Appendix

Analysis of Sample 1

Load the data.

```
library(readxl)
sample1 = read_xlsx("term project data.xlsx", range = "A2:A32")$sample
n = length(sample1)
```

We want to test the hypothesis

$$H_0 : \mu = 0 \quad \text{v.s.} \quad H_1 : \mu > 0.$$

One sample t-test

$$Z_t = \frac{\bar{X} - 0}{S/\sqrt{n}}$$

$$p\text{-value} = \text{upper quantile of } Z_t \text{ in } t_{n-1} \text{ distribution}$$

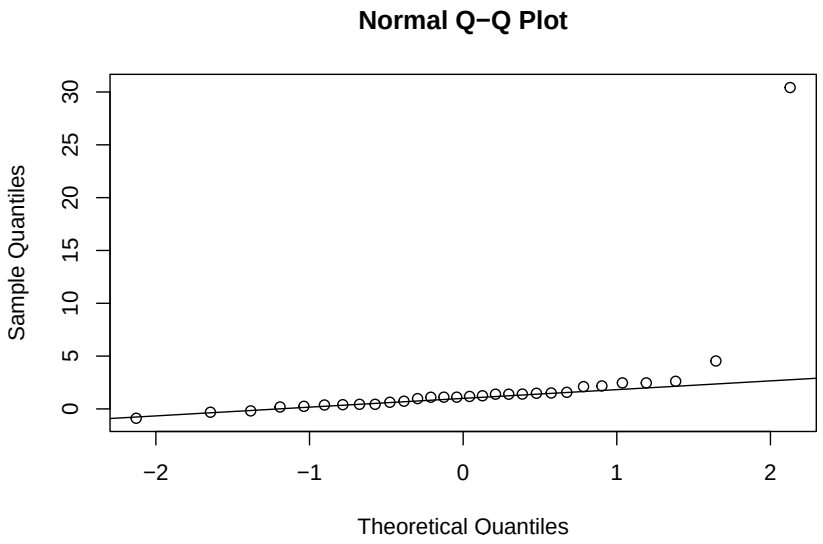
```
t.test(sample1, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  sample1
## t = 2.1569, df = 29, p-value = 0.01972
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  0.4549496       Inf
## sample estimates:
## mean of x
##  2.143567
```

$0.020 < 0.05$, indicating that we should reject the null hypothesis.

Use the qqplot to investigate the normality of sample 1.

```
qqnorm(sample1); qqline(sample1)
```



We find a significant outlier in the tail. We know that t-test is not a good test for data with outliers and extreme values. Therefore, we should use a non-parametric test.

Binomial test

$$B = \#\text{ of } \{X_i \geq 0\}$$

First calculate B_{obs}.

```
B_obs = sum(sample1 >= 0)
B_obs
```

```
## [1] 27
```

Two ways of getting p-value:

Exact p-value

$$\text{exact } p\text{-value} = P(B \geq 27 \mid H_0) = \sum_{i=27}^{30} \binom{30}{i} 0.5^{30}$$

```
p_value2 = sum(choose(n, B_obs:n)) * (0.5 ^ n)
p_value2
```

```
## [1] 4.215166e-06
```

Approximate p-value

$$Z_B = \frac{B - 0.5n}{\sqrt{0.25n}}$$

$$\text{approximate } p\text{-value} = 1 - \Phi(Z_B)$$

```
Z_B = (B_obs - 0.5 * n) / sqrt(0.25 * n)
p_value3 = 1 - pnorm(Z_B)
p_value3
```

```
## [1] 5.88567e-06
```

The exact p-value and the approximate p-value are very close and have the same order of magnitude of -6. Compared to the one sample t-test, the p-values of binomial test are much smaller, though they all give the same conclusion that the mean of population that the sample comes from is significantly greater than 0. This can be attributed to the greater power of binomial test on distribution with extreme values.

Analysis of Sample 2

Load the data.

```
sample2 = read_xlsx("term project data.xlsx", range = "B2:C32")
head(sample2)
```

```
## # A tibble: 6 x 2
##   group1 group2
##    <dbl>  <dbl>
## 1  0.431   2.06
## 2  2.14    1.94
## 3  0.662   5.89
## 4  1.28    9.20
## 5  0.946   5.14
## 6  3.50    3.04
```

We want to test the hypotheses

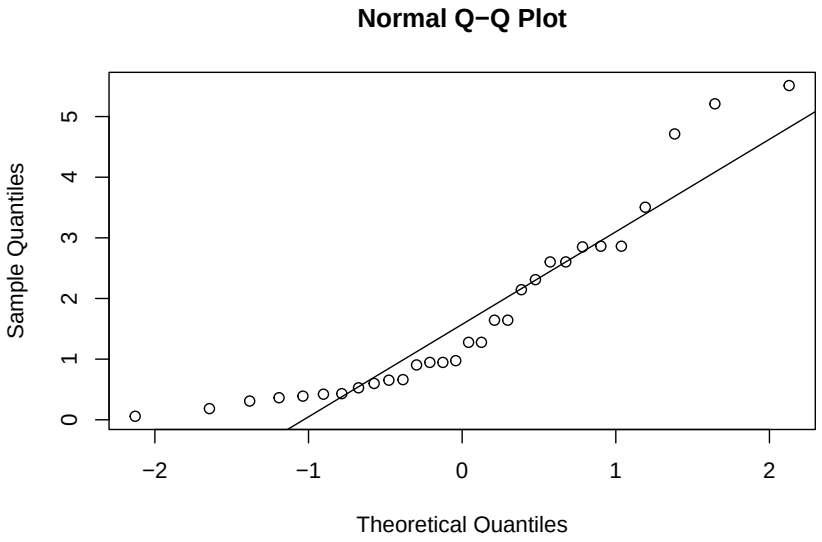
$$H_0 : \mu_1 = \mu_2 \quad \text{v.s.} \quad H_1 : \mu_1 < \mu_2,$$

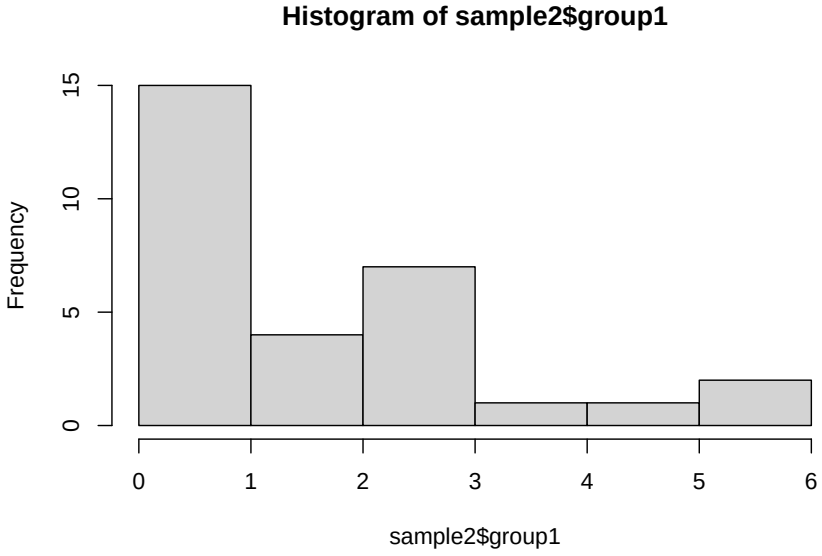
and

$$H_0 : \sigma_1 = \sigma_2 \quad \text{v.s.} \quad H_1 : \sigma_1 \neq \sigma_2.$$

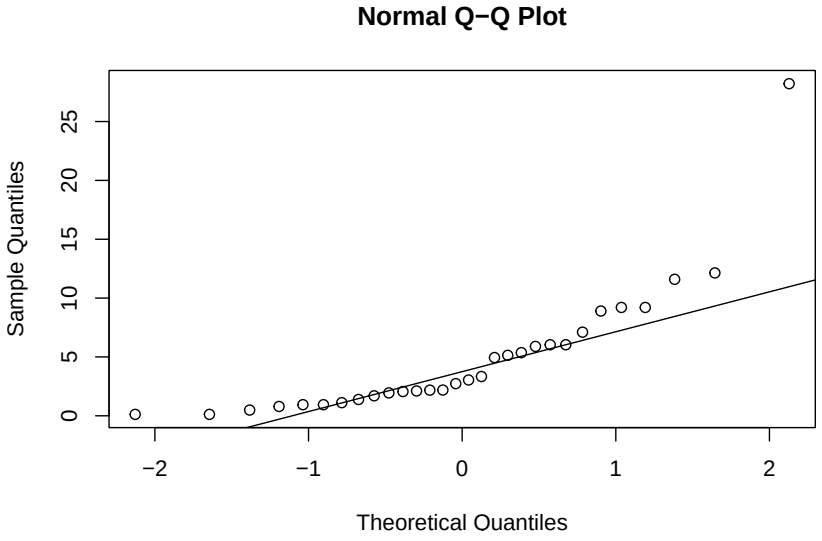
Preview the distribution of data. The data is heavily-tailed and highly-skewed.

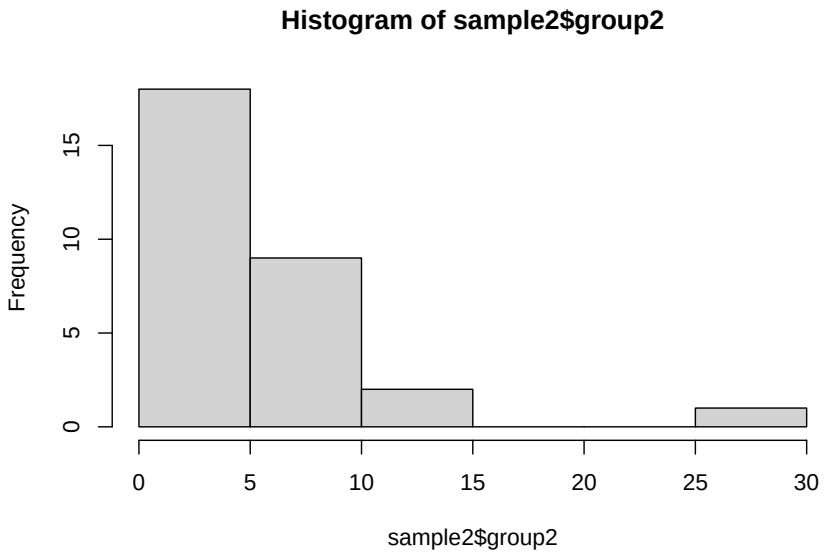
```
qqnorm(sample2$group1); qqline(sample2$group1); hist(sample2$group1)
```






```
qqnorm(sample2$group2); qqline(sample2$group2); hist(sample2$group2)
```





Comparison of Means

Two sample t-test

```
t.test(sample2$group1[order(sample2$group1)],
       sample2$group2[order(sample2$group2)], alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  sample2$group1[order(sample2$group1)] and sample2$group2[order(sample2$group2)]
## t = -3.0183, df = 33.249, p-value = 0.002426
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -1.400068
## sample estimates:
## mean of x mean of y
##    1.7128    4.8989
```

The p-value is $0.0024 < 0.05$, indicating that there's a true location shift.

Permutation Test

Define the function of permutation test.

```
permut_test = function(X, Y, method = "mean", alternative = "two.sided", sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  if (method == "mean" | method == "trimmed"){
    score = c
  }else if (method == "median"){
```

```
    score = rank
  }else if (method == "waerden"){
    score = function(data){
      waerden_score = function(i, N){qnorm(i / (N + 1))}
      waerden_score(rank(data), N)}
  }else if (method == "exp"){
    score = function(data){
      exp_score = function(i, N){qnorm(i / (N + 1))}
      exp_score(rank(data), N)}
  }
  df_c = data.frame(data = c(X, Y),
                    group = c(rep(1, m), rep(2, n)),
                    score = score(c(X, Y)))
  if (method == "trimmed"){
    D_obs = mean(df_c$score[df_c$group == 1], trim = 0.05) -
      mean(df_c$score[df_c$group == 2], trim = 0.05)
  }else{
    D_obs = mean(df_c$score[df_c$group == 1]) -
      mean(df_c$score[df_c$group == 2])
  }
  perm_D = rep(0, sim_num)
  set.seed(2023)
  if (method == "trimmed"){
    for (i in 1:sim_num){
      permut = sample(df_c$group)
      perm_D[i] = mean(df_c$score[permut == 1], trim = 0.05) -
        mean(df_c$score[permut == 2], trim = 0.05)}
  }else{
    for (i in 1:sim_num){
      permut = sample(df_c$group)
      perm_D[i] = mean(df_c$score[permut == 1]) -
        mean(df_c$score[permut == 2])}
  }
  if (alternative == "two.sided"){
    p = sum(abs(perm_D) >= abs(D_obs)) / sim_num
  }else if (alternative == "greater"){
    p = sum(perm_D >= D_obs) / sim_num
  }else if (alternative == "less"){
    p = sum(perm_D <= D_obs) / sim_num
  }
  return (p)
}
```

Difference of means:

```
permut_test(sample2$group1, sample2$group2,
            method = "mean",
            alternative = "less",
            sim_num = 100000)
```

```
## [1] 0.00022
```

Difference of medians:

```
permut_test(sample2$group1, sample2$group2,
            method = "median",
            alternative = "less",
            sim_num = 100000)
```

```
## [1] 0.00125
```

Difference of 10% trimmed means:

```
permut_test(sample2$group1, sample2$group2,
            method = "trimmed",
            alternative = "less",
            sim_num = 100000)
```

```
## [1] 0.00028
```

Difference of Van Der Waerden scores:

```
permut_test(sample2$group1, sample2$group2,
            method = "waerden",
            alternative = "less",
            sim_num = 100000)
```

```
## [1] 0.00104
```

Difference of exponential scores:

```
permut_test(sample2$group1, sample2$group2,
            method = "exp",
            alternative = "less", sim_num = 100000)
```

```
## [1] 0.00104
```

All the p-values show that we should reject the null hypothesis, so there exists a significant difference between the mean of data.

Wilcoxon Rank-sum Test

Define the function of Wilcoxon rank-sum test that can apply to data with ties (using normal approximation and Monte Carlo simulation permutation).

```
wilcoxon_test = function(X, Y, alternative = "two.sided", sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  df_c = data.frame(data = c(X, Y),
                    group = c(rep(1, m), rep(2, n)),
                    rank = rank(c(X, Y)))
  W1_obs = sum(df_c$rank[df_c$group == 1])
```

```
  # normal approximation
  AF = 0
  for (d in unique(df_c$data)){
    df_c[df_c$data == d, 3] = mean(df_c[df_c$data == d, 3])
    t_i = sum(df_c$data == d)
    AF = AF + t_i ^ 3 - t_i
  }
  AF = AF * m * n / (12 * N * (N - 1))
  Z_W = (W1_obs - 0.5 * m * (N + 1)) / sqrt(m * n * (N + 1) / 12 - AF)
  if (alternative == "two.sided"){
    p1 = 2 * (1 - pnorm(abs(Z_W)))
  }else if (alternative == "greater"){
    p1 = 1 - pnorm(Z_W)
  }else if (alternative == "less"){
    p1 = pnorm(Z_W)
  }
  # simulation of permutation test.
  perm_W1 = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(df_c$group)
    perm_W1[i] = sum(df_c$rank[permut == 1])
  }
  if (alternative == "two.sided"){
    p2 =  2 * min(sum(perm_W1 >= W1_obs), sum(perm_W1 <= W1_obs)) / sim_num
  }else if (alternative == "greater"){
    p2 = sum(perm_W1 >= W1_obs) / sim_num
  }else if (alternative == "less"){
    p2 = sum(perm_W1 <= W1_obs) / sim_num
  }
  return (list(p_value_normal_approximation = p1,
               p_value_monte_carlo_simulation_permutation = p2))
}
```

```
wilcoxon_test(sample2$group1, sample2$group2,
              alternative = "less", sim_num = 100000)
```

```
## $p_value_normal_approximation
## [1] 0.001409444
##
## $p_value_monte_carlo_simulation_permutation
## [1] 0.00125
```

The two p-values are close and both much less than 0.05, leading to the same conclusion as before.

Comparison of Variances

The location parameters are unknown and we cannot assume that they are the same (they are significantly different), so the following tests are conducted based on the original values minus the median of their group.

F-test

Define the function of F test.

```
F_test = function(X, Y, alternative = "two.sided"){
  m = length(X)
  n = length(Y)
  F_obs = var(X) / var(Y)
  p = pf(F_obs, m - 1, n - 1)
  if (alternative == "two.sided"){
    p = 2 * min(p, 1 - p)
  }else if (alternative == "greater"){
    p = 1 - p
  }else if (alternative == "less"){
    p = p
  }
  return (p)
}
```

```
F_test(sample2$group1, sample2$group2)
```

```
## [1] 4.318526e-10
```

The p-value is far smaller than 0.05, indicating that the variances are significantly different.

Siegel-Tukey Test

Define the function of Siegel-Tukey test with ties (using normal approximation and Monte Carlo simulation permutation).

```
siegel_tukey = function(X, Y, alternative = "two.sided", sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  df_c = data.frame(data = c(X - median(X), Y - median(Y)),
                    group = c(rep(1, m), rep(2, n)),
                    rank_ST = NA)
  iterator1 = matrix(seq(1, N, 4)) - 1
  rank1 = apply(iterator1, 1, function(x) x + c(1, 4))
  iterator2 = matrix(seq(2, N, 4))
  rank2 = apply(iterator2, 1, function(x) x + c(0, 1))
  if (length(rank1) == length(rank2)) {
    r = c(rank1[1:floor(N/2)], rev(rank2[1:ceiling(N/2)]))
  }else{
    r = c(rank1[1:ceiling(N/2)], rev(rank2[1:floor(N/2)]))
  }
  df_c[order(df_c$data), 3] = r
  # normal approximation
  AF = 0
  for (d in unique(df_c$data)){
    df_c[df_c$data == d, 3] = mean(df_c[df_c$data == d, 3])
    t_i = sum(df_c$data == d)
```

```
    AF = AF + t_i ^ 3 - t_i
  }
  AF = AF * m * n / (12 * N * (N - 1))
  W1_obs = sum(df_c$rank_ST[df_c$group == 1])
  Z_W = (W1_obs - 0.5 * m * (N + 1)) / sqrt(m * n * (N + 1) / 12 - AF)
  if (alternative == "two.sided"){
    p1 = 2 * (1 - pnorm(abs(Z_W)))
  }else if (alternative == "greater"){
    p1 = 1 - pnorm(Z_W)
  }else if (alternative == "less"){
    p1 = pnorm(Z_W)
  }
  # simulation of permutation test.
  perm_W1 = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(df_c$group)
    perm_W1[i] = sum(df_c$rank_ST[permut == 1])
  }
  if (alternative == "two.sided"){
    p2 = 2 * min(sum(perm_W1 >= W1_obs), sum(perm_W1 <= W1_obs)) / sim_num
  }else if (alternative == "greater"){
    p2 = sum(perm_W1 >= W1_obs) / sim_num
  }else if (alternative == "less"){
    p2 = sum(perm_W1 <= W1_obs) / sim_num
  }
  return (list(p_value_normal_approximation = p1,
               p_value_monte_carlo_simulation_permutation = p2))
}
```

```
siegel_tukey(sample2$group1, sample2$group2, sim_num = 10000)
```

```
## $p_value_normal_approximation
## [1] 1.204691e-05
##
## $p_value_monte_carlo_simulation_permutation
## [1] 0
```

The two p-values are both far smaller than 0.05, indicating that the variance of the two samples are significantly different, which is the same as the conclusion of F-test.

Ansari-Bradley Test

Define the function of Ansari-Bradley test with ties (using normal approximation and Monte Carlo simulation permutation).

```
ansari_bradley = function(X, Y, alternative = "two.sided", sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  df_c = data.frame(data = c(X - median(X), Y - median(Y)),
                    group = c(rep(1, m), rep(2, n)),
```

```
                    rank_AB = NA)
  if (N %% 2 == 0){
    r = c(1:(N / 2), (N / 2):1)
  }else{
    r = c(1:(N / 2 + 1), (N / 2 - 1):1)
  }
  df_c[order(df_c$data), 3] = r
  # Normal approximation
  AB = 0
  for (d in unique(df_c$data)){
    r_i = mean(df_c[df_c$data == d, 3])
    df_c[df_c$data == d, 3] = r_i
    t_i = sum(df_c$data == d)
    AB = AB + t_i * (r_i ^ 2)
  }
  C_obs = sum(df_c$rank_AB[df_c$group == 1])
  if (N %% 2 == 0){
    mu = m * (N + 2) / 4
    sigma = sqrt(m * n * (16 * AB - N * (N + 2) ^ 2) / (16 * N * (N - 1)))
  }else{
    mu = m * (N + 1) ^ 2 / (4 * N)
    sigma = sqrt(m * n * (16 * N * AB - (N + 1) ^ 4) / (16 * N ^ 2 * (N - 1)))
  }
  Z_C = (C_obs - mu) / sigma
  if (alternative == "two.sided"){
    p1 = 2 * (1 - pnorm(abs(Z_C)))
  }else if (alternative == "greater"){
    p1 = 1 - pnorm(Z_C)
  }else if (alternative == "less"){
    p1 = pnorm(Z_C)
  }
  # simulation of permutation test
  perm_C = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(df_c$group)
    perm_C[i] = sum(df_c$rank_AB[permut == 1])
  }
  if (alternative == "two.sided"){
    p2 = 2 * min(sum(perm_C >= C_obs), sum(perm_C <= C_obs)) / sim_num
  }else if (alternative == "greater"){
    p2 = sum(perm_C >= C_obs) / sim_num
  }else if (alternative == "less"){
    p2 = sum(perm_C <= C_obs) / sim_num
  }
  return (list(p_value_normal_approximation = p1,
               p_value_monte_carlo_simulation_permutation = p2))
}
```

```
ansari_bradley(sample2$group1, sample2$group2, sim_num = 10000)
```

```
## $p_value_normal_approximation
## [1] 1.35758e-05
```

```
##
## $p_value_monte_carlo_simulation_permutation
## [1] 0
```

The two p-values are both far smaller than 0.05, indicating that the variance of the two samples are significantly different, which is the same as the previous conclusions.

Permutation test based on RMD

Define the function of RMD test (using Monte Carlo simulation permutation).

```
RMD_test = function(X, Y, alternative = "two.sided", sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  df_c = data.frame(dev = abs(c(X - median(X), Y - median(Y))),
                    group = c(rep(1, m), rep(2, n)))
  if (alternative == "two.sided"){
    dev_X = mean(abs(X - median(X)))
    dev_Y = mean(abs(Y - median(Y)))
    RMD_obs = max(dev_X, dev_Y) / min(dev_X, dev_Y)
    # simulation of permutation test
    perm_RMD = rep(0, sim_num)
    set.seed(2023)
    for (i in 1:sim_num){
      permut = sample(df_c$group)
      dev_X = mean(df_c$dev[permut == 1])
      dev_Y = mean(df_c$dev[permut == 2])
      perm_RMD[i] = max(dev_X, dev_Y) / min(dev_X, dev_Y)
    }
    p = sum(perm_RMD >= RMD_obs) / sim_num
  }else{
    RMD_obs = mean(abs(X - median(X))) / mean(abs(Y - median(Y)))
    # simulation of permutation test
    perm_RMD = rep(0, sim_num)
    set.seed(2023)
    for (i in 1:sim_num){
      permut = sample(df_c$group)
      perm_RMD[i] = mean(df_c$dev[permut == 1]) / mean(df_c$dev[permut == 2])
    }
    if (alternative == "greater"){
      p = sum(perm_RMD >= RMD_obs) / sim_num
    }else if (alternative == "less"){
      p = sum(perm_RMD <= RMD_obs) / sim_num
    }
  }
  return (p)
}
```

```
RMD_test(sample2$group1, sample2$group2, sim_num = 100000)
```

```
## [1] 0.00039
```

p-value is 0.00039<0.05, indicating that the variance of the two samples are significantly different, which is the same as the previous conclusions.

Analysis of Sample 3

Load the data.

```
sample3 = read_xlsx("term project data.xlsx", range = "D2:E32")
head(sample3)
```

```
## # A tibble: 6 x 2
##   distrib1 distrib2
##      <dbl>    <dbl>
## 1     6.29   15.3
## 2    11.1     2.11
## 3     7.50    0.556
## 4    15.4     4.93
## 5     9.57   12.8
## 6    13.3    10.4
```

We want to test the hypothesis

$$H_0 : F_1(x) = F_2(x) \quad \text{v.s.} \quad H_1 : F_1(x) \neq F_2(x).$$

Kolmogorov-Smirnov Test

Use Kolmogorov-Smirnov test to compare the two distributions.

```
ks_test = function(X, Y, sim_num = 10000){
  m = length(X)
  n = length(Y)
  N = m + n
  df_c = data.frame(data = c(X, Y),
                    group = c(rep(1, m), rep(2, n)))
  x_ticks = sort(unique(df_c$data))
  F_1 = cumsum(tabulate(match(sort(X), x_ticks),
                        nbins = length(x_ticks))) / m
  F_2 = cumsum(tabulate(match(Y, x_ticks),
                        nbins = length(x_ticks))) / n
  KS_obs = max(abs(F_1 - F_2))
  # simulation permutation
  perm_KS = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(df_c$group)
    F_1 = cumsum(tabulate(match(df_c$data[permut == 1], x_ticks),
                          nbins = length(x_ticks))) / m
    F_2 = cumsum(tabulate(match(df_c$data[permut == 2], x_ticks),
                          nbins = length(x_ticks))) / n
    perm_KS[i] = max(abs(F_1 - F_2))
  }
```

```
  p = sum(perm_KS >= KS_obs) / sim_num
  return (p)
}
```

```
ks_test(sample3$distrib1, sample3$distrib2, sim_num = 100000)
```

```
## [1] 0.01464
```

p-value is $0.015 < 0.05$, indicating that the two distributions are significantly different.

Analysis of Sample 4

Load the data.

```
sample4 = read_xlsx("term project data.xlsx", range = "F2:I32")
head(sample4)
```

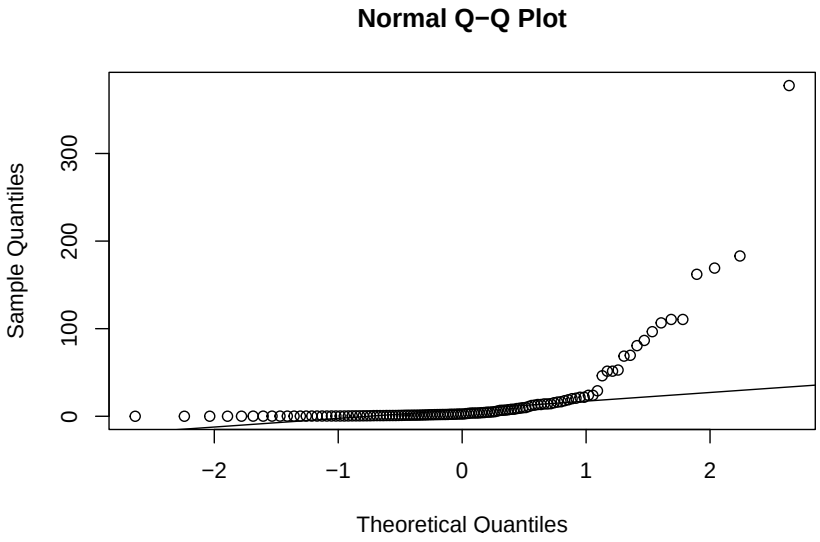
```
## # A tibble: 6 x 4
##      cat1    cat2    cat3   cat4
##     <dbl>   <dbl>   <dbl>  <dbl>
## 1   0.172 183.     80.6   46.2
## 2   1.74    0.198   0.564  3.67
## 3 169.      6.86    4.94   3.00
## 4   0.778   2.56  111.     0.874
## 5   1.03    5.26    9.72   0.966
## 6   1.32    1.50    0.026  1.40
```

First construct a long format of data frame.

```
data = data.frame(values = c(sample4$cat1, sample4$cat2,
                             sample4$cat3, sample4$cat4),
                  cat = rep(1:4, each = nrow(sample4)))
```

The data does not follow normal distribution and has a heavy tail.

```
qqnorm(data$values); qqline(data$values)
```



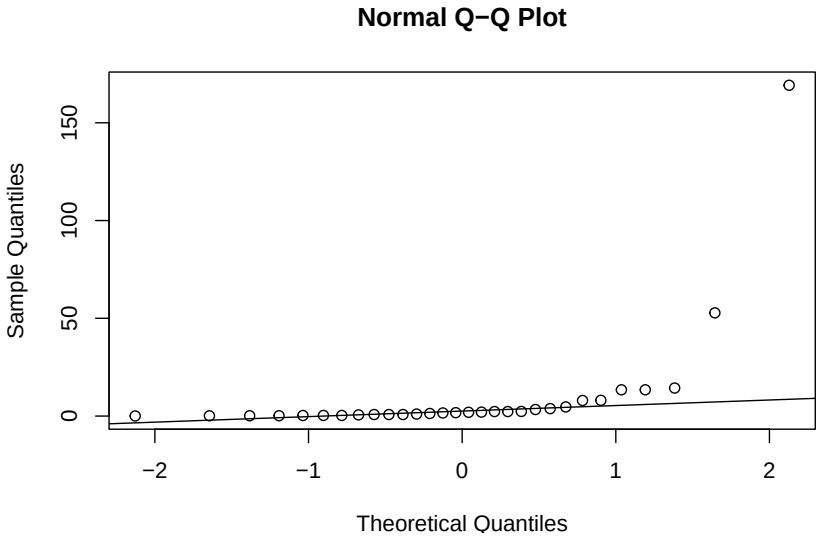
Use Shapiro-Wilk test to test normality. The p-value is really small, so the data is not normally distributed.

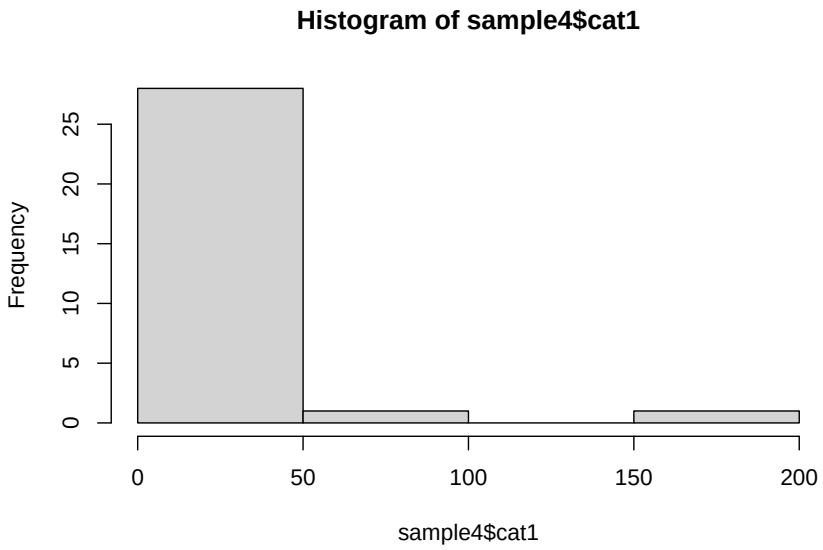
```
shapiro.test(data$values)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data$values
## W = 0.44346, p-value < 2.2e-16
```

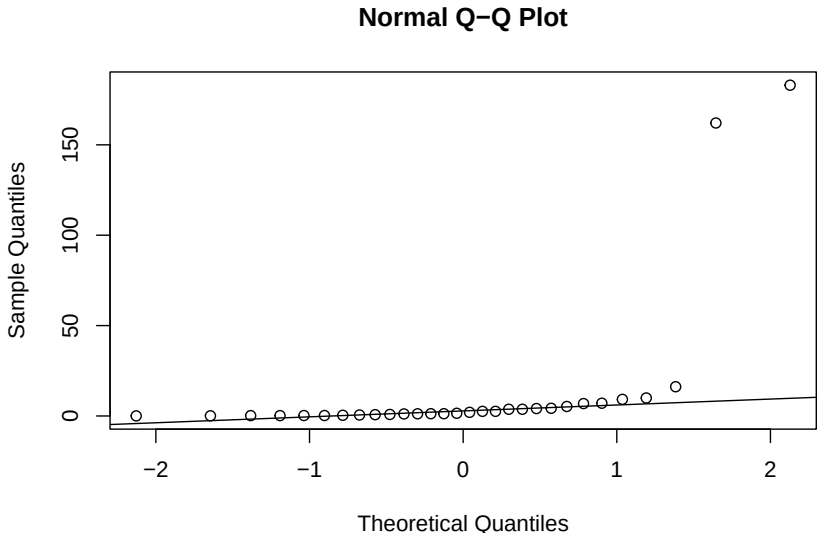
Moreover, in each group, the data follows a heavily-tailed and highly-skewed distribution.

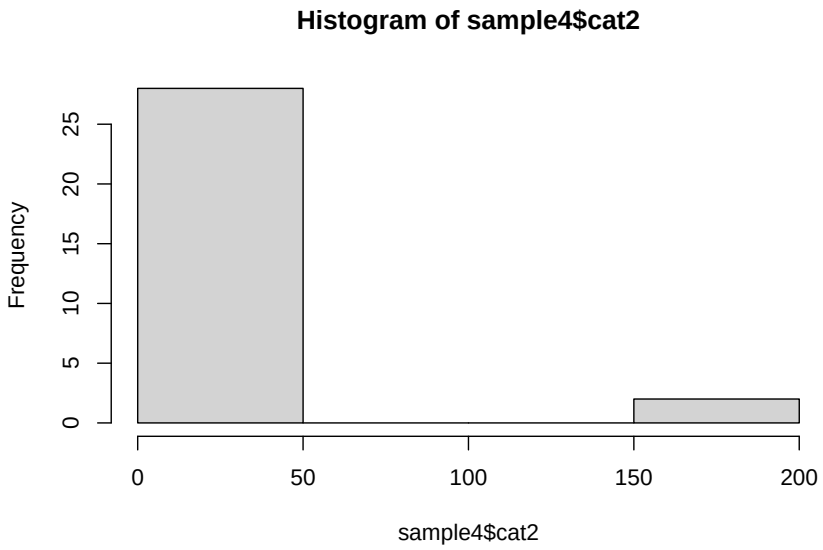
```
qqnorm(sample4$cat1); qqline(sample4$cat1); hist(sample4$cat1)
```



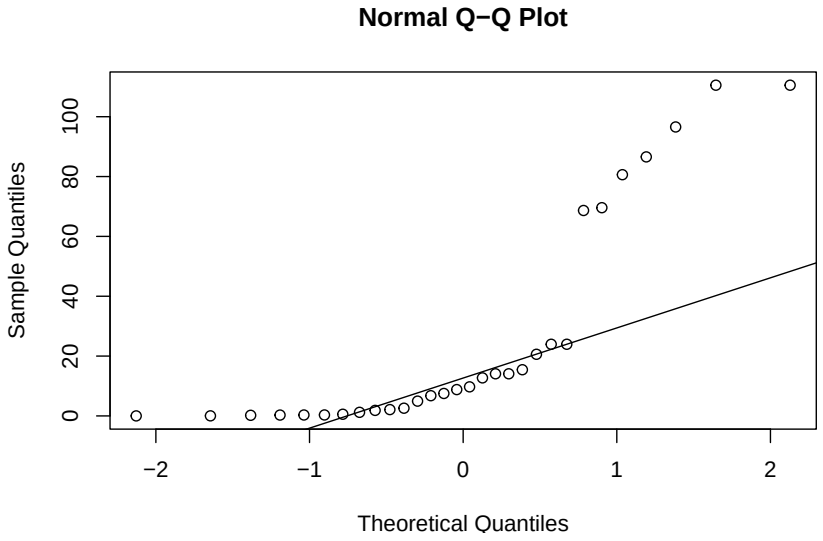


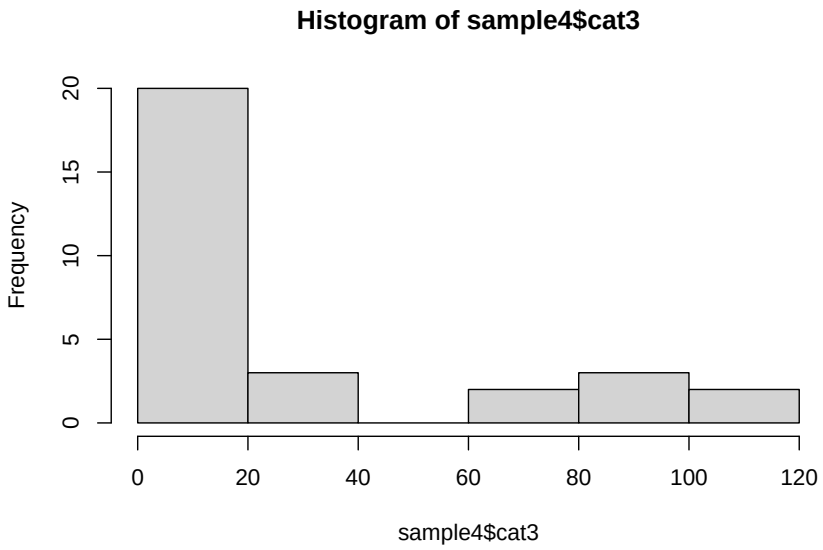

```
qqnorm(sample4$cat2); qqline(sample4$cat2); hist(sample4$cat2)
```



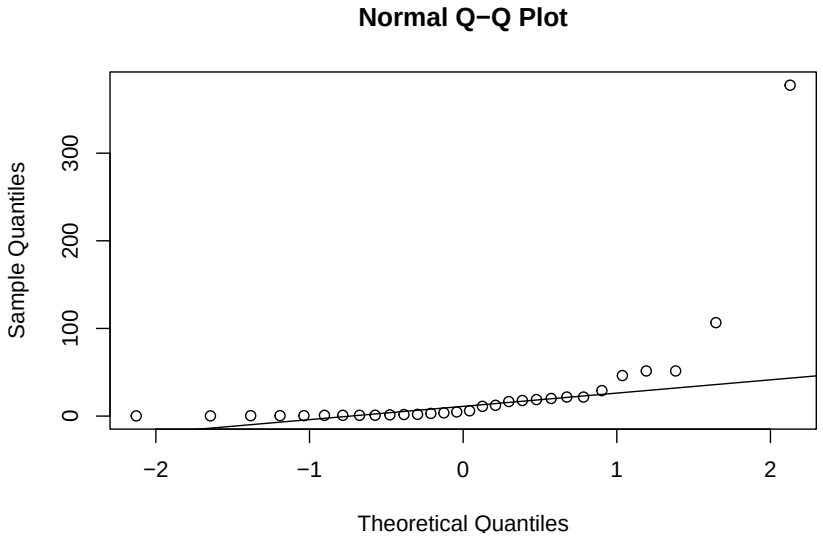


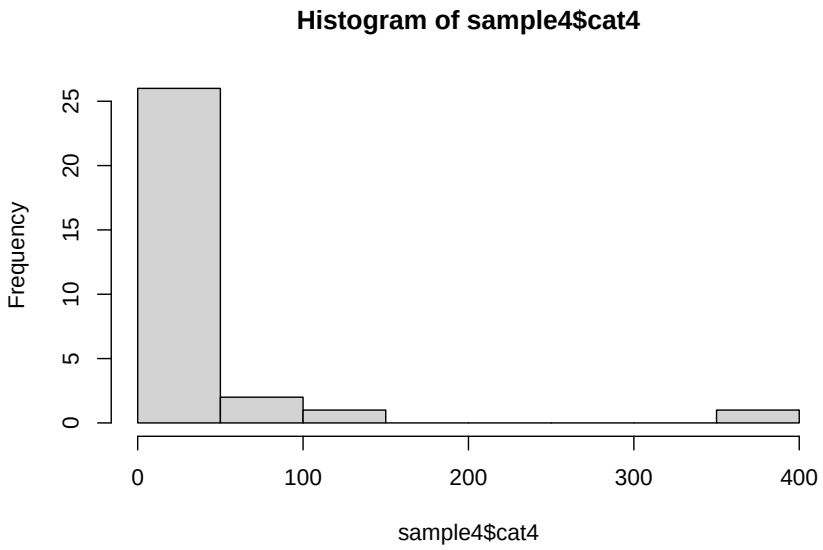

```
qqnorm(sample4$cat3); qqline(sample4$cat3); hist(sample4$cat3)
```






```
qqnorm(sample4$cat4); qqline(sample4$cat4); hist(sample4$cat4)
```





We want to test the hypotheses

$$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4 \quad \text{v.s.} \quad H_1 : \text{at least some means are not equal.}$$

ANOVA

```
fit = aov(values ~ cat, data = data)
summary(fit)[[1]][['Pr(>F)']][1] # p-value of one-way anova
```

```
## [1] 0.1007436
```

p-value is $0.10 > 0.05$, indicating that there's no significant difference in any pair of groups.

Permutation F-test

```
permut_F_test = function(values, group, sim_num = 10000){
  fit = lm(values ~ group)
  F_obs = summary(fit)[[10]][1]
  perm_F = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(group)
    fit = lm(values ~ permut)
    perm_F[i] = summary(fit)[[10]][1]
  }
  p = sum(perm_F >= F_obs) / sim_num
  return (p)
}
```

```r
permut_F_test(data$values, data$cat, sim_num = 100000)
```

```
## [1] 0.1
```

Though the data does not follow normal distribution, the permutation F-test has a similar p-value with Anova and leads to the same conclusion.

Fisher's Protected Least Significance Difference

Kruskal-Wallis Test

Define the function of Kruskal Wallis test.

```r
kruskal_test = function(values, group, sim_num = 10000){
  N = length(values)
  r <- rank(values)
  ti <- table(values)
  KW_obs <- (12 / (N * (N + 1)) *
               sum(tapply(rank(values), group, "length") *
               (tapply(rank(values), group, "mean") -
                  (N + 1) / 2) ^ 2) /
                (1 - sum(ti ^ 3 - ti) / (N ^ 3 - N)))
  # chi square approximation
  k = length(unique(group))
  p1 = 1 - pchisq(KW_obs, k - 1)
  # simulation of permutation test
  perm_KW = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(group)
    perm_KW[i] = (12 / (N * (N + 1)) *
                    sum(tapply(rank(values), permut, "length") *
                    (tapply(rank(values), permut, "mean") -
                       (N + 1) / 2) ^ 2) /
                    (1 - sum(ti ^ 3 - ti) / (N ^ 3 - N)))
  }
  p2 = sum(perm_KW >= KW_obs) / sim_num
  return (list(p_value_chisq_approximation = p1,
               p_value_monte_carlo_simulation_permutation = p2))
}
```

```r
kruskal_test(data$values, data$cat, sim_num = 100000)
```

```
## $p_value_chisq_approximation
## [1] 0.02311257
##
## $p_value_monte_carlo_simulation_permutation
## [1] 0.02191
```

The p-value is $0.023 < 0.05$, indicating that we should reject the null hypothesis, which is a totally different conclusion from ANOVA F-test and permutation F-test. This may be attributed to the heavy tail of the data.

Since the result is significant, we need to do multiple comparisons between means. Because of the non-normality, we should use rank-based methods in the following tests.

Define the function of two-sample pairwise comparisons, including LSD test and HSD test (using large sample approximation and Monte Carlo simulation).

```
multiple_test = function(values, group, sim_num = 10000){
  N = length(values)
  r = rank(values)
  k = length(unique(group))
  n = table(group)[1]
  mr = tapply(r, group, "mean")
  tmp = matrix(rep(mr, k), nrow = k)
  # Large sample approximation for LSD
  MSE = sum((n - 1) * tapply(r, group, "var")) / (N - k)
  Rij_obs = abs(t(tmp) - tmp) / sqrt(2 * MSE / n)
  p1 = 2 * (1 - pnorm(Rij_obs[upper.tri(Rij_obs)]))
  # Large sample approximation for Tukey's HSD
  Rij_obs = abs(t(tmp) - tmp) / sqrt(MSE / n)
  p2 = 1 - ptukey(Rij_obs[upper.tri(Rij_obs)], k, Inf)
  # Permutation LSD & HSD
  Tij_obs = abs(t(tmp) - tmp)
  perm_Tij = rep(0, sim_num)
  perm_Q = rep(0, sim_num)
  set.seed(2023)
  for (i in 1:sim_num){
    permut = sample(r)
    perm_Tij[i] = abs(mean(permut[1:n]) - mean(permut[(n + 1):(2 * n)]))
    perm_Q[i] = diff(range(tapply(permut, group, "mean")))
  }
  ecdf_Tij = ecdf(perm_Tij)
  p3 = 1 - ecdf_Tij(Tij_obs[upper.tri(Tij_obs)])
  ecdf_Q = ecdf(perm_Q)
  p4 = 1 - ecdf_Q(Tij_obs[upper.tri(Tij_obs)])
  # Output
  res = cbind(which(upper.tri(Rij_obs), arr.ind = TRUE),
              p_LSD_approx = p1,
              p_LSD_permut = p3,
              p_HSD_approx = p2,
              p_HSD_permut = p4)
  return(res)
}
```

The p-values are as follows.

```
multiple_test(data$values, data$cat, sim_num = 50000)
```

```
##      row col p_LSD_approx p_LSD_permut p_HSD_approx p_HSD_permut
## [1,]   1   2   0.89967465      0.90666   0.99928455      0.99946
## [2,]   1   3   0.01463777      0.01864   0.06954963      0.08380
## [3,]   2   3   0.02060490      0.02532   0.09455944      0.11150
## [4,]   1   4   0.03118602      0.03728   0.13611495      0.15900
## [5,]   2   4   0.04249714      0.04968   0.17740913      0.20226
## [6,]   3   4   0.77447370      0.78526   0.99180629      0.99276
```

Fisher's LSD

Since the number of samples in each group is the same in this problem, i.e. $n_i = n = 30, i = 1, \ldots, 4$, we declare the distributions of treatments i and j to be different if

$$\left|\bar{R}_i - \bar{R}_j\right| \geq z_{1-\frac{\alpha}{2}} \sqrt{\text{MSE}_{\text{rank}} \frac{2}{n}}.$$

We compare the p-value of LSD with the significance level 0.05, and find that (1, 3), (2, 3), (1, 4), (2, 4) have significant difference. The permutation LSD test has similar p-values and leads to the same conclusion.

Tukey's HSD

We declare the distributions of treatments i and j to be different if

$$\left|\bar{R}_i - \bar{R}_j\right| \geq q_{1-\alpha}(k, \infty) \sqrt{\text{MSE}_{\text{rank}} \frac{1}{n}}.$$

No pairwise comparison is significant. The permutation HSD test has similar p-values and leads to the same conclusion.

Bonferroni

We carry out two-sample test for each comparison at adjusted significance level

$$\alpha^* = \frac{\alpha}{\binom{k}{2}},$$

where α is the overall experimental significance level, k is the number of group.

In this analysis, we need to compare the p-value with 0.008.

```
alpha_star = 0.05 / choose(4, 2)
alpha_star
```

```
## [1] 0.008333333
```

Define the function of pairwise comparisons for k groups.

```
pairwise_test = function(values, group, sim_num = 10000){
  N = length(values)
  r = rank(values)
  k = length(unique(group))
  n = table(group)[1]
  res = which(upper.tri(matrix(NA, k, k)), arr.ind = TRUE)
  # Rank-based permutation test & Wilcoxon Rank-sum Test
  p1 = rep(0, nrow(res))
  p2 = rep(0, nrow(res))
  p3 = rep(0, nrow(res))
  for (j in 1:nrow(res)){
    r_idx = res[j, 1]
    c_idx = res[j, 2]
```

```
    v = r[group == r_idx | group == c_idx]
    g = group[group == r_idx | group == c_idx]
    # Simulation of permutation test
    Rij_obs = abs(diff(tapply(v, g, "mean")))
    perm_Rij = rep(0, sim_num)
    set.seed(2023)
    for (i in 1:sim_num){
      permut = sample(g)
      perm_Rij[i] = abs(diff(tapply(v, permut, "mean")))
    }
    p1[j] = sum(perm_Rij >= Rij_obs) / sim_num
    # Wilcoxon rank sum test
    p_w = wilcoxon_test(values[group == r_idx],
                        values[group == c_idx],
                        sim_num = sim_num)
    p2[j] = p_w[[1]]
    p3[j] = p_w[[2]]
  }
  # Output
  res = cbind(res, p_diff_permut = p1,
              p_wilcox_approx = p2,
              p_wilcox_permut = p3)
  return (res)
}
```

```
pairwise_test(data$values, data$cat, sim_num = 50000)
```

```
##      row col p_diff_permut p_wilcox_approx p_wilcox_permut
## [1,]   1   2       0.89440      0.89413437         0.89084
## [2,]   1   3       0.02120      0.02026631         0.01912
## [3,]   2   3       0.02808      0.02758815         0.02656
## [4,]   1   4       0.03006      0.03448045         0.03448
## [5,]   2   4       0.04264      0.04433545         0.04568
## [6,]   3   4       0.79008      0.72269394         0.72660
```

All the p-values are larger than 0.008, so no pairwise comparison is significant, which is the same as the conclusion of Tukey's HSD.