ST551: HOMEWORK 4

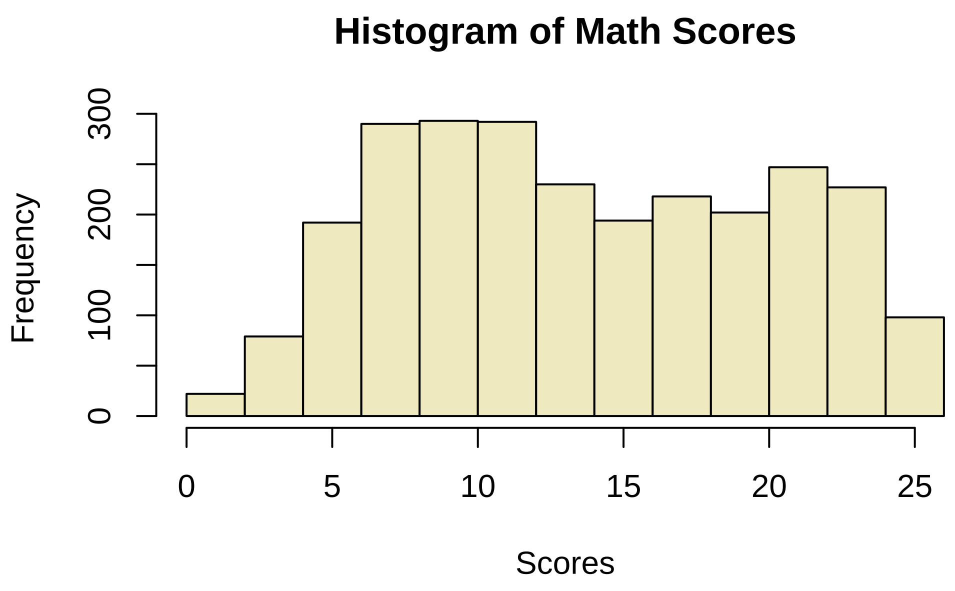
Brian Cervantes Alvarez
November 15, 2023

Question 1

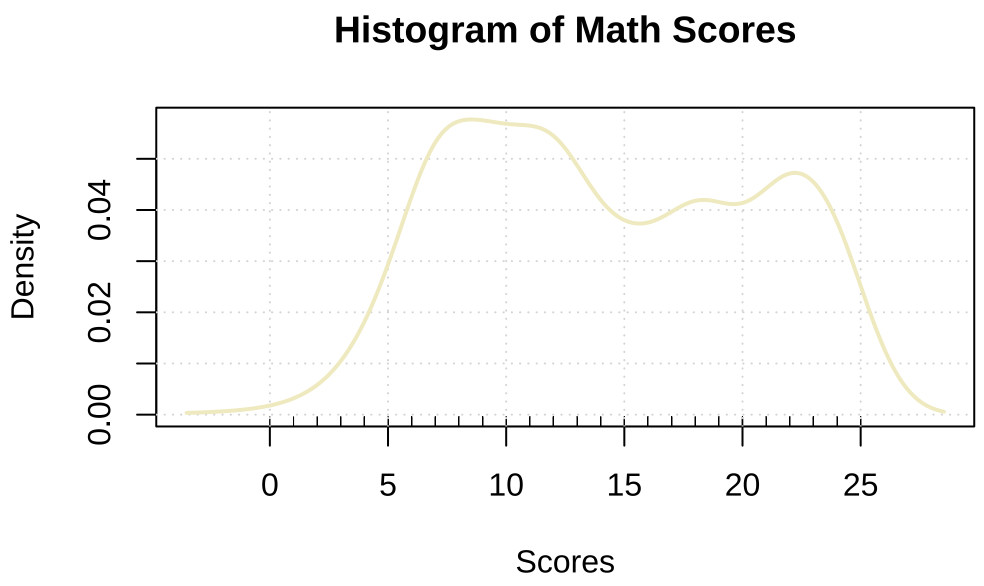
Part A

Assuming the population distribution is symmetric in its current form would be a mistake; it is bi-modal, indicating the presence of a confounding variable in this data.

```r
library(Sleuth3)
library(MASS)
library(car)

data("ex0222")
hist(ex0222$Math,
     main = "Histogram of Math Scores",
     xlab = "Scores",
     col = "lemonchiffon2")
```



```r
densityPlot(ex0222$Math,
     main = "Histogram of Math Scores",
     xlab = "Scores",
     col = "lemonchiffon2")
```


Histogram of Math Scores
Density
0.00
0.02
0.04
0
5
10
15
20
25
Scores

Part B

```r
mu <- mean(ex0222$Math)
sd <- sd(ex0222$Math)
n <- length(ex0222$Math)
conf <- 0.95

ME <- qt((1 + conf) / 2, df = n - 1) * (sd / sqrt(n))
confInt <- c(mu - ME, mu + ME)
```

Part C

With 95% confidence, the median student math score is estimated to be between 14.00000 and 14.49996, with a range of approximately 0.5.

```
wilcoxTest <- wilcox.test(ex0222$Math, conf.int = TRUE, conf.level = conf)
wilcoxConfInt <- wilcoxTest$conf.int
wilcoxConfInt
```

```
[1] 14.00000 14.49996
attr(,"conf.level")
[1] 0.95
```

The T-test confidence interval suggests the mean student math score is estimated to be between 13.95303 and 14.43784 with 95% confidence. Meanwhile, the Wilcoxon signed-rank test indicates the median score falls within the interval of 14.00000 to 14.49996 with the same confidence level.. In this case, I would choose the Wilcoxon signed-rank test given that it is deemed more appropriate due to the math score distribution being bimodal.

Part D

```r
print(paste0("T-Test Confidence Interval"))
```

```
[1] "T-Test Confidence Interval"
```

```r
confInt
```

```
[1] 13.95303 14.43784
```

```r
print(paste0("Wilcoxon Signed-Rank Test Confidence Interval:"))
```

```
[1] "Wilcoxon Signed-Rank Test Confidence Interval:"
```

```r
wilcoxConfInt
```

```
[1] 14.00000 14.49996
attr(,"conf.level")
[1] 0.95
```

Question 2

My simulation uses a sample size of 30 with factors set at 0.5, 1, and 2, each iteration being simulated 1000 times. The outcomes, including factor values, average precision, and average power, are systematically recorded in a dataframe. Making the function was quite fun since I desired a method to simplify my simulations.

```
library(dplyr)

# Set parameters
n <- 30
factors <- c(0.5, 1, 2)
nSim <- 1000

results <- data.frame(Factor = numeric(),
                      Average_Exactness = numeric(),
                      Average_Power = numeric())

# Function to perform the simulation and return results
createSim <- function(distributionName) {
  cat(paste("\n", distributionName, "Distribution \n"))
  cat("Factor\t   Average_Exactness\tAverage_Power\n")

  # Initialize variables to accumulate results
  totalExactness <- 0
  totalPower <- 0

  for (i in factors) {
    for (j in 1:nSim) {
      # Generate data for two groups
      grp1 <- switch(distributionName,
                     "Gamma" = rgamma(n, shape = 1),
                     "Exponential" = rexp(n, rate = 1))
      data1 <- rnorm(n, mean = 0, sd = i)
      grp1 <- grp1 + data1

      grp2 <- switch(distributionName,
                     "Gamma" = rgamma(n, shape = 1),
                     "Exponential" = rexp(n, rate = 1))
      data2 <- rnorm(n, mean = 0, sd = i)
      grp2 <- grp2 + data2

      # Perform Wilcoxon signed-rank test
      wilcoxResult <- wilcox.test(grp1, grp2, paired = TRUE)

      # Accumulate results
      totalExactness <- totalExactness + (wilcoxResult$p.value > 0.05)
      totalPower <- totalPower + (1 - wilcoxResult$p.value)
    }

    # Calculate averages for the current factor
    avgExactness <- totalExactness / nSim
    avgPower <- totalPower / nSim
    # Add results to the df
    results <- rbind(results, c(i, avgExactness, avgPower))

    # Display results for the current factor
    cat(paste(i, "\t  ", avgExactness, "\t       ", avgPower, "\n"))
  }

  # Assign column names
  colnames(results) <- c("Factor", "Average_Exactness", "Average_Power")

  # Print results
  print(results)
}

# Perform simulations for Gamma Distribution
createSim("Gamma")
```

```
 Gamma Distribution
Factor     Average_Exactness    Average_Power
0.5        0.952                0.501616657188162
1      1.896                0.994773385245353
2      2.845                1.50310466326959
  Factor Average_Exactness Average_Power
1    0.5             0.952     0.5016167
2    1.0             1.896     0.9947734
3    2.0             2.845     1.5031047
```

```
# Perform simulations for Exponential Distribution
createSim("Exponential")
```

```
 Exponential Distribution
Factor     Average_Exactness    Average_Power
0.5        0.935                0.50380264909938
1      1.878                1.01934559555538
2      2.834                1.50225697369315
  Factor Average_Exactness Average_Power
1    0.5             0.935     0.5038026
2    1.0             1.878     1.0193456
3    2.0             2.834     1.5022570
```

Question 3

The statistical test's consistency implies that as $n \to \infty$, it correctly rejects a false null hypothesis with a probability approaching 1, increasing test power and reducing the Type II error probability to zero.

The chi-squared test for population variance uses the statistic $\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$, where n is the sample size, s^2 is the sample variance, and σ_0^2 is the hypothesized population variance (H_0).

The sample variance s^2 consistently estimates population variance σ^2 as $n \to \infty$. As n increases, the chi-squared statistic approaches $\lim_{n\to\infty} \frac{n\times\sigma^2}{\sigma_0^2}$, converging to n under H_0 and diverging if $\sigma^2 \neq \sigma_0^2$. In both cases, as $n \to \infty$, the chi-squared statistic provides evidence against H_0, showcasing the test's consistency irrespective of data distribution.

Question 4

Part A

A rejection frequency of 0.049 means that the null hypothesis was rejected in approximately 4.9% of the tests. Given that the significance level is 0.05, a rejection frequency of 0.049 suggests that the test is performing reasonably close to the chosen significance level.

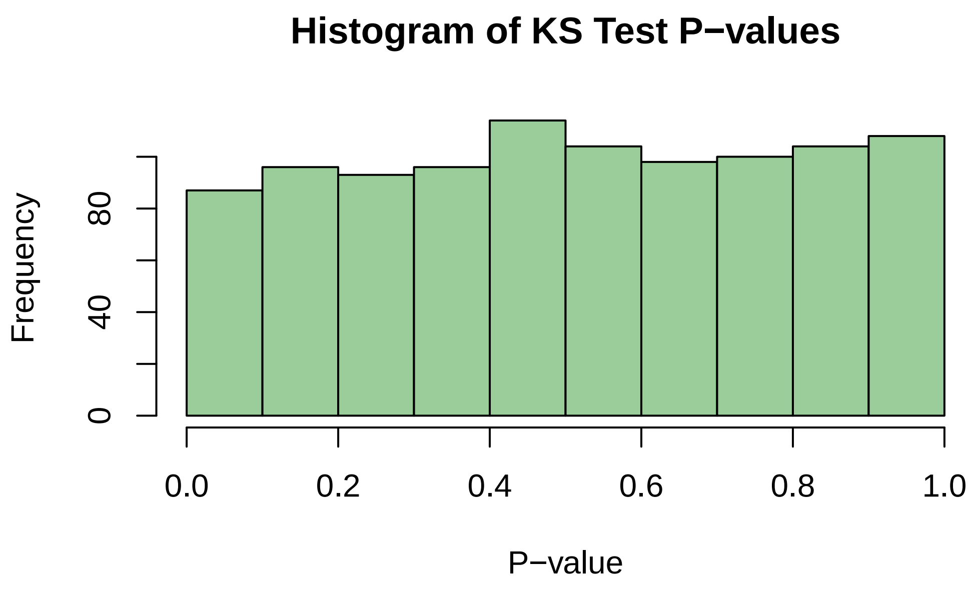
```r
set.seed(579)
# Set parameters
nSim <- 1000
nObservations <- 50
alpha <- 0.05
rejectionCount <- 0


# This store p-values for historgram in part b
pvals <- numeric(nSim)

# Perform the test for each dataset
for (i in 1:nSim) {
  # Generate data
  data <- rnorm(nObservations)

  # Perform KS test
  ksResult <- ks.test(data, "pnorm", mean = 0, sd = 1)

  # Store the p-value
  pvals[i] <- ksResult$p.value

  # Check if null hypothesis is rejected
  if (ksResult$p.value < alpha) {
    rejectionCount <- rejectionCount + 1
  }
}

# Calculate rejection frequency
rejectionFrequency <- rejectionCount / nSim

# Print results
print(paste("Rejection frequency:", rejectionFrequency))
```

```
[1] "Rejection frequency: 0.049"
```

Part B

The p-values appear to follow a uniform distribution, which is a good sign. This is because under the null hypothesis, the p-values should be uniformly distributed between 0 and 1. If it was calibrated near-perfectly, it should be a near perfect uniform distribution $U(0,1)$

```
# Create a histogram of p-values
hist(pvals, main = "Histogram of KS Test P-values",
     xlab = "P-value",
     ylab = "Frequency", col = "darkseagreen3")
```



Part C

This test rejects little to none of the null hypothesis tests. Even at a signficiance level of 0.05, it still have extremely low rejection (R can't even process the lower rejection because of memory).

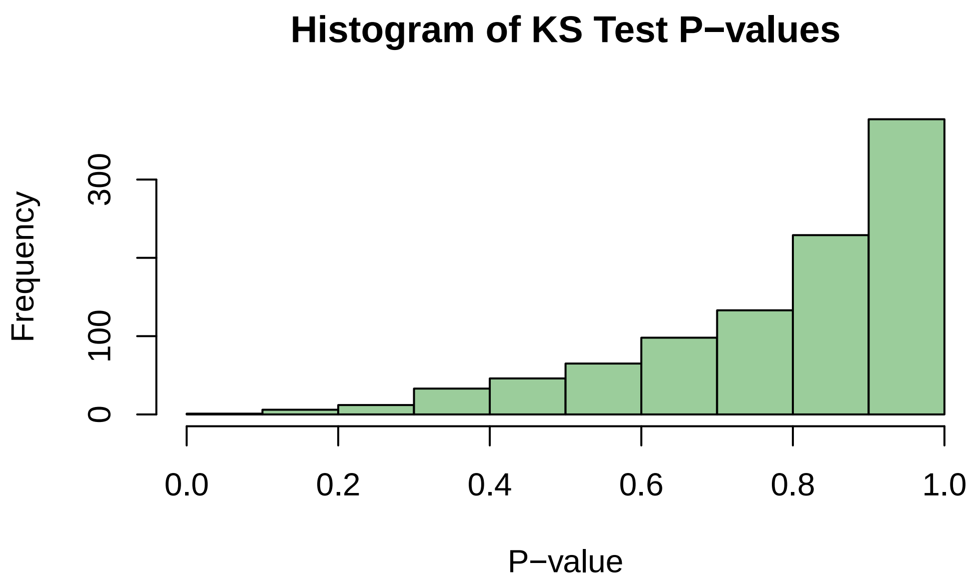
```r
set.seed(579)
# Set parameters
nSim <- 1000
nObservations <- 500
alpha <- 0.05
rejectionCount <- 0

# This stores p-values for the histogram in part B
pvals <- numeric(nSim)

# Perform the test for each dataset
for (i in 1:nSim) {
  # Generate data
  data <- rnorm(nObservations)

  # Calculate sample mean and sample standard deviation
  sampMean <- mean(data)
  sampSd <- sd(data)

  # Perform KS test with sample mean and sample standard deviation
  ksResult <- ks.test(data, "pnorm", mean = sampMean, sd = sampSd)

  # Store the p-value
  pvals[i] <- ksResult$p.value

  # Check if null hypothesis is rejected
  if (ksResult$p.value < alpha) {
    rejectionCount <- rejectionCount + 1
  }
}

# Calculate rejection frequency
rejectionFrequency <- rejectionCount / nSim

# Print results
print(paste("Rejection frequency:", rejectionFrequency))
```

```
[1] "Rejection frequency: 0.001"
```

Part D

The data is not following a normal distribution. Again, it the p-values should follow a uniform distribution of $U(0, 1)$

```
# Create a histogram of p-values
hist(pvals, main = "Histogram of KS Test P-values",
     xlab = "P-value",
     ylab = "Frequency",
     col = "darkseagreen3")
```



Question 5

Part A

The two-sample t-test statistics for equal variance (s_p^2) and unequal variance (s_1^2, s_2^2) are given by:

Equal-variance t-statistic:

$$t_{\text{equal}} = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Unequal-variance t-statistic:

$$t_{\text{unequal}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Given that the sample size is the same for both samples n, we can prove that the equal and unequal variance test statistics are equal.

When $n_1 = n_2 = n$, we can simplify the equal-variance t-statistic:

$$t_{\text{equal}} = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{n} + \frac{1}{n}}} = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{2}{n}}}$$

Now, let's look at the unequal-variance t-statistic with equal sample sizes:

$$t_{\text{unequal}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{n}}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2+s_2^2}{n}}}$$

In our case, $s_p^2 = \frac{s_1^2+s_2^2}{2}$, so we can replace the denominator in the unequal-variance t-statistic:

$$t_{\text{unequal}} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{2s_p^2}{n}}} = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{2}{n}}}$$

Therefore,

$$t_{\text{equal}} = t_{\text{unequal}}$$

Part B

Intriguing, the choice between equal and unequal variance t-tests depends on the specific characteristics of the data. Equal variance assumption may not be suitable in all cases, and the performance varies across different scenarios as shown in the simulations.

```r
set.seed(529)

# Function to perform t-tests and return rejection results
performTTests <- function(X, Y) {
  equalVarResult <- t.test(X, Y, var.equal = TRUE)$p.value < 0.05
  unequalVarResult <- t.test(X, Y, var.equal = FALSE)$p.value < 0.05
```

```r
    return(data.frame(
      EqualVariance = as.logical(equalVarResult),
      UnequalVariance = as.logical(unequalVarResult)
    ))
  }

  # Create table (dataframe)
  results <- data.frame(
    Scenario = character(),
    EqualVariance = numeric(),
    UnequalVariance = numeric(),
    stringsAsFactors = FALSE
  )

  # Simulate 10,000 times
  nSim <- 10000

  for (i in 1:nSim) {
    # Scenario 1: Normal(0, 1) vs Normal(0, 4), m=10, n=30
    X <- rnorm(10, mean = 0, sd = 1)
    Y <- rnorm(30, mean = 0, sd = 4)
    rejectionResults <- performTTests(X, Y)
    results <- bind_rows(results, tibble(Scenario = "Normal(0,1) vs Normal(0,4), m=10, n=30",

    # Scenario 2: Exponential(1) vs Normal(1, 4), m=30, n=10
    X <- rexp(30, rate = 1)
    Y <- rnorm(10, mean = 1, sd = 4)
    rejectionResults <- performTTests(X, Y)
    results <- bind_rows(results, tibble(Scenario = "Exponential(1) vs Normal(1,4), m=30, n=1

    # Scenario 3: Chi-squared(5) vs Normal(5, 1), m=20, n=50
    X <- rchisq(20, df = 5)
    Y <- rnorm(50, mean = 5, sd = 1)
    rejectionResults <- performTTests(X, Y)
    results <- bind_rows(results, tibble(Scenario = "Chi-squared(5) vs Normal(5,1), m=20, n=5

    # Scenario 4: t(5) vs t(3), m=30, n=10
    X <- rt(30, df = 5)
    Y <- rt(10, df = 3)
    rejectionResults <- performTTests(X, Y)
    results <- bind_rows(results, tibble(Scenario = "t(5) vs t(3), m=30, n=10", rejectionResul
  }

  # Calculate the average results
  averageResults <- results %>%
    group_by(Scenario) %>%
    summarise(across(c(EqualVariance, UnequalVariance), mean))

  # Print the average results
  print(averageResults)
```

```
# A tibble: 4 x 3
  Scenario                                  EqualVariance UnequalVariance
  <chr>                                             <dbl>          <dbl>
1 Chi-squared(5) vs Normal(5,1), m=20, n=50       0.189           0.0657
2 Exponential(1) vs Normal(1,4), m=30, n=10       0.239           0.0461
3 Normal(0,1) vs Normal(0,4), m=10, n=30          0.00260         0.0492
4 t(5) vs t(3), m=30, n=10                        0.0710          0.0421
```

Question 6

Part A

A negative population covariance between X and Y indicates opposite directional movement. The paired t-test formula,

$$t = \frac{\bar{d}}{\sqrt{\frac{s_d^2}{n}}},$$

where $\bar{d}$ is the mean of differences $d_i = x_i - y_i$, s_d^2 is the sample variance of differences, and n is the number of pairs, is affected. Negative covariance suggests a likelihood of large $\bar{d}$ when X is small and Y is large, and vice versa, resulting in a relatively large s_d^2. This inflates the t-value denominator, increasing the t-value and the likelihood of rejecting the null hypothesis. Negative covariance thus elevates the Type I error rate in the paired t-test.

Part B

The simulation results indicate that, on average, neither the paired t-test nor the unequal variance t-test yielded statistically significant differences (p-value < 0.05) across the specified scenarios, suggesting that observed mean differences are likely due to random chance.

```r
# Set seed
set.seed(529)

# Function to perform paired and unequal t-tests and return rejection results
performTTests <- function(X, Y) {
  paired_p_value <- t.test(X, Y, paired = TRUE)$p.value
  unequal_p_value <- t.test(X, Y, var.equal = FALSE)$p.value

  return(data.frame(
    PairedTTest = as.logical(paired_p_value < 0.05),
    UnequalTTest = as.logical(unequal_p_value < 0.05)
  ))
}

# Create table (dataframe)
results <- data.frame(
  Scenario = character(),
  PairedTTest = numeric(),
  UnequalTTest = numeric(),
  stringsAsFactors = FALSE
)

# Simulate 10,000 times
nSim <- 10000

# Scenario 1: Normal(0, 1) vs Normal(0, 4), m=10, n=10
for (i in 1:nSim) {
  X <- rnorm(10, mean = 0, sd = 1)
  Y <- rnorm(10, mean = 0, sd = 4)
  rejectionResults <- performTTests(X, Y)  # Ensure X and Y have the same length
  results <- bind_rows(results, tibble(Scenario = "Normal(0,1) vs Normal(0,4), m=10, n=10",
}

# Scenario 2: Chi-squared(5) vs Normal(5, 4), m=30, n=30
for (i in 1:nSim) {
  X <- rchisq(30, df = 5)
  Y <- rnorm(30, mean = 5, sd = 4)
  rejectionResults <- performTTests(X, Y)  # Ensure X and Y have the same length
  results <- bind_rows(results, tibble(Scenario = "Chi-squared(5) vs Normal(5,4), m=30, n=3
}

# Scenario 3: Normal(0, 1) vs Normal(2, 4), m=10, n=10
for (i in 1:nSim) {
  X <- rnorm(10, mean = 0, sd = 1)
  Y <- rnorm(10, mean = 2, sd = 4)
  rejectionResults <- performTTests(X, Y)  # Ensure X and Y have the same length
  results <- bind_rows(results, tibble(Scenario = "Normal(0,1) vs Normal(2,4), m=10, n=10",
}

# Scenario 4: Chi-squared(5) vs Normal(3, 4), m=30, n=30
for (i in 1:nSim) {
  X <- rchisq(30, df = 5)
  Y <- rnorm(30, mean = 3, sd = 4)
  rejectionResults <- performTTests(X, Y)  # Ensure X and Y have the same length
  results <- bind_rows(results, tibble(Scenario = "Chi-squared(5) vs Normal(3,4), m=30, n=3
}

# Calculate the average results
averageResults <- results %>%
  group_by(Scenario) %>%
  summarise(across(c(PairedTTest, UnequalTTest), mean))

# Print the average results
print(averageResults)
```

```
# A tibble: 4 x 3
  Scenario                                  PairedTTest UnequalTTest
  <chr>                                           <dbl>        <dbl>
1 Chi-squared(5) vs Normal(3,4), m=30, n=30      0.545        0.560
2 Chi-squared(5) vs Normal(5,4), m=30, n=30      0.0490       0.0517
3 Normal(0,1) vs Normal(0,4), m=10, n=10         0.0512       0.0491
4 Normal(0,1) vs Normal(2,4), m=10, n=10         0.277        0.284
```