Lab 12: Declare Design: Answers

Eric Parajon with code adapted from Colin Case

Pre-Registration, and Registred Reports

In recent years, there has been a shift within political science, particularly in political behavior, towards the use of pre-registration. Much of this motivation can be stemmed from the replication crisis and publication bias that has existed within the social sciences more broadly. That is to say, statistically significant results and hypothesis testing has driven hypothesis and theory generation. By moving towards pre-registration, where researchers develop hypotheses and an analysis plan before analyzing or collecting data, there can be an increased distinctiveness between hypothesis generation and hypothesis testing (as well as an increased on the former *a priori*) that helps to improve research credibility.

A number of political science journals have embraced pre-registration and registered reports. The Journal of Experimental Political Science accepts the registered report format where you receive conditional acceptance before collecting data. The Journal of Politics recently made the requirement for all experimental work to be pre-registered. JOP also recently announced it would be accepting registered reports starting in January 2023. Given this, it is likely some other journals will follow suit and this will become a more common practice across the discipline.

As a result of this shift, it is important to be familiar with tools that can help you develop strong pre-analysis plans, especially for registered reports (it is also just a good idea regardless of this shift)! In particular, simulating results, although not required, is a good practice for helping to diagnose potential issues with the research design. It also makes it easy to see what an eventual paper will look like with the results and eliminates some the arbitrariness that can be associated with unclear pre-analysis plans. Plus, you can then have all your code written *before* collecting the data and just click run once you have the data! Today, we are going to look at one of the more common tools, `DeclareDesign`, for describing research designs and simulating them without coding simulations from scratch. For our lab today, we are going to be producing simulated results for two separate analyses: one just simulating a model and the other simulating random assignment that is often associated with pre-analysis plans.

Declare Design Software

Let's start first by loading the `DeclareDesign` package. If you have not used this before, make sure to install the package. Within `DeclareDesign`, there are three subsequent packages that are related to different steps of the research process: `fabricatr` can be used to simulate data, `randomizr` can be used for random sampling and random assignment, and `estimatr` for design-based estimators. For our lab, we are going to work on two concurrent model simulations. In one, we will be simulating a model with observational data and in the second we will use random assignment to estimate population average treatment effects.

```
# Clear Environment
rm(list = ls())
# Load Packages
pacman::p_load(DeclareDesign,tidyverse,MASS)
# Set Seed for Reproducibility
set.seed(100)
```

Simulating Data

We are first going to work on simulating data to use in our observational analysis. To do this, we will use `fabricatr`. Note, `fabricatr` also lets you work with pre-existing data. Depending on the data structure might be more conducive (i.e. in Qualtrics, given a survey design, you can simulate survey results and use this pre-existing data versus hand coding everything).

`fabricatr` allows you to create common variable types, including assignment to treatment, count data, ordinal data (including "Likert scale" data, popular in surveys and survey experiments), categorical data (popular for modeling demographic characteristics). It also supports the creation of data with fixed intra-cluster correlations, so individual observations can be modeled as being part of groups or regions.

Let's first start by creating a *sample*, df1, with 500 observations. We are going to focus on the following variable characteristics to try and generate data that might look like our actual data from the real world:

X_1: Continuous value with mean of 0 and standard deviation of 1.\

X_2: Continuous value with mean of 4 and standard deviation of 2. X_1 and X_2 should have a covariance of 0.6.\

For our outcome variable, Y, we are going to assume a relationship between our independent variables of $.5 * X_1 + 3 * X_2$. Because we don't want a perfect relationship for our simulation, allow β_1 (the effect of X_1 on Y) to follow a normal distribution with a mean of and a standard deviation of 10 and β_2 (the effect of X_2 on Y) to follow a normal distribution with a mean of 3 and a standard deviation of 3. You can think about these as the true effect and standard deviation for the population.

Check the characteristics of your data after you simulate it. Do they roughly match the parameters we set above here?

```
df1 <- fabricate(
  N = 500, # Specify sample size. All subsequent variables need to have this
           # number of observations for fabricate to work.
  draw_multivariate(c(X_1, X_2) ~ mvrnorm( # Create variable names and simulate
    n = N, # Specify sample size
    mu = c(0, 4), # Specify means
    Sigma = matrix(c(1, 1.2, 1.2, 4), 2, 2))), # Specify covariance matrix.
                                               # Think carefully about the
                                               # structure of the matrix
                                               # and the values.
  Y = rnorm(500, mean = .5, sd = 10)*X_1 + rnorm(500, mean = 3, sd = 3)*X_2
)

# Summary statistics of X_1
mean(df1$X_1)
```

```
## [1] -0.0773127
```

```
sd(df1$X_1)
```

```
## [1] 1.035671
```

```
# Summary statistics of X_2
mean(df1$X_2)
```

```
## [1] 3.943323
```

```
sd(df1$X_2)
```

```
## [1] 2.008737
```

```r
# Correlation between X_1 and X_2
cor(df1$X_1, df1$X_2)
```

```
## [1] 0.5802619
```

`df1` will be our dataframe for the observational data. As mentioned above, we will look at a second design to estimate average treatment effects. However, we will create that data in the next section as we go through the steps to see a different way of using `DeclareDesign` following the potential outcomes framework.

Building Steps

Now that we have our data for the first model, we can start to build our research design. Research designs within `DeclareDesign` are constructed by building "steps." Almost all steps take a dataset as input and return a dataset as output. Each step of the research design process can be written as `declare_*` functions for the following steps: model, inquiry, sampling, assignment, measurement, and estimator. We are going to work through these steps and then simulate a design. We are going to work through two concurrent processes: one where we have pre-existing data that we simulated above and are specifying an observational model, and another where we are interested in the PATE of a treatment effect. It should be noted, you do not have to use all steps of the process. The use of certain steps and not others will depend on your research design.

Model

The model defines the structure of the world, both its size and background characteristics as well as how interventions in the world determine outcomes. The model defines the number of units in the population, any multilevel structure to the data, and its background characteristics. We can define the population in several ways. In some cases, you may start a design with data on the population. When that happens, we do not need to simulate it at this step. We can simply declare the data as our population.

Because we already built simulated data above, we are going to use df1 for the observational model, m1. Our model can be specified as:

$$Y = \beta_1 X_1 + \beta_2 X_2 + U$$

To specify the model, all you need to do is use the data argument of `declare_model`.

```r
# Declare Model 1, specify the data argument
m1 <- declare_model(data = df1)
```

For our second design, we will simulate data where we are interested in the PATE for a treatment assignment, Z, on an outcome, Y. In this design, we will define a potential outcomes framework where the PATE is 0.1. Allow this PATE to have a normal distribution and a standard deviation of 0.2. It is important to note, as we are declaring this model to create the simulated data, we are creating a *population* of 1,000,000 people, *not* a sample. This is important for later on in the process.

```r
# Declare Model 2
m2 <-
  declare_model(
    N = 1000000, # Specify population size
    # Define potential_outcomes and specify formula with uncertainly around PATE
    potential_outcomes(Y ~ Z*rnorm(N, mean  = (0.1), sd = .2))
  )
```

You will see this at the end of the lab, but this type of declare model setup will create two rows in the dataframe, Y_Z_1 (treatment outcome) and Y_Z_0 (control outcome), for each row in your dataframe. You will need call these variables later in the lab.

Inquiry

Now that we have declared the model we plan on using, we are going to specify our inquiry that we are interested in. Our use of this step depends on the structure of our data for the previous step and our research design. For our first design, because we are using observational data without a random treatment, we do not have to specify the quantity of inquiry. For our second design, we are interested in randomly assigning a treatment, Z, and we specified the potential outcomes framework. To do this, specify the PATE as the average difference between the treatment and control groups.

```
# Declare Inquiry for PATE (Think -- what do we want to calculate)
inquiry2 <- declare_inquiry(PATE = mean(Y_Z_1 - Y_Z_0))
```

Data Strategy

Now it is time to set up our data strategy. This part is made up of a few steps. Again, we will be working primarily with our second design because our first design (1) is already a sample (2) lacks random assignment and (3) already has a specified outcome for the model whereas the second design has both Y_Z_1 and Y_Z_0 for each observation.

As noted above, m2 is generating a dataframe at the *population* level. In many cases, we don't have the capability to study the whole population, so we need to specify a sampling procedure. To provide substantive understanding for why this might be the case, recall the example used in class discussing random assignment of GOTV mailers. We may be interested in specifying our sampling procedure if we have data at the population level. For the purposes of this lab, we will create a simple sample of 100 people from our population chosen completely at random. There are a number of different sampling techniques you can use here if your design calls for it (i.e. stratified sampling).

```
# Use declare_sample and specify S as completely random (complete_rs)
sample2 <- declare_sampling(S = complete_rs(N, n = 100))
```

Next, we need to randomly assign our treatment. To do this, we will use declare_assignment and specify Z to be completely random. As with our sampling procedure above, you can also specify specific arguments for assignment depending on your research design. This is helpful if you are using blocked randomization. Here, we will just randomly assign treatment and control groups with probability 0.5

```
# Define random assignment and specify Z as completely random with prob = 0.5
assignment2 <- declare_assignment(Z = complete_ra(100, prob = 0.5))
```

Finally, for our second design, we need to specify which outcome, Y_Z_1 or Y_Z_0 we want included in our SATE. While in our simulated data we are able to observe both outcomes for each individual, no design has this capability (recall the fundamental problem of causal inference). As a result, we need to use the treatment assignment, Z, to specify which of the outcomes we actually observe in the estimate for Y, our observed outcome.

```
# Declare measurement, specify Y to reveal outcomes of assigned treatment
measurement2 <- declare_measurement(Y = reveal_outcomes(Y ~ Z))
```

Answer Strategy

Now that we have our model, quantity of inquiry, and design strategy, we need to specify the estimator we are interested in. For our observational model, we are going to declare an estimator that looks at the effect of X_1 and X_2 on Y using an OLS regression.

```
# Declare estimator
estimator1 <- declare_estimator(Y ~ X_1+X_2, # Formula
                                term = c('X_1','X_2'), # Use term to identify which
                                                       # Variables you want to see
                                .method = lm) # Specify .method (what model?)
```

For our experimental study, we are interested in comparing the difference in mean values from the control group ($Z = 0$) and the treatment group ($Z = 1$). Same as above, declare the estimator and specify the variables you are interested in. Because we are not using a model this time, we need to specify our method as difference in means. We also need to specify our quantity of inquiry from above (PATE) so the model knows which variables in the dataframe to use.

```
# Declare estimator
estimator2 <- declare_estimator(Y ~ Z, # Specify formula
                                .method = difference_in_means, # Specify measure
                                inquiry = "PATE") # Identify QOI
```

Build Design From Steps

From all the steps above, declare design creates values that need to be combined into a single element. To do this, you simply take the steps from above and add them to create the full design with + between each step. Be advised, the order of the steps matters!

```
# Create Design1 object by adding m1 and estimator1
design1 <- m1 + estimator1

# Create Design2 object by adding all steps for design2 above
design2 <- m2 + inquiry2 + sample2 + assignment2 + measurement2 + estimator2
```

Simulate Research Design

Now that we have our design elements with all steps included, it is time to see the simulated results. There are a few different components you can pull out of this. The first is the simulated dataframe. Call draw_data on the design objects for the first ten rows. Does the data look like you expect? Take note, if we had not specified measurement2 in the prior step, we would not have a single observable Y.

```
# Use draw_data to view first ten rows of data for design1
draw_data(design1)[1:10,]
```

```
##     ID         X_1      X_2         Y
## 1  001  0.68901460 2.639316  6.699202
## 2  002 -0.13526244 4.340527 17.400556
## 3  003  0.19119060 3.757090 20.706126
## 4  004  2.00401807 5.272972 44.883139
## 5  005 -0.09295207 4.293242 -7.042469
## 6  006  0.48224988 4.540811 17.272494
## 7  007 -2.15176990 3.458448 23.703397
## 8  008  0.07213766 5.566825 17.662685
## 9  009 -0.16061116 2.217488 10.558209
## 10 010 -0.54244093 3.388429 13.623949
```

```
# Use draw_data to view first ten rows of data for design2
draw_data(design2)[1:10,]
```

```
##        ID Y_Z_0        Y_Z_1 S Z          Y
## 1   08259     0 -0.183817836 1 1 -0.1838178
## 2   09787     0  0.084681849 1 0  0.0000000
## 3   23000     0 -0.282606175 1 0  0.0000000
## 4   24360     0  0.225390093 1 1  0.2253901
## 5   34245     0  0.192997684 1 0  0.0000000
## 6   49872     0  0.123448784 1 1  0.1234488
## 7   91576     0  0.003744113 1 0  0.0000000
```

```
## 8  104376     0  0.483011773 1 1  0.4830118
## 9  114805     0  0.127790423 1 1  0.1277904
## 10 137789     0 -0.025096446 1 0  0.0000000
```

We can also see the results of our analysis given the simulated data. To see this, call `draw_estimates` on each design object. How closely do the quantities of interest match the relationships we specified in the above parts?

```
# View Estimates for design1. Do they match expectations?
draw_estimates(design1)
```

```
##   estimator term  estimate std.error statistic      p.value  conf.low conf.high
## 1 estimator  X_1 0.7984976 0.8453649 0.9445596 3.453427e-01 -0.862432  2.459427
## 2 estimator  X_2 3.3633031 0.4358559 7.7165492 6.600889e-14  2.506956  4.219650
```

```
# View PATE for design2. Do they match expectations?
draw_estimates(design2)
```

```
##   estimator term   estimate  std.error statistic    p.value    conf.low
## 1 estimator    Z 0.06315932 0.02862204  2.206667 0.03205408 0.005641176
##   conf.high df outcome inquiry
## 1 0.1206775 49       Y    PATE
```

One of the great advantages of declare design is your ability to play with different parameters. For example, how does sample size change the outcomes you observe? The same goes with anticipated effect size. Especially in the experimental where you are randomly assigning a treatment, this can help with research design decisions.