001: CEFs, inference, simulation, *etc.*

EC 607

Due *before* midnight on Sunday, 09 May 2021

Part 1/3: CEFs and regression

Let's start with generating data. We want a nonlinear CEF, define our data-generating process (DGP) as

$$y_i = 3 + \mathbb{I}(x_i < 5)\left(x_i^2 + 1\right) + \mathbb{I}(x_i \geq 5)\left(-0.25 * x^2 + 25\right) + u_i$$

where

- $\mathbb{I}(x)$ denotes an indicator function that takes a value of 1 whenever x is true.
- x_i is distributed as a continuous uniform random variable taking on values from $[0, 10]$. I'm going to round x_i to 1 decimal.
- u_i is a heteroskedastic disturbance that follows a normal distribution with mean zero and standard deviation $0.5 + \backslash\text{abs}5 - x$.

Notice that this DGP is really just two separate DGPs determined by whether x_i is above or below 5 (plus the disturbance u_i).

01. Time to generate data. Given this is the first problem of your first problem set, I'll give you some code (for free).

```
# Load packages
library(pacman)
p_load(tidyverse, estimatr, huxtable, magrittr, here)
# Set a seed
set.seed(12345)
# Set sample size to 1,000
n = 1e3
# Generate data
dgp_df = tibble(
  x = runif(n = n, min = 0, max = 10) %>% round(1),
  u = rnorm(n = n, mean = 0, sd = 0.5 + abs(5 - x)),
  y = (x < 5) * (x^2 + 1) + (x >= 5) * (-0.25 * x^2 + 25) + u
)
# Summarize the dataset
dgp_df %>% summary()
```

```
#>        x                u                 y         
#>  Min.   : 0.00   Min.   :-15.340   Min.   :-13.53  
#>  1st Qu.: 2.70   1st Qu.: -1.637   1st Qu.:  4.75  
#>  Median : 5.20   Median : -0.024   Median : 10.25  
#>  Mean   : 5.14   Mean   : -0.084   Mean   :  9.93  
#>  3rd Qu.: 7.60   3rd Qu.:  1.554   3rd Qu.: 15.62  
#>  Max.   :10.00   Max.   : 15.159   Max.   : 25.42
```

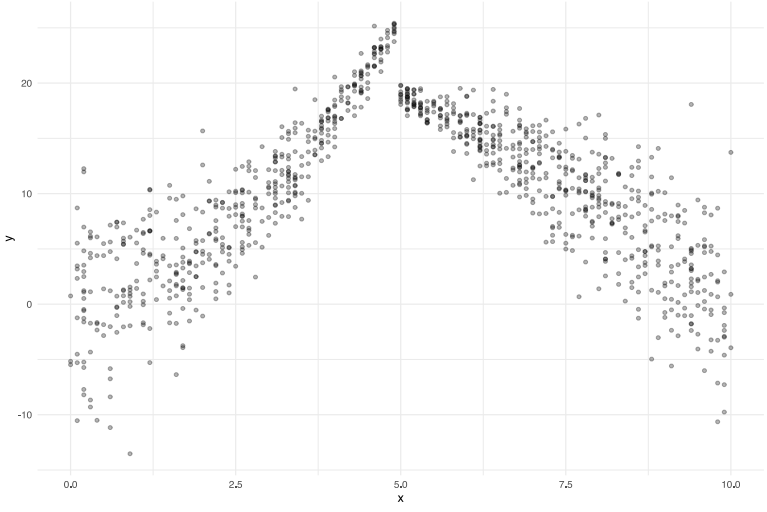
Run this code.

Make sure your output is pretty close to my output (and that you have a sense of what's going on).

02. Create a scatter plot of your dataset (*e.g.*, using `geom_point` from `ggplot2`).

Answer:

```
ggplot(
  data = dgp_df,
  aes(x = x, y = y)
) +
geom_point(alpha = 0.3) +
theme_minimal(base_size = 12)
```



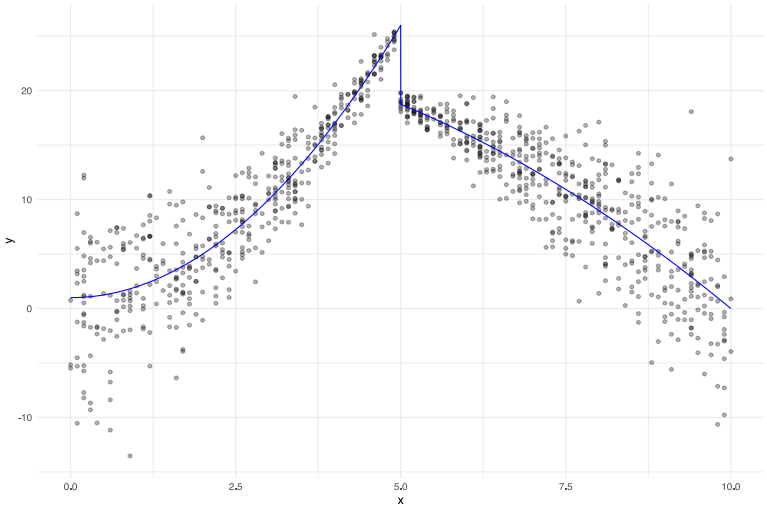
03. Derive the CEF and add it to your scatter plot.

Hint: Keep in mind the definition of the CEF (the expected value of y given x).

Hint: You can plot a function in `ggplot2` using `stat_function`.

Answer: Because $E[u|x] = 0$, our CEF is simply the definition of y excluding the u component.

```
# The CEF function
cef = function(x) (x < 5) * (x^2 + 1) + (x >= 5) * (-0.25 * x^2 + 25)
# Plot it
ggplot(dgp_df, aes(x = x, y = y)) +
geom_point(alpha = 0.3) +
stat_function(fun = cef, color = "blue", n = 1e4) +
theme_minimal(base_size = 12)
```



04. Regress y on x. Calculate standard errors assuming **homoskedasticity**. Report your results.

Answer:

```
# 'Classical' standard errors
lm_robust(y ~ x, data = dgp_df, se_type = "classical")
```

```
#>             Estimate Std. Error t value  Pr(>|t|) CI Lower CI Upper  DF
#> (Intercept)   8.5011    0.46657  18.220 2.949e-64   7.5856   9.4167 998
#> x             0.2771    0.07942   3.489 5.067e-04   0.1212   0.4329 998
```

05. Do heteroskedasticity-robust standard errors "matter" here? Why? Explain your reasoning.

Answer: As the table below illustrates, heteroskedasticity-robust standard errors *may* matter a little bit here. The "classical" standard errors are approximately 11% smaller than they should be, but the significance of the point estimates does not really change. More generally: We would expect heteroskedasticity to at least be present due (1) the disturbances in the CEF are heteroskedastic (varying with x) and (2) the CEF is nonlinear, while our regression is linear.

```
# Het-robust standard errors
est_ols = lm_robust(y ~ x, data = dgp_df, se_type = "classical")
est_hc2 = lm_robust(y ~ x, data = dgp_df, se_type = "HC2")
# Table
list("Classical" = est_ols, "Het. Robust" = est_hc2) %>% huxreg()
```

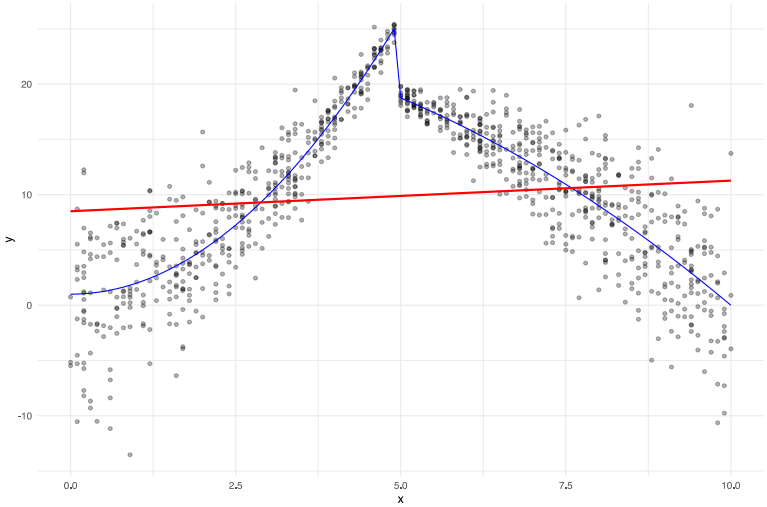
	Classical	Het. Robust
(Intercept)	8.501 ***	8.501 ***
	(0.467)	(0.511)
x	0.277 ***	0.277 **
	(0.079)	(0.089)
N	1000	1000
R2	0.012	0.012

*** p < 0.001; ** p < 0.01; * p < 0.05.

06. Add your regression line to your scatter plot. You can do this in `ggplot2` using `geom_abline()` and `geom_smooth()` (among other options).

Answer:

```
# Plot it
ggplot(dgp_df, aes(x = x, y = y)) +
geom_point(alpha = 0.3) +
stat_function(fun = cef, color = "blue") +
geom_smooth(method = lm, se = F, color = "red") +
theme_minimal(base_size = 12)
```



07. For each of our 31 values of x (-15 through 15), calculate the sample mean of y conditional on x and the number of observations for each x.

Now run a regression using this sample-based CEF: Regress the conditional mean of $y \mid x$ on x, weighting by the number of observations. Do your results from this CEF regression match your results in **04**? Should they for this sample? Comment on the point estimates and the standard errors—and explain why each should or should not match.

Hint: You can use the `weights` argument in `lm()` and `lm_robust()` to run a weighted regression.

Answer:

```
# Build the sample-based CEF
cef_df = dgp_df %>%
  mutate(x) %>%
  group_by(x) %>%
  summarize(y = mean(y), n = n())
# Run the regression
est_cef = lm_robust(y ~ x, data = cef_df, weights = n)
# Table
list("Classical" = est_ols, "Het. Robust" = est_hc2, "Aggregated" = est_cef) %>% huxreg()
```

	Classical	Het. Robust	Aggregated
(Intercept)	8.501 ***	8.501 ***	8.501 ***
	(0.467)	(0.511)	(1.422)
x	0.277 ***	0.277 **	0.277
	(0.079)	(0.089)	(0.246)
N	1000	1000	101
R2	0.012	0.012	0.015

*** p < 0.001; ** p < 0.01; * p < 0.05.

The point estimates *do* match. The standard errors do not. These results are expected: The aggregated dataset, when weighted, produces the same OLS estimates (since the OLS estimator can be written as weighted sums of aggregated sample moments). The standard errors differ because the aggregated observations do not produce the same residuals or sample size as the micro-data.

08. Does OLS provide a decent linear approximation to the CEF in this setting? Under what conditions would this linear approximation of the CEF be helpful? Under what conditions would it be less helpful?

Answer: Maybe... it depends what you mean by "decent". Does OLS's straight line perfectly match the nonlinear CEF? No. Of course not. OLS is doing what we asked it to do: providing a linear approximation to the conditional expectation function. In this case, the linear approximation differs quite a bit from the CEF. We might want to model this process a bit more flexibly (obviously still possible with OLS).

Part 2/3: Inference and simulation

Now it's time for a good, old-fashioned simulation.

Now imagine you're working on a project, and it occurs to you that

1. You have a pretty small sample size (but could spend a lot of money to get bigger n).
2. It's unlikely that your disturbance is actually normally distributed.
3. You might have an endogenous treatment D_i but have a sense of how treatment comes about.

Given that the small-sample properties of OLS generally use *well-behaved disturbance* and the large-sample properties are, by definition, for **big** n, you are wondering how well OLS is going to perform. Plus, you are really concerned about the endogenous treatment but optimistic that you know how the treatment is endogenous. Can we recover the *true* treatment effect?

This is the perfect scenario for a simulation.

I'll walk you through some of the steps of the simulation. But you have to write your own code.

Let's start by defining the DGP (using notation from class)

$$\begin{aligned}
\mathrm{Y}_{0i} &= X_i + u_i \\
\mathrm{Y}_{1i} &= \mathrm{Y}_{0i} + W_i + v_i \\
\mathrm{D}_i &= \mathbb{I}(X_i + \varepsilon_i > 10) \\
\mathrm{Y}_i &= \mathrm{Y}_{0i} + \mathrm{D}_i \tau_i
\end{aligned}$$

where

- $X_i \sim$ Normal with mean 10 and standard devation 3
- $W_i \sim$ Normal with mean 3 and standard devation 2
- $u_i \sim$ Uniform $\in [-10, 10]$
- $v_i \sim$ Uniform $\in [-5, 5]$
- $\varepsilon_i \sim$ Uniform $\in [-1, 1]$

10. Derive an expression for τ_i (individual i's treatment effect).

Answer: As defined in class, τ_i is equal to the difference in treated and untreated outcomes for individual i, *i.e.*,

$$\begin{aligned}
\tau_i &= \mathrm{Y}_{1i} - \mathrm{Y}_{0i} \\
&= (\mathrm{Y}_{0i} + W_i + v_i) - \mathrm{Y}_{0i} \\
&= W_i + v_i
\end{aligned}$$

11. What assumptions does the expression for the treatment effect in **10** depend upon?

Answer: None. If you really want to name an assumption, it is that we can define the causal effect of some arbitrary treatment as the difference between Y_{1i} and Y_{0i}.

You might also be able to say that our definition of the treatment effect assumes that individual i's treatment effect does not depend upon other individuals' treatment statuses. This second assumption is the *stable unit treatment value assumption* (SUTVA), which is safe in our setting (we know the DGP).

12. Based upon **10**, what is the average treatment effect in this population? (Your answer should be a number.)

Answer: The ATE is

$$\begin{aligned}\overline{\tau_i} &= E[\tau_i] \\ &= E[W_i + v_i] \\ &= E[W_i] + E[v_i] \\ &= 3 + 0 \\ &= 3\end{aligned}$$

13. If we regress Y_i on D_i should we expect to recover the average causal effect of treatment (D_i)? Explain.

Answer: No. Our potential outcomes are correlated with treatment: they all depend upon X_i. In other words: We have selection bias, since $E[\mathrm{Y}_{0i}|\mathrm{D}_i = 1] \neq E[\mathrm{Y}_{0i}|\mathrm{D}_i = 0]$.

14. Would conditioning on X and/or W help the regression in **13**? Explain.

Answer: Yes: Because selection is entering through X_i, if we can control for X_i, we will remove the selection bias.

15. Now back to R: Write some R code that generates a 1,000-observation sample from the DGP.

Answer:

```
# Set seed
set.seed(123)
# Sample size
n = 1e3
# Generate data
dgp_sample = tibble(
    u = runif(n, -10, 10),
    v = runif(n, -5, 5),
    e = runif(n, -1, 1),
    x = rnorm(n, mean = 10, sd = 3),
    w = rnorm(n, mean = 3, sd = 2),
    y0 = x + u,
    y1 = y0 + w + v,
    t = y1 - y0,
    d = (x + e > 10) %>% as.numeric(),
    # d = (x + w + e > 13) %>% as.numeric(),
    y = y0 + d * t
)
```

16. For your sample, what is the correlation between Y_{0i} and D_i? What about Y_{1i} and D_i? What do these correlations tell you?

Answer: The correlation between Y_{0i} and D_i is 0.317, and correlation between Y_{1i} and D_i is 0.319. This correlation suggests that we have substantial selection into treatment (and thus bias from selection).

```
# Correlation matrix for Y0, Y1, and D
dgp_sample %>% select(y0, y1, d) %>% cor()
```

```
#>         y0     y1      d
#> y0 1.0000 0.8722 0.3173
#> y1 0.8722 1.0000 0.3189
#> d  0.3173 0.3189 1.0000
```

17. Using your sample, calculate the average treatment effect (ATE), the average treatment effect on the treated (TOT or ATT), and the average treatment effect for the untreated. Why do these quantities differ?

Answer:

```
# ATE
ate = dgp_sample %>% summarize(ate = mean(y1 - y0))
# ATET
atet = dgp_sample %>% filter(d == 1) %>% summarize(ate_t = mean(y1 - y0))
# ATEC
atec = dgp_sample %>% filter(d == 0) %>% summarize(ate_c = mean(y1 - y0))
```

The sample's ATE is approximately 2.984, the ATET is approximately 3.237, and the ATEC is approximately 2.740.

The average treatment effects differ across groups because (1) the treatment effect is heterogeneous, and (2) we are only observing a small sample (*i.e.*, we have sampling variation). If you allow the size of the sample to get large enough, the difference in the group's means will disappear.

18. Run four regressions:

1. Regress Y_i on D_i
2. Regress Y_i on D_i and X_i
3. Regress Y_i on D_i and W_i
4. Regress Y_i on D_i, X_i, and W_i

Do the results of these regressions match your expectation for recovering the ATE or ATT? Explain.

Answer:

```
# The four regressions
r1 = lm_robust(y ~ d, data = dgp_sample)
r2 = lm_robust(y ~ d + x, data = dgp_sample)
r3 = lm_robust(y ~ d + w, data = dgp_sample)
r4 = lm_robust(y ~ d + x + w, data = dgp_sample)
# A table
list(r1, r2, r3, r4) %>% huxreg()
```

	(1)	(2)	(3)	(4)
(Intercept)	7.965 ***	0.645	6.696 ***	-0.590
	(0.271)	(0.864)	(0.403)	(0.888)
d	7.263 ***	2.873 ***	7.197 ***	2.820 ***
	(0.406)	(0.625)	(0.402)	(0.620)
x		0.948 ***		0.945 ***
		(0.107)		(0.106)
w			0.434 ***	0.429 ***
			(0.098)	(0.094)
N	1000	1000	1000	1000
R2	0.243	0.299	0.257	0.313

*** p < 0.001; ** p < 0.01; * p < 0.05.

The results for the regressions that do not control for X_i are clearly biased, as we expected from the fact that X_i is causing selection into treatment.

Conditional on X_i, treatment is independent of the conditional outcomes, so we should be able to recover an unbiased estimate. The regression, as we've specified, estimates the ATE. Further, because dimensions of treatment-effect heterogeneity ($W_i + v_i$) are uncorrelated with treatment, the ATE and the ATT are equal.

Finally, because regressions that include W_i allow us to model the treatment-effect heterogeneity and modestly reduce residual variation (increasing precision; reducing standard errors)—but they also use up an additional degree of freedom (could matter for small-ish samples).

19. Now wrap your code from **15** and **18** into a function. This function will be a single iteration of the simulation. The function should output the estimated treatment effect in each of the four regressions in **18**.

Hint 1: Help your future self by writing this function so that you can easily change the sample size.

Hint 2: Use `tidy()` from the `broom` package to easily convert regression results into a data frame.

Hint 3: Label the output of the four regressions so that you can distinguish between each specification.

Answer:

```
# Function for one iteration
one_iter = function(n) {
    # Generate data
    dgp_it = tibble(
        u = runif(n, -10, 10),
        v = runif(n, -5, 5),
        e = runif(n, -1, 1),
        x = rnorm(n, mean = 10, sd = 3),
        w = rnorm(n, mean = 3, sd = 2),
        y0 = x + u,
        y1 = y0 + w + v,
        t = y1 - y0,
        d = (x + e > 10) %>% as.numeric(),
        # d = (x + w + e > 13) %>% as.numeric(),
        y = y0 + d * t
    )
    # Regression time
    bind_rows(
        lm(y ~ d, data = dgp_it) %>% broom::tidy() %>% filter(term == "d"),
        lm(y ~ d + x, data = dgp_it) %>% broom::tidy() %>% filter(term == "d"),
        lm(y ~ d + w, data = dgp_it) %>% broom::tidy() %>% filter(term == "d"),
        lm(y ~ d + x + w, data = dgp_it) %>% broom::tidy() %>% filter(term == "d")
    ) %>% mutate(controls = c("none", "x", "w", "x + w"))
}
```

20. Run a simulation with at least 500 iterations. Each iteration should

- take a new **15-observation** sample from our DGP
- output **four treatment-effect estimates** (one for each regression in **18**)
- output **four standard errors** (one for each estimate)

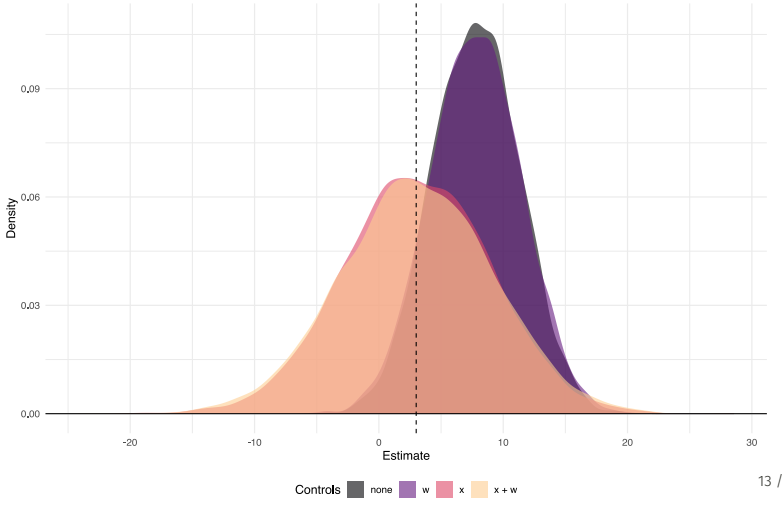
Summarize your results with a figure (*e.g.*, `geom_density()`) and/or a table.

Hints: The `apply()` family (*e.g.*, `lapply()`) works well for tasks like this, as does the `map` family from the `purrr` package (see the `future_map` family from the `furrr` package for parallelization). Also: The notes from class.

Answer:

```
# Load 'furrr'
p_load(furrr)
# Run the simulation
set.seed(1234)
# Set up the parallelization
plan(multiprocess, workers = 12, .progress = T)
invisible(future_options(seed = 1234L))
# Run the simulation
small_df = future_map_dfr(rep(15, 5000), one_iter)
```

```
# Plot simulation
ggplot(data = small_df, aes(x = estimate, fill = controls)) +
geom_density(color = NA, alpha = 0.6) +
geom_hline(yintercept = 0) +
geom_vline(xintercept = 3, linetype = "dashed") +
labs(x = "Estimate", y = "Density") +
scale_fill_viridis_d("Controls", option = "magma", end = 0.9) +
theme_minimal(base_size = 12) +
theme(legend.position = "bottom")
```



```r
# Table of summaries
small_df %>%
  group_by(controls) %>%
  summarize(
    "Mean Coef." = mean(estimate),
    "Median Coef." = median(estimate),
    "Std. Dev. Coef" = sd(estimate),
    "Rejection Rate" = mean(p.value < 0.05)
  ) %>%
  hux() %>% add_colnames()
```

controls	Mean Coef.	Median Coef.	Std. Dev. Coef	Rejection Rate
controls	Mean Coef.	Median Coef.	Std. Dev. Coef	Rejection Rate
none	7.79	7.8	3.51	0.523
w	7.8	7.8	3.64	0.494
x	3.06	3.04	5.89	0.08
x + w	3.06	3.01	6.13	0.0822

21. Are any of the estimation strategies (the four regressions) providing *reasonable* estimates of the average treatment effect?

Answer: As we discussed before, the methods that do not control for X_i are hopelessly biased. On the other hand, the methods that control for X_i are appear to be producing unbiased estimates for the ATE.

22. With 15 observations, do you think think you have enough power to *detect* a treatment effect? Explain.

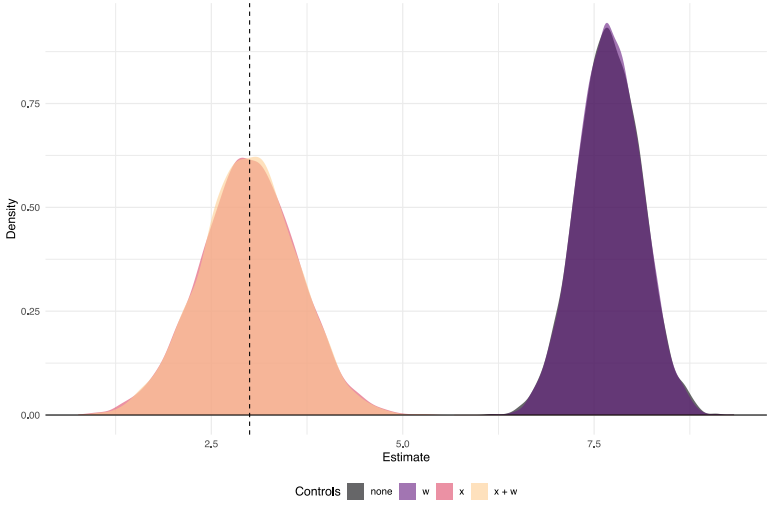
Answer: Our sample size seems to be hurting us quite a bit. As the summary table above shows: For the specifications that control for X_i, we reject the null hypothesis about 7.5% of the time (using a 5% significance level). The null hypothesis is indeed false, so we would hope to reject the null *way* more than 7.5% of the time—ideally at least 80% of the time. We need more power.

23. Increase the sample size to 1,000 observations per sample and repeat the simulation (including graphical/table summary). Does anything important change for causal estimates (*e.g.*, centers of the distributions) or inference (*e.g.*, rejection rates)?

Answer:

```
# Run the simulation
set.seed(1234)
# Set up the parallelization
plan(multiprocess, workers = 12, .progress = T)
invisible(future_options(seed = 1234L))
# Run the simulation
big_df = future_map_dfr(rep(1000, 5000), one_iter)
```

```
# Plot simulation
ggplot(data = big_df, aes(x = estimate, fill = controls)) +
geom_density(color = NA, alpha = 0.6) +
geom_hline(yintercept = 0) +
geom_vline(xintercept = 3, linetype = "dashed") +
labs(x = "Estimate", y = "Density") +
scale_fill_viridis_d("Controls", option = "magma", end = 0.9) +
theme_minimal(base_size = 12) +
theme(legend.position = "bottom")
```



```r
# Table of summaries
big_df %>%
  group_by(controls) %>%
  summarize(
    "Mean Coef." = mean(estimate),
    "Median Coef." = median(estimate),
    "Std. Dev. Coef" = sd(estimate),
    "Rejection Rate" = mean(p.value < 0.05)
  ) %>%
  hux() %>% add_colnames()
```

controls	Mean Coef.	Median Coef.	Std. Dev. Coef	Rejection Rate
controls	Mean Coef.	Median Coef.	Std. Dev. Coef	Rejection Rate
none	7.7	7.69	0.417	1
w	7.7	7.69	0.411	1
x	2.99	2.99	0.638	0.997
x + w	2.99	2.99	0.628	0.998

Increasing the size of the sample

24. Would getting even bigger data help the regressions that appear to be biased? *Related:* Is it worth paying for a bigger sample in this setting? Explain.

Answer: No: Getting more data does not help us with the bias. The selection bias is not a function of the sample size. Bigged data do not generally get rid of biased coefficients.

However, it *may* be worth paying for a bigger sample *if we can remove the selection bias.* Our rejection rates jump from 7.5% to 99.9% by increasing our sample size from 15 to 1,000.

In general: You pay for sample size to *increase precision*—not to reduce bias.

25. Should we control for W_i? Explain.

Answer: Controlling for W_i does nothing for the selection bias. It *does* allow you to model the treatment-effect hetergeneity *and* slightly increases your precision (with this larger sample size). So it's probably worth it. That said: Our conditional-independence assumption does not require it, so it is not necessary.

26. Draw the DAG for this DGP. What are the pathways from D to Y? How do you close the open pathways to get to the causal effect of D on Y?

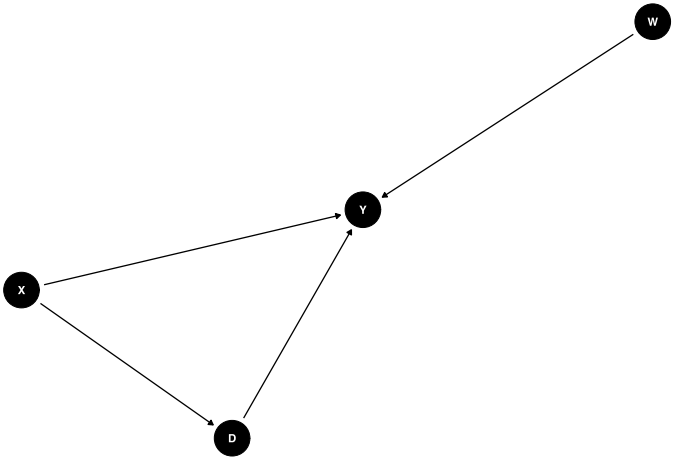
Hint: Check out the ggdag package for drawing DAGs in R.

Answer: There are only two paths from D to Y:

1. $D \rightarrow Y$
2. $D \leftarrow X \rightarrow Y$

Both paths are open, but we can close the second (non-causal) path by conditioning on X.

```
p_load(ggdag)
dagify(
  Y ~ X,
  Y ~ D,
  Y ~ W,
  D ~ X,
  outcome = "Y",
  exposure = "D"
) %>% ggdag(layout = "nicely") + theme_dag()
```



Part 3/3: Function time

27. Write your own function(s) that (1) produce the OLS-based coefficients for a regression and (2) produce the homoskedasticity-based standard errors for the coefficients. Confirm that your function is "working" by using your function to re-estimate the regression you ran in question **04** above.

You should be able to do most of this by converting your dataset to matrices (`as.matrix()` or `matrix()`) and then applying a little matrix math. In R, `%*%` is matrix multiplication, `solve()` produces the inverse of an invertible matrix, `crossprod()` calculates cross products, and `diag()` allows you to define a diagonal matrix or access the diagonal of an existing matrix.

Part 4/3: Bonus!

B01. Does anything important change if $\mathrm{D}_i = \mathbb{I}(X_i + W_i + \varepsilon_i > 13)$?

B02. Repeat the simulation steps—but use a Normal distribution for u, v, and ε (try to match the mean and variance). What changes (now that we're using a very well-behaved distribution)?

B03. Repeat the simulation steps—but use a very poorly behaved distribution for u, v, and ε (try to match the mean and variance, if they are defined). What changes?

B04. When we regress $\mathbf{Y}_i$ on D_i (and potentially controls), are we estimating the ATE or the ATT?