Lecture 9: Synthetic Control Methods, Part 2

James Sears*

AFRE 891 SS 24

Michigan State University

*Parts of these slides are adapted from **“Causal Inference for the Brave and True”** by Matheus Facure Alves.

Table of Contents

Prologue

Prologue

Part 1

- Matching
- Canonical Synthetic Control

Part 2

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

Prologue

Packages we'll use today:

```r
# If not installed, add in packages from GitHub not on CRAN
if (!require("augsynth")) remotes::install_github("ebenmichael/augsynth")
```

```r
if (!require("pacman")) install.packages("pacman")
pacman::p_load(augsynth, fixest, gsynth, synthdid, tidyverse)
```

As well, let's load the event study data from **Sears et al. (2023)** we finished last lecture with (plus some covariates)

```r
sah <- readRDS("data/sah_covar.rds")
```

Prologue

Last time we talked through the canonical **Synthetic Control Method (SCM)** for comparative case studies.

While it provides an avenue for overcoming violations of parallel trends, it is limited in several ways

1. Intended for a single treated aggregate unit ("comparative case studies")
2. Results primarily presented visually
3. Inference doesn't match our usual approaches

What would be *great* is if there were ways of combining the matching benefits of SCM with the identification and structure of our typical econometric approaches...

Prologue

Okay so I've already spoiled this: yes, there are methods that do this - and we're going to learn about them today.

1. Synthetic Difference-in-Differences: using a synthetic control unit as the counterfactual in a difference-in-differences
2. Partially Pooled Synthetic Control: same idea but for event study settings with dynamic treatment effects

We refer to these two approaches as doubly-robust estimators, in that you have two chances to be right:

1. In the matching design, or
2. In the regression specification

Synthetic Difference-in-Differences

Synthetic Difference-in-Differences

Synthetic Difference-in-Differences by Arkhangelsky et al. (2021) AER offers a solution to CIA violations in a typical difference-in-differences setting.

The gist:

1. Re-weight and match pre-exposure trends by constructing a synthetic counterfactual
    - Reduces reliance on parallel trends
2. Estimate a treatment effect in a standard TWFE regression

Synthetic Difference-in-Differences

More formally:

Suppose you have **balanced panel** with

- T time periods
- N units
    - First N_{co} units are never exposed to treatment (pure controls)
    - Next $N_{tr} = N - N_{co}$ units receive treatment after time T_{pre}
        - Can be block adoption or staggered adoption

You observe

- The time-varying outcome Y_{it},
- A binary treatment indicator W_{it},
- and (optionally) time-varying covariates X_{it}

i.e. same data requirements as a standard TWFE Diff-in-Diff approach

Synthetic Difference-in-Differences

The Concern: even after conditioning on observables, CIA/parallel trends doesn't hold.

The Solution: construct a synthetic control to use as the counterfactual in a Difference-in-Differences design.

Like with SCM, the goal is to find weights $\hat{\omega}^{sdid}$ so that the pre-treatment trends in the treated units' outcome align with the synthetic control's:

$$\underbrace{\sum_{i=1}^{N_{co}} \hat{\omega}^{sdid} Y_{it}}_{\mathit{Synthetic\ Control}} \approx \underbrace{\frac{1}{N_{tr}} \sum_{i=N_{co}+1}^{N} Y_{it}}_{\mathit{Treated}}$$

Synthetic Difference-in-Differences

Choosing Unit Weights: ensure average outcome for treated unit(s) is roughly parallel to the synthetic counterfactual (i.e. weighted average of control units)

Choose **unit weights** $(\hat{\omega}_0, \hat{\omega})$ that yield the arg min of

$$\underbrace{\sum_{t=1}^{T_{pre}} \left(\omega_0 + \sum_{i=1}^{N_{co}} \omega_i Y_{it} - \frac{1}{N_{tr}} \sum_{i=N_{co}+1}^{N} Y_{it} \right)^2}_{Sum\ Squared\ Error} + \underbrace{\zeta^2 T_{pre} ||\omega||_2^2}_{L_2\ Regularization\ Penalty}$$

Synthetic Difference-in-Differences

$$\underbrace{\sum_{t=1}^{T_{pre}} \left(\omega_0 + \sum_{i=1}^{N_{co}} \omega_i Y_{it} - \frac{1}{N_{tr}} \sum_{i=N_{co}+1}^{N} Y_{it} \right)^2}_{\textit{Sum Squared Error}} + \underbrace{\zeta^2 T_{pre} ||\omega||_2^2}_{L_2 \textit{ Regularization Penalty}}$$

- Intercept term ω_0 allows synthetic control to match on pre-trends
  - Unit FE α_i in regression step get rid of level differences
- Penalty term helps non-zero weights be more distributed across control units
  - If $\zeta = 0$, weights are identical to Abadie, Diamond, and Hainmueller (2010) SCM weights for a single treated unit
  - ζ is *complicated*, see the paper if you want more details
  - L_2/Euclidean norm as in ridge regression (we'll chat more about this in ML lecture)

Synthetic Difference-in-Differences

Choosing Time Weights: ensure pre and post-treatment periods are balanced for control units

Choose **time weights** $(\hat{\lambda}_0, \hat{\lambda})$ that yield arg min of

$$\sum_{i=1}^{N_{co}} \left(\underbrace{\lambda_0 + \sum_{t=1}^{T_{pre}} \hat{\lambda}_t Y_{it}}_{\substack{Weighted\ Average \\ in\ Pre-Period}} - \underbrace{\frac{1}{T_{post}} \sum_{t=T_{pre}+1}^{T} Y_{it}}_{\substack{Average\ in \\ Post-Period}} \right)^2$$

- No regularization in time weights
  - Allows for correlation over time for same unit, but not across units (beyond the systematic component in a latent factor model)

Synthetic Difference-in-Differences

Once unit and time weights (and ζ) are obtained, estimate $\hat{\tau}^{sdid}$ as arg min of

$$\sum_{i=1}^{N}\sum_{t=1}^{T}(Y_{it} - \mu - \alpha_i - \beta_t - W_{it}\tau)^2 \hat{\omega}_i^{sdid} \hat{\lambda}_t^{sdid}$$

Note that without weights, this is just the TWFE Diff-in-Diff solution:

$$\sum_{i=1}^{N}\sum_{t=1}^{T}(Y_{it} - \mu - \alpha_i - \beta_t - W_{it}\tau)^2$$

Synthetic Diff-in-Diff Application

Now that we know how it's working, let's try it out by looking at the impact of California's Proposition 99, which introduced a 25 cent tax per pack of cigarettes.

```
data(california_prop99)
```

Synthetic Diff-in-Diff Application

We *could* evaluate the impact of Prop 99 on cigarette sales per capita using a typical TWFE Diff-in-Diff:

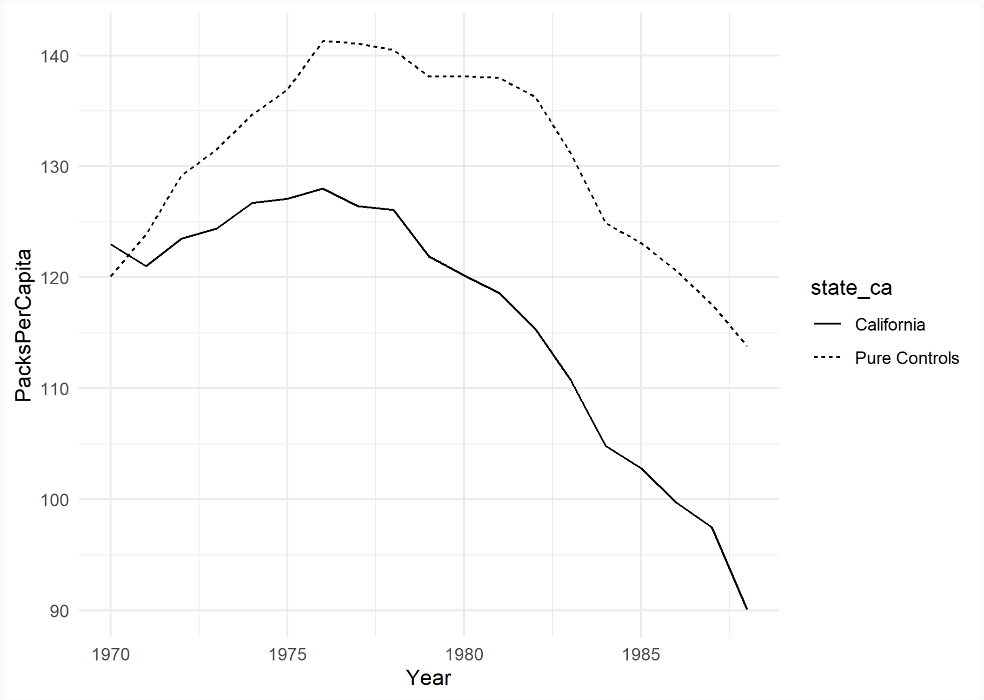
```
did ← feols(PacksPerCapita ~ treated | State + Year, data = california_p
summary(did)
```

```
## OLS estimation, Dep. Var.: PacksPerCapita
## Observations: 1,209
## Fixed-effects: State: 39,  Year: 31
## Standard-errors: Clustered (State)
##          Estimate Std. Error  t value   Pr(>|t|)
## treated -27.3491    2.80238 -9.75925 6.6913e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 11.5     Adj. R2: 0.870229
##              Within R2: 0.032671
```

Synthetic Diff-in-Diff Application

Looking at parallel trends:

```
mutate(california_prop99, state_ca = ifelse(State == "California", "California", "Pure Controls")) %>%
group_by(state_ca, Year) %>%
  summarise(PacksPerCapita = mean(PacksPerCapita)) %>%
  filter(Year < 1989) %>%
  ggplot() +
  geom_line(aes(x = Year, y = PacksPerCapita, linetype = state_ca)) +
  theme_minimal()
```



Synthetic Diff-in-Diff Application

Let's use the **synthdid** package to estimate a diff-in-diff using a synthetic California as the counterfactual.

First, we'll need to do some se the `panel.matrices()` function to set up the data

- Balanced panel (have)
- Simultaneous adoption (have mechanically)

```r
synth_ca_prep ← panel.matrices(
  panel = as.data.frame(california_prop99), # the dataframe
  unit = "State", # unit column (name or column #)
  time = 2, # time column (name or column #)
  outcome = 3, # outcome var (name or column #)
  treatment = "treated", # treatment var (name or column #)
  treated.last = TRUE # sort treated units to be at bottom
)
```

Synthetic Diff-in-Diff Application

Now compute the synthetic Diff-in-Diff estimate with `synthdid_estimate()`:

```
sdid ← synthdid_estimate(Y = synth_ca_prep$Y, # outcome var
                          N0 = synth_ca_prep$N0, # number of control unit
                          T0 = synth_ca_prep$T0 # number of pre-treatment
                          )
sdid
```

```
## synthdid: -15.604 +- NA. Effective N0/N0 = 16.4/38~0.4. Effective T0/T0 = 2.
```

Which yields a much smaller treatment effect estimate than the Diff-in-Diff.

Synthetic Diff-in-Diff Application

Looking at control unit weights ω:

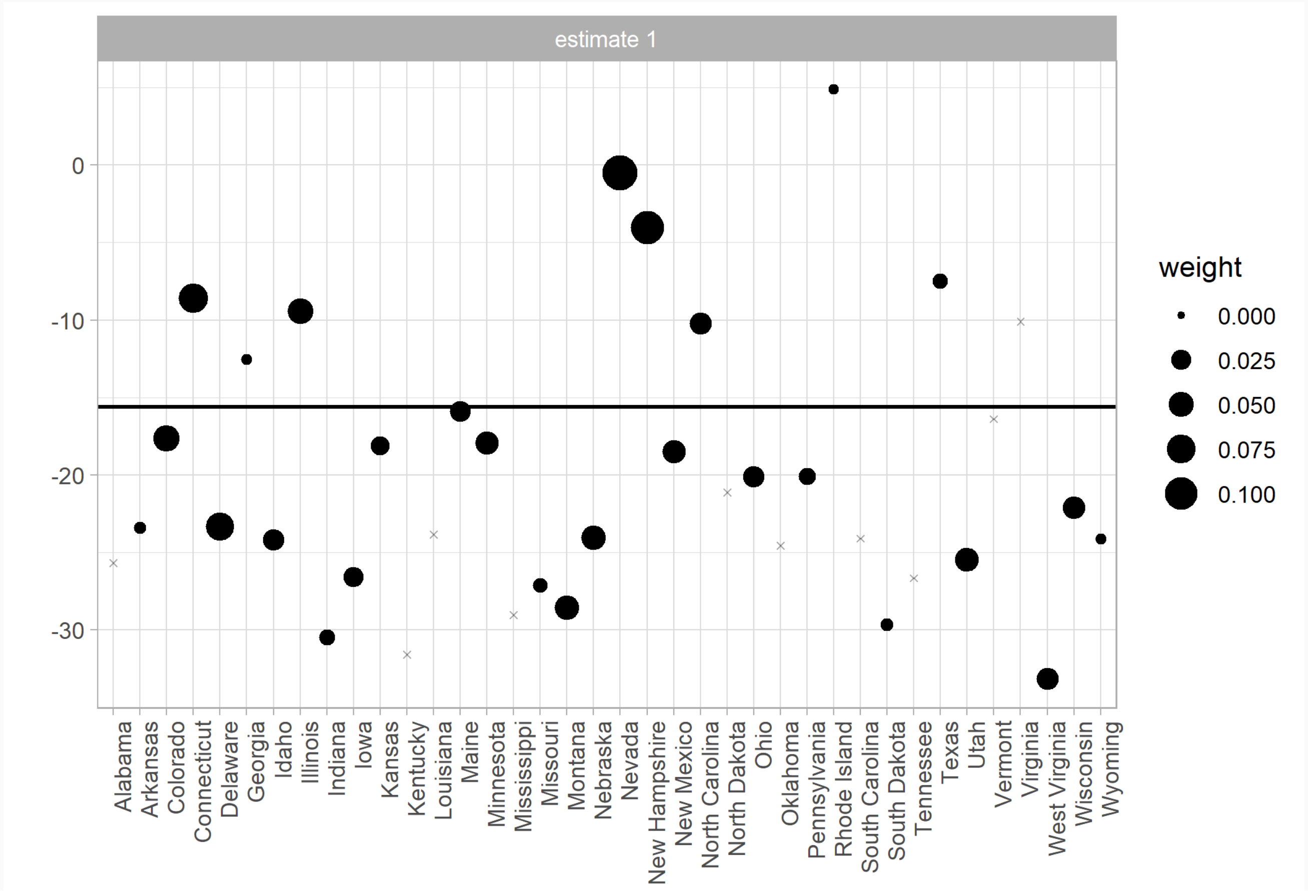
```
synthdid_controls(sdid)
```

```
##                estimate 1
## Nevada         0.12448923
## New Hampshire  0.10504758
## Connecticut    0.07828729
## Delaware       0.07036812
## Colorado       0.05751279
## Illinois       0.05338782
## Nebraska       0.04785319
## Montana        0.04513521
## Utah           0.04151766
## New Mexico     0.04056827
## Minnesota      0.03949464
## Wisconsin      0.03666708
## West Virginia  0.03356911
## North Carolina 0.03280518
## Idaho          0.03146821
```

Synthetic Diff-in-Diff Application

Plotting the control unit weights ω:

```
synthdid_units_plot(sdid)
```



Synthetic Diff-in-Diff Application

And the time weights λ

```
summary(sdid)$periods
```

```
##      estimate 1
## 1988      0.427
## 1986      0.366
## 1987      0.206
```

Synthetic Diff-in-Diff Application

Note that with only one treated unit we can only use the (untrustworthy) `placebo` method to get standard errors, by calling `vcov()` on our `synthdid_estimate` object:

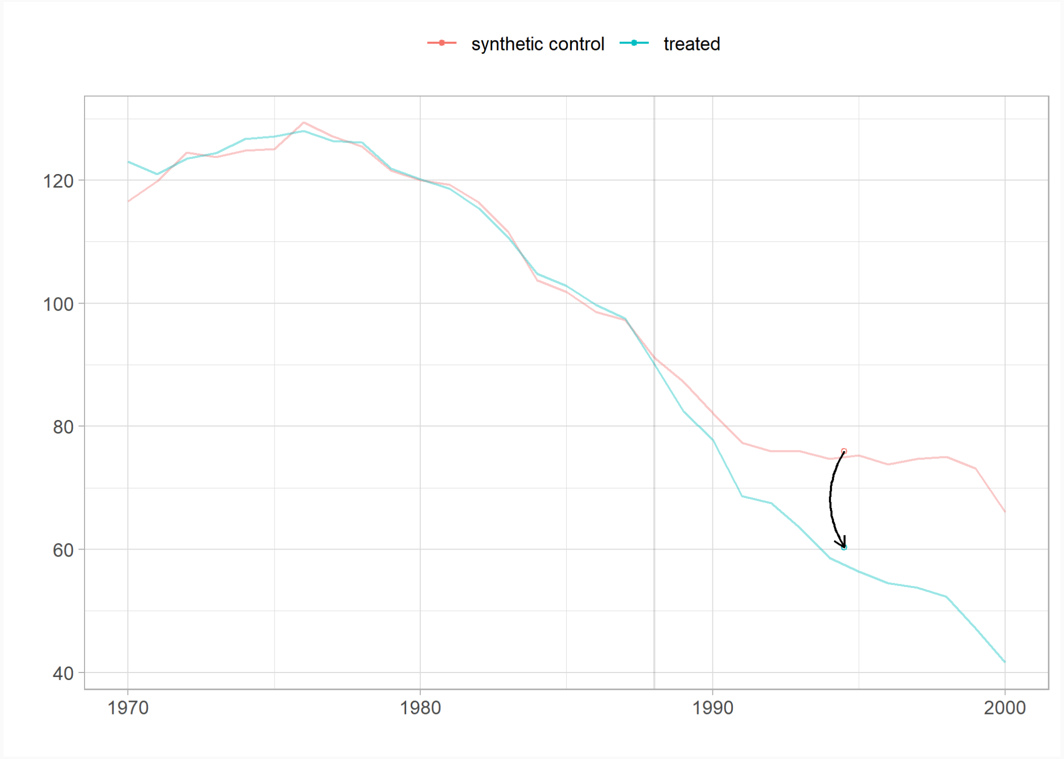
```r
se <- sqrt(vcov(sdid, method='placebo'))
sprintf('95%% CI (%1.2f, %1.2f)', sdid - 1.96 * se, sdid + 1.96 * se)
```

```
## [1] "95% CI (-34.00, 2.80)"
```

Synthetic Diff-in-Diff Application

We can look at pre-treatment parallel trends by overlaying the two series:

```
synthdid_plot(sdid, overlay = 1)
```



Synthetic Diff-in-Diff Application

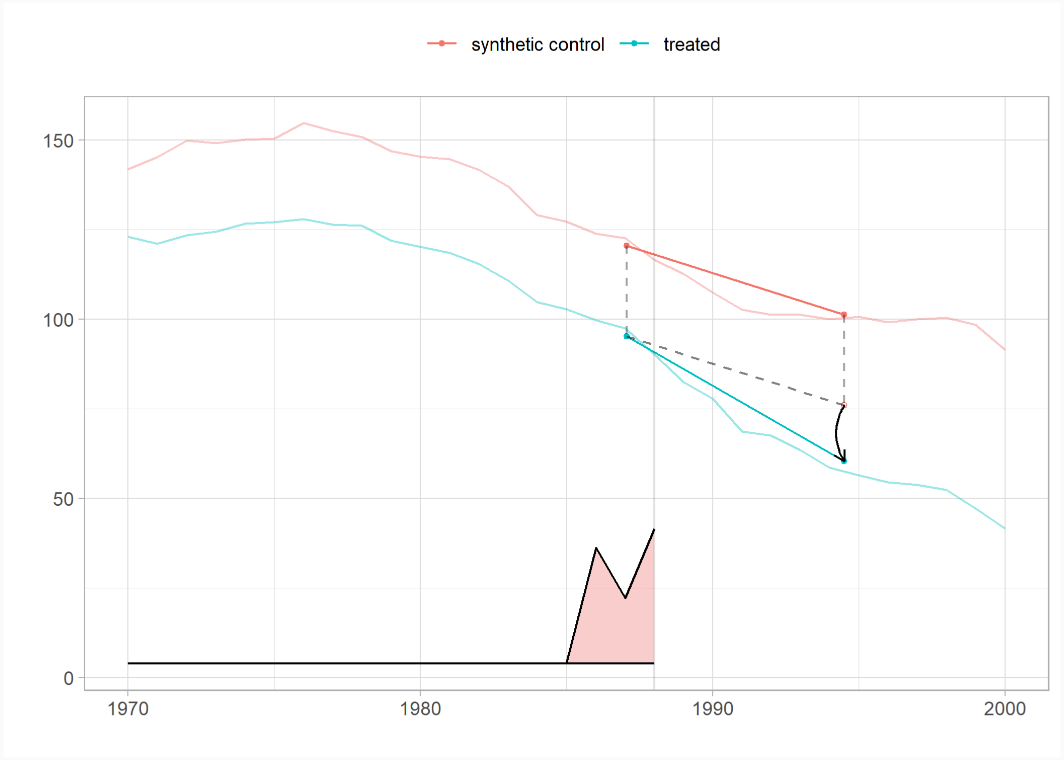
The built-in `synthdid` plot (a `ggplot` object) displays a lot of information by default.

- Point and line segments for simple 2x2 Diff-in-Diff comparison
- Time period weights (bottom red line)
- Customizable further with `theme()` and **other adjustments**

Synthetic Diff-in-Diff Application

The built-in `synthdid` plot (a `ggplot` object) displays a lot of information by default.

```
synthdid_plot(sdid)
```



Synthetic Diff-in-Diff Application

Comparing to canonical SCM and Diff-in-Diff reveals the differences well:

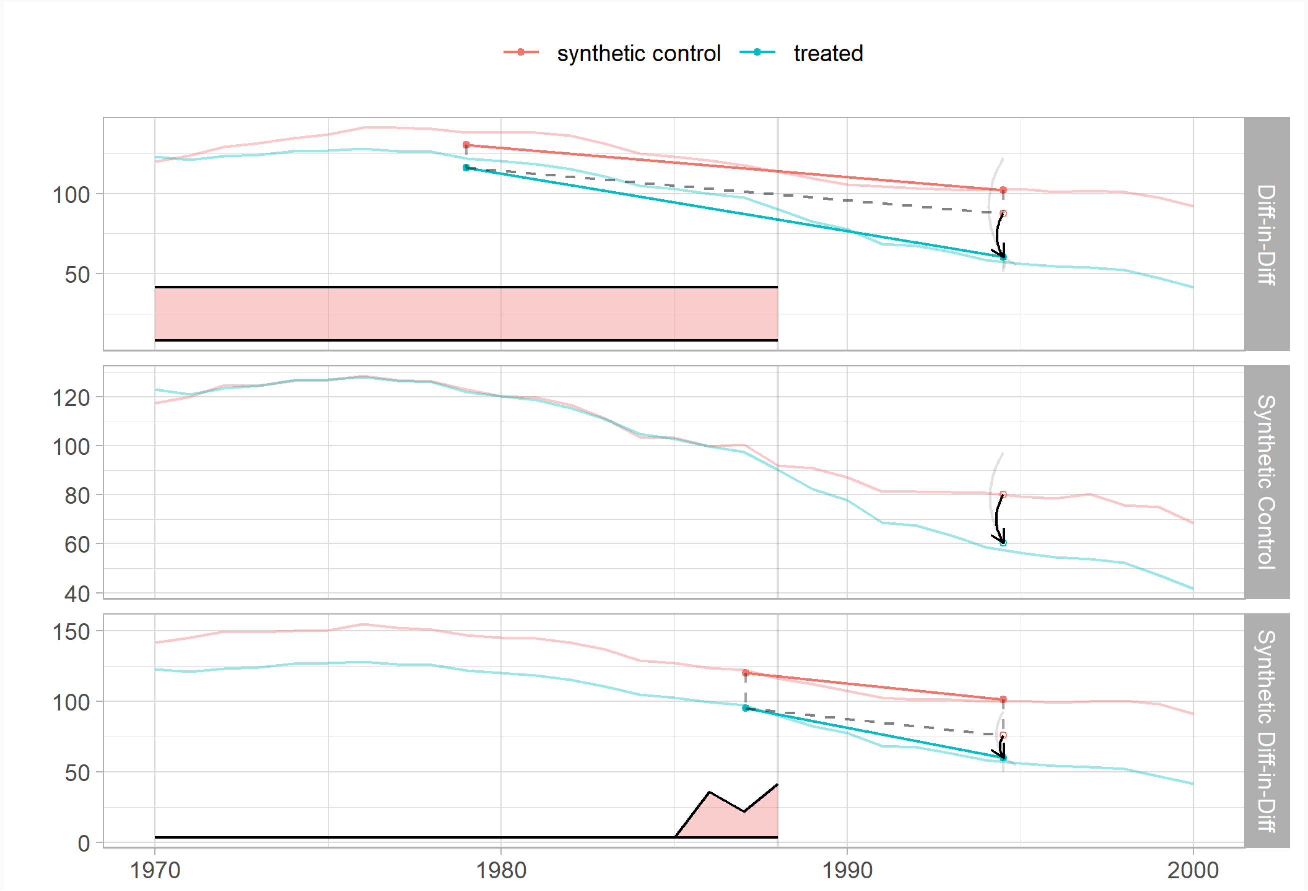
```
est_sc ← sc_estimate(Y = synth_ca_prep$Y, N0 = synth_ca_prep$N0, T0 = sy
est_did ← did_estimate(Y = synth_ca_prep$Y, N0 = synth_ca_prep$N0, T0 = s
estimates ← list(est_did, est_sc, sdid)
names(estimates) = c('Diff-in-Diff', 'Synthetic Control', 'Synthetic Diff-
print(unlist(estimates))
```

```
##            Diff-in-Diff       Synthetic Control Synthetic Diff-in-Diff
##               -27.34911               -19.61966              -15.60383
```

Synthetic Diff-in-Diff Application

Comparing to canonical SCM and Diff-in-Diff reveals the differences well:

```
synthdid_plot(estimates, se.method='placebo')
```



Staggered Adoption

The methods in **synthdid** easily extend to

- Controlling for covariates in the second step regression
- Cases of more than one unit adopting simultaneously.

One current limitation is lack of direct support for **staggered adoption**

Until they add native support, you can use the **Ssynthdid** package

```r
# install.remotes("remotes")
remotes::install_github("tjhon/ssynthdid")
```

Partially Pooled Synthetic Control

Partially Pooled Synthetic Control

An alternate but related estimator is the **partially pooled synthetic control method (PPSCM)** of Ben-Michael, Feller, and Rothstein (2021)

The gist:

- Extend canonical SCM to the many-treated unit and staggered adoption case
- Incorporate unit-level intercepts and balancing on covariates
    - Equivalent to balancing on residualized unit-level outcomes
- Estimate dynamic treatment effects
- Obtain standard errors through bootstrapping/jackknifing

Partially Pooled Synthetic Control

More formally:

Suppose you have a panel with

- T time periods
- N units
  - Some units $j = 1..., J$ receive treatment, potentially at different times T_i
  - Non-zero number of pure controls N_0 with $T_i = \infty$
    - Can be block adoption or staggered adoption

You observe

- The time-varying outcome Y_{it},
- A binary treatment indicator W_{it},
- and (optionally) time-varying covariates X_{it}

Partially Pooled Synthetic Control

Assumptions:

- Stable treatment and no interference across units (SUTVA)
- Prior to treatment, a unit's potential outcomes are equal to its never-treated potential outcomes

- **No Anticipation:**

$$Y_{it}(s) = Y_{it}(\infty) \text{ for } t < s, \text{ with treatment time } s$$

Partially Pooled Synthetic Control

Assumptions:

- All treated units are observed for at least several pre-periods and several post-periods
  - Needed to ensure sufficient identification in unbalanced event time
- Can express the data generating process as following
  1. Following a time-varying $AR(L)$ process:

     $Y_{it}(\infty) = \sum_{\ell=1}^{L} \rho_{t\ell} Y_{it-\ell}(\infty) + \epsilon_{it}$, or
     - Rules out correlation between treatment timing and noise terms for any period
  2. Composed of time-varying latent factors and time-invariant unit loadings: $Y_{it}(\infty) = \phi_i \cdot \mu_t + \epsilon_{it}$
     - Rules out correlation between treatment timing and noise terms *after* treatment
  3. Noise term ϵ_{it} are sub-Gaussian random variables

Partially Pooled Synthetic Control

If we want to extend canonical SCM to the **many-treated** case, we could take one of two approaches.

1. Separate SCM: estimate a separate SCM for each treated unit

- i.e. minimize pre-treatment imbalance for each treated unit separately
- Can lead to poor mean fit, biasing the ATE

2. Pooled SCM: estimate one SCM for average of treated units

- i.e. minimize the average pre-treatment imbalance across all treated units
- Can achieve strong average fit but obtain poor unit-specific treatment effects

The Solution: "pool" the two estimators, improving on each approach in isolation

Partially Pooled Synthetic Control

Choose SCM weights to minimize the weighted average of the pooled and unit-specific pre-treatment balance:

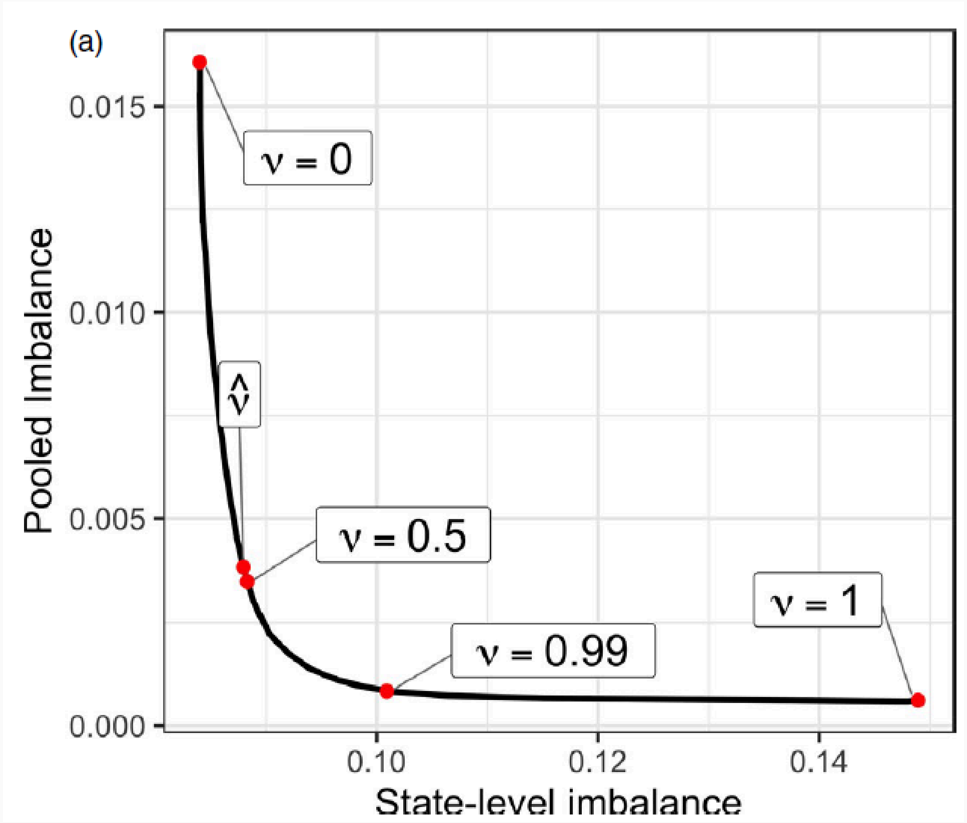
$$\min_{\Gamma \in \Delta^{scm}} \nu \underbrace{(\tilde{q}^{pool}(\Gamma))^2}_{\text{Normalized pooled imbalance}} + (1-\nu)\underbrace{(\tilde{q}^{sep}(\Gamma))^2}_{\text{Normalized separate imbalance}} + \underbrace{\lambda||\Gamma||_F^2}_{\text{penalize sum of squared weights}}$$

- $\hat{q}^{sep}, \hat{q}^{pool}$ the (normalized) root mean square of separate and pooled pre-treatment fit
- $\lambda||\Gamma||_F^2$ a penalty term (as in SCM)
- ν the hyperparameter determining the degree of "partial pooling"
    - $\nu = 0 \Rightarrow \text{Separate SCM}$
    - $\nu = 1 \Rightarrow \text{Pooled SCM}$
    - $0 < \nu < 1 \Rightarrow \text{Partially-Pooled SCM}$

Partially Pooled Synthetic Control

So, what ν should we use?

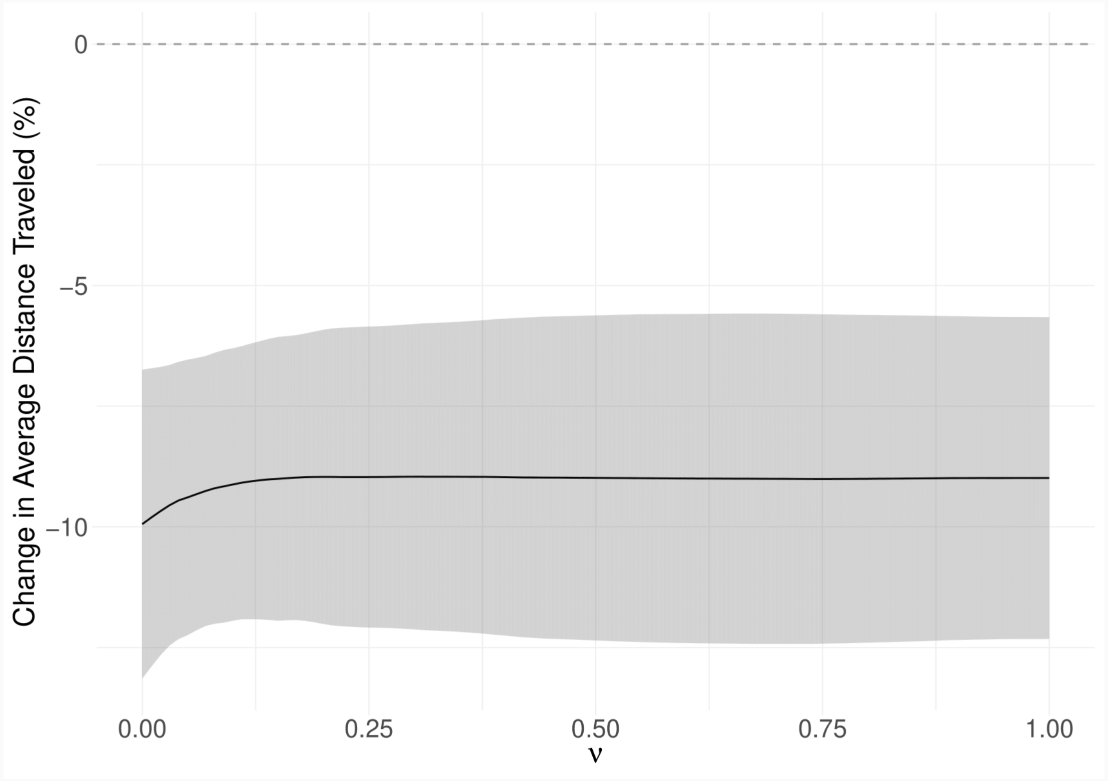
Turns out the relationship between pooled and separate imbalance is **highly convex**, with even slight interior ν offering large improvements.



Partially Pooled Synthetic Control

So, what ν should we use?

In my experience, overall ATEs are highly similar across the space of ν



PPSCM Application

So how does it actually work? Let's use the **augsynth** package and use PPSCM to revisit mobility responses to stay-at-home mandates.

Recall the setup:

- States adopted stay-at-home mandates on different days (staggered adoption)
- Observe many pre-period dates (Feb 24 to Mar 19)
- Observe many post-adoption dates (# varies by state, from 22 to 42)

PPSCM Application

One advantage to PPSCM is less data prep - we can jump directly to estimation with `augsynth()`

```
augsynth(y ~ treat | weighting covars | approx match covars | exact
        match covars, unit, time, data, n_leads, n_lags)
```

- `formula` requires just Y_{it} and W_{it}
  - Optional weighting covariates, approximate/exact matching covariates
  - `unit/time` the names of unit and time variables
  - can be text, numeric, or dates
  - Don't need manual event time
  - `data` the dataframe
  - `n_leads/n_lags` the number of lead/lag event times to estimate
  - `n_lead` default: # post-treatment dates for last-treated unit (same as our binned event time before)
  - `n_lags` default: balance all periods

PPSCM Application

Estimating with default settings and storing the summary object

- Most everything is hanging out in the `summary()` object

```
sah_ppscm ← multisynth(cadt ~ post_treat, state, date,  data = sah, fixe
sum_ppscm ← summary(sah_ppscm)
```

PPSCM Application

Looking at the output shows

- Average ATT and standard error (across all time periods + treated units)
- Imbalances and improvement over pooled/separate SCM

```
sum_ppscm
```

```
##
## Call:
## multisynth(form = cadt ~ post_treat, unit = state, time = date,
##     data = sah, fixedeff = TRUE)
##
## Average ATT Estimate (Std. Error): -6.585  (1.377)
##
## Global L2 Imbalance: 1.136
## Scaled Global L2 Imbalance: 0.347
## Percent improvement from uniform global weights: 65.3
##
## Individual L2 Imbalance: 5.543
## Scaled Individual L2 Imbalance: 0.701
## Percent improvement from uniform individual weights: 29.9
##
```

PPSCM Application

The coefficient table contains two main types of data

1. "Average" period-specific ATT across all treated units (top rows, "Average" Level)
2. Unit-specific ATTs for each period

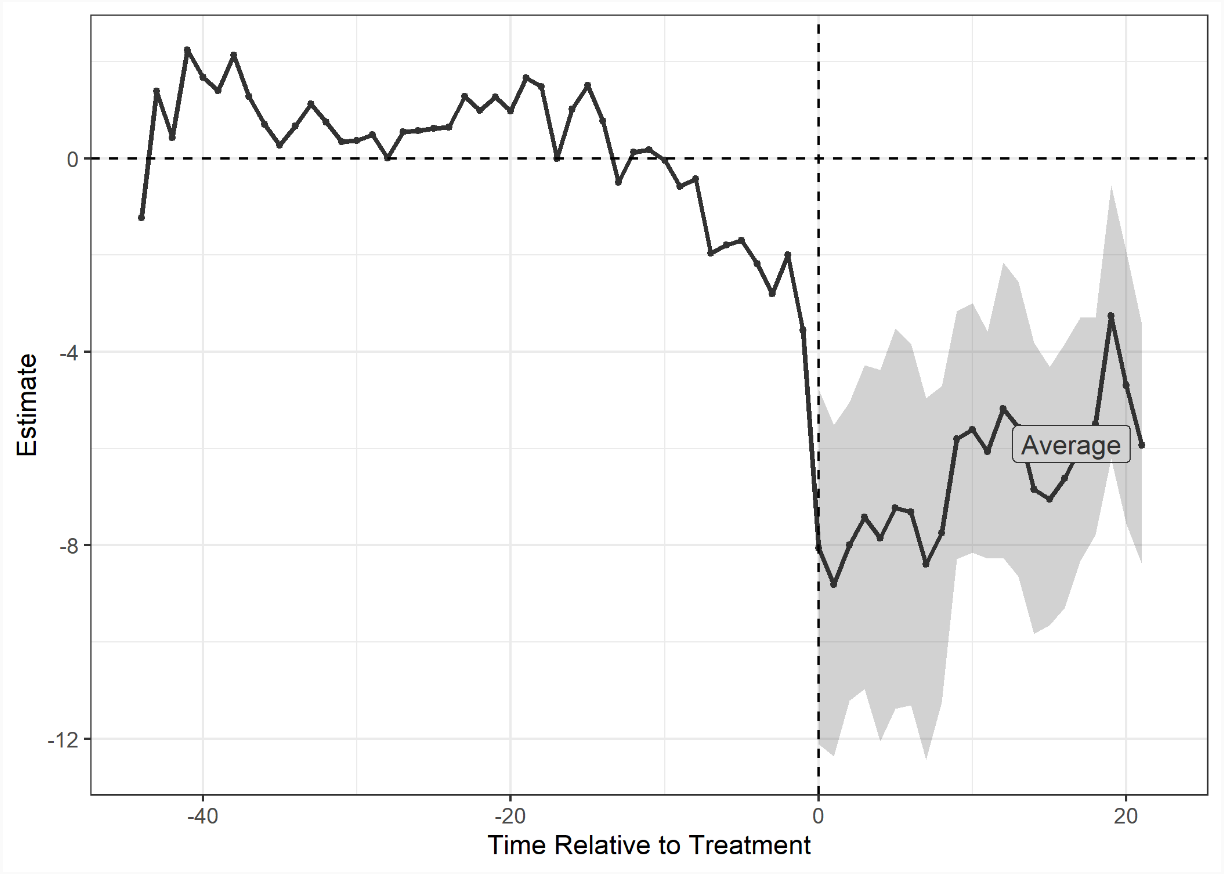
```
att_ppscm <- sum_ppscm$att
att_ppscm
```

```
##      Time   Level     Estimate   Std.Error  lower_bound upper_bound
## 1     -44 Average  -1.231575053  7.62062180 -12.99918639 11.77831295
## 2     -43 Average   1.389609406  5.72877966  -8.60717118 10.05516486
## 3     -42 Average   0.429718524  7.10399766 -12.88063008 13.30605575
## 4     -41 Average   2.236318929  5.15455150  -7.57826530  9.53654382
## 5     -40 Average   1.676580849  5.40498187  -8.97773381  8.27550067
## 6     -39 Average   1.405283286  4.29299237  -7.03945287  6.70049903
## 7     -38 Average   2.138179308  3.16688132  -4.46094486  6.54919831
## 8     -37 Average   1.286090944  2.79572246  -4.56265158  5.91151395
## 9     -36 Average   0.707601619  2.17451197  -3.75461024  4.54226054
```

PPSCM Application

Plotting the Average ATT event study:

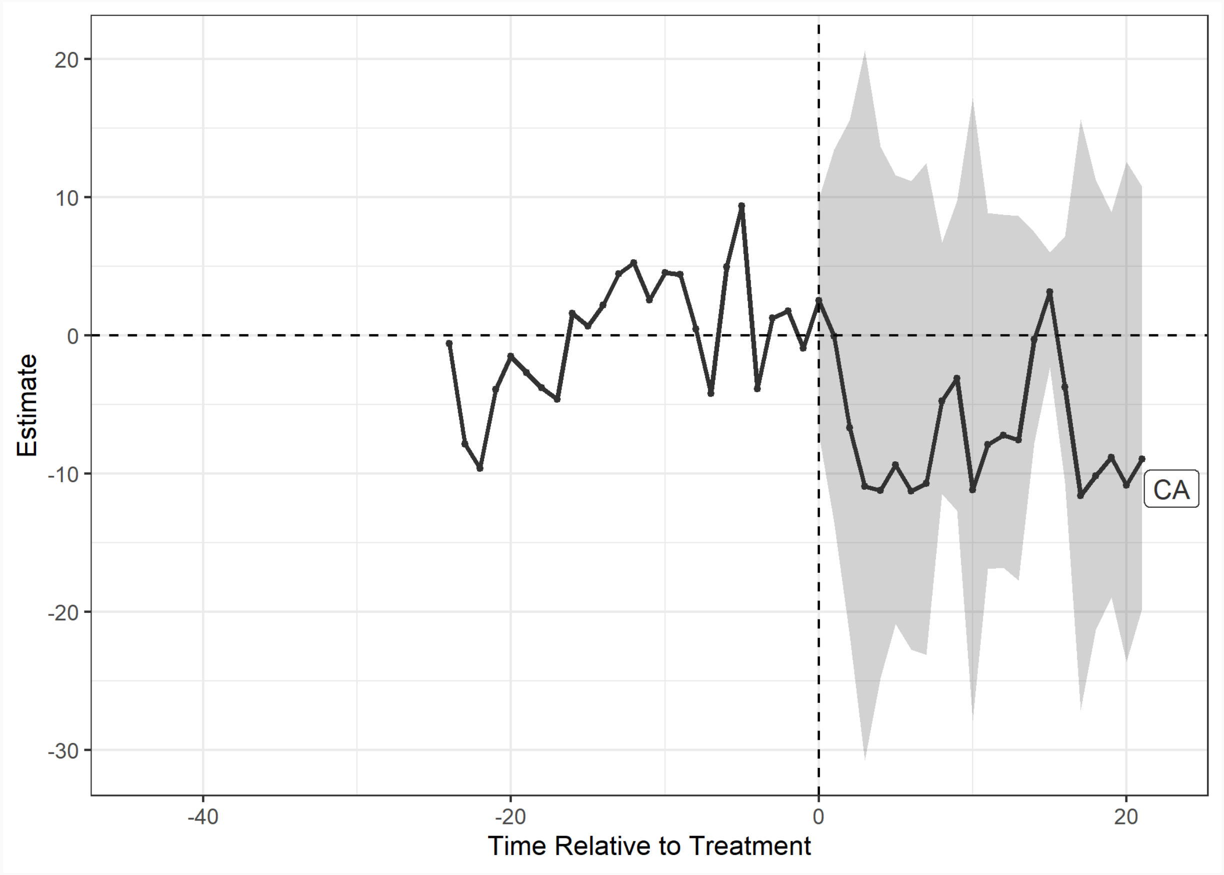
```r
plot(sum_ppscm, levels = "Average")
```



PPSCM Application

Or the event study for just CA relative to its synthetic control:

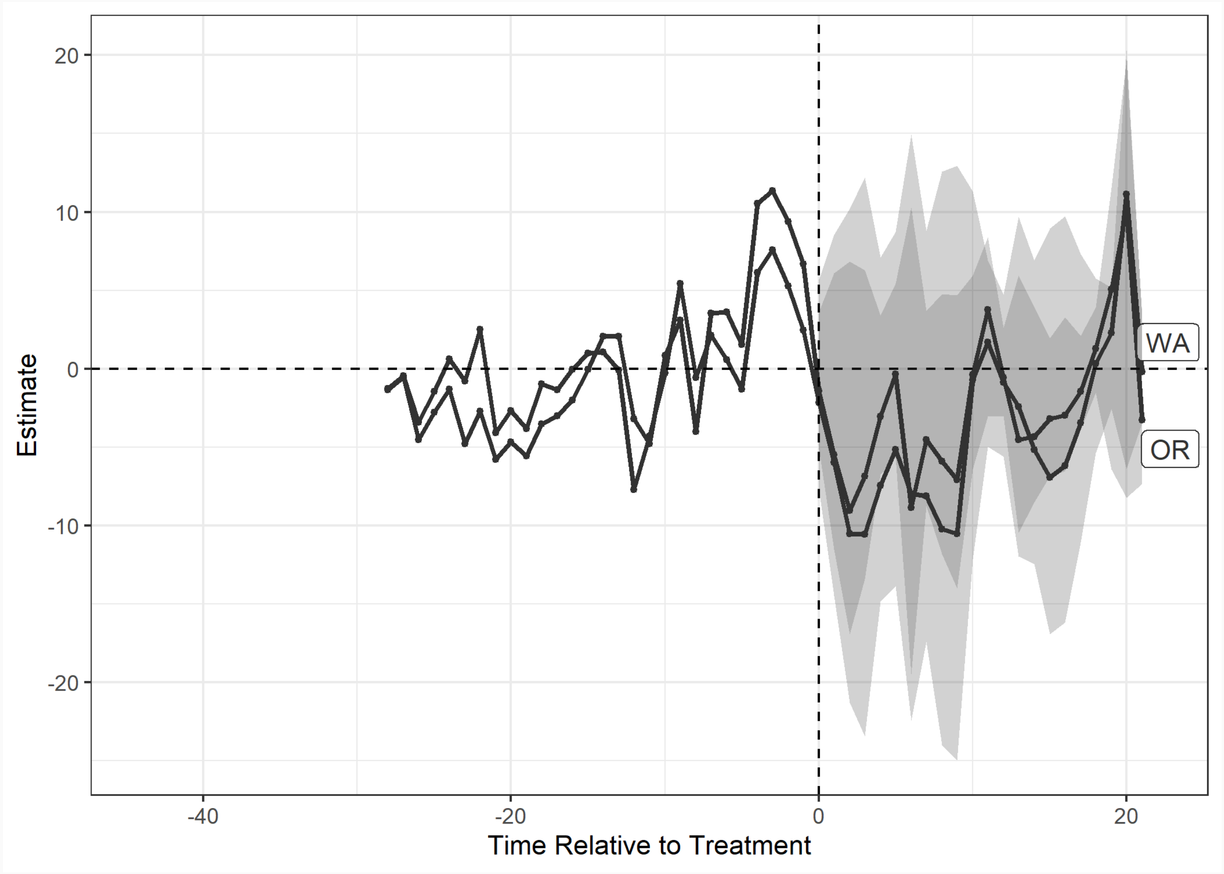
```
plot(sum_ppscm, levels = "CA")
```



PPSCM Application

Or a subset (say, OR and WA):

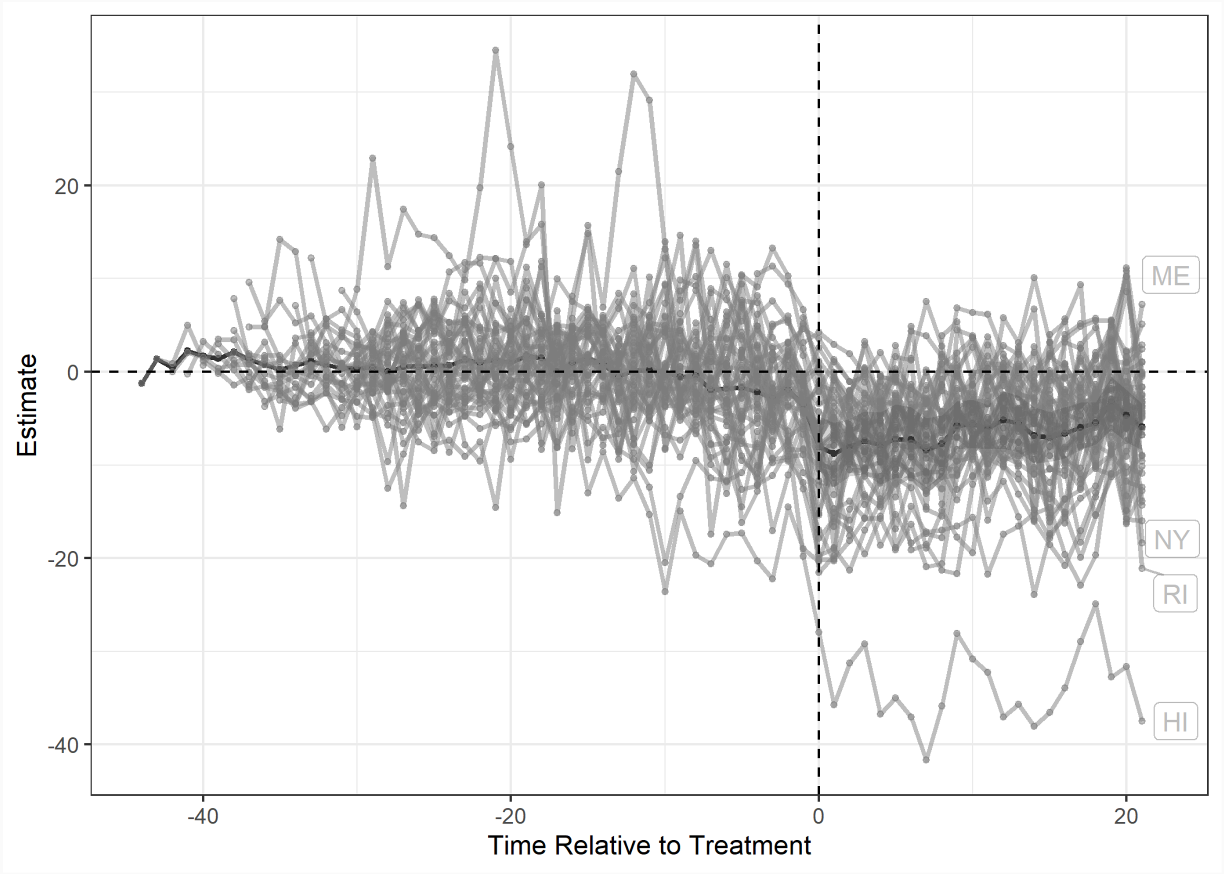
```
plot(sum_ppscm, levels = c("OR", "WA"))
```



PPSCM Application

Or every ATT all at once:

```
plot(sum_ppscm)
```



PPSCM Application

Looking at the weights:

- $N \times N_{tr}$ matrix (51 states + DC by 43 SAH adopters)

```
sah_ppscm$weights
```

```
##                [,1]          [,2]          [,3]          [,4]          [,5]
## AK  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## AL  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## AR -1.584198e-07 -1.957248e-07 -1.784517e-07  5.358271e-02 -1.606645e-07
## AZ  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## CA  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## CO  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## CT  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## DC  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## DE  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## FL  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## GA  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## HI  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## IA -3.209071e-08 -1.206382e-08  4.242161e-02  2.790426e-08 -7.483020e-08
```

PPSCM Application

What is CA (first adopter)'s synthetic control?

```
ca_wts ← sah_ppscm$weights %>%
  as.data.frame() %>%
  select(V1)

ca_wts ← mutate(ca_wts, state = rownames(ca_wts)) %>%
  rename(weight = V1) %>%
  filter(weight ≠ 0)
ca_wts
```

```
##           weight state
## AR -1.584198e-07    AR
## IA -3.209071e-08    IA
## ND -3.647051e-08    ND
## NE -5.096455e-08    NE
## OK -9.131559e-08    OK
## SD -9.114748e-08    SD
## UT  1.000001e+00    UT
## WY -1.585420e-07    WY
```

PPSCM Application

Looking at ν, we see we're about halfway between separate and pooled SCM:

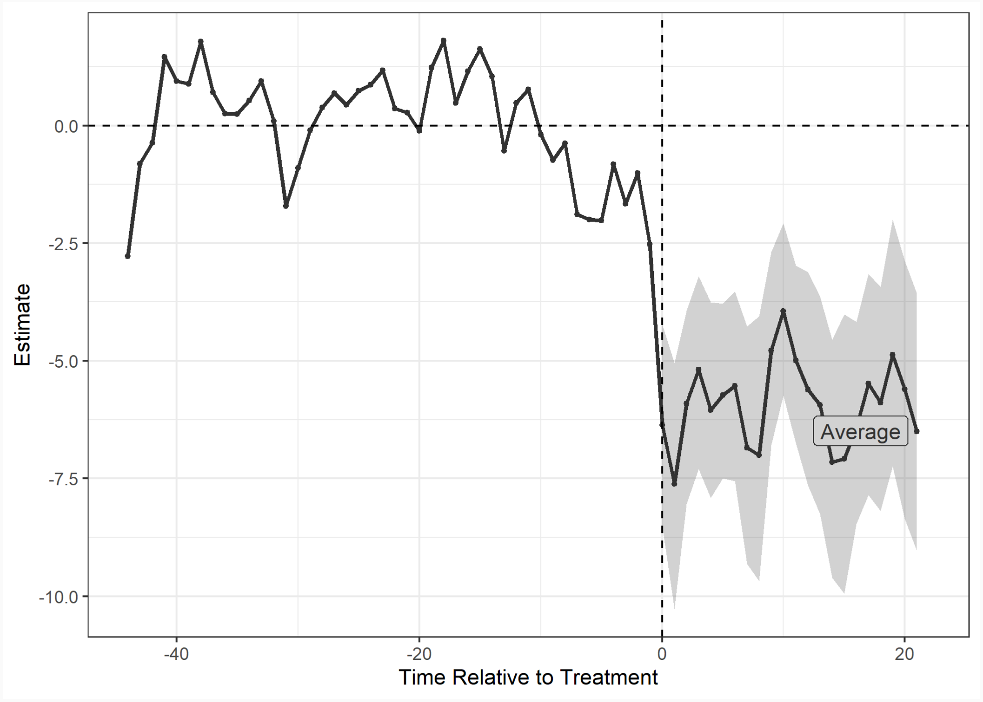
```
sah_ppscm$nu
```

```
## [1] 0.4902127
```

PPSCM Application

By default, the regularization penalty and number of factors for interactive fixed effect is zero. What happens if we turn them on?

```r
sah_ppscm2 <- multisynth(cadt ~ post_treat, state, date,  data = sah, fixedeff = TRUE,
                         lambda = 0.5,
                         n_factors = 2)
sum_ppscm2 <- summary(sah_ppscm2)
plot(sum_ppscm2, levels = "Average")
```



PPSCM Application

We can also add covariates for

- **Weighting:** covariates to weight on
- **Approx. Matching:** covariates to approximately match on *before* weighting
- **Exact Matching:** covariates to exactly match on *before* weighting

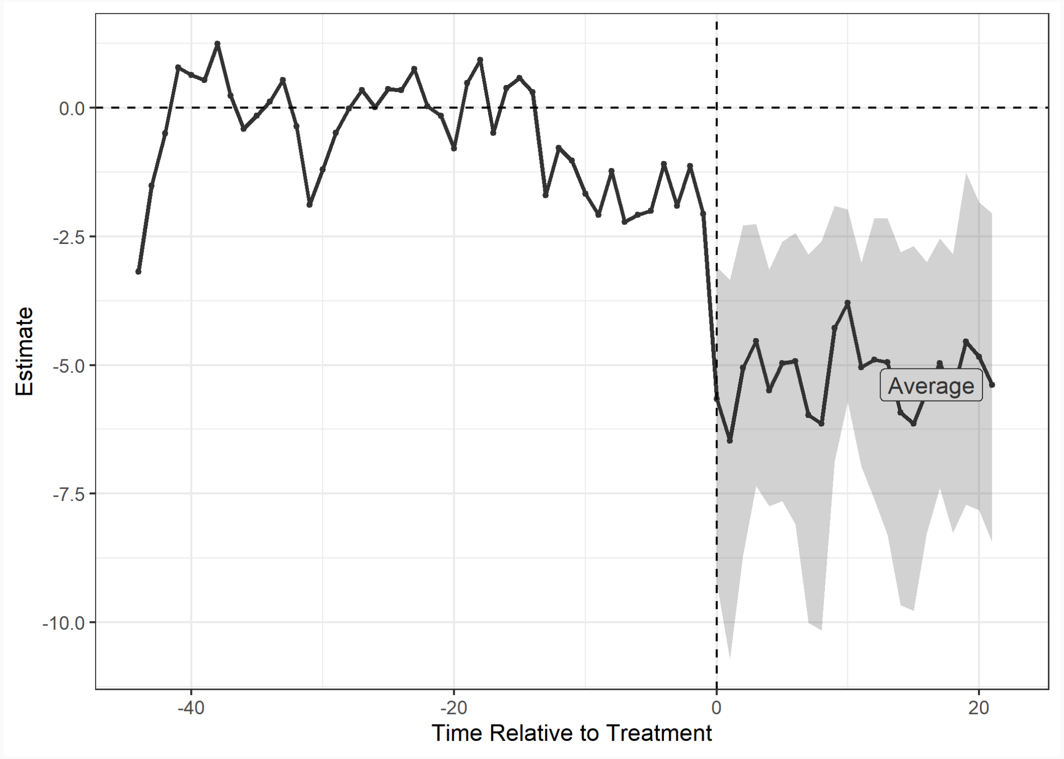
Let's weight on average pre-period values of a few covariates:

```
sah_ppscm3 <- multisynth(cadt ~ post_treat | pct_wfh + pct_pub_trans + pc
                         data = sah, fixedeff = TRUE,
                         lambda = 0.5,
                         n_factors = 2)
sum_ppscm3 <- summary(sah_ppscm3)
```

PPSCM Application

Looking at balance under this specification:

```r
plot(sum_ppscm3, levels = "Average")
```



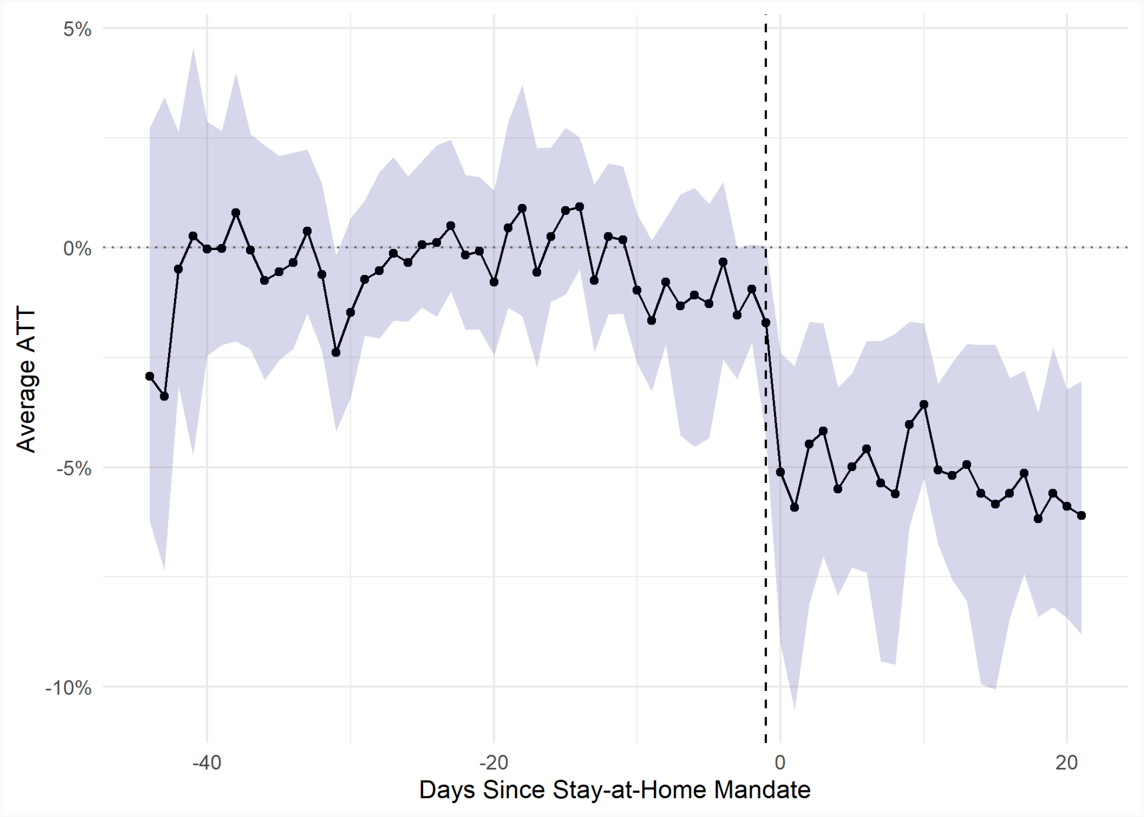
PPSCM Application

Alternatively, we can approximately match on population and democratic vote share in 2016 election using 3 nearest neighbors:

```
sah_ppscm4 ← multisynth(cadt ~ post_treat | pct_wfh + pct_pub_trans + pc
                         data = sah, fixedeff = TRUE,
                         lambda = 0.5,
                         n_factors = 2,
                         k = 3)
sum_ppscm4 ← summary(sah_ppscm4)
```

PPSCM Application

Plotting the new Average ATTs manually with ggplot():



PPSCM Application

Comparing to the standard event study from last lecture:

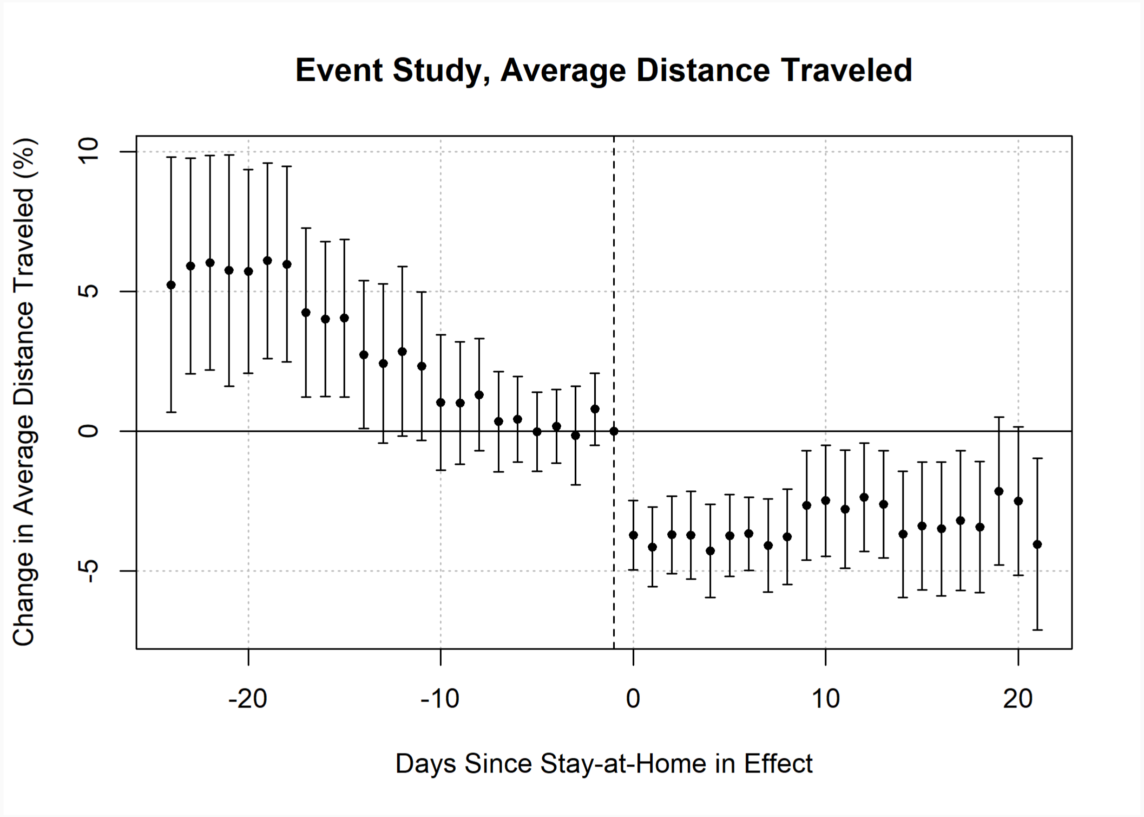


Table of Contents