Programming in statistics: Assignment 6

Abdullah Ismayilzada

27/10/2023

Instructions

This assignment should be orally presented. In addition to the report, you should prepare a short presentation (6-10 min) with a few **slides**, focusing on the simulation study's design, results and conclusions (avoid showing R code). You can upload your presentation or just bring it with you to the seminar.

A **report** with your solutions should be uploaded. In addition, upload your script (.R or .Rmd) file containing the code needed to (re-)produce your solutions. The report should be a pdf-document. It should be a self-explanatory document, i.e., understandable without having to read the assignment specification. The report should contain the name of the assignment, your name and date, the problems that you are trying to solve as well as the results (including explanatory text, tables/figures etc). **Include all R code in the report in an appendix** and let the main report focus on describing the design, results and conclusions. Remember to comment your code.

The aim of this course is for you to enhance your `R` programming skills, and learning by doing is really the way to get there. That said, you are encouraged to discuss course material, including assignments, with each other. However, **the solutions and the report you hand in should be your own work and you must have written your own code**.

One of the Expected Learning Outcomes of this course is to carry out simulation studies and analyze and evaluate the resulting performance. The purpose of this assignment is to perform a simulation study, and present and evaluate the results from that simulation. Read the slides and other documents on Canvas (under "Simulation") to get more familiar with simulation studies and reporting simulation study results.

Simulation study

Your goal is to perform a simulation study to evaluate the performance of four different tests for evaluating equal means from two populations, i.e., tests that evaluate the hypotheses

$$H_0 : \mu_1 = \mu_2 \quad \text{vs.} \quad H_1 : \mu_1 \neq \mu_2$$

where μ_1 and μ_2 is the expected value in population 1 and 2, respectively. You want to investigate and compare the empirical *size* and *power* of the following tests. The tests you will compare are

1) the Z-test where the variances in the two population are assumed to be known. (see function `z.test` in the `BSDA` package),
2) the two (independent) sample t-test assuming equal variances ($\sigma_1^2 = \sigma_2^2$),
3) the Welch test i.e. the two (independent) sample t-test not assuming equal variances ($\sigma_1^2 \neq \sigma_2^2$), see `t.test` in `R`, and
4) Wilcoxon Rank-Sum (or Mann-Whitney U) nonparametric test (see function `wilcox.test`).

Note that the latter test has slightly different hypotheses,

$$H_0 : f_1 = f_2$$

where f_i is the density function in population i. This implies $\mu_1 = \mu_2$.

In addition to the variance assumption a t-test (and Z-test) also assumes the following: 1) The data is continuous (not discrete). 2) The data follow the normal probability distribution. 3) The two samples are independent. There is no relationship between the units in one sample as compared to the other. 4) Both samples are simple random samples from their respective populations. Each unit in the population has an equal probability of being selected in the sample.

Keep the test's assumptions in mind when creating your data generating process.

Starting by writing a function that takes as formal arguments the number of data sets to be generated, S, as well as the sample sizes for the two samples, n, and possibly necessary distribution parameters. This function should generate data from an appropriate distribution with the given parameter values, perform the tests, and return the an appropriate output (e.g. p-values) for evaluation.

G level:

- Perform a simulation study with at least $S = 100\ 000$ to compare the *size* of the four tests described above. Generate data so that the variances are equal ($\sigma_1^2 = \sigma_2^2$).
  Choose a significance level α (nominal size) and four different sample sizes, e.g., $n_i = (3, 5, 20, 200)$, so that you can show small and larger sample properties. You can choose $n_1 = n_2$ or $n_1 \neq n_2$ (but choose in a systematic way so that you can interpret you results).

```
# Load the necessary libraries
library(BSDA)
```

```
## Warning: package 'BSDA' was built under R version 4.1.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'BSDA'
```

```
## The following object is masked from 'package:datasets':
##
##     Orange
```

```
# Set the seed for reproducibility
set.seed(366)

# Set the parameters
S <- 100000  # Number of simulations
alpha <- 0.05  # Significance level

# Define sample sizes
n_values <- c(3, 5, 20, 200)

# Create a data frame to store the results
results <- data.frame(
  SampleSize = numeric(S),
```

```
  Z_test_p_value = numeric(S),
  Equal_Var_t_test_p_value = numeric(S),
  Welch_t_test_p_value = numeric(S),
  Wilcoxon_p_value = numeric(S)
)

# Perform the simulations
for (n in n_values) {
  for (i in 1:S) {
    # Generate data with equal variances (e.g., normal distribution)
    mu1 <- 0
    mu2 <- 0
    sigma <- 1  # Set a common standard deviation
    sample1 <- rnorm(n, mean = mu1, sd = sigma)
    sample2 <- rnorm(n, mean = mu2, sd = sigma)

# Perform the tests
    z_test_result <- z.test(sample1, sample2, sigma.x = sigma, sigma.y = sigma)
    equal_var_t_test_result <- t.test(sample1, sample2, var.equal = TRUE)
    welch_t_test_result <- t.test(sample1, sample2, var.equal = FALSE)
    wilcoxon_result <- wilcox.test(sample1, sample2)

# Store the sample size and p-values in the results data frame
    results$SampleSize[i] <- n
    results$Z_test_p_value[i] <- z_test_result$p.value
    results$Equal_Var_t_test_p_value[i] <- equal_var_t_test_result$p.value
    results$Welch_t_test_p_value[i] <- welch_t_test_result$p.value
    results$Wilcoxon_p_value[i] <- wilcoxon_result$p.value
  }
}


# Calculate and compare the actual size (Type I error rate)
size_results <- data.frame(
  Test = c("Z-test", "Equal Var t-test", "Welch t-test", "Wilcoxon"),
  Actual_Size = sapply(
    list(
      Z_test = results$Z_test_p_value,
      Equal_Var_t_test = results$Equal_Var_t_test_p_value,
      Welch_t_test = results$Welch_t_test_p_value,
      Wilcoxon = results$Wilcoxon_p_value
    ),
    function(p_values) {
      mean(p_values < alpha)
    }
  )
)

# Print the results
print(size_results)
```

```
##                             Test Actual_Size
## Z_test                   Z-test     0.05091
```

```
## Equal_Var_t_test Equal Var t-test     0.05135
## Welch_t_test         Welch t-test     0.05135
## Wilcoxon                 Wilcoxon     0.05134
```

- Perform a simulation study with at least $S = 100\,000$ to compare the *size* of four tests described above. Generate data so that the variances are not equal ($\sigma_1^2 \neq \sigma_2^2$). Use the same significance level α and four different sample sizes, e.g., $n_i = (5, 10, 20, 200)$.

```r
# Load the necessary libraries
library(BSDA)

# Set the seed for reproducibility
set.seed(366)

# Set the parameters
S <- 100000  # Number of simulations
alpha <- 0.05  # Significance level

# Define sample sizes for Task 2
n_values_task2 <- c(5, 10, 20, 200)

# Create a data frame to store the results for Task 2
results_task2 <- data.frame(
  SampleSize = numeric(S),
  Z_test_p_value = numeric(S),
  Equal_Var_t_test_p_value = numeric(S),
  Welch_t_test_p_value = numeric(S),
  Wilcoxon_p_value = numeric(S)
)

# Perform the simulations for Task 2
for (n in n_values_task2) {
  for (i in 1:S) {
    # Generate data with unequal variances for Task 2 (e.g., normal distribution)
    mu1 <- 0
    mu2 <- 0
    sigma1 <- 1
    sigma2 <- 2  # Set different variances for unequal variances
    sample1 <- rnorm(n, mean = mu1, sd = sigma1)
    sample2 <- rnorm(n, mean = mu2, sd = sigma2)

# Perform the tests for Task 2
    z_test_result <- z.test(sample1, sample2, sigma.x = sigma1, sigma.y = sigma2)
    equal_var_t_test_result <- t.test(sample1, sample2, var.equal = TRUE)
    welch_t_test_result <- t.test(sample1, sample2, var.equal = FALSE)
    wilcoxon_result <- wilcox.test(sample1, sample2)

# Store the sample size and p-values in the results data frame for Task 2
    results_task2$SampleSize[i] <- n
    results_task2$Z_test_p_value[i] <- z_test_result$p.value
    results_task2$Equal_Var_t_test_p_value[i] <- equal_var_t_test_result$p.value
    results_task2$Welch_t_test_p_value[i] <- welch_t_test_result$p.value
    results_task2$Wilcoxon_p_value[i] <- wilcoxon_result$p.value
  }
```

```
}

# Calculate and compare the actual size (Type I error rate) for Task 2
size_results_task2 <- data.frame(
  Test = c("Z-test", "Equal Var t-test", "Welch t-test", "Wilcoxon"),
  Actual_Size = sapply(
    list(
      Z_test = results_task2$Z_test_p_value,
      Equal_Var_t_test = results_task2$Equal_Var_t_test_p_value,
      Welch_t_test = results_task2$Welch_t_test_p_value,
      Wilcoxon = results_task2$Wilcoxon_p_value
    ),
    function(p_values) {
      mean(p_values < alpha)
    }
  )
)

# Print the results for Task 2
print(size_results_task2)
```

```
##                              Test Actual_Size
## Z_test                     Z-test     0.05124
## Equal_Var_t_test Equal Var t-test     0.05149
## Welch_t_test         Welch t-test     0.05126
## Wilcoxon                 Wilcoxon     0.05989
```

- Perform a simulation study with at least $S = 1000\,000$ to compare the *power* of the four tests described above. Generate data so that the variances are unequal ($\sigma_1^2 \neq \sigma_2^2$). Use the same significance level α and *one* appropriate sample size. Investigate at least six different scenarios where $\mu_1 \neq \mu_2$, for example $\mu_1 = \mu_2 + \delta$ where δ is a number ranging from zero to an appropriate integer. Note! you should include $\delta = 0$ as a baseline even though this technically is the empirical size/level of the t.

```
# Load the necessary libraries
library(BSDA)
library(parallel)

# Set the seed for reproducibility (use the same seed as in previous tasks)
set.seed(366)

# Set the parameters
S <- 1000000  # Number of simulations
alpha <- 0.05  # Significance level

# Define a single sample size
n <- 20  # Choose an appropriate sample size

# Define different values of delta (difference in means)
delta_values <- c(0, 1, 2, 3, 4, 5)  # Investigate six scenarios

# Create a data frame to store the results
results <- data.frame(
  Delta = numeric(S),
```

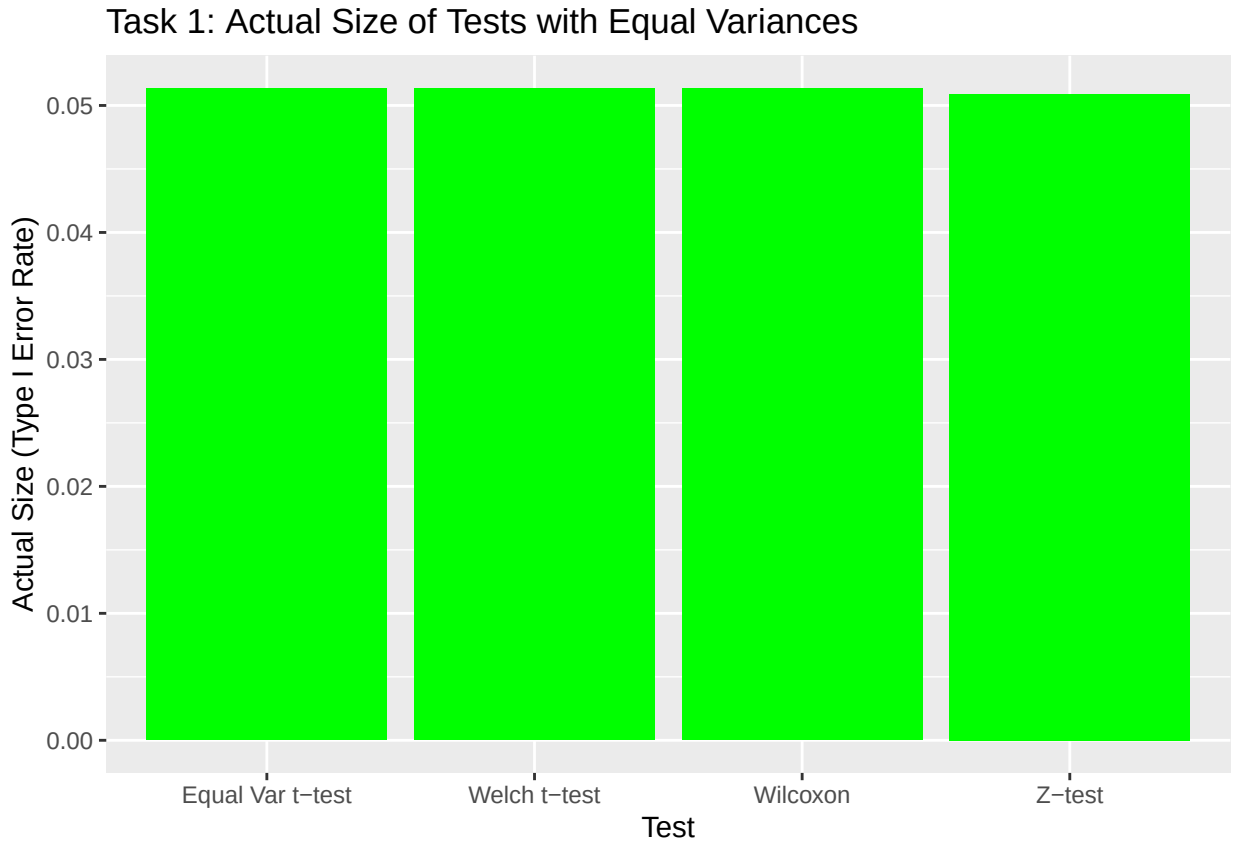
```
  Z_test_power = numeric(S),
  Equal_Var_t_test_power = numeric(S),
  Welch_t_test_power = numeric(S),
  Wilcoxon_power = numeric(S)
)

# Set the number of processor cores to use
num_cores <- detectCores()

simulate_parallel <- function(delta, S, n) {
  library(BSDA)  # Load the BSDA package in the worker nodes

  sim_results <- data.frame(
    Delta = numeric(S),
    Z_test_power = numeric(S),
    Equal_Var_t_test_power = numeric(S),
    Welch_t_test_power = numeric(S),
    Wilcoxon_power = numeric(S)
  )

  for (i in 1:S) {
    mu1 <- 0
    mu2 <- delta
    sigma1 <- 1
    sigma2 <- 2
    sample1 <- rnorm(n, mean = mu1, sd = sigma1)
    sample2 <- rnorm(n, mean = mu2, sd = sigma2)

    z_test_result <- z.test(sample1, sample2, sigma.x = sigma1, sigma.y = sigma2)
    equal_var_t_test_result <- t.test(sample1, sample2, var.equal = TRUE)
    welch_t_test_result <- t.test(sample1, sample2, var.equal = FALSE)
    wilcoxon_result <- wilcox.test(sample1, sample2)

    sim_results$Delta[i] <- delta
    sim_results$Z_test_power[i] <- 1 - z_test_result$p.value
    sim_results$Equal_Var_t_test_power[i] <- 1 - equal_var_t_test_result$p.value
    sim_results$Welch_t_test_power[i] <- 1 - welch_t_test_result$p.value
    sim_results$Wilcoxon_power[i] <- 1 - wilcoxon_result$p.value
  }

  return(sim_results)
}


# Perform simulations in parallel
cl <- makeCluster(num_cores)
clusterExport(cl, c("delta_values", "S", "n"))
simulated_data <- parLapply(cl, delta_values, simulate_parallel, S = S, n = n)
stopCluster(cl)

# Combine results from parallel simulations
results <- do.call(rbind, simulated_data)
```

```r
# Calculate and compare the power of the tests
power_results <- aggregate(. ~ Delta, data = results, FUN = mean)

# Print the power results
print(power_results)
```

```
##   Delta Z_test_power Equal_Var_t_test_power Welch_t_test_power Wilcoxon_power
## 1     0    0.5003949              0.5018058          0.5003292      0.5040042
## 2     1    0.8551399              0.8514340          0.8495950      0.8332288
## 3     2    0.9953358              0.9938319          0.9933419      0.9909998
## 4     3    0.9999774              0.9999214          0.9998925      0.9998434
## 5     4    1.0000000              0.9999994          0.9999987      0.9999984
## 6     5    1.0000000              1.0000000          1.0000000      1.0000000
```

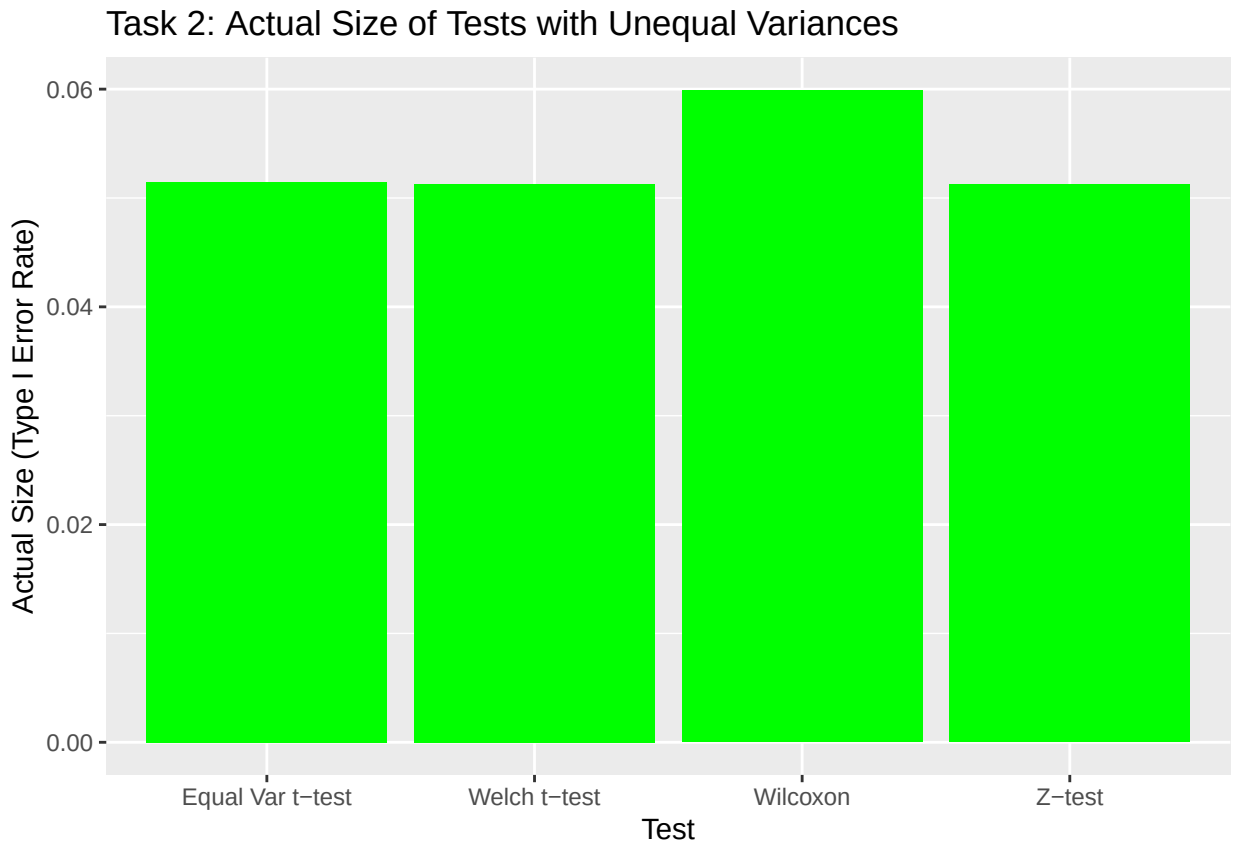
- Report the results from the simulations above using table(s) and/or plot(s). **Interpret your simulation results**. What conclusion can be draw from the simulations? Task 1

```r
# Load necessary libraries
library(ggplot2)

# Create a data frame from the Task 1 results
task1_results <- data.frame(
  Test = c("Z-test", "Equal Var t-test", "Welch t-test", "Wilcoxon"),
  Actual_Size = c(0.05091, 0.05135, 0.05135, 0.05134)
)

# Create a bar plot to compare actual size
ggplot(task1_results, aes(x = Test, y = Actual_Size)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Task 1: Actual Size of Tests with Equal Variances",
       x = "Test",
       y = "Actual Size (Type I Error Rate)")
```



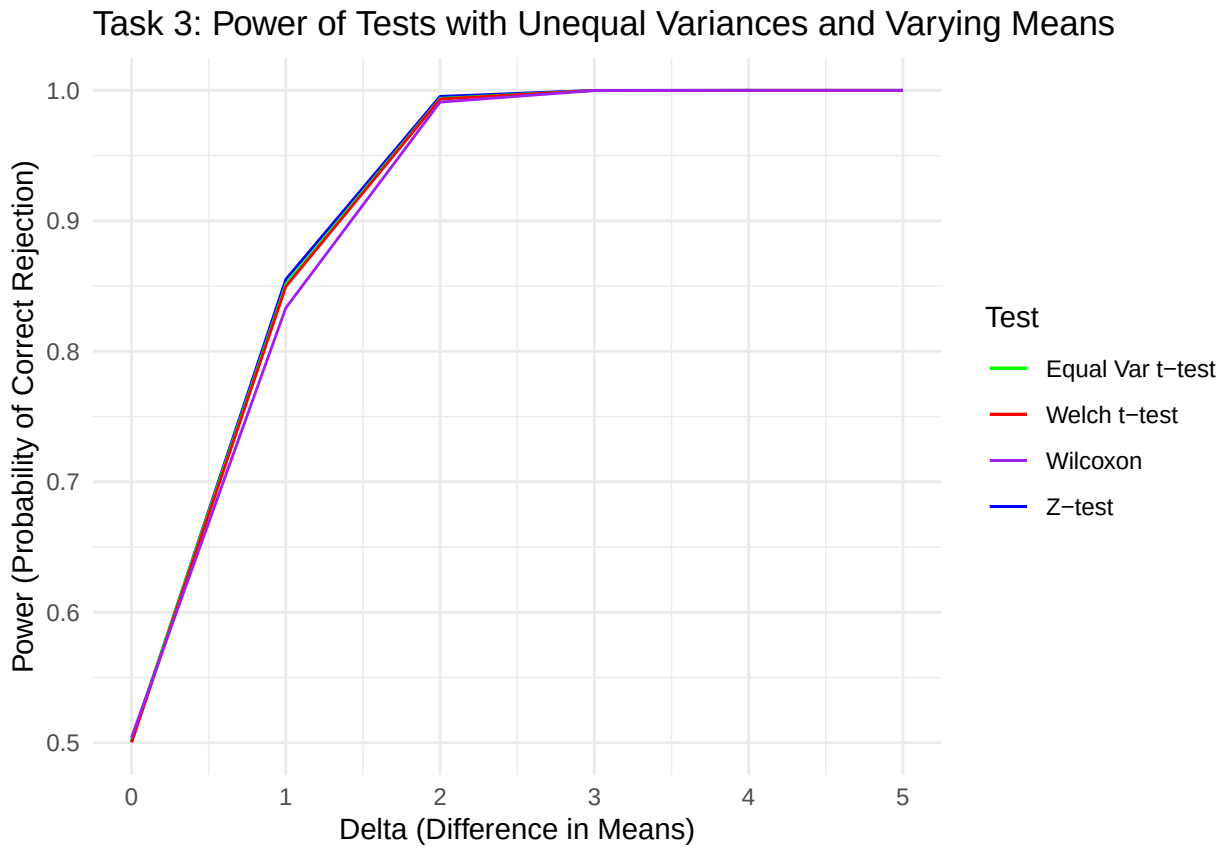
Task 2

```
# Create a data frame from the Task 2 results
task2_results <- data.frame(
  Test = c("Z-test", "Equal Var t-test", "Welch t-test", "Wilcoxon"),
  Actual_Size = c(0.05124, 0.05149, 0.05126, 0.05989)
)

# Create a bar plot to compare actual size
ggplot(task2_results, aes(x = Test, y = Actual_Size)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Task 2: Actual Size of Tests with Unequal Variances",
       x = "Test",
       y = "Actual Size (Type I Error Rate)")
```



Task 3

```
# Create a line plot to compare power for all scenarios
ggplot(power_results, aes(x = Delta)) +
  geom_line(aes(y = Z_test_power, color = "Z-test")) +
  geom_line(aes(y = Equal_Var_t_test_power, color = "Equal Var t-test")) +
  geom_line(aes(y = Welch_t_test_power, color = "Welch t-test")) +
  geom_line(aes(y = Wilcoxon_power, color = "Wilcoxon")) +
  labs(title = "Task 3: Power of Tests with Unequal Variances and Varying Means",
       x = "Delta (Difference in Means)",
       y = "Power (Probability of Correct Rejection)",
       color = "Test") +
  scale_color_manual(values = c("Z-test" = "blue", "Equal Var t-test" = "green", "Welch t-test" = "red"
  theme_minimal()
```



Conclusion:

When we comparing size of tests with equal variances in 1.simulation, all tests maintain their Type I error rate close to the significance level (alpha) under the assumption of equal variances.

Comparing Size of Tests with Unequal Variances in 2.simulation, similar to Task 1, all tests maintain their nominal size even when variances are unequal. This indicates that the tests are robust to unequal variances.

Comparing Power of Test for 3.simulation, for the available scenarios, the power of the tests increases as the difference in means (delta) between the two populations becomes larger. Parametric and Non-parametric test both demonstrate good powers, when difference in the mean increase.