

ENGR 113: Lab 4

Confidence Intervals

Objectives:

1. To explore the Central Limit Theorem
2. To learn about normal probability plots
3. To explore confidence intervals

Note that there is a text file of all the R commands on the class web page, so that if you wish, you can cut and paste those in instead of retyping them.

The Central Limit Theorem

Recall the Central Limit Theorem:

If X_1, \dots, X_n are independent identically distributed random variables with mean μ and variance σ^2 , then $\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} N(0, 1)$

If the X_i 's are normal random variables, the sample average has a normal distribution exactly, with mean μ and variance $\frac{\sigma^2}{n}$. Let's do a simulation and see if this holds true. We'll generate a vector of 100 observations from a $N(5,4)$, calculate the mean, and repeat 500 times. Enter the following commands (on separate lines, as shown below, but don't type in the ">" or "+" prompts):

```
> xbar <- c(rep(0,500))
> for(i in 1:500) {
+ x <- rnorm(100,5,2)
+ xbar[i] <- mean(x)
+ }
```

The first line initializes `xbar` to be a vector of 500 0's. Before we can use elements of a vector (like in the fourth line), we have to initialize the vector (this is like declaring a variable in C). The next line sets up a loop with index `i`. Notice how the prompt changes to a `+` inside the loop, as R is waiting for the closing brace of the loop before it does any computations. The third line generates a vector of 100 observations from a normal distribution with mean 5 and standard deviation 2 (variance 4), and then the fourth line computes the mean of the sample and stores it in `xbar`.

Now we'll take a look at `xbar`. Are its mean and variance close to what you expect? (What is the theoretical mean and variance for the average of 100 observations?) Is its shape normal? Enter the following and see:

```
> mean(xbar)
> var(xbar)
> hist(xbar)
> boxplot(xbar)
```

To get the standard deviation, you would use `sqrt(var(xbar))`. In class, we also talked a little about normal probability plots, where we plot the observed quantiles (the sorted data points) on the y-axis, and the theoretical quantiles (from the CDF for the normal distribution) on the x-axis and see if the data fall on a straight line. To do this in R, we use the `qqnorm` command:

```
> qqnorm(xbar)
```

You should get something pretty close to a straight line. Note that since `xbar` is not a standard normal (mean 0, standard deviation 1), the line has an intercept different from zero and a slope different from one, but it is still a straight line.

The power of the Central Limit Theorem is that it holds for any distribution with a finite variance. Let's try the exponential distribution (recall that its density is $f(x|\lambda) = \lambda e^{-\lambda x}$ for $x > 0$, with mean $1/\lambda$). Here we do an example with $\lambda = 1/2$, i.e., mean 2 and variance 4:

```
> for(i in 1:500) {  
+   x <- rexp(100, .5)  
+   xbar[i] <- mean(x)  
+ }
```

To remind yourself of the shape of the exponential distribution, you can make a histogram of `x`. But notice how `xbar` looks normal, just as the Central Limit Theorem says:

```
> hist(x)  
> hist(xbar)  
> mean(xbar)  
> var(xbar)  
> qqnorm(xbar)
```

The Central Limit Theorem still holds for discrete distributions, even something as simple as a Bernoulli (a binomial with $n=1$):

```
> for(i in 1:500) {  
+   x <- rbinom(100,1, .5)  
+   xbar[i] <- mean(x)  
+ }  
> hist(x)  
> hist(xbar)  
> mean(xbar)  
> var(xbar)  
> qqnorm(xbar)
```

For a binomial with $n = 1$ and $p = \frac{1}{2}$, what is its expected value? Its variance? For a sample of 100, what are the expected value and variance of the sample average? Do your empirical (observed) results match the theoretical results? Finally, note the interesting behavior in the normal probability plot. It should look sort of like a set of steps. Why does it look this way?

Confidence Intervals

Now we will do some simulations to try to get a better understanding of confidence intervals. Let's spend a little more time beating the tea-bag problem to death (because we've already calculated intervals in class, so we won't need to re-calculate them, you can just reference your class notes). Recall that we assumed the true process filled bags with an amount of tea normally distributed with mean 5.501g and standard deviation 0.106g. Suppose we generate 50 observations from this distribution and then calculate a confidence interval. There will be a 95% probability that this interval will include the true mean of 5.501, as we think about it before we have made this interval.

```
> x <- rnorm(50, 5.501, .106)
> xbar <- mean(x)
> ci.min <- xbar - 1.96 * .106 / sqrt(50)
> ci.max <- xbar + 1.96 * .106 / sqrt(50)
> print(c(ci.min, ci.max))
```

Now that you have your interval, what is the probability that it contains the true value of 5.501? It is 1 if it does, 0 if it doesn't. You know the probability isn't .95, because you can see whether or not the interval includes the true value. But it's not just because you can observe it, but it is a fundamental assumption that the right answer is fixed, so it is either in the interval, or it isn't. Even if you don't know what the true population mean is, once you have observed (calculated) your interval, nothing is random, so the true mean is either in it or not, even though you don't know which.

Now we will repeat the above process 1000 times, and see how often our confidence interval does include the true mean (since we know what it is). Since we are making 95% intervals, we expect about 950 of them to contain the true mean.

```
> mean.in.ci <- 0
> for(i in 1:1000) {
+   x <- rnorm(50, 5.501, .106)
+   xbar <- mean(x)
+   ci.min <- xbar - 1.96 * .106 / sqrt(50)
+   ci.max <- xbar + 1.96 * .106 / sqrt(50)
+   if(ci.min < 5.501 && ci.max > 5.501) mean.in.ci <- mean.in.ci + 1
+ }
> print(mean.in.ci)
```

What this set of commands does is, first it initializes a counter, `mean.in.ci` that will count how many times out of 1000 that the 95% CI actually covers the true mean. We then generate 50 random observations and compute the lower and upper endpoints of a 95% confidence interval. Finally we see if the true mean is in this interval, and if so, we augment the counter by one.

So you should have roughly 950 of your confidence intervals including the true mean. If you repeat this a couple of times, you will see that the number varies, but should always be somewhere close to 950.

What if we didn't know the true standard deviation? Then we would have to estimate σ_X with $s = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n-1}}$. Because we have to estimate an additional parameter, we have additional uncertainty, and this is reflected by using a t distribution instead of a normal for creating our confidence interval. In this next simulation, we will make both a normal confidence interval and t confidence interval, and see that the t does contain the right answer about 95% of the time, while the normal one tends to do so less than 95% of the time. To increase the chance of the simulation working out the way it is supposed to, we will now use 10,000 simulations, which means you may have to wait briefly as it runs:

```
> mean.in.normci <- 0
> mean.in.tci <- 0
> for(i in 1:10000) {
+   x <- rnorm(50, 5.501, .106)
+   xbar <- mean(x)
+   s <- sqrt(var(x))
+   ci.norm.min <- xbar - 1.96 * s / sqrt(50)
+   ci.norm.max <- xbar + 1.96 * s / sqrt(50)
+   ci.t.min <- xbar - 2.0096 * s / sqrt(50)
+   ci.t.max <- xbar + 2.0096 * s / sqrt(50)
+   if(ci.norm.min < 5.501 && ci.norm.max > 5.501)
+     mean.in.normci <- mean.in.normci + 1
+   if(ci.t.min < 5.501 && ci.t.max > 5.501) mean.in.tci <- mean.in.tci + 1
+ }
> print(c(mean.in.normci, mean.in.tci))
```

Note that 2.0096 is the t value for 49 degrees of freedom and 2.5% in the upper tail, i.e., $qt(.975, 49) = 2.0096$. So your simulation should give you a number of normal CI's that is noticeably smaller than 9500, while the number of t CI's should be close to 9500, so you can see that when you have to estimate the standard deviation, you need to use the t distribution to get accurate results.