Complete Project Walkthrough

From Raw Data to Publication-Ready Results

Theodore Caputi

14.33 · MIT · Spring 2026

The Research Question

Does a state-level policy reduce fatal car crashes?

- States adopt a policy at different times
- We observe crash data before and after adoption
- This is a **staggered difference-in-differences** setting

The Estimating Equation

$$Y_{st} = \gamma_s + \delta_t + \beta \cdot (\text{Treat} \times \text{Post})_{st} + \varepsilon_{st}$$

- Y_{st}: fatal crashes in state s, year t
- γ_s: state fixed effects
- δ_t: year fixed effects
- $(\text{Treat} \times \text{Post})_{st} = 1$ after state s adopts the policy
- β: the treatment effect we want to estimate

Question: What do the subscripts st tell us?

What the Subscripts Tell Us

The subscripts st tell us everything:

- $s =$ state, $t =$ year
- This is **panel data**: repeated observations of the same units over time
- Each observation is a **state-year**

One row per state per year

Every decision we make about the data flows from this.

What Must the Data Look Like?

The ideal **analysis dataset**:

state_fips	year	fatal_crashes	post_treated	population	median_incom
1	2000	842	0	4447100	35120
1	2001	819	0	4467634	35840
1	2002	856	1	4480089	36290
6	2000	3753	0	33871648	42160
6	2001	3650	0	34479458	42880

One row per state-year. Columns for the outcome, treatment indicator, and controls.

What's the Level of Analysis?

State × **Year**

This determines:

- How we **collapse** raw data (aggregate to state-year)
- How we **merge** datasets (match on state-year or state)
- How we **cluster** standard errors (at the state level)
- What **fixed effects** we include (state and year)

Rule: If you're unsure what to do at any step, go back to the estimating equation.

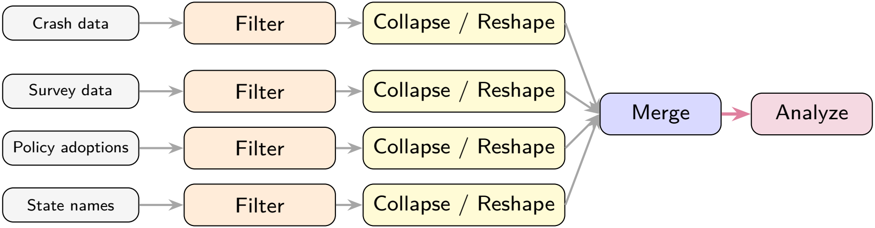
But Our Raw Data Doesn't Look Like This

We have **four separate datasets**:

1. **Crash data** — one row per *crash* (not state-year!)
2. **Demographic survey** — individual-level survey data (must clean and collapse to state-year)
3. **Policy adoptions** — one row per state with adoption year
4. **State names** — FIPS codes to state names and regions

We need to **build** the analysis dataset from these raw ingredients.

The Roadmap



- **Filter** → **Collapse / Reshape**: raw data → analysis-ready panels
- **Merge**: combine all four on `state_fips` (and `year`)
- **Analyze**: regressions, event studies, tables, figures

Directory Structure

```
pkg-stata/
|-- master.do
|-- build/
|   |-- code/
|   |   |-- 01_filter_crashes.do
|   |   |-- 02_collapse_crashes.do
|   |   |-- 03_reshape_crashes.do
|   |   |-- 04_append_demographics.do
|   |   |-- 05_collapse_demographics.do
|   |   |-- 06_merge_datasets.do
|   |-- input/       (raw data, never modified)
|   |   |-- crash_data.csv
|   |   |-- demographic_survey/
|   |   |-- policy_adoptions.csv
|   |   |-- state_names.csv
|   |-- output/      (processed data)
|-- analysis/
    |-- code/
    |   |-- 01_descriptive_table.do
    |   |-- 02_dd_regression.do
    |   |-- 03_event_study.do
    |   |-- 04_dd_table.do
    |   |-- 05_iv.do
    |   |-- 06_rd.do
```

Key Principles

1. **Raw data is read-only.** Never modify files in `build/input/`.
2. **Build vs. analysis separation.** Build scripts process data; analysis scripts produce results.
3. **Numbered scripts.** `01_`, `02_`, ... run in order. No ambiguity.
4. **Reproducibility.** Anyone can run `master.do` and regenerate everything.
5. **Save intermediate files.** Separates expensive processing from fast analysis iterations.

The Master File

```
clear all
set more off

global root "."
global build "$root/build"
global analysis "$root/analysis"

cd "$root"

* Build
do "$build/code/01_filter_crashes.do"
do "$build/code/02_collapse_crashes.do"
do "$build/code/03_reshape_crashes.do"
do "$build/code/04_append_demographics.do"
do "$build/code/05_collapse_demographics.do"
do "$build/code/06_merge_datasets.do"

* Analysis
do "$analysis/code/01_descriptive_table.do"
do "$analysis/code/02_dd_regression.do"
do "$analysis/code/03_event_study.do"
do "$analysis/code/04_dd_table.do"
do "$analysis/code/05_iv.do"
do "$analysis/code/06_rd.do"
```

Reading Data

CSV files:

```
import delimited "$build/input/crash_data.csv", clear
describe
summarize
```

Reading one survey file:

```
import delimited ///
    "$build/input/demographic_survey/demographic_survey_2000.csv
        ", ///
    clear
```

Saving:

```
save "$build/output/crashes_state_year.dta", replace
```

Always use `$build/input/` for raw data, `$build/output/` for processed data.

Remember Our Goal

We need to go from this:

state_fips	year	severity	crash_id
1	2000	fatal	00001
1	2000	serious	00002
1	2000	minor	00003
1	2000	fatal	00004

To this:

state_fips	year	fatal_crashes	serious_crashes	post_treated
1	2000	842	1203	0
1	2001	819	1187	0

The estimating equation tells us exactly what this must look like.

Step 1: Filter

Drop observations we don’t need:

```
import delimited "$build/input/crash_data.csv", clear

* Drop minor crashes -- we only care about
* fatal and serious
drop if severity == "minor"
```

Why filter first?

- Reduces dataset size before expensive operations
- Makes subsequent steps faster and cleaner
- Always document what you drop and why

Step 2: Collapse

Go from crash-level to state-year-severity counts:

```
* Create a counter variable
gen one = 1

* Collapse: count crashes by state-year-severity
collapse (sum) n_crashes = one, ///
    by(state_fips year severity)
```

Before: One row per crash
After: One row per state-year-severity

Warning: `collapse` replaces your dataset! This is why we save intermediate files.

Step 3: Reshape

Each severity type becomes its own column:

```
* Reshape from long to wide
reshape wide n_crashes, ///
    i(state_fips year) j(severity) string

* Rename for clarity
rename n_crashesfatal fatal_crashes
rename n_crashesserious serious_crashes
```

Before:

state_fips	year	severity	n_crashes
1	2000	fatal	842
1	2000	serious	1203

After:

state_fips	year	fatal_crashes	serious_crashes
1	2000	842	1203

Create Variables & Save

```
* Derived variables
gen total_crashes = fatal_crashes + serious_crashes
gen fatal_share = fatal_crashes / total_crashes

* Save intermediate file
save "$build/output/crashes_state_year.dta", replace
```

Why save intermediate files?

- Separates expensive processing from fast analysis iterations
- If analysis code changes, you don't re-run the build
- Makes debugging easier — you can inspect the data at each stage

Collapsing Demographics: Loop & Append

The demographic survey is split into yearly files. We loop over years and append:

```
* Convert each CSV to .dta
forvalues y = 1995/2015 {
    import delimited ///
        "$build/input/demographic_survey/demographic_survey_‘y’.
            csv", clear
    gen year = ‘y’
    save "$build/output/survey_‘y’.dta", replace
}

* Append all years
clear
forvalues y = 1995/2015 {
    append using "$build/output/survey_‘y’.dta"
}
```

Pattern: First loop converts CSVs to `.dta`. Second loop appends them all together.

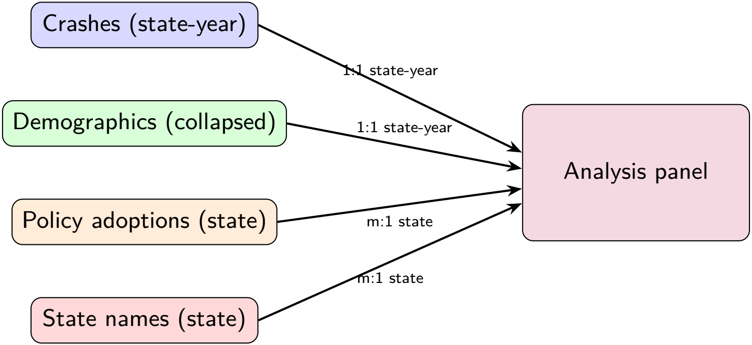
Collapsing Demographics: Clean & Collapse

```
* Inspect and filter
tab year
tab state_fips if state_fips == 51
drop if state_fips == 51          // DC
drop if year < 2000

* Clean: income has dollar signs and commas
destring income, replace ignore("$,")

* Weighted collapse to state-year
collapse (rawsum) population = weight ///
    (mean) median_income = income ///
    (mean) pct_urban = urban ///
    [aweight=weight], by(state_fips year)

* Verify expected row count
assert _N == 800

save "$build/output/demographics_state_year.dta", replace
```

New commands: `destring`, `tab`, `assert`, `log using`/`log close`.

Merging Multiple Datasets

We have 4 datasets to combine:



- **1:1**: one observation per key in both datasets
- **m:1**: many state-years match one state-level record

Merge Code

```
* Load main dataset
use "$build/output/crashes_state_year.dta", clear

* Merge 1: Demographics (1:1 on state-year)
merge 1:1 state_fips year ///
    using "$build/output/demographics_state_year.dta", ///
    keep(match) nogen

* Merge 2: Policy adoptions (m:1 on state)
merge m:1 state_fips ///
    using "$build/output/policy_adoptions.dta", ///
    keep(master match) nogen

* Merge 3: State names (m:1 on state)
merge m:1 state_fips ///
    using "$build/output/state_names.dta", ///
    keep(match) nogen
```

Always check `_merge` values before dropping!

Understanding Merge Types

1:1 — each row matches exactly one row in the other dataset.
m:1 / **1:m** — many rows match one row (or vice versa).

Example: Merging individual-level data with state-year unemployment:

Individual × state × year:

ID	State	Year	Income
1	MA	2000	45000
2	MA	2000	52000
3	MA	2001	48000
4	PA	2000	50000

State × year:

State	Year	Unemp.
MA	2000	3.4%
MA	2001	4.5%
PA	2000	6.7%

This is an **m:1** merge: many individuals per state-year, one unemployment rate per state-year. Both MA individuals in 2000 get matched to 3.4%.

The keep() Option

In Stata, your original data is “master” and the merged-in data is “using.” After merging, each observation is one of:

- **master** — rows from master with no match in using
- **match** — rows that matched
- **using** — rows from using with no match in master

Example: You have a balanced state-year panel. You want to merge in earthquake counts, collapsed to state-year. But states with *zero earthquakes* don’t appear in the earthquake data!

```
* BAD: drops states with no earthquakes!
merge 1:1 state year using "earthquakes.dta", keep(match)

* GOOD: keeps all original observations
merge 1:1 state year using "earthquakes.dta", ///
    keep(master match) nogen
replace num_earthquakes = 0 if missing(num_earthquakes)
```

Rule of Thumb for Merges

Start with a dataset containing all the observations you want. Use `keep(master match)` almost always. Then `replace v = 0 if missing(v)` for merged variables.

Why this works:

- Your panel stays balanced — no silent observation drops
- Unmatched rows get missing values, which you replace with the correct value (often 0)
- In R/Python: use `how='left'` (left join) + `fillna(0)`

In our project: Merge 2 uses `keep(master match)` because not all states adopted the policy. States that never adopt get `adoption_year = .` (missing), which is correct.

Creating the Treatment Indicator

```
* Treatment indicators
gen ever_treated = !missing(adoption_year)
gen post_treated = (year >= adoption_year ///
    & ever_treated)
```

Key details:

- States that never adopt: `adoption_year` is missing ⇒ `post_treated = 0` always
- The `!missing()` guard is essential — without it, Stata treats missing as $+\infty$

```
* Other useful variables
gen log_pop = ln(population)

* Save final analysis panel
save "$build/output/analysis_panel.dta", replace
```

Descriptive Table

Show the reader what the data looks like **before** estimation:

	Treated × Post	Treated × Pre	Untreated
Fatal Crashes	742.3	891.4	654.2
Serious Crashes	1,108.6	1,245.1	987.3
Population	5,231,400	5,102,300	3,891,200
Median Income	41,520	38,740	36,890
Pct. Urban	72.4	71.8	63.1
N	312	480	1,108

What to look for: Pre-treatment balance between treated and untreated groups.

Descriptive Table: Code

```
* Label variables
label variable fatal_crashes "Fatal Crashes"
label variable serious_crashes "Serious Crashes"
label variable population "Population"
label variable median_income "Median Income"
label variable pct_urban "Pct. Urban"

* Create group variable
gen group = 3 if missing(adoption_year)
replace group = 1 if !missing(adoption_year) ///
    & year >= adoption_year
replace group = 2 if !missing(adoption_year) ///
    & year < adoption_year

label define grp 1 "Treated After" ///
    2 "Treated Before" 3 "Untreated"
label values group grp
```

Descriptive Table: dtable

```
dtable fatal_crashes serious_crashes ///
    total_crashes fatal_share ///
    population median_income pct_urban ///
    i.census_region, ///
    by(group) ///
    nformat(%14.2fc mean sd) ///
    sample(, statistics(freq) place(seplabels))

collect export ///
    "$analysis/output/tables/descriptive_table.tex", ///
    tableonly replace
```

`dtable` is Stata 18's built-in descriptive statistics command. It handles formatting, group comparisons, and export automatically.

Back to the Equation

$$Y_{st} = \gamma_s + \delta_t + \beta \cdot (\text{Treat} \times \text{Post})_{st} + \varepsilon_{st}$$

Now we have the data. Run the regression:

```
use "$build/output/analysis_panel.dta", clear

reghdfe fatal_crashes post_treated, ///
    absorb(state_fips year) ///
    vce(cluster state_fips)
```

- `reghdfe`: fast fixed effects estimation (install: `ssc install reghdfe`)
- `absorb()`: state and year fixed effects (γ_s, δ_t)
- `vce(cluster state_fips)`: cluster SEs at the treatment level

Event Study: Why?

The TWFE regression gives us **one number** ($\hat{\beta}$).

An event study lets us:

- See **dynamic effects** — does the effect grow over time?
- Test **parallel trends** — are pre-treatment coefficients near zero?
- Detect **anticipation effects** — did behavior change before the policy?

$$Y_{st} = \gamma_s + \delta_t + \sum_{j=-5}^{5} \beta_j \cdot \mathbf{1}[\text{time-to-treat}_{st} = j] + \varepsilon_{st}$$

Omit $j = -1$ as the reference period $\Rightarrow$ all coefficients are relative to the year before treatment.

Event Study: Setup

```
* Time-to-treatment
gen time_to_treat = year - adoption_year
replace time_to_treat = -99 ///
    if missing(adoption_year)

* Create event-time indicators
forvalues t = -5/5 {
    if ‘t’ < 0 {
        local name "m‘= abs(‘t’)’"
    }
    else {
        local name "‘t’"
    }
    gen rel_‘name’ = (time_to_treat == ‘t’)
}

* Bin endpoints (everyone beyond window)
replace rel_m5 = (time_to_treat <= -5) ///
    & !missing(adoption_year)
replace rel_5  = (time_to_treat >= 5) ///
    & !missing(adoption_year)
```

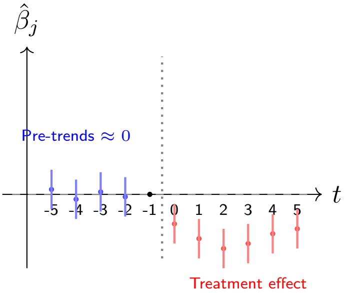
Event Study: Regression

```
* Omit rel_m1 (t = -1 is the reference)
reghdfe fatal_crashes ///
    rel_m5 rel_m4 rel_m3 rel_m2 ///
    rel_0 rel_1 rel_2 rel_3 rel_4 rel_5 ///
    log_pop, ///
    absorb(state_fips year) ///
    vce(cluster state_fips)
```

Note:

- `rel_m1` is excluded — this is our reference period $(t = -1)$
- `rel_m5` and `rel_5` are *binned endpoints* — they capture everyone ≤ -5 or ≥ 5 years from treatment
- We include `log_pop` as a time-varying control

Event Study: The Plot



What to look for:

- Pre-treatment coefficients near zero $\Rightarrow$ parallel trends plausible
- Discontinuous jump at $t = 0 \Rightarrow$ treatment effect
- No pre-trends drifting toward the effect $\Rightarrow$ no anticipation

The reference period $(t = -1)$ is normalized to zero.

Making Figures Publication-Ready

Schwabish (2014) principles:

1. **Labels, not legends** — label lines/points directly
2. **Horizontal text** — no rotated axis labels
3. **Eliminate chartjunk** — white background, minimal grid
4. **Single accent color** — one color for emphasis

```
twoway (rarea ub lb t, color("44 95 138%20")) ///
       (connected coef t, ///
            mcolor("44 95 138") lcolor("44 95 138")), ///
    yline(0, lcolor(gs8)) ///
    xline(-0.5, lcolor(gs8) lpattern(dash)) ///
    xtitle("Years Relative to Policy Adoption") ///
    legend(off) ///
    graphregion(color(white))

graph export "$analysis/output/figures/event_study.png", ///
    replace width(2400)
```

DD Table with Subgroups

Dependent Variable:	Fatal Crashes				Serious
Sample:	All	All	South	Non-South	All
Model:	(1)	(2)	(3)	(4)	(5)
Variables					
Treat × Post	−45.3***	−42.1***	−38.7**	−44.9***	−12.4
	(12.8)	(13.1)	(18.4)	(14.2)	(22.6)
Log Population		28.4**	31.2*	25.1**	41.7**
		(11.3)	(16.7)	(12.0)	(18.5)
Fixed Effects					
State FE	Yes	Yes	Yes	Yes	Yes
Year FE	Yes	Yes	Yes	Yes	Yes
N	1,900	1,900	680	1,220	1,900
R^2	0.847	0.852	0.831	0.869	0.791

esttab Code

```
esttab m1 m2 m3 m4 m5 ///
  using "$analysis/output/tables/dd_results.tex", ///
  replace se(%9.3f) b(%9.3f) ///
  star(* 0.10 ** 0.05 *** 0.01) ///
  nomtitles nonumbers label fragment ///
  prehead("\begin{tabular}{l*{5}{c}}" ///
    "\midrule \midrule" ///
    "Dep. Var.: " ///
    "&\multicolumn{4}{c}{Fatal Crashes}" ///
    "&Serious\\" ///
    "\cmidrule(lr){2-5} \cmidrule(lr){6-6}" ///
    "Sample: & All & All & South" ///
    "& Non-South & All \\" ///
    "Model: & (1) & (2) & (3)" ///
    "& (4) & (5) \\" ///
    "\midrule" ///
    "\emph{Variables} \\") ///
  stats(state_fe year_fe N r2, ///
    labels("State FE" "Year FE" ///
    "Observations" "R\$^2\$") ///
    fmt(%s %s %9.0fc %9.3f)) ///
  postfoot("\midrule \midrule" ///
    "\end{tabular}")
```

Table Formatting Checklist

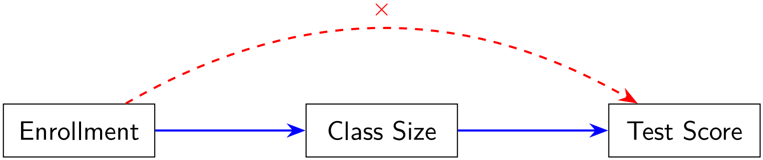
1. **Use booktabs**: `\toprule`, `\midrule`, `\bottomrule`. No vertical lines. No `\hline`.
2. **Standard errors in parentheses** below each coefficient.
3. **Pretty labels**: `log_pop` → "Log Population". Never show raw variable names.
4. **Model numbers on their own line**: (1), (2), (3) above the column headers.
5. **Fixed effects indicators**: "Yes" / "No" rows at the bottom.
6. **Group columns with `\cmidrule`**: visually separate subgroups and alternative outcomes.
7. **Significance stars**: standard convention (*10%, **5%, ***1%) with a note.

Instrumental Variables: Setup

Research question: Does class size affect test scores?

Problem: Class size is endogenous (schools with more resources may have both smaller classes and better outcomes).

Instrument: School enrollment (inspired by Angrist & Lavy, 1999).



Blue: causal pathway. Red dashed: exclusion restriction (no direct effect).

IV: First Stage, Reduced Form, 2SLS

```
* First stage: instrument -> endogenous variable
regress class_size enrollment pct_disadvantaged, ///
    vce(robust)
estimates store first_stage
test enrollment          // F-test for relevance

* Reduced form: instrument -> outcome
regress test_score enrollment pct_disadvantaged, ///
    vce(robust)
estimates store reduced_form

* 2SLS: the IV estimate
ivregress 2sls test_score pct_disadvantaged ///
    (class_size = enrollment), ///
    vce(robust) first
estimates store iv_2sls

* OLS for comparison (biased)
regress test_score class_size pct_disadvantaged, ///
    vce(robust)
estimates store ols
```

IV Results Table

Dep. Var.:	Class Size	Test Score		
Model:	First Stage	Reduced Form	2SLS	OLS
	(1)	(2)	(3)	(4)
Variables				
Class Size			−0.612**	−0.143
			(0.289)	(0.098)
Enrollment	0.034***	−0.021**		
	(0.008)	(0.010)		
Pct. Disadvantaged	0.152**	−0.641***	−0.548***	−0.614***
	(0.062)	(0.041)	(0.078)	(0.039)
N	2,019	2,019	2,019	2,019
R^2	0.231	0.507	0.481	0.512

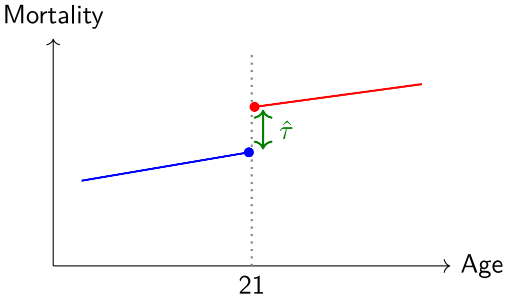
Robust standard errors in parentheses

Instrument: School enrollment

Regression Discontinuity: Setup

Research question: Does the legal drinking age affect mortality?

Design: Individuals just above vs. just below age 21 (inspired by Carpenter & Dobkin, 2009).



The **jump** at the cutoff estimates the causal effect of legal drinking access.

RD: Linear and Polynomial Specs

```
import delimited "$analysis/code/rd_data.csv", clear

* Running variable interaction
gen days_x_over21 = days_from_21 * over_21

* Linear RD (bandwidth = 365 days)
regress mortality_rate over_21 ///
    days_from_21 days_x_over21 ///
    if abs(days_from_21) <= 365, vce(robust)
estimates store rd_linear

* Quadratic RD
gen days_sq = days_from_21^2
gen days_sq_x_over21 = days_sq * over_21

regress mortality_rate over_21 ///
    days_from_21 days_x_over21 ///
    days_sq days_sq_x_over21 ///
    if abs(days_from_21) <= 365, vce(robust)
estimates store rd_quadratic
```

RD: Robustness

Key robustness checks for RD:

1. **Bandwidth sensitivity:** Does the estimate change with different windows?
2. **Polynomial order:** Linear, quadratic, cubic — results should be stable.
3. **Placebo cutoffs:** No jump at other values of the running variable.

Dep. Var.:	Mortality Rate		
Model:	Linear (1)	Quadratic (2)	Cubic (3)
Over 21	0.0847*** (0.0214)	0.0791*** (0.0231)	0.0823*** (0.0258)
N	8,412	8,412	8,412

Stable across specifications ⇒ robust.

Key Lessons

1. **Start at the end.** Write down the estimating equation first. Everything else follows.
2. **Every data decision flows from the equation.** The subscripts tell you the level of analysis, which determines how you collapse, merge, and cluster.
3. **Raw data is read-only.** Never modify input files.
4. **Save intermediate files.** Separate expensive build steps from fast analysis iterations.
5. **Make figures and tables publication-ready.** Labels not legends, pretty variable names, booktabs formatting.
6. **Anyone should be able to replicate your results** by running `master.do`.