Finance PhD Student Tutorial: Cleaning Data & Efficient Workflows

Kyle Zimmerschied[1]

[1]University of Arkansas

April 29, 2026

Today's Agenda

Focus: How do I create efficient workflows & code?

1. **Opening Discussion and Reflection (10 min):** What does my current coding & workflow process look like?
2. **Methods (40 min):** What is the ideal way to structure project workflows?

Opening Discussion

Reflect on Your Experience

1. What does my current workflow currently look like in terms of:
   - What programming language(s) I use?
   - How organized is my coding and file layout?
   - How long would it take me to produce the finalized data I use for analysis?

Discussion time: 10 minutes

What Bad Workflow Looks Like

Warning: You may recognize yourself here.

Chaotic File Structure

```
Immigration_Project/
  data_final.dta
  data_final_v2.dta
  data_final_v2_fixed.dta
  regression_new.do
  regression_new_try2.do
  regression_new_try2_final.do
  regression_FINAL_USE_THIS.do
  random_notes.txt
  temp_file.dta
```

Messy Code Practices

- Hard-coded file paths:
  - `cd "/Users/kyle/Desktop/project"`
- Cleaning, merging, regressions, and figures all in one file
- No comments
- Overwriting datasets repeatedly
- Unsure which dataset produced final table

Litmus Test: If your laptop crashed today, could you rebuild everything from raw data?

Efficient Workflows and Code Overview

- Clean, reproducible code and efficient workflows are often overlooked
- We focus on results — but workflow determines whether those results are credible
- Issues: Disorganized code and sloppy workflows lead to:
  1. Increased likelihood of data cleaning errors
  2. Difficulties collaborating or sharing code with co-author
  3. Significant issues extending or replicating empirical work

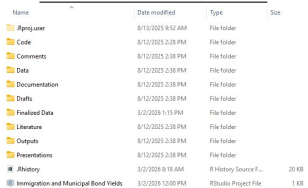
Choice: You either pay the time cost upfront — or you pay it later, repeatedly

Today's class: Walk through best practices for project organization, coding, and data cleaning

Best Practices File Organization

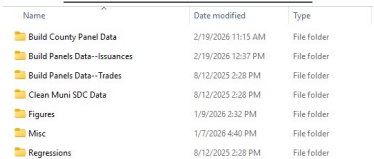
- Create consistent set of folders to store:
  1. Data: raw data, intermediate data, and finalized data
  2. Code: data cleaning, figures, and tables
  3. Outputs: tables and figures
  4. Project-related materials: related literature, paper or presentation drafts, comments

File Organization



Name	Date modified	Type	Size
.Rproj.user	8/13/2025 9:52 AM	File folder	
Code	8/12/2025 2:28 PM	File folder	
Comments	8/12/2025 2:38 PM	File folder	
Data	8/12/2025 2:38 PM	File folder	
Documentation	8/12/2025 2:38 PM	File folder	
Drafts	8/12/2025 2:38 PM	File folder	
Finalized Data	3/2/2026 1:15 PM	File folder	
Literature	8/12/2025 2:38 PM	File folder	
Outputs	8/12/2025 2:38 PM	File folder	
Presentations	8/12/2025 2:38 PM	File folder	
.Rhistory	3/2/2026 8:18 AM	R History Source F...	20 KB
Immigration and Municipal Bond Yields	3/2/2026 12:00 PM	RStudio Project File	1 KB

Code Organization



Name	Date modified	Type
Build County Panel Data	2/19/2026 11:15 AM	File folder
Build Panels Data--Issuances	2/19/2026 12:37 PM	File folder
Build Panels Data--Trades	8/12/2025 2:28 PM	File folder
Clean Muni SDC Data	8/12/2025 2:28 PM	File folder
Figures	1/9/2026 2:32 PM	File folder
Misc	1/7/2026 4:40 PM	File folder
Regressions	8/12/2025 2:28 PM	File folder

Data Organization

Name	Date modified	Type
Inputs	8/12/2025 2:37 PM	File folder
Outputs	8/12/2025 2:38 PM	File folder
SDC Query	8/12/2025 2:38 PM	File folder

Collaborating with Co-Authors

- Collaborating with co-authors requires:
  1. File sharing
  2. Directory calls to reference the proper directory for each user

1. File sharing: best done by creating shared OneDrive or Box folder for all project-related files to be stored across users
2. Directory sharing:
   - R projects reference the same relative directory path for each user
   - Stata allows for specification of directories based on local user name

```
* Detect current user
local usr = c(username)

* Set project root depending on user
if "`usr'" == "kylez" {
    global projroot "/Users/kylez/Dropbox/Immigration_Project"
}
else if "`usr'" == "matteo" {
    global projroot "/Users/matteo/Dropbox/Immigration_Project"
}
else {
    display as error "Unknown user -- set project root manually."
}
```

Overview: Coding Best Practices

- It's easy to get lost in the flow of files when cleaning data or reproducing tables and figures
- Solution: Have a master R or Stata file which calls other sub-files required for data cleaning or production of outputs

SOURCE_CODE

```
clear all
set more off
*==============================================================================
* Initialize Root Path
*==============================================================================
global projectroot "D:\Immigration and Municipal Bond Yields NTA"
cd "$projectroot"

*==============================================================================
* Initialize Output Paths
*==============================================================================
global main_output_path "$projectroot/Outputs/Tables/Burchardi Lag Prop Immigration/Main/"
global appendix_output_path "$projectroot/Outputs/Tables/Burchardi Lag Prop Immigration/Appendix/"
*==============================================================================
*Set the main endogenous variable and instrument
*==============================================================================
*Choose your endogenous variable
global endog_x new_prop_immigration

*Set your instrument
global instrument immigration_shock

di ">>> Starting Master Script: " c(current_time) ", " c(current_date)
set rmsg on
*==============================================================================
*LOAD IN AND CLEAN DATA
*==============================================================================
do "$projectroot\Code\Stata\Issuance Level Analysis.do"
*==============================================================================


*==============================================================================
*LOAD IN DATA AND SET GLOBALS
*==============================================================================
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\PRELIM--LOAD in Data and Set Global.do"
*==============================================================================
* Main Tables
*==============================================================================
    *qui do "Code\Stata\Tables\TABLE_SUMMARY.do" \"Summary TABLE 1 "\
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 5--Secure Communities Act.do"
keep if in_sample == 1

do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 1--SUMMARY.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 2--Main OLS Effect on Yields.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 3--Main IV Effect on Yields.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 4A--Conditional IV Effect on Undocumented Immigrant Yields.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 4B--Effect of Legal vs Illegal Immigration.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 6--County Labor Market Effects.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 7--County Operating Margin and Balance Sheet.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 8--County Revenue.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 9--County Expenses.do"
do "$projectroot\Code\Stata\Tables\Augmented Burchardi\Table 10--Immigrant Heterogeneity.do"
```

Coding Files Overview

- When structuring code, it's best to think of the unit of analysis
- For example, in my project, the unit of analysis is at the bond issuance level
  - Intermediate data files will be merged into this
- It can be helpful to organize coding files that produce intermediate files or outputs in separate folders
  - Folders contain separate sub-files that clean and process the raw data into intermediate data

```
#=============================================================================
#MERGE TOGETHER YOUR DATA
#=============================================================================
IssuanceData = IssuanceData %>%

  #Join in HASSAN DATA
  left_join(HassanData,by = c("state_county_fips" = "county_code","issue_year" = "year")) %>%

  #JOIN IN HASSAN DATA FILLED
  #left_join(HassanDataFilled,by = c("state_county_fips" = "county_code_f","issue_year" = "year_f")) %>%

  #Join in your county Data
  left_join(CountyData,by = c("state_county_fips","issue_year" = "year")) %>%

  #Join in your state panel
  left_join(StatePanel,by = c("state_of_issuer" = "state_name","issue_year" = "year")) %>%

  left_join(SanctuaryPolicy,by = c("state_county_fips","issue_year" = "year")) %>%

  #Bus DID
  left_join(BusDiD,by = c("state_county_fips")) %>%

  #Mutate a post variable
  mutate(post_bus = case_when(issue_dated_date >= date_bus_start ~ 1,
                              issue_dated_date < date_bus_start ~ 0,
                              TRUE ~ 0)) %>%
```

Keys to Coding

1. **Comments and white space are your friends**
   - Create commented overview at top of file that walks through the purpose of your code
   - Comment through each process and step to benefit your current and future self
   - Include whitespace and strong syntax for readability
2. **Verify intermediate outputs**
   - Assure that variable creation or functions produce intended output
   - Verify that observations are not unintentionally added or dropped in merges
3. **Improve coding syntax**
   - Investing upfront to write clean, reproducible code provides large benefits
   - Be as succinct as possible
   - AI tools can be helpful to pass along wrong draft of code, but make sure to verify output
4. **One file–One purpose**
   - Limit code to accomplish one purpose (e.g., clean one data source, produce one figure) rather than cleaning them all together

Everyone makes mistakes, but the key is to review frequently to catch them at an early stage

Data Cleaning Best Practices

1. **Never overwrite raw data**
   - Raw data is sacred — treat it as read-only
   - Save all cleaned versions to a separate `/data/intermediate/` folder
2. **Audit every merge**
   - Always check match rates: how many obs were matched, unmatched, duplicated?
   - In Stata: use `merge` with `assert()` or inspect `_merge` carefully
   - In R: use `dplyr::left_join()` and verify row counts before and after
3. **Handle missings intentionally**
   - Document *why* observations are missing — attrition vs. true missing vs. structural zero
   - Never drop observations silently
4. **Validate your variables**
   - Check ranges, distributions, and unit consistency after creating new variables
   - Cross-check against published statistics or summary stats from prior literature

Summary: The Reproducibility Standard

A project is well structured if:

- All results can be rebuilt from raw data
- No dataset is manually edited
- One master file recreates everything
- No file path is hard-coded
- No ambiguity about which file produces which output

If you cannot press “Run” and rebuild everything, the workflow is incomplete.

Before Next Week (1 Hour)

- Explore (30 minutes):
  1. Tables in Stata
- Reflect (30 minutes):
  1. How can my current workflow be improved?
  2. What areas of data processing and project organization am I strongest/weakest in?

Extra: Reflect from Last Week

- How compelling is my current research design?
- What assumptions am I relying upon and what arguments are they sufficient or insufficient to address?
- Can I provide a compelling design to answer my research question of interest?