Recitation 12: Weighting

Seamus Wagner

November 16, 2021

Building off of last week, we can write a function from the demo to create some diff-in-diff data.

```
library(data.table)
rm(list=ls())
set.seed(5642313)
# using a function that takes numbe rof units, mean and sd for baseline, and the populatio nate
sim_did <- function(n_units, base_trend_mean, base_trend_sd, pop_ate){
  # Using CJ to get the base structure
  panel_data <- CJ(i = 1:n_units, t = 1:2)
  # Adding the trends for time 1 then 2
  panel_data[t == 1, trend := rnorm(.N)]
  panel_data[t == 2, trend := rnorm(.N, base_trend_mean, base_trend_sd)]
  # Cumulatively sum the trend increments to get Y0
  panel_data[, Y0 := cumsum(trend), i]
  # Add some treated observations
  panel_data[, mu := rnorm(.N, pop_ate)]
  panel_data[, Y1 := Y0 + mu]
  # Create Y, remember no units should be treated at t1
  panel_data[t == 1, Y := Y0]
  # now we need to treat some units, we need to select in some way (hopefully randomized)
  treated_units <- sample(n_units, floor(n_units / 2))
  # what is D here?
  panel_data[, D := as.numeric(i %in% treated_units)]
  # Now that we have some treated units, we can invoke SUTVA, but only for t2 and t3 if we were to incl
  panel_data[t == 2, Y := D * Y1 + (1 - D) * Y0]
panel_data

}
dt <- sim_did(10,1,.1,1)
```

Dcast and wide/long data

We can also turn this into a wide data set. I typically work in wide formats, though that is just preference of the teams I have been on, not an endorsement that it is better. I typically work in wide to do all manipulation and then shift to long when I am merging after most manipulating is done.

What dcast is doing here is to cast the data out into columns based on the conditions in the function. Here, we are choosing our data, then setting our row variable, i in this case. Then, we are casting out the variables

in the value.var function, so we get x number of columns based on the value on the RHS of thr ~. For each i, we now have t times as many columns for trend, Y0, etc. In our case, it is two. We now have a Y0 for time 1 and time 2 and so on.

```
wide_data <- dcast(dt, i ~ t, value.var = c("trend", "Y0", "Y1", "mu", "D", "Y"))
wide_data
```

```
##      i     trend_1   trend_2        Y0_1       Y0_2       Y1_1      Y1_2
##  1:  1 -0.41567515 1.0595406 -0.41567515  0.6438655  1.0861549 1.8844147
##  2:  2 -1.64163635 0.9837006 -1.64163635 -0.6579357 -0.1968274 1.3821610
##  3:  3 -1.30299169 1.1481056 -1.30299169 -0.1548861 -0.9409752 0.2704935
##  4:  4  0.98865909 1.0168707  0.98865909  2.0055298  1.6030667 2.6872858
##  5:  5  0.92018526 1.0173502  0.92018526  1.9375355  2.9172885 2.0553855
##  6:  6  1.32495914 1.0644298  1.32495914  2.3893889  2.0105899 5.5245782
##  7:  7  0.24439290 1.0537880  0.24439290  1.2981809  1.4705261 2.5590339
##  8:  8  0.50499266 0.9501454  0.50499266  1.4551380  2.3152641 3.3146560
##  9:  9 -0.55723589 1.0035390 -0.55723589  0.4463031  0.4706147 2.1139513
## 10: 10  0.06927898 0.9958564  0.06927898  1.0651353  1.3751909 1.3585154
##          mu_1      mu_2 D_1 D_2         Y_1        Y_2
##  1: 1.5018301 1.2405493   1   1 -0.41567515  1.8844147
##  2: 1.4448090 2.0400967   0   0 -1.64163635 -0.6579357
##  3: 0.3620165 0.4253796   1   1 -1.30299169  0.2704935
##  4: 0.6144076 0.6817560   0   0  0.98865909  2.0055298
##  5: 1.9971032 0.1178500   1   1  0.92018526  2.0553855
##  6: 0.6856308 3.1351893   0   0  1.32495914  2.3893889
##  7: 1.2261332 1.2608530   1   1  0.24439290  2.5590339
##  8: 1.8102714 1.8595180   1   1  0.50499266  3.3146560
##  9: 1.0278505 1.6676482   0   0 -0.55723589  0.4463031
## 10: 1.3059119 0.2933800   0   0  0.06927898  1.0651353
```

You can also do the same thing in a number of ways, dcast is kind of the default in our circles at least. It is easier enough that I don't ever use the tidyverse equivalent as you have to do all the aggregation within the pipes and it is way less readable code.

If we wanted to check them from a model perspective, we can use the following three, and they should all be the same and if our seed is good with such a small n, we should be close to 1.

```
coef(lm(Y ~ D * t, dt))["D:t"]
```

```
##      D:t
## 1.013737
```

```
coef(lm(Y_2 -Y_1 ~ D_2, wide_data))["D_2"]
```

```
##      D_2
## 1.013737
```

```
coef(lm(Y ~ D * t + factor(i) + factor(t), dt))["D:t"]
```

```
##      D:t
## 1.013737
```