Introduction to Biostatistics (171:161)
Breheny

Lab #10

In lab #10, we are going to learn how to use SAS/R to calculate power for one-sample studies with continuous outcomes, as well as how to carry out χ^2-tests and Fisher's exact test. In SAS, the analysis of contingency tables is accomplished by adding options to `PROC FREQ`. R provides the separate functions `chisq.test` and `fisher.test` to carry out these procedures.

1 Power and sample size calculations

SAS can also perform power and sample size calculations for you using `PROC POWER`; there are several power calculation functions in R as well as multiple add-on packages available. To get one-sample power calculations for t-tests, you use the `ONESAMPLEMEANS` option in SAS, and the `power.t.test` function in R.

Let's look at what the power might have been for the cystic fibrosis. Glancing at the results of the study, it would seem reasonable that the true standard deviation is about 200 and the effect size of the drug is about 125. If this is true and we were to repeat the study with another 14 patients, our power would be given by:

SAS:

```
PROC POWER;
  ONESAMPLEMEANS
    MEAN = 125
    NTOTAL = 14
    STDDEV = 200
    POWER = .;
RUN;
```

R:

```
power.t.test(delta = 125, sd = 200,
             n = 14, type = "paired")
```

The implementation in both SAS and R is similar, in that you can leave out any of the above quantities (n, difference in means, SD, power), and SAS/R will solve for the remaining quantity. SAS note: in SAS, "." means "missing"; here it represents the quantity that you want SAS to calculate. You can only have one ".".

So, if we instead wanted to know the number of subjects we would need in order to get a power of 80%, we can submit:

SAS:

```
PROC POWER;
  ONESAMPLEMEANS
    MEAN = 125
    NTOTAL = .
    STDDEV = 200
    POWER = .8;
RUN;
```

R:

```
power.t.test(delta = 125, sd = 200,
             power = .8, type = "paired")
```

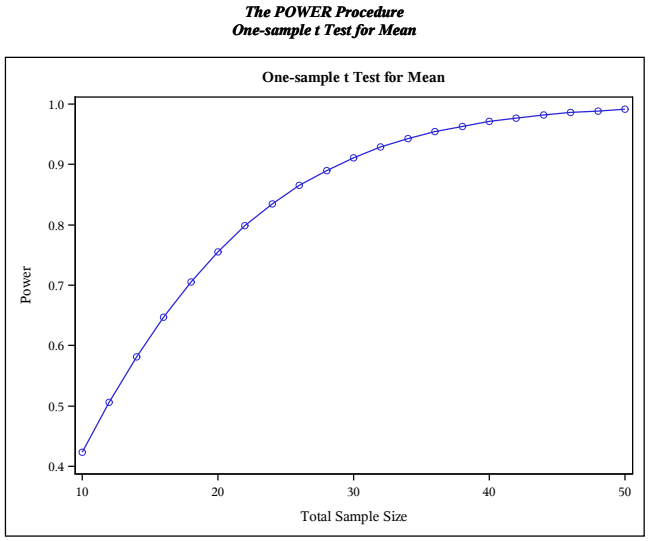
Both programs inform you that you need 23 subjects for $> 80\%$ power. SAS also has a neat built-in plotting feature (you can make the same sorts of plots in R, but it has to be done manually):

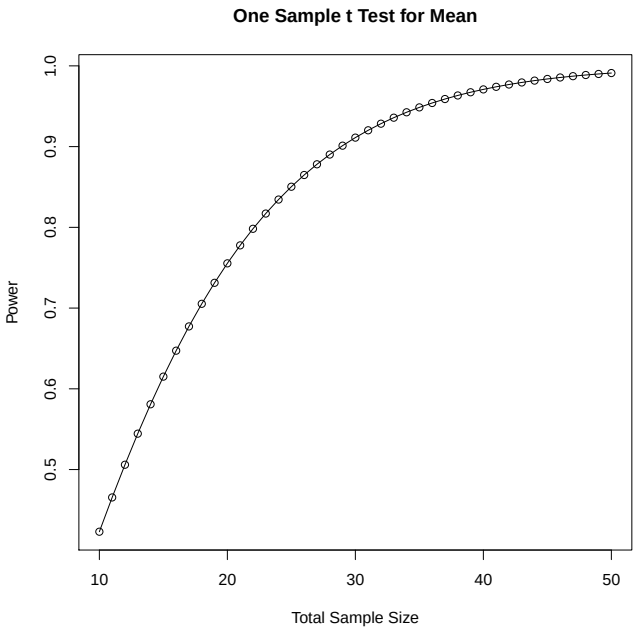
SAS:

```
PROC POWER;
  ONESAMPLEMEANS
    MEAN = 125
    NTOTAL = 20
    STDDEV = 200
    POWER = .;
  PLOT X = N MIN = 10 MAX = 50;
RUN;
```

R:

```
n <- 10:50
t <- power.t.test(delta = 125, sd = 200,
     n = 10:50, type = "paired")
plot(n, t$power, xlab = "Total Sample Size",
     ylab = "Power", type = 'o',
     main = "One Sample t Test for Mean")
```





Try making some different plots to make sure you understand how changing one factor affects the others – this kind of thing makes for a good quiz question.

2 Lister's experiment

Download and import the data set `lister.txt`. As we have seen before, we can get descriptive statistics using

SAS:

```
PROC FREQ DATA = lister;
  TABLES Group*Outcome;
RUN;
```

R:

```
tab <- table(lister)
tab
```

which reproduces the table displayed in lecture. Now, we can have SAS perform hypothesis tests by adding options to the `TABLES` statement. For some reason, SAS bundles χ^2-tests and Fisher's exact test together; you can't get one without the other. You can specify `FISHER` or `CHISQ`, but either way, you get both tests. In `R`, you use `fisher.test` or `chisq.test`, depending on which one you want:

SAS:

```
PROC FREQ DATA = lister;
  TABLES Group*Outcome / CHISQ;
RUN;
```

R:

```
chisq.test(tab, correct = FALSE)
fisher.test(tab)
```

SAS gives you lots of "Chi-Square" tests. They're all quite similar, but the one we discussed in class is the regular "Chi-Square" entry. As with `prop.test`, by default `R` makes a little adjustment/correction to the χ^2 test; `correct = FALSE` shuts it off. Again, it isn't necessarily wrong to make the adjustment, but the answer you get won't agree with the approach we discussed in class.

In SAS, we can get odds ratios and confidence intervals by tacking on a `RELRISK` option as well:

```
PROC FREQ DATA = lister;
  TABLES Group*Outcome / CHISQ RELRISK;
RUN;
```

As you might have guessed from the name, this option also yields relative risks. Be very careful with this, however. Relative risks are meaningless for case-control studies!

In `R`, note that `fisher.test` gives you confidence intervals for the odds ratio, while `chisq.test` does not. Unfortunately, there is no convenient way to get approximate confidence intervals for the odds ratio in `R` (you would have to install the `epitools` package for this).

In SAS, you can request exact confidence intervals for the odds ratio with an `EXACT` statement:

```
PROC FREQ DATA = lister;
  TABLES Group*Outcome;
  EXACT OR;
RUN;
```

3 Breast cancer and age at first labor

As you may recall from earlier labs, with categorical data, people often don't report every single observation – nor do they have to. The number of subjects that fell into each category is all you need to know. So, for example, recall the results of the CDC's study of the relationship between breast cancer and the age at which a woman gave birth to her first child:

	Cancer	
	No	Yes
Before age 25	4475	65
25 or older	1597	31

If we want to analyze this in SAS, we need to make a data set out of these numbers, and we certainly wouldn't want to type in all 6,168 results for each individual woman. Instead, we want to be able to type only the number of subjects that fell into each category, and then inform SAS of how the data set is organized. So, we will need to make a file (in, say, excel, or a simple text file) that looks like:

FirstLabor	BreastCancer	N
After25	Yes	31
After25	No	1597
Before25	Yes	65
Before25	No	4475

Then we can import and analyze as before, although we will need to add a line telling SAS that the number of subjects is in a column called "N":

```
PROC FREQ DATA = bc;
  TABLES FirstLabor*BreastCancer / CHISQ RELRISK;
  WEIGHT N;
RUN;
```

In `R`, it is easier to just enter the data directly:

```
tab <- cbind(c(4475, 1597), c(65, 31))
fisher.test(tab)
```

The `cbind` function binds lists of numbers together by column (there is also an `rbind` function if you would prefer to enter the numbers by row). If you look at `tab` and experiment with `cbind`, it should be pretty clear how it works. Alternatively, you can use `matrix`.