Randomized Experiments: Randomization Inference

Eduardo Fe

01/07/2019

Introduction.

For this exercise, continue to work with the data from Olken's paper. This time, however, we are going to analysise those data from the perspective of Fisher's randomization inference. There are a number of reasons why we might want to do this. First, the goal of Olken's project is to learn about corruption in general; however, rural Indonesia seems like a very specifica case study; it is unclear to which extent the conclusions obtained with these data might extrapolate to other settings. Second, the sample in the data is relatively small. In Neymam's approach, confidence intervals and tests rely on asymptotic approximations, all of which assume a very large sample size. Abundant research in statistics shows that the properties of these tests might be less than ideal even with moderate samples. In contrast to this, randomization inference focuses on the randomization distribution of the treatment in the specific application; because this distribution is known in a randomized experiment, we can then characterise what the exact distribution of any tests would be under any *sharp* null hypothesis.

In other words, the inferences drawn from these tests is very accurate, but out conclusions will be restricted to the sample we have (which we see as our population).

Just to recap, we are going to use real data from Benjamin Olken's paper "Monitoring Corruption: Evidence from a Field Experiment in Indonesia", appeared in 2007 in the Journal of Political Economy. The paper and the data can be downloaded from Olken's website at MIT,

https://economics.mit.edu/faculty/bolken/published.

For this exercise, we are going to focus on the subset of villages where the missing expenditure in major items in the road project was positive (that is, the reported expenditure exceed what was estimated to be the actual cost of the road project).

As a first step, let's load the original data set `jperoaddata.dta`.

Once you've done that, rather than dropping all the observations for which missing expenditure is negative, we are going to create a new data set which will include only those observations with positive missing expenditure; the latter will be our working file.

To do this, we can use the `subset(dataframe, condition)` function. Here `dataframe` is the data set we want to extract data from; `condition` will tell `R` the rules to follow to extract the subsample. In our specific application, we run the following command,

```
datos2 <- subset(datos, lndiffeall4>0)
```

Exerscise. Your first task now is to repeat the analysis in Lab 2, but using the new subset. Specifically, obtain estimates of the causal effect of the intervention on each of the missing expenditure variables.

Now, we are going to re-run the analysis using randomization inference. The key library to do this is `ri`, which can be installed and downloaded in the usual manner.

You can get some back help regarding how to use the library by typing `help(ri)`. In the help panel, if you scroll to the end, you will see a blue link "Index"; clicking on that link will take you to another panel with the decriptor for each command of the library.

We start by testing the shap null hypothesis that the treatment had no effect. To do this with the `ri` package, we need to break the task into a few steps. First, we need to compute each unit's likelihood of treatment.

This first step is important for situations when randomization occuss at cluster level or when there is a degree of stratification. To calculate this probability, we simply do,

```
probs <- genprobexact(datos2$audit)
```

Next, we need to tell R to, please, calculate all the permutations of the assignment; this is achieved with the command

```
perms <- genperms(datos2$audit)
```

```
## Too many permutations to use exact method.
## Defaulting to approximate method.
## Increase maxiter to at least 6.36636723400102e+108 to perform exact estimation.
```

R displays a message saying that there are too many permutations. This is not an error. Remember that, for instance, that with 20 observations, 10 of which are allocated to treatment, we have over 180,000 possible permutations. Our data has 369 observations, with 44 percent of these allocated to the treatment, so the number of possible permutations is too large even for the computer to bother with. This also implies that the p-values of our test will be *approximately* exact.

Next, let's compute the value of the statistic on which randomization inference will be based; the package ri only allows you to work with the difference in means; to check this is the true, note that

```
mean1 <- datos2 %>% filter(audit==1) %>% summarize(mean = mean(lndiffeall4))
mean0 <- datos2 %>% filter(audit==0) %>% summarize(mean = mean(lndiffeall4))
mean1 - mean0
```

```
##          mean
## 1 -0.05850137
```

while the command in ri to compute this difference in means is estate,

```
ateHat <- estate(datos2$lndiffeall4, datos2$audit, prob=probs)
ateHat
```

```
## [1] -0.05850137
```

Remember that you could use many other statistics (sums of ranks, Kolmogorov-Smirnov, difference in medians, etc) if you were able to write your own routine for randomziation inference (let's leave that for another course).

The last step, is to compute the p-value of the test statistics (which in this case is the difference in means) under the sharp null hypothesis. For this we need to tell R what is the value we expect to observe for the difference in potential outcomes under the sharp null, so that R can compute the values of the counterfactual potential outcomes, just as we did with the Honey example in the classroom. Let's start by testing the sharp null that there is no effect,

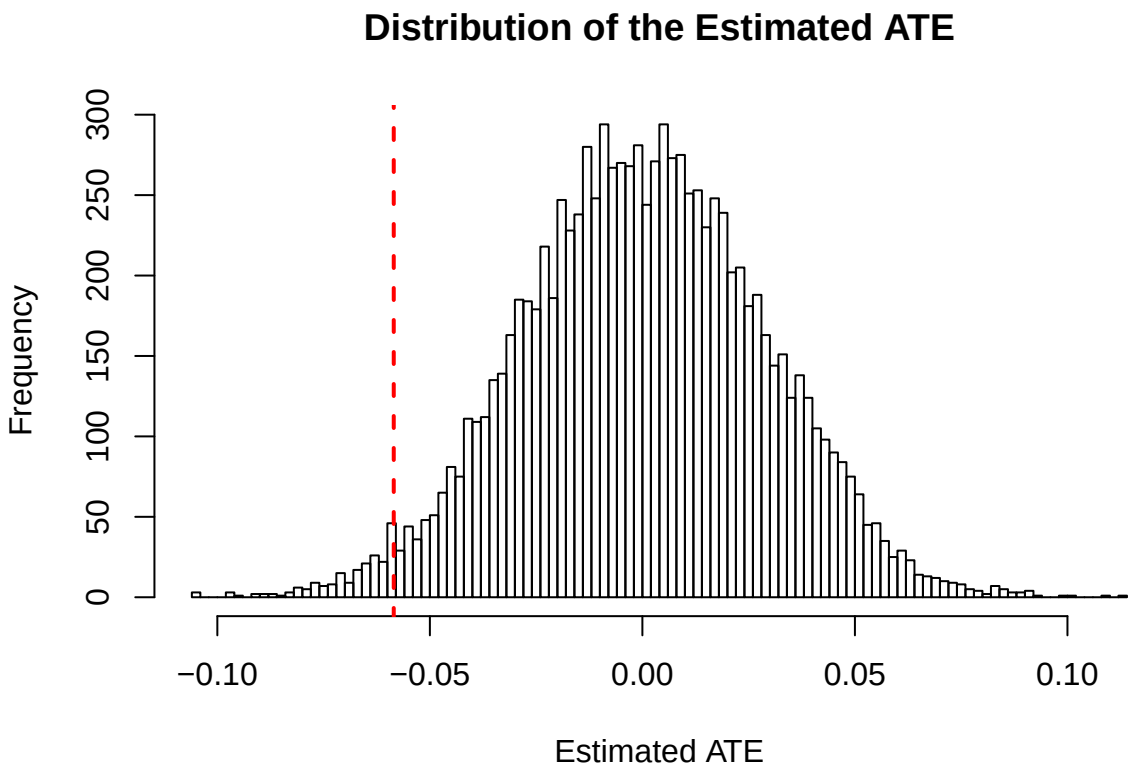
```
Ys <- genouts(datos2$lndiffeall4, datos2$audit, ate = 0)
```

Note that in the latter command, ate equals the value of the difference in potential outcomes under the sharp null hypothesis (which in this case we set to 0) and not the value of the test. Then comes the magic: compute the randomization distribution of the test,

```
distout <- gendist(Ys, perms, prob=probs)
```

And we are almost done; we only now need to compute the p-value (how extreme is the observed ateHat under the sharp null hypothesis)

```
dispdist(distout, ate =ateHat)
```




```
## $two.tailed.p.value
## [1] 0.04
##
## $two.tailed.p.value.abs
## [1] 0.0373
##
## $greater.p.value
## [1] 0.98
##
## $lesser.p.value
## [1] 0.02
##
## $quantile
##        2.5%       97.5%
## -0.05539101  0.05452449
##
## $sd
## [1] 0.02827766
##
## $exp.val
## [1] 0.0003600601
```

The number we need to focus on is the `$two taylied p.value` which equals 0.034. Remember that anything below 0.05 will be an invitation to reject the sharp null hypothesis that the treatment effect is 0. So, it looks like we now have a significant effect of auditing on corruption. THe graph produced by `dispdist` shows, first, the randomization distribution of the test statistic (the difference in sample means) under the sharp null hypothesis (no treatment effect in this case). The red line, is the value of `ateHat`, which you can already

see is located on the left tail of the distribution -so, under the sharp null hypothesis, this estimated value is unlikely.

Estimation, next... During the lectures we explained that estimation in Fisher's framework involved inverting our test, that is, to find the sharp null for which the test statistics yields the largest p-value. And yet, if use the difference in means as our test statistics, it turns that that the value of the sharp null hypothesis that returns the largets p-value is... the estimated value of the difference in means (a property not shared by other statistics). To see this, let's compute the p-values under the sharp null hypothesis that the treatment equals (a) the estimated difference in means (b) the estimated difference in means plus/minus a very small number...

```
Ys <- genouts(datos2$lndiffeall4, datos2$audit, ate = ateHat)
distout <- gendist(Ys, perms, prob=probs)
dispdist(distout, ate =ateHat, display.plot = FALSE)
```

```
## $two.tailed.p.value
## [1] 0.9908
##
## $two.tailed.p.value.abs
## [1] 0.4955
##
## $greater.p.value
## [1] 0.5046
##
## $lesser.p.value
## [1] 0.4954
##
## $quantile
##         2.5%        97.5%
## -0.113659778 -0.004126884
##
## $sd
## [1] 0.02810028
##
## $exp.val
## [1] -0.05813964
```

```
Ys <- genouts(datos2$lndiffeall4, datos2$audit, ate = ateHat-0.003)
distout <- gendist(Ys, perms, prob=probs,)
dispdist(distout, ate =ateHat, display.plot = FALSE)
```

```
## $two.tailed.p.value
## [1] 0.927
##
## $two.tailed.p.value.abs
## [1] 0.5365
##
## $greater.p.value
## [1] 0.4635
##
## $lesser.p.value
## [1] 0.5365
##
## $quantile
##         2.5%        97.5%
## -0.116637330 -0.006857712
```

```
##
## $sd
## [1] 0.02810009
##
## $exp.val
## [1] -0.06113955
```

```
Ys <- genouts(datos2$lndiffeall4, datos2$audit, ate = ateHat+0.003)
distout <- gendist(Ys, perms, prob=probs)
dispdist(distout, ate =ateHat, display.plot = FALSE)
```

```
## $two.tailed.p.value
## [1] 0.9078
##
## $two.tailed.p.value.abs
## [1] 0.454
##
## $greater.p.value
## [1] 0.5461
##
## $lesser.p.value
## [1] 0.4539
##
## $quantile
##         2.5%        97.5%
## -0.110646113 -0.001125017
##
## $sd
## [1] 0.02810135
##
## $exp.val
## [1] -0.05513972
```

In other words, we are done: We find that auditing lead to a significant reduction in the corruption of about 5.8 percentage points. This is qualitatively different to the findings from Neyman's approach... Well, not really.

Remember that in Olken's work, randomization happened at sub-district level; this is why we used cluster-robust standard errors. We need to do the same here. Fortunatelly this is a simple task to execute; when computing the probability of treatment and the permutation matrix, we only need to tell `R` what the clustering variable is (note, however, that clustering does not affect the value of the test statistics)

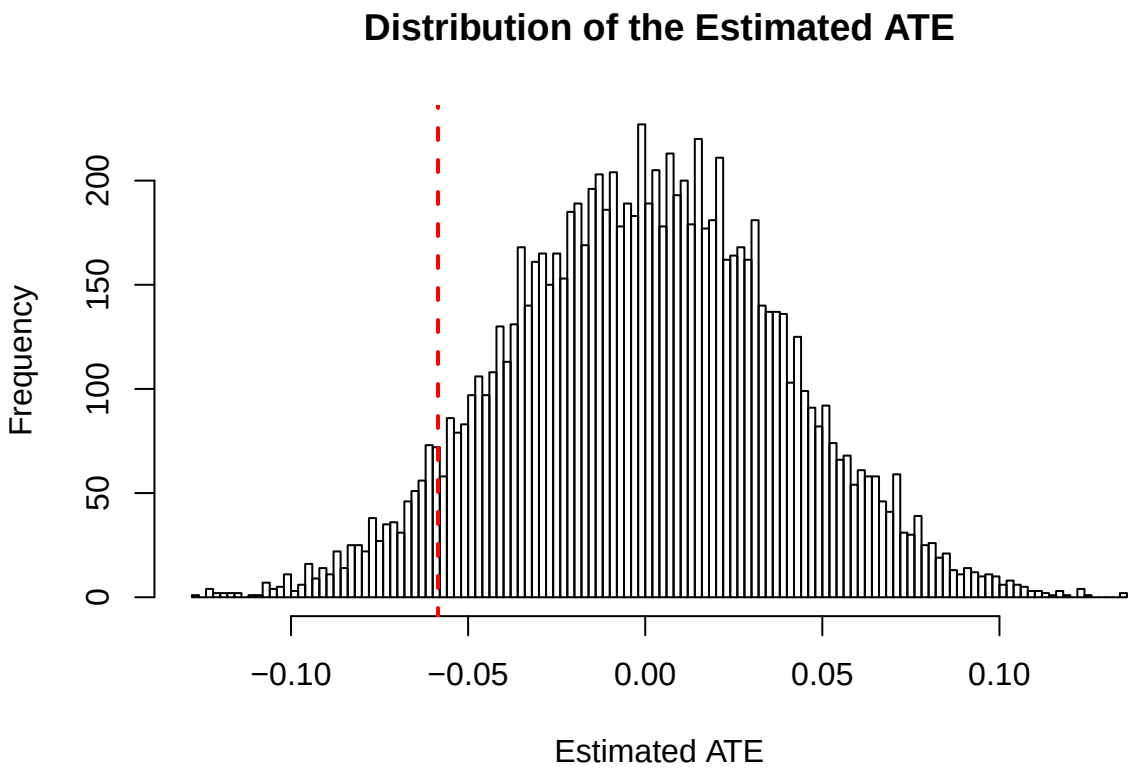
```
probs <- genprobexact(datos2$audit, clustvar = datos2$kecnum)
perms <- genperms(datos2$audit, clustvar= datos2$kecnum)
```

```
## Too many permutations to use exact method.
## Defaulting to approximate method.
## Increase maxiter to at least 2.59042077365168e+38 to perform exact estimation.
```

```
ateHat <- estate(datos2$lndiffeall4, datos2$audit, prob=probs)
ateHat
```

```
## [1] -0.05850137
```

```
Ys <- genouts(datos2$lndiffeall4, datos2$audit, ate = 0)
distout <- gendist(Ys, perms, prob=probs)
dispdist(distout, ate =ateHat)
```




```
## $two.tailed.p.value
## [1] 0.1306
##
## $two.tailed.p.value.abs
## [1] 0.1338
##
## $greater.p.value
## [1] 0.9347
##
## $lesser.p.value
## [1] 0.0653
##
## $quantile
##        2.5%       97.5%
## -0.07562742  0.07626584
##
## $sd
## [1] 0.03869586
##
## $exp.val
## [1] 0.0002006799
```

A very different story: the effect is not significant any longer! It is interesting to compare the randomization distribution under clustering and that without clustering; the former is wider because clustering increases the uncertainty surrounding our estimates.

Exercise. Now that you have this technique mastered, you can do a number of things. First, you can test the

balance of the pre-treatment covariates in the dataset `jperandomisation.dta`, as we did in the second lab. Second, you can use Fisher's randomization inference in order to estimate the causal effect of the intervention on other outcomes (such as `lndiffeall3mat`, etc).