WP5

Alex Stephenson

10/10/2021

Before continuing on this assignment

Install the package `formatR`. Install the `scales` package.

Use the following data for Questions 1 to 3.

```
data <- tibble(unit = 1:6, Y = c(1, NA_real_, 1, 0, 1, NA_real_),
    R = c(1, 0, 1, 1, 1, 0), x1 = c(0, 0, 0, 0, 1, 1), x2 = c(3,
        7, 9, 5, 4, 3))
```

For the data in this assignment, consider R to be an indicator of whether or not an outcome is missing or not.

Question 1

Suppose missingness of outcomes is completely at random. This is referred to as `MCAR`. In this situation, to get an estimate we impute the values of missing data as equivalent to the expectation of the group as a whole.

Estimate $E[Y]$ under MCAR.

Under MCAR, we don't need to know anything about the structure of why something is missing, so your estimate of Y is just the E[Y]

```
data %>%
    select(Y) %>%
    summarise(E_Y = mean(Y, na.rm = T)) %>%
    pull()
```

```
## [1] 0.75
```

How often do you think MCAR holds? Why?

Probably rare in actual data settings. Most times, units have reasons for why they do not stay in a study or provide an outcome.

Question 2

Suppose missingness of outcome is missing at random, once we condition on covariates. In this situation, we have one (or multiple) covariates and take the conditional expectation of our outcome within each value (or strata) of the covariate.

Have you seen a procedure that does the same algorithm as Q2? Assuming the answer is no explain why.

Question 3

If the answer to Q2 is yes, run that algorithm find an estimate of $E[Y]$. What has to hold about this algorithm?

For this problem, we need to mutate our missing values based on the values of our covariates. There are two covariates here that we are going to use, X1 and X2.

```
algo <- lm_robust(Y ~ x1 + x2, data = data)
mean(predict(algo, data))
```

```
## [1] 0.7797619
```

The model has to be correct. We are asserting that this model is all that is required in order to get the missing values of potential outcomes.

Question 4

For this problem, use the Q4 dataset contained in Q4.csv. The dataset is a pure simulation, but you can think of this as the effect of taking a class on causal inference on knowledge of political science.

```
set.seed(30)
N <- 2500
d4 <- tibble(unit = 1:N, salary = rnorm(N, mean = 1000, sd = 106)) %>%
    mutate(education_b = rnorm(N, 8, 1), education_s = salary *
        0.04, education = rescale(education_b + education_s +
        runif(N, 0, 1)), D_score = 0.5 * salary + 1.5 * education +
        rnorm(N), D_prob = rescale(D_score, to = c(0.05, 0.95)),
        D = rbinom(N, 1, D_prob), Y_base = rbeta(N, shape1 = 4,
            shape2 = 5) * 100, Y_eff = 20 * D + 2 * education +
            0.1 * salary + Y_base + rnorm(N), Y = rescale(Y_eff,
            to = c(5, 80))) %>%
    select(unit, salary, education, D, Y)
```

a) Run a usual estimate of Y on D. Do not worry about covariates. What answer do you get for the effect of D on Y?

```
lm_robust(Y ~ D, data = d4)
```

```
##             Estimate Std. Error   t value      Pr(>|t|) CI Lower CI Upper   DF
## (Intercept) 35.77937  0.3083012 116.05330  0.000000e+00 35.17482 36.38392 2498
## D           14.04995  0.4221538  33.28159 2.384136e-201 13.22214 14.87776 2498
```

The effect of D is about 13.4 units. We don't have any additional information in the problem about what the units are.

b) Often, we can get a better estimate of our outcome by using information about treatment using a procedure called inverse probability weighting. This is a multi-step procedure.

c) Thanks to a famous theorem (Rosenbaum and Rubin 1983), the propensity score is a way to cut down on all the possible covariates predicting treatment to a single uni-dimensional variable. A propensity score is the probability of being assigned to a certain treatment, conditional on pre-treatment characteristics.

To estimate this, for old reasons we often use a logistic regression to predict the value of the treatment variable given relevant pre-treatment covariates. In R, a logistic regression is done like the following:

```
example <- glm(D ~ x1 + x2, family = binomial(link = "logit"),
    data = data)
```

Run a propensity score model with the dataset. Save your results to a variable called pscore.

```
pscore <- glm(D ~ salary + education, family = binomial(link = "logit"),
    data = d4)
```

ii) Once we have the propensity score, we generate inverse probability weights following the formula.

$$\frac{Treatment}{Propensity} - \frac{1 - Treatment}{1 - Propensity}$$

In R, an example to do this is as follows:

```r
data_w_weights <- broom::augment_columns(example_model, data,
    type.predict = "response") %>%
    # Not necessary but makes it more readable
rename(prop = .fitted) %>%
    mutate(inverse_probability_weight = (D/prop) + ((1 - D)/(1 -
        prop)))
```

Adapt the example to get the inverse probability weights for the dataset in the problem.

```r
d4 <- broom::augment(pscore, d4, type.predict = "response") %>%
    rename(prop = .fitted) %>%
    mutate(ipw = (D/prop) + ((1 - D)/(1 - prop))) %>%
    # This line removes all the other augment junk we don't
    # need for this problem
select(-contains("."))

head(d4)
```

```
## # A tibble: 6 x 7
##    unit salary education     D     Y  prop   ipw
##   <int>  <dbl>     <dbl> <int> <dbl> <dbl> <dbl>
## 1     1   863.     0.312     0  29.6 0.378  1.61
## 2     2   963.     0.441     0  44.0 0.495  1.98
## 3     3   945.     0.402     0  28.0 0.480  1.92
## 4     4  1135.     0.695     0  36.9 0.679  3.11
## 5     5  1193.     0.748     1  53.7 0.745  1.34
## 6     6   840.     0.258     0  25.0 0.361  1.57
```

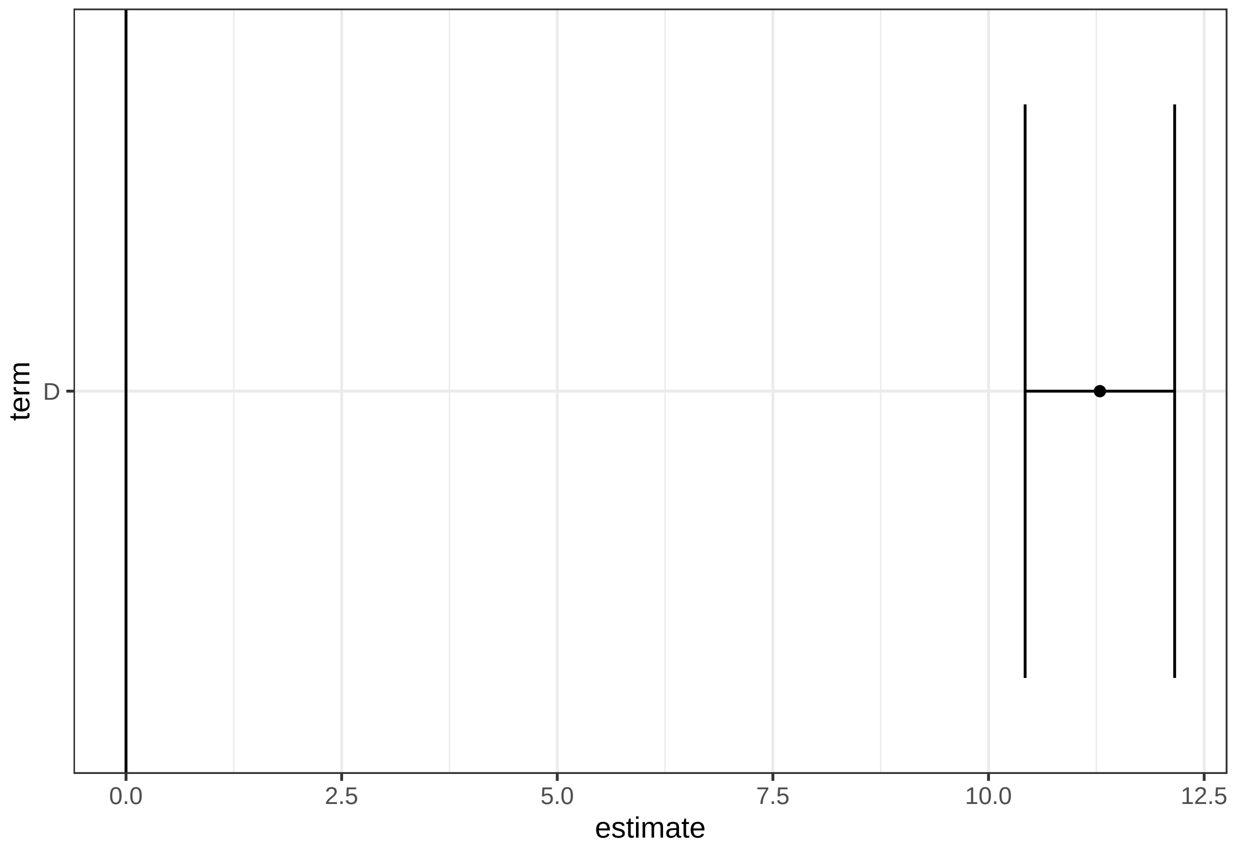
iii) Now that we have the weights, we can run a weighted least squares regression model. Happily, all this requires to compute is supplying R with the appropriate weights as a variable.

In `estimatr` we can do this with:

```r
lm_robust(Y ~ D, data = data, weights = column_of_weights) %>%
    tidy()
```

Adapt this example for this problem. Make and interpret a coefficient plot of your answer.

```r
coef_plot <- lm_robust(Y ~ D, data = d4, weights = ipw) %>%
    tidy() %>%
    # This is another way to filter out the intercept term
filter(!grepl("Int", term)) %>%
    ggplot(aes(x = estimate, y = term)) + geom_point() + geom_errorbarh(aes(xmin = conf.low,
    xmax = conf.high)) + theme_bw() + geom_vline(xintercept = 0)
coef_plot
```



Coda

We are doing this by hand to describe the process of inverse probability weighting. In a real project, we would be highly unlikely to do this by hand. Rather we would take advantage of whatever package was commonly used in our language of choice. Doing the example by hand allows us to see the mechanics so we understand what the "best" library is doing.

Inverse probability weighting is applied to account for different proportions of observations within strata in the population of interest. It can also be awfully unstable and prone to bias if the estimated propensity scores are small. This is because the propensity scores are showing up in the denominators, and because the common way to estimate them (logistic regression) can become unstable at the tails of the distribution.