Reproducible Workflows in Stata

Osaretin Olurotimi
AIDE Lab / University of Arizona

May 21, 2026

Why Reproducibility Matters

- Research should be transparent, verifiable, and easy to update
- Manual workflows create avoidable errors and wasted effort
- Reproducibility matters for dissertations, coauthorship, and publication
- The goal: move from ad hoc work to scripted, auditable analysis

Learning Goals

- Understand what a reproducible workflow looks like in Stata
- Organize projects clearly and consistently
- Use do-files for cleaning, analysis, and output generation
- Build habits that scale to collaboration and publication

What Is a Reproducible Workflow?

- Same code + same data = same results, every time
- Every step is documented in scripts
- Raw data are preserved and never overwritten
- Outputs can be regenerated automatically from code
- The workflow is understandable to your future self and collaborators

Common Problems in Empirical Research

Typical workflow

- Multiple versions of datasets and scripts
- Copy-paste tables into Word or slides
- Hard-coded file paths tied to one machine
- Cleaning decisions live in memory, not code

Reproducible workflow

- One structured project folder
- Outputs exported directly from Stata
- Portable paths and stable folder structure
- Every transformation recorded in do-files

Core Principles

- Keep raw data untouched — treat it as read-only
- Separate raw, cleaned, and analysis-ready data
- Write modular do-files with a single clear purpose each
- Use logs and comments intentionally
- Automate tables, figures, and exports
- Version control your code
- Save research environment

Globals and Locals in Stata

global

- Accessible anywhere in your Stata session
- Referenced with a $ prefix: `$varname`
- Best for project-wide paths and settings
- Persists until you clear it or close Stata

local

- Scoped to the current do-file or program
- Referenced with backtick/quote: `` `varname' ``
- Best for loops, temporary values, and flags
- Disappears when the do-file finishes running

example.do

```
* Globals: defined once, used everywhere
global data "C:/Users/me/project/data"
use "$data/survey.dta", clear

* Locals: scoped to this do-file only
local controls "age income educ"
reg y `controls', robust
```

Naming Conventions for Variables and Files

Variables

- Use lowercase : `hh_income`, `treat_assign`
- Avoid spaces, dots, and special characters
- Prefix by data source or survey round: `b_age`, `e_wage`
- Keep names short but self-explanatory
- Always pair with a `label var` statement

Files and do-files

- Number scripts by execution order: `01_import.do`
- Use descriptive suffixes: `_clean`, `_analysis`
- Match do-file names to the data they produce
- Never use spaces in file or folder names
- Use `v1`, `v2` only for archived snapshots

naming_examples.do

```
* Good variable names with labels
rename hhincome   hh_income
label var hh_income "Monthly household income (USD)"
label var b_age     "Age at baseline (years)"

* Good file naming: 01_import 02_clean 03_analysis
```

Installing and Managing User-Written Packages

- `ssc install` fetches packages from the SSC archive
- `net install` handles packages hosted elsewhere
- Use `which` to check if a package is already installed
- `adoupdate` keeps installed packages current
- Log installs in `project.do` so collaborators can replicate

project.do

```
* Install packages if not already
    present
capture which estout
if _rc ssc install estout, replace

capture which reghdfe
if _rc ssc install reghdfe, replace

capture which ftools
if _rc ssc install ftools, replace

* Update all installed packages
* adoupdate, update
```

A Good Project Structure

- Keep a stable folder layout across all projects
- Separate source files from generated outputs
- Reflect the flow of the project in the directory
- Make it obvious where code, data, logs, and outputs belong

repo structure

```
project/
  README.md
  project.do
  cleaning/
  analysis/
  data/
    logs/
    raw/
    refined/
    analysis/
      figures/
      tables/
```

An Example Workflow in Stata

1. Import raw data
2. Clean and document all transformations
3. Save cleaned dataset to the refined folder
4. Run analysis scripts
5. Export tables and figures
6. Rebuild your paper or slides with updated results

Project Orchestration

- One top-level do-file runs the whole project
- Standardizes paths and installs packages if needed
- Calls downstream scripts in order
- This is the backbone of a reproducible Stata workflow

project.do

```
global stataVersion 18.0
version $stataVersion

if `"`c(username)'"' == "yourname" {
    global code "C:/git/proj"
    global data "C:/OneDrive/proj/data
        "
    global out  "C:/OneDrive/proj/
        output"
}

do "$code/cleaning/01_import.do"
do "$code/cleaning/02_clean.do"
do "$code/analysis/03_main_results.do"
do "$code/analysis/04_tables_figures.
```

Managing File Paths

- Avoid hard-coded paths throughout your scripts
- Define all paths in one top-level setup block
- Collaborators only need to edit that one block
- Use globals so every script stays portable

project.do

```
if `"`c(username)'"' == "yourname" {
    global code  "C:/Users/yourname/
        git/proj"
    global raw   "C:/Users/yourname/
        data/raw"
    global clean "C:/Users/yourname/
        data/refined"
    global out   "C:/Users/yourname/
        data/analysis"
}

use "$raw/survey.dta", clear
save "$clean/survey_clean.dta",
    replace
```

Writing Better Do-Files

- Start each file with a clear purpose header
- Use meaningful names and consistent formatting
- Comment decisions that are not obvious from the code
- Make scripts readable for your future self

02_clean.do

```
* 02_clean.do
* Purpose: clean household survey
* Inputs : raw survey data
* Output : analysis-ready dataset

use "$raw/survey.dta", clear
//Drop variables missing ID
drop if missing(id)
label var income "Monthly income (USD)
    "
rename hhsize household_size
```

Data Cleaning as Documentation

- Every transformation should be visible in code
- Label variables and values clearly
- Record sample restrictions and dropped observations
- Save clean, analysis-ready data intentionally

02_clean.do

```
keep if inrange(age, 18, 65)
di "Obs remaining: " _N

* Winsorize income at 99th percentile
egen p99 = pctile(income), p(99)
replace income = p99 if income > p99
drop p99

gen poor = (income < 2.15 * 30)
label var poor "Below $2.15/day PPP"
```

Logs and Output Tracking

- Use logs to keep a record of analysis runs
- Save model output systematically with `eststo`
- Make the workflow auditable from start to finish
- Logs capture warnings, dropped obs, and messages

03_main_results.do

```
local logfile "$data/logs/03_main.log"
log using `logfile', replace text

eststo clear
eststo m1: reg y x1 x2, robust
eststo m2: reg y x1 x2 x3, robust

log close
```

Automating Tables and Figures

- Export outputs directly from Stata — no copy-paste
- Update tables and figures by rerunning the code
- Keep papers, appendices, and slides in sync
- `esttab` and `graph export` are your best friends

04_tables_figures.do

```
esttab m1 m2 using "$out/tables/t1.tex
    ", ///
    replace booktabs label se ///
    star(* 0.10 ** 0.05 *** 0.01)

twoway (scatter y x1), ///
    title("Income vs. Consumption") //
        /
    scheme(s2color)
graph export "$out/figures/fig1.pdf",
    replace
```

Version Control and Collaboration

Without version control

- Files passed around by email
- Hard to know which script is current
- Easy to lose earlier work
- Collaboration becomes slow and messy

With Git

- Code history is tracked clearly
- Changes can be reviewed and reversed
- Coauthors can work in parallel branches
- Projects are easier to maintain over time

A Minimal GitHub Workflow

- Create a repo on GitHub and clone it once
- Pull before you start work each session
- Stage and commit do-files after meaningful changes
- Push to GitHub so collaborators stay in sync
- Never commit `.dta` or raw data — code only
- Use `.gitignore` to exclude data, logs, and outputs

terminal

```
# one-time setup
git clone https://github.com/user/proj
    .git
cd proj

# daily workflow
git pull origin main
git add cleaning/*.do analysis/*.do
git commit -m "clean: restrict sample
    ages 18-65"
git push origin main
```

.gitignore

Reproducibility for Dissertation Work

- Easier to revisit a chapter after months away
- Cleaner handoff to advisors and coauthors
- Faster responses to reviewer comments — just rerun
- Better long-term research habits across all projects

Common Mistakes to Avoid

Avoid

- Editing datasets manually in Excel
- Overwriting raw data
- Keeping many vaguely named script versions
- Relying on memory instead of comments

Do instead

- Put every change in code
- Preserve raw inputs permanently
- Use clear names and a stable folder structure
- Test your project from a clean restart

Practical Recommendations

- Start simple: one project structure and one top-level do-file
- Automate repetitive outputs first — tables and figures
- Document decisions as you go, not months later
- Treat reproducibility as part of research design
- Build a workflow that someone else could rerun from scratch

Takeaway

Reproducibility saves time and reduces mistakes

Stata supports fully script-based research workflows

Small workflow improvements compound into large long-term benefits

Quick Reference: Key Commands

Setup & paths

- `global`, `local` — define macros
- `version` — pin Stata version
- `do` — run a do-file

Packages

- `ssc install` *pkg* — install from SSC
- `which` *pkg* — check if installed
- `adoupdate` — update all packages

Logging & output

- `log using` *file*`, replace text`
- `eststo` / `esttab` — model tables
- `graph export` — save figures

Good habits

- `assert` — validate assumptions in code
- `capture` — suppress errors gracefully
- `di "Obs: " _N` — document sample size

Acknowledgments

Thank you to our sponsors and partners

BITSS

Berkeley Initiative for Transparency in the Social Sciences
CEGA at UC Berkeley

For advancing transparency, reproducibility, and open research practice.

Thank you

Questions?

AIDE Lab · University of Arizona