HW 8 HYPO TEST

STAT 5400

Due: Oct 25, 2024. 9:30 AM

Problems

Submit your solutions as an .Rmd file and accompanying .pdf file. Include all the **relevant** R code and output. Always interpret your result whenever it is necessary.

Reading assignments.

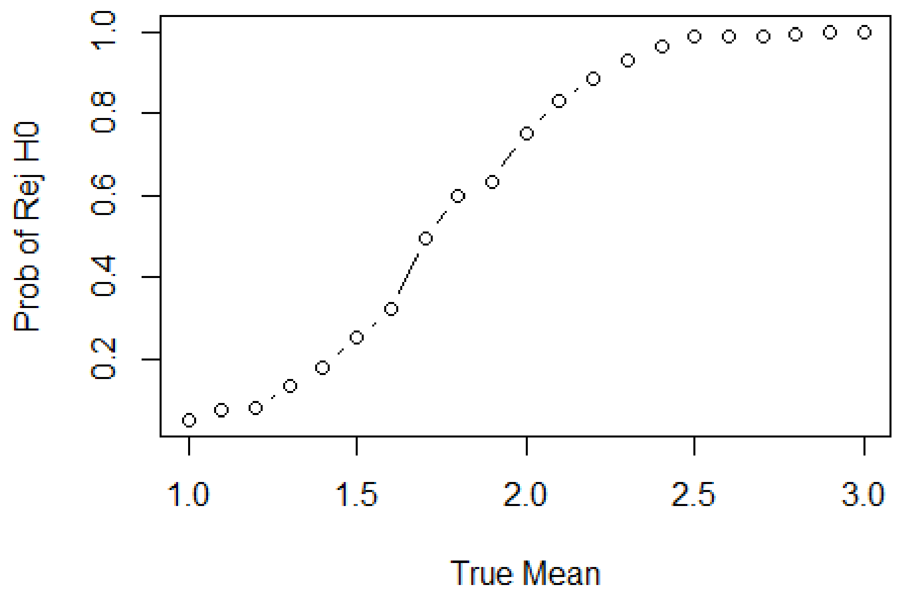
Find a textbook or google some online lecture notes if you are not familiar with (1) one-sample t-test with one-sided and two-wided alternatives, (2) two-sample independent t-test assuming or without assuming equal variance, (3) paired t-test.

Problems

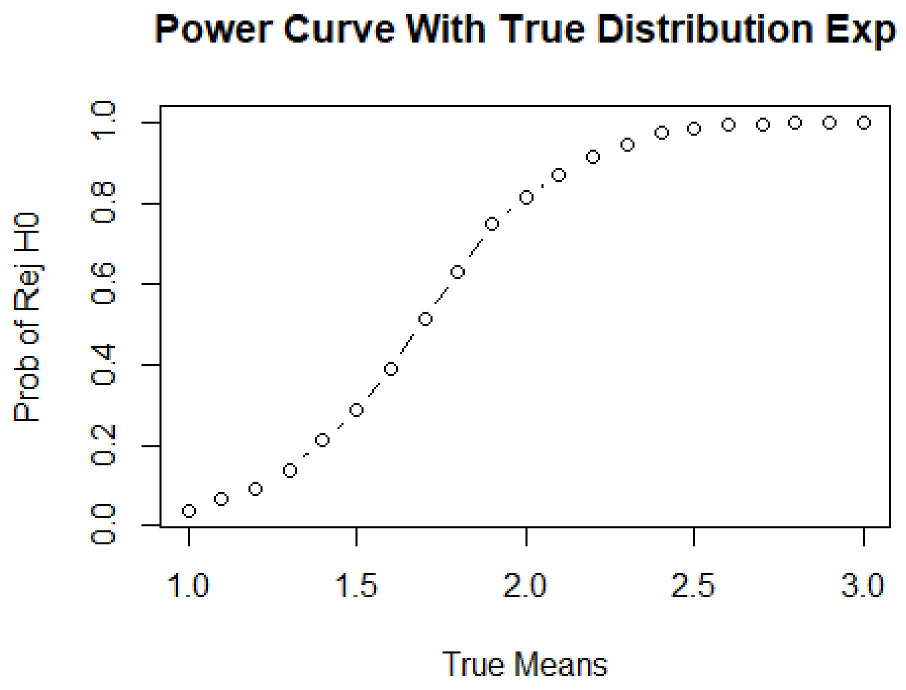
1. Filling in the missing pieces on lecture note S3P3

- Fill in the missing piece on slide 26 and 34. You only need to **submit the two missing lines** not the whole code. If you use the same seed as on the side, namely 5400, you should get the same plots on slide 27 and 35.

```
set.seed(5400)
n <- 30
mulist <- seq(1, 3, 0.1)
sig <- 2
alp <- 0.05
mu_0 <- 1
rejprobs <- sapply(mulist, function(mu) { pvals <- replicate(1000, t.test(rno
rm(n, mu, sig), mu = mu_0)$p.value)
mean(pvals < alp)
})




plot(y=rejprobs, x=mulist, type="b",
xlab="True Mean", ylab="Prob of Rej H0")
```



```r
#Power Curve
set.seed(5400)

n <- 10
mulist <- seq(1, 3, 0.1)
alp <- 0.05
mu_0 <- 1

pvalDist <- function(mu_true) {
  rejections <- 0

  for (j in 1:1000) {
    sample_data <- rnorm(n, mean = mu_true, sd = 1)
    t_test <- t.test(sample_data, mu = mu_0)

    if (t_test$p.value < alp) {
      rejections <- rejections + 1
    }
  }

  return(rejections / 1000)  # Calculate and return rejection probability
}

rejprobs <- sapply(mulist, pvalDist)

plot(y = rejprobs, x = mulist, type = "b",
     xlab = "True Means", ylab = "Prob of Rej H0",
     main = "Power Curve With True Distribution Exp")
```

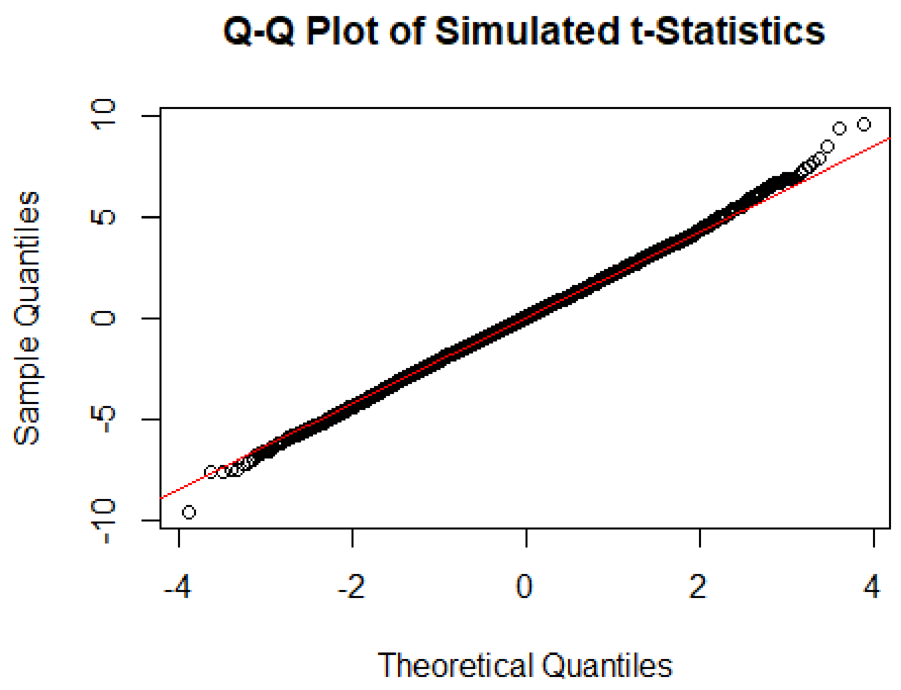


2. Hypothesis testing when the independence assumption is violated.

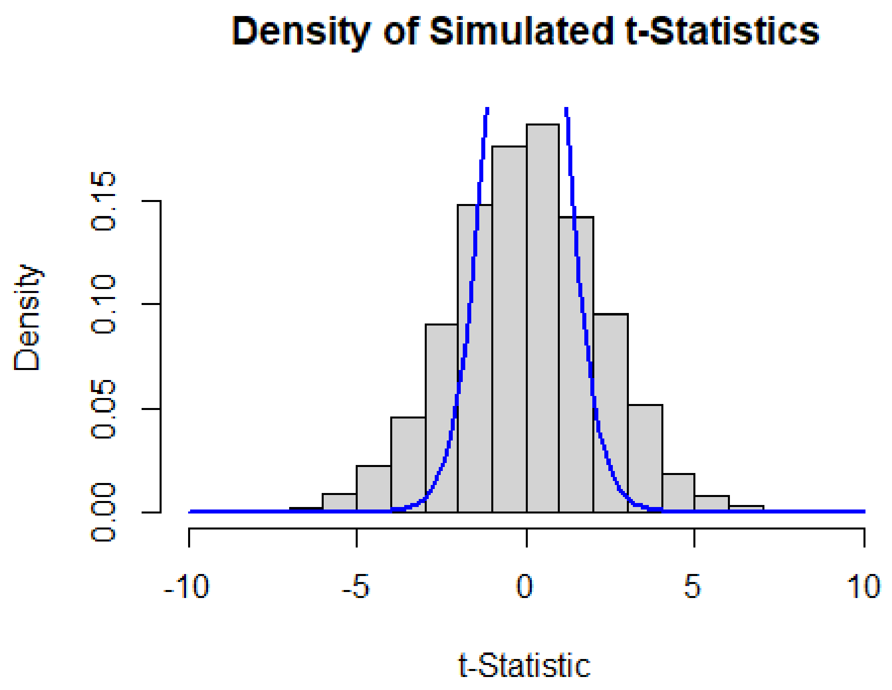
One-sample t-test assumes independent observations. The goal of this problem is to investigate the bad performance when one-sample t-test is applied on dependent samples.

- Generate a dependent sample $X_1, \dots, X_{30}$ from standard normal distribution with $\text{Cov}(X_i, X_j) = 0.1$ for each pair $i \neq j$. The true variance σ^2 is unknown.
- Conduct a one-sample t-test for $H_0: \mu = 0$ vs $H_1: \mu \neq 0$. Compute the test statistic and the observed p-value. What is your decision?
- Let us consider the case when H_0 is true. We then know the resulting t-statistics should follow a t_{29} distribution.
    - Design a simulation study to generate realizations of the t-statistics, and then produce a Q-Q plot to check if the generated t-statistics comfort with the distribution t_{29}.
    - Also use `density` function to plot the density of the generated t-statistics. Compare the density with the density of t_{29}.
- Use simulations to generate realizations of the random p-values. What is the estimated Type I error?
- Produce two plots for the estimated power curve against different sample sizes and different true means, respectively.

```r
library(MASS)
set.seed(5400)

# Define parameters for the multivariate normal distribution
mean_vals <- rep(0, 30)  # Mean vector of zeros
cov_matrix <- matrix(0.1, nrow = 30, ncol = 30) + 0.9 * diag(30)  # Covarianc
e matrix with 0.1 off-diagonal and 1.0 on-diagonal

# Generate a correlated sample
sample_data <- mvrnorm(n = 1, mu = mean_vals, Sigma = cov_matrix)

# Conduct a one-sample t-test
t_test_result <- t.test(sample_data, mu = 0)

# Print the test results
cat("Test statistic:", t_test_result$statistic, "\n")

## Test statistic: -4.031398

cat("p-value:", t_test_result$p.value, "\n")

## p-value: 0.0003673444

# Simulation study for generating t-statistics
simulated_t_statistics <- replicate(10000, {
  sim_data <- mvrnorm(n = 1, mu = mean_vals, Sigma = cov_matrix)
  sim_t_test <- t.test(sim_data, mu = 0)
  sim_t_test$statistic
})

# Q-Q plot of the simulated t-statistics
qqnorm(simulated_t_statistics, main = "Q-Q Plot of Simulated t-Statistics")
qqline(simulated_t_statistics, col = "red")
```

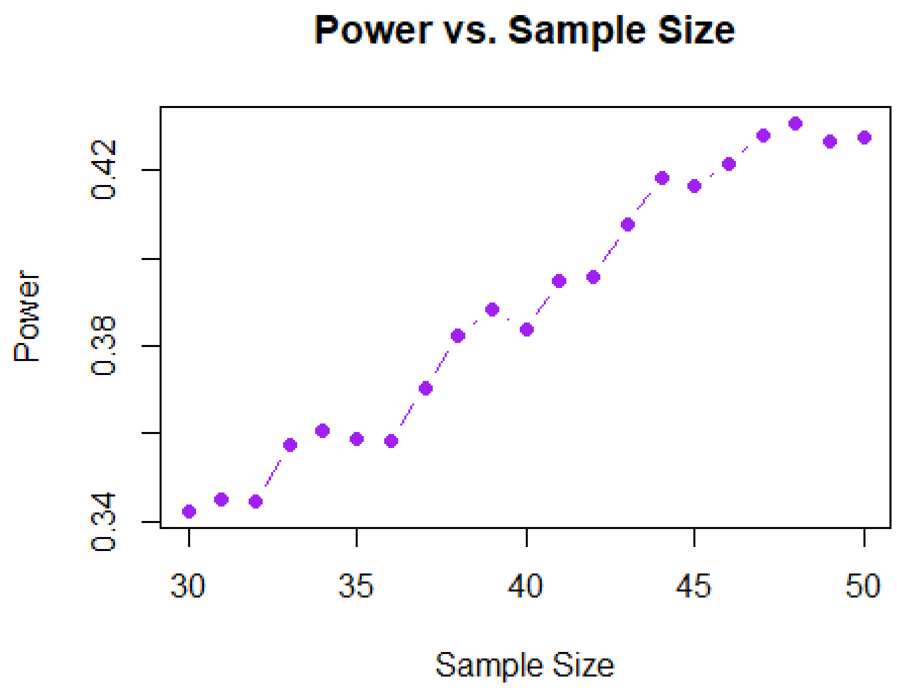


```r
# Histogram of simulated t-statistics with a theoretical t-distribution overl
ay
hist(simulated_t_statistics, freq = FALSE, main = "Density of Simulated t-Sta
tistics", xlab = "t-Statistic")
curve(dt(x, df = 29), add = TRUE, col = "blue", lwd = 2)
```

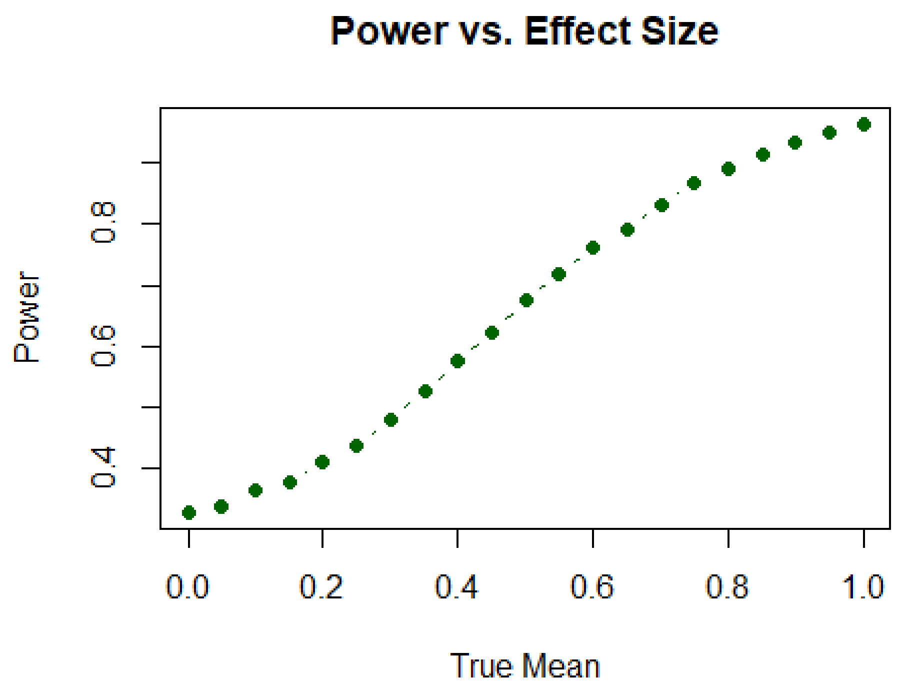


```
# Estimation of p-values and Type I error
simulated_p_values <- replicate(10000, {
  sim_data <- mvrnorm(n = 1, mu = mean_vals, Sigma = cov_matrix)
  sim_p_value <- t.test(sim_data, mu = 0)$p.value
  sim_p_value
})
estimated_type_I_error <- mean(simulated_p_values < 0.05)
cat("Estimated Type I error rate:", estimated_type_I_error, "\n")

## Estimated Type I error rate: 0.3386

# Power curve based on varying sample sizes
test_sample_sizes <- 30:50
power_estimates_by_size <- sapply(test_sample_sizes, function(size) {
  adjusted_cov_matrix <- matrix(0.1, nrow = size, ncol = size) + 0.9 * diag(s
ize)
  simulated_p_values_size <- replicate(10000, {
    sim_data_size <- mvrnorm(n = 1, mu = rep(0, size), Sigma = adjusted_cov_m
atrix)
    sim_p_value <- t.test(sim_data_size, mu = 0)$p.value
    sim_p_value
  })
  power_estimate <- mean(simulated_p_values_size < 0.05)
  power_estimate
})

# Plot power curve versus sample sizes
```

```r
plot(test_sample_sizes, power_estimates_by_size, type = "b", pch = 19, col =
"purple",
     xlab = "Sample Size", ylab = "Power", main = "Power vs. Sample Size")
```



```r
# Power curve for different effect sizes (true means)
effect_sizes <- seq(0, 1, length.out = 21)
power_estimates_by_effect <- sapply(effect_sizes, function(true_mean) {
  adjusted_mean_vals <- rep(true_mean, 30)
  simulated_p_values_effect <- replicate(10000, {
    sim_data_effect <- mvrnorm(n = 1, mu = adjusted_mean_vals, Sigma = cov_ma
trix)
    sim_p_value <- t.test(sim_data_effect, mu = 0)$p.value
    sim_p_value
  })
  power_estimate <- mean(simulated_p_values_effect < 0.05)
  power_estimate
})

# Plot power curve versus true mean values (effect sizes)
plot(effect_sizes, power_estimates_by_effect, type = "b", pch = 19, col = "da
rkgreen",
     xlab = "True Mean", ylab = "Power", main = "Power vs. Effect Size")
```



3. Two-sample t-test.

Suppose $X_1, \dots, X_{n_1}$ are iid $\sim N(\mu_1, \sigma^2)$, $Y_1, \dots, Y_{n_2}$ are iid $\sim N(\mu_2, \sigma^2)$, and $X_i's$ and $Y_i's$ are also independent. We want to test $H_0: \mu_1 = \mu_2$ vs $H_0: \mu_1 \neq \mu_2$. We typically use the following test statistic:

$$T = \frac{\bar{X} - \bar{Y}}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}},$$

where

$$S_p = \sqrt{\frac{1}{n_1 + n_2 - 2}[(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2]};$$

$\bar{X}$ and $\bar{Y}$ are sample means and S_1^2 and S_2^2 are sample variances. When H_0 is true, the test statistic T should follow a $t_{n_1+n_2-2}$ distribution. Therefore we reject H_0 if $|T| > t_{\alpha/2, n_1+n_2-2}$.

- For the above two-sample t-test, Our goal is to simulation the distribution of random p-values. Write a function `pval2SampleT` that generates realizations of random p-values. The data-generating model in this function is normal. Your function should have the following arguments:
  - `mu1`, μ_1, mean of X,
  - `mu2`, μ_2, mean of Y,

- var1, σ_1^2, variance of X,
- var2, σ_2^2, variance of Y,
- n1, sample size of X,
- n2, sample size of Y,
- alp, the significance level, default is 5e-2.
- B, number of replications.

The function should generate a B-vector containing realizations of random p-values.

```r
pval2SampleT <- function(mu1, mu2, var1, var2, n1, n2, alp = 0.05, B = 1000)
{
  p_values <- replicate(B, {
    sample_x <- rnorm(n1, mean = mu1, sd = sqrt(var1))
    sample_y <- rnorm(n2, mean = mu2, sd = sqrt(var2))

    t_test <- t.test(sample_x, sample_y)
    p_values <- t_test$p.value
    p_values


  })

  return(p_values)

}

#Parameters
mu1 <- 0
mu2 <- 0
var1 <- 1
var2 <- 1
n1 <- 30
n2 <- 30
alp <- 0.05
B <- 1000  # Number of replications



p_values <- pval2SampleT(mu1, mu2, var1, var2, n1, n2, alp, B)


# Estimate Type I error
type_I_error_estimate <- mean(p_values < alp)
print(paste("Estimated Type I error:", type_I_error_estimate))

## [1] "Estimated Type I error: 0.054"
```

- Give values of `mu1`, `mu2`, `var1`, `var2`, `n1`, `n2`, `alp`, by yourself, and use your `pval2SampleT` function to estimate the Type I error.

4. Inference for Poisson distributions.

Suppose $X_1, \dots, X_{20}$ is a random sample for Poison distribution with mean $\lambda = 5$. Large sample theory suggests that $\bar{X}$ is approximately $\mathrm{N}(\lambda, \lambda/n)$. Let $\alpha = 0.05$.

- We want to test $H_0: \lambda = 5$ against $H_1: \lambda \neq 5$ with a test statistic:

  $$T = \frac{\bar{X} - \lambda}{S/\sqrt{n}},$$

  where S is the sample standard deviation. We reject H_0 if $|T| > t_{\alpha/2,n-1}$. Denote the Type I error by p. Use simulations with 10^4 replicates to estimate p. Create a 99% score interval for p. Is α captured in your 99% score confidence interval?

- We want to construct a 95% confidence interval for λ by

  $$\bar{X} \pm t_{\alpha/2,n-1} S/\sqrt{n}.$$

  Denote the coverage probability by p. Use simulations with 10^4 replicates to estiamte p. Create a 99% score confidence interval for p. Is $1 - \alpha$ captured in your 99% score confidence interval?

```r
# Parameters
true_lambda <- 5
sample_size <- 20
significance_level <- 0.05
num_iterations <- 10000

# Initialize vectors to store the results
error_rates <- numeric(num_iterations)
ci_coverage <- numeric(num_iterations)

for (iteration in 1:num_iterations) {
  # Generate a sample from the Poisson distribution
  sample_data <- rpois(sample_size, true_lambda)

  # Compute the sample mean and standard deviation
  sample_avg <- mean(sample_data)
  sample_sd <- sd(sample_data)

  # Compute the test statistic
  test_statistic <- (sample_avg - true_lambda) / (sample_sd / sqrt(sample_siz
e))

  # Compute the critical value
  critical_threshold <- qt(1 - significance_level / 2, sample_size - 1)
  # Check if we reject the null hypothesis
  error_rates[iteration] <- abs(test_statistic) > critical_threshold

  # Construct a 95% confidence interval for lambda
  ci_lower_bound <- sample_avg - critical_threshold * (sample_sd / sqrt(sampl
e_size))
  ci_upper_bound <- sample_avg + critical_threshold * (sample_sd / sqrt(sampl
e_size))

  # Check if the confidence interval covers the true value of lambda
  ci_coverage[iteration] <- ci_lower_bound <= true_lambda & true_lambda <= ci
_upper_bound
}

# Estimate the Type I error rate and coverage probability
estimated_error_rate <- mean(error_rates)
estimated_coverage <- mean(ci_coverage)

# Compute 99% score confidence intervals for estimated_error_rate and estimat
ed_coverage
error_rate_ci <- qnorm(c(0.005, 0.995), mean = estimated_error_rate, sd = sqr
t(estimated_error_rate * (1 - estimated_error_rate) / num_iterations))
coverage_ci <- qnorm(c(0.005, 0.995), mean = estimated_coverage, sd = sqrt(es
timated_coverage * (1 - estimated_coverage) / num_iterations))

# Print results
cat("Estimated Type I error rate: ", estimated_error_rate, "\n")

## Estimated Type I error rate:  0.0519

cat("99% score confidence interval for Type I error rate: ", error_rate_ci, "
\n")

## 99% score confidence interval for Type I error rate:  0.04618616 0.0576138
4

cat("Estimated coverage probability: ", estimated_coverage, "\n")

## Estimated coverage probability:  0.9481

cat("99% score confidence interval for coverage probability: ", coverage_ci,
"\n")

## 99% score confidence interval for coverage probability:  0.9423862 0.95381
38
```