Section 4

Alex Stephenson

9-15-2021

What are we doing today?

Demonstrating how to do Regression in R (not by hand)

Packages we need for today

```
library(estimatr)
library(fabricatr)
library(randomizr)
library(tidyverse)
```

If you do not have these packages, install them.

Regression in R base way

Without loading any packages, it is possible to run regressions in R.

For OLS, the command is `lm()`

Example

```
set.seed(42)
dat <- fabricate(
  N = 100,                          # sample size
  x = runif(N, 0, 1),               # pre-treatment covariate
  y0 = rnorm(N, mean = x),          # control potential outcome
  y1 = y0 + 0.35,                   # treatment potential outcome
  z = complete_ra(N),               # complete random assignment to treatment
  y = ifelse(z, y1, y0),            # observed outcome

  # We will also consider clustered data
  clust = sample(rep(letters[1:20], each = 5)),
  z_clust = cluster_ra(clust),
  y_clust = ifelse(z_clust, y1, y0)
)
```

The dataset for today

```
kable(head(dat))
```

ID	x	y0	y1	z	y	clust	z_clust	y_clust
001	0.9148060	1.2367313	1.5867313	1	1.5867313	s	1	1.5867313
002	0.9370754	0.1532365	0.5032365	1	0.5032365	q	0	0.1532365
003	0.2861395	1.8618671	2.2118671	1	2.2118671	b	1	2.2118671
004	0.8304476	1.4733469	1.8233469	1	1.8233469	k	1	1.8233469
005	0.6417455	0.7315062	1.0815062	0	0.7315062	b	1	1.0815062
006	0.5190959	0.7956467	1.1456467	0	0.7956467	o	0	0.7956467

Regression the base way

```
m <- lm(y~z + x, data = dat)
kable(tidy(m))
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.1831761	0.2074911	-0.882814	0.3795194
z	0.2056310	0.1855039	1.108499	0.2703871
x	1.4386941	0.3087272	4.660082	0.0000101

What's wrong with these errors?

The estimatr package

`estimatr` is a package dedicated to providing estimators commonly used by social scientists.

The package provides estimators tuned for design-based inference.

There are other packages that do this as well, but I personally like `estimatr` hence why I'm teaching it.

The package also has some handy mathematical notes for explaining calculations.

lm_robust()

The function we should use instead as a default is `lm_robust()`

This function lets us quickly fit linear models with the most common variance estimators and degrees of freedom corrections used in social science

We can easily fit heteroskedastic standard errors, clustered standard errors, and "stata" standard errors

Regression the correct way

```
m <- lm_robust(y~z + x, data = dat, se_type = "HC0")
kable(tidy(m))
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high	df	outcome
(Intercept)	-0.1831761	0.1687923	-1.085216	0.2805158	-0.5181820	0.1518299	97	y
z	0.2056310	0.1828161	1.124797	0.2634507	-0.1572083	0.5684703	97	y
x	1.4386941	0.2816971	5.107238	0.0000016	0.8796033	1.9977849	97	y

Regression with Cluster Assignment

To include cluster information, pass the cluster variable to the clusters argument

```
m2 <- lm_robust(
  y_clust ~ z_clust + x,
  data = dat,
  clusters = clust
)
kable(tidy(m2))
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high	df	outcom
(Intercept)	-0.2961247	0.1982991	-1.493323	0.1573090	-0.7210042	0.1287547	14.15262	y_clust
z_clust	0.4470588	0.1820699	2.455423	0.0244914	0.0644865	0.8296310	17.96232	y_clust
x	1.4238885	0.2653906	5.365255	0.0000571	0.8626182	1.9851588	16.48145	y_clust

`lm_lin()`

Adjusting for pre-treatment covariates is common practice. We have seen how it can increase precision

However, under certain condition (Freedman 2008) covariate adjustment via regression can bias our estimate of the ATE when we have groups of unequal size.

If we take the concern seriously, there is an alternative estimator that reduces bias and improves precision (Lin 2013)

lm_lin()

```
m3 <- lm_lin(
  y~z,
  covariates = ~x,
  data = dat
)

kable(tidy(m3))
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high	df	outcome
(Intercept)	0.5719507	0.1194132	4.7896781	0.0000061	0.3349174	0.8089839	96	y
z	0.2056204	0.1861152	1.1048015	0.2720069	-0.1638154	0.5750562	96	y
x_c	1.5904821	0.4143589	3.8384164	0.0002219	0.7679861	2.4129780	96	y
z:x_c	-0.3007169	0.5771479	-0.5210395	0.6035391	-1.4463464	0.8449127	96	y

Plotting Regression Coefficients

Regression tables tend to be lengthy and unhelpful for presenting results, experimental or otherwise.

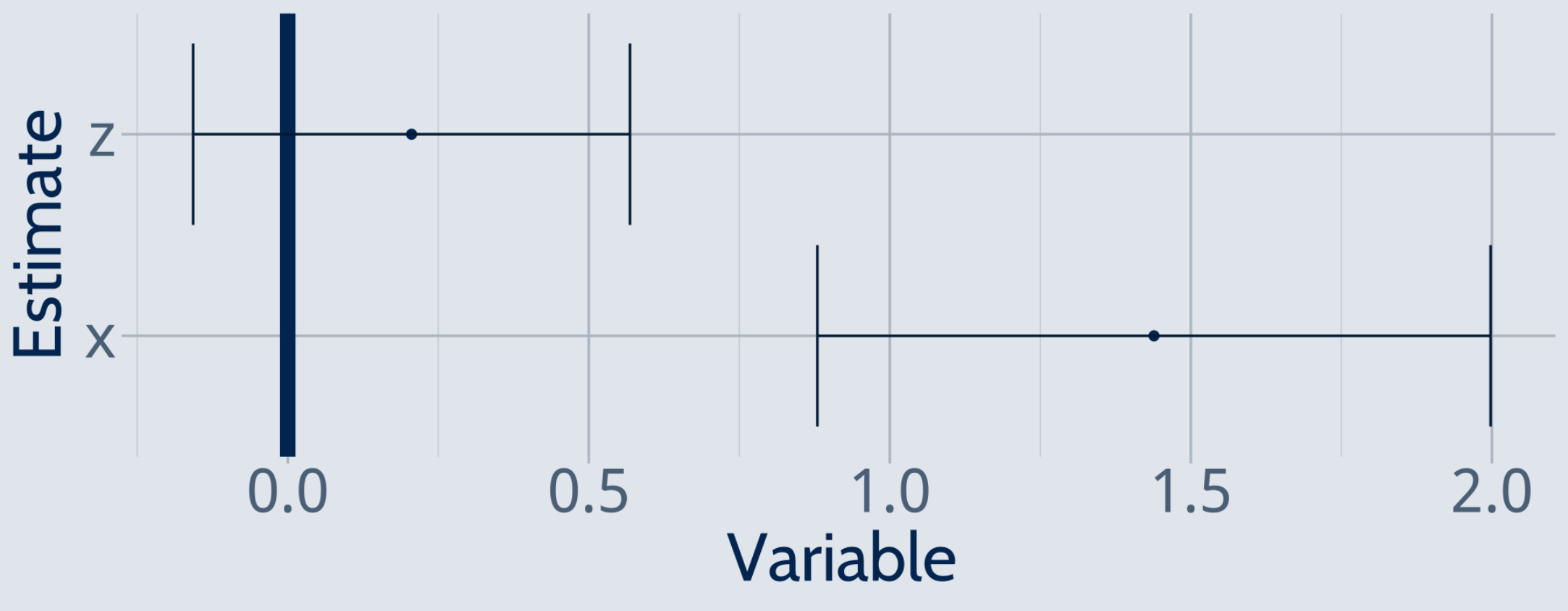
It is preferable to make a coefficient plot to directly show coefficients

The output of `lm_robust()` makes that easy to plug into ggplot2

Plotting Regression Coefficients

```
m %>%
  tidy %>%
  filter(term != "(Intercept)")%>%
  ggplot(aes(x = term, y = estimate))+
  geom_point()+
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))+
  geom_hline(yintercept = 0)+
  coord_flip()
```

Plotting Regression Coefficients



What do I do for my assignments?

Unless explicitly instructed to conduct regression by hand or with `lm()`, use `lm_robust()` as a default.

Alternative estimators that we will cover in class also are implemented in `estimatr`

If I ask on a problem set for a coefficient plot, it is wrong to present regression output in a table.