0.515625                    -0.718750
##                      A:C:D:E                      B:C:D:E
##                    -0.109375                    -0.328125
##                    A:B:C:D:E
##                    -0.437500
```

```r
(multiplier1 = qt(0.1/2/31,df = 64-31-1-1,lower.tail = F))
```

```
## [1] 3.192655
```

```r
######################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci = cbind(effects_user - multiplier2 * sd_user, effects_user + multiplier2 * sd_user)
colnames(sim_ci) <- c('Lower Bound','Upper Bound')
sim_ci
```

```
##                             Lower Bound Upper Bound
## (Intercept)                   8.5029263 10.99707368
## A                            -0.2001987  2.29394868
```

```
## B                            -0.6064487  1.88769868
## C                             1.3310513  3.82519868
## D                            -1.8564487  0.63769868
## E                            -0.2939487  2.20019868
## as.factor(block_variable_)2   0.9404263  3.43457368
## as.factor(block_variable_)3  -4.4658237 -1.97167632
## as.factor(block_variable_)4   2.2216763  4.71582368
## A:B                          -0.9189487  1.57519868
## A:C                          -0.3564487  2.13769868
## B:C                          -1.1376987  1.35644868
## A:D                          -0.9189487  1.57519868
## B:D                          -1.3251987  1.16894868
## C:D                          -1.0126987  1.48144868
## A:E                          -0.9814487  1.51269868
## B:E                          -1.0751987  1.41894868
## C:E                          -1.7626987  0.73144868
## D:E                          -1.7001987  0.79394868
## A:B:C                        -1.5751987  0.91894868
## A:B:D                        -1.2626987  1.23144868
## A:C:D                        -1.3251987  1.16894868
## B:C:D                        -1.4814487  1.01269868
## A:B:E                        -0.5126987  1.98144868
## A:C:E                        -1.3251987  1.16894868
## B:C:E                        -2.4189487  0.07519868
## A:D:E                        -1.1376987  1.35644868
## B:D:E                        -0.9814487  1.51269868
## C:D:E                        -1.4189487  1.07519868
## A:B:C:D                      -1.1689487  1.32519868
## A:B:C:E                      -0.7314487  1.76269868
## A:B:D:E                      -1.9658237  0.52832368
## A:C:D:E                      -1.3564487  1.13769868
## B:C:D:E                      -1.5751987  0.91894868
## A:B:C:D:E                    -1.6845737  0.80957368
```

```
#####################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci2 = cbind(effects_user - multiplier1 * sd_user, effects_user + multiplier1 * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
sim_ci2
```

```
##                             Lower Bound Upper Bound
## (Intercept)                   8.4848343 11.01516574
## A                            -0.2182907  2.31204074
## B                            -0.6245407  1.90579074
## C                             1.3129593  3.84329074
## D                            -1.8745407  0.65579074
## E                            -0.3120407  2.21829074
## as.factor(block_variable_)2   0.9223343  3.45266574
## as.factor(block_variable_)3  -4.4839157 -1.95358426
## as.factor(block_variable_)4   2.2035843  4.73391574
## A:B                          -0.9370407  1.59329074
## A:C                          -0.3745407  2.15579074
## B:C                          -1.1557907  1.37454074
## A:D                          -0.9370407  1.59329074
## B:D                          -1.3432907  1.18704074
```

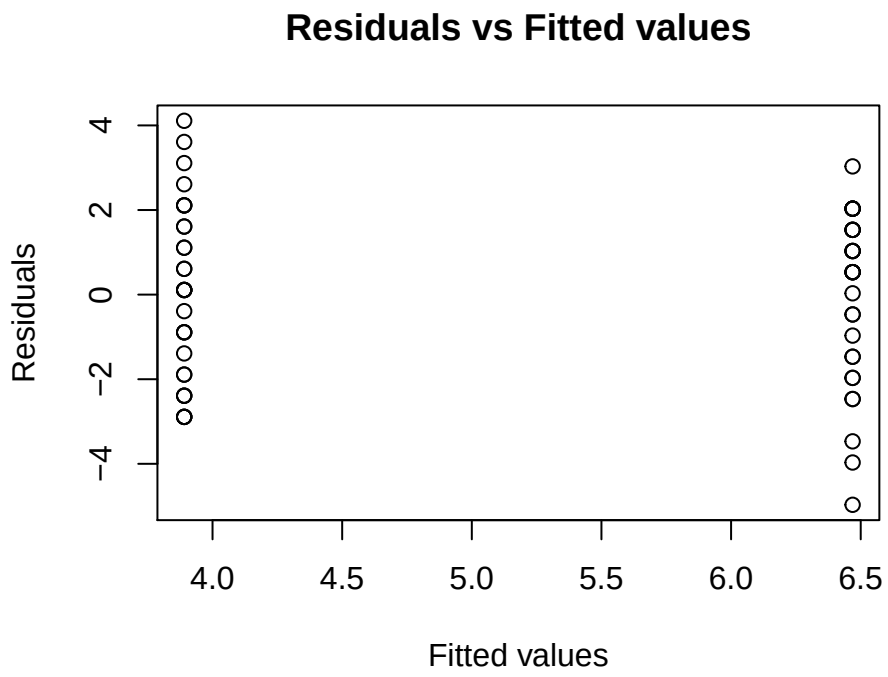
```
## C:D                          -1.0307907  1.49954074
## A:E                          -0.9995407  1.53079074
## B:E                          -1.0932907  1.43704074
## C:E                          -1.7807907  0.74954074
## D:E                          -1.7182907  0.81204074
## A:B:C                        -1.5932907  0.93704074
## A:B:D                        -1.2807907  1.24954074
## A:C:D                        -1.3432907  1.18704074
## B:C:D                        -1.4995407  1.03079074
## A:B:E                        -0.5307907  1.99954074
## A:C:E                        -1.3432907  1.18704074
## B:C:E                        -2.4370407  0.09329074
## A:D:E                        -1.1557907  1.37454074
## B:D:E                        -0.9995407  1.53079074
## C:D:E                        -1.4370407  1.09329074
## A:B:C:D                      -1.1870407  1.34329074
## A:B:C:E                      -0.7495407  1.78079074
## A:B:D:E                      -1.9839157  0.54641574
## A:C:D:E                      -1.3745407  1.15579074
## B:C:D:E                      -1.5932907  0.93704074
## A:B:C:D:E                    -1.7026657  0.82766574
```

SECTION A.5.2.3 :FITTING THE MODEL AND DIAGNOSTIC PLOTS

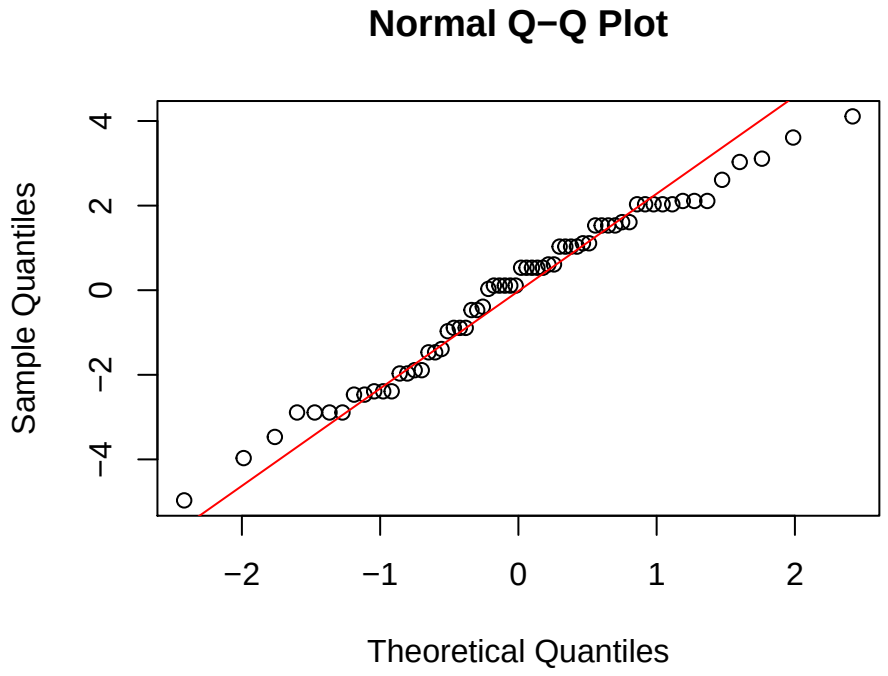
```
#####################

m_fit_6=lm(y~C,data=full_data)
summary_fullmodel_2 = summary(m_fit_6)
2 * m_fit_6$coefficients
```

```
## (Intercept)           C
##   10.359375    2.578125
```

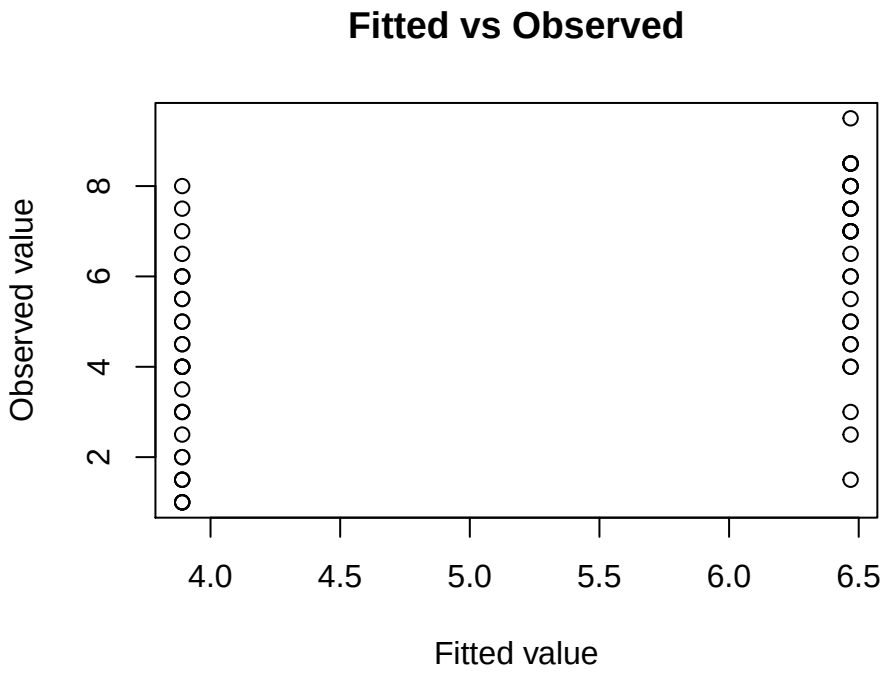
```
resid = m_fit_6$resid
fitted = m_fit_6$fit
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs Fitted values")
```



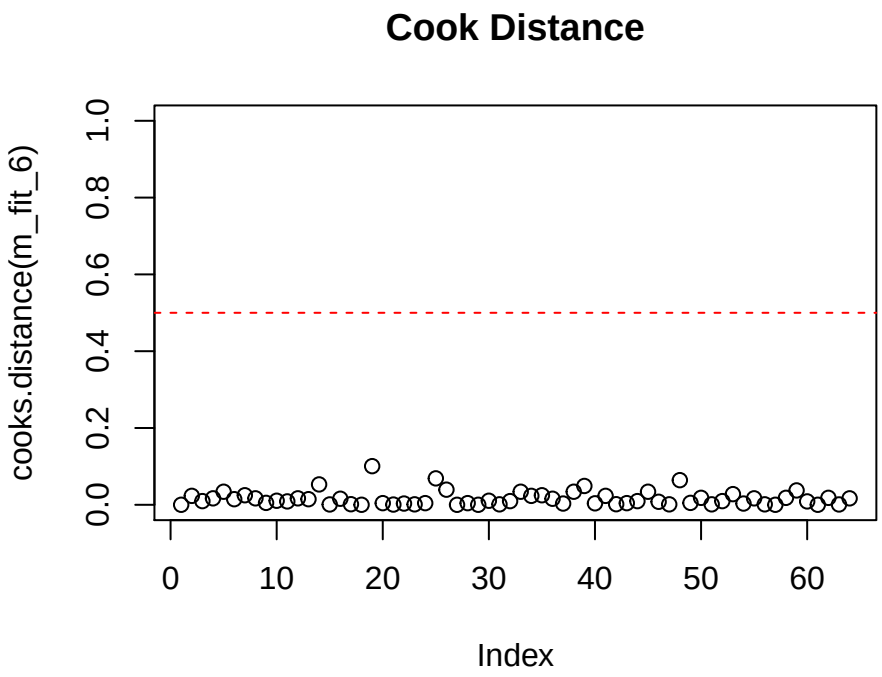

```
plot.new()
qqnorm(resid)
qqline(resid,col="red")
```



```
plot(fitted,full_data$y,xlab="Fitted value",ylab="Observed value",main="Fitted vs Observed")
```




```
group = as.factor(c(rep(c(1,2),8),rep(c(3,4),8)))
plot(cooks.distance(m_fit_6), ylim =c(0,1)  ,main = 'Cook Distance')
abline(h = 0.5,lty = 2, col = 'red')
```



SECTION A.5.2.4 : STUDYING THE EFFECT OF TRANSFORMATIONS

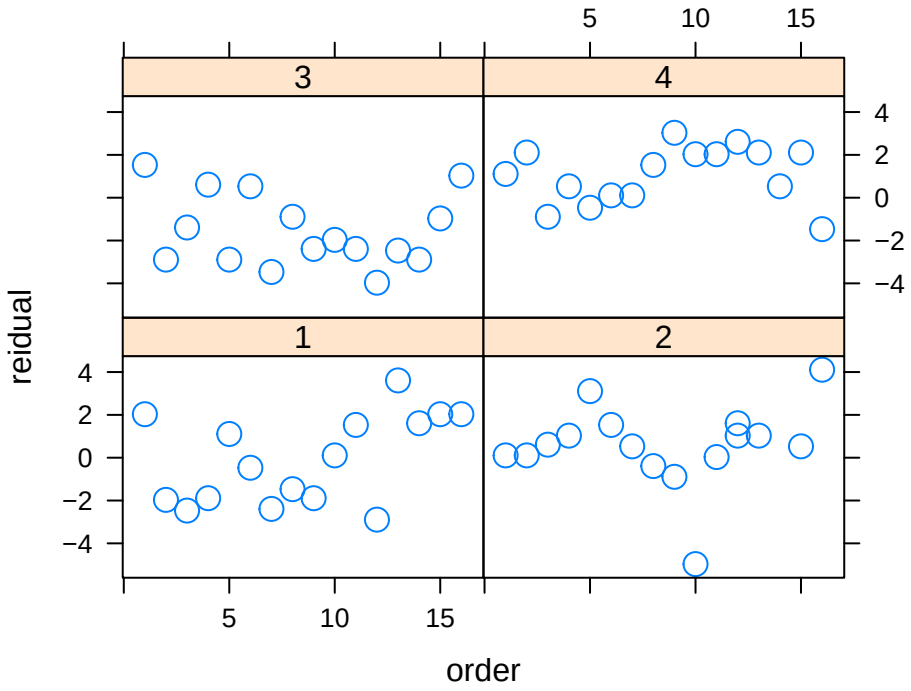
```
######################
A= full_data$A
B= full_data$B
C= full_data$C
D= full_data$D
E= full_data$E
y = full_data$y

y0=full_data$y
gm <- exp(mean(log(y0))) ## geometric mean
lambda.seq <- seq(from=-2,to=3,length.out=20)
ssr=function(lambda){
  if(lambda == 0){
    y <- gm*log(y0)
  } else {
    y <- (y0^lambda-1)/(lambda * gm^{lambda-1})
  }
  fit <- lm(y~ A+B+C+D+E+as.factor(block_variable_))
  return(sum(fit$resid^2))
}

op=optimize(ssr,interval=c(-1,3))
lambda=op$minimum
lambda
```

```
## [1] 0.9993003
```

```
#lambda close to 1 so no use of BOX COX TRANSFORMATION
```

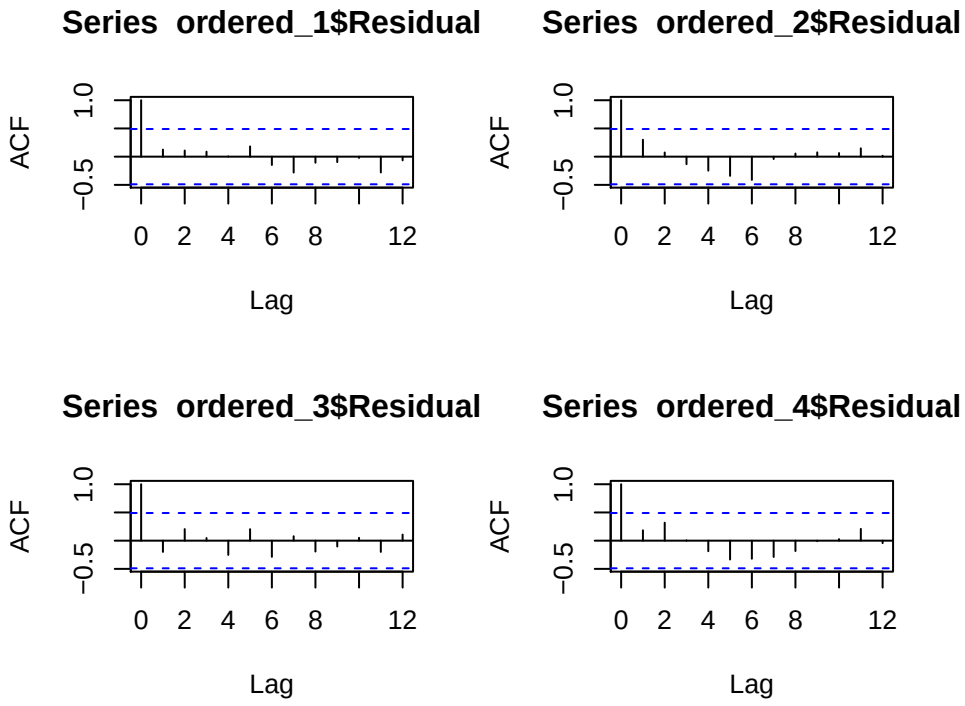
SECTION A.5.2.5 : STUDYING IF THERE IS ANY TREND

```
################ PLOT TO SEE IF THERE IS TREND IN RESIDUALS ################

order_= c(files_1$O_KJ, files_2$O_KJ)
ord_data_ = data.frame(order = order_, reidual =resid, days= block_variable_)


library(lattice)
xyplot(reidual ~ order | as.factor(block_variable_), data = ord_data_,
       auto.key = list(corner = c(0, .98)), cex = 1.5)
```




```
ordered1 = ord_data_[ord_data_$days==1,]
colnames(ordered1) <- c("Ordering", "Residual","Days")
ordered_1 = ordered1[order(ordered1$Ordering),]

ordered2 = ord_data_[ord_data_$days==2,]
colnames(ordered2) <- c("Ordering", "Residual","Days")
ordered_2 = ordered2[order(ordered2$Ordering),]

ordered3 = ord_data_[ord_data_$days==3,]
colnames(ordered3) <- c("Ordering", "Residual","Days")
ordered_3 = ordered3[order(ordered3$Ordering),]

ordered4 = ord_data_[ord_data_$days==4,]
```

```
colnames(ordered4) <- c("Ordering", "Residual","Days")
ordered_4 = ordered4[order(ordered4$Ordering),]

par(mfrow=c(2,2))
acf(ordered_1$Residual)
acf(ordered_2$Residual)
acf(ordered_3$Residual)
acf(ordered_4$Residual)
```



ANALYSIS_JUDGE_AG_DIFF_AD

Abhijeet Bhardwaj

4/28/2021

SECTION A.6: ANALYSIS OF SCORES BY TAKING DIFFERENCE BETWEEN JUDGE AG AND AD

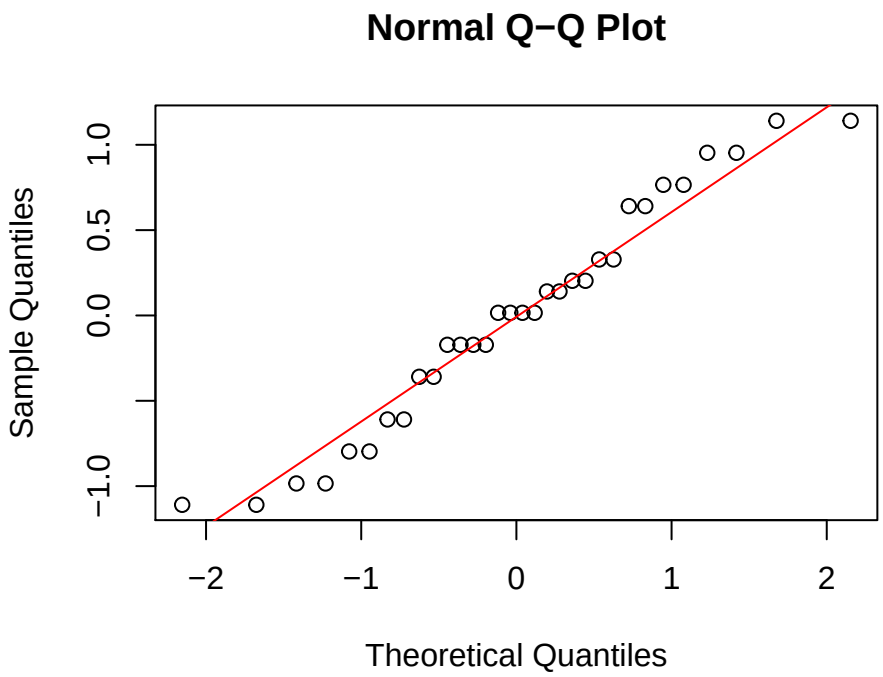
SECTION A.6.1 :Analysis of first run by differece

#SECTION A.6.1.1 DATA VISUALIZATION

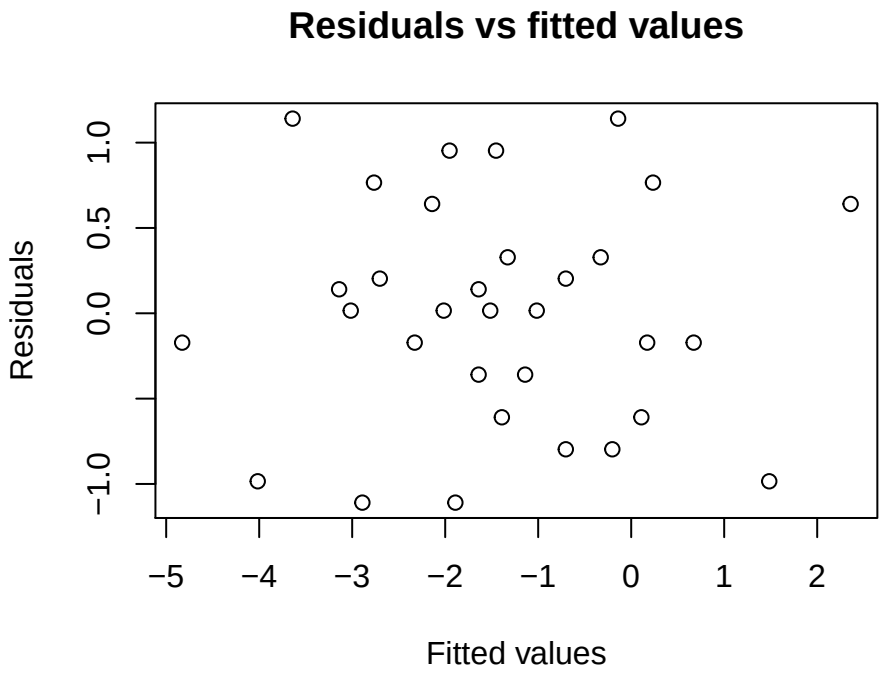
```
setwd("C:\\Users\\abhij\\Desktop\\project")
# Reading data for first replicate # Note blocking on ABCDE
files_1 = read.csv('Run_1_with_scores.csv')

#Avg score calculation
#files_1$J_AVG = (files_1$J_AD+files_1$J_AG+files_1$J_KJ)/3
A = files_1$A
B = files_1$B
C = files_1$C
D = files_1$D
E = files_1$E
block = files_1$Block_ABCDE
avg_score = files_1$J_AG-files_1$J_AD
subset_dat_ = data.frame(y = avg_score,A=A,B=B,C=C,D=D,E=E)

m_fit_1 = lm(y~.^3+block,data=subset_dat_)
```

```
## Warning in terms.formula(formula, data = data): 'varlist' has changed (from
## nvar=6) to new 7 after EncodeVars() -- should no longer happen!
```

```
resid = m_fit_1$resid
fitted = m_fit_1$fit
qqnorm(resid)
qqline(resid,col="red")
```




```
plot(resid~fitted,xlab="Fitted values",ylab="Residuals",main="Residuals vs fitted values")
```



SECTION A.6.1.2 ANALYZING SIGNIFICANCE USING CONFIDENCE INTERVALS

```
# ASSUMING 4 FI's TO BE ZERO but not ABCDE as it is a block effect for sigma
#fitting complete model to get estimates of 4fis
m_fit_2 = lm(y~.^5,data=subset_dat_)
summary_fullmodel_ = summary(m_fit_2)
2*summary_fullmodel_$coefficients
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.84375        NaN     NaN      NaN
## A            0.40625        NaN     NaN      NaN
## B           -0.09375        NaN     NaN      NaN
## C            0.84375        NaN     NaN      NaN
## D            0.40625        NaN     NaN      NaN
## E            1.09375        NaN     NaN      NaN
## A:B         -0.46875        NaN     NaN      NaN
## A:C         -0.15625        NaN     NaN      NaN
## A:D         -0.09375        NaN     NaN      NaN
## A:E          0.21875        NaN     NaN      NaN
## B:C          0.46875        NaN     NaN      NaN
## B:D          0.53125        NaN     NaN      NaN
## B:E         -1.03125        NaN     NaN      NaN
## C:D         -0.03125        NaN     NaN      NaN
## C:E          0.03125        NaN     NaN      NaN
## D:E         -0.15625        NaN     NaN      NaN
## A:B:C       -0.40625        NaN     NaN      NaN
## A:B:D        0.40625        NaN     NaN      NaN
## A:B:E        0.21875        NaN     NaN      NaN
## A:C:D       -0.28125        NaN     NaN      NaN
## A:C:E       -1.09375        NaN     NaN      NaN
## A:D:E       -0.03125        NaN     NaN      NaN
## B:C:D       -0.53125        NaN     NaN      NaN
## B:C:E        0.28125        NaN     NaN      NaN
## B:D:E        0.59375        NaN     NaN      NaN
## C:D:E        0.40625        NaN     NaN      NaN
## A:B:C:D      0.34375        NaN     NaN      NaN
## A:B:C:E      0.03125        NaN     NaN      NaN
## A:B:D:E     -0.15625        NaN     NaN      NaN
## A:C:D:E      0.78125        NaN     NaN      NaN
## B:C:D:E     -0.96875        NaN     NaN      NaN
## A:B:C:D:E   -1.71875        NaN     NaN      NaN
```

```
coeffs_4fis = 2*summary_fullmodel_$coefficients[27:31,1]
sd_4 = sqrt(sum(coeffs_4fis^2)/5)
threshold_4 = sd_4*qt(0.1/(2*25),5,lower.tail = FALSE)
all_coeffs_ = 2*summary_fullmodel_$coefficients[,1]
eff = as.numeric(all_coeffs_)
which(abs(eff)>threshold_4) # only intercept
```

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

```
# ASSUMING 3 and 4 FI's TO BE ZERO but not ABCDE as it is a block effect for estimating sigma
#fitting complete model to get estimates of 4fis
coeffs_34fis = 2*summary_fullmodel_$coefficients[17:31,1]
```

```
var_34 = sum(coeffs_34fis^2)/15
sd_34 = sqrt(var_34)
threshold_34 = sd_4*qt(0.1/(2*15),15,lower.tail = FALSE)
threshold_34
```

```
## [1] 1.830045
```

```
which(abs(eff)>threshold_34) #
```

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

SECTION A.6.1.3 ANALYZING SIGNIFICANCE USING DANIEL METHOD

```
## APPLY DANIEL METHOD
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

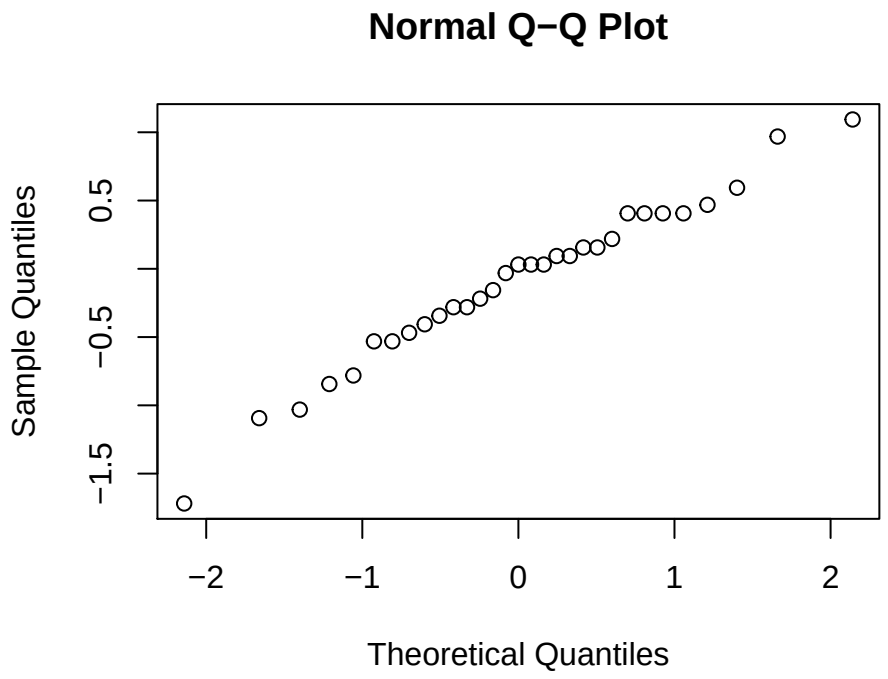
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
y= avg_score
k = 5 # input the # of main effects in your design
coef_user <- function(num){ # generate the coefficient
  x <- as.integer(rev(intToBits(num))) %>%
    paste(collapse = "") %>%
    substr(.,start = 32-k + 1,stop = 32) #
  res = sapply(1:k,function(t) as.numeric(substr(x,t,t))*2-1)
  return(res)
}
perm = sapply(0:(2^k-1),coef_user)


median_abs = numeric(32)
for(i in 1:32){
  A_tmp = perm[1,i] * A
  B_tmp = perm[2,i] * B
  C_tmp = perm[3,i] * C
  D_tmp = perm[4,i] * D
  E_tmp = perm[5,i] * E
  dat_tmp = data.frame(y,A_tmp,B_tmp,C_tmp,D_tmp,E_tmp)
  fit_tmp = lm(y~(.)^5,data = dat_tmp)
  # summary(fit_tmp)
  median_abs[i] = abs(median(fit_tmp$coefficients))
}
which.min(median_abs) # 3
```

```
## [1] 3
```

```r
A_final = perm[1,3] * A
B_final = perm[2,3] * B
C_final = perm[3,3] * C
D_final = perm[4,3] * D
E_final = perm[5,3] * E
dat_final = data.frame(y,A_final,B_final,C_final,D_final,E_final)
fit_final = lm(y~(.)^5,data = dat_final)
(qq_user = qqnorm(2 * fit_final$coefficients[-1]))  # A_final:C_final; C_final ; E_final
```




```
## $x
##  [1] -0.60017878  0.32929135 -1.21123213  1.05741423 -1.66069761 -0.70009021
##  [7] -0.16242937  0.24500622  0.60017878  1.21123213 -0.92524456 -1.40074506
## [13]  0.16242937  0.00000000  0.50593365  0.80754104  0.92524456 -0.24500622
## [19] -0.32929135  2.14119812 -0.08094729 -0.80754104 -0.41598722  1.40074506
## [25]  0.70009021 -0.50593365  0.08094729  0.41598722 -1.05741423  1.66069761
## [31] -2.14119812
##
## $y
##                          A_final                          B_final
##                         -0.40625                          0.09375
##                          C_final                          D_final
##                         -0.84375                          0.40625
##                          E_final                 A_final:B_final
##                         -1.09375                         -0.46875
##                 A_final:C_final                 A_final:D_final
##                         -0.15625                          0.09375
##                 A_final:E_final                 B_final:C_final
##                          0.21875                          0.46875
##                 B_final:D_final                 B_final:E_final
```

```
##                                -0.53125                                -1.03125
##                          C_final:D_final                          C_final:E_final
##                                 0.03125                                 0.03125
##                          D_final:E_final                  A_final:B_final:C_final
##                                 0.15625                                 0.40625
##                  A_final:B_final:D_final                  A_final:B_final:E_final
##                                 0.40625                                -0.21875
##                  A_final:C_final:D_final                  A_final:C_final:E_final
##                                -0.28125                                 1.09375
##                  A_final:D_final:E_final                  B_final:C_final:D_final
##                                -0.03125                                -0.53125
##                  B_final:C_final:E_final                  B_final:D_final:E_final
##                                -0.28125                                 0.59375
##                  C_final:D_final:E_final          A_final:B_final:C_final:D_final
##                                 0.40625                                -0.34375
##          A_final:B_final:C_final:E_final          A_final:B_final:D_final:E_final
##                                 0.03125                                 0.15625
##          A_final:C_final:D_final:E_final          B_final:C_final:D_final:E_final
##                                -0.78125                                 0.96875
## A_final:B_final:C_final:D_final:E_final
##                                -1.71875
```

```
# Step2: Fit a lm ------------------------------------------------------------
qq_x = qq_user$x
qq_y = qq_user$y
fit1 = lm(qq_y~qq_x)
beta1 = fit1$coefficients[2]
# Step3: Remove the outlier --------------------------------------------------
L = quantile(qq_y,0.25) - 1.5 * IQR(qq_y) # left bound
U = quantile(qq_y,0.75) + 1.5 * IQR(qq_y) # right bound
CC = (U-L)/2

which(abs(qq_y) > CC) # ONLY BLOCK EFFECT IS SIGNIFICANT A_final:B_final:C_final:D_final:E_final
```

```
## A_final:B_final:C_final:D_final:E_final
##                                     31
```

```
x_inlier = qq_x[which(abs(qq_y) <= CC)]
y_inlier = qq_y[which(abs(qq_y) <= CC)]
# Step4: Fit another lm ------------------------------------------------------
fit2 = lm(y_inlier ~ x_inlier)
beta2 = fit2$coefficients[2]
beta1 / beta2 # 1.186
```

```
##    qq_x
## 1.05153
```

```
# From the table in the paper,
# n = 31, alpha = 0.05 -> threshold = 1.108
# So we have the significance here

m = length(x_inlier)
y_mean = mean(y_inlier)
x_mean = mean(x_inlier)
n_prime = round(31/4,1)
sigma_user = sqrt(sum((fit2$residuals)^2)/fit2$df.residual)
```

```
LHS = abs(y_inlier - y_mean + beta2 * (x_inlier - x_mean))
RHS = n_prime * sqrt(qf(p = 0.975,df1 = n_prime,df2 = m-2)) * sigma_user *
  sqrt(1 + 1/m + (x_inlier - x_mean)^2 / (var(x_inlier) * (m+1) ))
final_idx = which(LHS <= RHS)
D_ = max(abs(y_inlier[final_idx]))
c(D_, CC)
```

```
##             75%
## 0.46875 1.50000
```

```
final_threshold = max(CC,D_)
print(paste0("after daniels method :", names(which(abs(qq_y) > final_threshold)))) # again only C_final
```

```
## [1] "after daniels method :A_final:B_final:C_final:D_final:E_final"
```

SECTION A.6.1.4 ANALYZING SIGNIFICANCE USING DONG's METHOD

```
##############################################################################
#ANALYSIS BY DONGS METHOD
coeff=coef(m_fit_2)*2
theta1=abs(na.omit(coeff[-c(1,31)]))
s0=1.5*median(theta1)
m1=sum(theta1<=2.5*s0)
s1=sqrt(sum(theta1[theta1<=2.5*s0]^2)/m1)
m2=sum(theta1<=2.5*s1)
s2=sqrt(sum(theta1[theta1<=2.5*s1]^2)/m2)
g=length(theta1)
gamma=0.5*(1-(1-0.05)^(1/g))
print(paste0("after DONG's method significant effect :", names(theta1[theta1>s2*qt(gamma,m2,lower.tail =
```

```
## [1] "after DONG's method significant effect :"
```

```
##############################################
# none of the factors are significant
```

SECTION A.6.2 :Analysis of Second run and combined two runs by difference

SECTION A.6.2.1 :Estimates from second run

```
# COMBINING THE SECOND COMPLETE REPLICATE

files_2 = read.csv('Run_2_with_scores.csv')
#files_2$J_AVG = (files_2$J_AD+files_2$J_AG+files_2$J_KJ)/3
A2 = files_2$A
B2 = files_2$B
C2 = files_2$C
D2 = files_2$D
E2 = files_2$E
avg_score2 = files_2$J_AG - files_2$J_AD
```

```
subset_dat_2 = data.frame(y = avg_score2,A=A2,B=B2,C=C2,D=D2,E=E2)
m_fit_4 = lm(y~.^5,data=subset_dat_2)
summary_fullmodel_2 = summary(m_fit_4)
2*summary_fullmodel_2$coefficients
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.312500e+00        NaN     NaN      NaN
## A           -3.125000e-01        NaN     NaN      NaN
## B            1.875000e-01        NaN     NaN      NaN
## C            3.125000e-01        NaN     NaN      NaN
## D            5.000000e-01        NaN     NaN      NaN
## E            2.500000e-01        NaN     NaN      NaN
## A:B          8.125000e-01        NaN     NaN      NaN
## A:C          8.125000e-01        NaN     NaN      NaN
## A:D          2.502335e-16        NaN     NaN      NaN
## A:E         -8.750000e-01        NaN     NaN      NaN
## B:C          6.250000e-02        NaN     NaN      NaN
## B:D         -5.000000e-01        NaN     NaN      NaN
## B:E         -6.250000e-01        NaN     NaN      NaN
## C:D         -1.000000e+00        NaN     NaN      NaN
## C:E          1.375000e+00        NaN     NaN      NaN
## D:E          1.875000e-01        NaN     NaN      NaN
## A:B:C       -8.125000e-01        NaN     NaN      NaN
## A:B:D        1.250000e-01        NaN     NaN      NaN
## A:B:E        3.750000e-01        NaN     NaN      NaN
## A:C:D       -2.453269e-17        NaN     NaN      NaN
## A:C:E        7.500000e-01        NaN     NaN      NaN
## A:D:E       -1.875000e-01        NaN     NaN      NaN
## B:C:D        5.000000e-01        NaN     NaN      NaN
## B:C:E        2.500000e-01        NaN     NaN      NaN
## B:D:E       -1.187500e+00        NaN     NaN      NaN
## C:D:E       -5.625000e-01        NaN     NaN      NaN
## A:B:C:D      6.250000e-01        NaN     NaN      NaN
## A:B:C:E     -7.500000e-01        NaN     NaN      NaN
## A:B:D:E     -9.375000e-01        NaN     NaN      NaN
## A:C:D:E      6.250000e-02        NaN     NaN      NaN
## B:C:D:E      1.062500e+00        NaN     NaN      NaN
## A:B:C:D:E    8.125000e-01        NaN     NaN      NaN
```

SECTION A.6.2.2 :Calculating Confidence Intervals by combining the replicates

```
# CACULATION OF VARIANCE BY FIRST PRINCIPLES
coeffs_1st_rep = 2*summary_fullmodel_$coefficients[-1,1]
coeffs_2st_rep = 2*summary_fullmodel_2$coefficients[-1,1]
avg_coeff_ = (coeffs_1st_rep+coeffs_2st_rep)/2
diff_1 = (coeffs_1st_rep-avg_coeff_)^2
diff_2 = (coeffs_2st_rep-avg_coeff_)^2
diff_1_ = diff_1[-c(31,28)]
diff_2_ = diff_2[-c(31,28)]

var_ = (diff_1_+diff_2_)/(2-1)
```

```r
var_hat_ = sum(var_)/29 #estimate of 4sigma^2/2^k) \hat{var(Ai)}
multiplier1 = qt(0.1/2/31,df = 64-31-3-1,lower.tail = F)
boundary_pt =   multiplier1*sqrt(var_hat_/2)
print(paste0("standard deviation using first principle: ", sqrt(var_hat_/2)))
```

```
## [1] "standard deviation using first principle: 0.454090084034108"
```

```r
avg_coeff_up = avg_coeff_+boundary_pt
avg_coeff_dwn = avg_coeff_-boundary_pt
CI_interval = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn, Up_=

######################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval
```

```
##           variable_  estimate       Lwr_       Up_
## A                 A  0.046875 -1.4112432 1.5049932
## B                 B  0.046875 -1.4112432 1.5049932
## C                 C  0.578125 -0.8799932 2.0362432
## D                 D  0.453125 -1.0049932 1.9112432
## E                 E  0.671875 -0.7862432 2.1299932
## A:B             A:B  0.171875 -1.2862432 1.6299932
## A:C             A:C  0.328125 -1.1299932 1.7862432
## A:D             A:D -0.046875 -1.5049932 1.4112432
## A:E             A:E -0.328125 -1.7862432 1.1299932
## B:C             B:C  0.265625 -1.1924932 1.7237432
## B:D             B:D  0.015625 -1.4424932 1.4737432
## B:E             B:E -0.828125 -2.2862432 0.6299932
## C:D             C:D -0.515625 -1.9737432 0.9424932
## C:E             C:E  0.703125 -0.7549932 2.1612432
## D:E             D:E  0.015625 -1.4424932 1.4737432
## A:B:C         A:B:C -0.609375 -2.0674932 0.8487432
## A:B:D         A:B:D  0.265625 -1.1924932 1.7237432
## A:B:E         A:B:E  0.296875 -1.1612432 1.7549932
## A:C:D         A:C:D -0.140625 -1.5987432 1.3174932
## A:C:E         A:C:E -0.171875 -1.6299932 1.2862432
## A:D:E         A:D:E -0.109375 -1.5674932 1.3487432
## B:C:D         B:C:D -0.015625 -1.4737432 1.4424932
## B:C:E         B:C:E  0.265625 -1.1924932 1.7237432
## B:D:E         B:D:E -0.296875 -1.7549932 1.1612432
## C:D:E         C:D:E -0.078125 -1.5362432 1.3799932
## A:B:C:D     A:B:C:D  0.484375 -0.9737432 1.9424932
## A:B:C:E     A:B:C:E -0.359375 -1.8174932 1.0987432
## A:B:D:E     A:B:D:E -0.546875 -2.0049932 0.9112432
## A:C:D:E     A:C:D:E  0.421875 -1.0362432 1.8799932
## B:C:D:E     B:C:D:E  0.046875 -1.4112432 1.5049932
## A:B:C:D:E A:B:C:D:E -0.453125 -1.9112432 1.0049932
```

```r
# USING STUDENTIZED MAX MODULUS
smm_user <- function(colnum, rownum, m){
  r1 = weighted.mean(m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r2 = 1 / weighted.mean(1 / m[,1], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r3 = weighted.mean(m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  r4 = 1 / weighted.mean(1 / m[,2], w = c(rownum[3] - rownum[2], rownum[2] - rownum[1]))
  rl = mean(r1,r2)
```

```
  rr = mean(r3,r4)
  c1 = weighted.mean(c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  c2 = 1 / weighted.mean(1 / c(rl,rr), w = c(colnum[3] - colnum[2], colnum[2] - colnum[1]))
  return(round(mean(c1,c2),3))
}
#g=31 dof = 64-31-3-1 #
(multiplier2 = smm_user(colnum = c(24,29,30),
                        rownum = c(28,31,36),
                        m = matrix(c(3.156,3.251,3.104,3.196),2,2))) # g =31, v = 64-31-3-1
```

```
## [1] 3.147
```

```
boundary_pt2 =   multiplier2*sqrt(var_hat_/2)
avg_coeff_up2 = avg_coeff_+boundary_pt2
avg_coeff_dwn2 = avg_coeff_-boundary_pt2
CI_interval2 = data.frame(variable_ =names(avg_coeff_), estimate = avg_coeff_, Lwr_ =avg_coeff_dwn2, Up
#####################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 1 (1sr Principles)
CI_interval2
```

```
##           variable_  estimate       Lwr_       Up_
## A                 A  0.046875 -1.3821465 1.4758965
## B                 B  0.046875 -1.3821465 1.4758965
## C                 C  0.578125 -0.8508965 2.0071465
## D                 D  0.453125 -0.9758965 1.8821465
## E                 E  0.671875 -0.7571465 2.1008965
## A:B             A:B  0.171875 -1.2571465 1.6008965
## A:C             A:C  0.328125 -1.1008965 1.7571465
## A:D             A:D -0.046875 -1.4758965 1.3821465
## A:E             A:E -0.328125 -1.7571465 1.1008965
## B:C             B:C  0.265625 -1.1633965 1.6946465
## B:D             B:D  0.015625 -1.4133965 1.4446465
## B:E             B:E -0.828125 -2.2571465 0.6008965
## C:D             C:D -0.515625 -1.9446465 0.9133965
## C:E             C:E  0.703125 -0.7258965 2.1321465
## D:E             D:E  0.015625 -1.4133965 1.4446465
## A:B:C         A:B:C -0.609375 -2.0383965 0.8196465
## A:B:D         A:B:D  0.265625 -1.1633965 1.6946465
## A:B:E         A:B:E  0.296875 -1.1321465 1.7258965
## A:C:D         A:C:D -0.140625 -1.5696465 1.2883965
## A:C:E         A:C:E -0.171875 -1.6008965 1.2571465
## A:D:E         A:D:E -0.109375 -1.5383965 1.3196465
## B:C:D         B:C:D -0.015625 -1.4446465 1.4133965
## B:C:E         B:C:E  0.265625 -1.1633965 1.6946465
## B:D:E         B:D:E -0.296875 -1.7258965 1.1321465
## C:D:E         C:D:E -0.078125 -1.5071465 1.3508965
## A:B:C:D     A:B:C:D  0.484375 -0.9446465 1.9133965
## A:B:C:E     A:B:C:E -0.359375 -1.7883965 1.0696465
## A:B:D:E     A:B:D:E -0.546875 -1.9758965 0.8821465
## A:C:D:E     A:C:D:E  0.421875 -1.0071465 1.8508965
## B:C:D:E     B:C:D:E  0.046875 -1.3821465 1.4758965
## A:B:C:D:E A:B:C:D:E -0.453125 -1.8821465 0.9758965
```

```
# NONE OF THE DIFFERENCE EFFECT IS SIGNIFICANT
```

```r
# calculating variance with anova
# Model with c only analysis
subset_dat_$block =files_1$Block_ABCDE
subset_dat_2$block =files_2$ABED
append_data_ = rbind(subset_dat_,subset_dat_2)
block_variable_ = c(rep(1,16),rep(2,16),rep(3,16),rep(4,16))
append_data_$final_block = block_variable_

A3 = append_data_$A
B3 = append_data_$B
C3 = append_data_$C
D3 = append_data_$D
E3 = append_data_$E
Y3 = append_data_$y
full_data = data.frame(y = Y3,A=A3,B=B3,C=C3,D=D3,E=E3)

m_fit_5 = lm(y~A*B*C*D*E+as.factor(block_variable_),data=full_data)
#summary(m_fit_3)
anova(m_fit_5)
```

```
## Analysis of Variance Table
##
## Response: y
##                              Df Sum Sq Mean Sq F value  Pr(>F)
## A                             1  0.035  0.0352  0.0107 0.91849
## B                             1  0.035  0.0352  0.0107 0.91849
## C                             1  5.348  5.3477  1.6209 0.21308
## D                             1  3.285  3.2852  0.9958 0.32659
## E                             1  7.223  7.2227  2.1892 0.14976
## as.factor(block_variable_)    3 40.043 13.3477  4.0458 0.01613 *
## A:B                           1  0.473  0.4727  0.1433 0.70781
## A:C                           1  1.723  1.7227  0.5221 0.47571
## B:C                           1  1.129  1.1289  0.3422 0.56310
## A:D                           1  0.035  0.0352  0.0107 0.91849
## B:D                           1  0.004  0.0039  0.0012 0.97279
## C:D                           1  4.254  4.2539  1.2894 0.26546
## A:E                           1  1.723  1.7227  0.5221 0.47571
## B:E                           1 10.973 10.9727  3.3259 0.07852 .
## C:E                           1  7.910  7.9102  2.3976 0.13236
## D:E                           1  0.004  0.0039  0.0012 0.97279
## A:B:C                         1  5.941  5.9414  1.8009 0.19002
## A:B:D                         1  1.129  1.1289  0.3422 0.56310
## A:C:D                         1  0.316  0.3164  0.0959 0.75901
## B:C:D                         1  0.004  0.0039  0.0012 0.97279
## A:B:E                         1  1.410  1.4102  0.4274 0.51840
## A:C:E                         1  0.473  0.4727  0.1433 0.70781
## B:C:E                         1  1.129  1.1289  0.3422 0.56310
## A:D:E                         1  0.191  0.1914  0.0580 0.81135
## B:D:E                         1  1.410  1.4102  0.4274 0.51840
## C:D:E                         1  0.098  0.0977  0.0296 0.86460
## A:B:C:D                       1  3.754  3.7539  1.1378 0.29491
## A:B:C:E                       1  2.066  2.0664  0.6263 0.43513
## A:B:D:E                       1  0.195  0.1953  0.0592 0.80948
## A:C:D:E                       1  2.848  2.8477  0.8631 0.36053
```

```
## B:C:D:E                    1  0.035  0.0352  0.0107 0.91849
## A:B:C:D:E                  1  5.281  5.2812  1.6008 0.21586
## Residuals                 29 95.676  3.2992
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(sigma2 = anova(m_fit_5)[33,3])
```

```
## [1] 3.299165
```

```
(sd_user = sqrt(4 * sigma2 / 64)) # sd of the effects
```

```
## [1] 0.4540901
```

```
print(paste0("standard deviation using ANOVA: ", sd_user))
```

```
## [1] "standard deviation using ANOVA: 0.454090084034108"
```

```
(effects_user = 2 * m_fit_5$coefficients)
```

```
##                  (Intercept)                            A
##                    -0.312500                     0.046875
##                            B                            C
##                     0.046875                     0.578125
##                            D                            E
##                     0.453125                     0.671875
## as.factor(block_variable_)2 as.factor(block_variable_)3
##                    -5.062500                    -0.218750
## as.factor(block_variable_)4                          A:B
##                    -1.781250                     0.171875
##                          A:C                          B:C
##                     0.328125                     0.265625
##                          A:D                          B:D
##                    -0.046875                     0.015625
##                          C:D                          A:E
##                    -0.515625                    -0.328125
##                          B:E                          C:E
##                    -0.828125                     0.703125
##                          D:E                        A:B:C
##                     0.015625                    -0.609375
##                        A:B:D                        A:C:D
##                     0.265625                    -0.140625
##                        B:C:D                        A:B:E
##                    -0.015625                     0.296875
##                        A:C:E                        B:C:E
##                    -0.171875                     0.265625
##                        A:D:E                        B:D:E
##                    -0.109375                    -0.296875
##                        C:D:E                      A:B:C:D
##                    -0.078125                     0.484375
##                      A:B:C:E                      A:B:D:E
##                    -0.359375                    -0.156250
##                      A:C:D:E                      B:C:D:E
##                     0.421875                     0.046875
##                    A:B:C:D:E
##                     0.812500
```

```r
(multiplier1 = qt(0.1/2/31,df = 64-31-1-1,lower.tail = F))
```

```
## [1] 3.192655
```

```r
#################################################################
# STUDENTIZED MAXIMUM MODULUS CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci = cbind(effects_user - multiplier2 * sd_user, effects_user + multiplier2 * sd_user)
colnames(sim_ci) <- c('Lower Bound','Upper Bound')
sim_ci
```

```
##                              Lower Bound Upper Bound
## (Intercept)                   -1.7415215   1.1165215
## A                             -1.3821465   1.4758965
## B                             -1.3821465   1.4758965
## C                             -0.8508965   2.0071465
## D                             -0.9758965   1.8821465
## E                             -0.7571465   2.1008965
## as.factor(block_variable_)2   -6.4915215  -3.6334785
## as.factor(block_variable_)3   -1.6477715   1.2102715
## as.factor(block_variable_)4   -3.2102715  -0.3522285
## A:B                           -1.2571465   1.6008965
## A:C                           -1.1008965   1.7571465
## B:C                           -1.1633965   1.6946465
## A:D                           -1.4758965   1.3821465
## B:D                           -1.4133965   1.4446465
## C:D                           -1.9446465   0.9133965
## A:E                           -1.7571465   1.1008965
## B:E                           -2.2571465   0.6008965
## C:E                           -0.7258965   2.1321465
## D:E                           -1.4133965   1.4446465
## A:B:C                         -2.0383965   0.8196465
## A:B:D                         -1.1633965   1.6946465
## A:C:D                         -1.5696465   1.2883965
## B:C:D                         -1.4446465   1.4133965
## A:B:E                         -1.1321465   1.7258965
## A:C:E                         -1.6008965   1.2571465
## B:C:E                         -1.1633965   1.6946465
## A:D:E                         -1.5383965   1.3196465
## B:D:E                         -1.7258965   1.1321465
## C:D:E                         -1.5071465   1.3508965
## A:B:C:D                       -0.9446465   1.9133965
## A:B:C:E                       -1.7883965   1.0696465
## A:B:D:E                       -1.5852715   1.2727715
## A:C:D:E                       -1.0071465   1.8508965
## B:C:D:E                       -1.3821465   1.4758965
## A:B:C:D:E                     -0.6165215   2.2415215
```

```r
#################################################################
# BONFERRONI CONFIDENCE INTERVAL USING SIGMA FROM METHOD 2 (ANOVA)
sim_ci2 = cbind(effects_user - multiplier1 * sd_user, effects_user + multiplier1 * sd_user)
colnames(sim_ci2) <- c('Lower Bound','Upper Bound')
sim_ci2
```

```
##                              Lower Bound Upper Bound
## (Intercept)                   -1.7622532   1.1372532
```

```
## A                             -1.4028782   1.4966282
## B                             -1.4028782   1.4966282
## C                             -0.8716282   2.0278782
## D                             -0.9966282   1.9028782
## E                             -0.7778782   2.1216282
## as.factor(block_variable_)2   -6.5122532  -3.6127468
## as.factor(block_variable_)3   -1.6685032   1.2310032
## as.factor(block_variable_)4   -3.2310032  -0.3314968
## A:B                           -1.2778782   1.6216282
## A:C                           -1.1216282   1.7778782
## B:C                           -1.1841282   1.7153782
## A:D                           -1.4966282   1.4028782
## B:D                           -1.4341282   1.4653782
## C:D                           -1.9653782   0.9341282
## A:E                           -1.7778782   1.1216282
## B:E                           -2.2778782   0.6216282
## C:E                           -0.7466282   2.1528782
## D:E                           -1.4341282   1.4653782
## A:B:C                         -2.0591282   0.8403782
## A:B:D                         -1.1841282   1.7153782
## A:C:D                         -1.5903782   1.3091282
## B:C:D                         -1.4653782   1.4341282
## A:B:E                         -1.1528782   1.7466282
## A:C:E                         -1.6216282   1.2778782
## B:C:E                         -1.1841282   1.7153782
## A:D:E                         -1.5591282   1.3403782
## B:D:E                         -1.7466282   1.1528782
## C:D:E                         -1.5278782   1.3716282
## A:B:C:D                       -0.9653782   1.9341282
## A:B:C:E                       -1.8091282   1.0903782
## A:B:D:E                       -1.6060032   1.2935032
## A:C:D:E                       -1.0278782   1.8716282
## B:C:D:E                       -1.4028782   1.4966282
## A:B:C:D:E                     -0.6372532   2.2622532
```

```
######################

#NO DIFFERENCE IS SIGNIFICANT
```

1 Appendix 7

Figure 1: Glass markings for measurement.

Figure 2: Measurement of spice.

Figure 3: Measurement of ginger.

Figure 4: Samples for a single day.