Summary of Lab Week 3

Descriptive statistics

Mean, median, variance, and standard deviation all have built-in functions:

```
> mean(X)
> median(X)
> var(X)
> sd(X)
```

You can also get standard deviation using `var(X)` and the `sqrt()` function, which does square root

```
> sqrt(var(X))
```

Computing statistics by hand

Even though there are built-in short-cut functions for most important statistics, a good way to learn and understand the formula for a statistic is to compute it "by hand" without using the short-cut function.

Let's do this with the mean. The mean equals the sum of `X` divided by `n`. Since `n` is the number of observations, it's equal to `length(X)`.

```
> meanX = sum(X)/length(X)
```

Verify we get the same answer as by using the built-in function

```
> meanX
> mean(X)
```

Now let's compute the median by hand. Sort the data:

```
> X_sorted = sort(X)
```

Pick out the middle item:

```
> midpoint = (length(X)+1)/2
> medianX = X_sorted[midpoint]
```

Check against the built-in function:

```
> medianX
> median(X)
```

If your sample has an even number of scores, the above calculations won't work, because `midpoint` is not an integer. In other words, your sample doesn't have an exact middle entry. You can approximate the median by looking at the two middlemost entries:

```
> medianX = X_sorted[midpoint-.5]
> medianX = X_sorted[midpoint+.5]
```

Notice that the `median()` function gives the average of these two scores. This is a form of interpolation.

Normal Distributions

Here's a custom function that plots Normal distributions:

```
normal = function(mu,sigma){
 z = (-4000:4000)/1000
 x = mu + sigma*z
 Density = 1/(sigma*sqrt(2*pi)) * exp(-z^2/2)
 plot(x,Density,type="l")
}
```

If you're interested, the important line is the one that defines `Density`. This is the mathematical equation that defines a Normal distribution.

Go to this address and paste the code for the `normal()` function into the R window.
http://matt.colorado.edu/teaching/stats/fall11/labs/lab3-1.txt

Here's how to use the `normal()` function. `mu` and `sigma` are numbers you supply or variables you define.

```
> normal(mu, sigma)
```

Start by plotting the Standard Normal, which means with a mean of zero and standard deviation of one (this is the standardized distribution, or distribution of z-scores, for any other Normal distribution).

```
> normal(0,1)
```

Plot a few more Normal distributions using other values of `mu` and `sigma`. Notice how the values you use affect the position and size of the distribution, although the shape always stays the same.

z-scores

To find a z-score for any value `x`, all you need are the mean and standard deviation. You should know how to compute these statistics already.

```
> x = ___
> z = (x-mu)/sigma
```

Compare your choice of `x` to the distribution of `X` to verify that the result is a sensible z-score. Is `x` above or below the mean? Is it close to the center or far away?

If you have a z-score, you can convert back to the raw score. Recall that `z` tells you how many standard deviations the raw score is above (or below) the mean. So, to get the raw score, we start at the mean and add `z` standard deviations.

```
> z = ___
> x = mu + z*sigma
```

Again, compare your result to the distribution of `X` to verify it is sensible based on the z-score you started with.

We can compute all the z-scores at once using vector arithmetic:

```
> Z = (X-mu)/sigma
> Z
```

`Z` is now the standardized distribution.

Plot the cumulative distribution functions for both the raw scores and the standardized scores, and notice how they are the same and how they are different. Before creating these plots, enter the following command, which tells R that you want to look at multiple plots at once (in this case, in a 1×2 arrangement).

```
> par(mfrow=c(1,2))
```

Probabilities of z-scores

A useful function for inferential statistics is `pnorm()`, which is the cumulative distribution function for the standard Normal. Therefore `pnorm(z)` gives the probability that an observation will have a z-score less than or equal to `z`, assuming that the distribution is Normal. Try a few values, including `0`, positive numbers, negative numbers, large numbers, and numbers close to `0`. Compare your results to the plot of the standardized Normal from above (you will have to redraw it).

```
> pnorm(___)
```

If you want the probability of a z-score greater than or equal to `z`, just subtract from `1`:

```
> 1 - pnorm(___)
```

To see what percentile a score is, multiply by `100`:

```
> pnorm(___) * 100
```

The inverse of `pnorm()` is `qnorm()`. `qnorm()` tells you what z-score is at a particular quantile. For example, the z-score for the first quartile is

```
> qnorm(.25)
```

Try finding the z-scores for the median and the 62nd percentile. Think about what the result should be before you do each calculation. Also notice that you can't give `qnorm()` inputs less than `0` or greater than `1` (try it), and test what happens if you input `0` or `1`.

Notice that the probability of being less than any quantile is the same as the number that defined that quantile.

```
> pnorm(qnorm(___))
```

Also, the quantile for any probability is the same as the original z-score.

```
> qnorm(pnorm(___))
```

Assignment

Imagine you are conducting a memory experiment, in which each subject is asked to recall 20 items. The possible scores are thus `0:20`. Start by generating fake data for 200 subjects using this command:

```
> X = round(runif(200)*21 - .5)
```

The `runif()` function creates random numbers that are uniformly distributed between 0 and 1. The rest of this command converts those numbers into random integers from 0 to 20.

1. Calculate the mean of your distribution. Do this by hand, meaning without using the `mean()` function.

2. Calculate the standard deviation of your distribution, again by hand. You can use your result from Question 1 as part of this calculation.*

3. Pick a value `x` in your distribution, between 13 and 18, and find which percentile it's at (there are several ways to do this, from last week).

4. Find the z-score for `x`.

5. Use the `pnorm()` function to predict the proportion of scores less than or equal to `x`. Convert the result to a percentage.

6. Notice the answers from questions 3 and 5 are different. Why?

*If you compare your result to the shortcut calculation using `sd()`, you'll notice your result is slightly smaller. That's because you used the formula for population variance, which is what we've learned so far in class, whereas R uses the formula for sample variance. That formula's slightly different, for reasons we'll learn about soon.