Summary of Lab Week 7

t-test

First, we'll walk through the steps of a single-sample t-test. Make up a sample with 5-10 data.

```
> X = c(_,_,…)
```

Look at your sample and come up with a reasonable number for the population mean under the null hypothesis (i.e., pick a value somewhere within the range of the sample). Imagine you want to test whether this sample could reasonably have come from a population with that mean.

To perform this hypothesis test, we'll go through the steps we learned in lecture:
1. State the null and alternative hypotheses
2. Compute the test statistic, *t*, from the data
3. Determine the likelihood function
4. Find the critical value, with your chosen alpha level
5. Compare the actual result to the critical value

Step 1 is straightforward. The null hypothesis is the same for all single-sample t-tests: $\mu = \mu_0$. The alternative hypothesis can be $\mu > \mu_0$ or $\mu < \mu_0$ if we have a reason to predict the direction (one-tailed test), or $\mu \neq \mu_0$ if we don't (two-tailed test). Since these are made-up data that you don't have any real theory for, you should do a two-tailed test, meaning the alternative hypothesis is $\mu \neq \mu_0$.

Step 2 requires that you first compute the sample mean and sample standard deviation. You can use `mean()` and `sd()` for this. Next, insert everything into the formula for `t`:

```
> t = (M-mu.0)/(s/sqrt(n))
```

`t` tells how far the sample mean is from the hypothesized population mean, `M-mu.0`, relative to how far would be expected by chance, `s/sqrt(n)`.

Step 3 is straightforward again. The likelihood function, or the sampling distribution under the null hypothesis, for a t statistic is always a t distribution. All you need to know is which t distribution. There's a different t distribution for every possible number of degrees of freedom. For a single-sample t-test, the degrees of freedom is always one less than the sample size (because that's how many squares went into computing `s`). Use `length()` to find your sample size and subtract 1. That's your degrees of freedom, and that tells you which t distribution to use.

In Step 4, you decide how extreme *t* needs to be to reject the null hypothesis. Extreme values of *t* indicate larger differences between *M* and μ_0, relative to the difference that would be expected by chance. We want a critical value that will give us a probability α of making a Type I error. Choose a value of α; it could be 5%, or it could be higher or lower (depending on how risky you're feeling).

Your Type I error rate is going to be the probability, according to the null hypothesis, of getting a value of *t* more extreme than the critical value. Since

we're doing a two tailed test, this can happen by getting $t > t_{crit}$ or $t < -t_{crit}$. Whatever value of t_{crit} you choose, these two probabilities will be equal, so you want to make each one equal $\alpha/2$ so that the total probability of a Type I error is $\alpha/2 + \alpha/2 = \alpha$.

To compute t_{crit}, use the `qt()` function. This function gives you a value of t_{crit} that has a specified probability of being exceeded. The first input is what you want that probability to be, that is $p(t > t_{crit})$. The second input is your degrees of freedom (which tells R which t distribution you want to use). Finally, give `lower.tail = FALSE` as the third input to indicate that you're specifying the probability above t_{crit}, not the probability below (i.e., the upper tail and not the lower tail).

You can also use `qt()` to find the lower critical value, $-t_{crit}$. Just input `lower.tail = TRUE` to indicate that you want to specify the probability in the lower tail. This should give the negative of the result you got above.

Step 5 tells you which hypothesis the data support. Compare your *t* to the critical values. Which hypothesis do you go with?

Try computing `t` again, using the same sample but a different null hypothesis about `mu`. If your first test was significant (i.e., you rejected the null hypothesis), use a new value of `mu.0` that's closer to `M`. If your first test was non-significant (i.e., you retained the null), use a new value of `mu.0` that's further from `M`.

Now that you've done the t-test "by hand," compare your result to R's built-in t-test function. For a one-sample t-test, the input to the `t.test()` function is the sample (`X`), and the hypothesized population mean (`mu.0`).

```
> t.test(X,mu=mu.0)
```

Notice the output of this function gives you the sample mean, `t`, and the degrees of freedom. These should agree with what you found yourself. The output doesn't give you the critical value for *t*, but instead it gives you two other ways of deciding between hypotheses. The first is the p-value, which measures how consistent *t* is with the null hypothesis. If the p-value is less than α, you reject the null hypothesis. The second is the confidence interval, which we'll learn about in lecture this week. The confidence interval is the range of all possible choices of μ_0 that would lead to retaining the null hypothesis (if you chose α = 5%). If your choice of μ_0 is outside the confidence interval, then you should have rejected the null hypothesis (if you used α = 5%).

t distribution

Now we'll look at the sampling distribution for a t statistic, using the procedure of generating a large number of different samples and computing `t` for each sample.

Pick a sample size between `3` and `5`. Generate a single sample from a Standard Normal population, and find its mean.

```
> X = rnorm(n)
```

```
> M = mean(X)
```

The standard error of `M` equals `sigma/sqrt(n)`. Since our population is Standard, `sigma` equals `1`, so the standard error is just `1/sqrt(n)`. This means that if we drew a large number of samples, the distribution of sample means would be `Normal(0,1/sqrt(n))`. So we can turn `M` into a z-score by dividing by the standard error.

```
> z = M/(1/sqrt(n))
```

However, if this were a real experiment, we wouldn't know `sigma`, and we'd have to estimate it using the sample variance.

```
> s = sqrt(var(X))
```

Then we'd use `s/sqrt(n)` as an estimate of the standard error, and use that estimate to compute `t`.

```
> t = M/(s/sqrt(n))
```

We just computed `t` in exactly the same way as in the first part of this lab. In this case, we used `mu.0 = 0`, which is why the numerator for `t` is just `M`.

The point of all this is to compare `z` and `t`. `z` will have a Standard Normal sampling distribution, but we can't compute it in practice because it assumes we know `sigma`. `t` differs from `z` in that it uses the sample standard deviation, `s`, in place of `sigma`. `s` will be different from one sample to the next, meaning there will be extra variability in `t` that's not present in `z`. How does this affect the distribution of `t` versus `z`?

To compare the sampling distributions of `z` and `t`, we'll generate a large number of samples, and compute `z` and `t` from each sample. In order to get the sampling distributions for the null hypothesis, we need to make the null hypothesis be true. In other words, because the true mean of the population you're sampling from is 0, use 0 for μ_0.

We do this using a loop like before, with `{}` braces to indicate there are multiple commands for each step of the loop.

```
> for(i in 1:10000) {
> X = rnorm(n)  #generate a sample
> M = mean(X)  #compute its mean
> z[i] = (M-0)/(1/sqrt(n))  #z-score for M, knowing sigma = 1
> s = sd(X)  #get the sample standard deviation
> t[i] = (M-0)/(s/sqrt(n))  #t, using s as an estimate of sigma
> }  #this tells R the loop is done
```

Now we have a large set of `z`s and `t`s for the same samples. Compare their distributions. First, use the following command to tell R to draw two graphs side by side. Then, make histograms of `z` and `t`.

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

Notice how the `t` distribution is a bit wider than the `z` (Standard Normal) distribution, and the tails are especially thicker. This is because of the variability in the sample standard deviation, which affects `t` but not `z`.