Repository Documentation

May 19, 2025

This PDF is generated from `writeup/tex/README.tex`. To re-build it, run

```
pdflatex -interaction=nonstopmode writeup/tex/README.tex
```

All content below explains
the directory structure, build workflow, and research pipeline for this project.

Contents

1 High-level Overview

The repository is structured around three tasks that mirror the typical empirical workflow of the project:

1) **Data preparation** – Stata `.do` files in `src/` ingest the raw data and construct analysis-ready panels.
2) **Estimation/specification** – individual empirical models live in `spec/`. Each script draws on the prepared panels and writes tidy results.
3) **Reporting** – Python and LaTeX code inside `writeup/` turns the raw results into publication-quality tables, figures, and the final paper.

A Makefile at `writeup/Makefile` orchestrates the reporting pipeline so that a single command produces the paper PDF together with clean LaTeXtables.

2 Top-level Directory Layout

`data/` Raw inputs as shipped by external sources (`data/raw`) and processed panels generated by our build scripts (`data/processed`). Small samples that *are* version-controlled reside in `data/samples`.

`src/` Re-usable Stata build scripts. Each file constructs a particular panel (e.g. firm-level, worker-level) starting entirely from the raw data.

`spec/` A "kitchen-sink" of empirical specifications. Every `.do` file is self-contained: it loads the prepared data, runs the model(s), and writes two types of artefacts:

a) `results/cleaned/` – publishable LaTeX tables.
b) `results/raw/` – result dumps used for robustness checks and diagnostics.

`py/` Lightweight Python helpers. They post-process the Stata output (e.g. merge standard errors, rename variables) and generate additional figures that are easier to code in Python.

`writeup/` Everything related to the paper: a `Makefile`, intermediate build folder, final PDF, and the LaTeX sources. All tables under `results/cleaned` are copied here when the paper builds.

`results/` Automatically generated outputs from the Stata and Python code. The directory is subdivided into `raw/`, `cleaned/`, and `figures/`.

3 Workflow in Detail

1. Data preparation (Stata `src/`)

1. All build scripts source `src/globals.do` first. The file defines global macros (e.g. `$raw_data`, `$processed_data`) so that every subsequent script writes to a consistent location.

2. Each build script checks whether its target panel already exists in `data/processed`. If yes, nothing happens; if not, the script performs the necessary joins, merges, and reshapes.

2. Estimation (Stata `spec/`)

Every specification script follows the same skeleton:

1. Load the required panel(s) from `data/processed`.
2. Run the main model plus any robustness checks.
3. Dump full result matrices to `results/raw`.
4. Write publication-ready tables to `results/cleaned` using a common `outreg2` template so that the look and feel are consistent across specifications.

3. Reporting (`writeup/`)

1. The Makefile builds all required tables by running the Python helpers in `writeup/py/`. Each helper reads the raw CSV dumps and turns them into compact LaTeXcode under `results/cleaned`.
2. After the tables are up-to-date, the Makefile compiles the main paper (`writeup/tex/results/consolidated-` with `pdflatex`.
3. A convenience target `make deploy` syncs the paper PDF and cleaned tables to an Overleaf-backed Dropbox folder so that collaborators can edit the manuscript online.

4 Typical Usage

Building the full paper

Run the following from the repository root (requires Stata, Python $\geq$ 3.9, and TEX Live):

```
# 1) Build data (only needs to run once)
stata -e src/build_firm_panel.do
stata -e src/build_user_panel.do

# 2) Run specifications
stata -e spec/firm_scaling.do
stata -e spec/worker_event_study.do

# 3) Build the paper
make -C writeup
```

Quick rebuild after code changes

If you only tweaked one specification, simply re-run that `.do` file followed by `make -C writeup report`. The existing panels and unaffected tables will be left untouched, resulting in a much faster build.

5 Development Notes

- **Version control.** Large raw datasets are *not* committed—only scripts and small samples live in Git. The canonical raw inputs reside on a shared network drive.
- **Reproducibility.** All generated artefacts depend solely on committed code and the raw data referenced in `globals.do`. Running the three-step workflow above on a clean machine should reproduce every result.
- **Python dependencies.** A minimal `requirements.txt` is provided at the repository root. Create a fresh virtual environment and install with `pip install -r requirements.txt`.
- **Styling / linting.** The project follows the standard Stata style guide and `black` for Python.

6 FAQ

Q: "I changed a raw data file—why is the script not re-building?"
A: Delete the corresponding processed file(s) in `data/processed` and rerun the build script. The scripts only rebuild panels that do not yet exist.

Q: "A table shows `(omitted)` for some coefficient."
A: Stata omits perfectly collinear regressors. Double-check the fixed-effect structure and ensure you are not including the same categorical variable twice.

Q: "How do I add a new specification?"
A:

1. Duplicate the template at `spec/template.do` (or any existing `.do` file).
2. Point the script towards the relevant panel(s).
3. Use the helper programs defined in `src/globals.do` to write cleaned LaTeXtables.