Psyc 2111, Jones

Summary of Lab Week 4

<u>Random numbers</u>

R has built-in functions that will generate random numbers. You can think of these numbers as samples from a mathematically ideal population. For example, the `rnorm()` function samples from a perfect Standard Normal distribution. (The Standard Normal is the Normal distribution that has mu = 0 and sigma = 1. It's what you get if you take any other Normal distribution and standardize it, i.e. convert to z-scores.)

Try sampling a single Normal random number. Do this several times and notice how the answer is always different.

```
> rnorm(1)
```

The input to `rnorm()` tells it how many random numbers to generate. If you want more random numbers at once, change the input value. Let's get a big vector of random numbers to play with:

```
> normvec = rnorm(100000)
```

Compute the mean and standard deviation of your sample. They should be very close to 0 and 1 (respectively), because those are the parameters of the population. Your statistics won't exactly match the true values of the parameters, because of sampling error.

Draw a histogram of the data with a lot of bins, e.g.

```
> hist(normvec, breaks = 200)
```

Notice how closely the sample matches the population (Normal) distribution. This is because the sample size (n) is so large.

Since the Standard Normal is the distribution of z-scores for any other Normal distribution, we can sample from other Normal distributions by convert backwards from Z to X. For example, IQ has a mean of 100 and a standard deviation of 15, meaning Z = (X-100)/15, so X = 100 + Z*15.

```
> IQZ = rnorm(_)
> IQ = 100 + IQZ*15
```

Make a histogram of your IQ sample to make sure it came out right. Then generate numbers from other Normal distributions, by using other values of mu and sigma.

<u>Uniform distribution</u>

Another common type of population distribution is called a *uniform distribution*. In a uniform distribution, all values are equally likely within a given range. The Standard Uniform distribution uses a range of 0 to 1. (The term Standard is a bit misleading here, because the mean and standard deviation are not 0 and 1; in fact you should be able to guess that the mean is 1/2.)

We can sample random numbers from the Standard Uniform using the runif() function.

```
> unifvec = runif(100000)
> hist(unifvec,breaks = 200)
```

Notice how the scores are all between 0 and 1, and the distribution is totally flat.

What are the minimum, maximum, median, and quartiles of the Standard Uniform distribution? You should be able to figure these out, before getting estimates from your sample.

```
> summary(unifvec)
```

(Note on *scientific notation*: Sometimes R gives you numbers like `5.0127e-01`. Technically, this means `5.0127 * 10^(-1)`, which is the same as `5.0127 * .1`, or `.50127`. In practice, the number after the `e` tells you how many places to shift the decimal point. A negative number means shift left. A positive number means shift right; e.g. `5.0127e+02 = 501.27`.)

You can sample from other uniform distributions using linear transformations just like with the Normal.

```
> otherUnif = runif(_)*_ + _
> hist(otherUnif, breaks = _)
```

Cumulative distributions and probability

What's the probability that a score from a Standard Uniform distribution will be less than .27? You should be able to figure this out just by thinking about it. You can estimate this property of the population by looking at your sample, and finding what proportion of numbers in the sample are less than .27.

```
> mean(unifvec<.27)
```

This command illustrates how you can use `mean()` to compute probabilities. When a logical (`TRUE/FALSE`) vector is input to `mean()`, `mean()` returns the fraction of entries that are `TRUE`. This is an important use of the `mean()` function that is different from how we've used it before. (The connection between the two uses is that `TRUE` is treated as `1` and `FALSE` as `0`; thus the mean is the fraction of `1`s, or `TRUE`s.)

It should be obvious by now that the cumulative distribution for the Standard Uniform is a very simple function: F(x) = x. Plot the cumulative distribution of the sample to verify this.

```
> P = ecdf(unifvec)
> plot(P)
```

(Notice that the cumulative distributions we're getting here are cumulative probabilities, not cumulative frequencies. That is, `P(x)` is the fraction of scores less than or equal to `x`, not just the count of scores less than or equal to `x`.)

Now that we're warmed up on the Uniform, let's work through the cumulative distribution of the Normal. Start by plotting the cumulative distribution function of your Normal sample, like you just did for the uniform sample.

Unlike with the Uniform, the cumulative distribution for the Normal never quite reaches 0 (for negative scores) or 1 (for positive scores). That's because under the Normal distribution, there's a non-zero probability of observing an arbitrarily extreme score in either direction. Technically, we say that the Normal is *unbounded*. However, you can also see that extreme scores are very unlikely, because `P` gets very close to 0 and 1 once `z` gets past +/-3 or so.

Use the Normal sample to estimate the probability of a Normal z-score beyond various values. Try +/-2, 3, 4, etc.

```
> mean(normvec<_)
> mean(normvec>_)
```

Recall that `pnorm()` gives the exact cumulative Normal distribution, and that `1-pnorm()` gives the complementary probability (i.e., of being above a certain value). Use this function to compare to the answers you got above from the sample.

Probabilities of scores within a range

What is the probability of a z-score between 1 and 2? It may be easier to think about this using the sample. If we start by counting all the scores less than 2, and then remove all the scores less than 1, we're left with scores between 1 and 2.

```
> sum(normvec<2) - sum(normvec<1)
```

That's counting, but the same thing works with proportions. You can find the proportion of scores between 1 and 2 by finding the proportion that are less than 2 and removing (subtracting) the proportion that are less than 1.

```
> mean(normvec<2) - mean(normvec<1)
```

What we just did with proportions in the sample also works with probabilities in the population. The probability of a score between 1 and 2 equal the probability of a score less than 2 minus the probability of a score less than 1.

```
> pnorm(2) - pnorm(1)
```

The proportion you got from the sample should be very close (but not equal) to the probability you got from the population. Try a few more ranges, in each case comparing sample proportion to population probability.

<u>Rounding</u>

The `round()` function does just what you think it does. If you give it a single input, it rounds it to the nearest integer. Try a few inputs, like 1.1, 1.6, and also negative numbers.

```
> round(_)
```

If the input is a vector, all the entries get rounded.

```
> X = c(_,_,_,_)
> round(X)
```

If `round()` is given a second input, that input determines how many digits to round to. Inputting 1 rounds to the nearest tenth (.1), inputting 2 rounds to the nearest hundredth (.01), and inputting a negative number like -1 rounds to the nearest ten (10).

```
> round(_, _)
```

Notice that `round(x,0)` is the same as `round(x)`.