STAT5400 - Homework 8

Dev Narayan Baiju

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-sided 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. Code lines for slide 26:

```
rejprobs <- rep(NA, length(mulist))
for (i in seq_along(mulist)) {
  rejprobs[i] <- mean(pvalDist(n, mulist[i], mu0, sig, alp) < alp)
}
```

Code line for slide 34:

```
rejprobs <- colMeans(sapply(mulist, function(x){pvalDist2(n, "exp", mu_0, alp, rate = 1/x)})<0.05)
```

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.

```r
rho <- 0.1
n <- 30
Sigma <- matrix(rho, n, n)
diag(Sigma) <- 1
eig <- eigen(Sigma)
Sigma.sq <- eig$vectors %*% tcrossprod(diag(sqrt(eig$values)), eig$vectors)
set.seed(5400)
X = rnorm(n) %*% Sigma.sq
print(X)
```

```
##           [,1]     [,2]     [,3]       [,4]     [,5]      [,6]      [,7]
## [1,] 1.945287 1.450833 1.161853 -0.4499519 1.689828 0.3140518 0.1298563
##            [,8]       [,9]     [,10]     [,11]      [,12]     [,13]     [,14]
## [1,] -0.1983359 -0.5525435 2.748525 -0.910386 -0.4176109 0.7690731 0.1081913
##          [,15]     [,16]      [,17]     [,18]     [,19]   [,20]     [,21]
## [1,] 1.159028 0.6645582 0.07521743 0.1256431 0.9361093 0.59486 -1.227542
##            [,22]     [,23]     [,24]     [,25]     [,26]      [,27]    [,28]
## [1,] -0.9804229 0.2794597 0.9216069 0.5319572 0.4361402 -0.5827753 1.035458
##           [,29]     [,30]
## [1,] -1.061946 0.8075877
```

- 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?

```r
mu_0 <- 0
alp <- 0.05
(t_stat <- (mean(X) - mu_0) / (sd(X) / sqrt(n)))
```

```
## [1] 2.23522
```

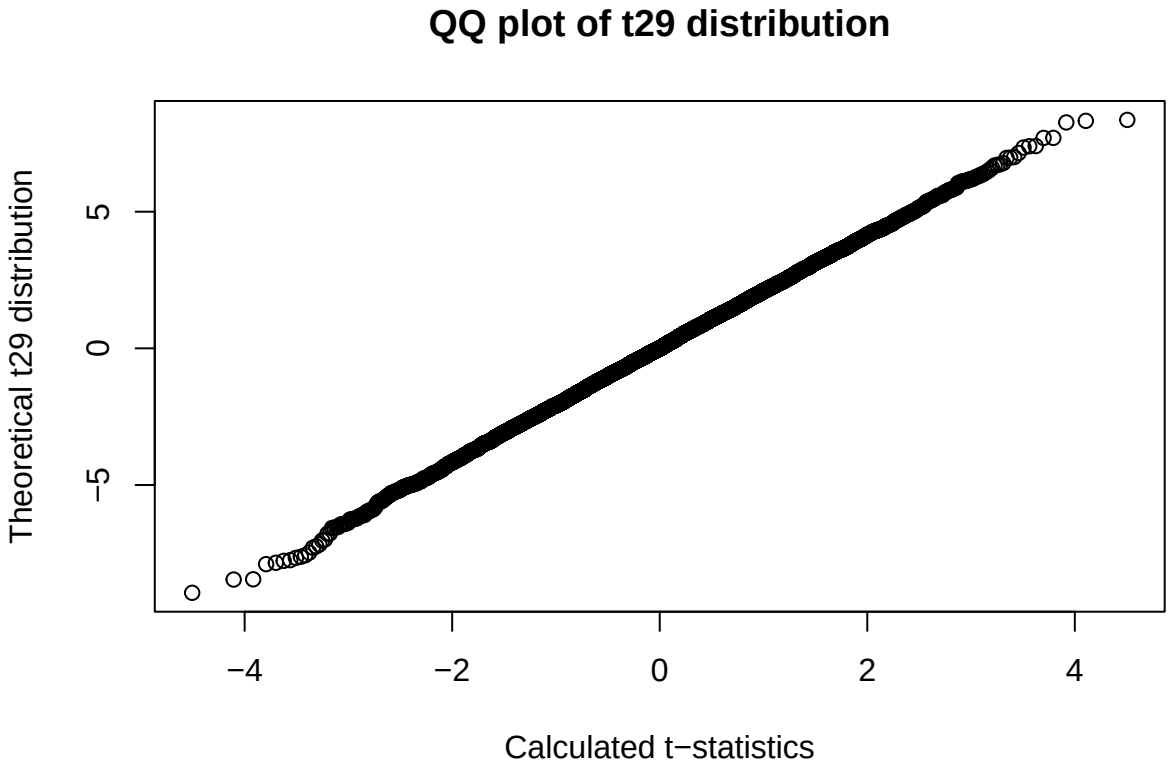
```r
(p_val <- 2 * pt(-abs(t_stat), n-1))
```

```
## [1] 0.03326837
```

Since the p-value (0.033) is less than 0.05, we reject H_0 and conclude that $\mu \neq 0$ (Type I error). + 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}.

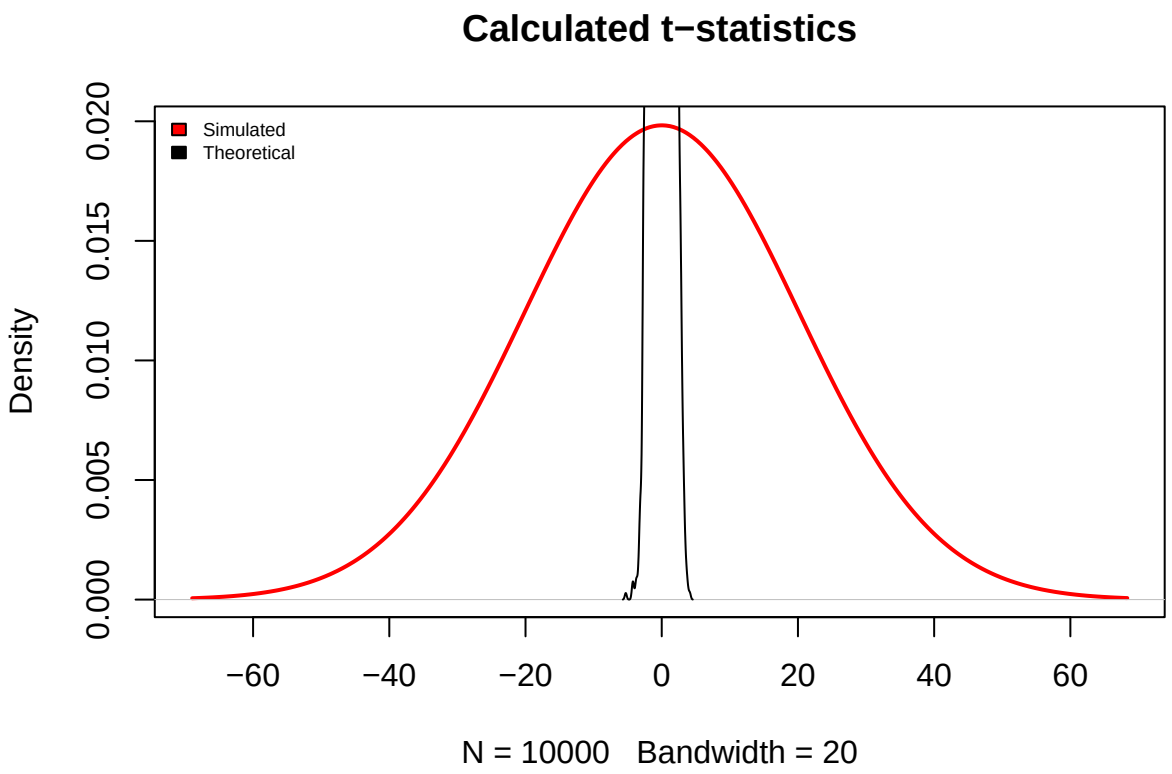
```r
n <- 30
B <- 10000
t_stat_list <- rep(NA, B)
p_val_list <- rep(NA, B)
mu_0 <- 0
set.seed(5400)
Xnormlist <- matrix(rnorm(B * n), B, n)
Xlist <- matrix(, B, n)
for(i in 1:B){
  Xlist[i, ] <- Xnormlist[i, ] %*% Sigma.sq
  t_stat_list[i] <- (mean(Xlist[i, ]) - mu_0) / (sd(Xlist[i, ]) / sqrt(n))
}
```

```r
t29_dist <- qt(ppoints(B), df =29)
plot(t29_dist, sort(t_stat_list),
     xlab = "Calculated t-statistics",
     ylab = "Theoretical t29 distribution", main = "QQ plot of t29 distribution")
```



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

```r
#par(mfrow = c(1,2))
plot(density(t_stat_list, bw = 20), lwd = 2, col = 'red', main = "Calculated t-statistics")
#x_dt <- seq(- 60, 60, by = 120/B)
lines(density(rt(10000, df = 29)))
#curve(dt(x, df = 29), add = TRUE)
legend("topleft", legend = c("Simulated", "Theoretical"), fill = c("red", "black"),
       cex = 0.6, bty = "n")
```



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

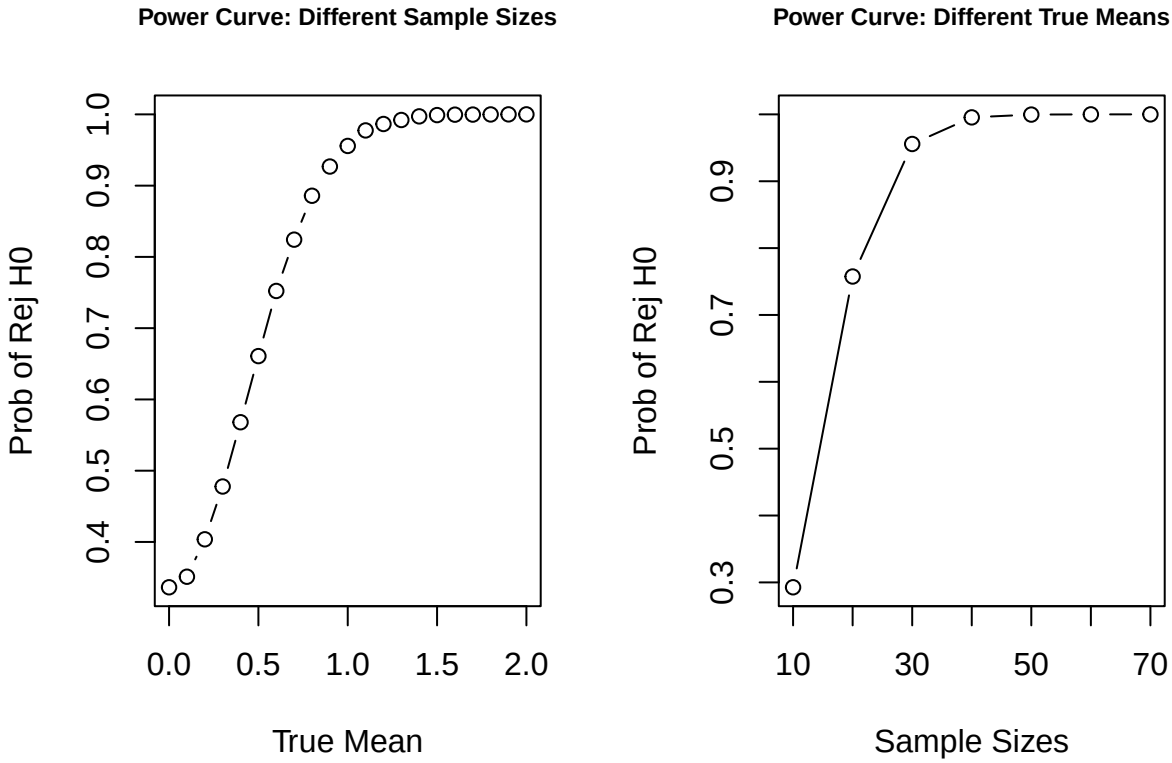
```
for(i in 1:B){
  p_val_list[i] <- 2 * pt(-abs(t_stat_list[i]), n-1)
}
(mean(p_val_list < alp)) #Estimated Type I error
```

```
## [1] 0.3364
```

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

```
pval_func <- function(n = 30, mu = 0){
  rho <- 0.1
  B <- 10000
  mu_0 <- mu
  Sigma <- matrix(rho, n, n)
  diag(Sigma) <- 1
  eig <- eigen(Sigma)
  Sigma.sq <- eig$vectors %*% tcrossprod(diag(sqrt(eig$values)), eig$vectors)
  set.seed(5400)
  Xnormlist <- matrix(rnorm(B * n, mu), B, n)
  Xlist <- matrix(, B, n)
  for(i in 1:B){
    Xlist[i, ] <- Xnormlist[i, ] %*% Sigma.sq
    t_stat_list[i] <- (mean(Xlist[i, ]) - mu_0) / (sd(Xlist[i, ]) / sqrt(n))
    p_val_list[i] <- 2 * pt(-abs(t_stat_list[i]), n-1)
  }
  return(p_val_list)
```

```
}
alp <- 0.05
mulist <- seq(0, 2, 0.1)
nlist <- seq(10, 70, 10)
rejprob_mu <- colMeans(sapply(mulist, function(x){pval_func(n = 30, mu = x)}) < alp)
rejprob_size <- colMeans(sapply(nlist, function(x){pval_func(n = x, mu = 1)}) < alp)
par(mfrow = c(1,2))
plot(y=rejprob_mu, x=mulist, type="b",
     xlab="True Mean", ylab="Prob of Rej H0",
     main = "Power Curve: Different Sample Sizes",
     cex.main = 0.75)
plot(y=rejprob_size, x=nlist, type="b",
     xlab="Sample Sizes", ylab="Prob of Rej H0",
     main = "Power Curve: Different True Means",
     cex.main = 0.75)
```



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.

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

```
pval2SampleT <- function(mu1, mu2, var1, var2, n1, n2, alp = 0.05, B){
  set.seed(432)
  #Generating X and Y samples B times
  X <- matrix(rnorm(B * n1, mu1, var1), B, n1)
  Y <- matrix(rnorm(B * n2, mu2, var2), B, n2)
  tstats_3 <- numeric(10000)
  pvals_3 <- numeric(10000)
  for(i in 1:B){
    Sp <- sqrt( ( (n1-1)*var(X[i, ]) + (n2-1)*var(Y[i, ]) ) / (n1 + n2 - 2) )
    tstats_3[i] <- ( mean(X[i,]) - mean(Y[i,]) )/( Sp * sqrt( (1/n1) + (1/n2) ) )
    pvals_3[i] <- 2 * pt(-abs(tstats_3[i]), n1 + n2 - 2)
    #pvals_3[i] <- pt(-abs(tstats_3[i]), n1 + n2 - 2) + 1 - pt(abs(tstats_3[i]), n1 + n2 - 2)
  }
  return(pvals_3)
}
#Calculating Rejection Probablity of H0 (Type I error)
print(mean(pval2SampleT(1, 1, 1.5, 2, 30, 20, 0.05, 10000) < 0.05))
```

```
## [1] 0.0613
```

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?

- 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
pval_pois_func <- function(n = 20, lam = 5){
  B <- 10000
  X_p <- matrix(rpois(B * n, lambda = lam), B, n)
  tstat_pois <- numeric(10000)
  pval_pois <- numeric(10000)
  CI_score <- matrix(, nrow = B, ncol = n)
  X_p_bar <- rowMeans(X_p)
  for(i in 1:B){
    tstat_pois[i] <- (X_p_bar[i] - lam)/(sd(X_p[i,])/sqrt(n))
    pval_pois[i] <- 2 * pt(-abs(tstat_pois[i]), n-1)
  }
  return(pval_pois)
}
B <- 10000
alp <- 0.05
p <- mean(pval_pois_func(20, 5) < alp)
print(p)
```

```
## [1] 0.054
```

```r
#Score Interval for rejection probability p
CI_p <- prop.test(x = (B*p), n = B, correct = FALSE, conf.level = 0.99 )$conf.int
print(CI_p)
```

```
## [1] 0.04846831 0.06012313
## attr(,"conf.level")
## [1] 0.99
```

We can see that $\alpha = 0.05$ is within this interval.