Midterm

Hakob Janesian

2023-10-20

Q1. Analysis error 1

"data_Q1a.txt" and "data_Q1b.txt" are data files of an experiment with three conditions of one factor. These two files represent the same data but in two different formats. However, the following script shows that results of ANOVA are different between the files. Explain why the results are different and revise the script so that the results of ANOVAs become the same.

```r
# Initial code (provided by the instructor)
dQ1a = read.delim("data_Q1a.txt", header = TRUE)
dQ1b = read.delim("data_Q1b.txt", header = TRUE)

print("Data Q1a")
```

```
## [1] "Data Q1a"
```

```r
Anova(lm(Measure ~ Condition, data=dQ1a), type=2)
```

```
## Anova Table (Type II tests)
##
## Response: Measure
##           Sum Sq Df F value  Pr(>F)
## Condition 10.542  2  4.5691 0.01954 *
## Residuals 31.147 27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
print("Data Q1b")
```

```
## [1] "Data Q1b"
```

```r
Anova(lm(Measure ~ Condition, data=dQ1b), type=2)
```

```
## Anova Table (Type II tests)
##
## Response: Measure
##           Sum Sq Df F value Pr(>F)
## Condition  0.350  1   0.237 0.6302
## Residuals 41.339 28
```

Explanation - Based on the above-mentioned results of ANOVA tests, I can clearly see a discrepancy. For the dataset "data_Q1a", the "Condition" factor is statistically significant, having a p-value equal to 0.01954, as it is less than the significance threshold of 0.05. On the other hand, in the dataset "data_Q1b", the "Condition" factor is not statistically significant, having a p-value of 0.6302 (greater than 0.05). So, according to the ANOVA test on "data_Q1a" there is a statistically significant difference in the "Measure" across the conditions, and the ANOVA test on "Data Q1b" does not provide evidence to support a statistically significant difference in the "Measure" across the conditions. After analyzing the data, I found out that in “data_Q1a” the “Condition” column is categorical, and its values are represented as characters (‘A’, ‘B’, ‘C’). On the other hand, in “data_Q1b” the column “Condition” is represented as integer numbers (1, 2, 3). Hence, in the “data_Q1a” the ‘Condition’ variable is correctly identified as a three level factor, having degree_of_freedom = 2 (because 3 - 1 = 2) during the ANOVA. On contrary, in the “data_Q1b” the ‘Condition’ variable is treated as numeric, having a degree_of_freedom = 1. So, the difference in formatting is the root problem of the discrepancy. To fix this problem, we must revise the code for “data_Q1b.txt”, and the column ‘Condition’ must be set as a factor before running the ANOVA test.”

```r
# Revised version ( in "data_Q1b.txt" the column 'Condition' should be set as a factor )
dQ1b$Condition = as.factor(dQ1b$Condition)

# Let me run ANOVAs again
print("Data Q1a")
```

```
## [1] "Data Q1a"
```

```r
Anova(lm(Measure ~ Condition, data=dQ1a), type=2)
```

```
## Anova Table (Type II tests)
##
## Response: Measure
##           Sum Sq Df F value  Pr(>F)
## Condition 10.542  2  4.5691 0.01954 *
## Residuals 31.147 27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
print("Revised Data Q1b")
```

```
## [1] "Revised Data Q1b"
```

```r
Anova(lm(Measure ~ Condition, data=dQ1b), type=2)
```

```
## Anova Table (Type II tests)
##
## Response: Measure
##           Sum Sq Df F value  Pr(>F)
## Condition 10.542  2  4.5691 0.01954 *
## Residuals 31.147 27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Final comment - After the revision, the results of ANOVAs are the same for both datasets.

Q2. Analysis error 2

The scripts “Valid_case” below show that a t-test and an ANOVA give the same results when these stat tests are used to compare data in two conditions in “data_Q2a.txt”. However, this is not the case for the script “Invalid_case” below. Note that “data_Q2a.txt” and “data_Q2b.txt” are the essentially the same data. However, in this script, a t-test and an ANOVA give different results. Explain why the results are different and revise the script so that the results of the t-test and the ANOVA become the same.

```r
dQ2a = read.delim("data_Q2a.txt", header = TRUE)
dQ2a$Subject   = as.factor(dQ2a$Subject)
dQ2a$Condition = as.factor(dQ2a$Condition)

print("t-test")
```

```
## [1] "t-test"
```

```
t.test(dQ2a$Measure[dQ2a$Condition=='A'],
       dQ2a$Measure[dQ2a$Condition=='B'], paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  dQ2a$Measure[dQ2a$Condition == "A"] and dQ2a$Measure[dQ2a$Condition == "B"]
## t = -2.4725, df = 11, p-value = 0.03098
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -13.2312185  -0.7687815
## sample estimates:
## mean of the differences
##                       -7
```

```
print("ANOVA Repeated Measure")
```

```
## [1] "ANOVA Repeated Measure"
```

```
print(ezANOVA(data = dQ2a, dv=.(Measure), wid=.(Subject), within=.(Condition),
              type=2, detailed=FALSE))
```

```
## $ANOVA
##      Effect DFn DFd        F          p p<.05       ges
## 2 Condition   1  11 6.113422 0.03098178     * 0.1138652
```

```
dQ2b = read.delim("data_Q2b.txt", header = TRUE)
dQ2b$Subject   = as.factor(dQ2b$Subject)
dQ2b$Condition = as.factor(dQ2b$Condition)

print("t-test")
```

```
## [1] "t-test"
```

```
t.test(dQ2b$Measure[dQ2b$Condition=='A'],
       dQ2b$Measure[dQ2b$Condition=='B'], paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  dQ2b$Measure[dQ2b$Condition == "A"] and dQ2b$Measure[dQ2b$Condition == "B"]
## t = -1.9403, df = 11, p-value = 0.0784
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -14.9403881   0.9403881
## sample estimates:
## mean of the differences
##                       -7
```

```
print("ANOVA Repeated Measure")
```

```
## [1] "ANOVA Repeated Measure"
```

```
print(ezANOVA(data = dQ2b, dv=.(Measure), wid=.(Subject), within=.(Condition),
              type=2,
              detailed=FALSE))
```

```
## $ANOVA
##      Effect DFn DFd        F          p p<.05       ges
## 2 Condition   1  11 6.113422 0.03098178     * 0.1138652
```

Explanation - The observed discrepancies between the “Valid_case” and the “Invalid_case” are actually evident in the differing p-values obtained from the t-test and ANOVA. We can observe that the “Valid_case” has the same p-value for both the t-test and the ANOVA (pvalue_t_test = pvalue_anova = 0.03098178). While the “Invalid_case” has different p-values (pvalue_anova = 0.03098178 but pvalue_t_test = 0.0784). In the “Invalid_case”, the ANOVA shows statistically significant difference between the two conditions (p-value = 0.03098178 $<$ 0.05), but the t-test does not indicate such thing (p-value = 0.0784 $>$ 0.05). Specifically, the misplacement of s1 (with category A) in data_Q2b.txt impacted the t-test by mismatching the pairing between conditions A and B, which consequently affected the pairwise differences. Hence the distorted difference measures, affected the t-test’s p-value (p-value = 0.0784), which did not match with the ANOVA’s p-value (0.03098178, that was not actually affected). The solution to this issue is reordering “data_Q2b.txt” such that the paired t-test correctly compares the same subject’s measures for both conditions without any mismatches.

```r
# Revising the code to make the results of the t-test and the ANOVA become the same

dQ2b_revised = read.delim("data_Q2b.txt", header = TRUE)
levels_order = c('s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10',
                 's11', 's12')
dQ2b_revised$Subject = factor(dQ2b_revised$Subject, levels = levels_order)
dQ2b_revised$Condition = as.factor(dQ2b_revised$Condition)

# Sort the data frame by Condition and then by Subject
dQ2b_revised = dQ2b_revised[order(dQ2b_revised$Condition, dQ2b_revised$Subject), ]

print("t-test")
```

```
## [1] "t-test"
```

```r
t.test(dQ2b_revised$Measure[dQ2b_revised$Condition=='A'],
       dQ2b_revised$Measure[dQ2b_revised$Condition=='B'], paired=TRUE)
```

```
##
##  Paired t-test
##
## data:  dQ2b_revised$Measure[dQ2b_revised$Condition == "A"] and dQ2b_revised$Measure[dQ2b_revised$Cond
## t = -2.4725, df = 11, p-value = 0.03098
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -13.2312185  -0.7687815
## sample estimates:
## mean of the differences
##                      -7
```

```r
print("ANOVA Repeated Measure")
```

```
## [1] "ANOVA Repeated Measure"
```

```r
print(ezANOVA(data = dQ2b_revised, dv=.(Measure), wid=.(Subject),
              within=.(Condition), type=2, detailed=FALSE))
```

```
## $ANOVA
##      Effect DFn DFd        F          p p<.05       ges
## 2 Condition   1  11 6.113422 0.03098178     * 0.1138652
```

Final comment - After the revision, the results of the t-test and the ANOVA become the same, just like in the "Valid_case".

Q3. Effect of number of levels

The following script is for a Monte-Carlo simulation that estimates probability of Type-1 error of a 1-way ANOVA. The number of conditions is three in the current script. The estimated probability is about 0.05 with this number of conditions. Revise the script so that the number of conditions become thirty and run the simulation with the 30 conditions for several times. Based on the simulation, discuss how probability of Type-1 error is affected by the number of conditions.

```
nSessions = 1000
numSamples = 10 # number of samples for each condition
countSignificant = 0
for(s in 1:nSessions)
{
  dQ3 = data.frame( "Measure"   = rnorm(numSamples*3),
                    "Condition" = rep(seq(3),each=numSamples) )
  dQ3$Condition = as.factor(dQ3$Condition)

  result = Anova(aov(Measure ~ Condition, data=dQ3), type=2)
  if(result$`Pr(>F)`[1]<0.05)
    countSignificant = countSignificant+1
}
countSignificant/nSessions # Type-1 error
```

```
## [1] 0.052
```

```
# Revised code where the number of conditions is 30

nSessions = 1000
numSamples = 10
numConditions = 30
countSignificant = 0

for(s in 1:nSessions)
{
  dQ3 = data.frame( "Measure"   = rnorm(numSamples*numConditions),
                    "Condition" = rep(seq(numConditions),each=numSamples) )
  dQ3$Condition = as.factor(dQ3$Condition)

  result = Anova(aov(Measure ~ Condition, data=dQ3), type=2)
  if(result$`Pr(>F)`[1]<0.05)
    countSignificant = countSignificant+1
}

countSignificant/nSessions # Type-1 error
```

```
## [1] 0.041
```

```
# results of 50 runs
results = c(
  0.054, 0.045, 0.072, 0.05, 0.049, 0.047, 0.051, 0.042, 0.05, 0.043,
  0.047, 0.057, 0.046, 0.055, 0.051, 0.045, 0.061, 0.055, 0.043, 0.048,
  0.041, 0.038, 0.045, 0.042, 0.048, 0.047, 0.045, 0.052, 0.043, 0.057,
  0.047, 0.055, 0.044, 0.048, 0.06, 0.052, 0.038, 0.041, 0.056, 0.047,
  0.044, 0.05, 0.048, 0.044, 0.053, 0.045, 0.049, 0.054, 0.047, 0.041
)
mean_error_rate = mean(results)

print(paste("With numConditions = 30, the mean Type-1 error rate across the 50 runs is:",
            round(mean_error_rate, 4)))
```

```
## [1] "With numConditions = 30, the mean Type-1 error rate across the 50 runs is: 0.0486"
```

Comment - As you can see above I have revised the script by putting numConditions = 30, and then ran the revised script 50 times. Across these 50 runs, the mean Type-1 error rate was found to be 0.0486, which is very close to the nominal level of 0.05. Hence increasing the number of conditions (from 3 to 30 in our example) in a 1-way ANOVA, does not appear to inflate the Type-1 error rate, based on the given results. In a 1way ANOVA test, the stability of Type-1 error rates, even with more conditions, is due to the characteristics of the F-distribution, which ensure robust statistical outcomes.

Q4. Cauchy distribution

The following script compute (i) 1000 samples from a Cauchy distribution and (ii) 1000 averages of 1000 samples from the Cauchy distribution. Revise the script so that it computes 25% and 75% quantiles of (i) the 1000 samples and (ii) the 1000 averages. Explain how this result becomes different if a normal distribution is used instead of the Cauchy distribution.

```
# Initial code (provided by the instructor)
nSamples = 1000
nSessions = 1000

samples = rcauchy(nSamples,location=0,scale=1) # (i)
#samples = rnorm(nSamples,mean=0,sd=1)

computed_averages = rep(0,nSessions)  # (ii)
for(s in 1:nSessions)
{
  samples = rcauchy(nSamples,location=0,scale=1)
  #samples = rnorm(nSamples,mean=0,sd=1)
  computed_averages[s] = mean(samples)
}
```

```
# Revised version of code with CAUCHY distribution
#set.seed(20)
nSamples = 1000
nSessions = 1000

samples = rcauchy(nSamples,location=0,scale=1) # (i)

# Compute 25% and 75% quantiles for the samples
quantiles_samples = quantile(samples, c(0.25, 0.75))

computed_averages = rep(0,nSessions)  # (ii)
for(s in 1:nSessions)
{
  samples_session = rcauchy(nSamples,location=0,scale=1)
  computed_averages[s] = mean(samples_session)
}
```

```
# Compute 25% and 75% quantiles for the computed averages
quantiles_averages = quantile(computed_averages, c(0.25, 0.75))

list(quantiles_samples, quantiles_averages)
```

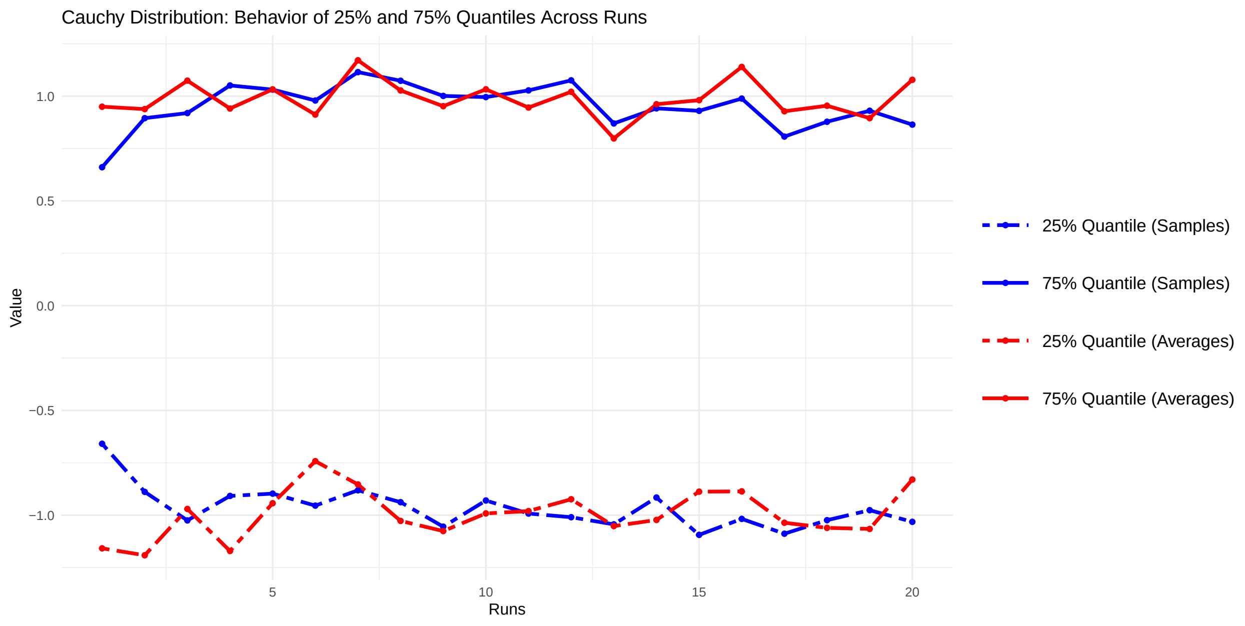
```
## [[1]]
##        25%        75%
## -0.9660796  0.9961441
##
## [[2]]
##        25%        75%
## -0.8878378  0.9691506
```

```
# Revised version of code with Normal distribution
set.seed(20)
nSamples = 1000
nSessions = 1000

samples = rnorm(nSamples,mean=0,sd=1) # (i)

# Compute 25% and 75% quantiles for the samples
quantiles_samples = quantile(samples, c(0.25, 0.75))

computed_averages = rep(0,nSessions)  # (ii)
for(s in 1:nSessions)
{
  samples_session = rnorm(nSamples,mean=0,sd=1)
  computed_averages[s] = mean(samples_session)
}

# Compute 25% and 75% quantiles for the computed averages
quantiles_averages = quantile(computed_averages, c(0.25, 0.75))

list(quantiles_samples, quantiles_averages)
```

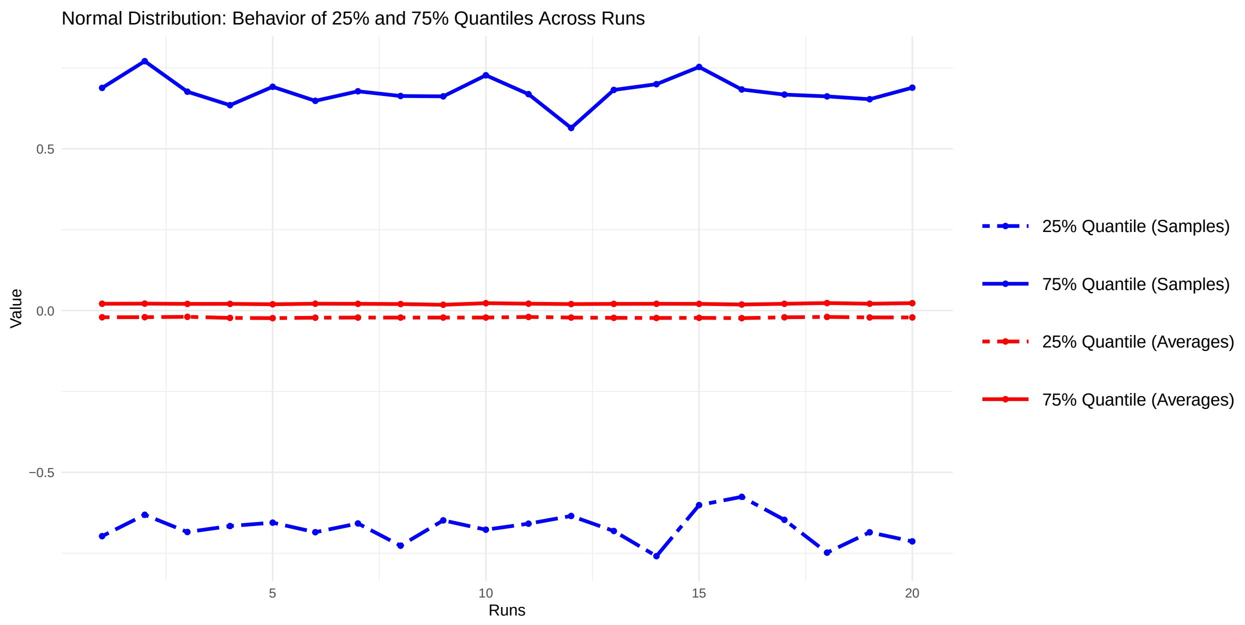
```
## [[1]]
##        25%        75%
## -0.7135274  0.6891248
##
## [[2]]
##         25%         75%
## -0.02116278  0.02295860
```

```
seeds = 1:20

# Cauchy distribution's results
cauchy_quantiles_samples_25 = c(-0.6592340, -0.8889651, -1.0251718, -0.9083494,
                                -0.8973477, -0.9545470, -0.8813467, -0.9382067,
                                -1.055064, -0.9301922, -0.9924637, -1.009905,
                                -1.0440028, -0.9159614, -1.0941650, -1.0179040,
                                -1.0886402, -1.0240897, -0.9764346, -1.0320084)
cauchy_quantiles_samples_75 = c(0.6608086, 0.8949452, 0.9192787, 1.0511170,
                                1.0313645, 0.9791878, 1.1148804, 1.0735509,
                                1.001118, 0.9955910, 1.0275856, 1.075464,
                                0.8697839, 0.9418021, 0.9299984, 0.9885194,
                                0.8069572, 0.8779009, 0.9304032, 0.8641989)
cauchy_quantiles_averages_25 = c(-1.1585927, -1.1915048, -0.9704574, -1.1710972,
                                 -0.9431258, -0.7423713, -0.8534313, -1.027570,
                                 -1.0760461, -0.9918616, -0.9805813, -0.9241974,
                                 -1.0523395, -1.0229500, -0.8879174, -0.8865885,
                                 -1.0362877, -1.0607767, -1.0657972, -0.8299852)
cauchy_quantiles_averages_75 = c(0.9497479, 0.9382058, 1.0741134, 0.9410221,
                                 1.0324786, 0.9120259, 1.1712806, 1.027244,
                                 0.9521988, 1.0327607, 0.9456033, 1.0208938,
                                 0.7981661, 0.9615237, 0.9808746, 1.1396553,
                                 0.9278658, 0.9543577, 0.8950274, 1.0781092)

# Normal distribution's results
normal_quantiles_samples_25 = c(-0.6973732, -0.6313009, -0.6845385, -0.6660838,
                                -0.6555508, -0.6851311, -0.6580562, -0.7268260,
                                -0.6485554, -0.6773931, -0.6587820, -0.6349178,
                                -0.6813779, -0.7591309, -0.6013262, -0.5757622,
                                -0.6468959, -0.7482944, -0.6854374, -0.7135274)
normal_quantiles_samples_75 = c(0.6884280, 0.7710561, 0.6766734, 0.6350464,
                                0.6918760, 0.6484364, 0.6779643, 0.6633979,
                                0.6623092, 0.7274625, 0.6691088, 0.5645019,
                                0.6821980, 0.6999347, 0.7531679, 0.6835095,
                                0.6677287, 0.6622054, 0.6534224, 0.6891248)
normal_quantiles_averages_25 = c(-0.02069163, -0.02035298, -0.01925586,
                                 -0.02266657, -0.02332102, -0.0219865,
                                 -0.02160582, -0.02163358, -0.02157333,
                                 -0.02153704, -0.01971377, -0.0216068,
                                 -0.02244544, -0.02288335, -0.02249039,
                                 -0.02338959, -0.02076908, -0.01957310,
                                 -0.02123830, -0.02116278)
normal_quantiles_averages_75 = c(0.02110866, 0.02155052, 0.02068702, 0.02078538,
                                 0.01949457, 0.0212784, 0.02092673, 0.02022019,
                                 0.01802176, 0.02283588, 0.02131310, 0.0201432,
                                 0.02070044, 0.02103077, 0.02077226, 0.01874528,
                                 0.02103220, 0.02322702, 0.02118061, 0.02295860)


cauchy_df = data.frame(
  Seed = rep(seeds, 4),
  Value = c(cauchy_quantiles_samples_25, cauchy_quantiles_samples_75,
            cauchy_quantiles_averages_25,
            cauchy_quantiles_averages_75),
  Type = factor(rep(c('25% Quantile (Samples)', '75% Quantile (Samples)',
                      '25% Quantile (Averages)',
                      '75% Quantile (Averages)'), each=length(seeds)))
)

normal_df = data.frame(
  Seed = rep(seeds, 4),
  Value = c(normal_quantiles_samples_25, normal_quantiles_samples_75,
            normal_quantiles_averages_25, normal_quantiles_averages_75),
  Type = factor(rep(c('25% Quantile (Samples)', '75% Quantile (Samples)',
                      '25% Quantile (Averages)', '75% Quantile (Averages)'),
                    each=length(seeds)))
)
```

```
custom_colors = c("25% Quantile (Samples)" = "blue",
                  "75% Quantile (Samples)" = "blue",
                  "25% Quantile (Averages)" = "red",
                  "75% Quantile (Averages)" = "red")

custom_linetypes = c("25% Quantile (Samples)" = "twodash",
                     "75% Quantile (Samples)" = "solid",
                     "25% Quantile (Averages)" = "twodash",
                     "75% Quantile (Averages)" = "solid")

cauchy_df$Type = factor(cauchy_df$Type, levels=c("25% Quantile (Samples)",
                                                 "75% Quantile (Samples)",
                                                 "25% Quantile (Averages)",
                                                 "75% Quantile (Averages)"))

cauchy_plot = ggplot(cauchy_df, aes(x=Seed, y=Value, color=Type, linetype=Type)) +
  geom_line(linewidth=1.2) +
  geom_point(size=1.44) +
  scale_color_manual(values=custom_colors) +
  scale_linetype_manual(values=custom_linetypes) +
  labs(title='Cauchy Distribution: Behavior of 25% and 75% Quantiles Across Runs',
       y='Value', x='Runs') +
  theme_minimal()+
  theme(legend.text = element_text(size=12),
        legend.title = element_blank(),
        legend.key.size = unit(1.41, "cm"))


print(cauchy_plot)
```



```r
normal_df$Type = factor(normal_df$Type, levels=c("25% Quantile (Samples)",
                                                 "75% Quantile (Samples)",
                                                 "25% Quantile (Averages)",
                                                 "75% Quantile (Averages)"))

normal_plot = ggplot(normal_df, aes(x=Seed, y=Value, color=Type, linetype=Type)) +
  geom_line(linewidth=1.2) +
  geom_point(size=1.44) +
  scale_color_manual(values=custom_colors) +
  scale_linetype_manual(values=custom_linetypes) +
  labs(title='Normal Distribution: Behavior of 25% and 75% Quantiles Across Runs',
       y='Value', x='Runs') +
  theme_minimal()+
  theme(legend.text = element_text(size=12),
        legend.title = element_blank(),
        legend.key.size = unit(1.41, "cm"))

print(normal_plot)
```

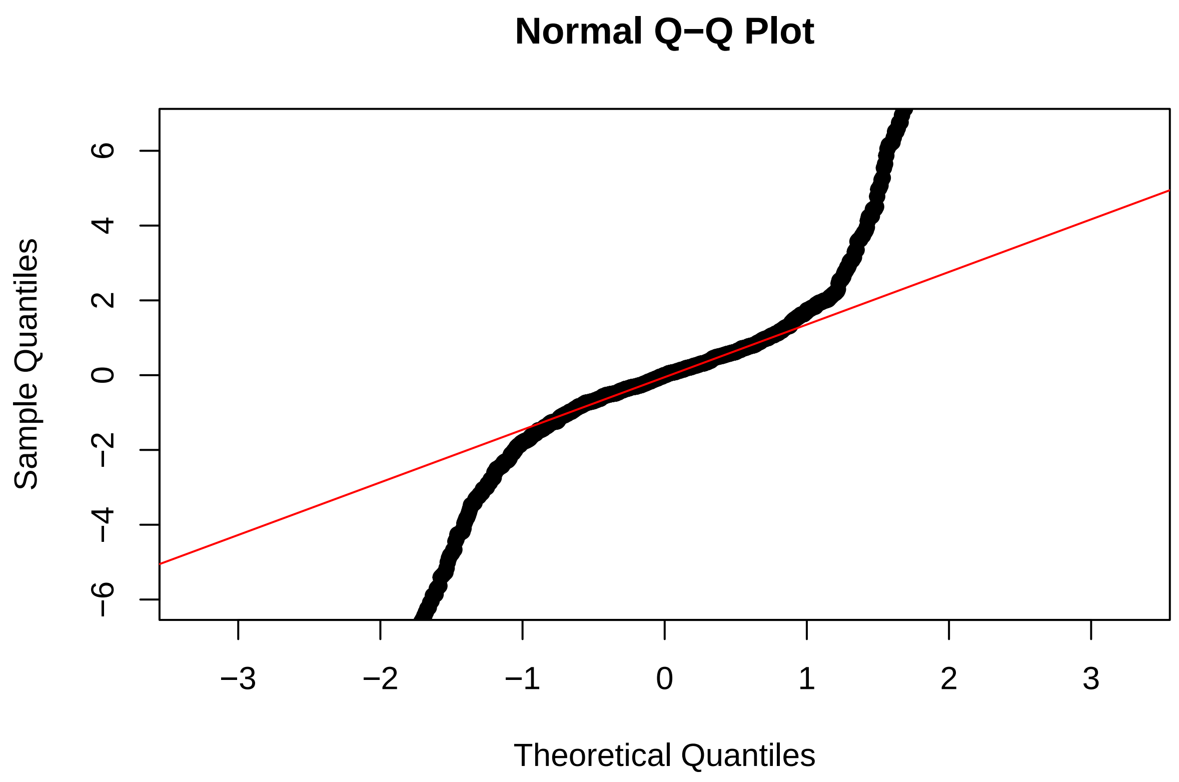


Comment - I have revised the code to compute the 25% and 75% quantiles for both Cauchy and Normal distributions. Upon executing the revised codes, 20 runs were performed for both distributions, each with a distinct fixed seed. The results of these runs were graphically represented in two plots, showing the behavior of the values of the quantiles across different runs, and illustrating the distinct behavior of the Cauchy and Normal distributions. After averaging, the Normal distribution's 25% and 75% quantiles come very close to zero, indicating the Central Limit Theorem in action. In contrast, the Cauchy distribution's quantiles remain unaffected by averaging. This behavior of the Cauchy distribution underscores the fact it does not follow the CLT. Ultimately we can see the plots vividly demonstrate this contrast.

Q5. CLT and outliers

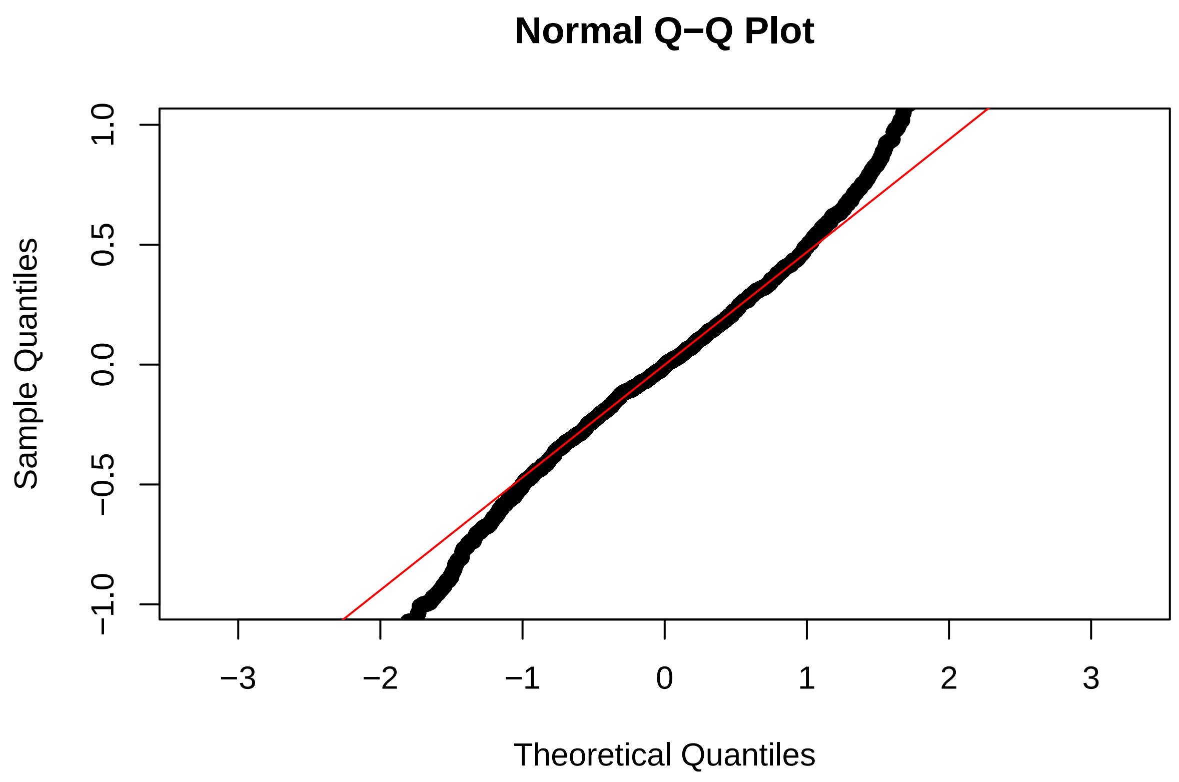
The following script compute 1000 averages of 1000 samples from the Cauchy distribution and draw a Normal probability plot from the 1000 averages. As you know, the Cauchy distribution does not satisfy conditions of the Central Limit Theorem. So, the 1000 averages does not become closer to a normal distribution no matter how large the sample size (nSamples) is. Add several lines in a specified place in the script so that: Outliers are removed from 1000 samples in each session. Note that the outlier is defined in this question as a sample whose distance from an estimated mean is more than 3 times of an estimated standard deviation. Both the estimated mean and the estimated standard deviation are computed from the 1000 samples before removing the outliers in each session. Run the revised script several times and report whether this operation removing the outliers makes a distribution of the 1000 averages closer to a normal distribution.

```
# Initial code (provided by the instructor)
nSamples = 100
nSessions = 1000
computed_averages = rep(0,nSessions)
for(s in 1:nSessions)
{
  samples = rcauchy(nSamples,location=0,scale=1)

  ########################
  # Q5. Additional lines #
  ########################

  computed_averages[s] = mean(samples)
}

quantile_averages = quantile(computed_averages, probs=c(0.05,0.95))
qqnorm(computed_averages, pch=19, ylim=c(quantile_averages[1],quantile_averages[2]))
qqline(computed_averages, col="red") # Passing 25% and 75% quantiles
```



```
# Revised version of code where the appropriate changed are made.
```

```r
nSamples = 100
nSessions = 1000
computed_averages = rep(0,nSessions)

for(s in 1:nSessions) {
  samples = rcauchy(nSamples,location=0,scale=1)

  # The additional lines
  estimated_mean = mean(samples)
  estimated_sd = sd(samples)

  lower_bound = estimated_mean - 3*estimated_sd
  upper_bound = estimated_mean + 3*estimated_sd

  samples_cleaned = samples[samples > lower_bound & samples < upper_bound]

  computed_averages[s] = mean(samples_cleaned)
}

quantile_averages = quantile(computed_averages, probs=c(0.05,0.95))
qqnorm(computed_averages, pch=19, ylim=c(quantile_averages[1],quantile_averages[2]))
qqline(computed_averages, col="red")
```



Comment - The first QQ plot for the averages of the Cauchy samples (without removing the outliers), notably deviates from the red line. This underlines the fact that the averages from the Cauchy distribution don't converge to normality as expected by the CLT (as Cauchy distribution does not follow to CLT). However after removing outliers we can observe that the points on the graph are much closer to the red line. This suggests that the operation of removing the outliers makes a distribution of the 1000 averages (from the Cauchy distribution) become closer to a normal distribution.

Q6. ANOVA 1-sample per cell (Montgomery, 2012, p.205, Example 5.2, Tabel 5.10)

Data in "data_Q5.txt" are results of an experiment with two factors: FactorA with five levels and FactorB with four levels. There is only one sample in each condition and there are 20 samples in total. Complete the following script based on the section "5.3.7 One Observation per Cell" in Montgomery (2012, pp.203-206) and compute two p-values of the main effects of FactorA and FactorB and one p-value of their interaction.

```
dQ6 = read.delim("data_Q6.txt", header = TRUE)
dQ6$FactorA = as.factor(dQ6$FactorA)
dQ6$FactorB = as.factor(dQ6$FactorB)
```

```
# Creating a table for our problem just like the example in page 205, Table 5.10

pivot_dQ6 = dQ6 %>% spread(FactorB, Measure)
pivot_dQ6$y_i. = rowSums(pivot_dQ6[, 2:5])
y_j = colSums(pivot_dQ6[, 2:6])
y_j_df = data.frame(FactorA = "y_.j", t(y_j))
pivot_dQ6 = rbind(pivot_dQ6, y_j_df)

pivot_dQ6
```

```
##   FactorA         B1         B2         B3        B4       y_i.
## 1      A1  -3.468310 -2.1844611 -1.5522343  1.397969 -5.8070368
## 2      A2  -3.554355 -2.8054731 -1.3312699  2.862431 -4.8286663
## 3      A3  -3.930561 -1.1179899 -0.3555411  5.131529 -0.2725624
## 4      A4  -3.000416 -0.9056813  1.3733775  7.465824  4.9331044
## 5      A5  -3.161469  0.4454867  3.5848155  7.496393  8.3652257
## 6    y_.j -17.115111 -6.5681187  1.7191477 24.354147  2.3900646
```

```
# Compute SS(A)
y_i_values = pivot_dQ6$y_i.[1:5]
y_dot_dot = sum(y_i_values)
sum_y_i_squared = sum(y_i_values^2)
a = 5
b = 4
SS_A = (1/b) * sum_y_i_squared - (y_dot_dot^2 / (a*b))
SS_A
```

```
## [1] 37.57051
```

```
# Compute SS(B)
y_j_values = pivot_dQ6[nrow(pivot_dQ6), 2:5]
y_dot_dot = sum(y_j_values)
sum_y_j_squared = sum(y_j_values^2)
a = 5
b = 4
SS_B = (1/a) * sum_y_j_squared - (y_dot_dot^2 / (a*b))
SS_B
```

```
## [1] 186.1438
```

```
# Compute SS(T)
sum_squared_y_ij = sum(pivot_dQ6[1:a, 2:(b+1)]^2)
```

```r
SS_T = sum_squared_y_ij - (y_dot_dot^2 / (a*b))
SS_T
```

```
## [1] 241.1949
```

```r
# Compute SS(Residual)
SS_Residual = SS_T - SS_A - SS_B
SS_Residual
```

```
## [1] 17.48059
```

```r
# Compute SS(N)
results_df = data.frame(
  y_ij = numeric(0),
  y_i_dot = numeric(0),
  y_dot_j = numeric(0),
  product = numeric(0)
)

for (i in 1:a) {
  for (j in 1:b) {
    y_ij_value = pivot_dQ6[i, j+1]
    y_i_dot_value = pivot_dQ6[i, b+2]
    y_dot_j_value = pivot_dQ6[a+1, j+1]
    product_value = y_ij_value * y_i_dot_value * y_dot_j_value

    results_df = rbind(results_df, data.frame(y_ij = y_ij_value,
                                              y_i_dot = y_i_dot_value,
                                              y_dot_j = y_dot_j_value,
                                              product = product_value))
  }
}

sum_of_products = sum(results_df$product)
SS_N = ((sum_of_products - y_dot_dot * (SS_A + SS_B + (y_dot_dot^2 / (a*b))))^2) / (a*b*SS_A*SS_B)
SS_N
```

```
## [1] 11.89888
```

```r
# Compute SS(Error)
SS_Error = SS_Residual - SS_N
SS_Error
```

```
## [1] 5.581708
```

```r
# Compute the degrees of freedom for each

factor_a_df = a - 1
factor_b_df = b - 1
nonadditivity_df = 1
error_df = (a - 1)*(b - 1) - nonadditivity_df
total_df = factor_a_df + factor_b_df + nonadditivity_df + error_df

cat("Degrees of Freedom for Factor A:", factor_a_df, "\n")
```

```
## Degrees of Freedom for Factor A: 4
```

```r
cat("Degrees of Freedom for Factor B:", factor_b_df, "\n")
```

```
## Degrees of Freedom for Factor B: 3
```

```r
cat("Degrees of Freedom for Nonadditivity:", nonadditivity_df, "\n")
```

```
## Degrees of Freedom for Nonadditivity: 1
```

```r
cat("Degrees of Freedom for Error:", error_df, "\n")
```

```
## Degrees of Freedom for Error: 11
```

```r
cat("Total Degrees of Freedom:", total_df, "\n")
```

```
## Total Degrees of Freedom: 19
```

```r
# Compute the Mean Squares for each source of variation

MS_factor_A = SS_A / factor_a_df
MS_factor_B = SS_B / factor_b_df
MS_nonadditivity = SS_N / nonadditivity_df
MS_E = SS_Error / error_df

cat("Mean Square for Factor A:", MS_factor_A, "\n")
```

```
## Mean Square for Factor A: 9.392626
```

```r
cat("Mean Square for Factor B:", MS_factor_B, "\n")
```

```
## Mean Square for Factor B: 62.04794
```

```r
cat("Mean Square for Nonadditivity:", MS_nonadditivity, "\n")
```

```
## Mean Square for Nonadditivity: 11.89888
```

```r
cat("Mean Square for Error:", MS_E, "\n")
```

```
## Mean Square for Error: 0.507428
```

```r
# Compute the F statistics

F_factor_A = MS_factor_A / MS_E
F_factor_B = MS_factor_B / MS_E
F_nonadditivity = MS_nonadditivity / MS_E

cat("F statistic for Factor A :", F_factor_A, "\n")
```

```
## F statistic for Factor A : 18.51027
```

```r
cat("F statistic for Factor B :", F_factor_B, "\n")
```

```
## F statistic for Factor B : 122.2793
```

```r
cat("F statistic for Nonadditivity :", F_nonadditivity, "\n")
```

```
## F statistic for Nonadditivity : 23.4494
```

```r
pvalue_factor_A = 1 - pf(F_factor_A, factor_a_df, error_df)
pvalue_factor_B = 1 - pf(F_factor_B, factor_b_df, error_df)
pvalue_nonadditivity = 1 - pf(F_nonadditivity, nonadditivity_df, error_df)

cat("p-value for Factor A:", pvalue_factor_A, "\n")
```

```
## p-value for Factor A: 7.53831e-05
```

```
cat("p-value for Factor B:", pvalue_factor_B, "\n")
```

```
## p-value for Factor B: 9.945587e-09
```

```
cat("p-value for Nonadditivity:", pvalue_nonadditivity, "\n")
```

```
## p-value for Nonadditivity: 0.0005169575
```

Conclusion - Based on the computed pvalues for FactorA (0.0000753831), FactorB (0.000000009945587), and Nonadditivity (0.0005169575), all of which are less than the significance level of 0.05, we can reject the null hypothesis. This indicates statistically significant effects for both factors and their interaction in the experiment.