

# **Empirical Strategy & Project Setup**

## **ECON 692 – Applied Economics Seminar**

---

Andrew Hobbs

University of San Francisco

February 12, 2026

# Today's Agenda

---

| Time | Duration | Activity                            |
|------|----------|-------------------------------------|
| 4:35 | 5 min    | Welcome & check-in                  |
| 4:40 | 30 min   | Methods overview: DiD, SC, SDID     |
| 5:10 | 30 min   | Empirical strategy workshop         |
| 5:40 | 10 min   | Break                               |
| 5:50 | 40 min   | Git & GitHub setup                  |
| 6:30 | 10 min   | Break                               |
| 6:40 | 60 min   | Data sprint: initial code & results |
| 7:40 | 30 min   | Lightning presentations             |
| 8:10 | 5 min    | Wrap-up                             |

# Check-in

---

- How did proposals go? Any surprises?
- Has anyone changed their topic since last week?
- Where are you on finding data?

## Reminder

Data Report is due **Friday, February 20** at 11:59pm.

# Methods Overview

# The Identification Problem

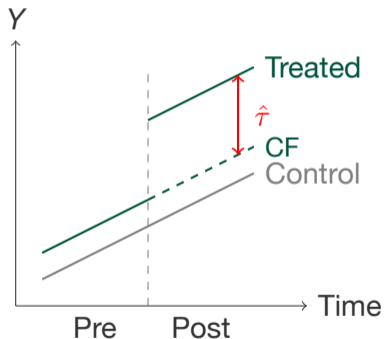
---

You have a research question and data. How do you **identify a causal effect**?

- Last week: potential outcomes, ATE, selection bias
- Today: three methods for **panel data** settings where treatment varies across units and time
  - Difference-in-Differences (DiD)
  - Synthetic Control (SC)
  - Synthetic Difference-in-Differences (SDID)
- Focus: **key assumptions** and **when to use each**

# Difference-in-Differences: Setup

**Setting:** Panel data with two groups and two periods.



- **Treatment group:** affected by policy/event
- **Control group:** not affected
- **Pre-period:** before treatment
- **Post-period:** after treatment

The DiD estimator:  $\hat{\tau}^{\text{did}} = (\bar{Y}_{T,\text{post}} - \bar{Y}_{T,\text{pre}}) - (\bar{Y}_{C,\text{post}} - \bar{Y}_{C,\text{pre}})$

## DiD: A Simple Example

---

Suppose New Jersey raises its minimum wage in 1992. Pennsylvania does not.

|              | Before                | After    | Change |
|--------------|-----------------------|----------|--------|
| NJ (treated) | 20.4 FTE              | 21.0 FTE | +0.6   |
| PA (control) | 23.3 FTE              | 21.2 FTE | -2.1   |
| <b>DiD</b>   | $0.6 - (-2.1) = +2.7$ |          |        |

- NJ employment *rose* relative to PA after the minimum wage increase
- DiD estimate: +2.7 FTE employees per restaurant (Card & Krueger, 1994)
- *For your project:* think about what fills each cell of this table

# DiD: Key Assumption

---

## Parallel Trends Assumption

In the absence of treatment, the treatment and control groups would have followed the **same trajectory** over time.

- This is **untestable** – we can never observe the counterfactual
- But we can check: did trends look parallel *before* treatment?
- Common to show a pre-trends plot as supporting evidence

*For your project: what event or trend could violate parallel trends?*

# DiD: Staggered Treatment

---

**Caution:** With staggered treatment adoption (units treated at different times), naïve two-way fixed effects (TWFE) regression can give **biased** estimates.

- Already-treated units can act as “controls” – biases estimates
- Effects can be heterogeneous across timing groups
- Modern solutions handle this correctly:
  - `did` package (Callaway & Sant’Anna)
  - `fixest::sunab()` (Sun & Abraham)

If your treatment rolls out at different times, use one of these packages instead of basic TWFE.

# DiD: When to Use It

---

## DiD is a good fit when you have:

- A clear treatment/control group distinction
- Pre- and post-treatment observations for both groups
- Reasonable parallel trends argument
- A **substantial number** of treated and control units

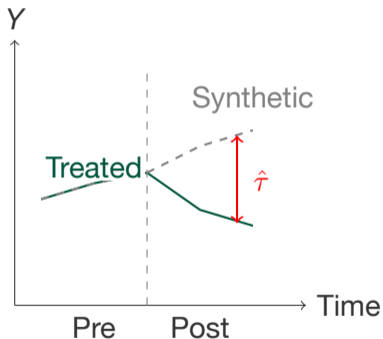
## Packages:

- R: `fixest`, `did`, `DIDmultipligt`
- Python: `pyfixest`, `linearmodels`

# Synthetic Control: Setup

---

**Setting:** A single (or few) treated unit(s), many potential controls.



- **Treated unit:** e.g., California, Germany
- **Donor pool:** untreated units (other states, countries)
- Construct a **weighted combination** of donors that matches the treated unit's *pre-treatment* trajectory
- Treatment effect = gap between treated and synthetic after treatment

# SC: Key Requirements

---

From Abadie (2021), practical requirements for synthetic controls:

1. **Sizable donor pool** of untreated units
2. **Long pre-treatment period** to achieve good match
3. **No spillovers**: treatment of one unit should not affect donors
4. **Convex hull**: treated unit's characteristics should be within the range of the donor pool (no extrapolation) – *relaxed by augmented SC*
5. Treatment is at an **aggregate level** (state, country, region)

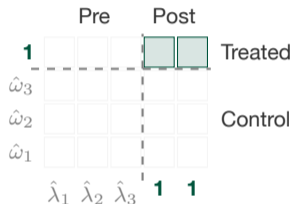
*Does your setting have enough untreated units to build a donor pool?*

## Packages:

- R: Synth, tidysynth, augsynth (ridge-augmented; allows extrapolation beyond convex hull)

# Synthetic Difference-in-Differences

Arkhangelsky, Athey, Hirshberg, Imbens, and Wager (2021):



- Combines the best of DiD and SC
- Like SC: **unit weights**  $\hat{\omega}$  reweight controls
- Novel: **time weights**  $\hat{\lambda}$  emphasize informative pre-periods
- Like DiD: unit FEs, valid large-panel inference

**Why SDID?** More robust than DiD when parallel trends is approximate; more robust than SC with few pre-periods. **Package:** R synthdid

# Comparison

---

|                       | DiD  | Synthetic Control                          | SDID                    |
|-----------------------|--|--|-------------------------|
| <b>Key assumption</b> | Parallel trends                              | Good pre-treatment fit; no spillovers      | Weaker than both        |
| <b>Best when</b>      | Many treated & control units                 | Few treated units, many controls           | Either setting          |
| <b>Limitations</b>    | Biased with staggered treatment (naïve TWFE) | Needs long pre-period; single treated unit | Newer, less established |
| <b>R packages</b>     | fixest, did                                  | Synth, augsynth                            | synthdid                |

*Based on your data structure, which method seems most promising for your project?*

# Continuous Treatments: The Standard Approach

Many DiD applications involve treatments that are not simply on/off:

- Varying tax rates, pollution levels, subsidy amounts, minimum wages
- Distance to a facility, intensity of exposure to a policy

**The standard approach** – TWFE with a continuous dose variable:

$$Y_{it} = \underbrace{\theta_t}_{\text{time FE}} + \underbrace{\eta_i}_{\text{unit FE}} + \beta^{twfe} \underbrace{D_i \cdot \text{Post}_t}_{\text{dose} \times \text{post}} + v_{it}$$

Researchers typically interpret  $\beta^{twfe}$  as the “effect of one more unit of treatment.”

*But is this correct?*

# Continuous Treatments: The Problem

---

**The problem** (Callaway, Goodman-Bacon, & Sant'Anna, 2024):

- $\beta^{twfe}$  is a weighted average of dose-specific effects, but **weights can be negative**
- Even with non-negative weights, TWFE conflates two different things:

## Level effects

Does getting dose  $d$  vs. 0 change outcomes?

*Requires:* standard parallel trends

## Causal responses

Does a marginal increase in dose change outcomes?

*Requires:* strong parallel trends (restricts heterogeneity)

These are *different parameters* requiring *different assumptions*. TWFE mixes them.

# Continuous Treatments: Solutions

---

## Two distinct causal parameters:

---

|                   | Level effects<br>(ATT at dose $d$ )                 | Causal responses<br>(marginal effect)                               |
|-------------------|---|---|
| <b>Assumption</b> | Standard parallel trends<br>(on untreated outcomes) | Strong parallel trends<br>(restricts heterogeneity<br>across doses) |
| <b>Compares</b>   | Treated vs. untreated                               | Adjacent dose groups  |

---

## Packages:

- R: `contdid` (Callaway et al.) – level effects and causal response curves as functions of dose
- de Chaisemartin et al. (2024) – extends to **no stayers** (every unit's treatment changes); R/Stata: `did_multiplegt_stat`, `did_multiplegt_dyn`

**Bottom line:** Don't run TWFE with a continuous  $D$ . Use `contdid`.

# Empirical Strategy Workshop

# What Is an Empirical Strategy?

---

Your empirical strategy is the **plan for answering your research question with data**.

Four components:

1. **Method:** Which estimation approach? (DiD, SC, SDID, OLS, IV, RDD, ...)
2. **Data:** What variables, from where, at what level?
3. **Identification:** What variation are you exploiting? Why is it credible?
4. **Threats:** What could go wrong? How will you address it?

# Strategy Worksheet

---

Take 10 minutes. Write your answers on paper or in a document.

1. What is your **treatment or intervention**?
2. What is your main **outcome variable**?
3. What is your **comparison/control group**?
4. What **time periods** do you have data for?
5. What are the main **threats to identification**?
6. Which **method** fits your setting, and why?

**Goal:** Write a 1-paragraph empirical strategy statement.

# Choosing Packages

---

Once you have a method in mind, pick your tools:

## If using R:

- DiD → `fixest` (fast, flexible), `did` (Callaway–Sant’Anna estimator)
- SC → `augsynth` (ridge-augmented SC), `tidysynth` (tidy interface to Synth)
- SDID → `synthdid`
- General regression → `fixest`, `lfe`, `estimatr`

## If using Python:

- DiD → `pyfixest`, `linearmodels`
- SC → `SparseSC`, or call R from Python via `rpy2`
- General → `statsmodels`, `linearmodels`

**Task:** Identify the specific package(s) you’ll use. Install them today.

# Share with a Partner

---

In pairs, spend 10 minutes each:

1. Read your empirical strategy statement to your partner
2. Partner asks:
  - What is the biggest threat to your identification?
  - Is there an alternative control group you could use?
  - Does the parallel trends / no spillovers assumption seem reasonable?
3. Revise your statement based on feedback

# Git & GitHub Setup

# Why Version Control?

---

## Without version control:

- `analysis_v2_final_FINAL.R`
- “Which version has the fix?”
- “I accidentally deleted my code”
- Impossible to reproduce results from 3 months ago

## With Git:

- Complete history of every change
- Undo mistakes instantly
- Collaborate without overwriting
- **Portfolio for job applications**

# Git Concepts (Just Three Things)

---

1. **Repository (repo):** A folder that Git tracks.
  - Every change you *commit* is saved forever.
2. **Commit:** A snapshot of your project at a point in time.
  - Like “Save As” but you keep *all* previous saves.
  - Each commit has a message describing what changed.
3. **Push / Pull:** Sync between your computer and GitHub.
  - **Push:** upload your commits to GitHub.
  - **Pull:** download changes from GitHub.

# Step 1: Install Git & Configure

---

Open your terminal and check if Git is installed:

```
git --version
```

If not installed: **Mac:** `xcode-select --install`    **Windows:**  
<https://git-scm.com>

Configure your identity (one-time setup):

```
git config --global user.name "Your Name"  
git config --global user.email "your.email@usfca.edu"
```

## Step 2: GitHub Education

---

GitHub offers a free **Student Developer Pack**:

- Unlimited private repositories
- GitHub Copilot (AI code assistant)
- Free domain name and other tools

**To sign up:**

1. Go to <https://github.com> – create an account if you don't have one
2. Visit <https://education.github.com/pack>
3. Click “Get your pack” and verify with your `@usfca.edu` email
4. Approval is usually instant or within a few hours

Do this now if you haven't already.

## Step 3: Create a Repository on GitHub

---

1. Go to <https://github.com> and click **“New repository”**
2. Name it something descriptive: e.g., econ692-capstone
3. Set it to **Private**
4. Check **“Add a README file”**
5. Check **“Add .gitignore”** and select the R or Python template
6. Click **“Create repository”**

## Step 4: Clone to Your Computer

---

On your repo page, click the green “**Code**” button and copy the URL.

In your terminal:

```
cd ~/Documents # or wherever you keep projects  
git clone https://github.com/yourusername/econ692-capstone.git  
cd econ692-capstone
```

You now have a local copy of your repository.

## Step 5: Project Structure

---

Create a clean folder structure for your project:

```
mkdir data code output paper
```

|         |                                   |
|---------|-----------------------------------|
| data/   | Raw and cleaned data files        |
| code/   | R scripts, Python notebooks, etc. |
| output/ | Figures, tables, results          |
| paper/  | Your written report / slides      |

Tip: Add a brief `README.md` describing your project.

## Step 6: The .gitignore File

---

Data files are often too large for GitHub (100 MB limit). The .gitignore file tells Git which files to *never* track.

```
# Data files (too large / sensitive)
data/
# R artifacts
.Rhistory
.RData
.Rproj.user/
# Python artifacts
__pycache__/
.ipynb_checkpoints/
```

GitHub's auto-generated .gitignore covers most of this. Just add data/.

## Step 7: Your First Commit

---

The three-step cycle you'll use every time:

```
# 1. Stage: tell Git which files to include  
git add .
```

```
# 2. Commit: save a snapshot with a message  
git commit -m "Set up project structure"
```

```
# 3. Push: upload to GitHub  
git push
```

**Let's do this together now.**

# Git Quick Reference

---

| Command                          | What it does                         |
|----------------------------------|--------------------------------------|
| <code>git status</code>          | See what's changed since last commit |
| <code>git add .</code>           | Stage all changes for commit         |
| <code>git add file.R</code>      | Stage a specific file                |
| <code>git commit -m "msg"</code> | Save a snapshot                      |
| <code>git push</code>            | Upload commits to GitHub             |
| <code>git pull</code>            | Download changes from GitHub         |
| <code>git log --oneline</code>   | View commit history                  |
| <code>git diff</code>            | See what changed (before staging)    |

**Golden rule:** Commit early, commit often. Write clear messages. Small, frequent commits are much better than one giant commit at the end.

# Data Sprint

# Data Sprint Goals

---

You have **60 minutes**. Try to accomplish as much as you can:

1. **Download your data** (or verify you already have it)
2. **Load it** into R or Python
3. **Summary statistics**: how many observations? Key variables? Missing data?
4. **One exploratory plot**: histogram, time series, scatterplot – anything
5. **Commit your code** to your new Git repository

Don't worry about perfection. The goal is to **get your hands on the data** and verify it's usable.

I'll be walking around to help. Raise your hand if you're stuck.

# Lightning Presentations

---

Each person gets **2 minutes**. Cover these four points:

1. **What are you studying?**
  - Research question in one sentence
2. **What data do you have?**
  - Source, size, key variables
3. **What did you accomplish today?**
  - Empirical strategy, git setup, initial analysis
4. **Goals for next week?**
  - What's your plan between now and next class?

# Wrap-Up & Next Week

---

## Due this week:

- Weekly progress report (Friday)
- **Data Report – due Friday, February 20**

## Next week (Week 4):

- Research design: causal pathways and DAGs
- Empirical strategy drafting
- More time to work on your project

## Before next class:

- Finish downloading and exploring your data
- Push your initial code to GitHub
- Start thinking about your DAG (what causes what?)