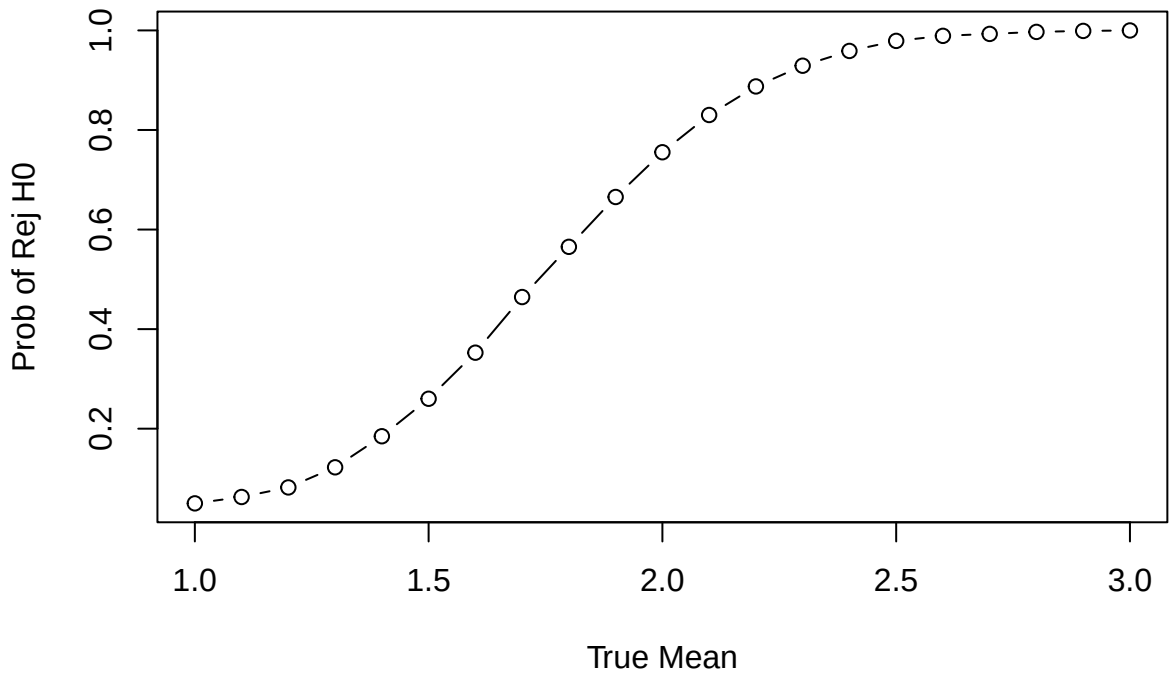
Homework 8 Elias Washor

Elias Washor

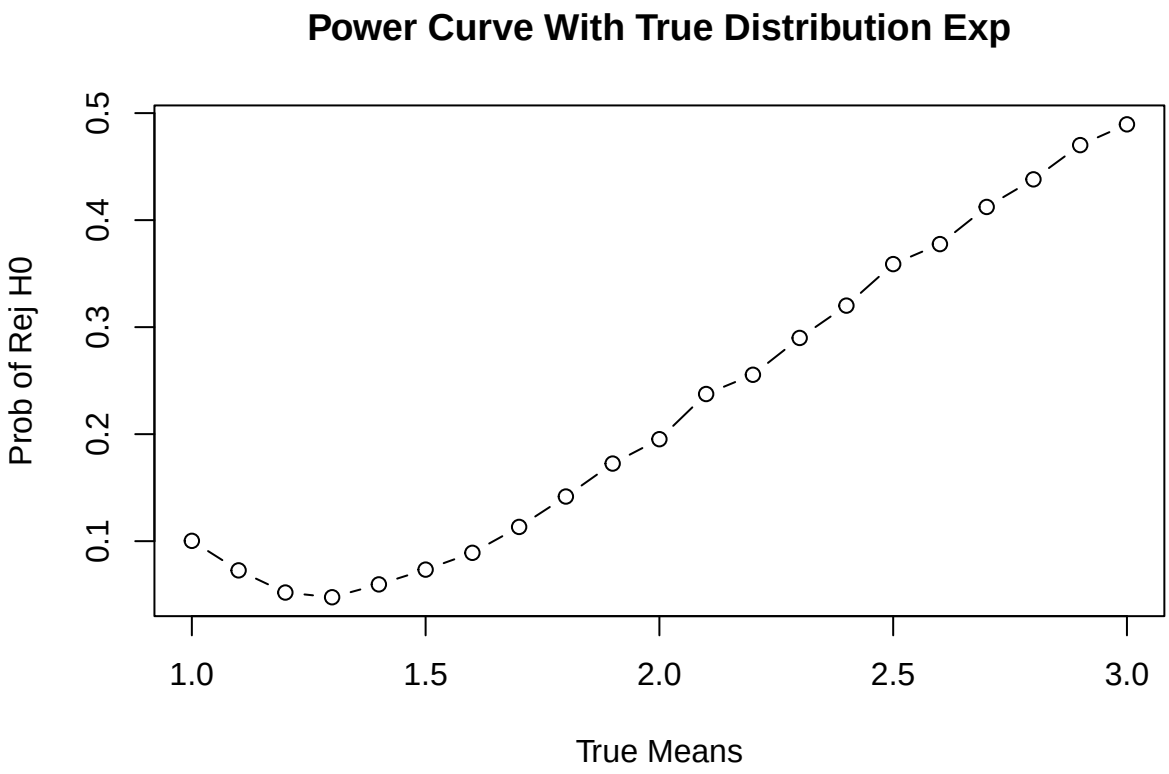
2024-10-24

Q1)

```
rejprobs <- sapply(mulist, function(mu) {mean(pvalDist(30, mu, mu_0, sig) < alp)})
## slide 26
## note pvaldist is defined in Rmd doc
```




```
## slide 34
rejprobs2 <- sapply(mulist, function(mu) {mean(pvalDistExp(10, mu, mu_0) < alp)})
```



Q2)

```r
##Gen dependent normal sample

library(MASS)

set.seed(5400)
r_dep_norm <- function(n, mus = rep(0,n)) {
  cov_m <- matrix(0.1, n, n)
  diag(cov_m) <- 1

  dep_data <- MASS::mvrnorm(1, mus, Sigma = cov_m)
  return(dep_data)
}

## b
test_stat <- function(D) {
  n <- 30
  xbar <- mean(D)
  s <- sd(D)

  test_stat1 <- (xbar - 0) / (s / sqrt(n))

  return (test_stat1)
```

```
}

dat1 <- r_dep_norm(30, mus = rep(0,30))
(ts <- test_stat(dat1))
```

```
## [1] -4.031398
```

```
(pv <- 2* pt(-abs(ts), 29))
```

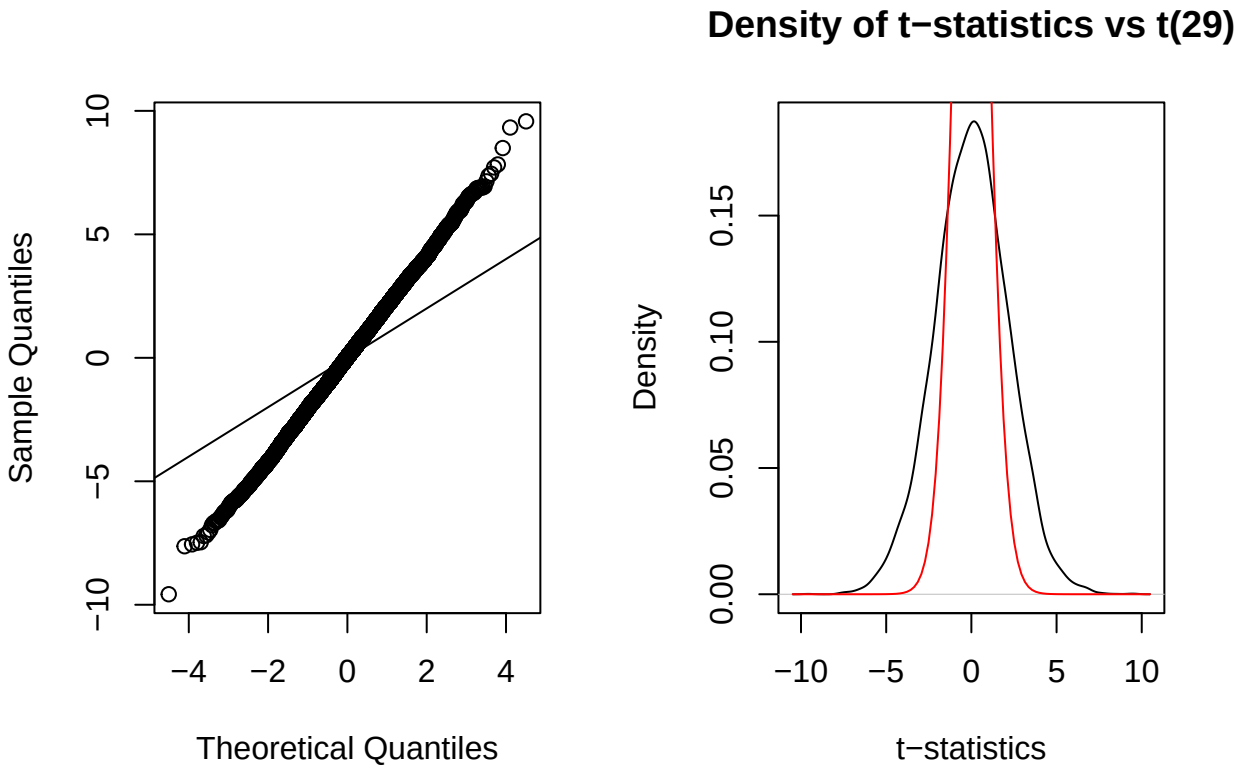
```
## [1] 0.0003673444
```

```
built_in <- t.test(dat1, alternative = "two.sided")
c(built_in$statistic, built_in$p.value)
```

```
##             t
## -4.0313975380  0.0003673444
```

The test statistic t is -4.0313975 and the corresponding p-value is 3.6734441×10^{-4} . Based on the p-value, which is much less than 0.05 (alpha) we reject the null hypothesis that mu = 0 and conclude that there is sufficient evidence supporting that the true mu differs from 0.

```
set.seed(5400)
sim_size <- 10000

data_matrix <- matrix(NA, 30, sim_size)

for (c in 1:ncol(data_matrix)) {
  data_matrix[,c] <- r_dep_norm(30)
}

## part B ###
Tstats <- apply(data_matrix, 2, test_stat)

par(mfrow =(c(1,2)))
quants <- qt(ppoints(sim_size), df = 29)

plot(quants, sort(Tstats), xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",)
abline(0,1)

### part c
density_tstat <- density(Tstats)

plot(density_tstat, main = "Density of t-statistics vs t(29)",
     xlab = "t-statistics", ylab = "Density")

curve(dt(x, df = 29), col = "red", add = TRUE) # Density of t(29)
```



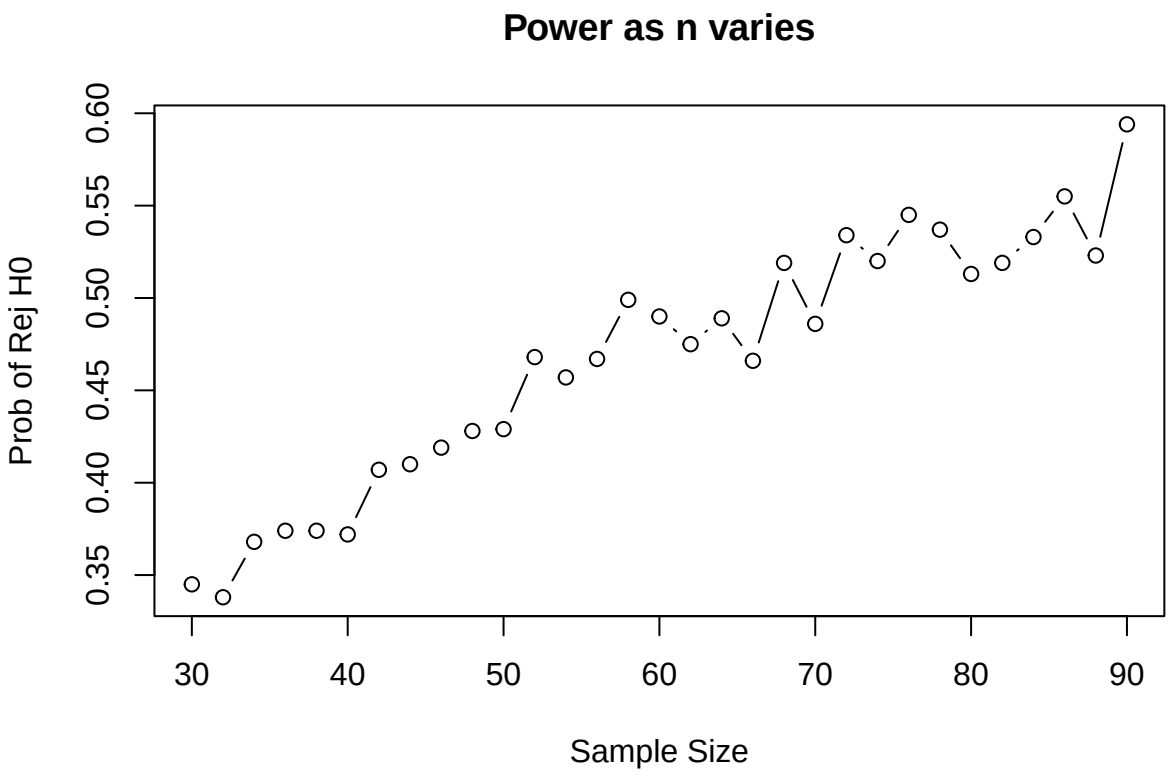

```
### d
get_p <- function(D) {
  n <- 30
  xbar <- mean(D)
  s <- sd(D)

  test_stat1 <- (xbar - 0) / (s / sqrt(n))

  return (2 * pt(-1*abs(test_stat1), n-1))
}

## pvals and type I error
pVals <- apply(data_matrix, 2, get_p)
(type1_err <- mean(pVals < 0.05))
```

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

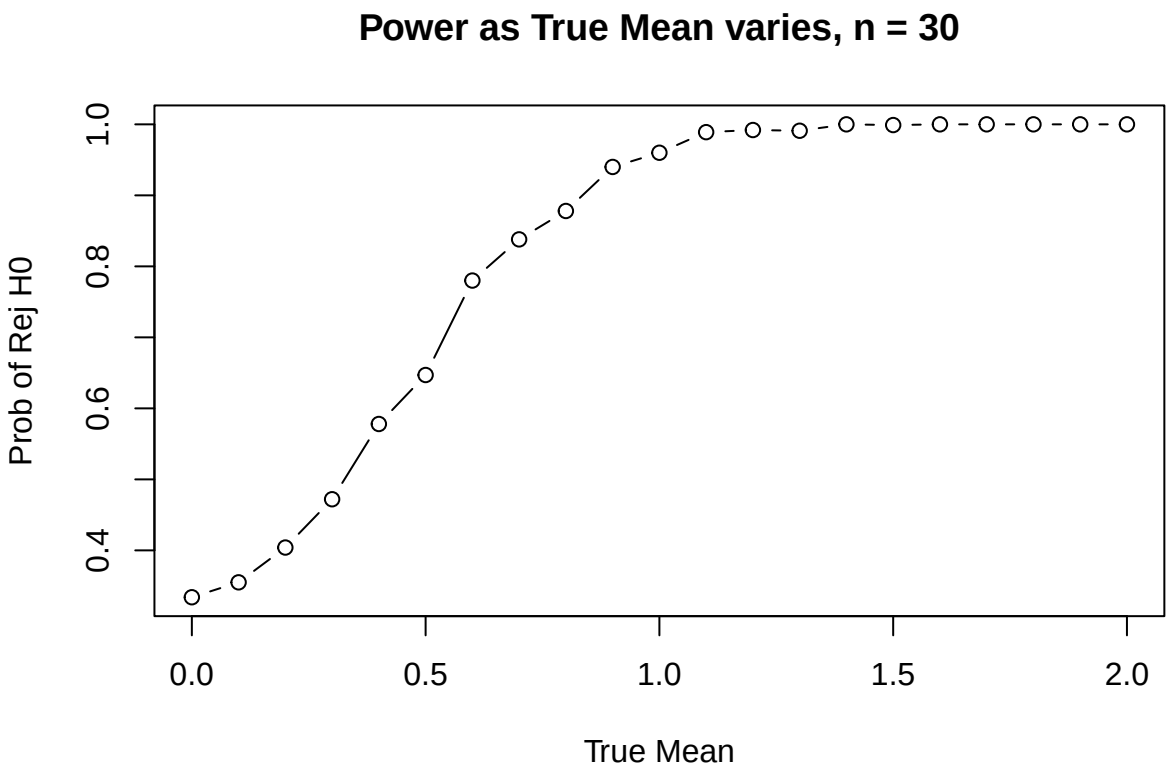
2c) The Q-Q plot shows that the sample does not conform to the t 29 distribution. The sample line is too far off from the theoretical quantiles. Further, comparing the density of the t-statistics and t29 shows that the sample does not conform to t29. We can see that the t29 dist. in red has a much higher density curve in the center and smaller variance than the sample t-statistics.

2d) The estimated Type I error is 0.3364

```r
library(MASS)

r_dep_norm <- function(n, mus = rep(0,n)) {
  cov_m <- matrix(0.1, n, n)
  diag(cov_m) <- 1

  dep_data <- MASS::mvrnorm(1, mus, Sigma = cov_m)
  return(dep_data)
}

pvalDist3 <- function(n, mu, mu0, alp=0.05, B = 1000) {
  data2 <- replicate(B, r_dep_norm(n, rep(mu, n)))
  xbar <- colMeans(data2)
  sdlist <- apply(data2, 2, sd)
  test_stats <- (xbar - mu_0) / (sdlist / sqrt(n))

  return (2 * pt(-abs(test_stats), n-1))
}

# ## diff sample sizes
nlist <- seq(30,90,2)
mu <- 0
alp <- 0.05
mu_0 <- 0

rejprobs <- sapply(nlist, function(n) {mean(pvalDist3(n, mu, mu_0) < alp)})

plot(y=rejprobs, x=nlist, type="b", xlab="Sample Size", ylab="Prob of Rej H0", main="Power as n varies"
```




```
## diff true means
n <- 30
mulist <- seq(0, 2, 0.1)
alp <- 0.05
mu_0 <- 0

rejprobs <- sapply(mulist, function(mu) {mean(pvalDist3(30, mu, mu_0) < alp)})

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



The power increases as the sample size n increases from 30 to 90. Further, when n is fixed at 30 and the true mean varies from 0 to 2, the power increases as the true mean increases. It gets close to 100% power at around the true mean value of 1.0.

Q3)

```
## function for pooled var
pooled <- function(n1,n2, S1, S2) {
  return (sqrt(((n1-1)*S1+(n2-1)*S2)/(n1 + n2 - 2)))
}

pval2SampleT <- function(mu1, mu2, var1, var2, n1, n2, alp = 5e-2, B=1000) {
  Xlist <- matrix(rnorm(B * n1, mu1, sqrt(var1)), n1, B)
  Xbarlist <- colMeans(Xlist)
  xvars <- apply(Xlist, 2, var)

  Ylist <- matrix(rnorm(B * n2, mu2, sqrt(var2)), n2, B)
  Ybarlist <- colMeans(Ylist)
  yvars <- apply(Ylist, 2, var)

  s_pool <- pooled(n1,n2, xvars, yvars)

  t_stats_2 <- (Xbarlist - Ybarlist) / (s_pool * sqrt(1/n1 + 1/n2))
  pvals <- 2 * pt(-abs(t_stats_2), df = n1 + n2 - 2)
```

```r
  return(pvals)
}

example_pvals <- pval2SampleT(mu1 =5, 5, 2, 3, 50,40, B = 10000)


## type I error
mean(example_pvals < 0.05)
```

```
## [1] 0.0577
```

```r
example_pvals2 <- pval2SampleT(15,15, 4, 10, 50,40, B = 10000)
mean(example_pvals2 < 0.05)
```

```
## [1] 0.0611
```

The type I error of the two sample t-test is roughly 0.05, even with different variances.

I tried values of mu1, mu2 = 5 and mu1, mu2 = 15 with variances (2,3) and (4,10). Both gave type I error rates close to 0.05. The latter was 0.06 because of the higher variance.

Q4)

```r
set.seed(5400)
n <- 20
lambda <- 5
alpha <- 0.05
B <- 10000

sample_matrix <- matrix(rpois(B * n, lambda), n, B)

L_means <- colMeans(sample_matrix)
sd_list <- apply(sample_matrix, 2, sd)

t_crit <- qt(1 - alpha / 2, df = n - 1)
T_stats <- (L_means - lambda) / (sd_list / sqrt(n))

rejections <- abs(T_stats) > t_crit

# Type I error rate (proportion of rejections)
p <- mean(rejections)
prop_test_result <- prop.test(sum(rejections), B, conf.level = 0.99)

cat("Type I error rate (p):", p, "\n")
```

```
## Type I error rate (p): 0.0511
```

```r
cat("99% score confidence interval for Type I error:", round(prop_test_result$conf.int,3 ))
```

```
## 99% score confidence interval for Type I error: 0.046 0.057
```

The type I error was 0.0511. The score interval was [0.046, 0.057].

Alpha (0.05) is captured in the 99% score conf int.

```
## 4 part II
# t_crit <- qt(1 - 0.01/2, n-1) # not sure whether to use 0.99 or 0.95

lower_bounds <- L_means - t_crit * (sd_list / sqrt(n))
upper_bounds <- L_means + t_crit * (sd_list / sqrt(n))

# Check if true lambda is covered by the confidence intervals
covered <- (lower_bounds <= lambda & upper_bounds >= lambda)
coverage_prob <- mean(covered)

# CI
coverage_test_result <- prop.test(sum(covered), B, conf.level = 0.99)

cat("coverage probability (p):", coverage_prob, "\n")
```

```
## coverage probability (p): 0.9489
```

```
cat("99% score confidence interval for coverage probability:",
    round(coverage_test_result$conf.int,3))
```

```
## 99% score confidence interval for coverage probability: 0.943 0.954
```

p, coverage probability is 0.949 and the score CI is [0.943, 0.954].

1 - alpha = 0.95, which is captured in the Score CI above.