Formatted Async 3 Code

Introduction

In this code we're going to work though an extended example that demonstrates how randomization inference works. As noted in the lecture, we are working through the example of a randomized assignment of men and women to eat (or not eat) soybeans and we are measuring the level of estrogen present in each of their bloodstreams (perhaps in parts/million).

First, we create a grouping variable with two groups, one called "Man", and another called "Woman".

```
group <- c(rep("Man",20),rep("Woman",20))
```

To these groups, we assign silly, but schematically heplful *potential outcomes* to treatment and control. We say that, by some chance, we sampled men into our study that had estrogen ppm levels that ranged from 1-20, in perfect increments. Also, what luck, but we sampled women into the study that had estrogen levels that ranged from 51-70. So, the women have on average higher estrogen beginning the study.

```
po_control <- c(seq(from = 1, to = 20), seq(from = 51, to = 70))
po_treatment <- po_control #no effect because potential outcomes in treatment are the same
po_control
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 51 52 53
## [24] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

```
po_treatment
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 51 52 53
## [24] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

Per our randomization scheme, we are going to randomly assign the individuals to either eat lots of tofu (`treatment == 1`) or to eat no tofu (`treatment == 0`). To do this, we write a simple function that will randomly place zeros and ones for the treatment and control.

```
randomize <- function() {
  sample(c(rep(0,20),rep(1,20)))
}
## from the help file
## For sample the default for size is the number of items inferred from the first argument
```

As is the case with David R's comments in the async material, if this code isn't strictly intuitive to you, we have lots of options to write other randomization functions.

```
cases <- c(0,1)
randomizeV2 <- function() {
  sample(cases, size = length(group), replace = TRUE)
  }

randomize()
```

```
##  [1] 1 1 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 1
## [36] 1 1 1 0 0
```

With our randomization function in hand, we can now set up our vector of treatment assignments. This is simply storing the results of our function randomize in a vector object called treatment.

What happens when you run this next cell a few times? Are you satisfied that you're getting a distribution of treatment assignemnts that is 50-50 every time?

```
treatment <- randomize() #Conduct randomization for this experiment
treatment
```

```
##  [1] 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1
## [36] 0 0 1 0 1
```

```
table(treatment)
```

```
## treatment
##  0  1
## 20 20
```

But, what happens when we also look at the distribution of the treatments, *within* the men and women? Run the next cell a few times. Are you satisfied that this assignment is producing randomizations that are also 50-50 with the sets of men and within the sets of women?

```
treatment <- randomize()
table(group, randomize())
```

```
##
## group    0  1
##   Man    9 11
##   Woman 11  9
```

Recall that we are setting up an experiment that has **no** effect. As Green and Gerber point out in *Field Experiments* in the case of the sharp-null, we are actually testing against the possiblity that we observe both the potential outcomes! (As a comprehension check, explain why this is true).

Next, we create a vector of realized outcomes, first using the compact notation that Green and Gerber prefer using maths. For those randomized to treamtent, we multiply the potential outcome to treatment time the treatment vector (which is a 1 when they were assigned to treatment), and for those in control, into this vector we assign the potential outcome to control time the quantity `(1 - treatment)` which will be one when they are in the treatment group.

```
outcomes <- po_treatment * treatment + po_control*(1-treatment)
outcomes
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 51 52 53
## [24] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

One of the points that David R. and David B. make in their lecture is that we could have alternatively written this as a *very* simple subset, if math isn't your thing but programming control flows is.

Standard practice in R would be the follow style of subset:

```
outcomesV2 <- rep(NA, length(group))
outcomesV2[treatment == 0] <- po_control[treatment == 0]
outcomesV2[treatment == 1] <- po_treatment[treatment == 1]
```

Although if you're coming from another language that doesn't place such a high value on vectorized operations, you might go for it in the following way:

```
outcomesV3 <- rep(NA, length(group))
for(i in 1:length(group)) {
  if(treatment[i] == 0) {
    outcomesV3[i] <- po_control[i]
    } else if(treatment[i] == 1) {
      outcomesV3[i] <- po_treatment[i]
```

```
    }
  }
```

Check that they're all doing the same thing

```
table(outcomes == outcomesV2)
```

```
##
## TRUE
##   40
```

```
table(outcomes == outcomesV3)
```

```
##
## TRUE
##   40
```

To my eye, the clarity of either the math version or the vector subset version are much more transparent in what they're accomplishing – they have the added benefit of working more quickly in R if you have a large number of opertaions to run.

Ok so we've got our data set up. Now what?

Now that we have the data set up, we can begin to examine what the lecture is really about, what is the distribution of ATE that we observe due to the different possibly assignments to treatment and control. A few points to remember:

1. From last week: The difference in sample means between the treatment and control groups is an unbiased estimator of the ATE.
2. Also from last week: This does not imply that any one realization of treatment/control assignment is guaranteed to exactly produce that ATE.

This is the entire point of understanding the distribution of the ATE.

To get here, first lets write another function that will calculate the ATE.

```
est_ate <- function(outcome, treat) {
  mean(outcome[treat==1]) - mean(outcome[treat==0])
  }
est_ate
```

```
## function(outcome, treat) {
##   mean(outcome[treat==1]) - mean(outcome[treat==0])
##   }
```

And then, let's actually compute the average treatment effect for this experiment:

```
ate <- est_ate(outcomes, treatment)
ate
```

```
## [1] -8.2
```

What gives! We created this data such that there is *exactly* zero treament effect. Note, even more specifically than creating the data so that there was no *average treatment effect* we constructed this so that there was no effect at all – this is at the potential outcomes level! Recall that $P(0) = P(1)$.

How big is that difference likely to be on average?

We can figure out what would happen if we did this same randomization a few of times.

```
est_ate(outcomes, randomize())
```

```
## [1] -1.9
```

```
est_ate(outcomes, randomize())
```

```
## [1] -8.6
```

```
est_ate(outcomes, randomize())
```

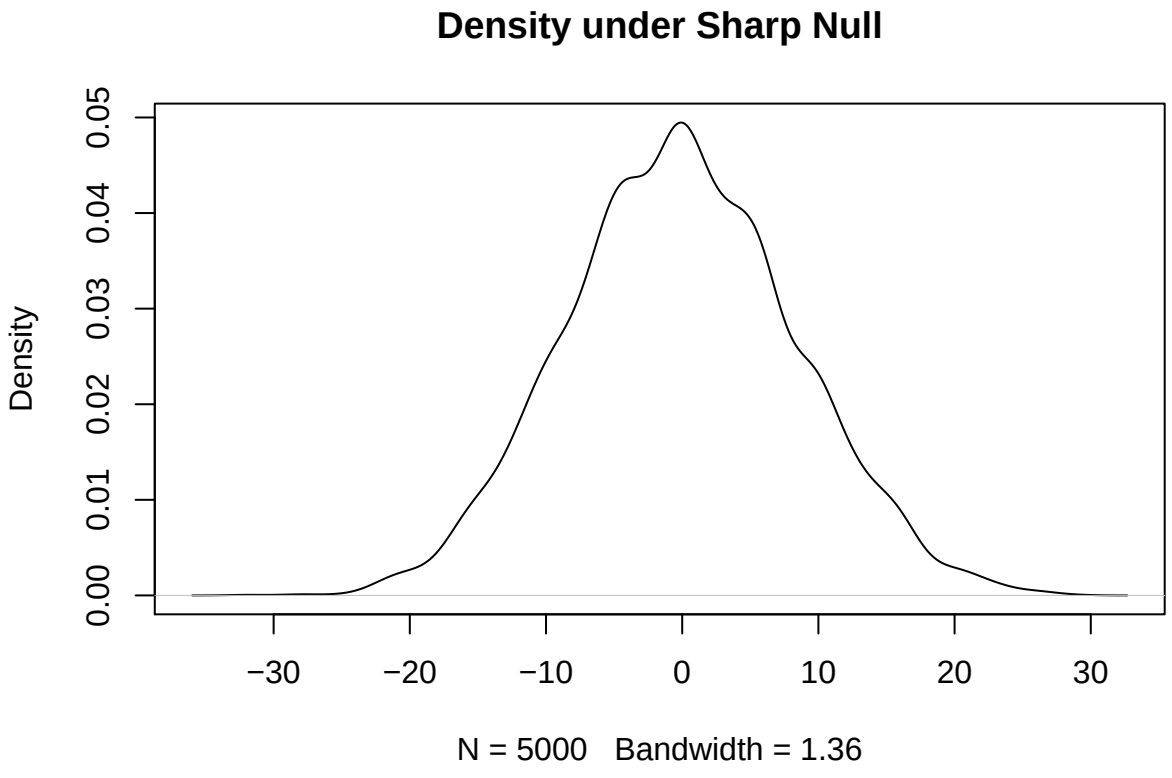
```
## [1] -5.5
```

So, what we're seeing is that there are some slightly different outcomes to different trement regimes. We can do this, a bunch more times using the `replicate` function. Specifically, do this 5,000 to get a sense of the distribution

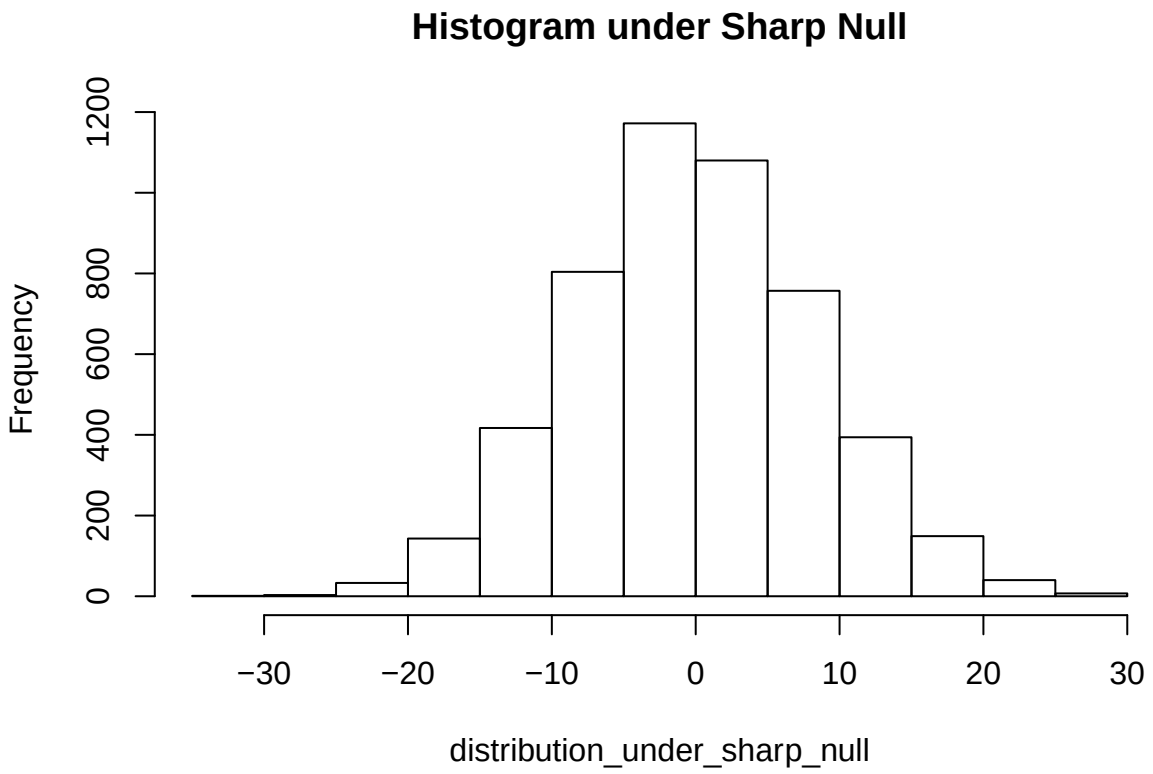
```
distribution_under_sharp_null <- replicate(5000, est_ate(outcomes, randomize()))
```

What does this look like if we examine it a little more?

```
plot(density(distribution_under_sharp_null),
     main = "Density under Sharp Null")
```

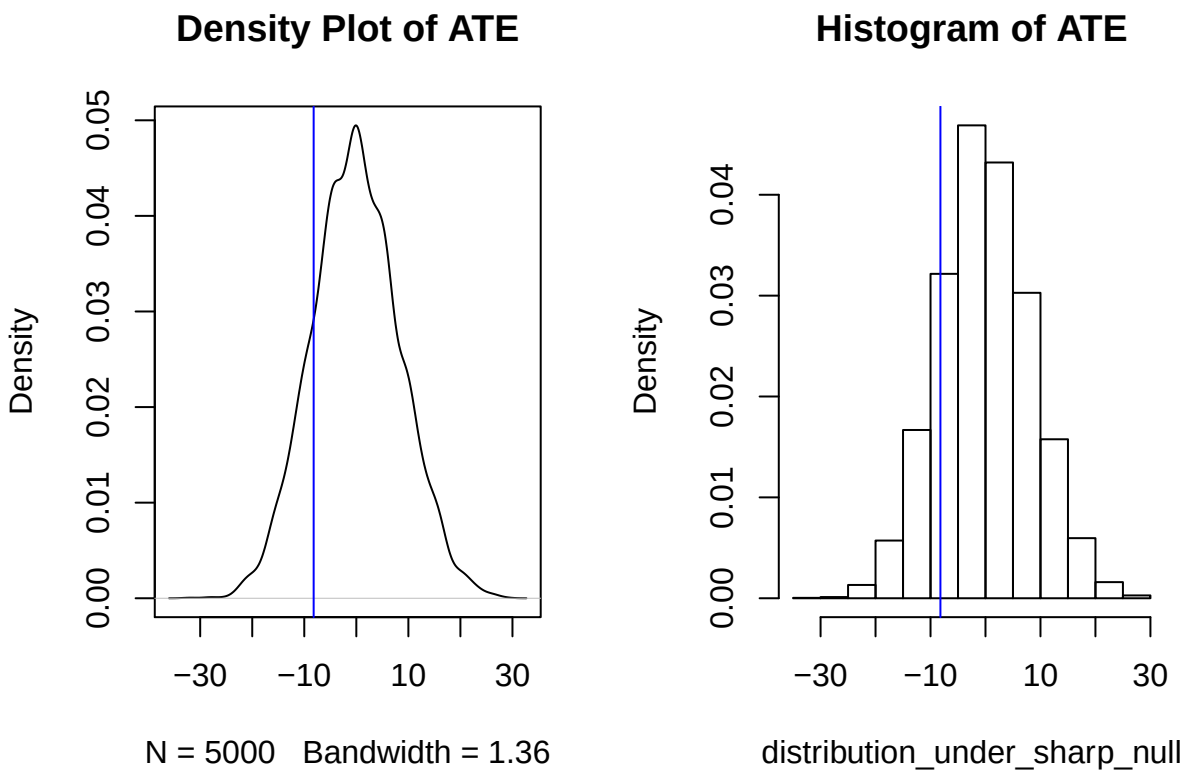



```
hist(distribution_under_sharp_null,
     main = "Histogram under Sharp Null")
```



How big was our observed difference?

```
par(mfrow = c(1,2))
plot(density(distribution_under_sharp_null),
     main = "Density Plot of ATE")
abline(v = ate, col = "blue")
hist(distribution_under_sharp_null,
     main = "Histogram of ATE",
     freq = FALSE)
abline(v = ate, col = "blue")
```



As it turns out, that was pretty similar to what we saw in our draw! In fact, what we've got here is pretty likely to turn up by chance.

Following David B. statment, we can really easily draw a probability of seeing an ATE of a given size under the repeated randomization regime.

```
m <- mean(ate <= distribution_under_sharp_null) #p-value
m
```

```
## [1] 0.8328
```

And so we we see that there is a 0.8328 probability of observing an ATE of this size, given the repreated randomization regime, under the sharp null hypothesis.

#Simulate an experiment with a large effect

We have seen that when there is no effect, our Randomization Inference regime does a good job at assigning a high probability of observing an effect size equal to or larger than the ATE we calculate from our particular randomization.

Now, let's show that when there is a big effect, our Randomization Inference regime does a good job at assigning a low probability of observing an effect size equal to or larger that the ATE we calculate in our regression.

```
po_treatment <- po_control + 25 # this is a big effect!
po_control
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 51 52 53
## [24] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

```
po_treatment
```

```
##  [1] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 76 77 78
## [24] 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
```

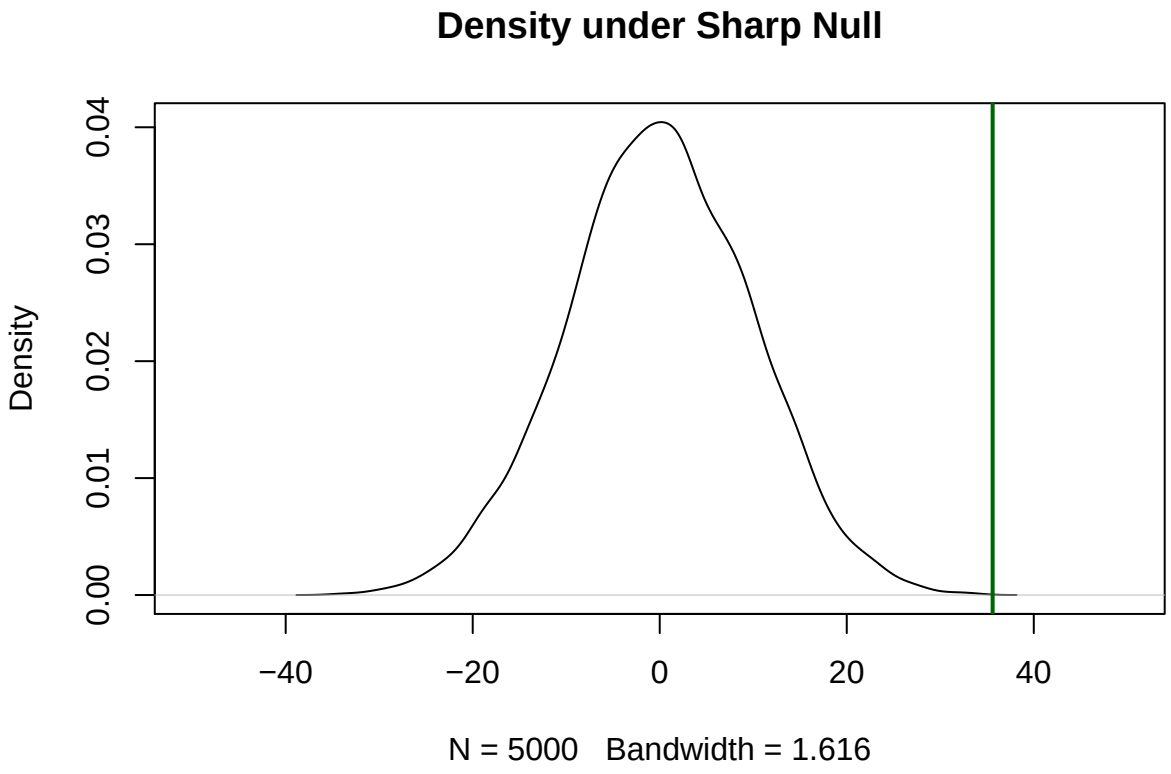
```r
# Randomize
treatment <- randomize()
outcomes <- po_treatment * treatment + po_control*(1-treatment)
outcomes
```

```
##  [1] 26 27  3  4 30 31  7  8  9 10 36 12 38 39 15 16 17 18 19 45 51 77 53
## [24] 79 55 56 57 83 59 85 86 87 88 89 90 91 67 68 94 95
```

```r
# Estimate ate
ate <- est_ate(outcomes, treatment)
ate
```

```
## [1] 35.6
```

```r
# What is the uncertainty?
distribution_under_sharp_null <- replicate(5000, est_ate(outcomes,
                                                         randomize()))
plot(density(distribution_under_sharp_null),
     xlim = c(-50, 50),
     main = "Density under Sharp Null")
abline(v=ate, col = "darkgreen", lwd = 2)
```




```r
mean(ate < distribution_under_sharp_null) #p-value
```

```
## [1] 0
```

Statistical power

To get a sense for how power increases or decreases with sample size and effect size, here we're going to wrap *everything* that we've done before in another function that will simulate our entire study.

As we note in the lecture, there is a third component to power that we are going to leave for next week: the variation in outcomes. (*Preview: if we decrease variation in outcomes either by including pre-treatment covariates, or explicitly designing our sampling to reduce this variation by blocking, we will increase our power.*)

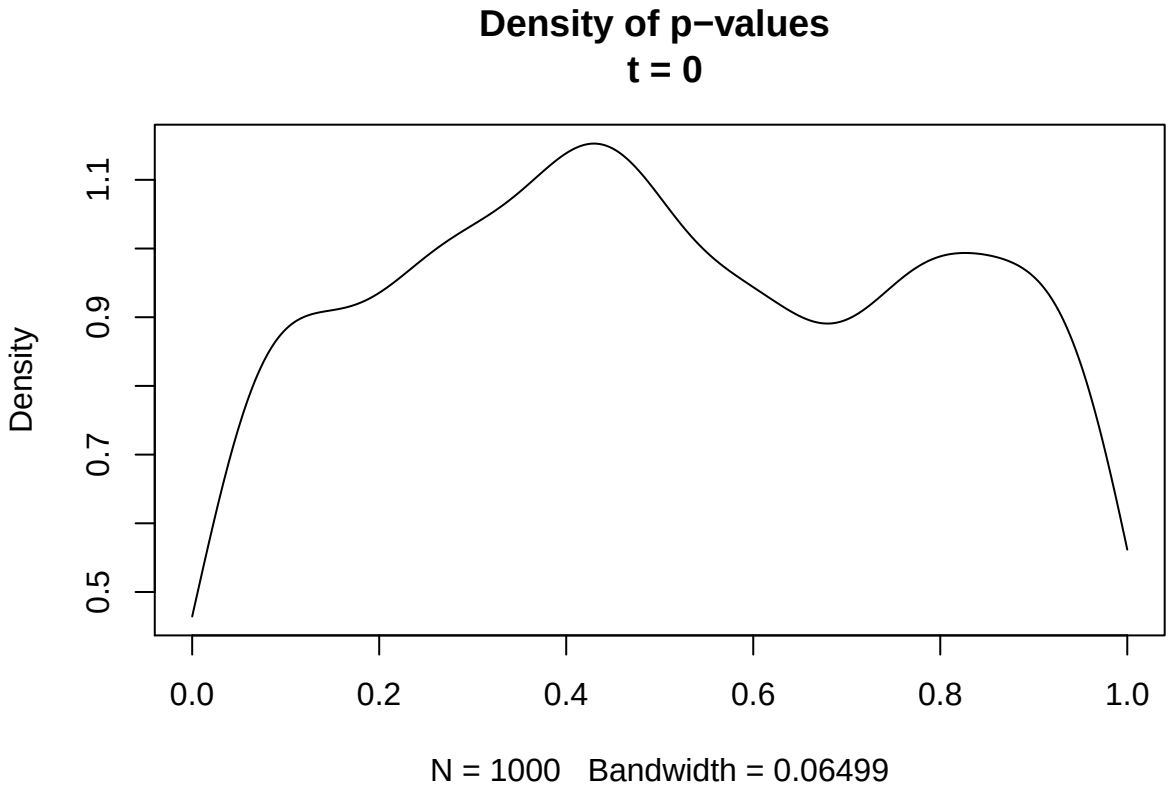
```
# Function to simulate a study of a given
# treatment effect and get the p-value

simulate_study <- function(treatment_effect_size) {
  po_control <- c(seq(from = 1,  to = 20),
                  seq(from = 51, to = 70) )
  po_treatment <- po_control + treatment_effect_size
  treatment <- randomize()
  outcomes <- po_treatment * treatment + po_control * (1- treatment)
  ate <- est_ate(outcomes, treatment)
  distribution_under_sharp_null <- replicate(1000, est_ate(outcomes, randomize()) )
  return(mean(ate < distribution_under_sharp_null))
}
```

With that function in hand, we can run with our simulations!

```
simulate_study(0) # p-value for no effect
```

```
## [1] 0.734
```

```
p_values <- replicate(1000, simulate_study(0)) # distribution of pvalues
plot(density(p_values, from = 0, to = 1), xlim = c(0,1),
     main = "Density of p-values \n t = 0") # uniform distribution
```



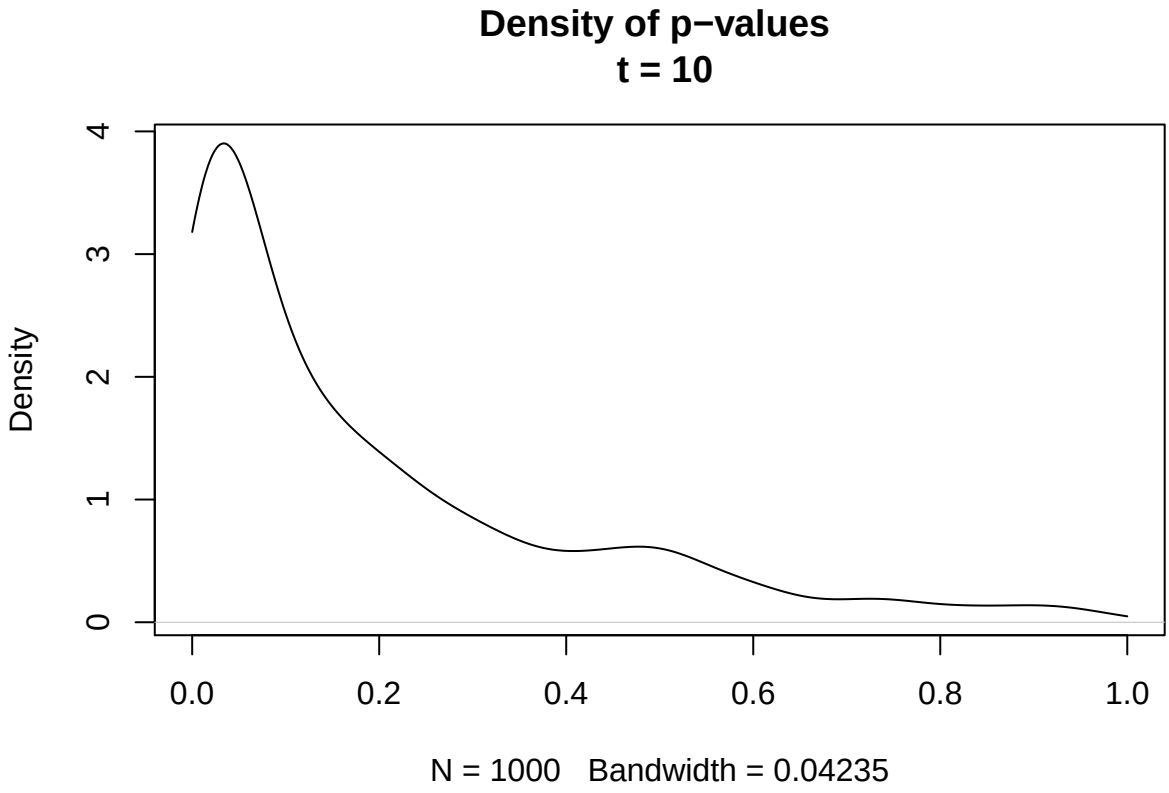

```
# how often is p_value under 0.05 when there is no effect?
```

That line is kind of warbeling around as a result of statistical noise, but more or less, we have a horizontal line – or at the very least, a line that would not be very informative if we had to place a bet about where on the range the p-value is most likely to lie.

In fact, if we look at the average p-value from this distribution, we see that it is 0.041. This means that the *average* probability of seeing a treatment size of size $\hat{ATE}$ under the sharp null hypothesis, across all 5,000 of our replications is 0.041. That is really, *really* likely.

What happens if we incrase the effect size in our simulated study? Now, we're moving from a simulation that supposes there is no effect of soy on estrogen levels to a simulation that supposes there is a 10 unit increase in the estrogen levels of subjects who are treated with soy.

```
p_values <- replicate(1000, simulate_study(10))
plot(density(p_values, from = 0, to = 1),
     main = "Density of p-values \n t = 10")
```



In the proceeding plot, we have quite a bit more information about the distribution of the p-values (which are themselves the result of a distribution... stay with us here). In this case, the average p-value from the 1,000 replications of the experiment under the sharp-null is 0.193771. This is *lower* than before – which is sensible given we have increased the simulated effect. With a larger effect, we should think that it is less likely to see an effect size of $\hat{ATE}$ due only to random chance, and as such, the mean p-value should be smaller to reflect this state of the world.

A little more about the average p-values: What does the "average" p-value look like under this regime? And how often do we see p-values that are smaller than 0.05? This is the RI relative to a classical hypothesis test that relies on strong statistical assumptions about sampling, distributions and the law of large numbers.

```
mean(p_values)
```

```
## [1] 0.193771
```

```r
mean(p_values < 0.05) # somewhat likely to detect this effect
```

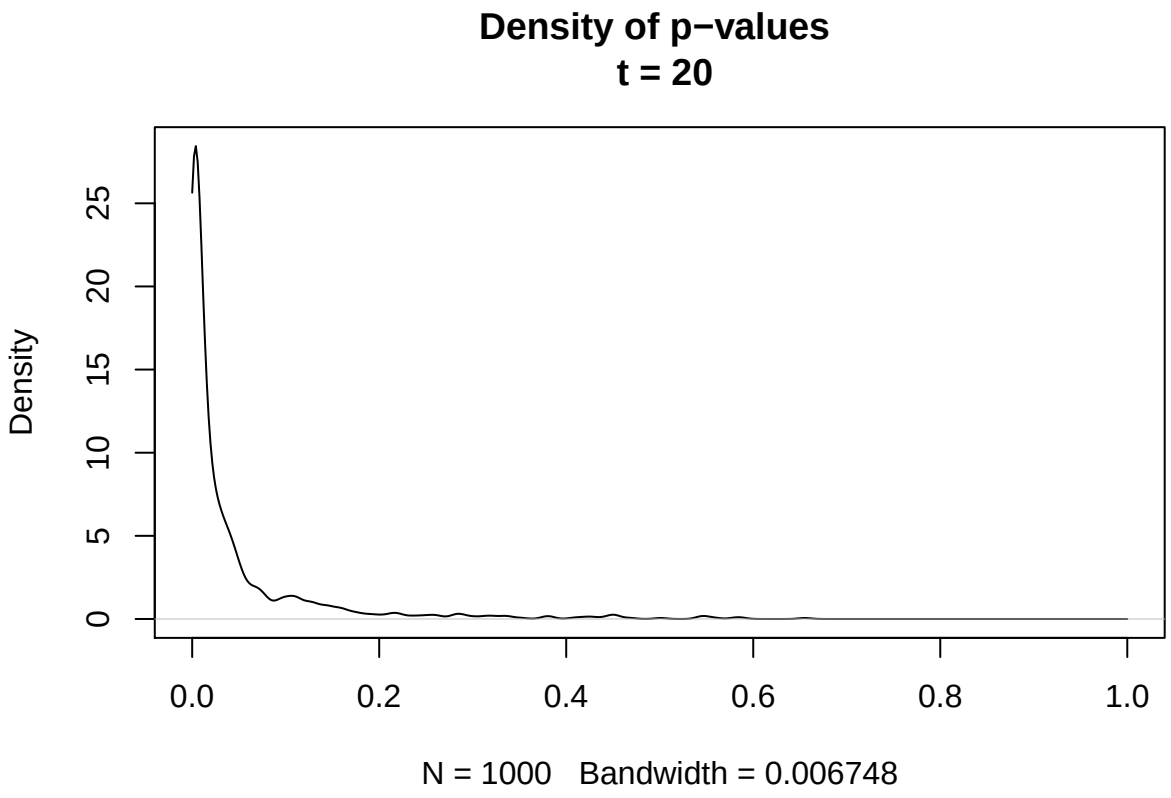
```
## [1] 0.358
```

```r
table(p_values < 0.05)
```

```
##
## FALSE  TRUE
##   642   358
```

Now, do it again, but with a simulated treatment effect size of twenty: $\tau_i = 20$ for everyone in the study. What then are the chances that we see an effect size as large as we do for a single randomization, just by chance?

```r
p_values <- replicate(1000, simulate_study(20))
plot(density(p_values, from = 0, to = 1),
     main = "Density of p-values \n t = 20")
```




```r
# What do those p-values look like?
mean(p_values)
```

```
## [1] 0.044564
```

```r
mean(p_values < 0.05)   # very likely to
```

```
## [1] 0.78
```

```r
table(p_values < 0.05)  # detect this effect
```

```
##
## FALSE  TRUE
##   220   780
```

#How does power behave?

In this section we're going to talk about power, which is the probability that the test we set up will *correctly* reject the null hypothesis. Said another way, what is the probability that our p-value is small, given our p-value *should* be small? Or, said even another way, how much **power** does our test have to detect a *true* effect.

Let's examine how power behaves in a very simple OLS regression. To do so, let's write two more short functions.

```
# First: Simulate the p-values in a regression
simulate_study_lm <- function(baseline, effect_size, sample_size) {
  control_units <- rbinom(sample_size, 1, baseline)
  treatment_units <- rbinom(sample_size, 1, baseline + effect_size)
  all_units <- c(control_units, treatment_units)
  treatment_vector <- c(rep(0, sample_size), rep(1, sample_size))
  p_value <- summary(lm(all_units ~ treatment_vector))$coefficients[2, 4]
  effect_detected <- p_value < 0.05
  return(effect_detected)
}
# Second:
get_power <- function(baseline, effect_size, sample_size) {
  return(mean(replicate(2000,
                        simulate_study_lm(baseline,
                                          effect_size,
                                          sample_size))) )
}
```

Functions set, let's roll!

First: What is the effect on our power if we incrase the effect size of the treatment that we are administering? In an experiment, we might be able to accomplish this incrased effect size by increasing the *dosage* or the intensity of the treatment that we're administering. However, we should note two things:

1. The actual treatment effect (τ_i) is an effect that is a parameter of the *real* world, something that is onobserved
2. We can't actually manipulate τ_i per unit application – but rather, we can increase the units applied.

```
#Increasing effect size
get_power(baseline = .1, effect_size = .05, sample_size = 100)
```

```
## [1] 0.201
```

```
get_power(baseline = .1, effect_size = .10, sample_size = 100)
```

```
## [1] 0.5265
```

```
get_power(baseline = .1, effect_size = .15, sample_size = 100)
```

```
## [1] 0.8245
```

```
get_power(baseline = .1, effect_size = .2, 100)
```

```
## [1] 0.959
```

```
get_power(baseline = .1, effect_size = .25, 100)
```

```
## [1] 0.994
```

Second: let's work with the other lever that we actually *can* pull, mainipulating the effect size. Even if we can't move τ, we can enroll 1x, 5x, or 50x the number of people in our study.

```
# Increasing sample size
get_power(baseline = .1, effect_size = .05, sample_size = 100)
```

```
## [1] 0.184
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 200)
```

```
## [1] 0.3335
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 300)
```

```
## [1] 0.448
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 400)
```

```
## [1] 0.587
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 500)
```

```
## [1] 0.668
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 1000)
```

```
## [1] 0.934
```

```
get_power(baseline = .1, effect_size = .05, sample_size = 5000)
```

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

That is a little bit less rosy, isn't it...

... this leads us to the notion of the concentrated test as described in lecture. If we can only afford 1e6 soybeans, how should we allocate them among the population? How many subjects should we enroll in our trial, and how many soybeans should we provide to each subject?

#Confidence interval

```
summary(lm(outcomes ~ treatment))
```

```
##
## Call:
## lm(formula = outcomes ~ treatment)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -39.80 -22.45   0.00  23.45  37.80
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   30.200      5.759   5.244 6.18e-06 ***
## treatment     35.600      8.144   4.371 9.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.76 on 38 degrees of freedom
## Multiple R-squared:  0.3346, Adjusted R-squared:  0.3171
## F-statistic: 19.11 on 1 and 38 DF,  p-value: 9.255e-05
```

```
estimate_in_confidence_interval <- function() {
  true_effect <- 25
  # Simulate outcomes
```

```
  po_control <- c(seq(from = 1,  to = 20),
                  seq(from = 51, to = 70))
  po_treatment <- po_control + true_effect
  treatment <- randomize()
  outcomes <- po_treatment * treatment + po_control*(1 - treatment)
  # Run regression
  regression <- summary(lm(outcomes ~ treatment))
  estimate <- regression$coefficients[2, 1]
  standard_error <- regression$coefficients[2, 2]
  lower_bound <- estimate - standard_error * 1.96
  upper_bound <- estimate + standard_error * 1.96
  # Is estimate in CI?
  estimate_in_ci <- lower_bound < true_effect & upper_bound > true_effect
  return(estimate_in_ci)
}
estimate_in_confidence_interval()
```

```
## [1] TRUE
```

```
mean(replicate(1000, estimate_in_confidence_interval()))
```

```
## [1] 0.939
```

That's all folks!