Power and Sample Size

Biol 697

September 27, 2012

So, in class today, we talked about a simulation based approach to assessing power. I didn't get quite the time to go into the code of such a simulation in sufficient detail, so, here's the example layed out much more clearly, piece by piece.

Remember, the problem is that we want to know whether administering a drug changes hear rate. H_o is that the average effect of the drug on heart rate is 0. In actuality, is speeds heart rate up by 15 beats per minute. What is the effect of sample size of patients on power, assuming a SD of 6?

How do we answer this question? Well, there are three steps.

1 Simulate the experiment

The first thing we want to do is simulate many trials of administering a drug to a sample population and seeing the average change in heart rate. Hopefully this is old hat, but, let's walk through it.

The first thing we're doing is setting a seed (so this is repeatable) as well as other parameters that may be used throughout the simulation. We may want to tweak the simulation in the future, and we want to be able to do that by changing one number in one place, not 10 numbers in ten places.

```
##First I need to simulate a number of samples
set.seed(09262012)


#my sample size vector
n<-rep(1:10, 50)

#the Standard Deviation for the whole simulation
simSD<-6
meanEffect<-15
```

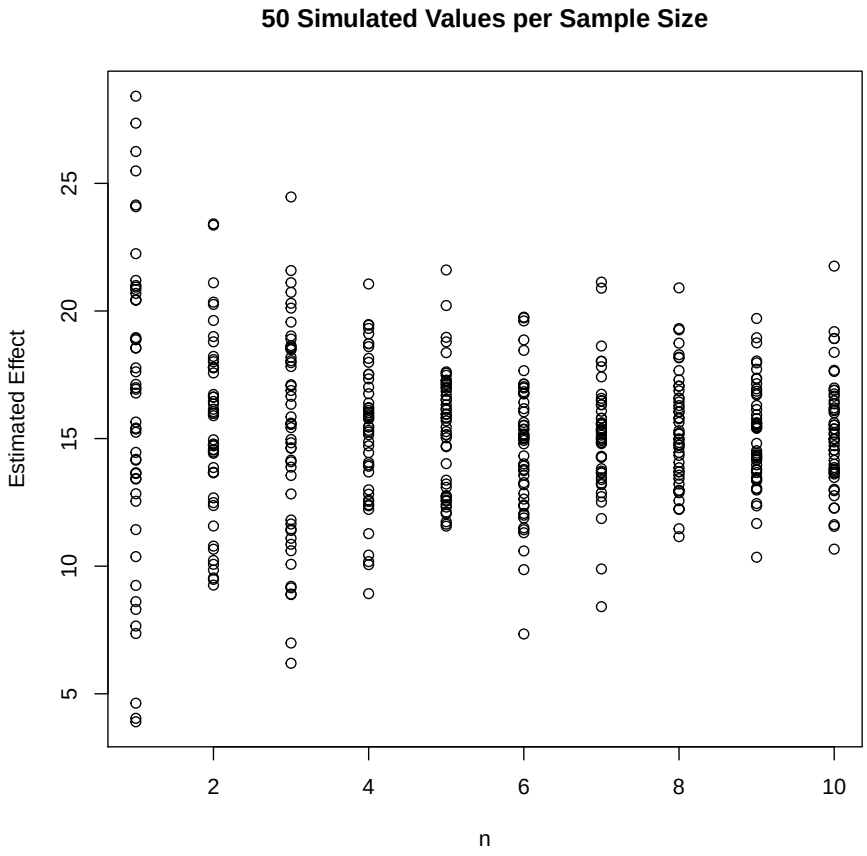
The next thing we'll do, is loop over all of those possible simulated experiments. With the vector n, we have the sample size for each sim. Great. We can

use that do draw n_i random normal variables, then get their mean which we'll use to fill in our vector `vec`.

```
#a For loop to simulate the experiments where we get means
vec<-rep(NA, length(n))

for(i in 1:length(n)){

  #ok, for each simulation, calculate the mean of a bunch of random
  # variables drawn from a population
  vec[i]<-mean(rnorm(n[i], mean=meanEffect, sd=simSD))

}
```

Great - let's see what this looks like:

```
#plot the result
plot(vec ~ n, ylab="Estimated Effect",
     main="50 Simulated Values per Sample Size")
```



As you can see, with a larger sample size, we get closer to a mean of 15, although there's still a good bit of variation.

2 Calculating p values for each simulation

Next, as each simulation is its on test statistic, we need to calculate the p-value for each one. Now, our null distribtion has a mean of 0, but, we're allowing it to have the same SD as our population. So, we want to compare each mean to a normal distribution with mean 0 and SD of 6.

Now, this is a 2-tailed test, so, we have two issues to grapple with. First, we're going to have values that fall in the lower and upper tails of this null normal distribution (i.e., they can be positive and negative). Fortunately, we can just take the absolute value of any simulated mean, and then use the lower.tail argument to get the correct output from `pnorm`. We could have also done

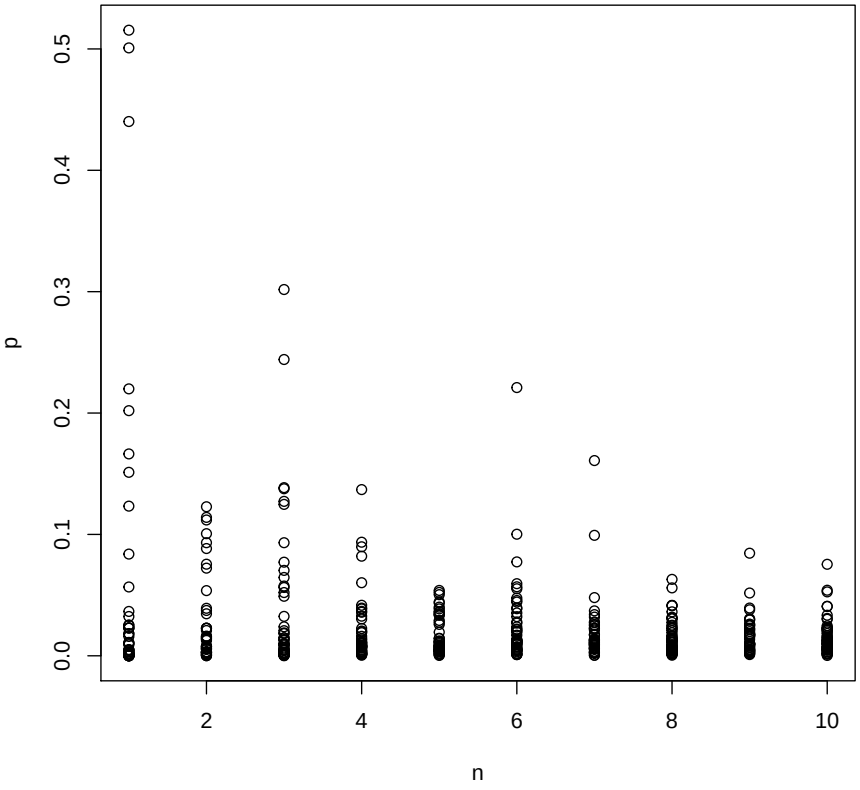
`1-pnorm` and gotten the same result. Second, because this is a two-tailed test, we're going to need to multiple the result by 2.

Rather than looping over the whole thing, `pnorm` is nice in that we can give it a vector as an argument and it will give us the correct vector of responses. We'll talk about *vectorization* another day.

```
##
# Now, calculate the p value for each mean.
# We're using abs and lower.tail=F to deal with both
# positive and negative values. Note, the mean of
# the null distribution is 0, and this is a 2-tailed test
pvec<-pnorm(abs(vec), sd=simSD, lower.tail=FALSE)*2
```

Great, now let's plot the result.

```
#plot the p values
plot(pvec ~ n, ylab="p")
```



3 Assessing Power

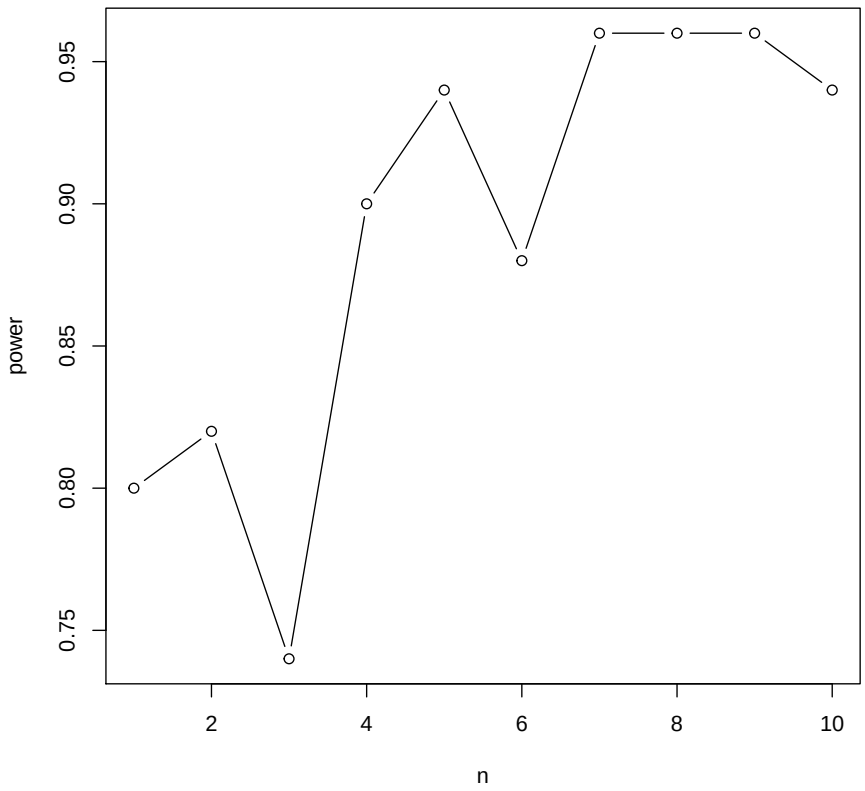
Now that we have a vector of p values that are tied to a vector of sample sizes, we can use this and a little clever application of the which statement to calculate the power at each sample size. Remember, power = 1 - the probability of falsely failing to reject the null hypothesis.

So, what we'll do is create a blank that is the same length as the number of sample sizes we have (i.e., 10, as the possible sample sizes are 1, 2, 3...10). Next, we'll loop over all of those sample sizes. For each one, we'll extract just the piece of the `pvec` that matches the relevant sample size, and then look at the fraction of tests that got it wrong.

```
##
# Now let's look at how sample size influences power

#first, we need a blank vector of power values
power<-rep(NA, max(n))

#now loop over all possible sample sizes
for( i in 1:( max(n) ) ){

  # pull out the relevant p values based on their
  # matching up with the sample size we're interested
  # in using indices derived from the vector of sample
  # sizes
  subPvec <- pvec[which(n==i)]

  #ok, now that we have all of the p values, see what fraction of
  #simulations contained a type II error.  1-p(error) = power
  power[i] <- 1 - sum(subPvec > 0.05) / length(subPvec)
}
```

Now let's plot that power vector to see how n and power are related.

```
#plot the relationship between sample size and power
plot(power ~ I(1:10), xlab="n", ylab="power", type="b")
```



Great - we can see that power increases as sample size increases. However, what's great is that our difference is so large that even at low sample sizes, we have decent power. Try playing around the the attached script a bit. Increase the SD. Change the mean. What do you see? For example, if we kick the SD up very high, we actually find that power decreases with sample size. This is counter-intuitive, but, remember, with a large SD, we're very likely to include 0 in our 95% confidence interval. This, we SHOULD have low power, as we've setup a simulation where 0 is possible as an estimate A single value alone may not make for a good test of our hypothesis, and we'll need some better statistical techniques.