Problem Set #8

Econ 103

Part I – Problems from the Textbook

No problems from the textbook on this assignment.

Part II – Additional Problems

1. For this question assume that we have a random sample from a normal distribution with unknown mean but *known* variance.

   (a) Suppose that we have 36 observations, the sample mean is 5, and the population variance is 9. Construct a 95% confidence interval for the population mean.

   > **Solution:** Since the population variance is 9, the population standard deviation is 3. Hence, the desired confidence interval is $5 \pm 2 \times 3/\sqrt{36} = 5 \pm 1 = (4, 6)$

   (b) Repeat the preceding with a population variance of 25 rather than 9.

   > **Solution:** The population standard deviation becomes 5 so the confidence interval becomes $5 \pm 2 \times 5/\sqrt{36} = 5 \pm 10/6 \approx (3.3, 6.7)$

   (c) Repeat the preceding with a sample size of 25 rather than 36.

   > **Solution:** $5 \pm 2 \times 5/\sqrt{25} = 5 \pm 2 = (3, 7)$

   (d) Repeat the preceding but construct a 50% rather than 95% confidence interval.

   > **Solution:** Here we need to use R to get the appropriate quantile:

```r
SE <- 5/sqrt(25)
ME <- qnorm(1 - 0.5/2) * SE
Lower <- 5 - ME
Upper <- 5 + ME
c(Lower, Upper)
## [1] 4.326 5.674
```

(e) Repeat the preceding but construct a 99% rather than a 50% confidence interval.

Solution: Again we use R to get the appropriate quantile:

```r
SE <- 5/sqrt(25)
ME <- qnorm(1 - 0.01/2) * SE
Lower <- 5 - ME
Upper <- 5 + ME
c(Lower, Upper)
## [1] 2.424 7.576
```

2. In this question you will carry out a simulation exercise similar to the one I used to make the plot of twenty confidence intervals from lecture 16.

   (a) Write a function called `my.CI` that calculates a confidence interval for the mean of a normal population when the population standard deviation is known. It should take three arguments: `data` is a vector containing the observed data from which we will calculate the sample mean, `pop.sd` is the population standard deviation, and `alpha` controls the confidence level (e.g. `alpha = 0.1` for a 90% confidence interval). Your function should return a vector whose first element is the lower confidence limit and whose second element is the upper confidence limit. Test out your function on a simple example to make sure it's working properly.

Solution:

```
my.CI <- function(data, pop.sd, alpha){

  x.bar <- mean(data)
  n <- length(data)

  SE <- pop.sd / sqrt(n)
  ME <- qnorm(1 - alpha/2) * SE

  lower <- x.bar - ME
  upper <- x.bar + ME

  out <- c(lower, upper)
  return(out)

}
```

Testing this out on fake data containing twenty-five zeros and assuming a population variance of one, we have

```
fake.data <- rep(0, 25)
my.CI(fake.data, pop.sd = 1, alpha = 0.05)
## [1] -0.392  0.392
```

If we calculated the corresponding interval by hand, assuming a population standard deviation of one, we'd get

$$0 \pm 2 \times 1 \times 1/5 = (-0.4, 0.4)$$

Which is almost exactly the same. The reason for the slight discrepancy is that when working by hand we use the approximation `qnorm(0.975)`≈ 2 whereas the exact value, which R provides, is more like 1.96.

(b) Write a function called `CI.sim` that takes a single argument `sample.size`. Your function should carry out the following steps. First generate `sample.size` draws from a standard normal distribution. Second, pass your sample of standard normals to `my.CI` with `alpha` set to 0.05 and `pop.sd` set to 1. Third, return the resulting confidence interval. Test your function on a sample of size 10. (What we're doing here is constructing a 95% confidence interval for the mean of a normal population using simulated data. The population mean is in fact zero, but we want to see how our confidence interval procedure works. To do this we "pretend" that we don't know the population mean and only know the population variance. Think about

this carefully and make sure you understand the intuition.)

Solution:

```
CI.sim <- function(sample.size){
    sims <- rnorm(sample.size)
    CI <- my.CI(sims, pop.sd = 1, alpha = 0.05)
    return(CI)
}
CI.sim(10)
## [1] 0.01118 1.25077
```

(c) Use `replicate` to construct 10000 confidence intervals based on simulated data using the function `CI.sim` with `sample.size` equal to 10. (Note that `replicate` will, in this case, return a matrix with *2* rows and *10000* columns. Each column corresponds to one of the simulated confidence intervals. The first row contains the lower confidence limit while the second row contains the upper confidence limit.) Calculate the proportion of the resulting confidence intervals contain the true population mean. Did you get the answer you were expecting?

Solution:

```
simCIs <- replicate(10000, CI.sim(10))
simCIs[,1:5]
##          [,1]    [,2]    [,3]    [,4]     [,5]
## [1,] -1.17035 -1.0628 -0.7007 -0.2710 -1.22072
## [2,]  0.06924  0.1768  0.5389  0.9686  0.01887
lower <- simCIs[1,]
upper <- simCIs[2,]
covers.truth <- (lower < 0) & (upper > 0)
sum(covers.truth)/length(covers.truth)
## [1] 0.9513
```

The answer is pretty much dead on: almost exactly 95% of the intervals contain the true population mean (zero).

(d) Repeat the preceding but rather than using `CI.sim` write a new function called `CI.sim2`. This new function should be identical to `CI.sim` except that, when calling `my.CI`, it sets `pop.sd = 1/2` rather than 1. How do your results change? Try to provide some intuition for any differences you find.

Solution:

```
CI.sim2 <- function(sample.size){
    sims <- rnorm(sample.size)
    CI <- my.CI(sims, pop.sd = 1/2, alpha = 0.05)
    return(CI)
}
simCIs <- replicate(10000, CI.sim2(10))
simCIs[,1:5]
##         [,1]     [,2]    [,3]     [,4]    [,5]
## [1,] -1.141 -0.52405 -0.1795 -0.06908 -0.4866
## [2,] -0.521  0.09575  0.4403  0.55071  0.1331
lower <- simCIs[1,]
upper <- simCIs[2,]
covers.truth <- (lower < 0) & (upper > 0)
sum(covers.truth)/length(covers.truth)
## [1] 0.6761
```

In this case the procedure didn't work: many fewer than 95% of the intervals contain the true population mean. The problem is that `CI.sim2` constructs a confidence interval using the *wrong* population standard deviation! Since it uses 1/2 rather than 1, the resulting intervals are too short, so too few of them contain the true population mean.