Lecture 8: Regression Analysis, Part 2

James Sears*

AFRE 891 SS 24

Michigan State University

*Parts of these slides are adapted from **“Data Science for Economists”** by Grant McDermott and **“Advanced Data Analytics”** by Nick Hagerty.

Table of Contents

Prologue

Empirical Analysis

We've spent the first half of our course building a **foundational knowledge** of data manipulation in R.

While cleaning, wrangling, and visualizing data can be fun, it's generally not the end goal - that's usually some form of **statistical/econometric analysis**.

Over the next weeks we're going to focus on the analysis side, using both **traditional** and **non-traditional** methods.

Empirical Analysis

This Lecture: Regression

- Standard regression: `lm()`
- Fixed effects regression: **fixest** and `feols`
- Formulas
- Choice of standard errors
- Regression Tables
- Visualizing regression output
- IV Regression
- Diff-in-Diff
    - Staggered Adoption
- Event Study and Sun-Abraham estimator

Next Lecture: Synthetic Control Methods

- Canonical Synthetic Control
- Synthetic Diff-in-Diff
    - Uniform Adoption
    - Staggered Adoption
- Partially Pooled Synthetic Control

Prologue

Packages we'll use today:

```r
remotes::install_github("lrberge/fixest")
```

```r
if (!require("pacman")) install.packages("pacman")
pacman::p_load(broom, fixest, stargazer, tidyverse)

options(scipen=999) # disable scientific notation
```

Prologue

As well, let's load data necessary to replicate results from **Taylor and Druckenmiller AER (2022)**[1]

```
wl_df <- read_csv("data/zip_PANEL_data.csv") %>%
  drop_na(county_fips, state_fips, claims)

wl_ld_df <- readRDS("data/zip_LD_data.rds")

coastal <- read_csv("data/coastal_zips.csv")

did_df <- left_join(wl_ld_df, coastal, by = "zip") %>%
  filter(coastal == FALSE)

rm(wl_ld_df, coastal)
```

[1] There is a *lot* of great stuff in this paper and it's all super replicable and well-documented. I highly recommend checking it out if you're curious (only caution is that there are some *big* files in there).

Regression Output

Running regression is fun and all, but generally we don't just want it internal to R. Rather, we want some nicely formatted **regression output**

- Regression Tables
- Regression Plots

Regression Output

To see convenient ways to get regression output, let's run a model looking at the relationship between NFIP claims and

1. Same zipcode wetlands
2. Wetlands upstream of the zipcode (that could help prevent flood damage)
3. Wetlands upstream or downstream of the zipcode

We'll add in state fixed effects and control for population, housing value, and income:

Q: Using `feols()`, how can we write this?

---

1. Developed area (hectares)
2. Indicators for developed area in each of the following bins:
   - [Min, 1st Quartile), [1st Quartile, Median), [Median, 3rd Quartile), [3rd Quartile, Max]

Regression Output

We *could* write a bunch of code using **conditional logic:**

```
wl_df ← mutate(wl_df,
                dev_area_1 = ifelse(developed_ha < quantile(developed_ha,
                dev_area_2 = ifelse(developed_ha ⩾ quantile(developed_ha
                dev_area_3 = ifelse(developed_ha ⩾ quantile(developed_ha
                dev_area_4 = ifelse(developed_ha ⩾ quantile(developed_ha
                )

select(wl_df, developed_ha, starts_with("dev_")) %>% summarise(across(whe
```

```
## # A tibble: 1 × 10
##   developed_ha_1 developed_ha_2 dev_area_1_1 dev_area_1_2 dev_area_2_1
##            <dbl>          <dbl>        <dbl>        <dbl>        <dbl>
## 1              0          7339.            0            1            0
## # ℹ 5 more variables: dev_area_2_2 <dbl>, dev_area_3_1 <dbl>,
## #   dev_area_3_2 <dbl>, dev_area_4_1 <dbl>, dev_area_4_2 <dbl>
```

Bins

Alternatively, we could use the `percent_rank()` function in **tidyverse**:

```
wl_df <- mutate(wl_df,
                dev_area_1b = ifelse(percent_rank(developed_ha) < 0.25, 1
                dev_area_2b = ifelse(percent_rank(developed_ha) >= 0.25 &
)

identical(wl_df$dev_area_1, wl_df$dev_area_1b)
```

```
## [1] TRUE
```

```
identical(wl_df$dev_area_2, wl_df$dev_area_2b)
```

```
## [1] TRUE
```

Regression Output

1. Wetland area (hectares)
2. Indicators for developed area in each of the following bins:
    - [Min, 1st Quartile), [1st Quartile, Median), [Median, 3rd Quartile), [3rd Quartile, Max]
3. We'll add in state fixed effects and control for population, housing value, and income:
4. Robust SE

Let's specify our regression:

```
out_reg <- feols(claims ~ wetland_ha + dev_area_1 + dev_area_2 + dev_area_
                   population + housing_value + income | state_fips, data
```

Regression Output

Looking at our regression output:

```
summary(out_reg)
```

```
## OLS estimation, Dep. Var.: claims
## Observations: 124,386
## Fixed-effects: state_fips: 52
## Standard-errors: Heteroskedasticity-robust
##                      Estimate   Std. Error   t value        Pr(>|t|)
## wetland_ha          -5.664510     3.038201 -1.864429 0.062263805352 .
## dev_area_1      -76130.485716 48398.803102 -1.572983 0.115725341261
## dev_area_2      -95947.564283 46054.100961 -2.083366 0.037219880682 *
## dev_area_3     -110658.734494 43580.480628 -2.539181 0.011112435541 *
## population           7.727624     1.632395  4.733920 0.000002204670 ***
## housing_value        0.638841     0.118570  5.387890 0.000000071419 ***
## income              -0.005189     0.585438 -0.008864 0.992927660910
## ... 1 variable was removed because of collinearity (dev_area_4)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 3,217,539.3     Adj. R2: 0.016317
```

Bins

Even more alternatively, we could use the `bins()` function to first add a bin membership factor variable:

```
bins <- bin(wl_df$developed_ha, "cut::p25[p50[p75[p100]")
wl_df <- mutate(wl_df, dev_bins = bins)
head(wl_df$dev_bins)
```

```
## [1] [0.000; 77.700] [0.000; 77.700] [0.000; 77.700] [0.000; 77.700]
## [5] [0.000; 77.700] [0.000; 77.700]
## 4 Levels: [0.000; 77.700] [77.702; 258.417] ... [793.156; 7339.131]
```

```
quantile(wl_df$developed_ha, 0.25)
```

```
##      25%
## 77.70124
```

Bins

And then use the `i()` helper function in `feols()` to treat the variable as a factor:

```r
out_reg <- feols(claims ~ wetland_ha +
                   i(dev_bins, ref = "[793.156; 7339.131]") +
                   population + housing_value + income | state_fips, data = wl_df, vcov = "h
coeftable(out_reg)
```

```
##                                    Estimate     Std. Error      t value
## wetland_ha                         -5.664510197      3.0382007 -1.864429234
## dev_bins::[0.000; 77.700]      -76130.485715756 48398.8031022 -1.572982819
## dev_bins::[77.702; 258.417]    -95947.564282775 46054.1009610 -2.083366351
## dev_bins::[258.421; 793.151] -110658.734494468 43580.4806275 -2.539181140
## population                          7.727624318      1.6323945  4.733919570
## housing_value                       0.638840970      0.1185698  5.387890316
## income                             -0.005189323      0.5854383 -0.008863996
##                                       Pr(>|t|)
## wetland_ha                    0.06226380535185
## dev_bins::[0.000; 77.700]     0.11572534126138
## dev_bins::[77.702; 258.417]   0.03721988068209
## dev_bins::[258.421; 793.151]  0.01111243554068
## population                    0.00000220467047
## housing_value                 0.00000007141864
## income                        0.99292766090955
```

Bins

We can manually change the reference category by adding the `ref` argument and the variable value to omit:

```
feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339.131]") +
                   population + housing_value + income | state_fips, data
  summary()
```

```
## OLS estimation, Dep. Var.: claims
## Observations: 124,386
## Fixed-effects: state_fips: 52
## Standard-errors: Heteroskedasticity-robust
##                                     Estimate   Std. Error   t value
## wetland_ha                         -5.664510     3.038201 -1.864429
## dev_bins::[0.000; 77.700]     -76130.485716 48398.803102 -1.572983
## dev_bins::[77.702; 258.417]   -95947.564283 46054.100961 -2.083366
## dev_bins::[258.421; 793.151] -110658.734494 43580.480628 -2.539181
## population                          7.727624     1.632395  4.733920
## housing_value                       0.638841     0.118570  5.387890
## income                             -0.005189     0.585438 -0.008864
##                                     Pr(>|t|)
```

Regression Table

To format our data as a regression table we have two choices

1. Use **stargazer** on the summary dataframe (or `lm` object)
   - Works well for summary statistics tables too
2. Use `etable()` to create a well-formatted regression table
   - Works on one or multiple `fixest` objects

Stargazer

The `stargazer` package has great functionality for obtaining tables in HTML, LaTeX, or plain text format

- Also the preferred method for regression tables with `lm()` or `felm()` objects
- Doesn't play nice with `fixest` objects

For example, let's say you want to produce a **summary statistics table** for our included variables:

```
sumstats <- select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %>
  stargazer(type ="text")
```

```
##
## ==================================
## Statistic N Mean St. Dev. Min Max
## ==================================
```

Stargazer

This empty table reveals one restriction of `stargazer`: it requires the input be formatted as a **data frame**.

```r
sumstats <- select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %>
  as.data.frame() %>%
  stargazer(type = "text")
```

```
##
## ===================================================================
## Statistic      N      Mean       St. Dev.     Min        Max
## -------------------------------------------------------------------
## claims      131,588 109,330.100 3,155,095.000 0.000 418,203,035.000
## wetland_ha  131,588  1,301.656    4,533.807   0.000   261,518.300
## dev_area_1  131,588    0.250        0.433       0          1
## dev_area_2  131,588    0.250        0.433       0          1
## dev_area_3  131,588    0.250        0.433       0          1
## dev_area_4  131,588    0.250        0.433       0          1
## dev_area_1b 131,588    0.250        0.433       0          1
## dev_area_2b 131,588    0.250        0.433       0          1
## -------------------------------------------------------------------
```

Stargazer

We can easily get LaTeX output:

```
sumstats <- select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %
  as.data.frame() %>%
  stargazer(type = "latex")
```

```
##
## % Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institut
## % Date and time: Thu, Mar 14, 2024 - 12:52:45 PM
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{
## \hline \\[-1.8ex]
## claims & 131,588 & 109,330.100 & 3,155,095.000 & 0.000 & 418,203,035.000 \\
## wetland\_ha & 131,588 & 1,301.656 & 4,533.807 & 0.000 & 261,518.300 \\
## dev\_area\_1 & 131,588 & 0.250 & 0.433 & 0 & 1 \\
```

Stargazer

Or HTML:

```
sumstats <- select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %
  as.data.frame() %>%
  stargazer(type = "html")
```

Statistic	N	Mean	St. Dev.	Min	Max
claims	131,588	109,330.100	3,155,095.000	0.000	418,203,035.000
wetland_ha	131,588	1,301.656	4,533.807	0.000	261,518.300
dev_area_1	131,588	0.250	0.433	0	1
dev_area_2	131,588	0.250	0.433	0	1
dev_area_3	131,588	0.250	0.433	0	1
dev_area_4	131,588	0.250	0.433	0	1
dev_area_1b	131,588	0.250	0.433	0	1

Stargazer

Built-in styles also allow you to easily replicate the appearance of tables in several popular journals

```
sumstats ← select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %
  as.data.frame() %>%
  stargazer(style = "aer", type = "html")
```

Statistic	N	Mean	St. Dev.	Min	Max
claims	131,588	109,330.100	3,155,095.000	0.000	418,203,035.000
wetland_ha	131,588	1,301.656	4,533.807	0.000	261,518.300
dev_area_1	131,588	0.250	0.433	0	1
dev_area_2	131,588	0.250	0.433	0	1
dev_area_3	131,588	0.250	0.433	0	1
dev_area_4	131,588	0.250	0.433	0	1

Stargazer

Built-in styles also allow you to easily replicate the appearance of tables in several popular journals

```
sumstats ← select(wl_df, claims, wetland_ha, starts_with("dev_area_")) %
  as.data.frame() %>%
  stargazer(style = "qje", type = "html")
```

Statistic	N	Mean	St. Dev.	Min	Max
claims	131,588	109,330.100	3,155,095.000	0.000	418,203,035.000
wetland_ha	131,588	1,301.656	4,533.807	0.000	261,518.300
dev_area_1	131,588	0.250	0.433	0	1
dev_area_2	131,588	0.250	0.433	0	1
dev_area_3	131,588	0.250	0.433	0	1
dev_area_4	131,588	0.250	0.433	0	1

Stargazer

We can also use stargazer to build regression tables directly with `lm` output:

```r
lm1 ← lm(claims ~ wetland_ha, data = wl_df)
lm2 ← lm(claims ~ wetland_ha + population, data = wl_df)
lm3 ← lm(claims ~ wetland_ha + population + housing_value, data = wl_df)

stargazer(lm1, lm2, lm3, style = "aer", type = "html")
```

	claims		
	(1)	(2)	(3)
wetland_ha	5.948^{***}	6.758^{***}	7.492^{***}
	(1.918)	(1.932)	(2.035)
population		10.203^{***}	9.821^{***}
		[illegible]	[illegible]

Stargazer

`stargazer` has a full suite of options for adding titles, notes, changing variable names, column labeling, etc.

```r
stargazer(lm1, lm2, lm3, style = "aer", type = "html",
          title = "NFIP Claims and Wetland Area", # chart title
          dep.var.labels = c("NFIP Claims Amount"), # label dep. variable
          covariate.labels = c("Wetland Area (Ha)", "Population", "Housin
          ci = c(T,T,T), # replace std errors with conf. intervals
          digits = 4, # include 4 digits
          initial.zero = T, # add intitial zero for values 0-1
          label = "table_ols", # latex label
          note = c("This Table reports regression results of NFIP claims
          out = "output/stargazer_table.html" # path to save out the tabl
          )
```

Stargazer

NFIP Claims and Wetland Area

	NFIP Claims Amount		
	(1)	(2)	(3)
Wetland Area (Ha)	5.9476***	6.7584***	7.4920***
	(2.1877, 9.7075)	(2.9726, 10.5442)	(3.5032, 11.4808)
Population		10.2027***	9.8212***
		(8.9614, 11.4439)	(8.4968, 11.1455)
Housing Value			0.1361**
			(0.0087, 0.2636)
Constant	101,588.3000***	4,782.7480	-14,013.8800

etable

While we can't use stargazer with `fixest` objects, the package provides the equivalent `etable()` function that allows for fully professional regression tables.

Suppose we wanted to show the value of our wetland coefficient across several different fixed effects specifications:

```
reg_state ← feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339
                   population + housing_value + income | state_fips, data
reg_cty ← feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339.
                   population + housing_value + income | county_fips, dat
reg_styr ← feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339
                   population + housing_value + income | state_fips + yea
reg_ctyr ← feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339
                   population + housing_value + income | county_fips + yea
```

etable

Putting together the regression table:

```
etable(reg_state, reg_cty, reg_styr, reg_ctyr,
       title = "Fixed Effects Progression",
       fixef_sizes = T,
       digits = 4, # digits for coefficients and std. errors
       digits.stats = 4, # digits for fit stats
       tex = F # whether output is LaTex (T) or a df (F)
       )
```

```
##                                             reg_state               reg_cty
## Dependent Var.:                                claims                claims
##
## wetland_ha                            -5.665. (3.038)        -3.251 (2.529)
## dev_bins = [0.000;77.700]        -76,130.5 (48,398.8)  -63,319.6 (57,328.2)
## dev_bins = [77.702;258.417]     -95,947.6* (46,054.1)  -74,749.1 (53,209.3)
## dev_bins = [258.421;793.151] -110,658.7* (43,580.5) -98,888.3. (53,014.5)
## population                           7.728*** (1.632)       6.101** (2.189)
## housing_value                      0.6388*** (0.1186)     0.3384** (0.1082)
## income                               -0.0052 (0.5854)      1.676** (0.5679)
```

etable

Another convenient feature of `etable`: we can **change standard error estimators** on the fly!

```
etable(reg_state, reg_cty, reg_styr, reg_ctyr,
       title = "Fixed Effects Progression",
       fixef_sizes = T,
       digits = 4, # digits for coefficients and std. errors
       digits.stats = 4, # digits for fit stats
       tex = F, # whether output is LaTex (T) or a df (F)
       se = "standard"
       )
```

```
##                                              reg_state                reg_cty
## Dependent Var.:                                 claims                 claims
##
## wetland_ha                           -5.665** (2.178)         -3.251 (2.758)
## dev_bins = [0.000;77.700]       -76,130.5* (38,511.7)   -63,319.6 (42,225.7)
## dev_bins = [77.702;258.417]    -95,947.6** (35,421.5)  -74,749.1. (38,261.3)
## dev_bins = [258.421;793.151] -110,658.7*** (31,613.6) -98,888.3** (33,363.8)
## population                           7.728*** (1.017)       6.101*** (1.180)
```

etable

More efficient: take advantage of **stepwise functions** for multiple estimation:

```
regs_fe ← feols(claims ~ wetland_ha + i(dev_bins, ref = "[793.156; 7339.1
                   population + housing_value + income |
                  sw0(state_fips, county_fips, state_fips + year, county_
                  data = wl_df, vcov = "hetero")
etable(regs_fe)
```

```
##                                            regs_fe.1              regs_fe.2
## Dependent Var.:                               claims                 claims
##
## Constant                        142,577.5* (65,774.1)
## wetland_ha                           6.198** (1.792)       -5.665. (3.038)
## dev_bins = [0.000;77.700]     -181,068.7** (53,079.4)  -76,130.5 (48,399.0)
## dev_bins = [77.702;258.417]   -183,517.4** (50,834.8) -95,947.6* (46,054.3)
## dev_bins = [258.421;793.151] -165,192.3** (46,304.8) -110,658.7* (43,580.7)
## population                           5.807** (1.566)       7.728** (1.632)
## housing_value                        0.1360 (0.0894)     0.6388** (0.1186)
## income                               0.2983 (0.5802)      -0.0052 (0.5854)
```

etable

The `dict` argument also makes it straightforward to replace variable names:

```
etable(regs_fe,
       dict = c(wetland_ha = "Wetland Area (Ha)",
                dev_bins = "Developed Area",
                state_fips = "State",
                county_fips = "County",
                year = "Year"))
```

```
##                                                   regs_fe.1
## Dependent Var.:                                      claims
##
## Constant                              142,577.5* (65,774.1)
## Wetland Area (Ha)                          6.198*** (1.792)
## Developed Area = [0.000;77.700]    -181,068.7*** (53,079.4)
## Developed Area = [77.702;258.417]  -183,517.4*** (50,834.8)
## Developed Area = [258.421;793.151] -165,192.3*** (46,304.8)
## population                                 5.807*** (1.566)
## housing_value                               0.1360 (0.0894)
```

etable

You can also set up a default dictionary with `setFixest_dict()` that can be used in all following tables.

```
# set dictionary for use across the entire script
setFixest_dict(c(wetland_ha = "Wetland Area (Ha)",
                dev_bins = "Developed Area",
                state_fips = "State",
                county_fips = "County",
                year = "Year"))

# Variables are automatically re-labeled
etable(regs_fe)
```

```
##                                                    regs_fe.1
## Dependent Var.:                                       claims
##
## Constant                               142,577.5* (65,774.1)
## Wetland Area (Ha)                           6.198*** (1.792)
## Developed Area = [0.000;77.700]    -181,068.7*** (53,079.4)
## Developed Area = [77.702;258.417]  -183,517.4*** (50,834.8)
```

etable

This is only a **fraction of the settings** available. Don't worry, we'll get more practice with them in the remainder of the lecture (and in the next problem set)!

IV Regression

IV Regression

`fixest` also makes it easy to estimate IV regressions.

All we need is to add our IV formula after a second `|`

```
feols(y ~ x_1 + ...  + X_n | fe_1 +  ...  + fe_n | x_endo1 + x_endo2 ~
x_inst1 + x_inst2, data = df)
```

- `x_endo`: the endogenous regressor(s)
- `x_inst`: the instrument(s)

IV Regression

Let's suppose that wetland area in 2016 was endogenous due to recent policies, but that the area in 2001 was a valid instruments[2]. Using the `did_df` data:

```
iv_1 <- feols(claims_2016_3yr ~ developed_ha_2016 + population_2016 +
              income_2016 + housing_value_2016 |
              state_fips | wetland_ha_2016 ~ wetland_ha_2001,
              data = did_df)
```

[2] This is purely a toy example - there isn't an endogeneity issue in the paper.

IV Regression

By default, `summary()` now reports the **second stage results**.

```
summary(iv_1)
```

```
## TSLS estimation - Dep. Var.: claims_2016_3yr
##                   Endo.    : wetland_ha_2016
##                   Instr.   : wetland_ha_2001
## Second stage: Dep. Var.: claims_2016_3yr
## Observations: 25,006
## Fixed-effects: state_fips: 49
## Standard-errors: Clustered (state_fips)
##                       Estimate Std. Error  t value Pr(>|t|)
## fit_wetland_ha_2016   9.024196   8.350857  1.08063  0.28526
## developed_ha_2016   103.424292  90.923739  1.13748  0.26098
## population_2016      14.942096  10.430715  1.43251  0.15848
## income_2016           4.731703   3.268778  1.44754  0.15424
## housing_value_2016   -0.402308   0.273227 -1.47243  0.14743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 2,652,351.7     Adj. R2: 0.04286
```

IV Regression

We can get **first-stage results** with the `stage = 1` argument:

```r
summary(iv_1, stage = 1)
```

```
## TSLS estimation - Dep. Var.: wetland_ha_2016
##                   Endo.    : wetland_ha_2016
##                   Instr.   : wetland_ha_2001
## First stage: Dep. Var.: wetland_ha_2016
## Observations: 25,006
## Fixed-effects: state_fips: 49
## Standard-errors: Clustered (state_fips)
##                     Estimate Std. Error    t value  Pr(>|t|)
## wetland_ha_2001     1.000634   0.000810 1235.28173 < 2.2e-16 ***
## developed_ha_2016  -0.012556   0.004855   -2.58637  0.012788 *
## population_2016     0.000255   0.000206    1.23581  0.222544
## income_2016        -0.000074   0.000039   -1.86483  0.068325 .
## housing_value_2016 -0.000015   0.000011   -1.43371  0.158139
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 102.7     Adj. R2: 0.999458
```

IV Regression

Or both at the same time with `stage = 2:1` or `1:2`

```
summary(iv_1, stage = 1:2)
```

```
## IV: First stage: wetland_ha_2016
## TSLS estimation - Dep. Var.: wetland_ha_2016
##                   Endo.    : wetland_ha_2016
##                   Instr.   : wetland_ha_2001
## First stage: Dep. Var.: wetland_ha_2016
## Observations: 25,006
## Fixed-effects: state_fips: 49
## Standard-errors: Clustered (state_fips)
##                      Estimate Std. Error    t value  Pr(>|t|)
## wetland_ha_2001      1.000634   0.000810 1235.28173 < 2.2e-16 ***
## developed_ha_2016  -0.012556   0.004855   -2.58637  0.012788 *
## population_2016      0.000255   0.000206    1.23581  0.222544
## income_2016        -0.000074   0.000039   -1.86483  0.068325 .
## housing_value_2016 -0.000015   0.000011   -1.43371  0.158139
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

IV Regression

And the output table (adding a few IV fit statistics):

```
etable(iv_1, stage = 1:2, fitstat = ~ . + ivfall + ivwaldall)
```

```
##                                 iv_1.1           iv_1.2
## IV stages                        First           Second
## Dependent Var.:         wetland_ha_2016  claims_2016_3yr
##
## wetland_ha_2001      1.001*** (0.0008)
## developed_ha_2016    -0.0126* (0.0049)    103.4 (90.92)
## population_2016        0.0003 (0.0002)    14.94 (10.43)
## income_2016        -7.36e-5. (3.95e-5)    4.732 (3.269)
## housing_value_2016  -1.52e-5 (1.06e-5) -0.4023 (0.2732)
## wetland_ha_2016                           9.024 (8.351)
## Fixed-Effects:     ------------------- ----------------
## State                              Yes              Yes
## __________________ ___________________ ________________
## S.E.: Clustered              by: State        by: State
## Observations                    25,006           25,006
## R2                             0.99946          0.04489
## Within R2                      0.99938          0.01183
## F-test (IV only)          38,825,154.0           4.7368
## Wald (IV only)             1,525,921.0           1.1678
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Difference-in-Differences

Difference-in-Differences

Let's now replicate some actual results from the paper: the **upstream/downstream diff-in-diff**.

$$\Delta F_{is} = \beta \Delta W_{is} + \gamma \Delta W_{is}^{UP} + \lambda \Delta W_{is}^{ALL} + \theta \Delta X_{is} + \alpha_s + \epsilon_{is}$$

- ΔF_{is}: the change in flood insurance claims from 2001 to 2016
- ΔW_{is}: the change in **within-zipcode** wetland area from 2001 to 2016
- ΔW_{is}^{UP}: the change in **upstream** wetland area from 2001 to 2016
- ΔW_{is}^{ALL}: the change in **upstream and downstream** wetland area from 2001 to 2016
- ΔX_{is}: the change in time-varying covariates from 2001 to 2016
- α_s state fixed effects

Difference-in-Differences

$$\Delta F_{is} = \beta \Delta W_{is} + \gamma \Delta W_{is}^{UP} + \lambda \Delta W_{is}^{ALL} + \theta \Delta X_{is} + \alpha_s + \epsilon_{is}$$

Diff-in-diff style coefficients[2] of interest are

- β: the effect of "local" wetlands (within the same zipcode)
- γ : the differential effect of upstream wetlands
  - i.e. the "direct protective services" of the wetland

Difference-in-Differences

$$\Delta F_{is} = \beta \Delta W_{is} + \gamma \Delta W_{is}^{UP} + \lambda \Delta W_{is}^{ALL} + \theta \Delta X_{is} + \alpha_s + \epsilon_{is}$$

```
did_reg ← feols(claims_2001_2016 ~ wetland_2001_2016 +
                      wetlands_change_up + wetlands_change_all +
                      developed_2001_2016 +
                      income_2001_2016 +
                      population_2001_2016 +
                      housing_units_2001_2016 +
                      housing_value_2001_2016 +
                      CRS_2001_2016
                    |state_fips, cluster = ~county_fips, data = did_df)
```

Difference-in-Differences

$$\Delta F_{is} = \beta \Delta W_{is} + \gamma \Delta W_{is}^{UP} + \lambda \Delta W_{is}^{ALL} + \theta \Delta X_{is} + \alpha_s + \epsilon_{is}$$

```
coeftable(did_reg)
```

```
##                               Estimate    Std. Error    t value    Pr(>|t|)
## wetland_2001_2016         -157.7758561    102.112020 -1.5451252 0.122423316
## wetlands_change_up        -499.9636949    211.760190 -2.3609900 0.018290952
## wetlands_change_all        -21.0437746     15.282017 -1.3770287 0.168607941
## developed_2001_2016        337.5158492    166.667757  2.0250818 0.042948914
## income_2001_2016            -0.5575620      1.188346 -0.4691917 0.638967340
## population_2001_2016        -5.5928337     11.429141 -0.4893486 0.624631357
## housing_units_2001_2016     76.3330446     29.054958  2.6271951 0.008653789
## housing_value_2001_2016      0.2008849      0.305430  0.6577117 0.510774778
## CRS_2001_2016           169617.7738679 117832.013949  1.4394880 0.150118470
## attr(,"type")
## [1] "Clustered (county_fips)"
```

Difference-in-Differences

Comparing to the published paper results (Column 2):

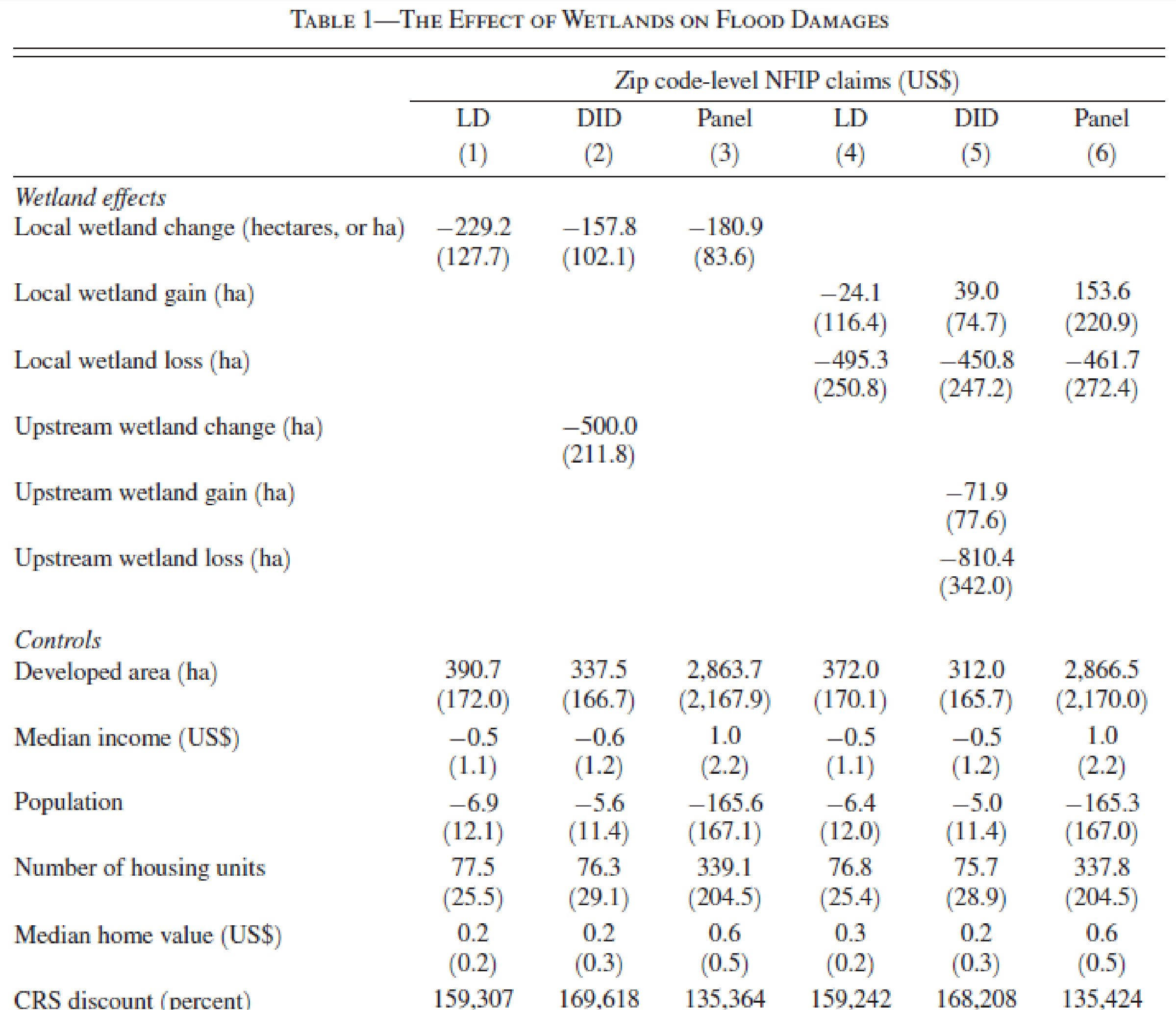


Table 1—The Effect of Wetlands on Flood Damages

	Zip code-level NFIP claims (US$)					
	LD	DID	Panel	LD	DID	Panel
	(1)	(2)	(3)	(4)	(5)	(6)
Wetland effects						
Local wetland change (hectares, or ha)	–229.2	–157.8	–180.9			
	(127.7)	(102.1)	(83.6)			
Local wetland gain (ha)				–24.1	39.0	153.6
				(116.4)	(74.7)	(220.9)
Local wetland loss (ha)				–495.3	–450.8	–461.7
				(250.8)	(247.2)	(272.4)
Upstream wetland change (ha)		–500.0				
		(211.8)				
Upstream wetland gain (ha)					–71.9	
					(77.6)	
Upstream wetland loss (ha)					–810.4	
					(342.0)	
Controls						
Developed area (ha)	390.7	337.5	2,863.7	372.0	312.0	2,866.5
	(172.0)	(166.7)	(2,167.9)	(170.1)	(165.7)	(2,170.0)
Median income (US$)	–0.5	–0.6	1.0	–0.5	–0.5	1.0
	(1.1)	(1.2)	(2.2)	(1.1)	(1.2)	(2.2)
Population	–6.9	–5.6	–165.6	–6.4	–5.0	–165.3
	(12.1)	(11.4)	(167.1)	(12.0)	(11.4)	(167.0)
Number of housing units	77.5	76.3	339.1	76.8	75.7	337.8
	(25.5)	(29.1)	(204.5)	(25.4)	(28.9)	(204.5)
Median home value (US$)	0.2	0.2	0.6	0.3	0.2	0.6
	(0.2)	(0.3)	(0.5)	(0.2)	(0.3)	(0.5)
CRS discount (percent)	159,307	169,618	135,364	159,242	168,208	135,424

Difference-in-Differences

Historically, we've thought mostly about whether **parallel trends** are likely to hold for validity of our difference-in-differences estimator.

Logic:

- Control and treated groups are trending similarly prior to treated group receiving treatment
- Ensures validity of control group's post-treatment trend as counterfactual for treatment group absent the treatment

Most often (and especially with staggered adoption timing), we estimate these models in the canonical two-way fixed effects specification (TWFE)

$$Y_{it} = \beta T_{it} + \delta_t + \alpha_i + \epsilon_{it}$$

Difference-in-Differences

However, recent literature has shown that even if parallel trends hold, our estimates might still be biased

- Due to unintended weighting of all possible 2x2 comparisons (**Goodman-Bacon (2021)**)
- Resulting weights may even be negative (**de Chaisemartin and D'Haultfoeuille (2020)**)

We'll get into some **alternate estimators** that overcome these limitations soon (and next lecture), but for now let's chat briefly about another situation that warrants a different empirical specification: **dynamic treatment effects**

Event Study and Dynamic Treatment Effects

Dynamic Treatment Effects

Recall the TWFE diff-in-diff model:

$$Y_{it} = \beta T_{it} + \delta_t + \alpha_i + \epsilon_{it}$$

This yields an estimate of β, the **average treatment effect for the treated (ATT)**.

- Average across all treated units
- Average across all post-treatment time periods

However, there are plenty of settings where we might expect **dynamic treatment effects**: treatment effects that differ by the amount of time since receiving treatment.

Event Study

This is where **event study**[2] comes in.

Idea: instead of a single $\hat{\beta}$ estimate a vector of treatment effects $\hat{\beta}^{\boldsymbol{k}}$

- Each coefficient estimates a period-specific ATT for being $\boldsymbol{k}$ time periods relative to receiving treatment
    - i.e. ATT 1 day post-treatment may look quite different than 100 days post-treatment

Estimating this requires thinking about time in an alternate fashion (no, this isn't a multiverse situation)...

[2] Note that the "original" event study methods refer to techniques in finance related to the impact of various "events" on stock prices, generally for a single asset. The event studies we're talking about today refer to the approach used in applied economics for estimating dynamic treatment paths.

Time-to-Event Data

To work through this, let's use some data I'm pretty familiar with: COVID-19 stay-at-home mandate data.

```
sah <- read_csv("data/sah.csv") %>%
  drop_na(cadt) %>%
  select(state, date, weekday, cadt, visits, mandate_date)

head(sah)
```

```
## # A tibble: 6 × 6
##   state date       weekday  cadt visits mandate_date
##   <chr> <date>       <dbl> <dbl>  <dbl> <date>
## 1 AK    2020-02-24       2  3.19  -2.14 2020-03-29
## 2 AK    2020-02-25       3  2.38   2.01 2020-03-29
## 3 AK    2020-02-26       4  7.36   2.94 2020-03-29
## 4 AK    2020-02-27       5  6.97   5.25 2020-03-29
## 5 AK    2020-02-28       6  6.40   6.50 2020-03-29
## 6 AK    2020-02-29       7  1.53  10.0  2020-03-29
```

Aside: Diff-in-Diff

Here we have information on the unit (`state`), the time period (`date`), and the date of treatment (`mandate_date`).

- States adopted mandates at different times $\Rightarrow$ **staggered adoption**

To estimate the TWFE diff-in-diff, we just need a treatment indicator

- $= 1$ for treated units after treatment occurred
  - mandate-adopting states post-mandate adoption
- $= 0$ for treated units prior to treatment
  - mandate-adopting states prior to mandate adoption
- $= 0$ for control units
  - states that never adopted a statewide stay-at-home mandate

Aside: Diff-in-Diff

To estimate the TWFE diff-in-diff, we just need a treatment indicator.

First, identify which states ever adopted a stay-at-home mandate:

```
sah_did ←   group_by(sah, state) %>%
  # get state-specific max value of mandate adoption date
  dplyr::mutate(sah_max = max(mandate_date, na.rm = T))
```

This should yield a date for adopting states, and `-Inf` for non-adopters (i.e. pure controls)

```
unique(sah_did$sah_max)
```

```
##  [1] "2020-03-29" "2020-04-05" "-Inf"       "2020-04-01" "2020-03-19"
##  [6] "2020-03-26" "2020-03-24" "2020-04-03" "2020-03-25" "2020-03-22"
## [11] "2020-03-30" "2020-03-27" "2020-03-31" "2020-04-02" "2020-03-28"
## [16] "2020-04-06" "2020-04-04" "2020-03-23" "2020-04-08"
```

Aside: Diff-in-Diff

Next, determine whether we are in the post-adoption period for our adopting states (i.e. our treatment dummy):

```
# note: data are still grouped by state
sah_did <- dplyr::mutate(sah_did,
      post_treat = case_when(
        sah_max == -Inf ~ 0, # 0 for pure controls
        date >= mandate_date ~ 1, # 1 for adopting states in post-period
        TRUE ~ 0 # 0 for adopting states pre-treatment
      )
      ) %>%
  ungroup()

summary(sah_did$post_treat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4204  1.0000  1.0000
```

Aside: Diff-in-Diff

Estimating the TWFE diff-in-diff for the impact of stay-at-home on average distance traveled:

```
did_1 ← feols(cadt ~ post_treat | state + date, data = sah_did, cluster =
summary(did_1)
```

```
## OLS estimation, Dep. Var.: cadt
## Observations: 3,366
## Fixed-effects: state: 51,  date: 66
## Standard-errors: Clustered (state)
##             Estimate Std. Error  t value     Pr(>|t|)
## post_treat -5.50772    1.04595 -5.26575 0.0000029507 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 5.22448     Adj. R2: 0.929189
##                 Within R2: 0.059987
```

Aside: Diff-in-Diff

Comparing to results from **Sears et al. (2023)** (Column 5):

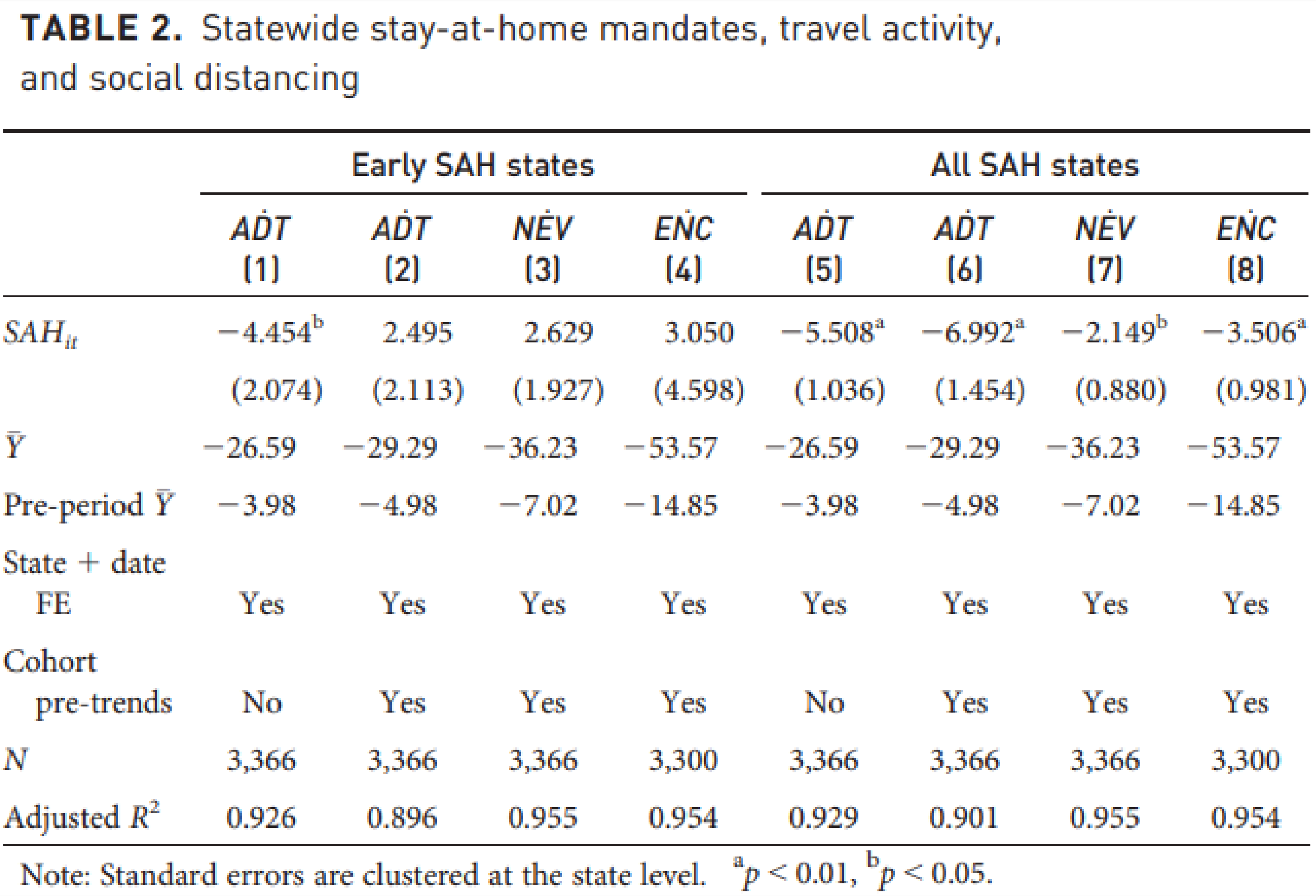


TABLE 2. Statewide stay-at-home mandates, travel activity, and social distancing

	Early SAH states				All SAH states			
	$A\dot{D}T$ (1)	$A\dot{D}T$ (2)	$N\dot{E}V$ (3)	$E\dot{N}C$ (4)	$A\dot{D}T$ (5)	$A\dot{D}T$ (6)	$N\dot{E}V$ (7)	$E\dot{N}C$ (8)
SAH_{it}	-4.454^{b}	2.495	2.629	3.050	-5.508^{a}	-6.992^{a}	-2.149^{b}	-3.506^{a}
	(2.074)	(2.113)	(1.927)	(4.598)	(1.036)	(1.454)	(0.880)	(0.981)
$\bar{Y}$	−26.59	−29.29	−36.23	−53.57	−26.59	−29.29	−36.23	−53.57
Pre-period $\bar{Y}$	−3.98	−4.98	−7.02	−14.85	−3.98	−4.98	−7.02	−14.85
State + date FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Cohort pre-trends	No	Yes	Yes	Yes	No	Yes	Yes	Yes
N	3,366	3,366	3,366	3,300	3,366	3,366	3,366	3,300
Adjusted R^2	0.926	0.896	0.955	0.954	0.929	0.901	0.955	0.954

Note: Standard errors are clustered at the state level. $^{a}p < 0.01$, $^{b}p < 0.05$.

Diff-in-Diff: Weighting Issues

If we decompose the overall ATT into each 2x2 estimator, we can see can get more information as to which comparisons are driving our treatment effects:

```
pacman::p_load(bacondecomp)

# first, get outcome residualized of fixed effects
sah_did ← sah_did %>%
  dplyr::mutate(cadt_r = feols(cadt ~ 0 | state + date, data = sah_did)$re
                date_num = as.numeric(date))


# run decomp of residualized outcome on treatment
decomp ← bacon(cadt_r ~ post_treat,
                     data = sah_did,
                     id_var = "state",
                     time_var = "date_num",
                quietly = T) %>%
      mutate(full_att = sum(estimate*weight))
```

Aside: Bacon Decomposition

Looking at the different weights:

```
decomp_tab <- decomp %>%
    dplyr::select(type, weight, estimate, full_att) %>%
      dplyr::mutate(att = weight*estimate) %>%
      group_by(type)  %>%
      mutate(group_weight = sum(weight)) %>%
      mutate(group_att = sum(att)/group_weight) %>%
      summarise(weight = sum(weight),
                estimate = mean(estimate),
                att = mean(group_att),
                count = n(),
                overall_att = mean(full_att))
decomp_tab
```

```
## # A tibble: 3 × 6
##   type                     weight estimate   att count overall_att
##   <chr>                     <dbl>    <dbl> <dbl> <int>       <dbl>
## 1 Earlier vs Later Treated  0.223    -4.36 -3.61   153       -5.51
## 2 Later vs Earlier Treated  0.213    -2.94 -2.62   153       -5.51
## 3 Treated vs Untreated      0.564    -6.23 -7.35    18       -5.51
```

Aside: Bacon Decomposition

Which shows us that the overall ATT estimated from the difference-in-differences is equal to

$$\boldsymbol{.23\hat{\beta}^{Early\ vs.\ Late} + 0.22\hat{\beta}^{Late\ vs.\ Early} + 0.56\hat{\beta}^{Treated\ vs.\ Untreated}}$$

```
# Recover overall ATT as weighted combination of 2x2 comparisons:

sum(decomp_tab$weight*decomp_tab$att)
```

```
## [1] -5.507721
```

Comparing to the diff-in-diff ATT:

```
coeftable(did_1)
```

```
##             Estimate Std. Error   t value       Pr(>|t|)
## post_treat -5.507721   1.045952 -5.265751 0.000002950744
## attr(,"type")
## [1] "Clustered (state)"
```

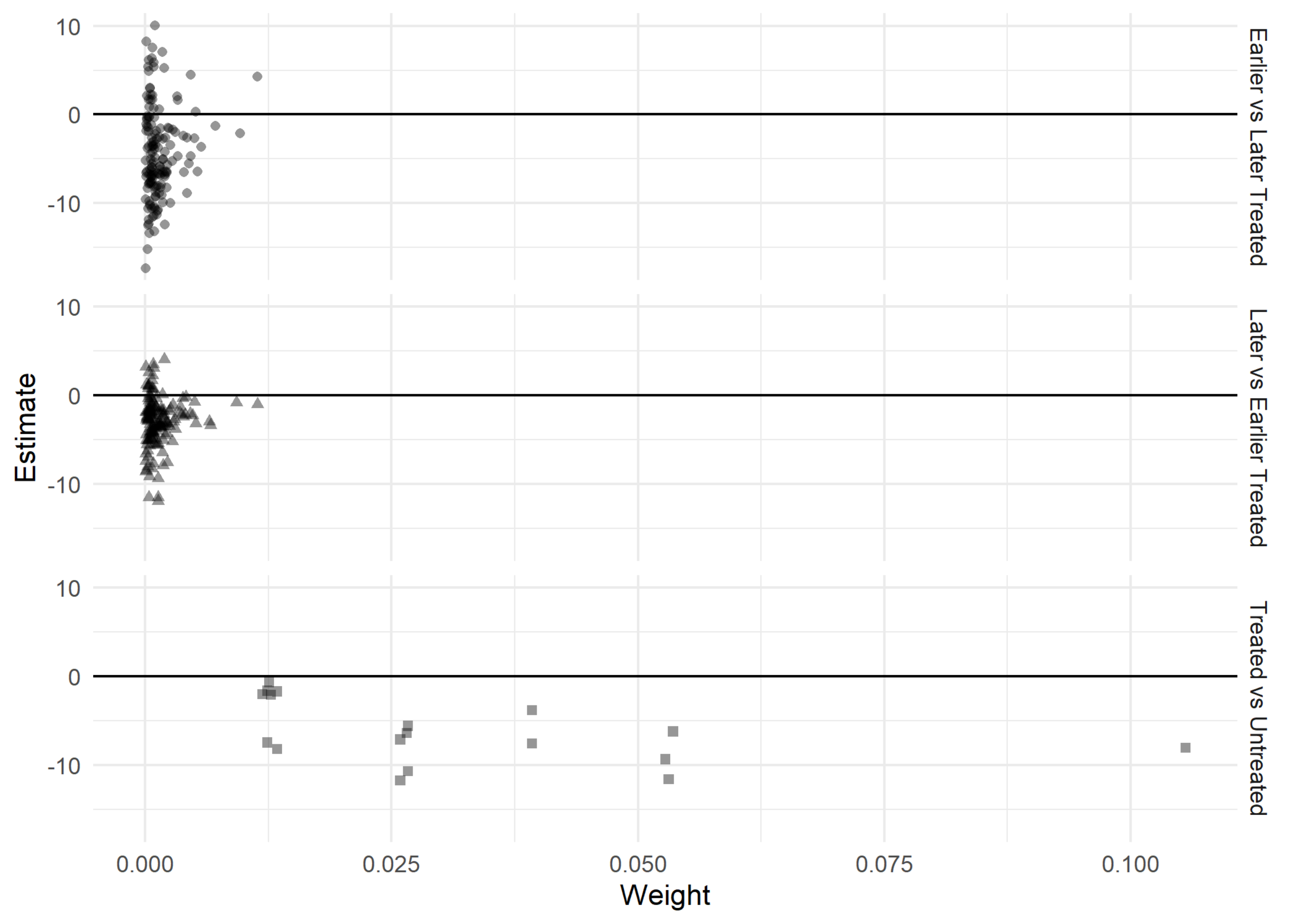
Aside: Bacon Decomposition

We can also plot the individual 2x2 estimates to see how much heterogeneity there is:

```r
ggplot(decomp) +
      aes(shape = factor(type)) +
      geom_point( aes(x = weight, y = estimate), alpha = 0.4) +
      geom_hline(yintercept = 0) +
      labs(x = "Weight", y = "Estimate", shape = "Type") +
      theme_minimal() +
      theme(legend.position = "none") +
      facet_grid(rows = vars(type), space = "fixed", scales = "fixed")
```

Aside: Bacon Decomposition

We can also plot the individual 2x2 estimates to see how much heterogeneity there is:



Aside: Bacon Decomposition

The decomposition reveals several things.

1. The most weight is given to the Treated vs. Pure Control comparison, which has the largest magnitude ATT
2. All Treated vs. Control estimates are negative
3. Lots of heterogeneity in Early vs. Late or Late vs. Early comparisons
    - Some are even positive!

Event Study

One thing that might be affecting these early/late comparisons is **dynamic treatment effects**

- We have more post-treatment observations for early adopters, fewer for late adopters
- If treatment response is dynamic, we miss that by averaging over the full post-treatment period

Estimating an event study will allow us to re-center our coefficients on **event time** and directly observe these dynamics.

Event Study

The first step is to code up an **event time** variable

- $= -7$ on the date 7 days prior to adoption (specific to each county)
- $= -1$ on the day prior to adoption
- $= 0$ on the date of mandate adoption
- $= 1$ on the first day after adoption
- $= 14$ on the date two weeks after adoption
- etc.

Event Time Variable

The first step is to code up an **event time** variable

```
sah_es ← group_by(sah_did, state) %>%
  mutate(event_time = case_when(
    # for treated states, set event_time by comparing date to mandate_dat
    !is.na(mandate_date) ~ as.numeric(date) - as.numeric(mandate_date),
    # for controls, set event time = Inf for control units
    TRUE ~ Inf
  ))
```

Event Time Variable

Checking an adopting state:

```r
filter(sah_es, state == "CA", event_time %in% -7:7) %>%
  select(state, date, mandate_date, event_time)
```

```
## # A tibble: 15 × 4
## # Groups:   state [1]
##    state date       mandate_date event_time
##    <chr> <date>     <date>            <dbl>
##  1 CA    2020-03-12 2020-03-19           -7
##  2 CA    2020-03-13 2020-03-19           -6
##  3 CA    2020-03-14 2020-03-19           -5
##  4 CA    2020-03-15 2020-03-19           -4
##  5 CA    2020-03-16 2020-03-19           -3
##  6 CA    2020-03-17 2020-03-19           -2
##  7 CA    2020-03-18 2020-03-19           -1
##  8 CA    2020-03-19 2020-03-19            0
##  9 CA    2020-03-20 2020-03-19            1
## 10 CA    2020-03-21 2020-03-19            2
## 11 CA    2020-03-22 2020-03-19            3
## 12 CA    2020-03-23 2020-03-19            4
## 13 CA    2020-03-24 2020-03-19            5
## 14 CA    2020-03-25 2020-03-19            6
## 15 CA    2020-03-26 2020-03-19            7
```

Event Time Variable

And a control state:

```
filter(sah_es, state == "AR") %>%
  select(state, date, mandate_date, event_time)
```

```
## # A tibble: 66 × 4
## # Groups:   state [1]
##    state date       mandate_date event_time
##    <chr> <date>     <date>            <dbl>
##  1 AR    2020-02-24 NA                  Inf
##  2 AR    2020-02-25 NA                  Inf
##  3 AR    2020-02-26 NA                  Inf
##  4 AR    2020-02-27 NA                  Inf
##  5 AR    2020-02-28 NA                  Inf
##  6 AR    2020-02-29 NA                  Inf
##  7 AR    2020-03-01 NA                  Inf
##  8 AR    2020-03-02 NA                  Inf
##  9 AR    2020-03-03 NA                  Inf
## 10 AR    2020-03-04 NA                  Inf
## # ℹ 56 more rows
```

Event Time

Perfect! We can now see the re-centering by looking at the distribution of treatment status relative to the **date**:

```r
date_freq <- sah_es %>%
  group_by(date) %>%
  summarise(count = sum(post_treat ==1)) %>%
  ggplot() +
    geom_col(aes(y = count, x = date), fill = "dodgerblue", alpha = 0.65)
  geom_vline(aes(xintercept = ymd("2020-03-19")), linetype = "dashed") +
  theme_minimal()
```

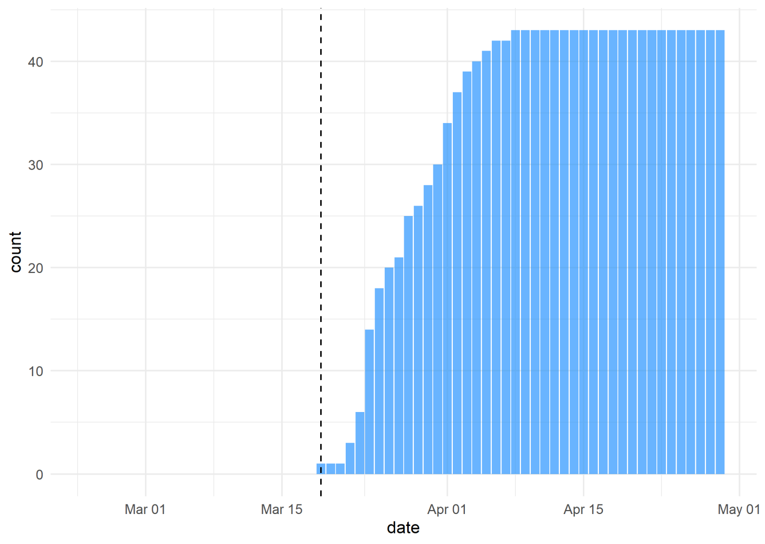
Event Time

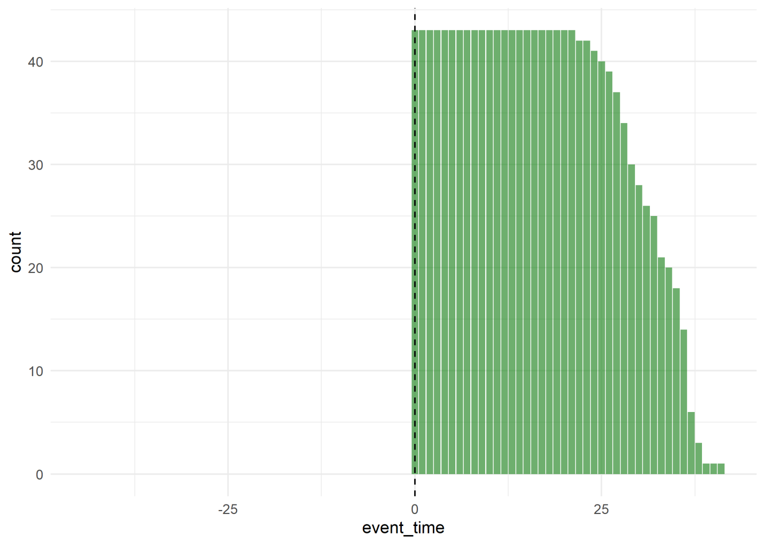
And the distribution of treatment status relative to **event-time**:

```r
eventtime_freq <- sah_es %>%
  group_by(event_time) %>%
  summarise(count = sum(post_treat ==1)) %>%
  filter(event_time < Inf) %>%
  ggplot() +
  geom_col(aes(y = count, x = event_time), fill = "forestgreen", alpha = 
  geom_vline(aes(xintercept = 0), linetype = "dashed") +
  theme_minimal()
```

Date-time vs. Event Time

And comparing the two shows the difference under staggered adoption





- By date: frequency of treated states increases over time as more states adopt stay-at-home mandates
- By event-time: flip from no treated to fully treated at event time 0
    - Frequencies fall off for later event times as we don't have as many post-treatment event times for late adopters

Binning Endpoints

Before we calculate our dynamic treatment vector, we might want to **bin the endpoints**

- Set every event-time ≤ -24 to 24
- Set every event-time ≥ -21 to 21

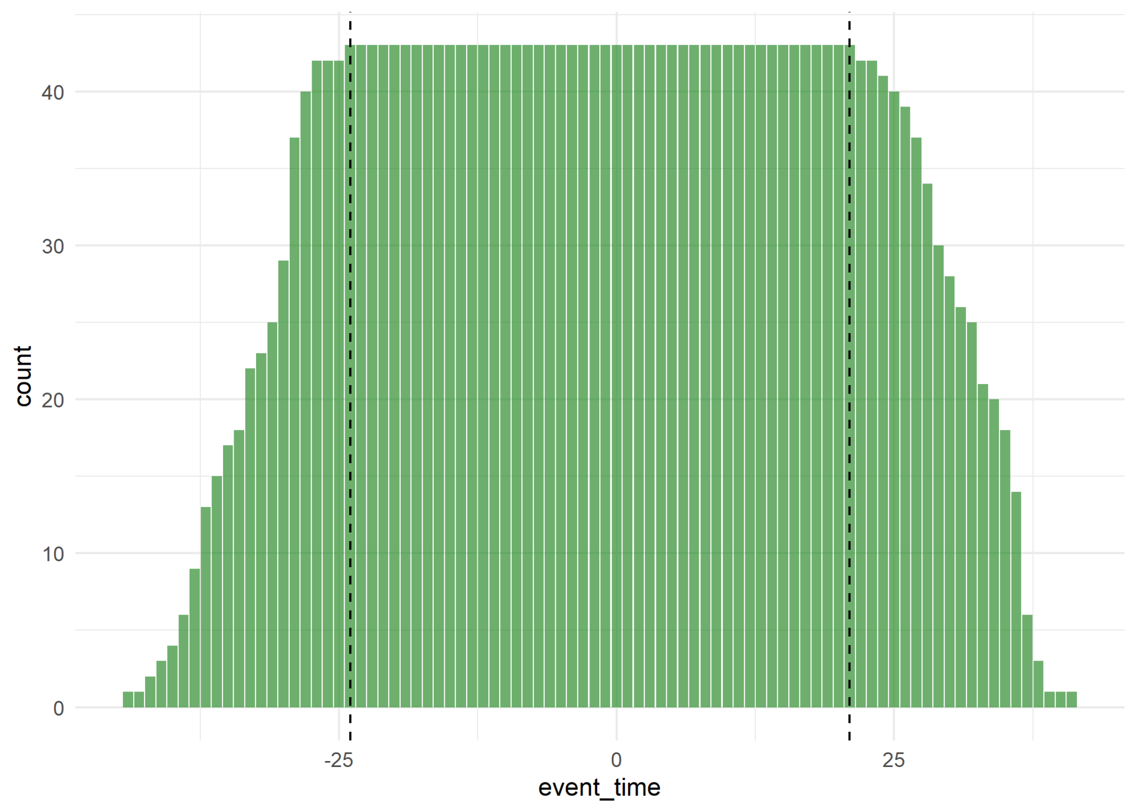
This accomplishes two goals:

1. Helps with weak identification issues
    - i.e. only CA has event times 39-41
2. Ensures dynamic treatment effects are identified separate from time trends even if we omit pure control units
    - **Schmidheiny and Siegloch (2019)**

Binning Endpoints

Q: Why -24 and 21 for the endpoints?

A: these are the max and min event times for **all treated states**



Binning Endpoints

Coding up a binned event time variable:

```
sah_es ← mutate(sah_es,
                event_time_bin = case_when(
                  event_time ⩽ -24 ~ -24, # set to -24 if ⩽ -24
                  event_time ⩾ 21 ~ 21, # set to 21 if ⩾ -21
                  TRUE ~ event_time # if in between, keep current value
                ))
```

Event Study

Perfect!

Now to estimate the event study, we need a **vector of indicator variables**, one for each `event_time`:

$$\text{-14 Days Since SAH} = 1 \text{ if event time } = 14, \; 0 \text{ otherwise}$$

$$\vdots$$

$$\text{0 Days Since SAH} = 1 \text{ if event time } = 0, \; 0 \text{ otherwise}$$

$$\vdots$$

$$\text{7 Days Since SAH} = 1 \text{ if event time } = 7, \; 0 \text{ othe}$$

We *could* code this by hand ...

Event Study and feols

… or we could use the **interaction operator** `i()` from **fixest** to automatically do it for us.

Need two things:

1. Our binned event time variable (have)
2. A treated unit dummy (need)

```
sah_es <- group_by(sah_es, state) %>%
  mutate(
    sah_state = ifelse(!is.na(mandate_date),1,0)
    ) %>%
  ungroup()
```

Event Study and feols

Now estimating the event study:

- Using `i()` to create the vector of event-time coefficients automatically
- Normalizing to the first day prior to adoption (event time -1)

```
es_reg <- feols(cadt ~ i(event_time_bin, sah_state, ref = -1) |
                  state + date, data = sah_es)
```

Event Study and feols

Looking at the output reveals all the dynamic treatment effect estimates: .font70[

```
coeftable(es_reg)
```

```
##                                Estimate Std. Error    t value          Pr(>|
## event_time_bin::-24:sah_state  5.24431750  2.2772006  2.3029669 0.0254784373
## event_time_bin::-23:sah_state  5.91884595  1.9222573  3.0791122 0.0033680660
## event_time_bin::-22:sah_state  6.03668263  1.9117363  3.1576963 0.0026955741
## event_time_bin::-21:sah_state  5.75074476  2.0646980  2.7852716 0.0075348287
## event_time_bin::-20:sah_state  5.72606565  1.8155892  3.1538333 0.0027254395
## event_time_bin::-19:sah_state  6.10467265  1.7414227  3.5055663 0.0009716771
## event_time_bin::-18:sah_state  5.97785810  1.7416719  3.4322527 0.0012101777
## event_time_bin::-17:sah_state  4.24352976  1.5060657  2.8176258 0.0069108599
## event_time_bin::-16:sah_state  4.00918806  1.3802400  2.9047035 0.0054612041
## event_time_bin::-15:sah_state  4.04706491  1.4052157  2.8800311 0.0058402927
## event_time_bin::-14:sah_state  2.73969032  1.3175311  2.0794123 0.0427277930
## event_time_bin::-13:sah_state  2.42408619  1.4152579  1.7128229 0.0929406324
## event_time_bin::-12:sah_state  2.85572278  1.5119888  1.8887196 0.0647369604
## event_time_bin::-11:sah_state  2.32238779  1.3237990  1.7543357 0.0855000900
```

Event Study Plots

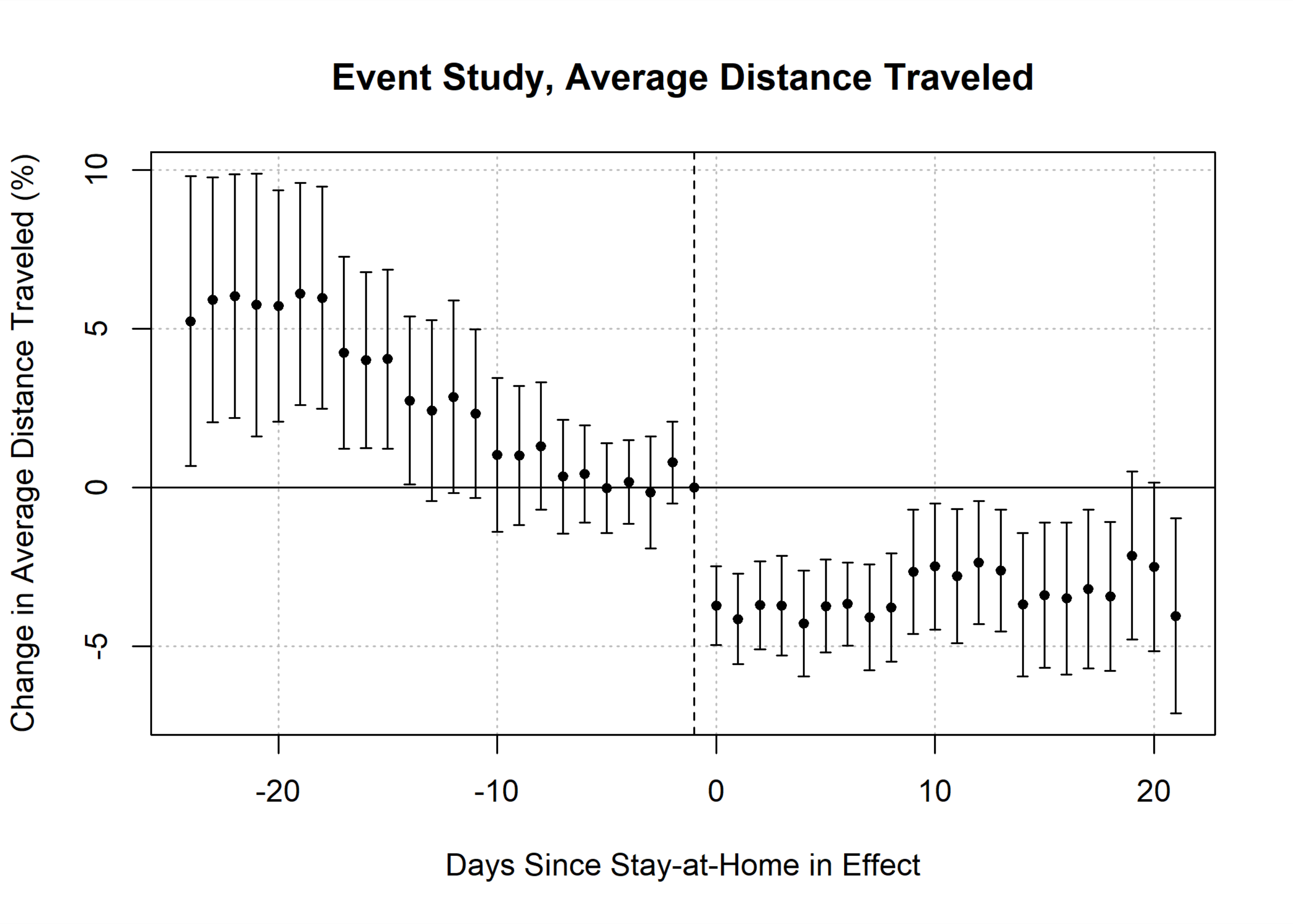
Unfortunately, this makes for a clunky regression table.

Better: plot it!

fixest makes this super convenient with the `iplot()` function:

```r
iplot(es_reg,
      ylab = "Change in Average Distance Traveled (%)",
      xlab = "Days Since Stay-at-Home in Effect",
      main = "Event Study, Average Distance Traveled"
)
```

iplot() Output



Event Study Plots

If you want the fully tidyverse functionality, we can `tidy()` up the regression output and customize the plot ourselves:

```
# get separate data frames for pre and post event times
es_pre <- tidy(es_reg) %>% filter(str_detect(term, "-"))
es_post<- tidy(es_reg) %>% filter(!str_detect(term, "-"))
# make a filler row for the reference period
es_m <- data.frame(term = "event_time_bin::-1:sah_state", estimate = 0, s

# combine all into a single figure, add event time and confidence interva
es_fig <- rbind(es_pre, es_m, es_post)  %>%
      mutate(event_time = str_extract(term, "-?[0-9]+") %>% as.numeric(),
             ci_l = estimate - qnorm(0.975)*std.error,
             ci_u = estimate + qnorm(0.975)*std.error)
```

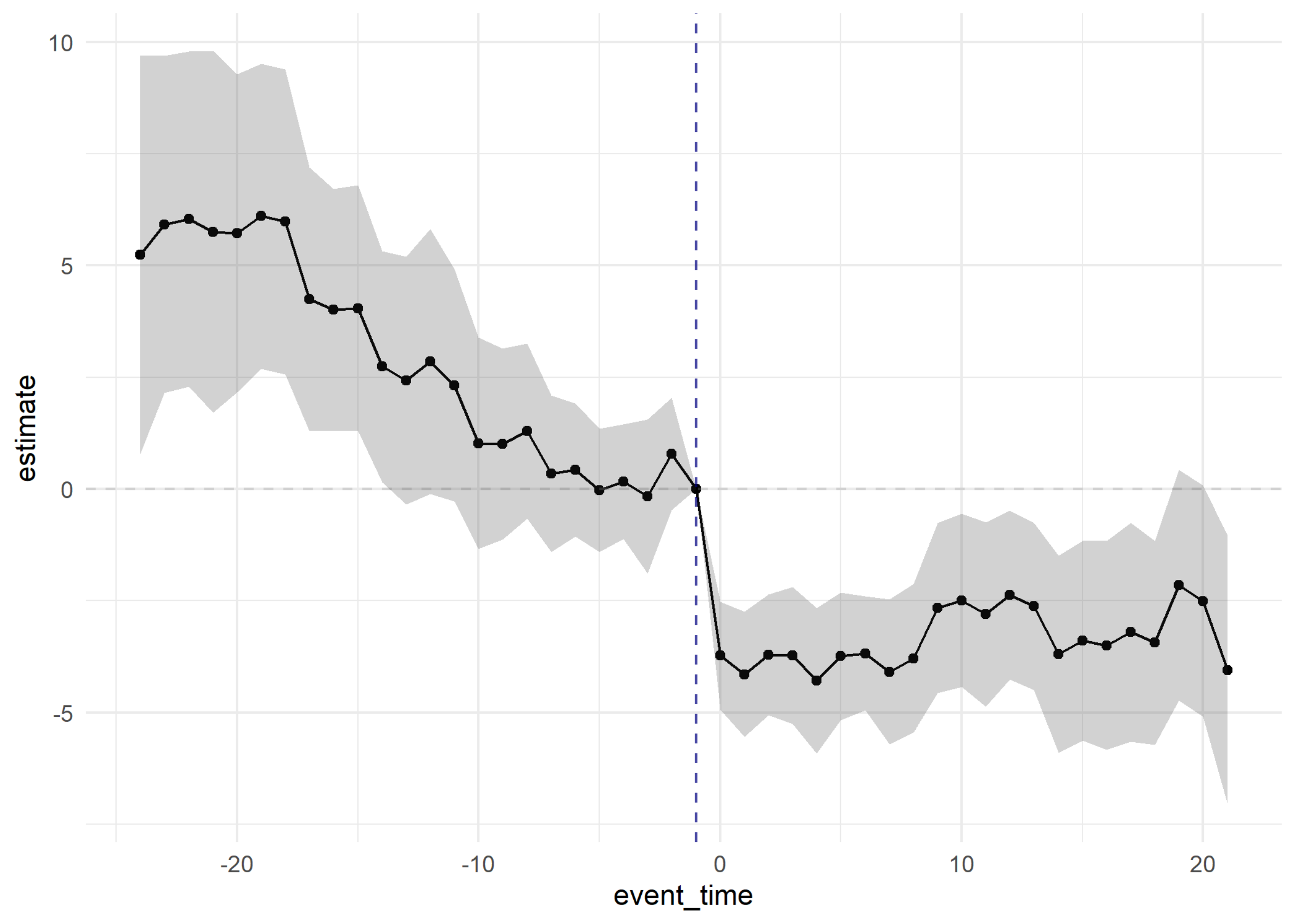
Event Study Plots

If you want the fully tidyverse functionality, we can `tidy()` up the regression output and customize the plot ourselves:

```r
ggplot(es_fig, aes(y = estimate, x = event_time)) +
  # add line and points for estimates at each event-time
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 0, colour = "grey60", linetype = 2, alpha = .3)
# add dashed line for SAH adoption date
  geom_vline(xintercept = -1, linetype = "dashed", alpha = 0.7, colour = "
# add ribbon for confidence interval
  geom_ribbon(aes(ymin = ci_l, ymax = ci_u), alpha = 0.2)+
  theme_minimal()
```

Event Study Plots

If you want the fully tidyverse functionality, we can `tidy()` up the regression output and customize the plot ourselves:



Event Study

The event study plot makes it easy to see a few things

- **Parallel Trends:** If both groups were trending the same prior to treatment period, we'd expect event-time coefficients for $k < 0$ to be **indistinguishable from zero**
- **Dynamic Treatment Effects:** slight rebound in point estimates after a week, but statistically indistinguishable from one another here

The lack of parallel trends holding will motivate additional, forefront methods that allow us to obtain a more valid counterfactual.

Table of Contents

1. **Regression Output**
2. **IV Regression**
3. **Difference-in-Differences**
4. **Event Study**