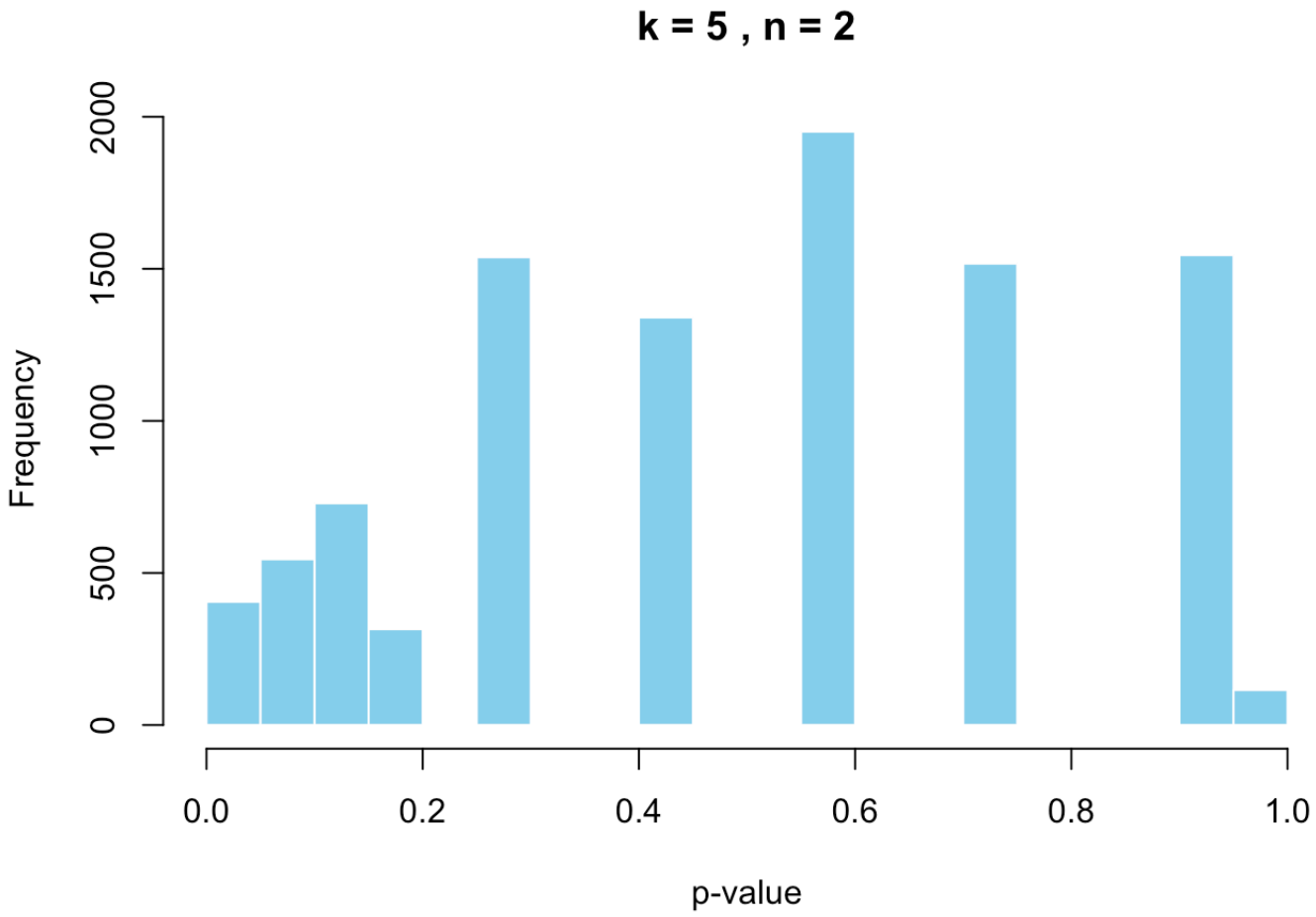
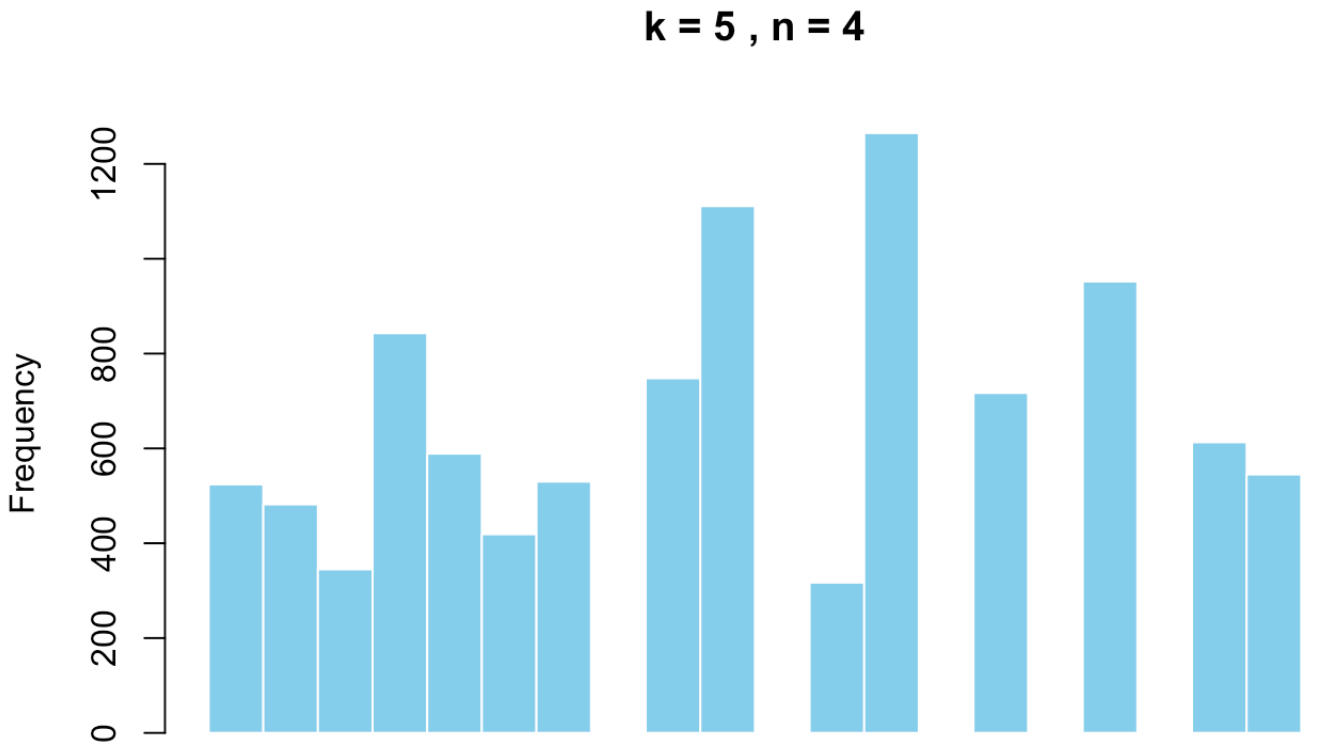
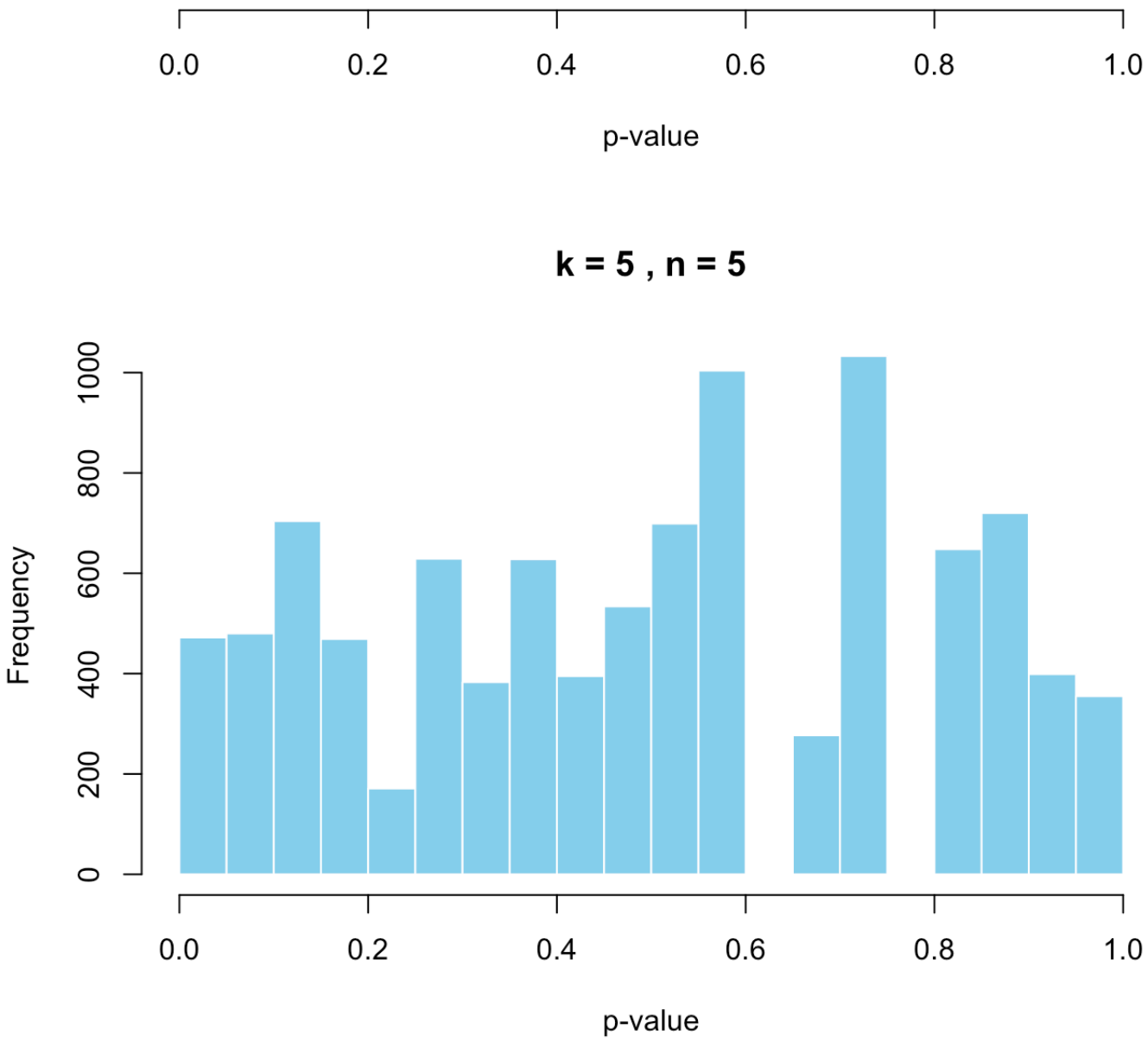
HW_2

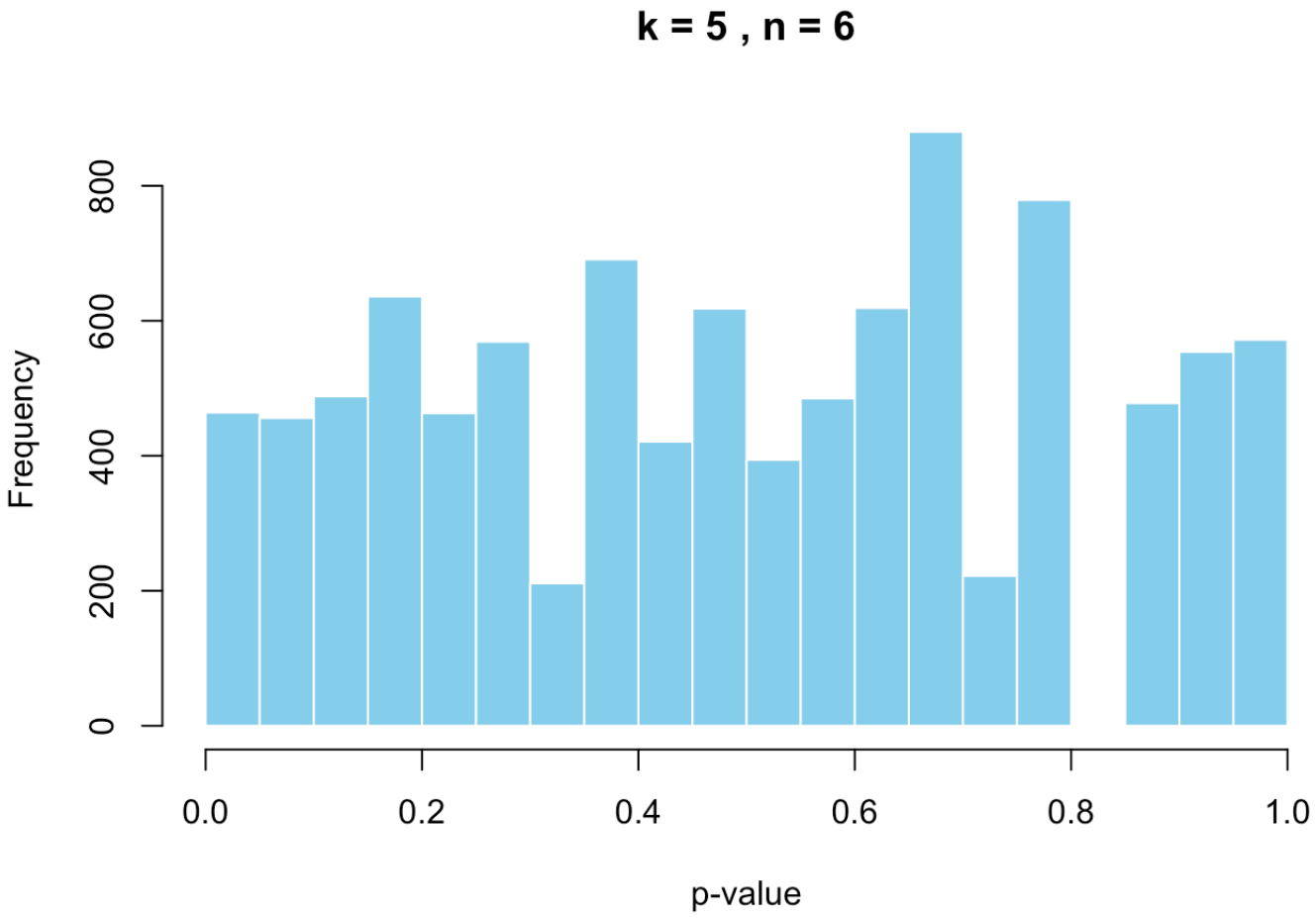
2024-04-17

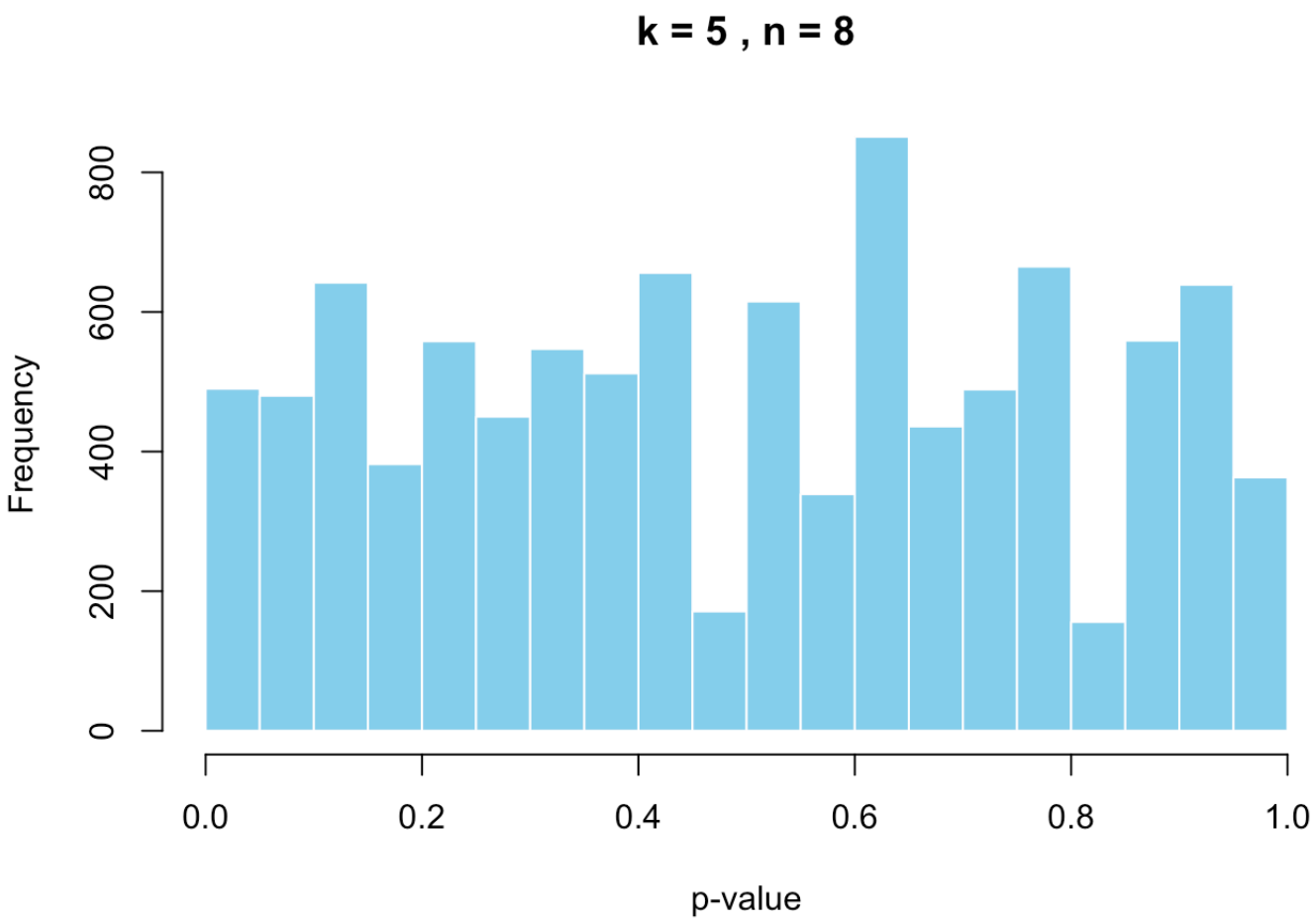
```
k <- c(5, 10, 20, 50, 100)
n <- c(2, 4, 5, 6, 8, 10)

p_values <- data.frame()
for (kk in k){
  for (nn in n){
    sample_size = nn * kk
    all_P <- c()
    # repeat 10000 times
    for (i in 1:10000){
      r <- sample(1:kk, size = sample_size, replace = TRUE, prob=rep(1/kk, kk))
      # Pearson Test
      result <- suppressWarnings({chisq.test(table(factor(r, levels =1:kk)))})

      # record p value
      p <- result$p.value
      all_P <- c(all_P, p)
    }

    # todo, plot the all_P
    hist(all_P, main = paste("k =", kk, ", n =", nn), xlab = "p-value", col = "skyblu
e", border = "white")
  }
}
```

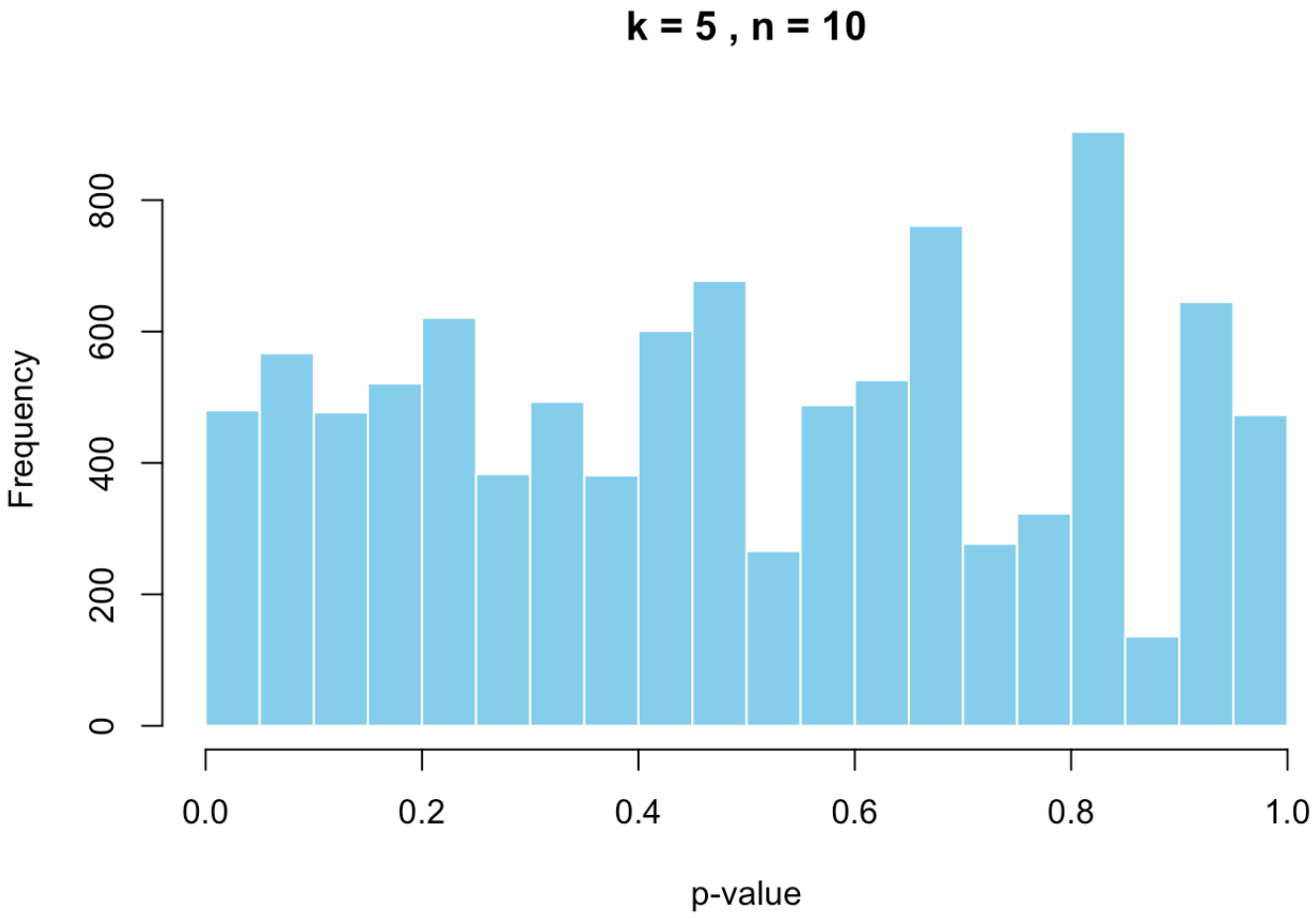

k = 5 , n = 2
Frequency
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0
p-value

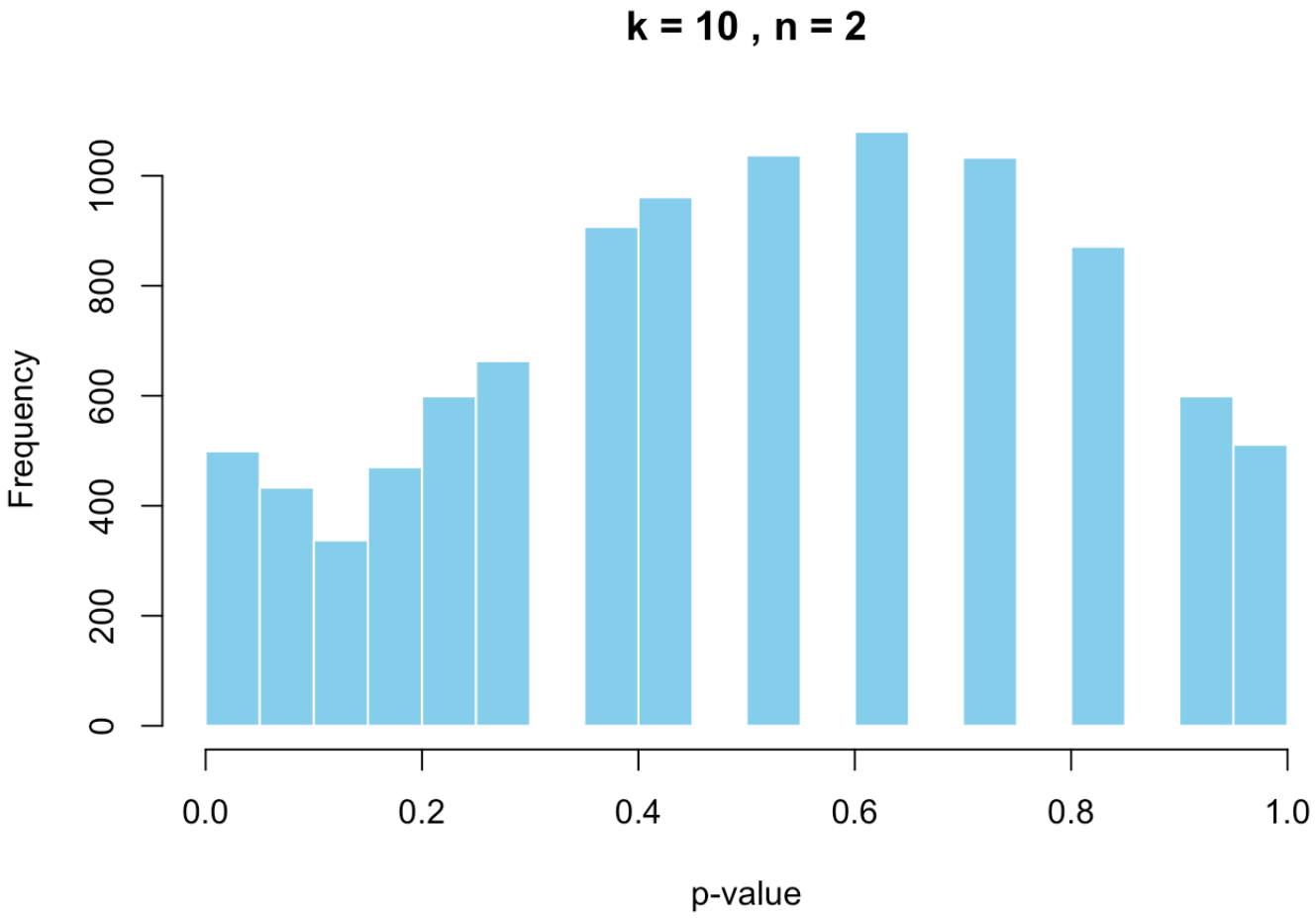

k = 5 , n = 4
Frequency
0
200
400
600
800
1200

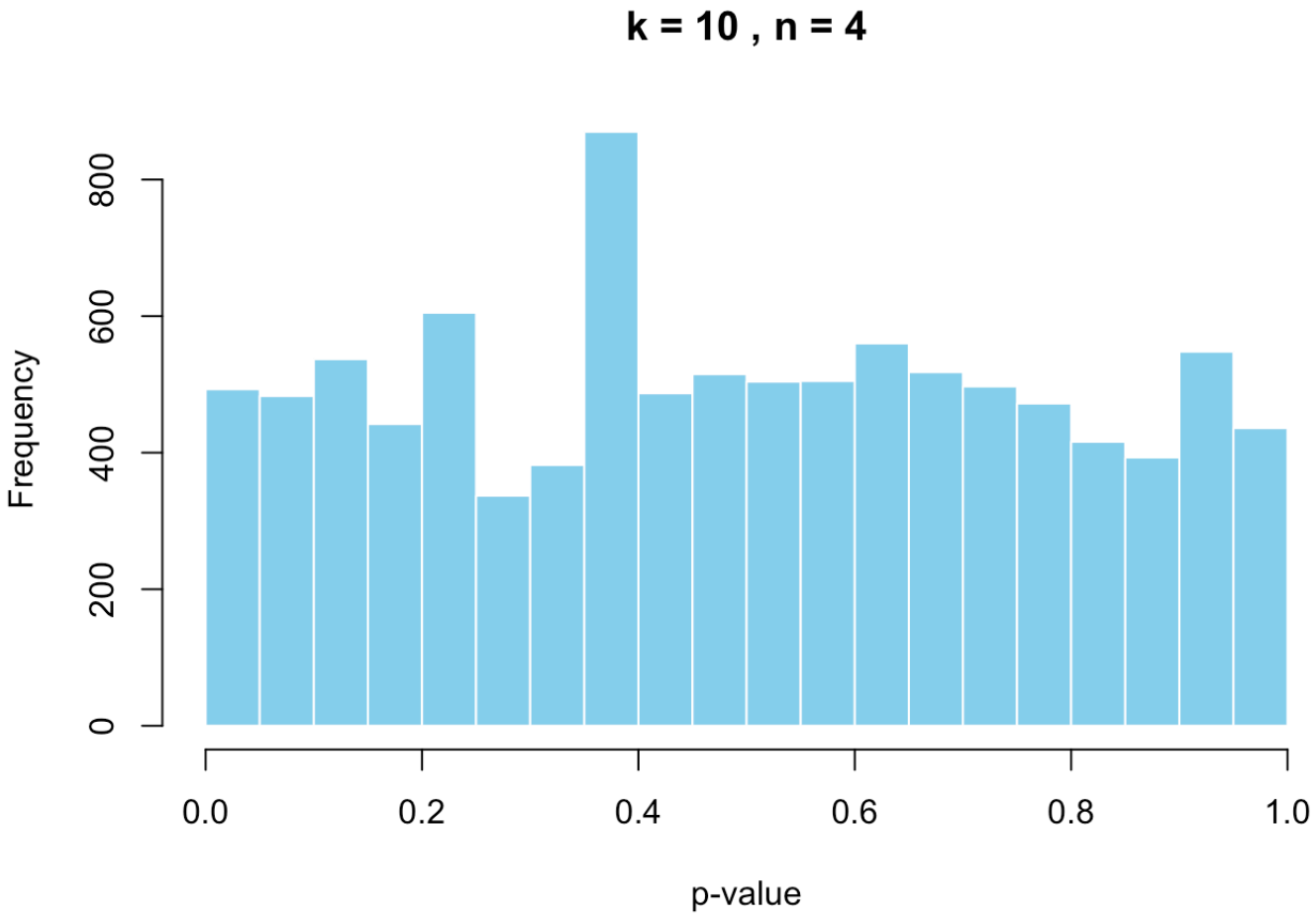

0.0
0.2
0.4
0.6
0.8
1.0
p-value
k = 5 , n = 5
Frequency
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0
p-value

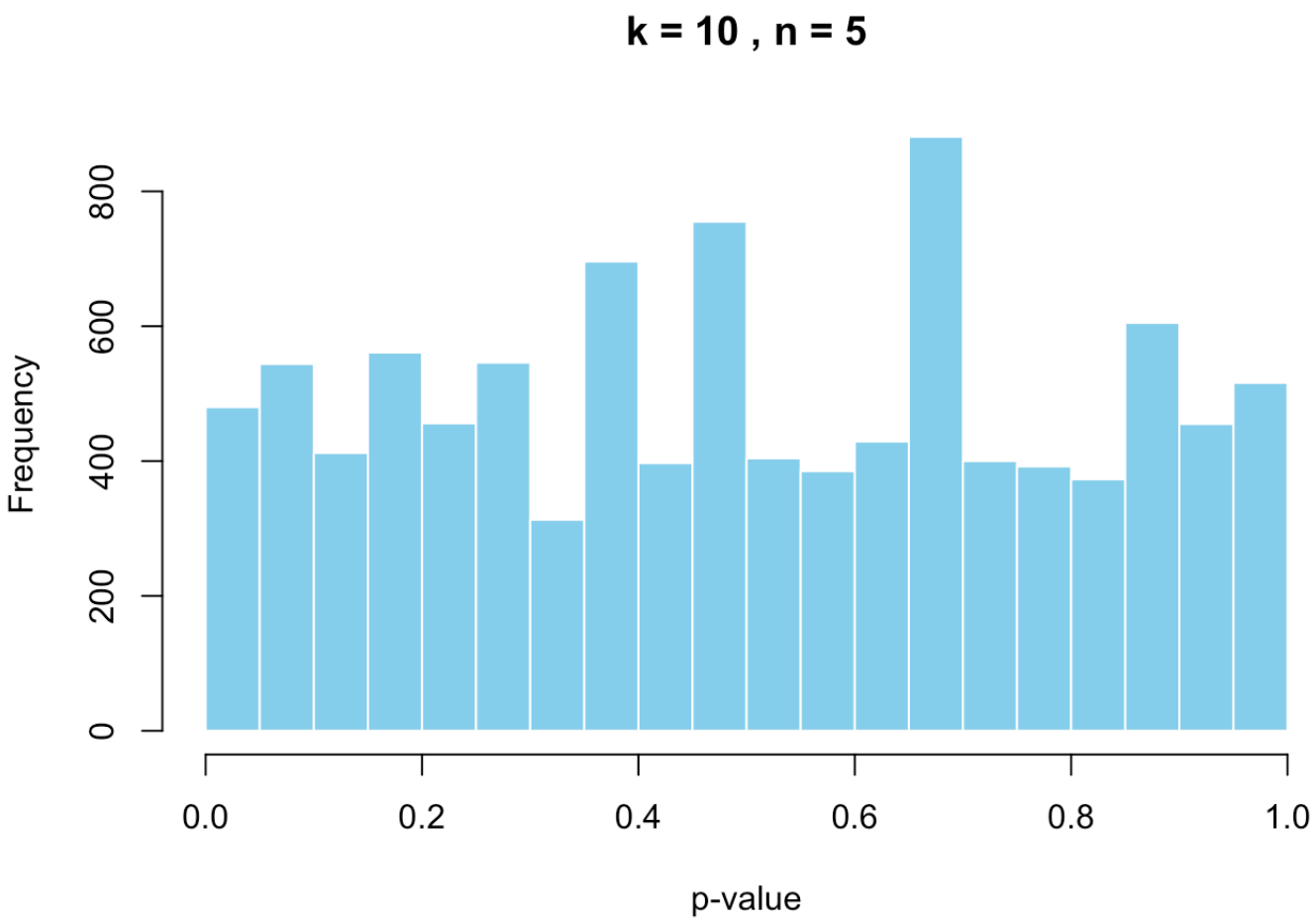

k = 5 , n = 6
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

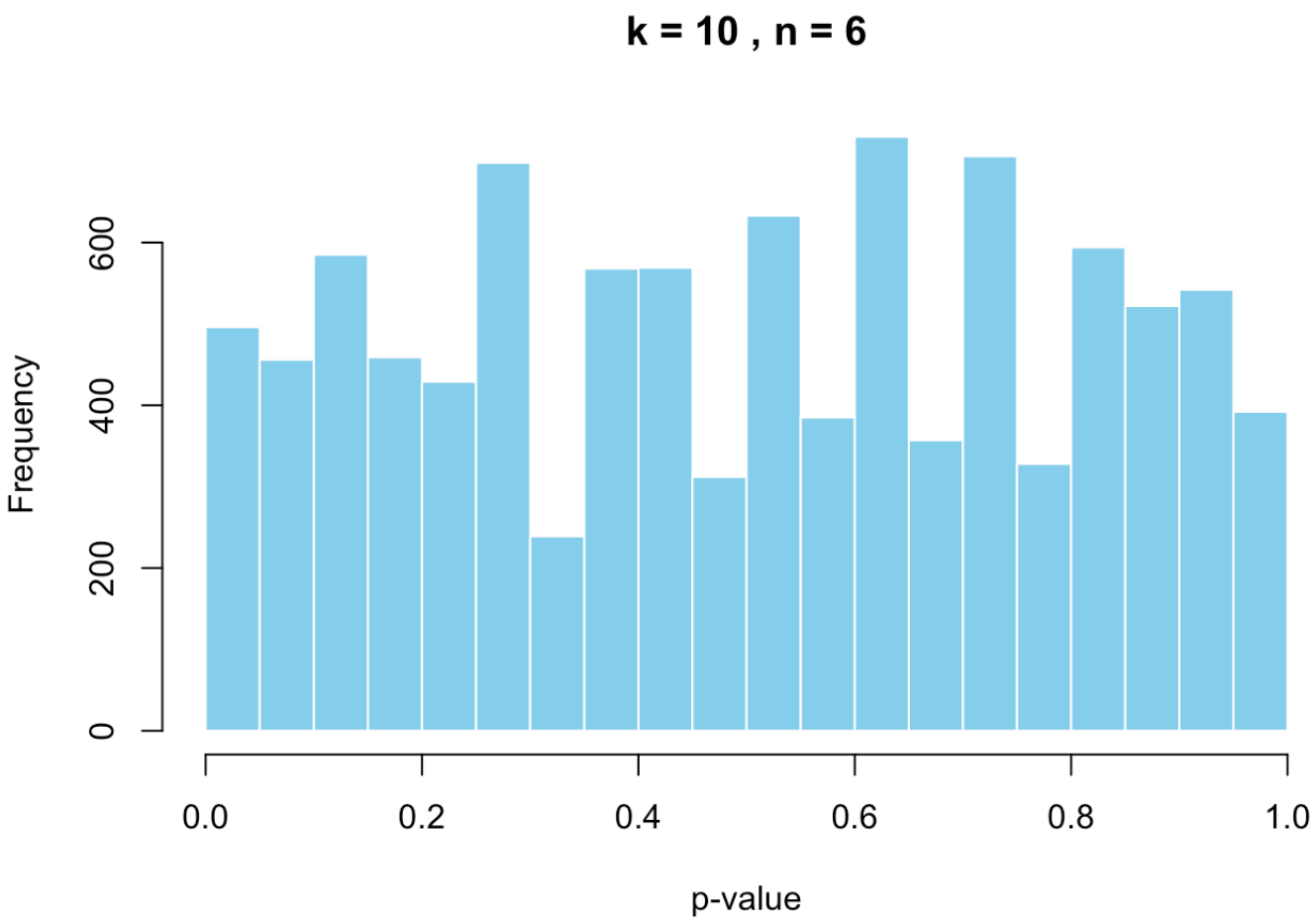

k = 5 , n = 8
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

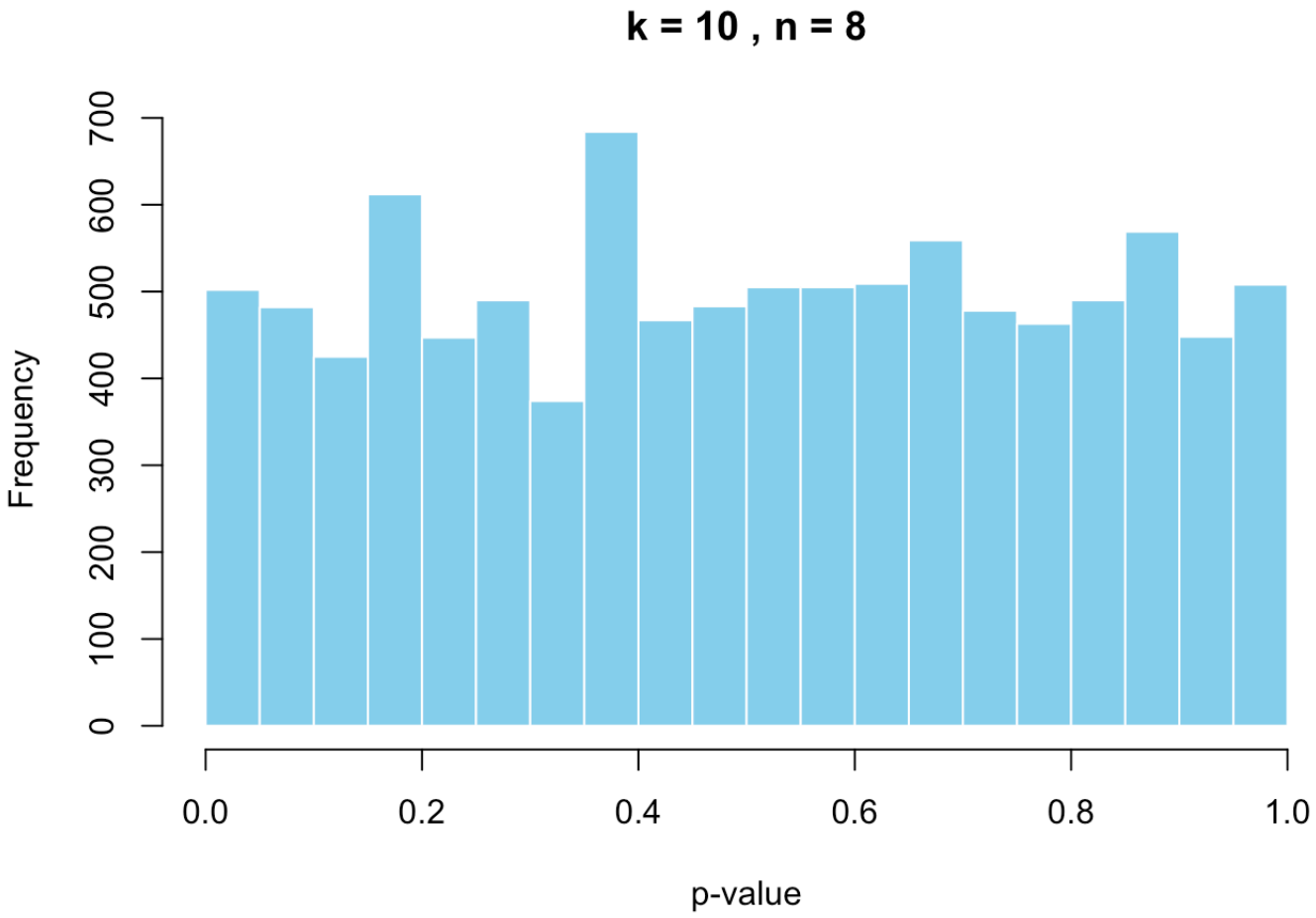

k = 5 , n = 10
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

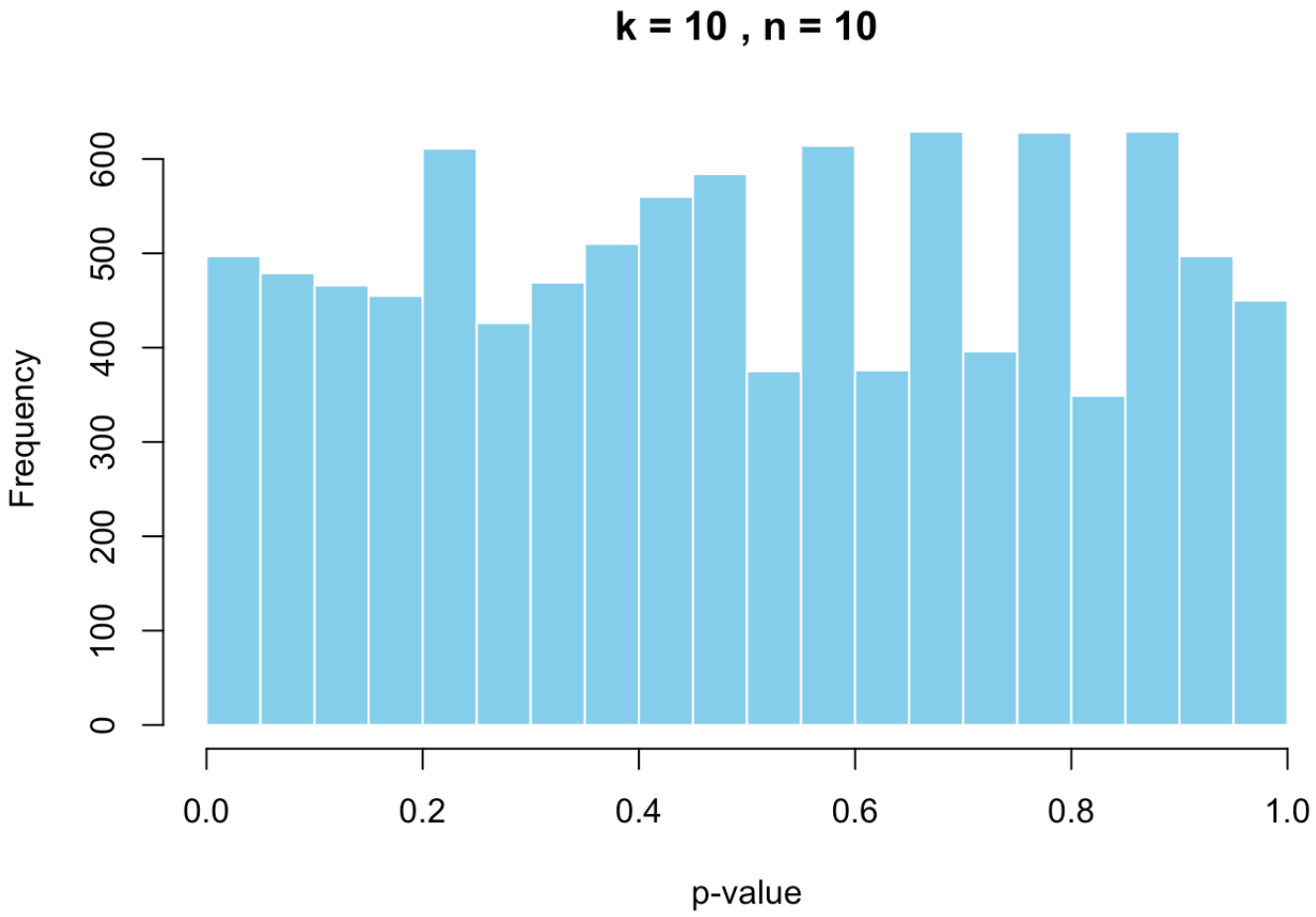

k = 10 , n = 2
Frequency
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0
p-value

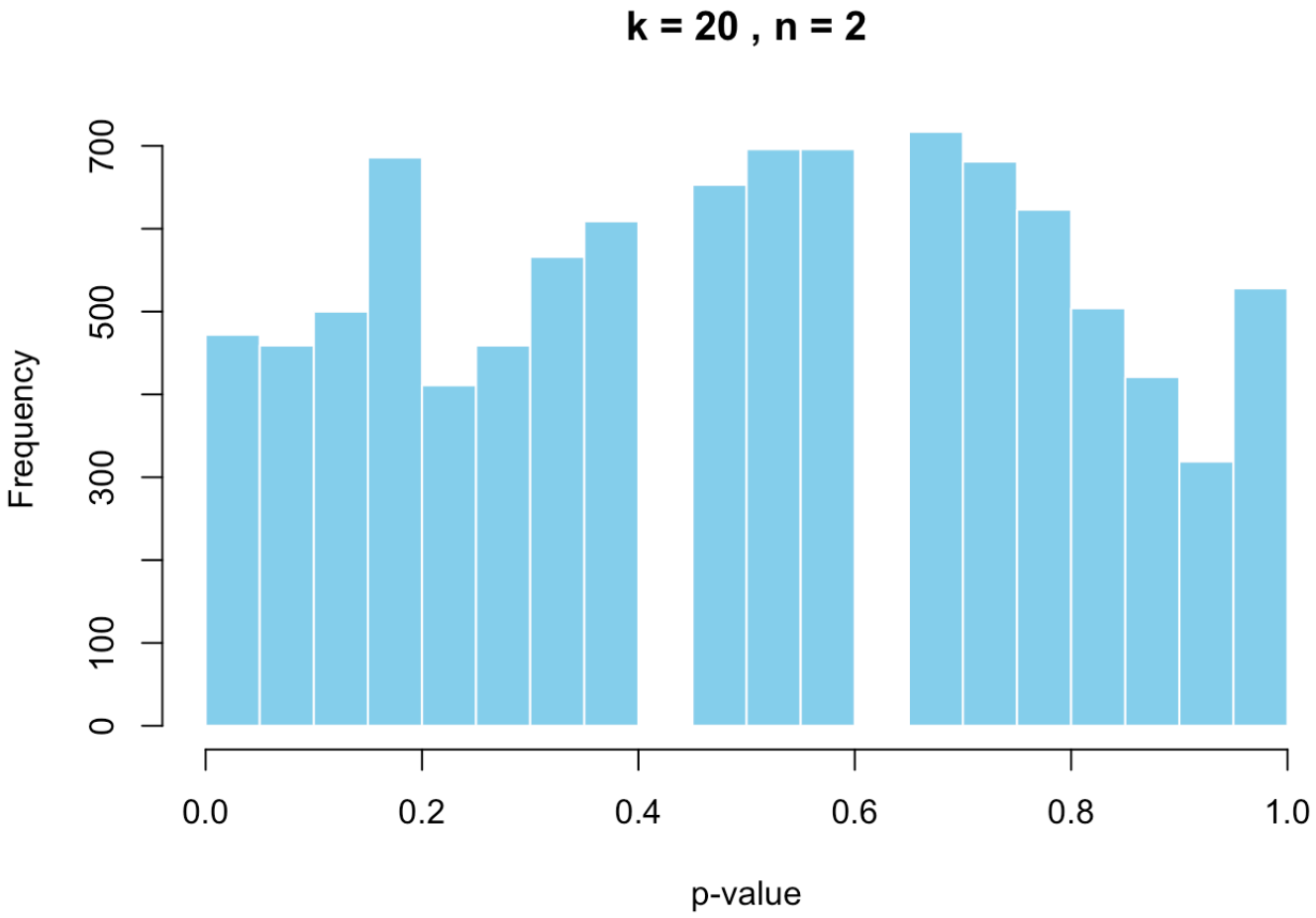

k = 10 , n = 4
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

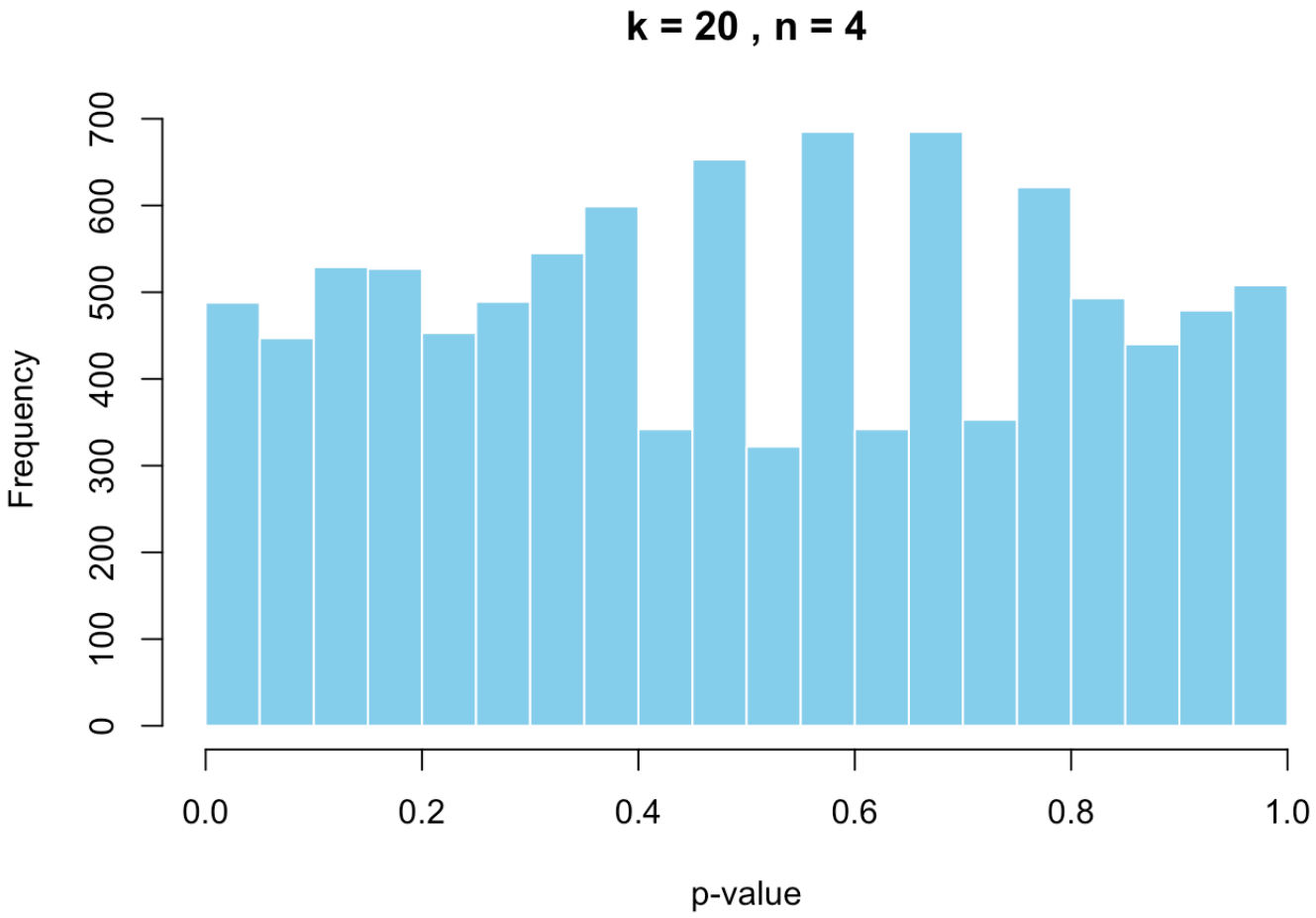

k = 10 , n = 5
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

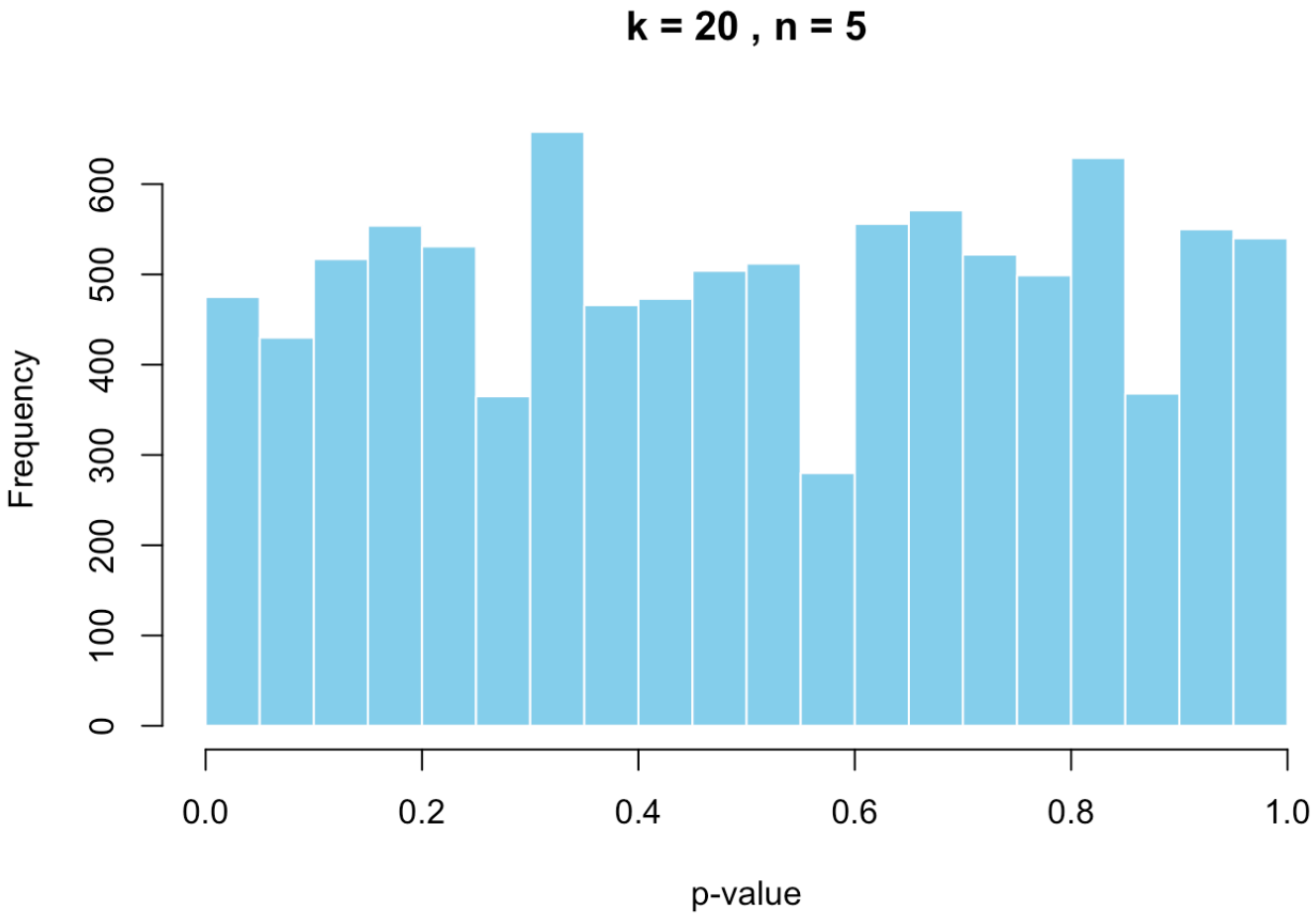

k = 10 , n = 6
Frequency
0
200
400
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

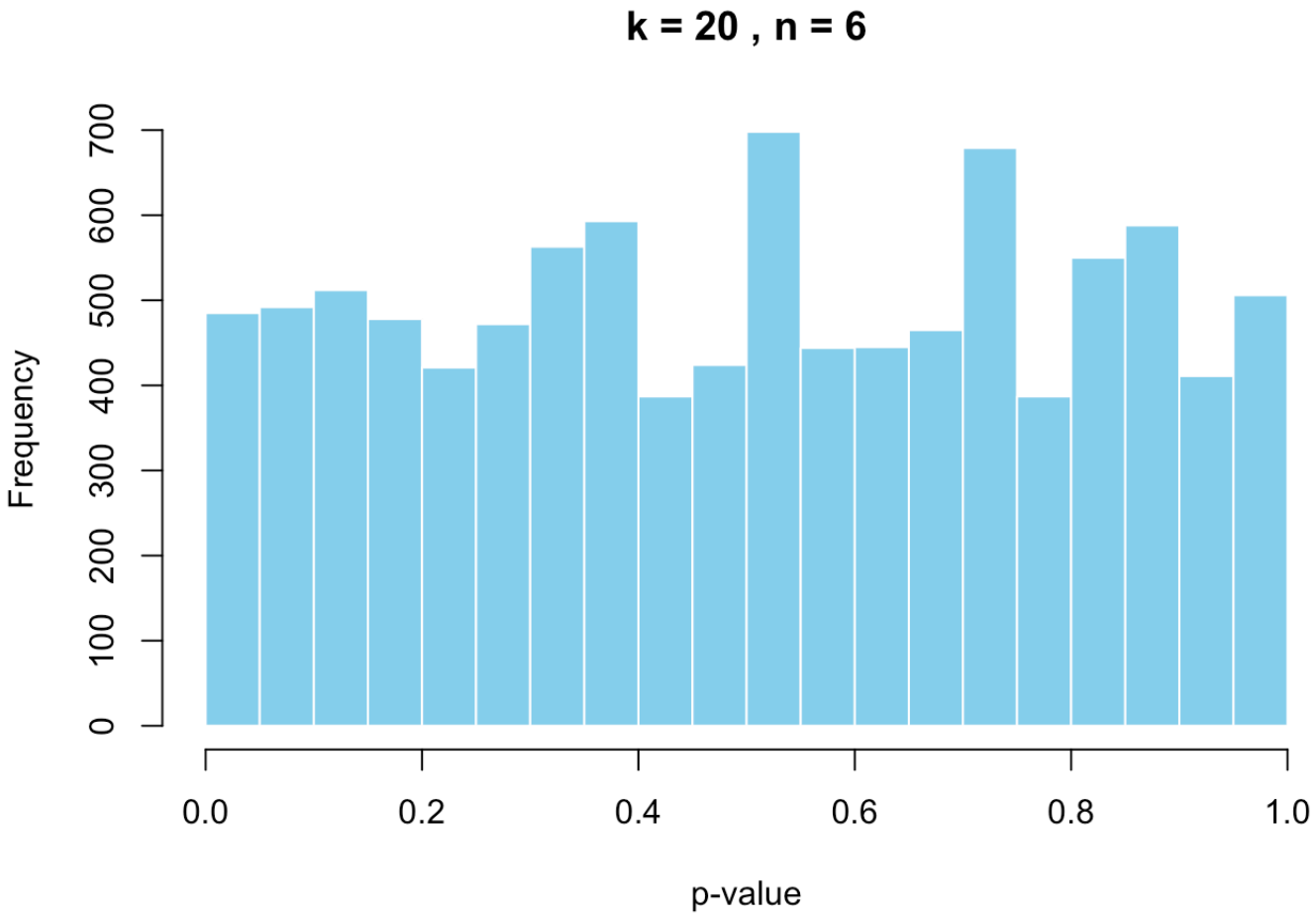

k = 10 , n = 8
Frequency
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
p-value

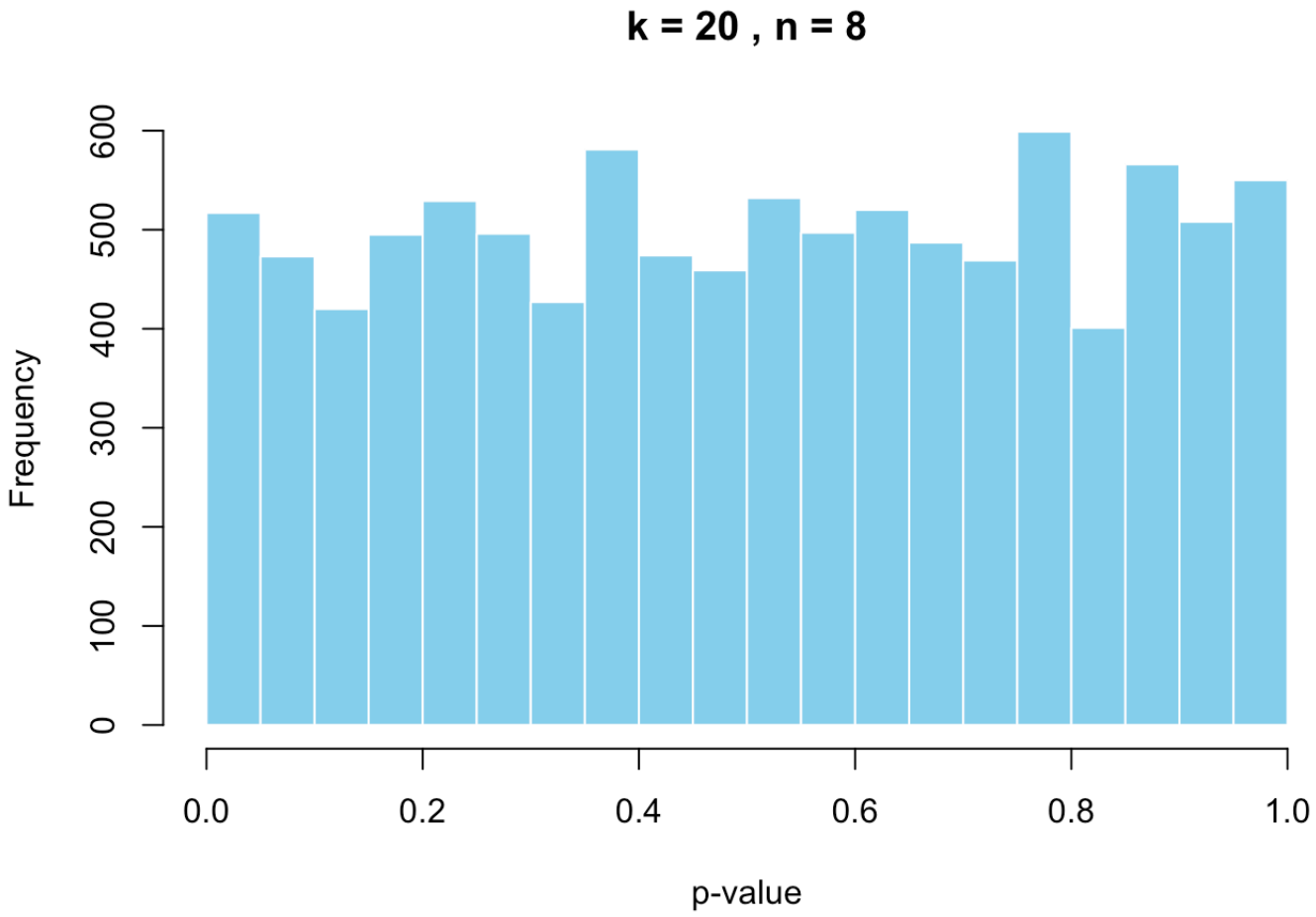

k = 10 , n = 10
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

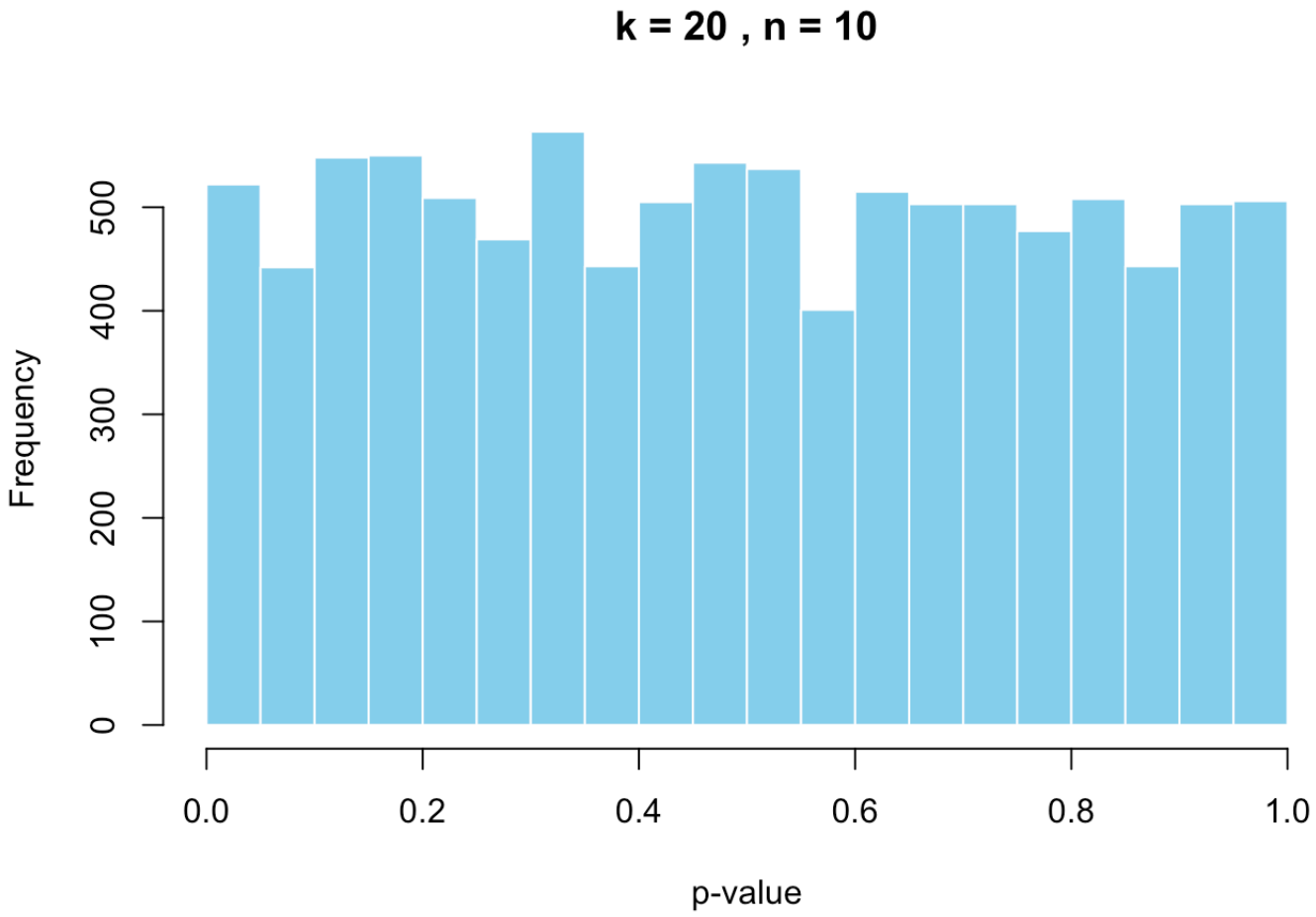

k = 20 , n = 2
Frequency
0
100
300
500
700
0.0
0.2
0.4
0.6
0.8
1.0
p-value

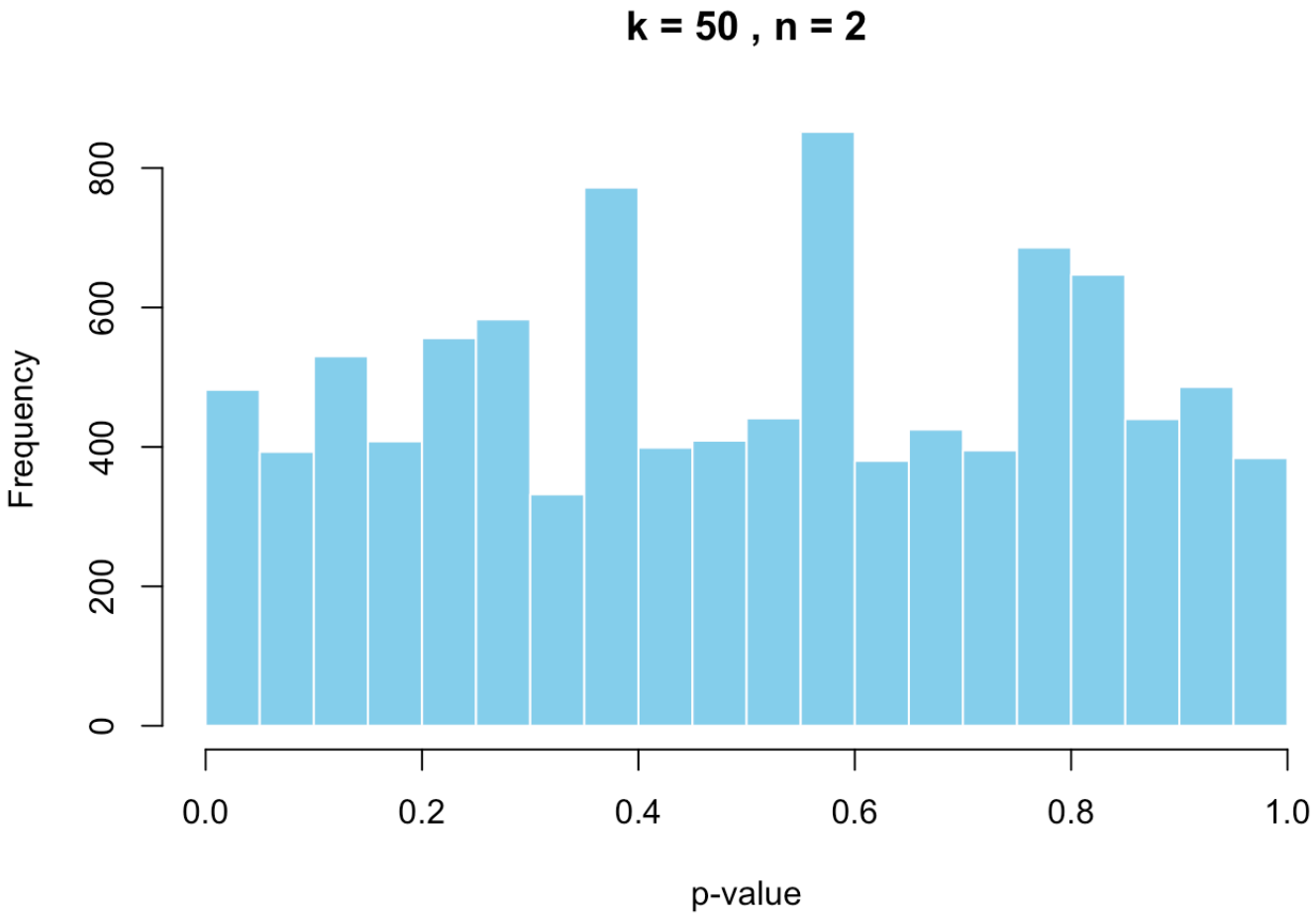

k = 20 , n = 4
Frequency
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
p-value

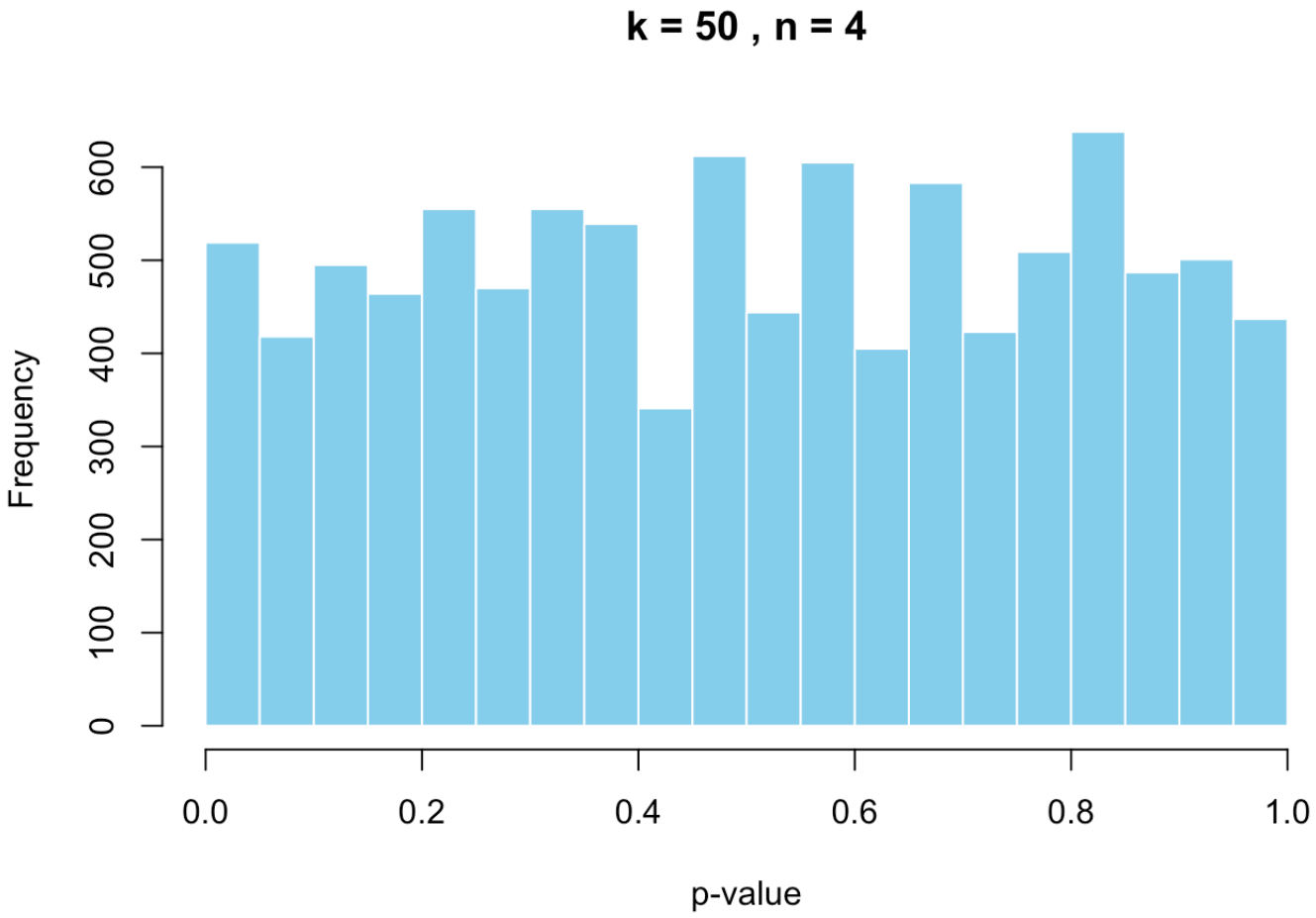

k = 20 , n = 5
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

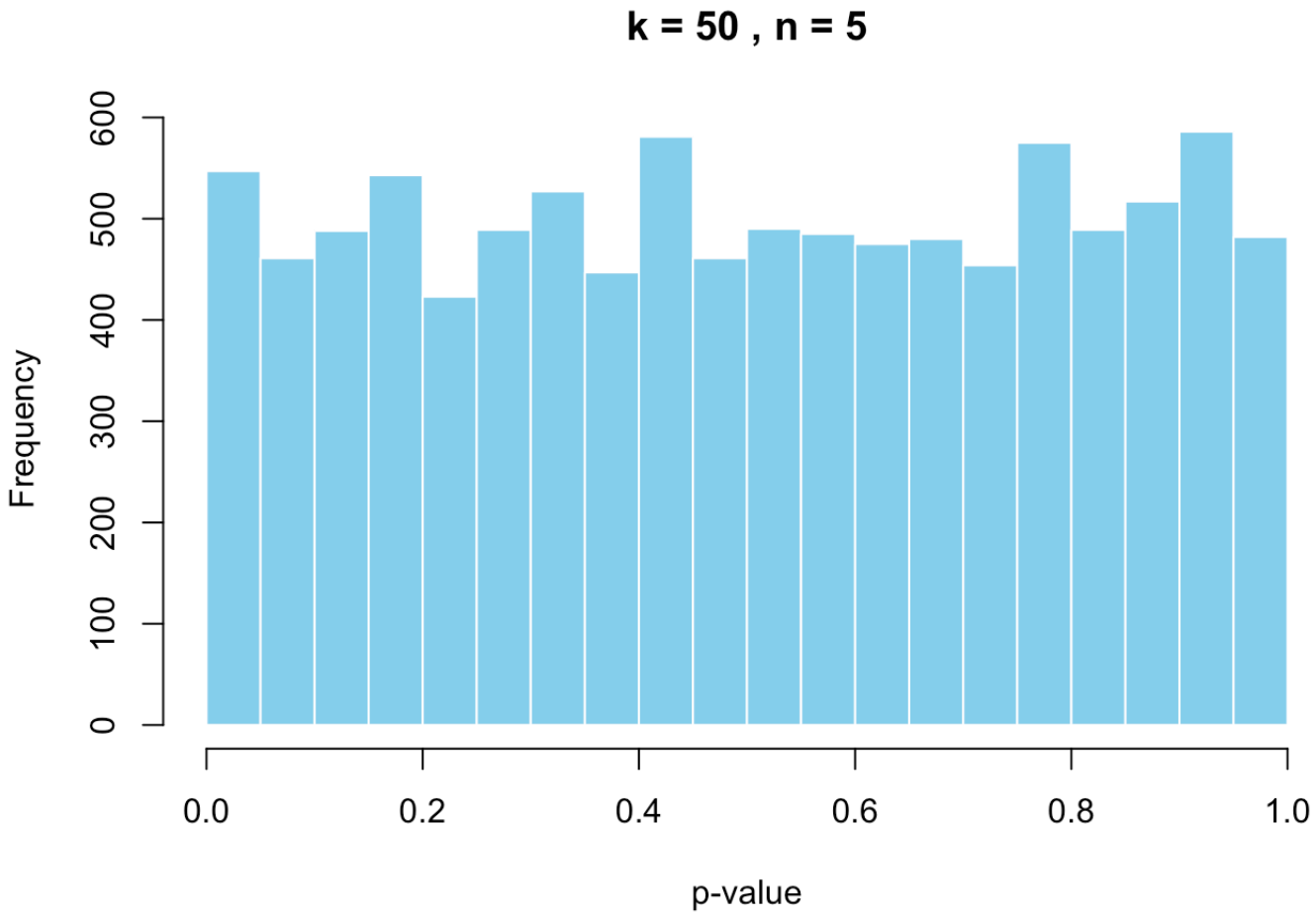

k = 20 , n = 6
Frequency
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
p-value

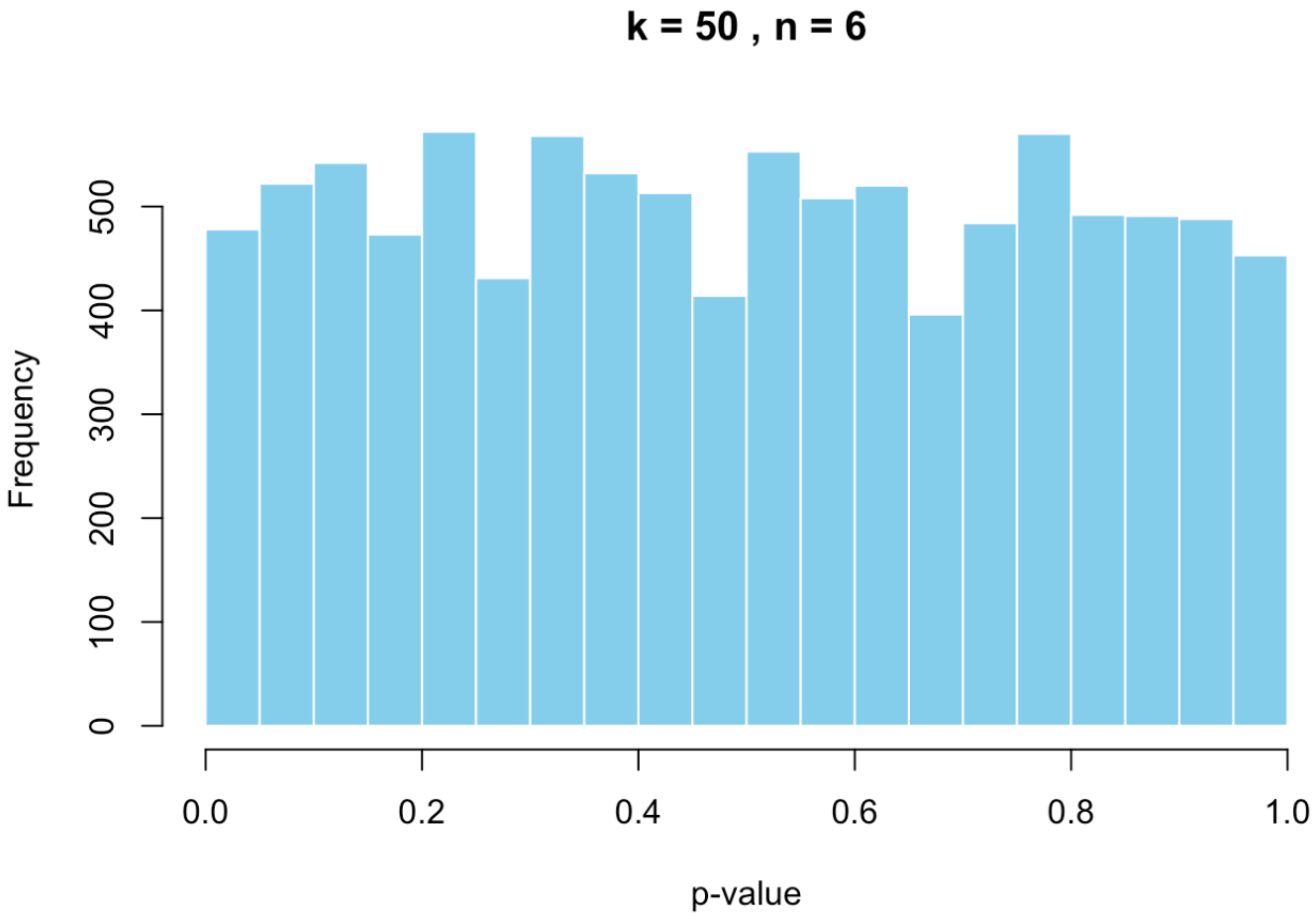

k = 20 , n = 8
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

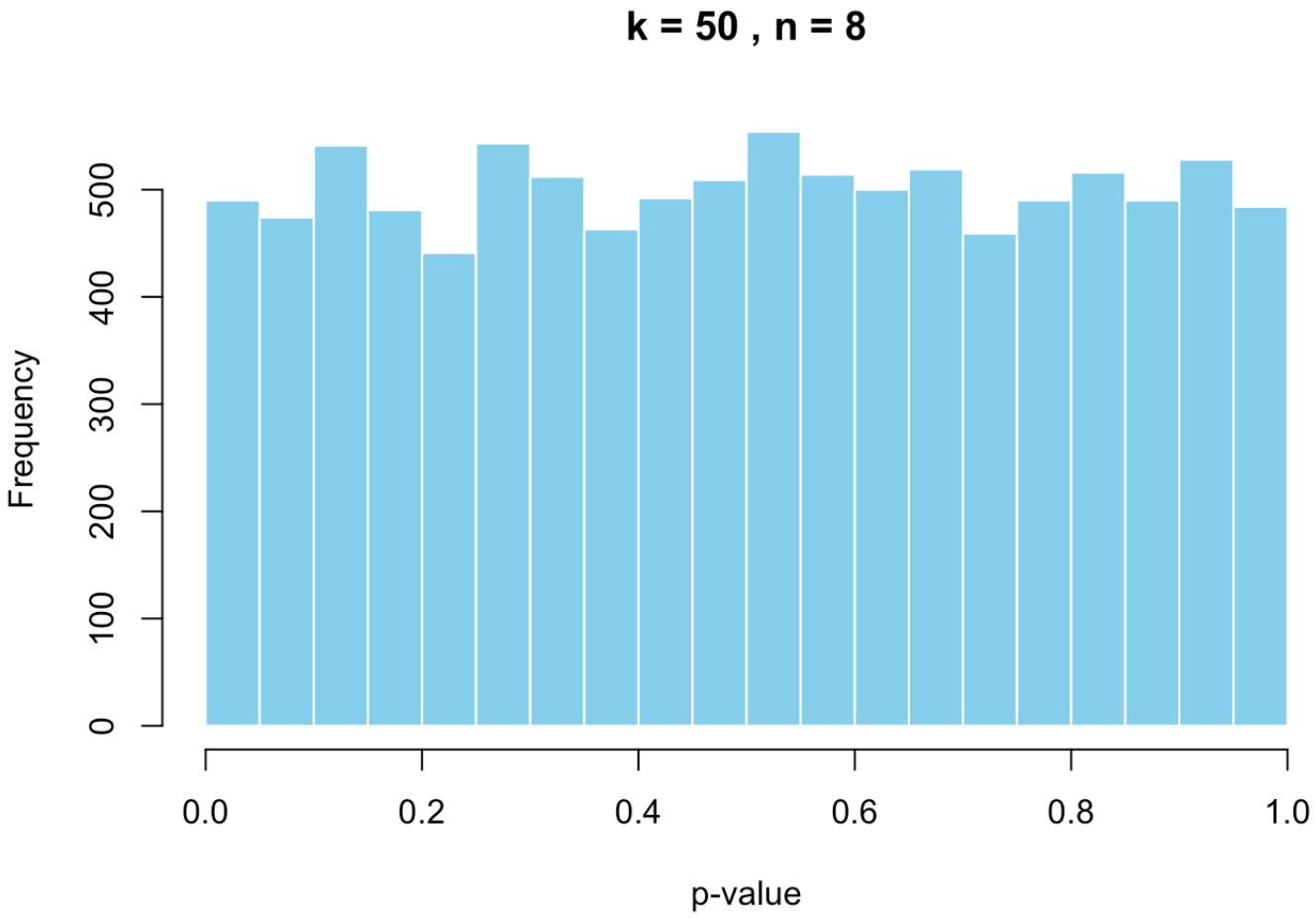

k = 20 , n = 10
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
p-value

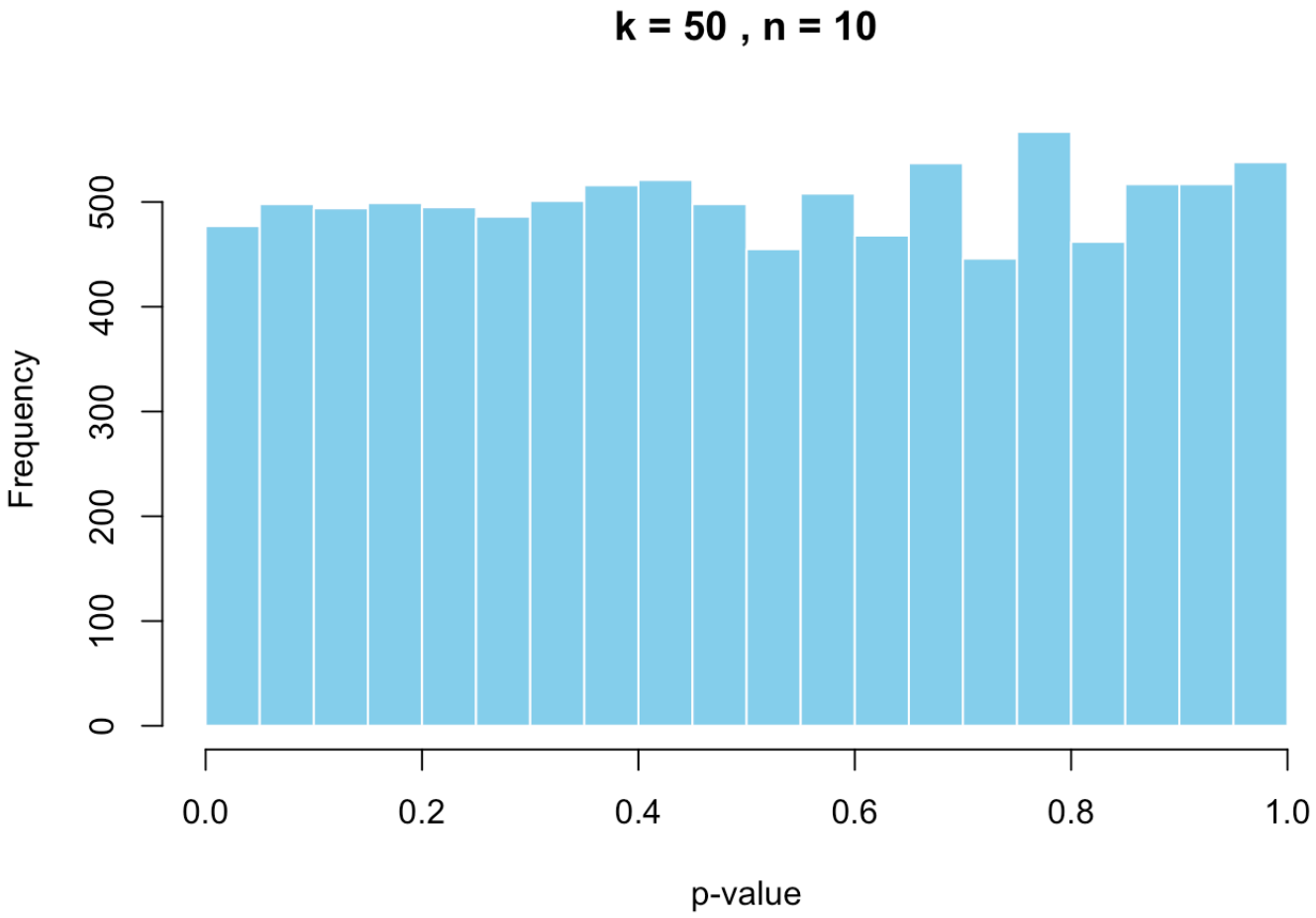

k = 50 , n = 2
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
p-value

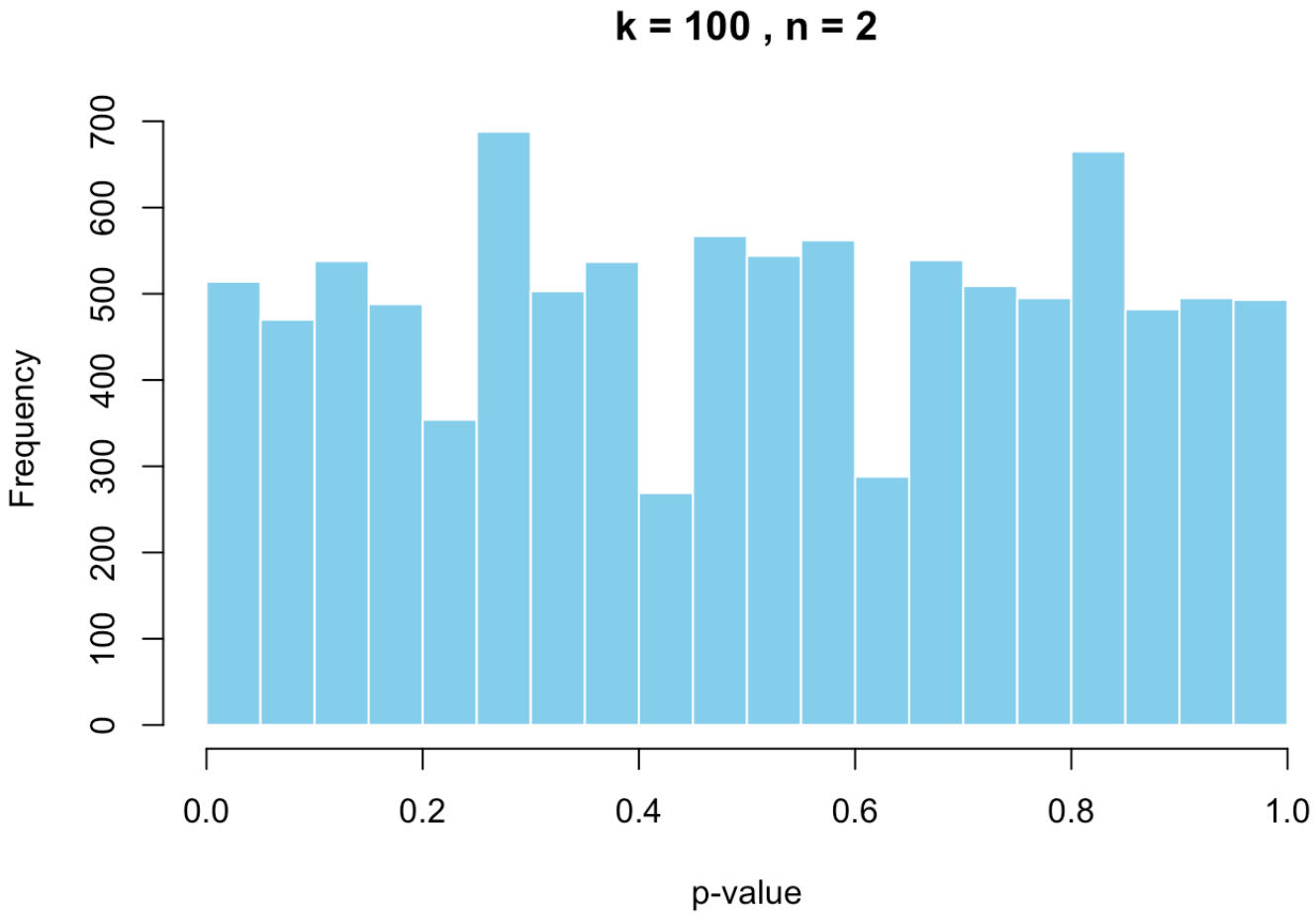

k = 50 , n = 4
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

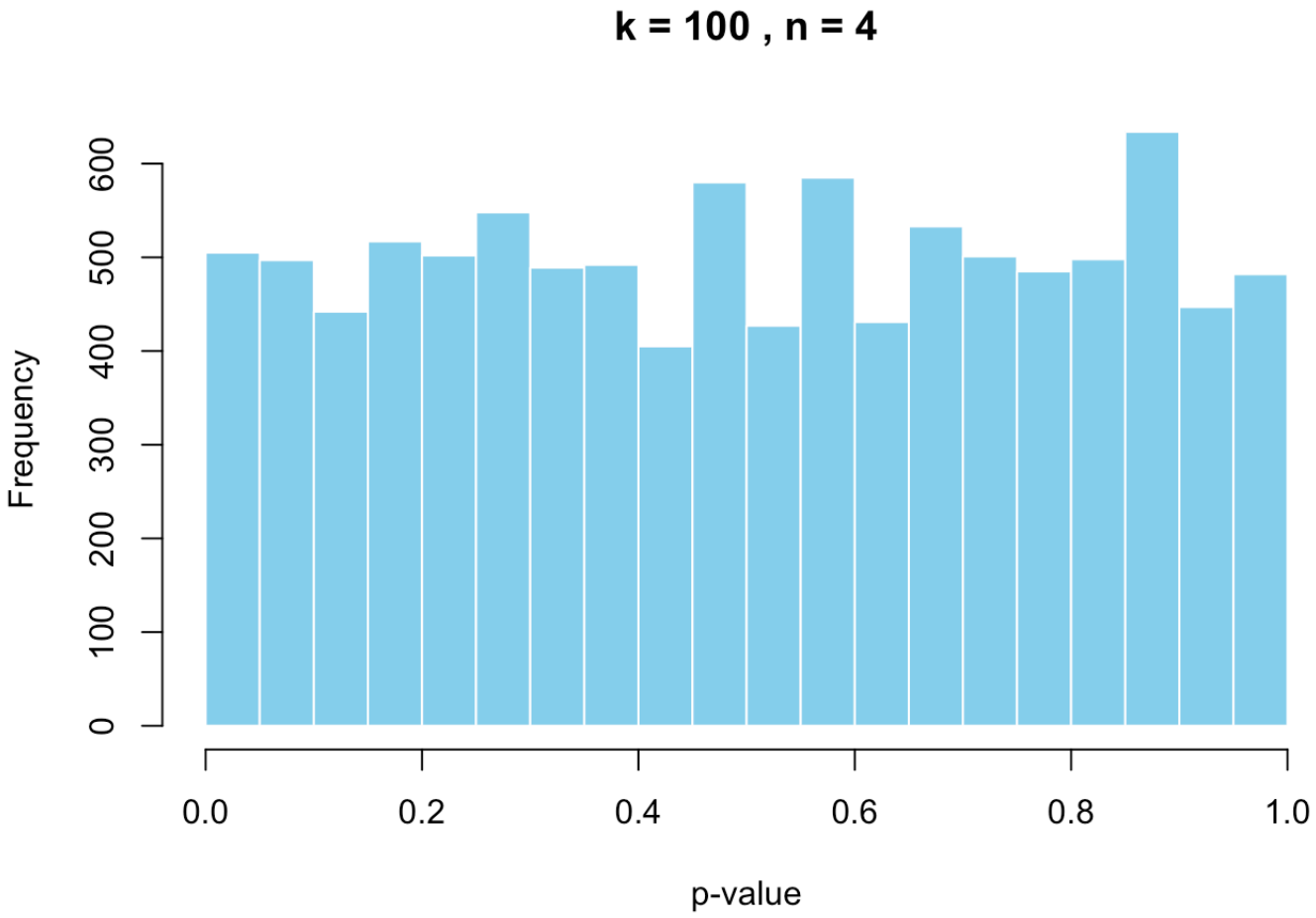

k = 50 , n = 5
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value

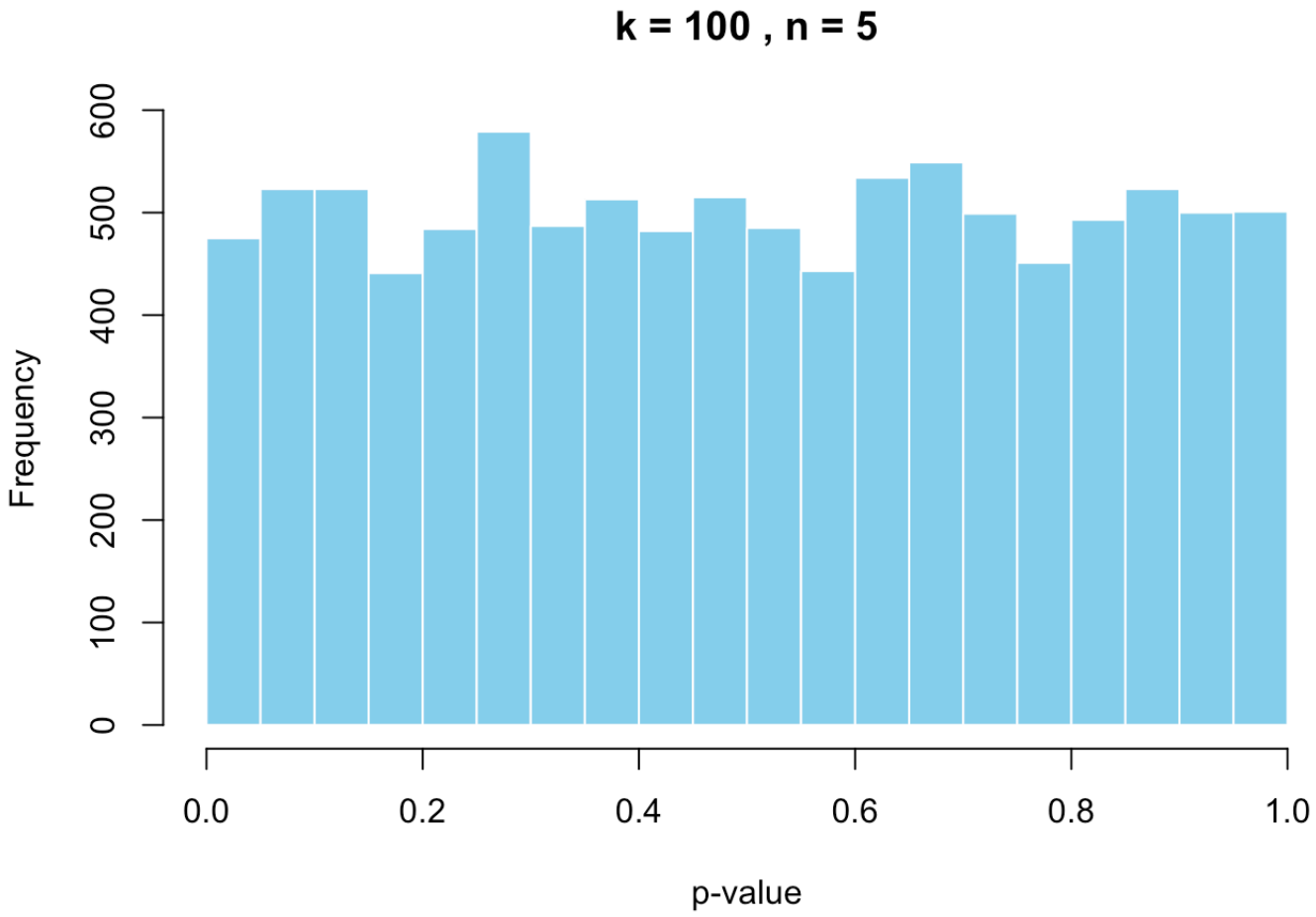

k = 50 , n = 6
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
p-value

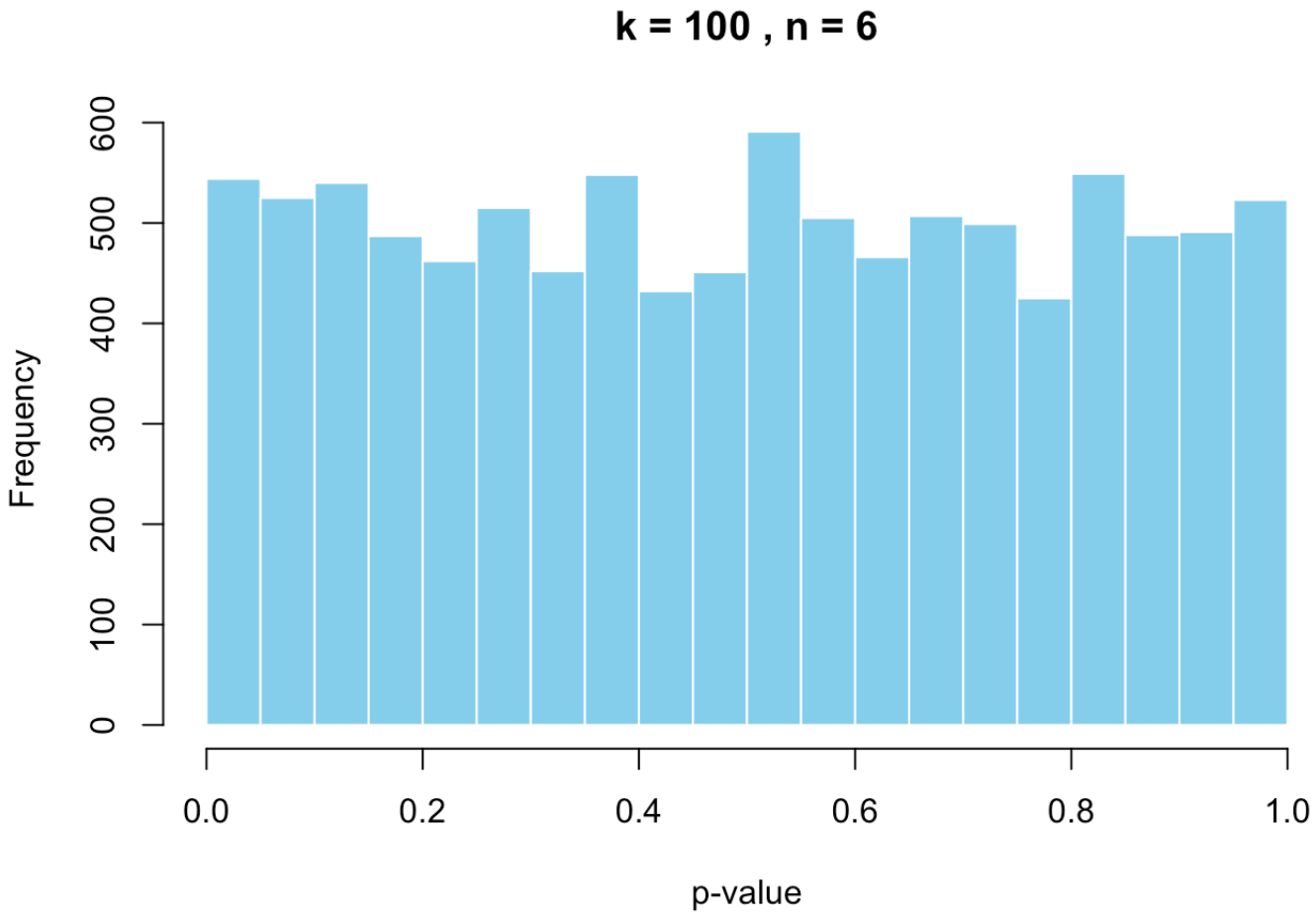

k = 50 , n = 8
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
p-value

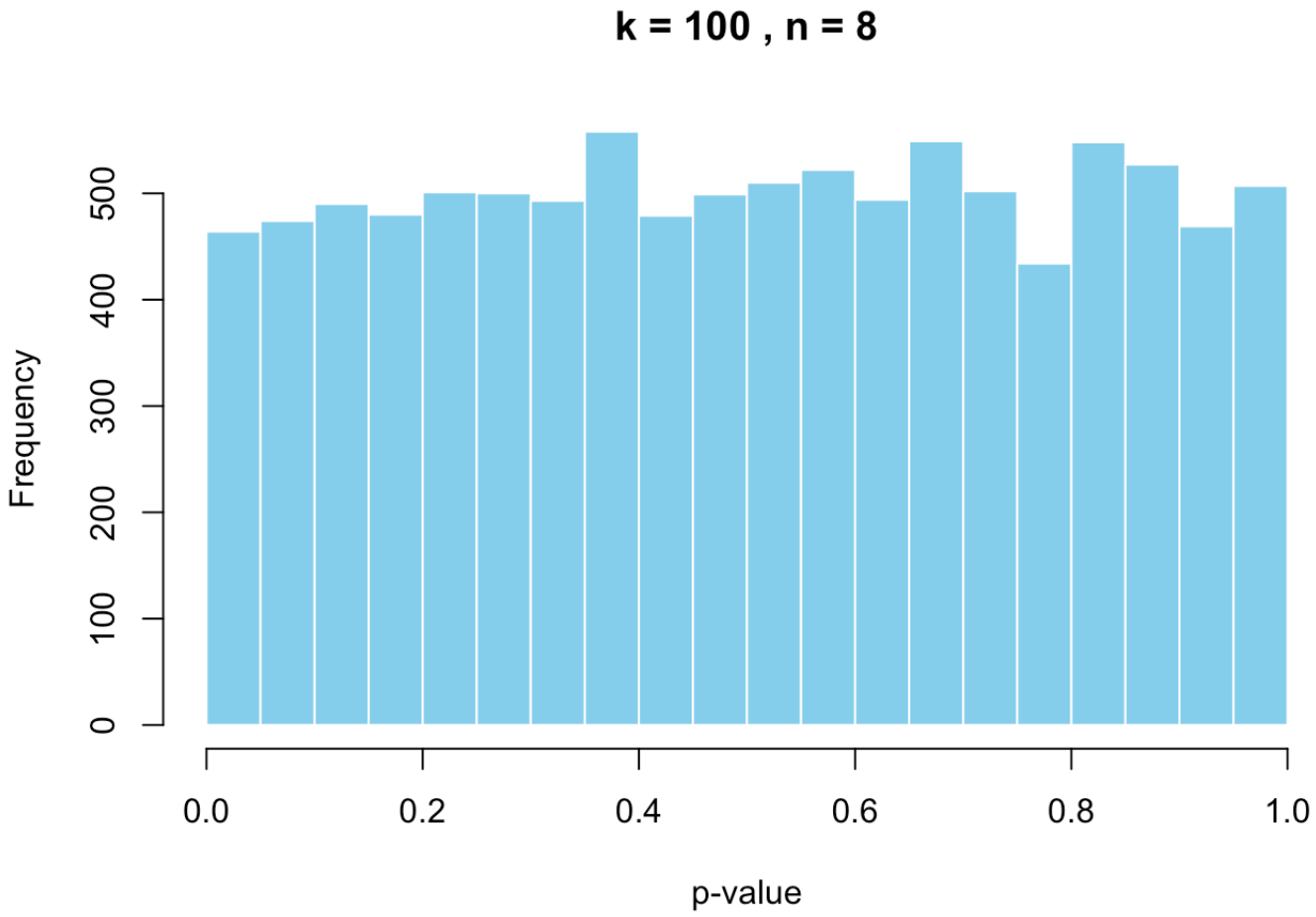

k = 50 , n = 10
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
p-value

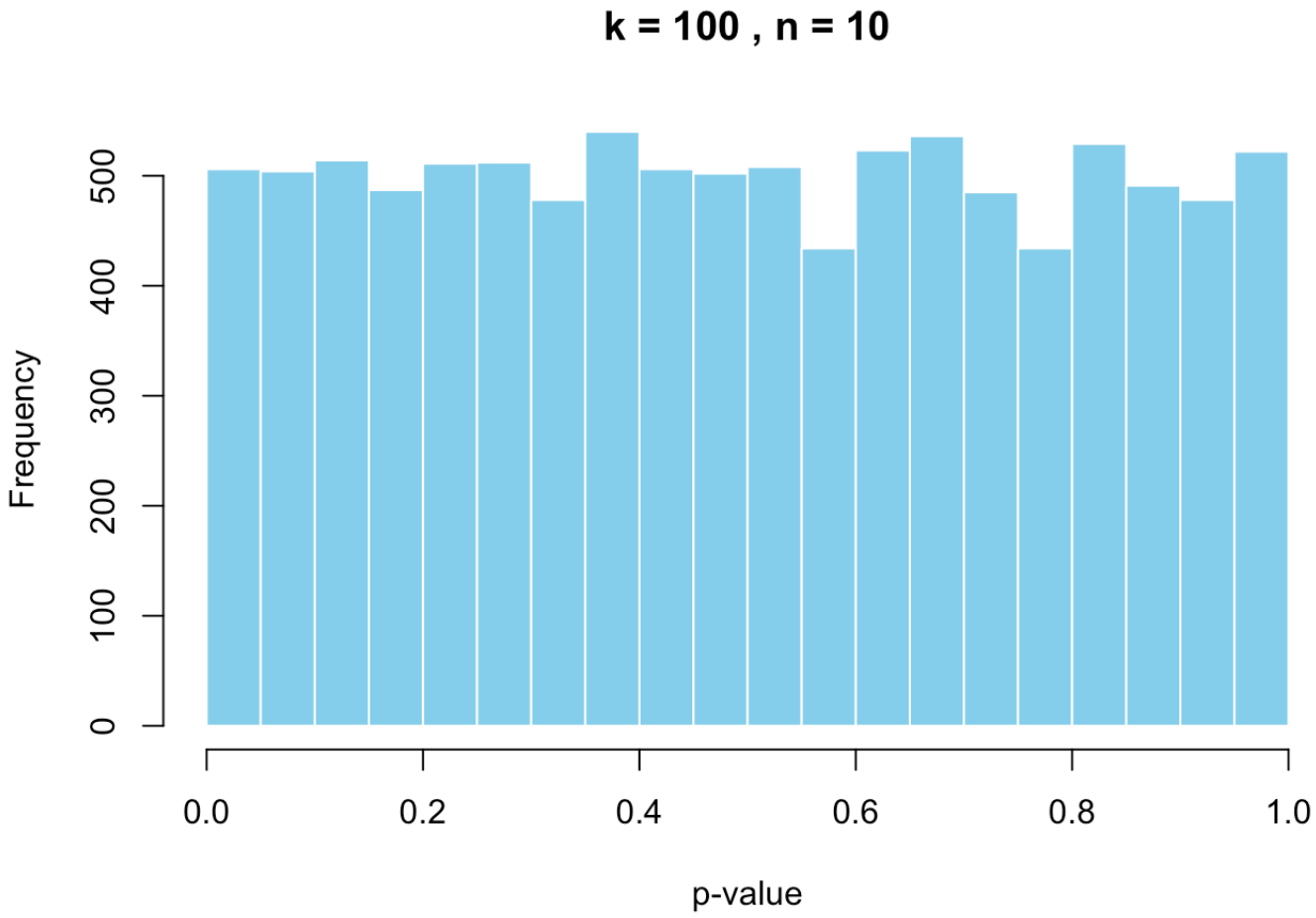

k = 100 , n = 2
Frequency
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0
p-value


k = 100 , n = 4
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value


k = 100 , n = 5
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value


k = 100 , n = 6
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
p-value


k = 100 , n = 8
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
p-value



In

my code, I first follow the instruction and find all the p-value. By plotting the p-value for each combination of n and k, we can check if the p-value is a valid one by observing the pattern of the graph.

Thus, based on the 30 histogram, we can see that that as the sample size increases, the distribution of p-values tends to become more uniform.

When the sample size is small (i.e: k = 5, n = 2k), the histograms is skew, some p value has much higher frequency compare to other. This result indicate that the distribution of p-values may not be uniform. However, as the sample size grows larger, the histograms become more uniform, with each bar become approximately at the same height. It is very clear that when the sample size become bigger and bigger, the graph is more uniform.

In summary, the trend observed in the histograms suggests that as sample size increases, the distribution of p-values tends to become more uniform, indicating the p-value is more valid when the sample size is bigger.

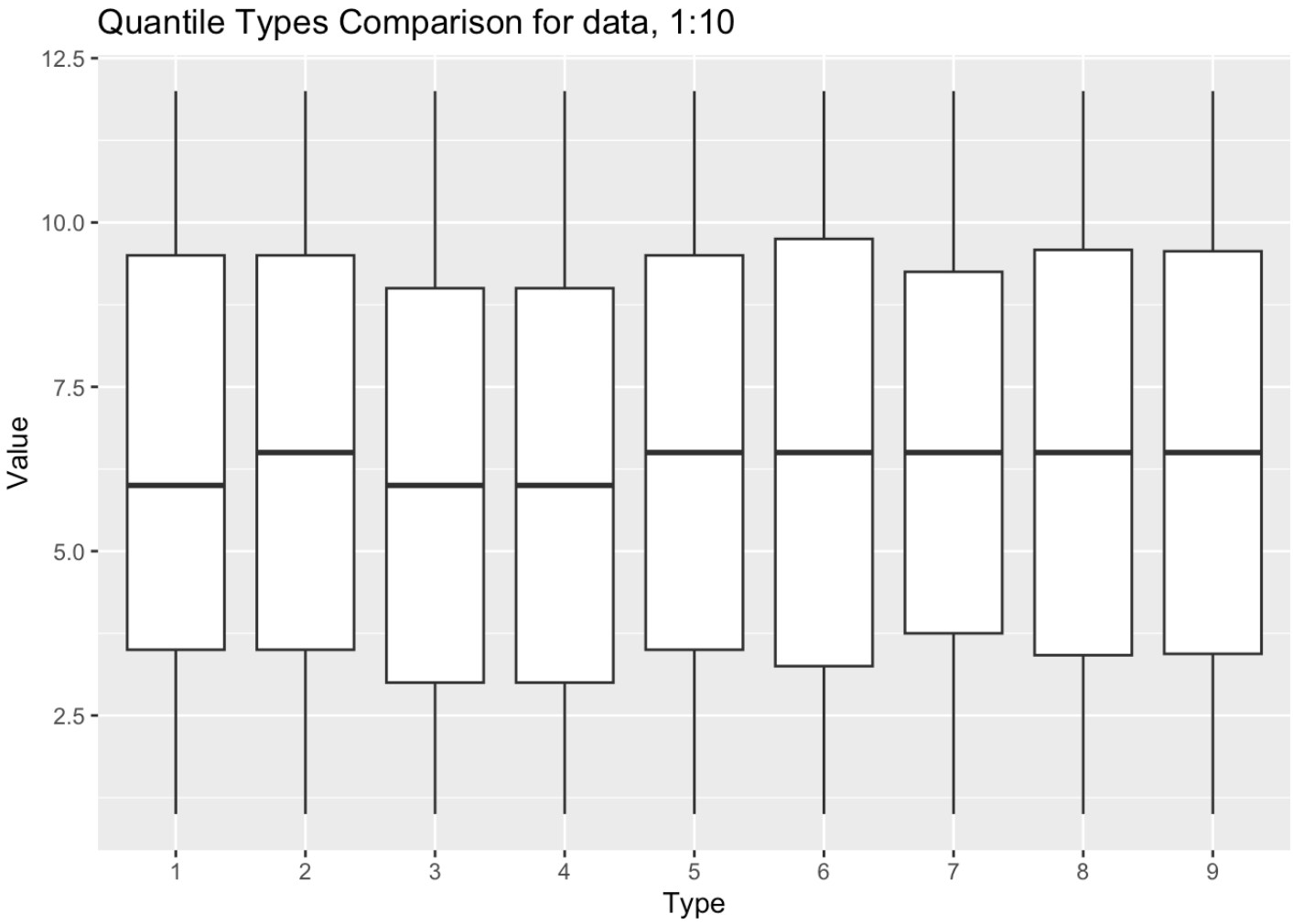
Q2:

Explain the difference:

```r
library(ggplot2)
```

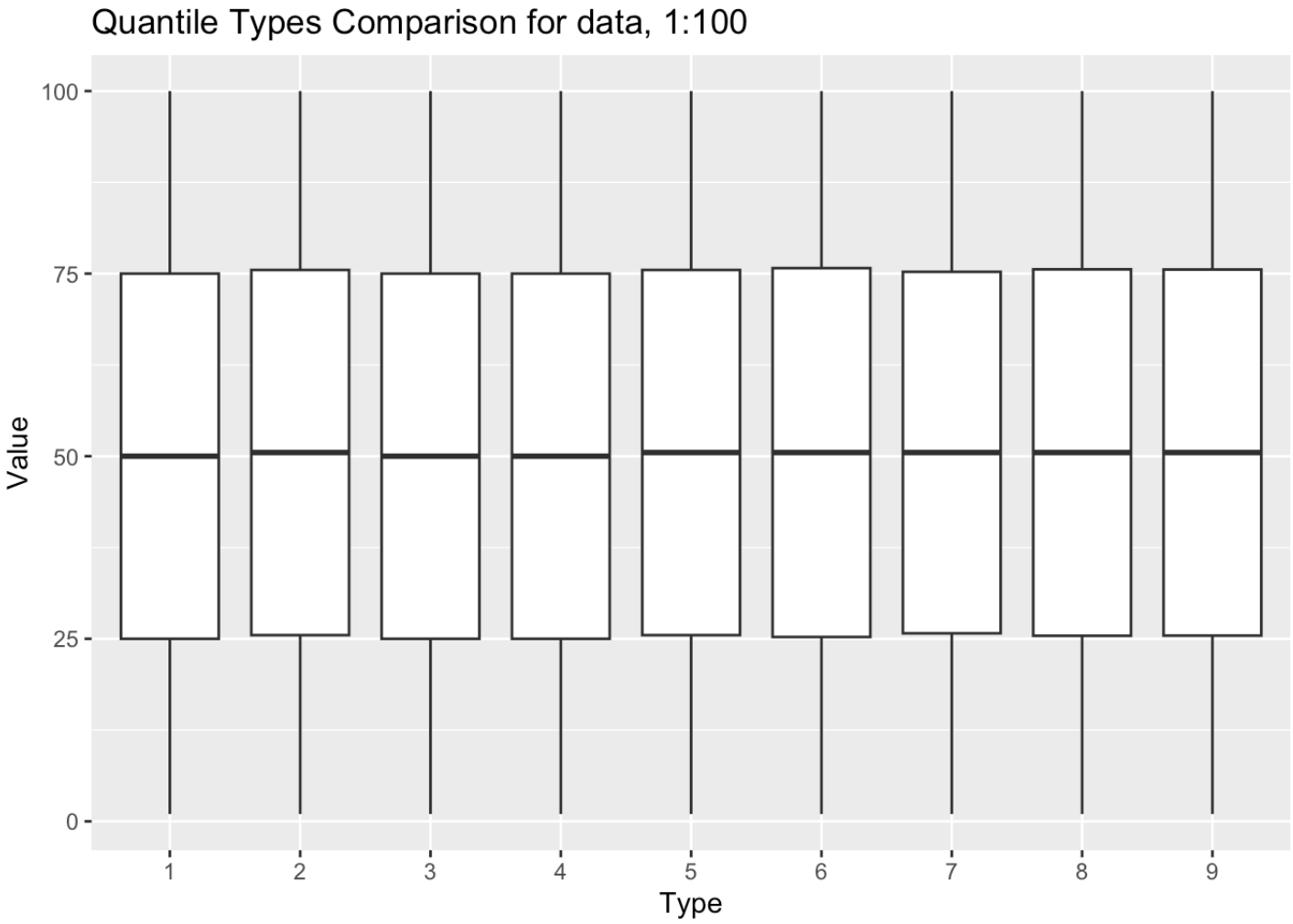
```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```r
# Define data and probabilities
data <- 1:12
probs <- seq(0, 1, 0.02)

df <- data.frame()

for (p in probs) {
  for (t in 1:9) {
    quantile_value <- quantile(data, probs = p, type = t)
    df <- rbind(df, data.frame(Probability = p, Type = t, Quantile = quantile_value))
  }
}

# Plot the box plot
ggplot(df, aes(x = as.factor(Type), y = Quantile)) +
  geom_boxplot() +
  labs(
    title = "Quantile Types Comparison for data, 1:10",
    x = "Type",
    y = "Value"
  )
```



From the boxplot, we can see that type 3 and type have has a similar result and type 8 and type 9 has a similar result. Type 6 has more spread in its result as it looks more spread out compare to other boxes.

```
# Define data and probabilities
data <- 1:100
probs <- seq(0, 1, 0.02)

df <- data.frame()

for (p in probs) {
  for (t in 1:9) {
    quantile_value <- quantile(data, probs = p, type = t)
    df <- rbind(df, data.frame(Probability = p, Type = t, Quantile = quantile_value))
  }
}

# Plot the box plot
ggplot(df, aes(x = as.factor(Type), y = Quantile)) +
  geom_boxplot() +
  labs(
    title = "Quantile Types Comparison for data, 1:100",
    x = "Type",
    y = "Value"
  )
```



When we generated more data, the difference between different type of quantile is very small. Almost every type of quantile generated the same result.

Q3:

```
uniform.test <- function(x, B=10000) {
  # Value of a,b under null hypothesis
  a <- min(x)
  b <- max(x)
  size <- length(x)

  # ks test
  ks1 <- ks.test(x, "punif", min = a, max = b)
# print(ks1)
  # init p value
  p <- c()

  for (i in 1:B) {
    # uniform distribution sample
    data <- runif(size, min = a, max = b)

    # ks test for data
    ks <- ks.test(data, "punif", min = a, max = b)

    # store p value
    p <- c(p, ks$statistic)

  }

  # Calculate
  # p_value <- mean(p <= ks1$p.value)
  p_value <- ((sum(p >= ks1$statistic) + 1) / (B+1))
  # Return the p-value
  return(p_value)
}
# test the design function
x <- rnorm(n=100)
print(uniform.test(x))
```

```
## [1] 9.999e-05
```

```
y <- runif(100, 1, 10)
print(uniform.test(y))
```

```
## [1] 0.3891611
```

Since the first value is less than 0.05, which indicated it is uniform distribution as expected as we reject the null hypothesis.

Since the second one is the random generated value, we expected to fail to reject null as it is not uniform distribution. As the value is > 0.05, it works as expected.