Homework 7

Max Ryoo

Problem 1

```
set.seed(12181998)
type1 <- NULL
j <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rnorm(i, mean=62.9,sd =13.3))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_z <- sapply(1:10000, function(x) abs(rep_mean[x]-62.9)/(13.3/sqrt(i)))
  rejects <- as.vector(which(rep_z >= 1.96))
  proportion <- length(rejects)/10000
  type1[j] <- proportion
  j <- j+1
}
print(type1)
```

```
## [1] 0.0515 0.0530 0.0499
```

The proportion that the practioner makes a Type1 error is 0.0515 for sample of 7, 0.0530 for a sample of 28, and 0.499 for a sample of 49. This is possible because it is close to the definition of type 1 error. To explain how is I set a NULL vector called type 1 and set a index variable called j equal to 1. I made a for loop to loop through with 7, 28, and 49 as the i variable. Inside the loop I replicated a rnorm of whatever i I was in with the given mean and sd. I then found the mean of all the 10000 samples for each i. I then proceeded to get the z-score for each sample. I looked to see inside the rep_z vector if there were z-scores above or equal to 1.96. I then got the length of that vector (resulting from testing the z-score $>$ 1.96) and divided it by 10,000, which would evidently show the statistics. I would then print the final type1 vector to get [1] 0.0515 0.0530 0.0499.

Problem 2

```
type1.2 <- NULL
j <- 1
critical <- c(2.447, 2.052, 2.011)
c <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rnorm(i, mean=62.9,sd =13.3))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_sd <- sapply(1:10000, function(x) sd(rep_sample[,x]))
```

```
  rep_t <- sapply(1:10000, function(x) abs(rep_mean[x]-62.9)/(rep_sd[x]/sqrt(i)))
  rejects <- as.vector(which(rep_t >= critical[c]))
  proportion <- length(rejects)/10000
  type1.2[j] <- proportion
  j <- j+1
  c <- c+1
}
print(type1.2)
```

```
## [1] 0.0544 0.0487 0.0520
```

In this particular case we would conduct a t test, which uses critical values. For sample of 7 we use n-1 degrees of freedom (6), which critical value is 2.447 For sample of 28 df=27, which has a critical value of 2.052 For n = 49, df=28, critical = 2.011 I used the same loop format as the previous question howver for in this case when I was finding the rep_t values I divided the mean - actual by the sd of the rnorm, which was again divided by the sqrt of i. The new addition I made was the rep_sd vector that has the sample sd for the rnorm function that was replicated 10000 times. I then found which rep_t was higher than the critical value for the n for that case, which I then found the proportion of to be [1] 0.1023 0.1007 0.1017 (stored in type1.2)

Problem 3

```
type1.3 <- NULL
j <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rnorm(i, mean=62.9,sd =13.3))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_sd <- sapply(1:10000, function(x) sd(rep_sample[,x]))
  rep_t <- sapply(1:10000, function(x) abs(rep_mean[x]-62.9)/(rep_sd[x]/sqrt(i)))
  rejects <- as.vector(which(rep_t >= 1.96))
  proportion <- length(rejects)/10000
  type1.3[j] <- proportion
  j <- j+1
}
print(type1.3)
```

```
## [1] 0.0964 0.0593 0.0593
```

I used the same format as number 2, but instead for this case instead of comparing the test with critical values I compared it with 1.96 for the z-score. A combination of problem 1 and 2 could be a good explanation.

Problem 4

```
tableoutput <- rbind(Problem1 = type1, Problem2 = type1.2,Problem3 = type1.3)
colnames(tableoutput) <- c("n=8", "n=28", "n=49")
print(tableoutput)
```

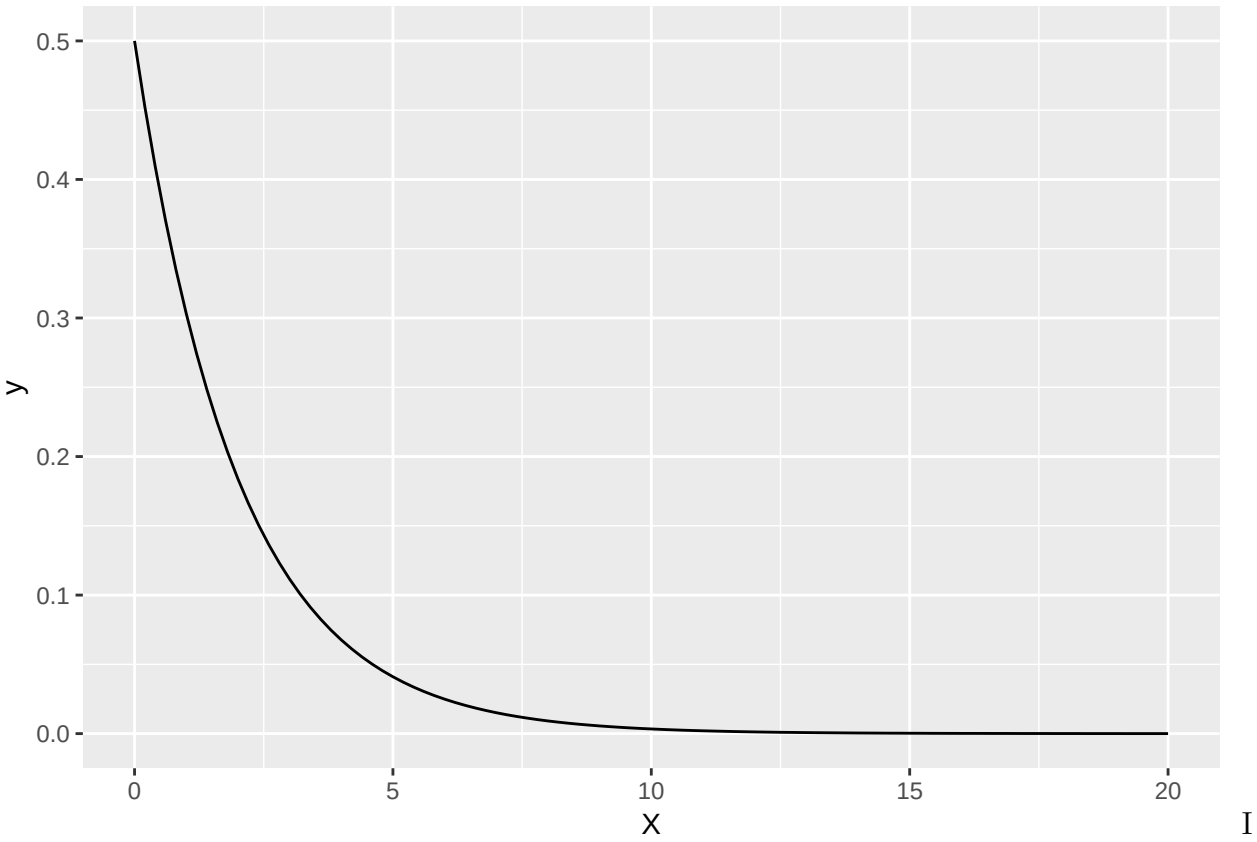
```
##              n=8   n=28   n=49
## Problem1 0.0515 0.0530 0.0499
## Problem2 0.0544 0.0487 0.0520
## Problem3 0.0964 0.0593 0.0593
```

Based on this output we can make many conclusions. We can see that for Problem 1 regardless of the sample size they were relatively close to 0.05, which was our alpha level, which means that it was a good predictor of type 1 error. We can also see that for problem 2 the proportion of making a type 1 error with t test was close to the alpha level of 0.05. However, the interesting thing happens in problem. The alpha level we chose was still 0.05, but the values show 0.0973, 0.0669, and 0.0578 for each sample respectivly. For a sample size of 8 the propbaility was nearly double the alpha value. We can see that this would not be an accurate test to conduct on small samples. The proportion of type1 error does decrease as sample increases, which logically makes sense because as the sample size increases it becomes more normal. However, for smaller sample sizes each point can have a huge factor as shown by the big proportion of type 1 error in the method used in problem 3. If I were to not know the standard deviation I would use a t test for it seems to be close to the actual value.

Problem 5

A

```
library(ggplot2)
Xdata2 <- data.frame(X=c(0,20))
dist2 <- ggplot(Xdata2, aes(x=X)) + stat_function(fun=dchisq, args=list(df=2))
print(dist2)
```



made a density curve of the population distribution with the material provided by the graphics in r file given as class material. The curve is concave up and is decreasing as x increases.

B

```
type1.5b <- NULL
j <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rchisq(i, df=2))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_z <- sapply(1:10000, function(x) abs(rep_mean[x]-2)/(2/sqrt(i)))
  rejects <- as.vector(which(rep_z >= 1.96))
  proportion <- length(rejects)/10000
  type1.5b[j] <- proportion
  j <- j+1
}
print(type1.5b)
```

```
## [1] 0.0444 0.0471 0.0500
```

I used a similar approach to problem 1 since it was asking for the same thing except with a different distriobution. Instead of rnorm I used the rchisq function for each i in the for

loop with a degree of freedom of 2. I then took the mean of that sample and subtracted 2 and divided by 2/sqrt(i) similar to the function in problem with only the 2 changing. I then found which values were greater than 1.96, which i then took the proportion by dividing the length of rejects by 10000. I then printed the vector storing the results, which was [1] 0.0465 0.0475 0.0471.

C

```
type1.5c <- NULL
j <- 1
critical <- c(2.447, 2.052, 2.011)
c <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rchisq(i, df=2))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_sd <- sapply(1:10000, function(x) sd(rep_sample[,x]))
  rep_t <- sapply(1:10000, function(x) abs(rep_mean[x]-2)/(rep_sd[x]/sqrt(i)))
  rejects <- as.vector(which(rep_t >= critical[c]))
  proportion <- length(rejects)/10000
  type1.5c[j] <- proportion
  j <- j+1
  c <- c+1
}
print(type1.5c)
```

```
## [1] 0.1118 0.0749 0.0651
```

I did a similar approach to number 2, but in this case I again changed teh rep_sample vector the a replication of doing rchisq(i, df=2) for all i values with a degree of freedom of 2. I then took the mean and sd of all the replicated trials to which i subtracted 2 from the mean and divided by the sd/sqrt(i), which i compared to the critical value for each sample size of n, which is described in more detail in number two on how I got those values. I then saw which values were greater than the critical value and found the proportion of making a type1 test, which I found was to be [1] 0.1146 0.0763 0.0664.

D

```
type1.5d <- NULL
j <- 1
for(i in c(7,28,49)) {
  rep_sample <- replicate(10000, rchisq(i, df=2))
  rep_mean <- sapply(1:10000, function(x) mean(rep_sample[,x]))
  rep_sd <- sapply(1:10000, function(x) sd(rep_sample[,x]))
  rep_t <- sapply(1:10000, function(x) abs(rep_mean[x]-2)/(rep_sd[x]/sqrt(i)))
```

```
  rejects <- as.vector(which(rep_t >= 1.96))
  proportion <- length(rejects)/10000
  type1.5d[j] <- proportion
  j <- j+1
}
print(type1.5d)
```

```
## [1] 0.1518 0.0874 0.0716
```

Similar to the approach I had in problem three in this case I substituted the rep_sample with replication function of rchisq with degree of freedom 2. I then took the mean and sd of each sample. I then wanted to find the z-score. I subtracted two from the mean and divided by the sd/sqrt(i) , which took values of either 7, 28, 49. I then saw which values were greater than 1.96 then I found the proportion of type 1 error. I stored it into a vector called type1.5d, which had the following result. [1] 0.1590 0.0874 0.0725.

Problem 6

```
tableoutput5 <- rbind(Problem_5b = type1.5b,
                      Problem_5c = type1.5c,
                      Problem_5d = type1.5d)
colnames(tableoutput5) <- c("n=8", "n=28", "n=49")
print(tableoutput5)
```

```
##               n=8   n=28   n=49
## Problem_5b 0.0444 0.0471 0.0500
## Problem_5c 0.1118 0.0749 0.0651
## Problem_5d 0.1518 0.0874 0.0716
```

It is evident that as sample size increases for all the problems it gets closer to the alpha value of 0.05. However, for a sample size that is small the best predictor would be to use the z test if the standard deviation is known and t if the standard deviation is not known. The z test seems to be fairly consistent meanwhile the other apporaches to be heavily unreliable for small sample sizes. If the sample size is big then it wouldn't be a big factor since the idea is that the proportio will get closer to the alpha value of 0.05 since the distribution becomes more normal. In this case if standard deviation is not known I would use t test, but if it is known I would use z test.

Problem 7

```
print(tableoutput)
```

```
##             n=8   n=28   n=49
## Problem1 0.0515 0.0530 0.0499
## Problem2 0.0544 0.0487 0.0520
## Problem3 0.0964 0.0593 0.0593
```

```
print(tableoutput5)
```

```
##               n=8   n=28   n=49
## Problem_5b 0.0444 0.0471 0.0500
## Problem_5c 0.1118 0.0749 0.0651
## Problem_5d 0.1518 0.0874 0.0716
```

We can see that knowing the population standard deviation helps a lot with keeping close to the alpha value of 0.05. We can also see that generally as the sample size increases the proportion of making a type 1 error gets closer to 0.05. We can also see that for normal distributions all the values deviate less than chisqr compared to the value of 0.05.

References

1. https://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm
2. https://www.statisticshowto.datasciencecentral.com/standardized-test-statistic/