Summary of Lab Week 5

Loops

Sometimes we want to repeat essentially the same command several times. We do this using what programmers call a loop. R does loops with the `for()` command. Try this:

```
> for(i in 1:5) print(i)
```

This command tells R to repeatedly execute `print(i)` (which simply displays the value of `i`), while `i` takes all the values in `1:5`, one at a time.

You can use any vector you want to define the values of `i`:

```
> X = c(..., ..., ...)
> for(i in X) print(i)
```

Of course you can loop other commands besides `print()`.

```
> for(i in 1:5) X[i] = i^2 + 5*i - 7
> X
```

Try a few variations on this.

If you want to execute multiple commands each time, use braces.

```
> for(i in 1:5) {
> j = i^2
> print(j)
> }
```

Binary variables

A binary variable is a random variable that can take only two possible values. Usually the two values are called 0 and 1, or sometimes `TRUE` and `FALSE`. One easy way to generate binary random variables is to use Uniform random variables.

```
> runif(1) < .5
```

This command first samples a number from a Standard Uniform distribution. If this number is less than .5 it outputs `TRUE` and otherwise it outputs `FALSE`. Therefore the result is `TRUE` half the time and `FALSE` half the time.

We can get a sample of many binary variables by starting with many Uniform variables.

```
> X = (runif(n) < .5)
> X
```

We can also sample from binary distributions with other values of *q* (the probability of `TRUE`). For example, the command

```
> runif(1) < .7
```

outputs `TRUE` 70% of the time, because 70% of the Standard Uniform distribution is below .7 (recall that all numbers between 0 and 1 are equally likely in the Standard Uniform distribution.)

Try creating samples from binary distributions with different values of *q*.

```
> X = (runif(n) < q)
```

Binomial distribution

The most interesting statistic (really, the only statistic) we can compute from a binary sample is the frequency of `TRUE`s in the sample. We can find this statistic by simply summing the sample, since `sum()` treats `TRUE` as `1` and `FALSE` as `0`.

```
> sum(X)
```

The frequency or sum from a sample of binary variables is called a binomial random variable. Its probability distribution is called a binomial distribution. This distribution specifies the probability of

getting any number of TRUEs out of a sample of size n. For example, it tells you the probability of getting any number of heads from flipping a coin n times (if you use q=.5).

Get a large number of samples, and compute the sum of each sample.

```
> for(i in 1:10000) {
> X = (runif(n) < q)
> s[i] = sum(X)
> }
```

Notice that the above commands give an error. R gets confused if you refer to components of a vector that doesn't exist yet. So, first s needs to be created. It doesn't matter what you define it to be. After that, the loop above will work.

```
> s = 37
```

s is now a large collection of binomial variables, each one the frequency of TRUEs in a different binary sample. Therefore the distribution of values in s approximates an exact binomial distribution. Make histograms of binomial distributions for a few different choices of n and q. Notice how the distribution becomes more Normal as n gets larger. This is basically because of the Central Limit Theorem, which we'll work on next week.

Binomial Test

Imagine you have observed a set of n binary variables, S of which were TRUE. Choose a number for n between 10 and 20, and choose a value for S that's less than n and greater than n/2.

```
> n = _
> S = _
```

(Notice that I'm using capital S for our actual result, and below I continue to use lowercase s for the distribution of hypothetical results. Whatever names you use, make sure to keep them separate.)

Now imagine you want to use your sample result, S, to test whether the proportion of TRUEs in the population (q) is reliably greater than .5. For this test, we define the null hypothesis as the hypothesis that $q = .5$ in the population, and the alternative hypothesis as the hypothesis that $q > .5$. The test will decide between these two hypotheses by asking whether the data (i.e., the value of S) is consistent with what you'd expect according to the null hypothesis.

First, let's estimate the probability distribution of s according to the null hypothesis. Do this by simulating a large number of samples from an artificial population where the null hypothesis is correct (i.e., using $q = .5$). Each simulated sample should have the same size (n) as the "real" sample you defined above. Then get the frequency of TRUEs in each simulated sample, just like you did above.

Make a frequency table of your hypothetical sums

```
> f = summary(factor(s))
> f
```

(Notice that the concept of frequency is being used at two distinct levels here. First, you found the frequency of TRUE in each hypothetical sample. Now you're looking at how many samples had each possible frequency, i.e. a "frequency of frequencies". This two-layer reasoning is common in inferential statistics: First we define a statistic as a function of all the observations in a single sample, and then we talk about the distribution of the statistic itself across lots of different samples.)

Convert the frequencies to proportions by dividing by the number of samples you took. Then make a plot of the distribution.

```
> p = f/length(s)
> p
> barplot(p)
```

This distribution is an estimate of the probability distribution of the sum of TRUEs according to the

null hypothesis. This is called the null hypothesis' likelihood function.

Notice that values of `s` near `n/2` have high probability according to the null hypothesis, but larger values of `s` are less likely. Therefore, if `S` is big enough, we can conclude the null hypothesis is a poor explanation for the data, and we can reject it. All we need to do is decide how large `S` can be before we reject the null hypothesis. This cutoff is called the critical value for `S`. We define the critical value so that IF the null hypothesis were true, `S` would have a 5% chance of being above the critical value. That is, we have a 5% chance of making an error, called a Type I error (rejecting the null hypothesis when it's actually true).

First, find the appropriate critical value by trial and error. Make a guess for what it might be, and calculate the fraction of entries in `s` that are greater than your guess. Make adjustments until you find the smallest critical value that gives a Type I error rate less than or equal to 5%. (Notice you can't achieve exactly 5% error, because your choice of cutoffs is discrete. You want to choose the option that gives slightly less than 5%, rather than slightly more, because the general strategy is to strictly limit the rate of Type I errors.)

Now that you've found the critical value, compare your actual result, `S`, to the critical value. Do you reject the null hypothesis, and conclude the rate of `TRUE`s in the population is greater than 50%, or do you stick with the null hypothesis? (If `S` is exactly equal to the critical value, you retain the null hypothesis. That's because you just defined the critical value based on the fraction of results that are strictly greater than it.)

Notice that, now that you have the critical value, you can easily redo your hypothesis test for any other possible value of `S`. Imagine you got a different result, and decide what conclusion that result would lead to.

Exact Binomial Test

Instead of estimating the likelihood function by simulating lots of samples, we can compute it directly, using the `pbinom()` function. This function is a lot like `pnorm()` -- it gives the probability that the sum of `TRUE`s from a binary sample will be greater than whatever value you input.

`pbinom()` needs 3 inputs: the critical value (`crit`), the sample size (`n`), and the success probability (`q`). The fourth input tells `pbinom()` that we want the probability of a score `>crit` (i.e., the probability in the upper tail of the binomial distribution), rather than the probability `≤crit` (i.e., the lower tail).

```
> pbinom(crit,n,.5,lower.tail=FALSE)
```

If you found the right critical value above, you should get an answer slightly less than 5%, and if you replace `crit` with `crit-1`, you'll get an answer slightly above 5% (because `crit-1` is not a stringent enough criterion). Notice that the probabilities you get from `pbinom()` are slightly different than what you got above using `s`, because `pbinom()` uses the exact math of the binomial distribution, whereas above you were just using simulations to approximate the distribution.

Finally, you can use the `qbinom()` function to directly find the critical value. `qbinom()` is the opposite of `pbinom()`. `pbinom()` gives the probability above any critical value, and `qbinom()` gives the critical value for any probability. (It's called `qbinom()` because it basically is computing quantiles. You should be able to understand why the critical value is the 95th percentile in the binomial distribution, since 1 – 5% = 95%.)

If you want the value of the binomial variable that has a probability of 5% of being exceeded, you can do this:

```
> qbinom(5%,n,.5,lower.tail=FALSE)
```

This should give the same answer for the critical value as you found above.