HW 8 Hypothesis testing

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.

```
#slide 26
rejprobs = sapply(mulist, function(mu){
mean(pvalDist(n,mu,mu_0,sig, alp) < alp)})

#slide 35
rejprobs <- sapply(mulist, function(mu) mean(pvalDist2(n, "exp", mu_0=mu, rate=1/2) < alp))
```

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 $X_1, \ldots, X_{30}$ 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?

```
library(mnorm)
```

```
## Warning: package 'mnorm' was built under R version 4.3.3
```

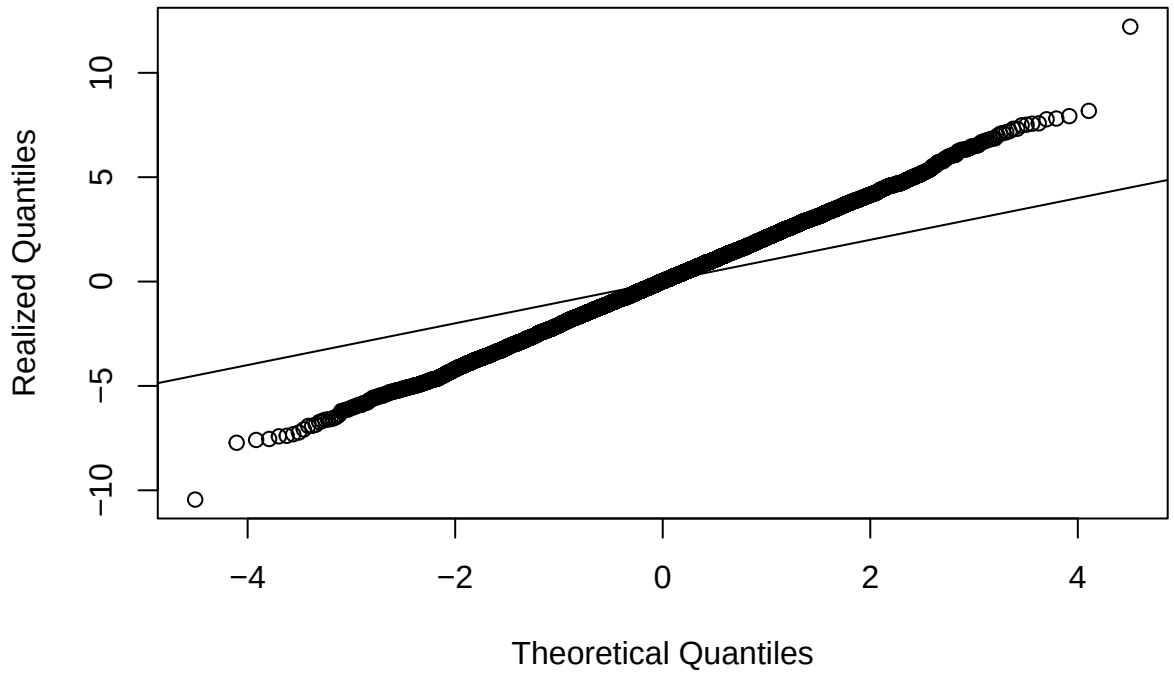
```r
set.seed(5400)

cov <- matrix(rep(0.1,900), 30, 30)
diag(cov) <- 1
XList<-rmnorm(1,rep(0,30),cov)

Xbar <- mean(XList)
t = (Xbar - 0)/(sd(XList)/sqrt(30))

t_test_val <- 2*(pt(-abs(t), df= 30-1, lower.tail = FALSE))

print(t_test_val)
```

```
## [1] 1.750046
```

```r
#As p-value = 0.25 > 0.05, null hypothesis is false and alternative hypothesis: true mean is greater th
```

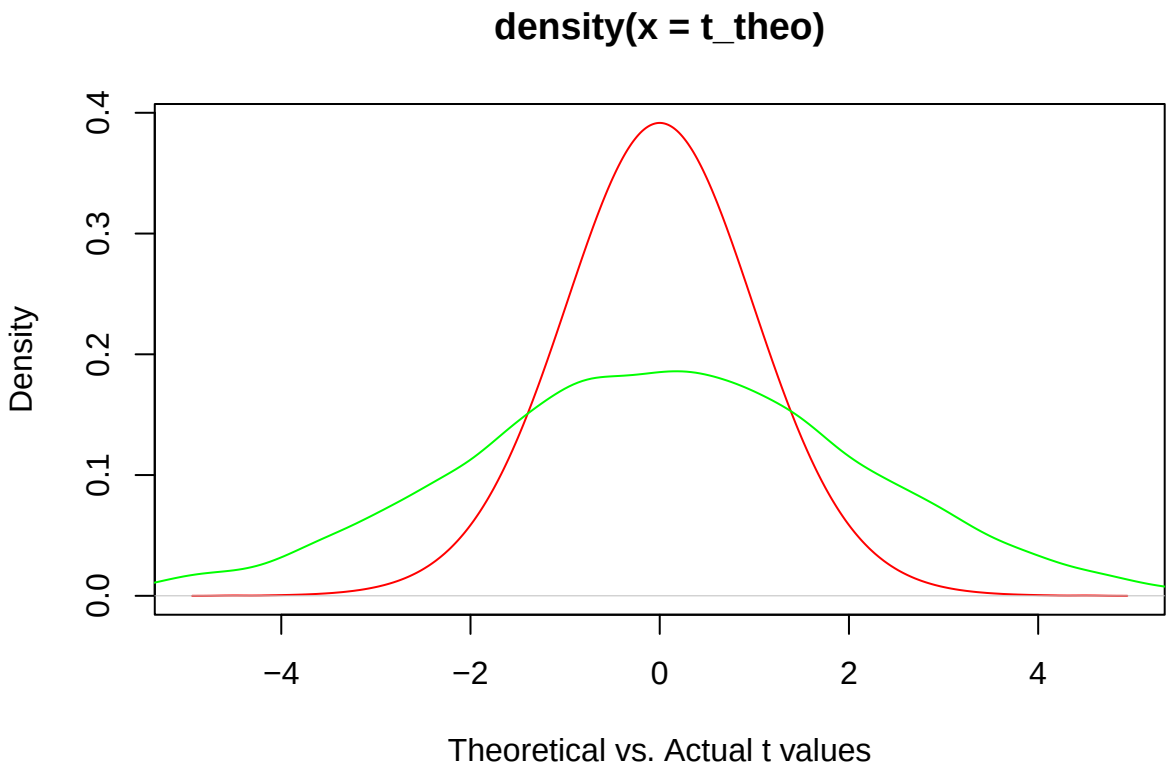
- Let us consider the case when H_0 is true. We then know the resulting t-statistics should follow a t_{29} distribution.
  - Design a simulation study to generate realizations of the t-statistics, and then produce a Q-Q plot to check if the generated t-statistics comfort with the distribution t_{29}.

```r
p_vals <- ppoints(10000)


t_realizations = vector(length = length(p_vals))

for(i in 1:length(p_vals)){

XList<-rmnorm(1,rep(0,30),cov)
 Xbar <- mean(XList)

 t = (Xbar - 0)/(sd(XList)/sqrt(30))
 t_realizations[i] <- t
}


t_theo = qt(p_vals,29)

qqplot(t_theo, t_realizations, xlab = "Theoretical Quantiles", ylab = "Realized Quantiles")
abline(0,1)
```



- Also use `density` function to plot the density of the generated t-statistics. Compare the density with the density of t_{29}.

```
plot(density(t_theo), xlab = "Theoretical vs. Actual t values",type="l",col="red")
lines(density(t_realizations),col="green")
```



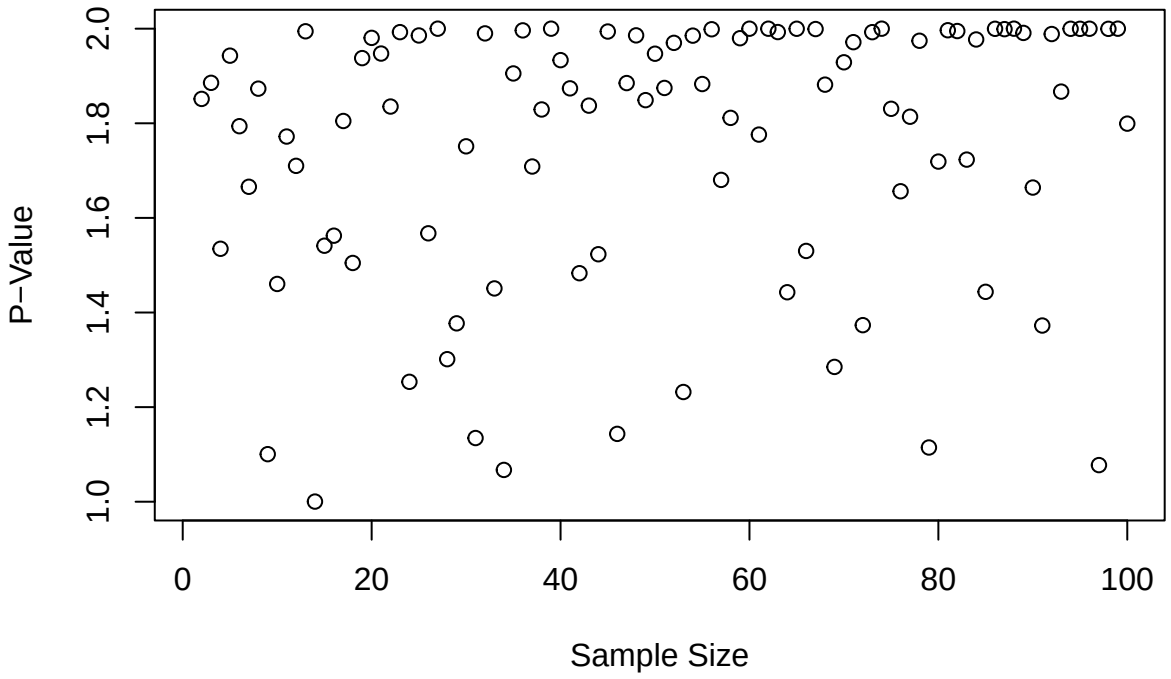
- Use simulations to generate realizations of the random p-values. What is the estimated Type I error?

```r
p_realizations = vector(length = length(p_vals))

for(i in 1:length(p_vals)){

XList<-rmnorm(1,rep(0,30),cov)
 Xbar <- mean(XList)

 t = (Xbar - 0)/(sd(XList)/sqrt(30))

 p_val <- 2*(pt(-abs(t), df= 30-1, lower.tail = FALSE))

 p_realizations[i] <- p_val
}

cat('Est Type 1 error : ', mean(p_realizations < 0.05))
```

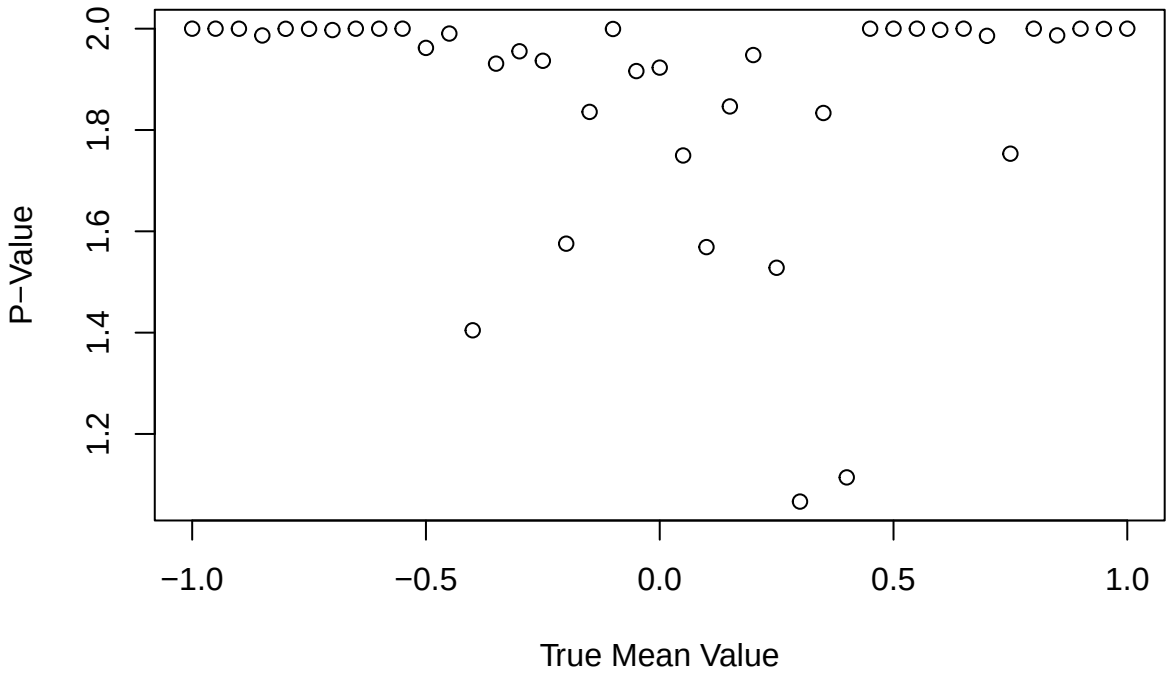
```
## Est Type 1 error :  0
```

- Produce two plots for the estimated power curve against different sample sizes and different true means, respectively.

```r
# Different Sample sizes
pvals = vector(length = 100)

for(n in 1:100) {
  cov <- matrix(rep(0.1,n*n), n, n)
  diag(cov) <- 1
  XList<-rmnorm(1,rep(0,n),cov)

  Xbar <- mean(XList)
  t = (Xbar)/(sd(XList)/sqrt(n))

  t_test_val <- 2*(pt(-abs(t), df= n-1, lower.tail = FALSE))

  pvals[n] <- t_test_val
}

plot(1:n, pvals, xlab = "Sample Size", ylab="P-Value")
```




```r
# Different Sample sizes

true_means = seq(-1,1,0.05)
n <- 30
pvals = vector(length = length(true_means))
for(mean_val in 1:length(true_means)) {
  cov <- matrix(rep(0.1,900), n, n)
  diag(cov) <- 1
  XList<-rmnorm(1,rep(true_means[mean_val],n),cov)

  Xbar <- mean(XList)
  t = (Xbar)/(sd(XList)/sqrt(n))

  t_test_val <- 2*(pt(-abs(t), df= n-1, lower.tail = FALSE))

  pvals[mean_val] <- t_test_val
}

plot(true_means, pvals, xlab = "True Mean Value", ylab="P-Value")
```



3. Two-sample t-test. Suppose $X_1, \ldots, X_{n_1}$ are iid $\sim N(\mu_1, \sigma^2)$, $Y_1, \ldots, Y_{n_2}$ 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.

```
pval2SampleT <- function(mu1, mu2, var1, var2, n1, n2, alp, B = 10000) {
  Xlist = matrix(rnorm(B * n1, mean = mu1, sd = sqrt(var1)), B, n1)
  Ylist = matrix(rnorm(B * n2, mean = mu2, sd = sqrt(var2)), B, n2)
```

```
  Xbarlist = apply(Xlist, 1, mean)
  Ybarlist = apply(Ylist, 1, mean)

  XSlist = apply(Xlist, 1, sd)
  YSlist = apply(Ylist, 1, sd)

  Splist = sqrt(
    (1/(n1+n2-2))*((n1-1)*XSlist*XSlist + (n2-1)*YSlist*YSlist)
  )


  T = (Xbarlist - Ybarlist) / (Splist * sqrt(1/n1 + 1/n2))

  #print(T)

  p_values = 2 * pt(-abs(T), df = n1 + n2 - 2, lower.tail = FALSE)
  #2*(pt(t, df= 30-1, lower.tail = FALSE))

  return (p_values)
}
```

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

```
set.seed(5400)

cat('Type 1 error: ', mean(pval2SampleT(0,2,1,4,20, 30, 0.05) < 0.05))
```

```
## Type 1 error:  0
```

4. Inference for Poisson distributions. Suppose $X_1, \ldots, X_{20}$ 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?

```
Xlist <- matrix(rpois(20*10^4,5), 10^4,20)

lambdalist <- apply(Xlist,1,mean)

XbarList <- rowMeans(Xlist)

XsdList <- apply(Xlist,1,sd)

T <- (XbarList - 5) / (XsdList / sqrt(20))
```

```r
p_values = 2 * pt(-abs(T), df = 20-1, lower.tail = FALSE)

type_1_error <- mean(p_values < 0.05)

cat('Type 1 error: ', type_1_error, "\n")
```

```
## Type 1 error:  0
```

```r
z_99 <- qnorm(0.995)
n <- 10^4
lower_bound <- type_1_error - z_99 * sqrt((type_1_error * (1 - type_1_error)) / n)
upper_bound <- type_1_error + z_99 * sqrt((type_1_error * (1 - type_1_error)) / n)
cat('99% score confidence interval for p: [', lower_bound, ',', upper_bound, ']\n')
```

```
## 99% score confidence interval for p: [ 0 , 0 ]
```

- 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
n_replicates <- 10000

t_alpha <- qt(1 - 0.05 / 2, df = 20 - 1)
alpha <- 0.05

CI_lower <- XbarList - t_alpha * (XsdList / sqrt(n))
CI_upper <- XbarList + t_alpha * (XsdList / sqrt(n))

# Check if lambda = 5 is within each CI
coverage <- mean((CI_lower <= 5) & (CI_upper >= 5))
cat('Coverage probability: ', coverage, "\n")
```

```
## Coverage probability:  0.0638
```

```r
# 99% Score Confidence Interval for Coverage Probability
coverage_lower_bound <- coverage - z_99 * sqrt((coverage * (1 - coverage)) / n_replicates)
coverage_upper_bound <- coverage + z_99 * sqrt((coverage * (1 - coverage)) / n_replicates)
cat('99% score confidence interval for coverage probability: [', coverage_lower_bound, ',', coverage_up
```

```
## 99% score confidence interval for coverage probability: [ 0.05750477 , 0.07009523 ]
```

```r
# Check if 1 - alpha is within the confidence interval
cat('Is 1 - alpha within the interval? ', ((1 - alpha) >= coverage_lower_bound & (1 - alpha) <= coverag
```

```
## Is 1 - alpha within the interval?  FALSE
```