Lab7_DianaKim

2025-10-29

Simulation and Power Analysis

Reading

Reference

Overview

Part of the course project is to conduct a simulation-based power-analysis. This document provides background and code for power-analyses and use of simulations for sample-size and design planning. Here we look at the simple case for t-tests. But, the framework and thinking followed here are relevant even as we move on to more complex designs and tests.

t-test

What is a t-test? A t-test is a statistical test that compares two means to determine if they're different.

The test calculates a **t-statistic**, which quantifies how many standard errors separate the two means. Larger t-statistic => groups are far apart (relative to variability in the data) => probably different. Smaller t-statistic => groups are close => difference is probably just noise

To interpret an observed the t-statistic, we compare it to the **sampling distribution of t-statistics under the null hypothesis** (i.e., assuming no true difference exists). This distribution tells us what t-statistics we'd expect to see if we repeatedly sampled from populations with identical means.

The **p-value** represents where our observed t-statistic falls in this null distribution. If $p < .05$, we conclude the observed difference is unlikely to have occurred by chance alone, and we reject the null hypothesis.

t-test and lm() The t-test is lm (Outcome ~ group) when 'group' has exactly 2 levels (e.g. control vs. treatment)

[if there are more than 2 levels it becomes ANOVA... More on these connections later in the course]

Null-hypothesis

The null-hypothesis is the hypothesis that your experimental manipulation didn't work. We can simulate null-hypotheses in R for any experimental design. We do this in the following way:

1. Use R to generate sample data in each condition of a design
2. Make sure the sample data comes from the very same distribution for all conditions (ensure that there are no differences)

3. Compute a test-statistic for each simulation, save it, then repeat to create the sampling distribution of the test statistic.
4. The sampling distribution of the test-statistic is the null-distribution. We use it to set an alpha criterion, and then do hypothesis testing.

Null for a t-test

Let's construct the null distribution for a t-test

```
# samples A and B come from the same normal distribution
A <- rnorm(n=10,mean=10, sd=5)
B <- rnorm(n=10,mean=10, sd=5)
```

```
# let's run the t-test for this one pretend simulation
t.test(A,B,var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  A and B
## t = 0.97777, df = 18, p-value = 0.3411
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.865221  5.112788
## sample estimates:
## mean of x mean of y
##  9.780760  8.156977
```

```
# let's explore using lm() to ask the same question
Outcome <- c(A, B)
group <- factor(rep(c("A","B"), each=10))
summary(lm(Outcome ~ group))
```

```
##
## Call:
## lm(formula = Outcome ~ group)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8785 -2.5220 -0.7766  3.1037  7.0229
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    9.781      1.174   8.329 1.37e-07 ***
## groupB        -1.624      1.661  -0.978    0.341
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.713 on 18 degrees of freedom
## Multiple R-squared:  0.05043,    Adjusted R-squared:  -0.002319
## F-statistic: 0.956 on 1 and 18 DF,  p-value: 0.3411
```

Q1: t-test vs lm() We've said t-tests and lm() can be equivalent. But the outputs do look quite different!

How would you go about deriving the results you got for the t.test from the output of lm()? More specifically, derive the following from the lm() output and provide your reasoning

- Mean of group A
- Mean of group B
- t-value : *[note that in lm() we don't care in this scenario about whether Group A is different from 0, we only care about if group B is different from Group A (which is the reference level)]*
- p-value associated with the t-value

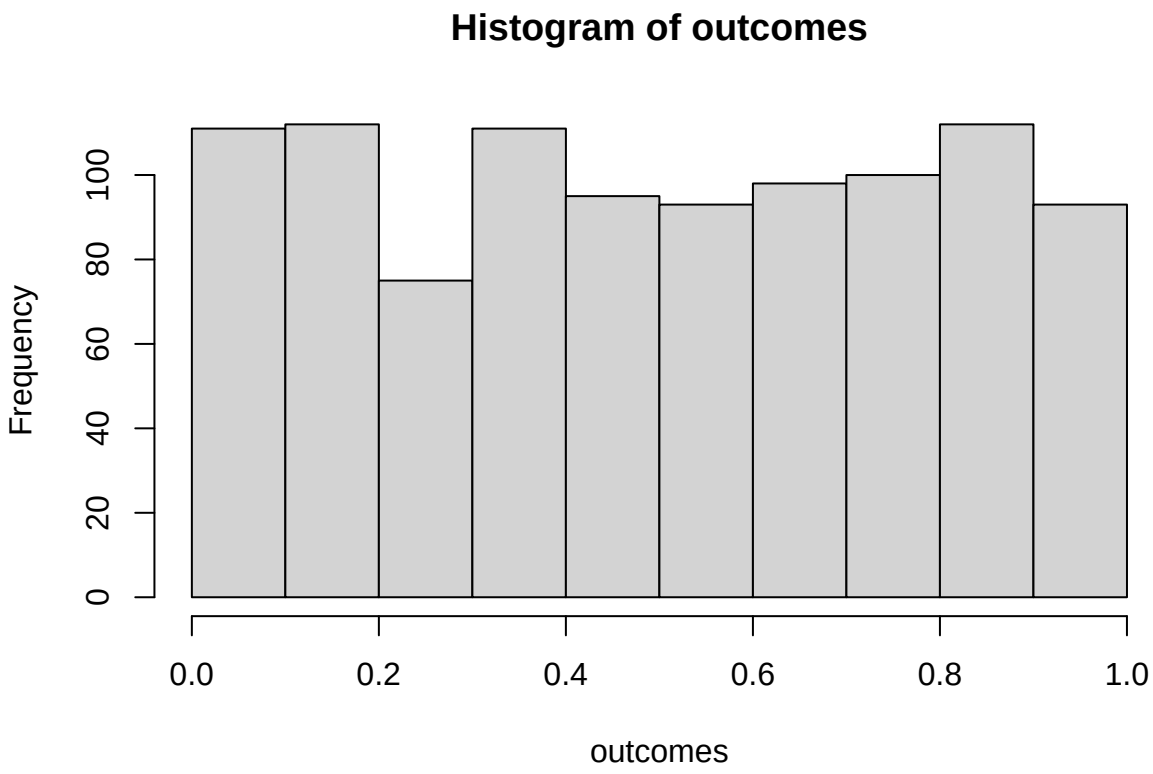
<Mean of group A is 'Intercept' of the model, and mean of group B is the sum of intercept and group B's coefficient. They match the results from t test!>

Now, with the explanation of t tests aside, let's return to simulation-based analyses – and where we were with the null hypothesis!

```
# running the simulation
# everytime we run this function we do one simulated experiment and return the p-value
sim_null <- function(){
  A <- rnorm(n=10,mean=10, sd=5)
  B <- rnorm(n=10,mean=10, sd=5)
  return(t.test(A,B,var.equal=TRUE)$p.value)
}

# use replicate to run the sim many times
outcomes <- replicate(1000,sim_null())

# plot the null-distribution of p-values
hist(outcomes)
```




```
# proportion of simulated experiments that had a p-value less than .05
length(outcomes[outcomes<.05])/1000
```

```
## [1] 0.064
```

We ran the above simulation 1000 times. By definition, we should get approximately 5% of the simulations returning a p-value less than .05. If we increase the number of simulations, then we will get a more accurate answer that converges on 5% every time.

Alternative hypothesis

The very general alternative to the null hypothesis (no differences), is often called the **alternative** hypothesis: that the manipulation does cause a difference. More specifically, there are an infinite number of possible alternative hypotheses, each involving a difference of a specific size.

Consider this. How often will we find a p-value less than .05 if there was a mean difference of 5 between sample A and B. Let's use all of the same parameters as before, except this time we sample from different distributions for A and B.

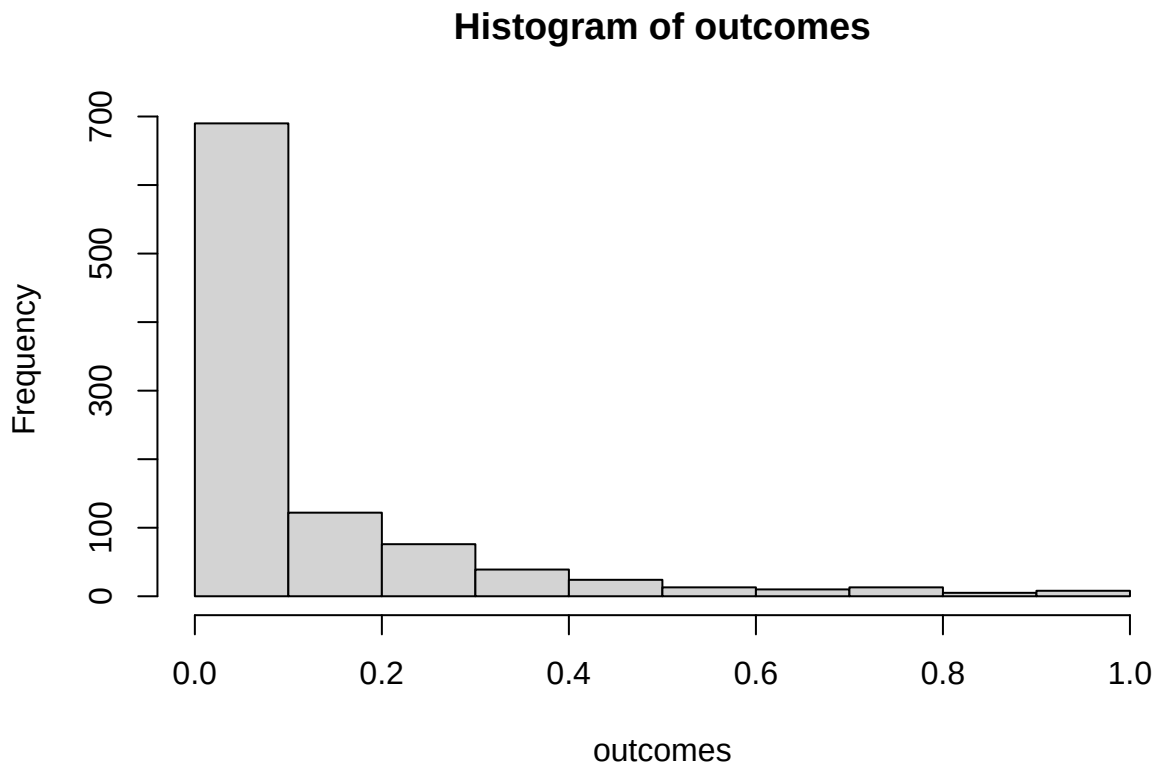
```
#Example of a single test
A <- rnorm(n=10,mean=10, sd=5)
B <- rnorm(n=10,mean=15, sd=5)
t.test(A,B,var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  A and B
## t = -3.6326, df = 18, p-value = 0.001904
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -12.039250  -3.216144
## sample estimates:
## mean of x mean of y
##  6.169879 13.797576
```

```r
# simulations: make the mean for B 15 (5 more than A)
sim_alternative <- function(){
  A <- rnorm(n=10,mean=10, sd=5)
  B <- rnorm(n=10,mean=15, sd=5)
  return(t.test(A,B,var.equal=TRUE)$p.value)
}

# use replicate to run the sim many times
outcomes <- replicate(1000,sim_alternative())

# plot the distribution of p-values
hist(outcomes)
```



```r
# proportion of simulated experiments had a p-value less than .05
length(outcomes[outcomes<.05])/1000
```

```
## [1] 0.545
```

We programmed a mean difference of 5 between our sample for A and B, and we found p-values less than .05 a much higher proportion of the time. This is sensible, as there really was a difference between the samples (we put it there).

Q2: Power? In your manipulation how many standard deviations does mean of B seem to be from mean of A in your simulations? What is the power you'd expect for such a study? What is the power you observed (you can use calculations of how often the p-value is significant)? Are these the same and why?

<The group means are 1 standard deviation away from each other but the calculation above shows 0.537. It can be higher in case our sample size gets very large.>

Power and Effect Size

Power: the probability of rejecting a null-hypothesis, given that there is a true effect of some size.

Effect-size: In general, it's the assumed size of the difference. In the above example, we assumed a difference of 5, so the assumed effect-size was 5.

There are many other ways to define and measure effect-size. Perhaps the most common way is Cohen's D. Cohen's D expresses the mean difference in terms of standard deviation units. In the above example, both distributions had a standard deviation of 5. The mean for A was 10, and the mean for B was 15. Using Cohen's D, the effect-size was 1. This is because 15 is 1 standard deviation away from 10 (the standard deviation is also 5).

When we calculated the proportion of simulations that returned a p-value less than .05, we found the power of the design to detect an effect-size of 1.

Power depends on three major things:

1. Sample-size
2. Effect-size
3. Alpha-criterion

Power is a property of a design. The power of a design increases when sample-size increases. The power of a design increases when the actual true effect-size increases. The power of a design increases then the alpha criterion increases (e.g, going from .05 to .1, making it easier to reject the null).

Power analysis with R

There are many packages and functions for power analysis. Power analysis is important for planning a design. For example, you can determine how many subjects you need in order to have a high probability of detecting a true effect (of a particular size) if it is really there.

pwr package

We can use the `pwr` package to find the power for an independent sample t-test, with n=10, to detect an effect-size of 1.

Q3: find power using pwr.t.test You can always execute ?pwr.t.test to get the documentation of how to use this function

This is a two-sample design (i.e. two independent groups are tested), needing a two samples t-test And our null hypothesis suggests that we are looking at a two-sided test.

```
library(pwr)
?pwr.t.test
```

```
## starting httpd help server ... done
```

```
# Your answer here
A <- rnorm(n=10,mean=10, sd=5)
B <- rnorm(n=10,mean=15, sd=5)
t.test(A,B,var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  A and B
## t = -3.0655, df = 18, p-value = 0.006663
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -9.605562 -1.793379
## sample estimates:
## mean of x mean of y
##  9.377684 15.077155
```

```
pwr.t.test(n = 10, d = 1, sig.level = 0.05, power = NULL,
    type = c("two.sample"),
    alternative = c("two.sided"))
```

```
##
##      Two-sample t test power calculation
##
##               n = 10
##               d = 1
##       sig.level = 0.05
##           power = 0.5620066
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

There are many functions for directly computing power in R. Feel free to use them. In this class, we learn how to use simulation to conduct power analyses. This can be a redundant approach that is not necessary, given there are other functions we can use. Additionally, we will not get exact solutions (but approximate ones).

Nevertheless, the existing power functions can be limited and may not apply to your design. The simulation approach can be extended to any design. Learning how to run the simulations will also improve your statistical sensibilities, and power calculations will become less of a black box.

Two more things before moving onto simulation: power-curves, and sample-size planning.

power-curves

A design's power is in relation to the true effect-size. The same design has different levels of power to detect different sized effects. Let's make a power curve to see the power of a t-test for independent samples (n=10) to detect a range of effect-sizes

```
# function to run a simulated t-test
sim_power <- function(x){
  A <- rnorm(n=10, mean=0, sd=1)
  B <- rnorm(n=10, mean=(0+x), sd=1)
  return(t.test(A,B, var.equal=TRUE)$p.value)
}
```

Q4. Power-curve (varying effect sizes) power is on y-axis; and effect-size on the x

```
#----- edit for answer (a) below
effect_sizes <- seq(0,2,length.out=10)
#effect_sizes <- seq(.1,2,.1)
```

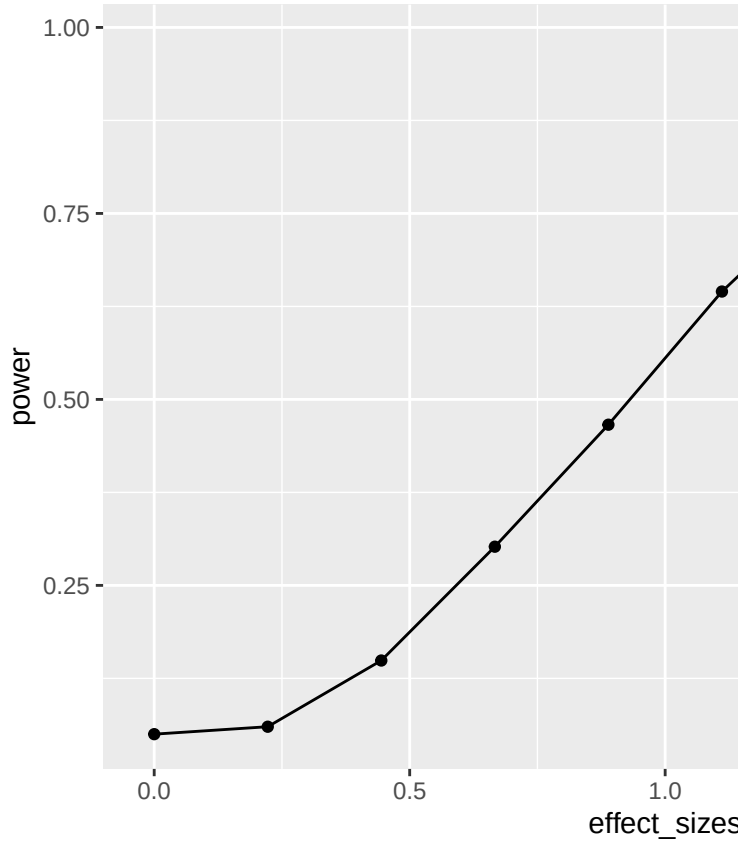
```
#now the code below takes each value in the effect_sizes vector
#calculates power for each effect size value
# and then plots it

power <- c()  # empty vector to store results

for(i in 1:length(effect_sizes)) {
  sims <- replicate(1000, sim_power(effect_sizes[i]))
  power[i] <- length(sims[sims<.05])/length(sims)
}

plot_df <- data.frame(effect_sizes,power)

library(ggplot2)
ggplot(plot_df, aes(x=effect_sizes,
                    y=power))+
  geom_point()+
  geom_line()
```



(a) define 10 effect sizes between the values of 0 to 2

#####(b) What does this curve tell you about detecting small effects (e.g. d= 0.2) with n=10?

<It will be very difficult to detect small effects given such small sample size, since the power gets low.>

#####(c) If you need to get a power-curve for n=1000 instead, what do you need to change in the current code? <In the sim_power function, the 'n' variable in each A and B distributions has to be 1000 not 10.>

This power curve applies to all independent-sample t-tests with n=10. It is a property, or fact about those designs. Every design has it's own power curve. The power curve shows us what **should** happen (on average), when the **true** state of the world involves effects of different sizes.

If you do not know the power-curve for your design, then you do not know how sensitive your design is for detecting effects of particular sizes. You might accidentally be using an under powered design, with only a very small chance of detecting an effect of a size you are interested in.

If you do know the power-curve for your design, you will be in a better position to plan your experiment, for example by modifying the number of subjects that you run.

Sample-size planning

Here is one way to plan for the number of subjects that you need to find an effect of interest.

1. Establish a smallest effect-size of interest
2. Create a curve showing the power of your design as a function of number of subjects to detect the smallest effect-size of interest.

It's not clear how you establish a smallest effect-size of interest. But let's say you are interested in detecting an effect of at least d = .2. This means that two conditions would differ by at least a .2 standard deviation

shift. If you find something smaller than that, let's say you wouldn't care about it because it wouldn't be big enough for you to care. How many subjects do you need to have a high powered design, one that would almost always reject the null-hypothesis?

This is for an independent samples t-test:

```
num_subjects <- seq(10,1000,10)
```

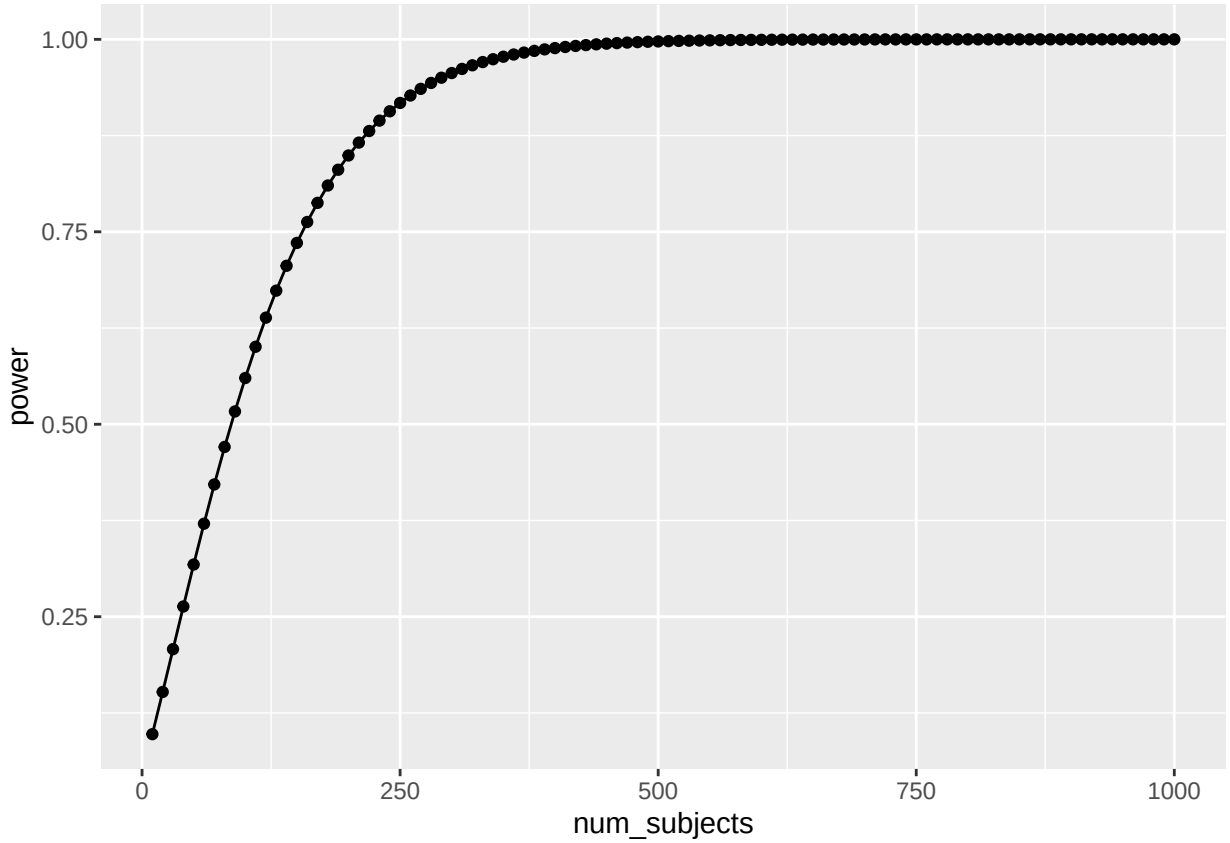
Q5: Write code to create an empty vector named power.

then use a loop to iterate through the different number of subjects (using pwr), and calculate the corresponding power that you save into power for plotting

```
#Your answer here
power <- c()
for (i in 1:length(num_subjects)) {
 power[i] <- pwr.t.test(n = num_subjects[i], d = 0.3, sig.level = 0.05,
                        type = "two.sample", alternative = "two.sided")$power
}
#power
```

```
plot_df <- data.frame(num_subjects,power)

#library(ggplot2)
ggplot(plot_df, aes(x=num_subjects,
                    y=power))+
  geom_point()+
  geom_line()
```



Well, it looks like you need many subjects to have high power. For example, if you want to detect the effect 95% of the time, you would need around 650 subjects. It's worth doing this kind of analysis to see if your design checks out. You don't want to waste your time running an experiment that is designed to fail (even when the true effect is real).

Simulation approach to power calculations

The simulation approach to power analysis involves these steps:

1. Use R to sample numbers into each condition of any design.
2. You can set the properties (e.g., n, mean, sd, kind of distribution etc.) of each sample in each condition, and mimic any type of expected pattern
3. Analyze the simulated data to obtain a p-value (use any analysis appropriate to the design)
4. Repeat many times, save the p-values
5. Compute power by determining the proportion of simulated p-values that are less than your alpha criterion.

For all simulations, increasing number of simulations will improve the accuracy of your results. We will use 1000 simulations throughout. 10,000 would be better, but might take just a little bit longer.

Simulated power for a t-test

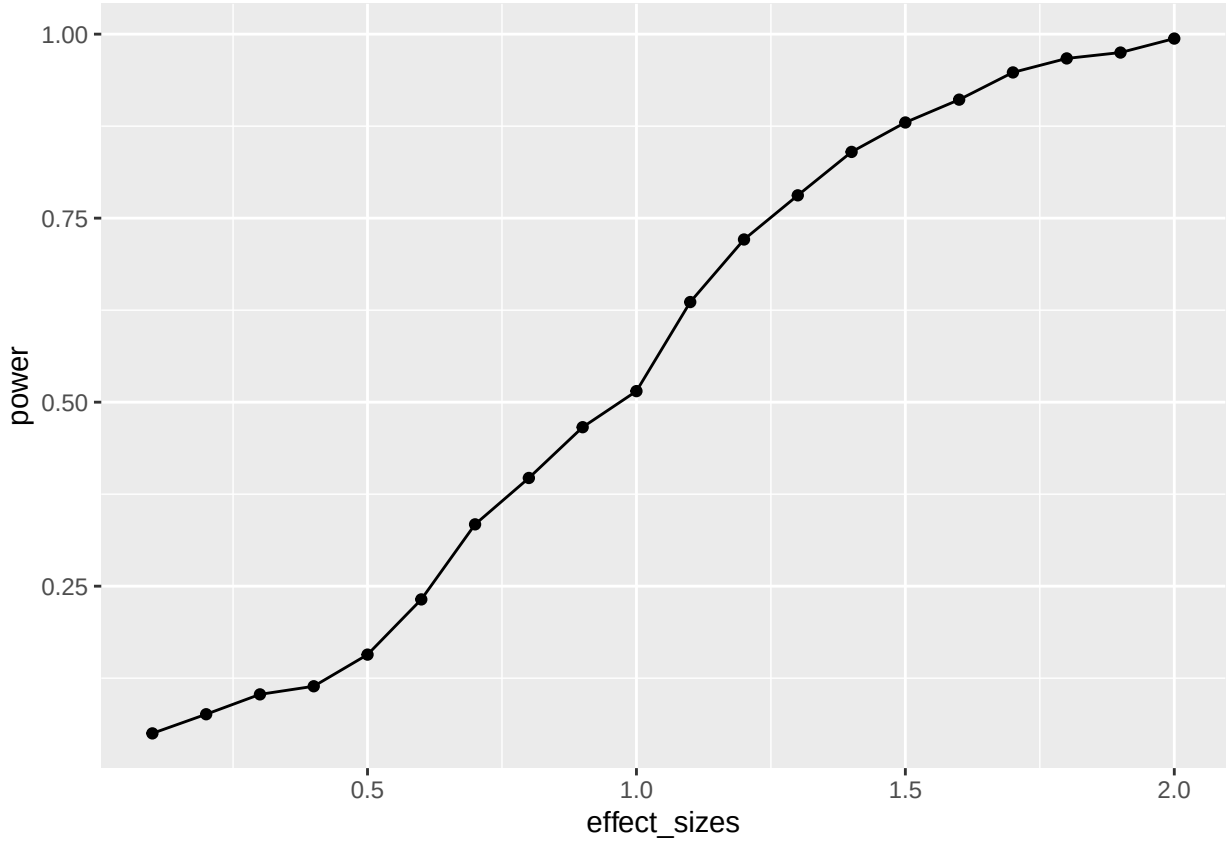
A power curve for n=10.

```r
# function to run a simulated t-test
sim_power <- function(x){
  A <- rnorm(n=10,mean=0, sd=1)
  B <- rnorm(n=10,mean=(0+x), sd=1)
  return(t.test(A,B,var.equal=TRUE)$p.value)
}

# vector of effect sizes
effect_sizes <- seq(.1,2,.1)

# run simulation for each effect size 1000 times
# (this is an alternate way to do what loops were doing)
power <- sapply(effect_sizes,
          FUN = function(x) {
            sims <- replicate(1000,sim_power(x))
            sim_power <- length(sims[sims<.05])/length(sims)
            return(sim_power)})
# combine into dataframe
plot_df <- data.frame(effect_sizes,power)

# plot the power curve
ggplot(plot_df, aes(x=effect_sizes,
                    y=power))+
  geom_point()+
  geom_line()
```



Q6. Compare your solution from Q4 with the output — what is your takeaway?