ST502: Final Project - Part 2

Apostolos Stamenos & Tyler Pollard

4/19/2022

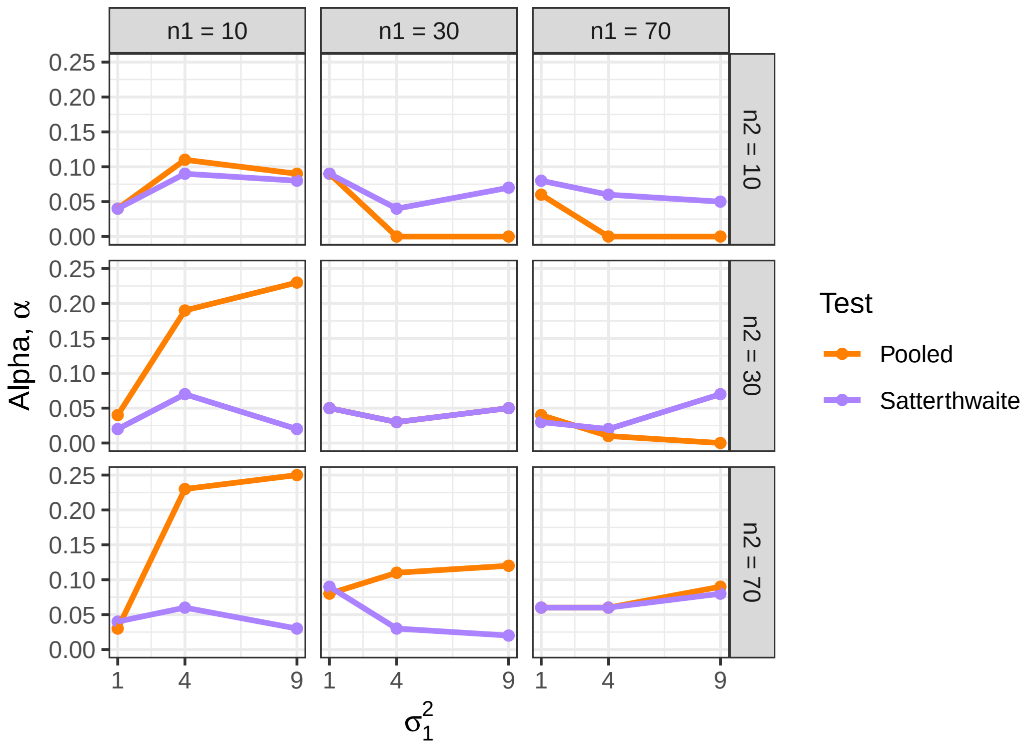
In the previous section, we concluded that the t-test with the Satterthwaite approximation is more appropriate when there is no evidence that the population variances are equal. In order to understand the properties of both tests and to verify that our conclusion makes sense, we conducted a simulation study. A numerical simulation study can be used to evaluate the two testing procedures because unlike in a real data analysis, we know exactly what the true population parameters are. By generating enough datasets through a data-generating mechanism that we control, we can evaluate how the two testing procedures perform when drawing inferences about the underlying populations. For this simulation study, two random samples of data were generated from normal distributions with varying population characteristics outlined in Table 1.

Table 1: **Varying Population Characteristics**

Population Characteristics	Values
True Variance	$\sigma_1^2 = 1, 4, 9$ and $\sigma_2^2 = 1$
Sample Size	$n_1 = 10, 30, 70$ and $n_2 = 10, 30, 70$
True Mean Difference	$\Delta = \mu_1 - \mu_2 = -5, -1, 0, 1, 5$

All combinations of these population characteristics were used to generate data for a total of 135 unique combinations. Any data that was generated for a true mean difference of 0 correspond to data from a null situation and the rest of the generated data correspond to data from an alternative situation. For a given combination of population characteristics, 100 unique data sets of two samples were randomly generated from their corresponding normal distribution. Both two-sample t-tests when equal variance is assumed (pooled) and when unequal variances are assumed (with Satterthwaite approximation for the degrees of freedom) were conducted, using the p-value method, on each of the 100 randomly generated data sets for all 135 combinations of population characteristics. The p-value for each of the randomly generated data sets was compared to the set significance level of $\alpha = 0.05$. If the p-value was less than alpha then the test would reject the null hypothesis of equal population means. The proportion of times we rejected the null hypothesis was used to estimate alpha when the data was generated from a null situation and power when the data was generated from an alternative situation.

Figure 1: **Approximate alpha curves of the pooled and Satterthwaite t-tests for varying sample sizes and population variances**



The simulated alpha value curves corresponding to the Type I error rate for the assumed equal variance and assumed unequal variance tests, in Figure 1, were approximately equal when the variance of both populations were equal (i.e., $\sigma_1^2 = \sigma_2^2$), regardless of the sample sizes. This makes sense because the equal variance assumption holds and the tests should be expected to produce similar results. When the sample sizes are the same (i.e., $n_1 = n_2$), the simulated alpha values of the two tests are also approximately equal, regardless of the varying σ_1^2. For $n_1 > n_2$ and $\sigma_1^2 \neq \sigma_2^2$, the simulated alpha values for the pooled variance t-test are less than that of the Satterthwaite t-test and approach 0 as σ_1^2 increases, whereas for $n_1 < n_2$, the alpha values for the pooled variance t-test are much larger than that of the Satterthwaite t-test as σ_1^2 increases. Since we used the significance level of $\alpha = 0.05$ in our rejection criteria, we should expect the simulated Type I error rate to be upper bounded by 0.05. Regardless of the population charactersitics used to generate the data, the assumed unequal variance t-test performed well, estimating the Type I error rate to be within ± 0.05 of the controlled alpha value with about half of the empirical values above and half below, possibly due to randomness. The assumed unequal variance test performed well when $\sigma_1^2 = \sigma_2^2$ or $n_1 = n_2$, however when $\sigma_1^2 \neq \sigma_2^2$ and $n_1 \neq n_2$, the test performed poorly, with an empirical Type I error rate deviating more than expected from our controlled alpha value.

Figure 2: **Approximate power curves of the pooled and Satterthwaite t-tests for varying samples sizes and population variances**

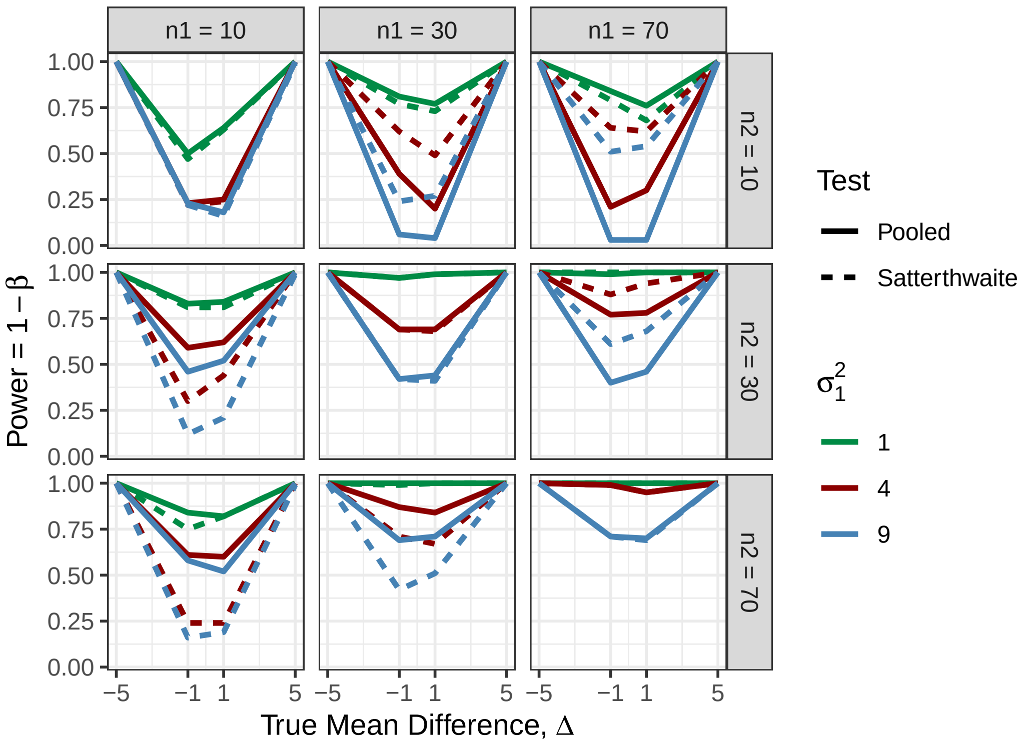


Figure 2 shows the simulated power curves based on the study design. Regardless of n_1 and n_2, the simulated power of both the pooled and Satterthwaite t-tests is greater for smaller values of σ_1^2. This result makes sense intuitively. When the variance of one population is small, it is easier for the hypothesis testing procedure to detect the true mean difference. When the sample sizes are the same (i.e., $n_1 = n_2$), the simulated powers of the pooled and Satterthwaite tests are approximately equal, regardless of σ_1^2. For $n_1 > n_2$, the Satterthwaite t-test tends to perform better, whereas for $n_1 < n_2$, the pooled variance t-test tends to perform better. As n_1 and n_2 increase, so does the power of both the pooled and Satterthwaite t-tests. This makes sense because for both tests, the standard error (i.e., the denominator of the test statistic) gets smaller and the degrees of freedom get larger for large sample sizes.

Our simulation study indicates that when the sample sizes are the same, the two tests give equivalent results. When the sample sizes are different, there is a tradeoff between power and the Type I error rate. The power and and Type I error rate vary based on the true population variances. Thus, which test performs best empirically depends on characteristics of the sample and the population.

Appendix A: R Code

```r
## Load required libraries
library(tidyverse)
library(data.table)
library(MESS)
library(car)

## =========================================== PART 2 ===============================================
## Varying parameters
# True variances
true_var1 <- c(1, 4, 9)
true_var2 <- 1

# Sample Sizes
n1 <- c(10, 30, 70)
n2 <- c(10, 30, 70)

# True mean difference mu1 - mu2
true_mean_diff <- c(-5, -1, 0, 1, 5)

# Setting the seed
set.seed(4242022)

## Function to generate datasets for simulation study
generate_data <- function(replicates = 100, n1, n2, var1, var2, mean_difference, mu1 = 137) {
  nonsmokers <- matrix(NA, nrow = n1, ncol = replicates)
  smokers <- matrix(NA, nrow = n2, ncol = replicates)

  # Generate multiple datasets
  for(i in 1:replicates) {
    nonsmokers[, i] <- rnorm(n1, mu1, sqrt(var1))
    smokers[, i] <- rnorm(n2, mu1 - mean_difference, sqrt(var2))
  }
  return(list(nonsmokers = nonsmokers, smokers = smokers))
}

## Generate all possible data sets
generated_data_sets <- list()
for(i in true_mean_diff){
  for(j in n1){
    for(k in n2){
      for(l in true_var1){
        data_name <- paste0( "true_mean_diff=", i, ".", "n1=", j,
                             ".", "n2=", k, ".", "var1=", l, ".", "var2=1")
        generated_data_sets[[data_name]] <- generate_data(replicates = 100,
                                                          n1 = j,
                                                          n2 = k,
                                                          var1 = 1,
                                                          var2 = 1,
                                                          mean_difference = i,
                                                          mu1 = 137)
      }
    }
  }
}

## Pooled and Satterthwaite Approximation Two Sample t-test p-value method function
# Create function for calculating p-value
p_value <- function(data1, data2, equal.variance = FALSE){
  # Determine number of data points in each data set
  n1 <- length(data1)
  n2 <- length(data2)

  # Calculate sample means
  samp_mean1 <- sum(data1)/n1
  samp_mean2 <- sum(data2)/n2

  # Calculate sample variances
  samp_var1 <- sum((data1 - samp_mean1)^2)/(n1 - 1)
  samp_var2 <- sum((data2 - samp_mean2)^2)/(n2 - 1)

  # Calculate degrees of freedom and observed t statistic for Pooled t test
  if(equal.variance == TRUE){
    ## Pooled Two Sample t-test p-value method
    # Calculate degrees of freedom
    df <- n1 + n2 - 2

    # Calculate sample mean difference
    diff <- samp_mean1 - samp_mean2
    # Assign hypothesized difference equal to 0 to represent Null hypothesis
    D_0 <- 0

    # Calculate pooled sample variance
    samp_var_pooled <- ((n1-1)*samp_var1+(n2-1)*samp_var2)/df

    # Calculate pooled standard error
    se_pooled <- (sqrt(samp_var_pooled)*sqrt(1/n1+1/n2))

    # Calculate pooled t statistic
    T_pooled <- (diff - D_0)/se_pooled

    # Calculate pooled probability
    p_value <- 2*pt(abs(T_pooled), df = df, lower.tail = FALSE)
  }
  # Calculate degrees of freedom and observed t statistic for Satterthwaite t test
  else{
    # Calculate degrees of freedom for t test
    nu <- (samp_var1/n1 + samp_var2/n2)^2/(
      (samp_var1/n1)^2/(n1 - 1) + (samp_var2/n2)^2/(n2 - 1)
    )
    # Round down nu value to whole number
    nu <- floor(nu)

    # Calculate observed t test statistic
    t_obs_satterthwaite <- (samp_mean1 - samp_mean2)/sqrt(samp_var1/n1 + samp_var2/n2)

    # Calculate p-value
    p_value <- 2*pt(abs(t_obs_satterthwaite), df = nu, lower.tail = FALSE)
  }

  return(p_value)
}

## Calculate Simulated alpha and power values
simulated_values <- function(data, equal.variance = FALSE){
  # Create empty vectors to store simulated alpha and power values
  # for each unique combination of parameters in generated data
  prob_values <- c(rep(NA, length(data)))
  metric <- c(rep(NA, length(data)))

  # Create for loop to run through each of the data sets in generated data
  for(m in 1:length(data)){
    # Create empty vectors to store simulated alpha and power values
    # for each of the 100 generated data sets
    # within a unique combination of parameter values in generated data
    simulated_prob_values <- c(rep(NA, 100))

    # Create for loop to run through all 100 generated data sets
    # within a unique combination of parameter values in generated data
    for(n in 1:100){
      # If statement to determine if generated data is from null hypothesis.
      # If it is, calculate p-value and store in simulated_alpha_values vector
      if(grepl("true_mean_diff=0", names(data)[m])){
        # Calculate simulated alpha values for Pooled T Test
        if(equal.variance == TRUE){
          simulated_prob_values[n] <- p_value(data[[m]]$nonsmokers[,n],
                                              data[[m]]$smokers[,n],
                                              equal.variance = TRUE)
          simulated_metric <- "alpha"
        }
        # Calculate simulated alpha values for Satterthwaite T Test
        else{
          simulated_prob_values[n] <- p_value(data[[m]]$nonsmokers[,n],
                                              data[[m]]$smokers[,n],
                                              equal.variance = FALSE)
          simulated_metric <- "alpha"
        }
        # If statement to determine if generated data is from alternative hypothesis.
        # If it is, calculate p-value and store in simulated_power_values vector
      }else{
        # Calculate simulated power values for Pooled T Test
        if(equal.variance == TRUE){
          simulated_prob_values[n] <- p_value(data[[m]]$nonsmokers[,n],
                                              data[[m]]$smokers[,n],
                                              equal.variance = TRUE)
          simulated_metric <- "power"
        }
        # Calculate simulated power values for Satterthwaite T Test
        else{
          simulated_prob_values[n] <- p_value(data[[m]]$nonsmokers[,n],
                                              data[[m]]$smokers[,n],
                                              equal.variance = FALSE)
          simulated_metric <- "power"
        }
      }
    }

    # Calculate proportion of p-values that would result in rejecting null hypothesis
    simulated_prob <- sum(simulated_prob_values < 0.05)/length(simulated_prob_values)

    # Store simulated proportion into alpha and power vectors.
    # If the true mean difference is 0, then the simulated
    # proportion will correspond to alpha and stored, otherwise it is NA.
    # If the true mean difference is not 0, then the
    # simulated proportion will correspond to power and stored, otherwise it is NA.
    prob_values[m] <- simulated_prob
    metric[m] <- simulated_metric
  }
  # Combine results into data frame to output
```

```r
  simulated_df <- data.frame(test = names(data),
                             probs = prob_values,
                             metric = metric
  )
  # Output simulated alpha and power values data frame
  return(simulated_df)
}

# Calculate simulated Satterthwaite alpha and power values data frame from generated data
satterthwaite_simulated_power_df <- simulated_values(generated_data_sets,
                                                      equal.variance = FALSE)
# Add column for test type identifier
satterthwaite_simulated_power_df$type <- "Satterthwaite"

# Calculate simulated Pooled alpha and power values data frame from generated data
pooled_simulated_power_df <- simulated_values(generated_data_sets,
                                              equal.variance = TRUE)
# Add column for test type identifier
pooled_simulated_power_df$type <- "Pooled"

## Data Visuals for Part 2
# Combine the Pooled and Satterthwaite power data frames into one
combined_df <- rbind(pooled_simulated_power_df, satterthwaite_simulated_power_df)
# Create column identifiers for the varying parameters that will be used to parse data in the visuals
combined_df2 <- combined_df %>%
  mutate(
    delta = case_when(
      grepl("true_mean_diff=-5", combined_df$test) ~ -5,
      grepl("true_mean_diff=-1", combined_df$test) ~ -1,
      grepl("true_mean_diff=0", combined_df$test) ~ 0,
      grepl("true_mean_diff=1", combined_df$test) ~ 1,
      grepl("true_mean_diff=5", combined_df$test) ~ 5
    ),
    n1 = case_when(
      grepl("n1=10", combined_df$test) ~ 10,
      grepl("n1=30", combined_df$test) ~ 30,
      grepl("n1=70", combined_df$test) ~ 70,
    ),
    n2 = case_when(
      grepl("n2=10", combined_df$test) ~ 10,
      grepl("n2=30", combined_df$test) ~ 30,
      grepl("n2=70", combined_df$test) ~ 70,
    ),
    var1 = case_when(
      grepl("var1=1", combined_df$test) ~ 1,
      grepl("var1=4", combined_df$test) ~ 4,
      grepl("var1=9", combined_df$test) ~ 9,
    ),
    var2 = 1
  )

# Filter combined data frame to create separate power and alpha data frames
combined_power_df <- combined_df2 %>%
  filter(metric != "alpha") %>%
  mutate(type = as.factor(type),
         var1 = as.factor(var1))
combined_alpha_df <- combined_df2 %>% filter(metric == "alpha")

# Create label names to facet the alpha and power plots by.
# This will allow us to produce a 3x3 grid of plots faceted by the varying sample sizes
n1.labs <- c('n1 = 10', 'n1 = 30', 'n1 = 70')
n2.labs <- c('n2 = 10', 'n2 = 30', 'n2 = 70')
names(n1.labs) <- c(10, 30, 70)
names(n2.labs) <- c(10, 30, 70)

# For alpha create 3x3 figure of varying n's
# On x axis plot var1 and y axis alpha and parse by test
# Create using points with line
ggplot(data = combined_alpha_df) +
  geom_point(aes(x = var1, y = probs, color = type), size = 1.5) +
  geom_line(aes(x = var1, y = probs, color = type), size = 1) +
  facet_grid(n2 ~ n1, labeller = labeller(
    n1 = n1.labs,
    n2 = n2.labs
  )) +
  scale_x_continuous(breaks = c(1,4,9)) +
  scale_colour_manual(name = "Test",
                      values = c('Pooled' = 'darkorange1', 'Satterthwaite' = 'mediumpurple1')) +
  labs(x = expression(sigma[1]^2), y = expression(paste("Alpha, ", alpha))) +
  theme_bw()

# For power create 3x3 figure of varying n's
# On x axis plot true mean diff and y axis power and parse by test and var
# Create 6 line plots for each of the 9 graphs.
# Different line types for each test and different color for each variance.
ggplot(data = combined_power_df) +
  geom_line(aes(x = delta, y = probs, color = var1, linetype = type), size = 1) +
  facet_grid(n2 ~ n1, labeller = labeller(
    n1 = n1.labs,
    n2 = n2.labs
  )) +
  scale_x_continuous(breaks = c(-5,-1,1,5)) +
  scale_linetype_discrete(name = "Test") +
  scale_colour_manual(name = expression(sigma[1]^2),
                      values = c('1' = 'springgreen4', '4' = 'darkred', '9' = 'steelblue')) +
  labs(x = expression(paste("True Mean Difference, ", Delta)),
       y = expression(paste('Power = ', 1-beta))) +
  theme_bw()
```

Team Member Contributions

These reports are the result of extensive collaboration facilitated by Zoom meetings and GitHub. Both group members took turns coding and writing the report. We had long discussions to figure out the best way to conduct the hypothesis tests and simulation study, and how to effectively present our findings.

Apostolos: performed the pooled variance t-test, wrote the code for many of the plots comparing smokers and nonsmokers, wrote the section assessing the normality assumption, wrote the section on testing the equality of variances, wrote a function to generate datasets for the simulation, wrote the section justifying the use of a simulation study, and wrote the section discussing the power of the tests.

Tyler: managed pull requests to the GitHub repository, performed the Satterthwaite t-test, wrote a function to calculate the probability of rejecting H_0 for the simulation, wrote code to calculate and process the simulated values of α and power, wrote the section discussing the design of the simulation study, and wrote the section discussing the Type I error rate of the tests.