Fundamentals of Scientific Computing with Stata

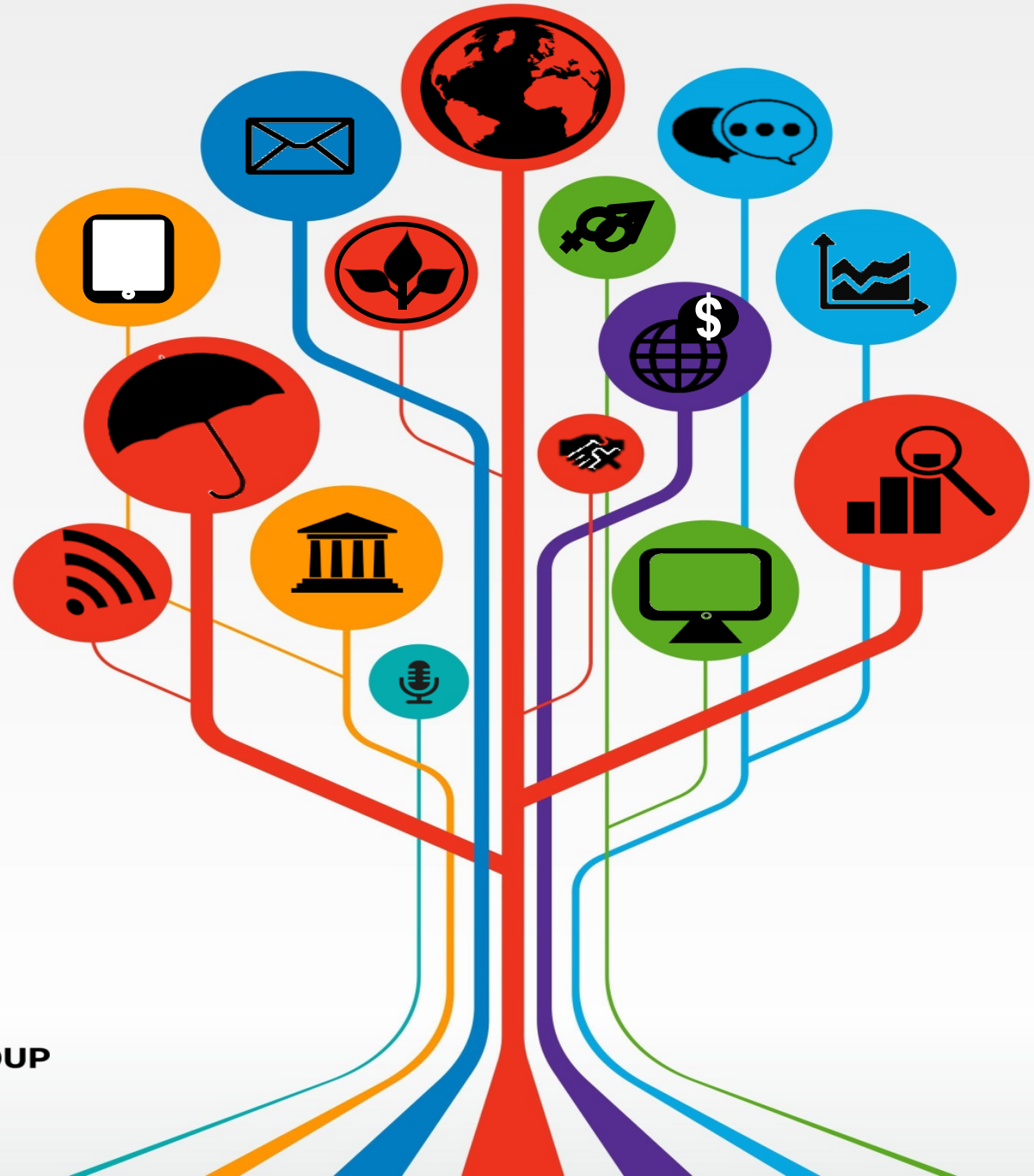
Research Assistant Onboarding

Prepared by **DIME Analytics**

dimeanalytics@worldbank.org

Presented by **Benjamin Daniels**

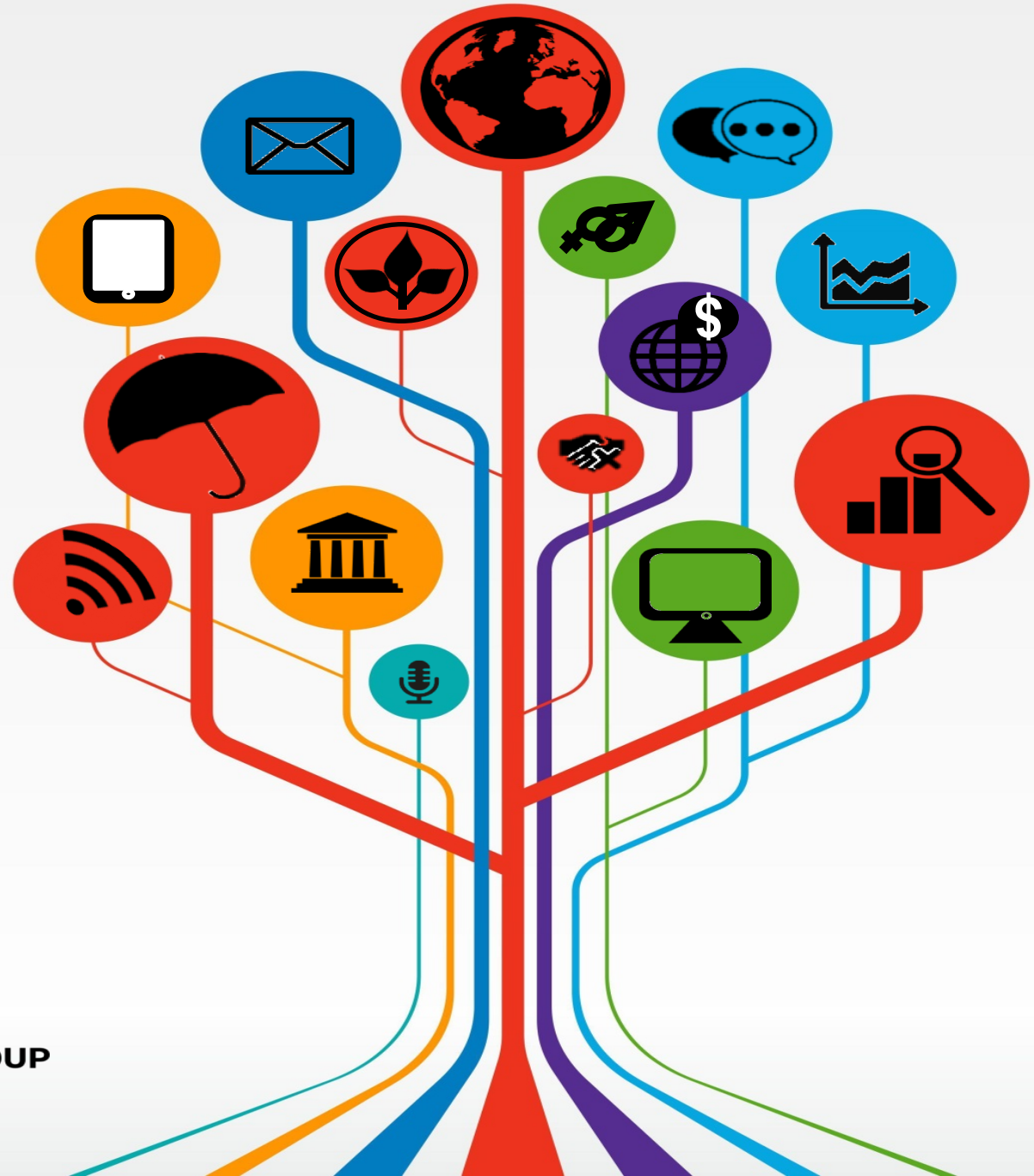
bdaniels@worldbank.org



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Keeping Stuff Organized



i2i DIME
TRANSFORM DEVELOPMENT

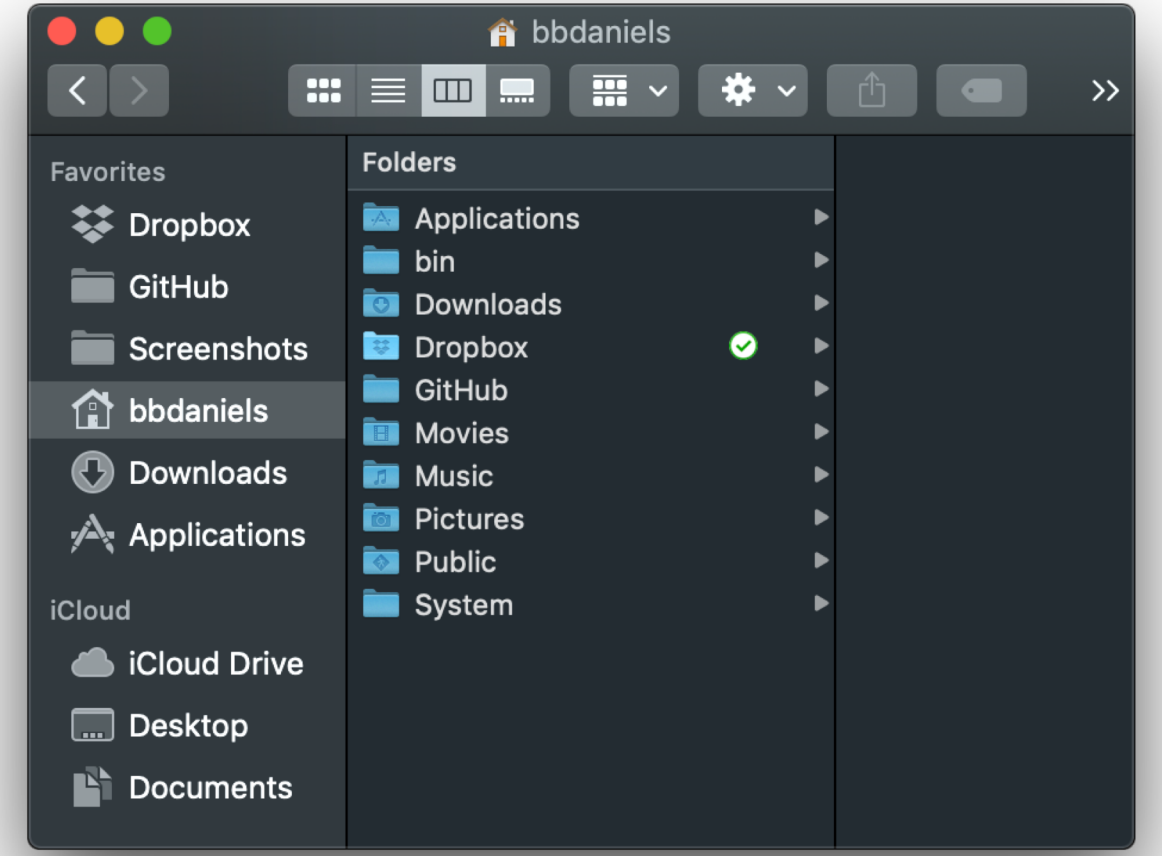
WORLD BANK GROUP

Folder Structure

- Basic location: “Home” folder (~)
- Everything lives here

- NOT your desktop
- NOT a synced folder (including iCloud and Dropbox)

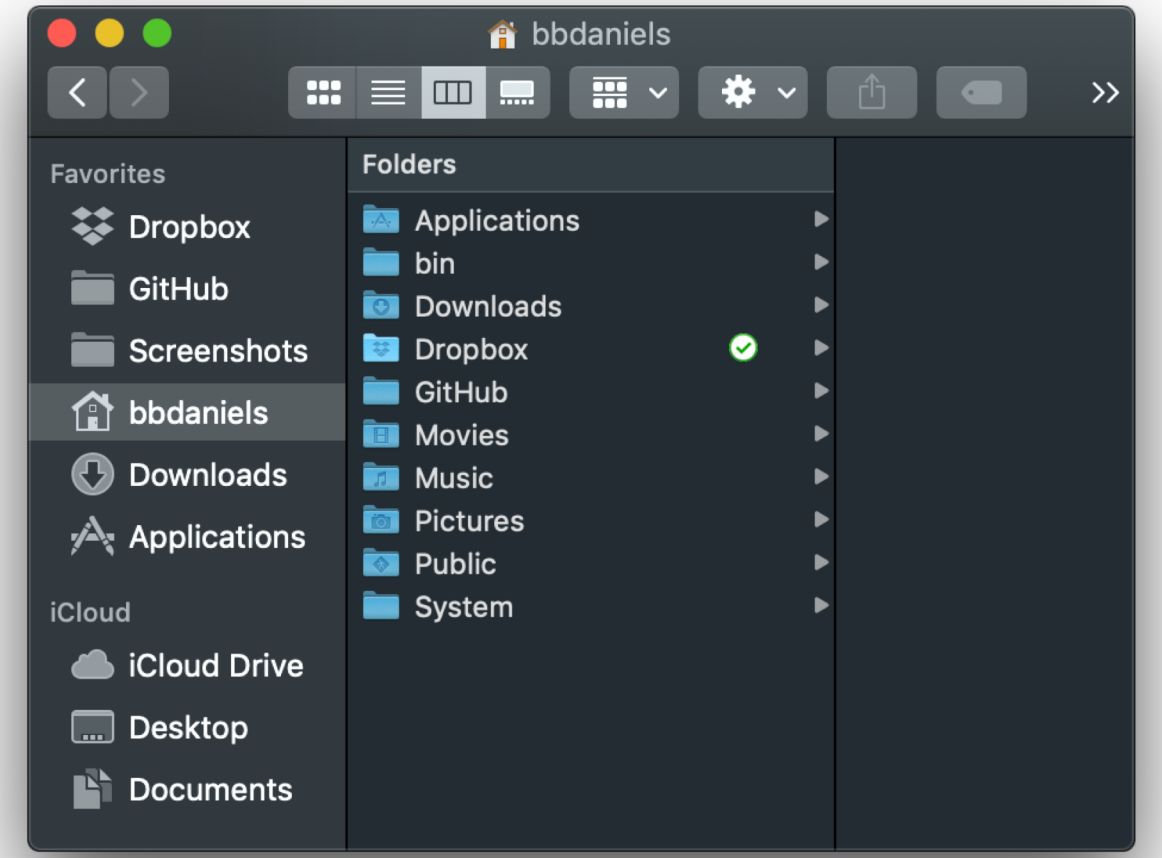
- Many people don’t have one!
- Everyone here must – it will standardize most of your ${directory} globals in Stata



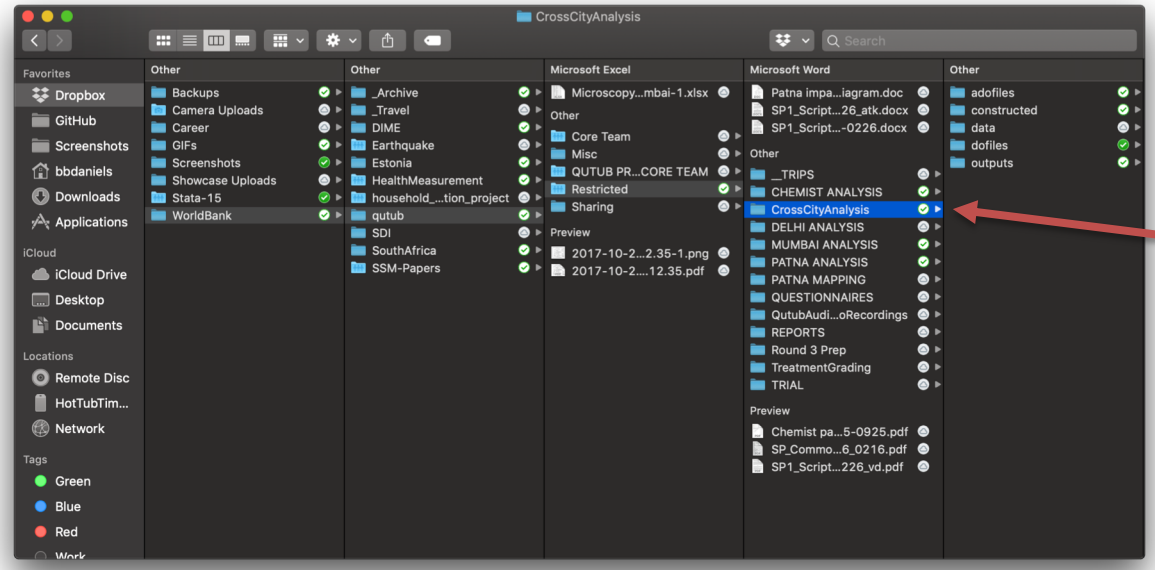
Folder Structure

Your work and tools live in:

- GitHub and Dropbox
- Program files
- Downloads
- Screenshots (in Dropbox)

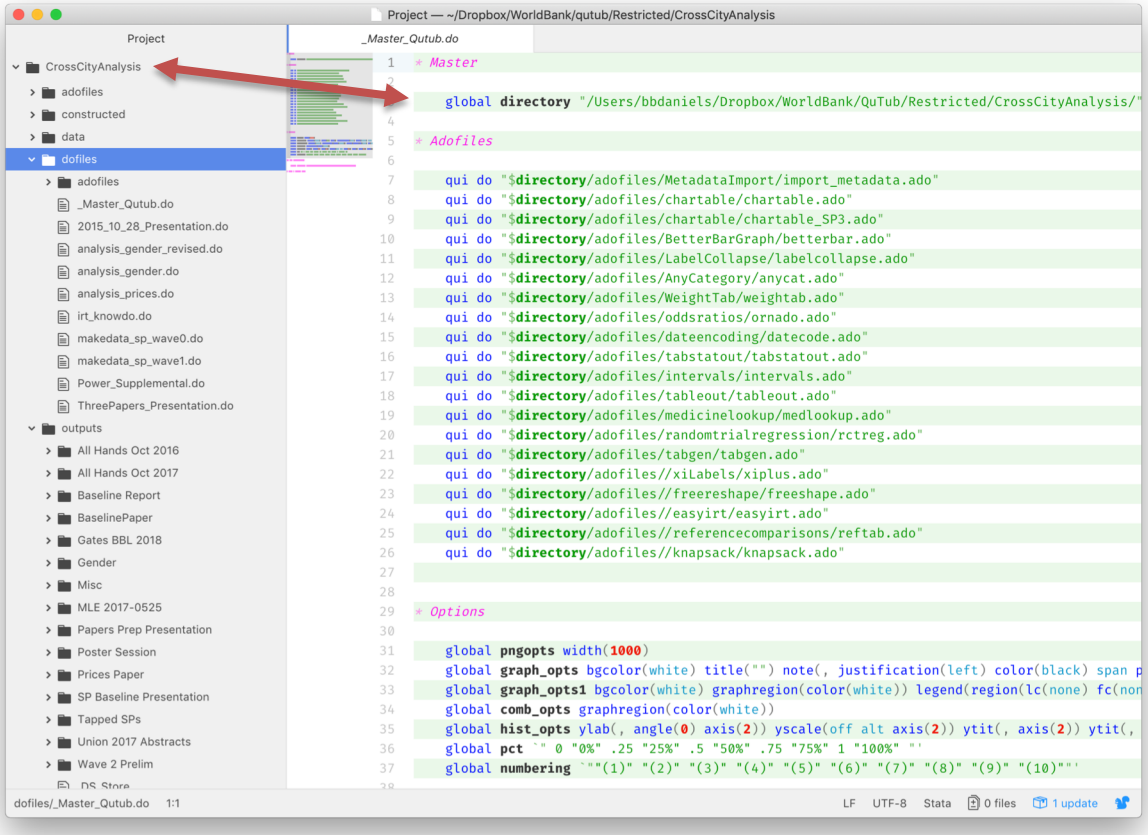


Make file structure part of your workflow



Use a file browser and layout that helps you remember and interact with the file structure…

…and a code editor that zooms in on the portion of the file structure you need to work with.



File and folder names must be structured

Three principles for names:

1. **Machine Readable**
2. **Human Readable**
3. **Orders Well**

NO

- myabstract.docx
- Joe's Filenames Use Spaces and Punctuation.xlsx
- figure 1.png
- fig 2.png

YES

- 2014-06-08_abstract-for-sla.docx
- joes-filenames-are-getting-better.xlsx
- fig01_scatterplot-talk-length-vs-interest.png
- fig02_histogram-talk-attendance.png
- 1986-01-28_raw-data-from-challenger-o-rings.txt

("naming things" by Jenny Bryan)

Follow a few simple rules in naming schemes

Rules for files & folders:

- **Underscore _ separates levels of information (order, purpose, date)**
- **En-dash - separates words**
- **Numbers are left-padded (01,02,03…)**

- **NO SPACES**
- **NO SPECIAL CHARACTERS**
- **NO CAPITAL LETTERS**

("naming things" by Jenny Bryan)

This list orders correctly and you can tell what everything is:

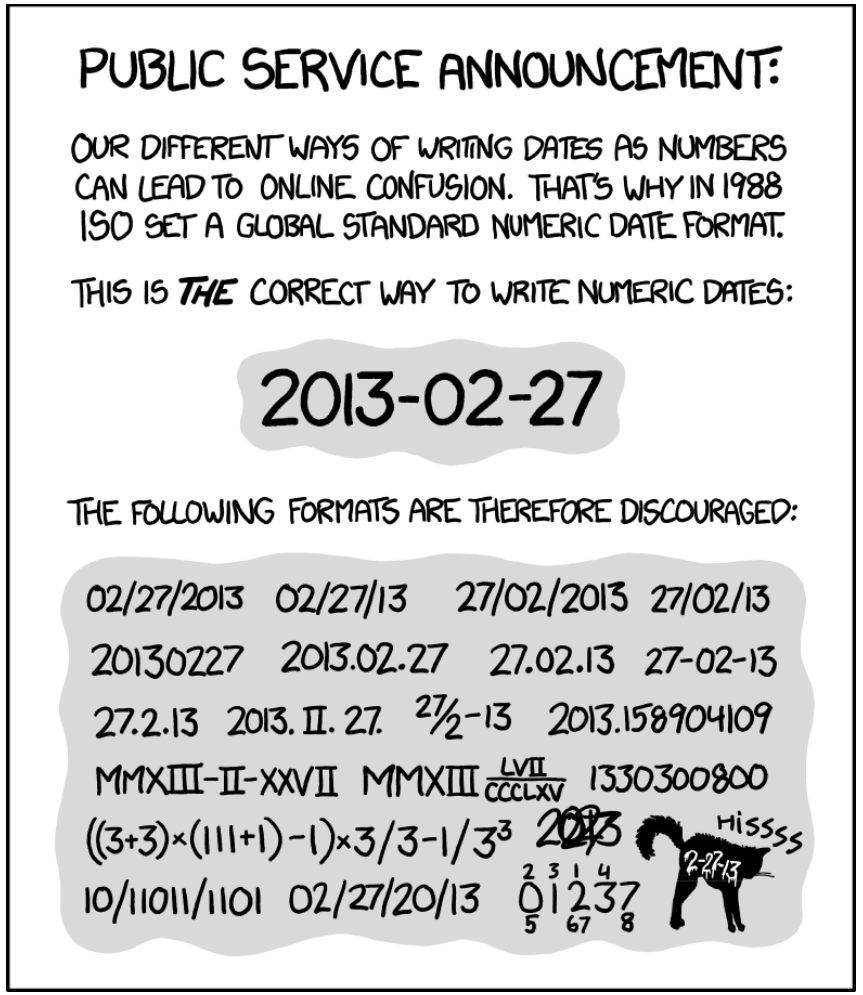
- 01_marshal-data.r
- 02_pre-dea-filtering.r
- 03_dea-with-limma-voom.r
- 04_explore-dea-results.r
- 90_limma-model-term-name-fiasco.r
- helper01_load-counts.r
- helper02_load-exp-des.r
- helper03_load-focus-statinf.r
- helper04_extract-and-tidy.r

Dates: **YYYY-MM-DD**

Global ISO 8601 standard:

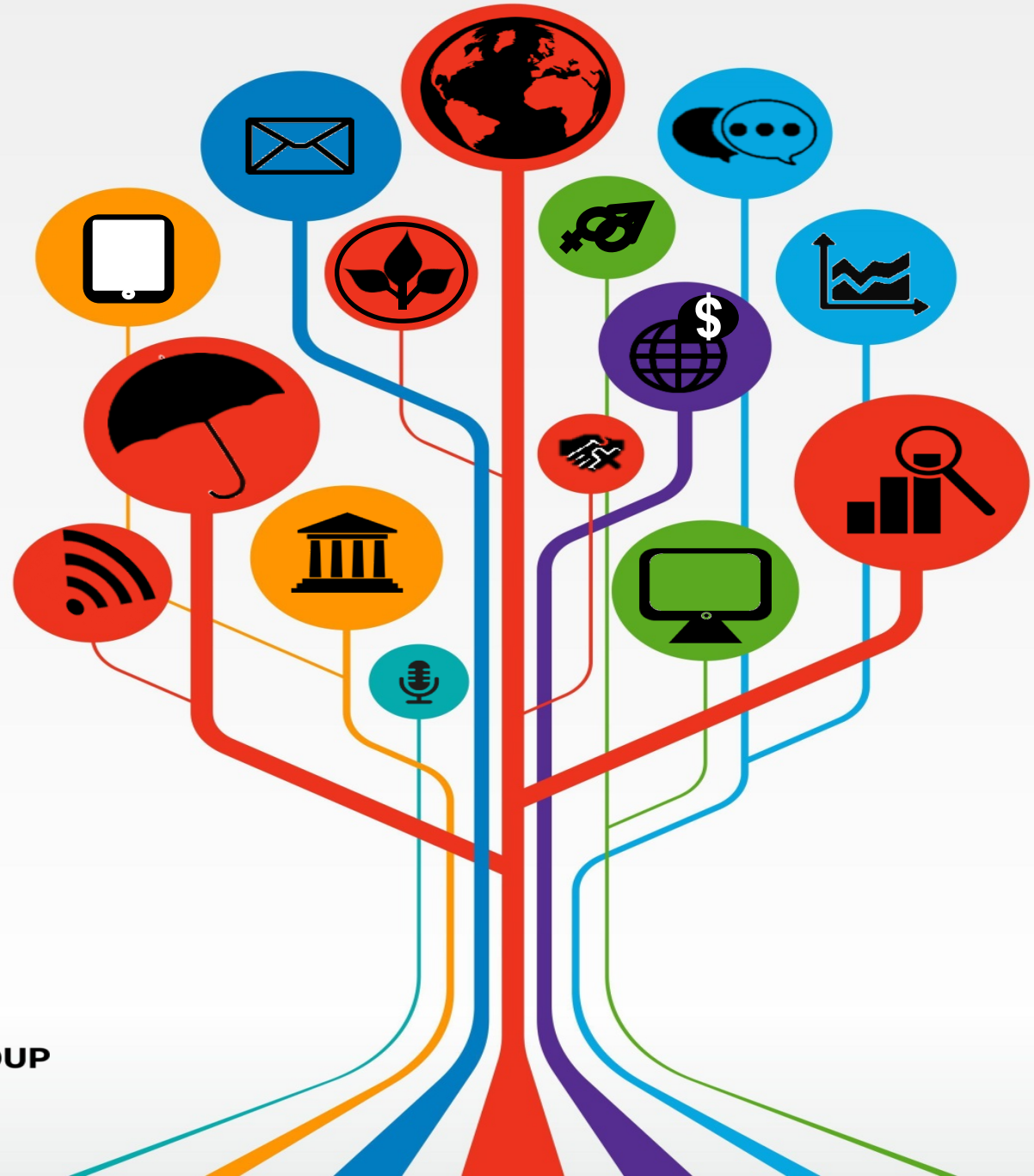
YYYY-MM-DD

If you do not do this, your files *will* be out of order.



https://xkcd.com/1179/

Keeping Code Organized



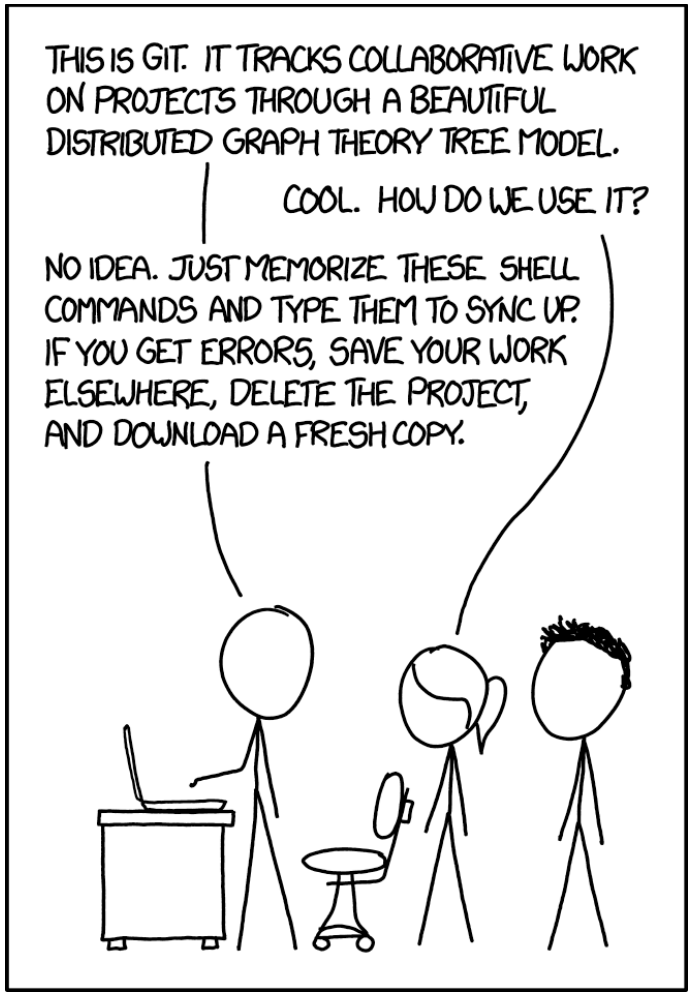
i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Top 8 Best Practices for Scientific Computing

1. **Write programs for people, not computers.**
2. **Let the computer do the work.**
3. **Make incremental changes.**
4. **Don't repeat yourself (or others).**
5. **Plan for mistakes.**
6. **Optimize software only after it works correctly.**
7. **Document design and purpose, not mechanics.**
8. **Collaborate.**

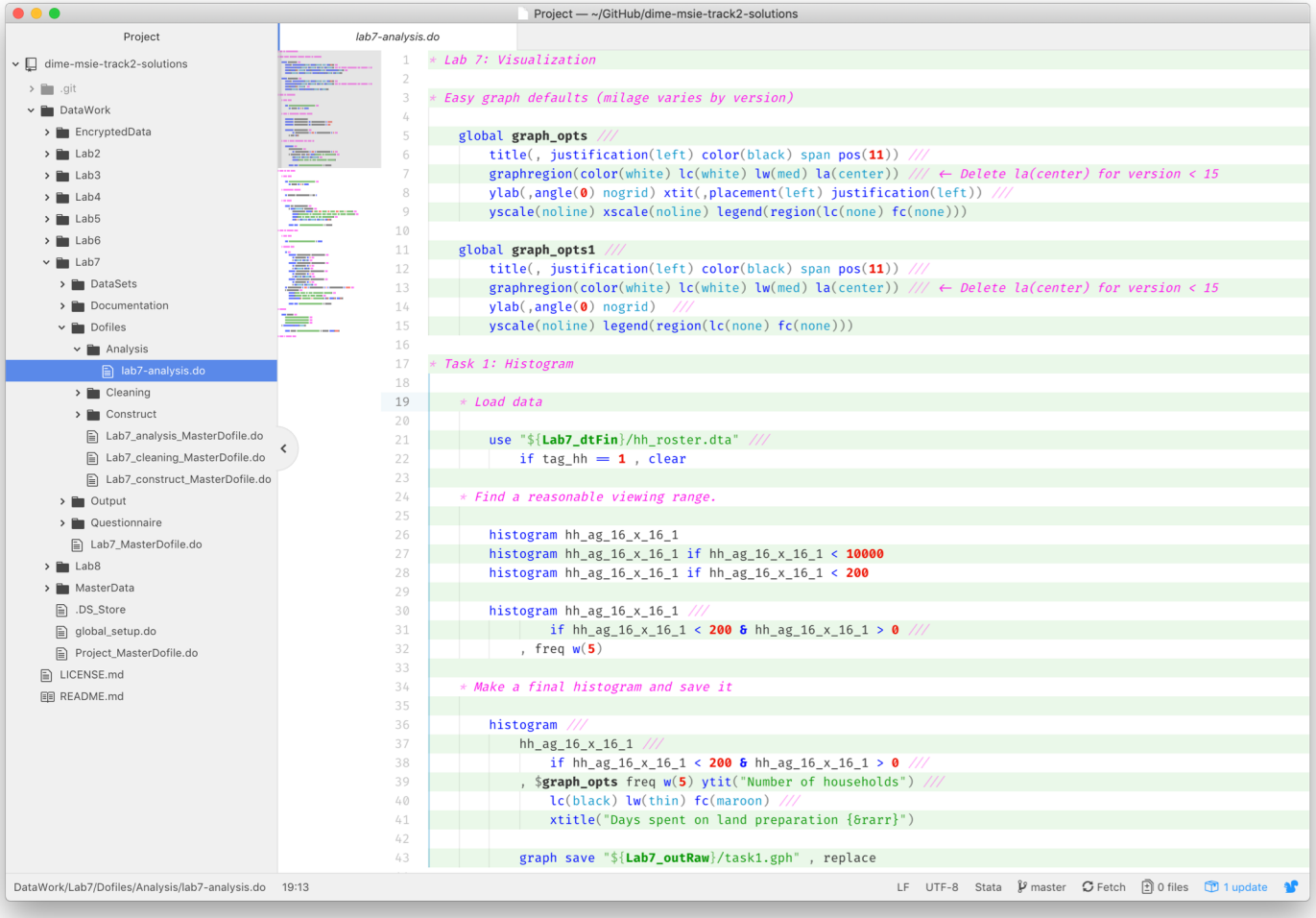
Wilson et al., "Best Practices for Scientific Computing".



https://xkcd.com/1597/

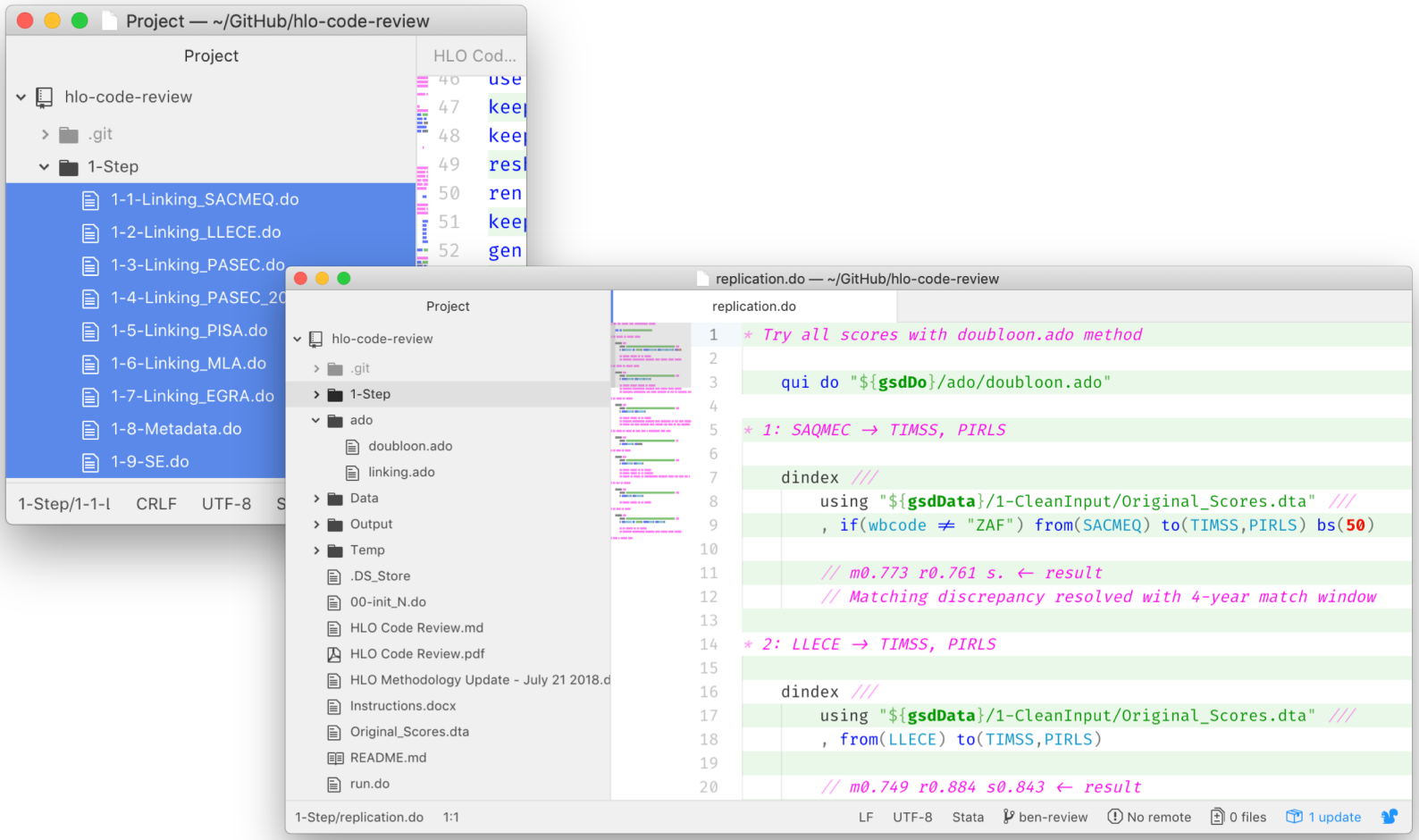
Write programs for people, not computers.

- Very short code “chunks”
- Clear purpose for each
- Many comments (easier if your editor allows pretty colors for comments)
- Logical indentation – shows “belonging” of every line of code



Let the computer do the work.

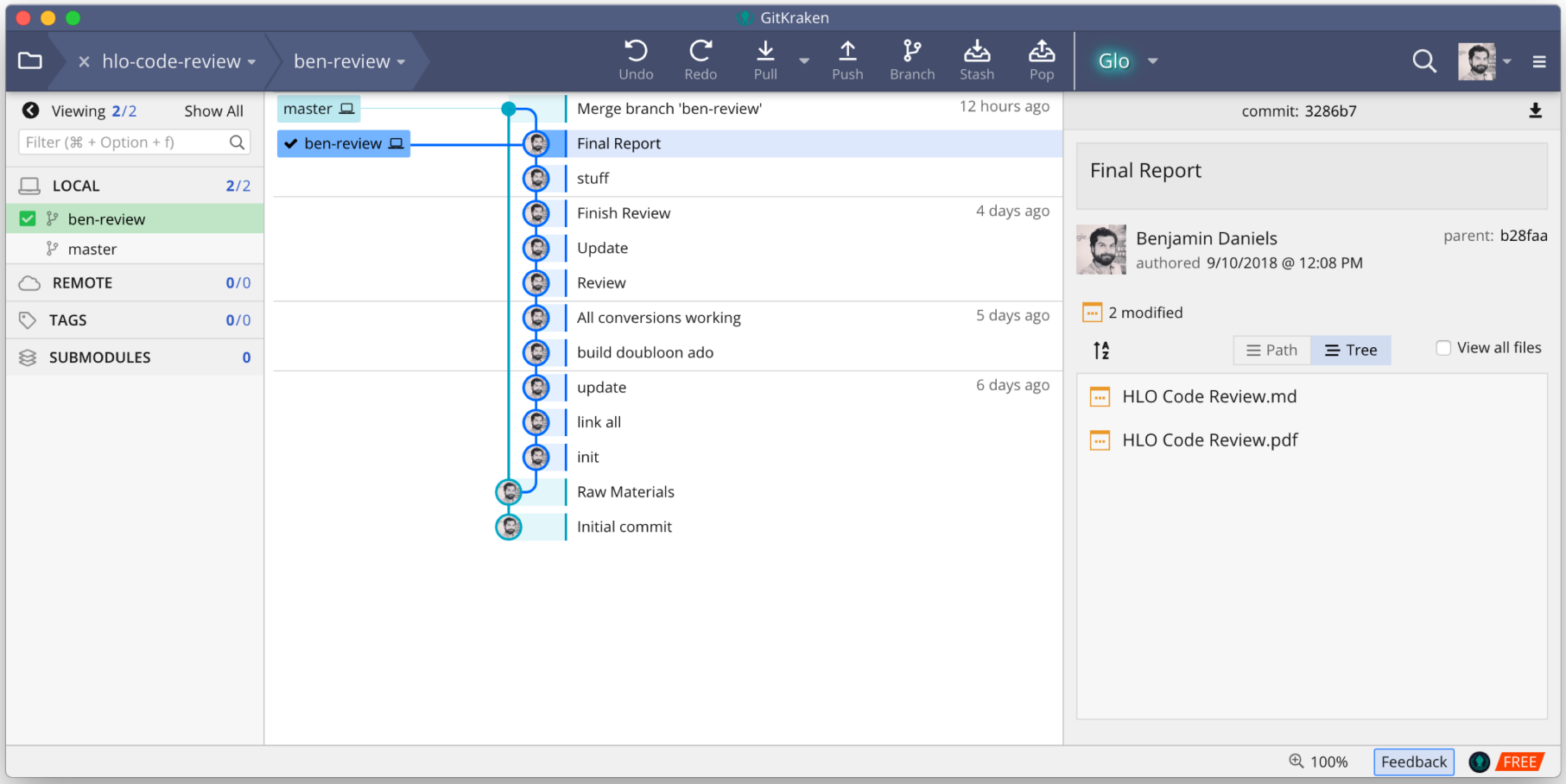
- You shouldn't be copy-pasting the same things
- Computers are for boring and repetitive tasks
- You are for figuring out how to make the computer do it!



Make incremental changes.

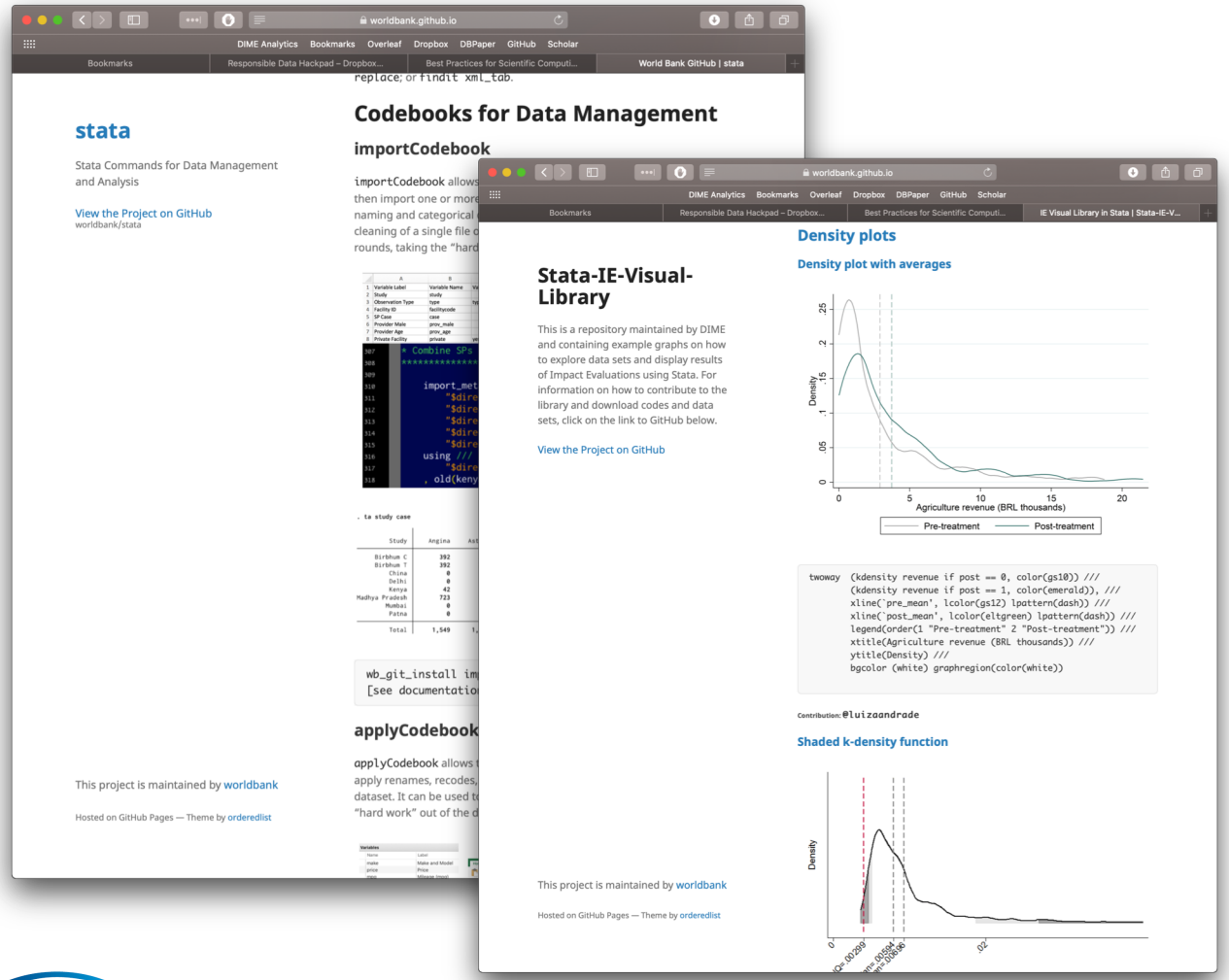
Avoid “task entanglement”

- One task at a time
- Easy method for switching between tasks



Don't repeat yourself (or others).

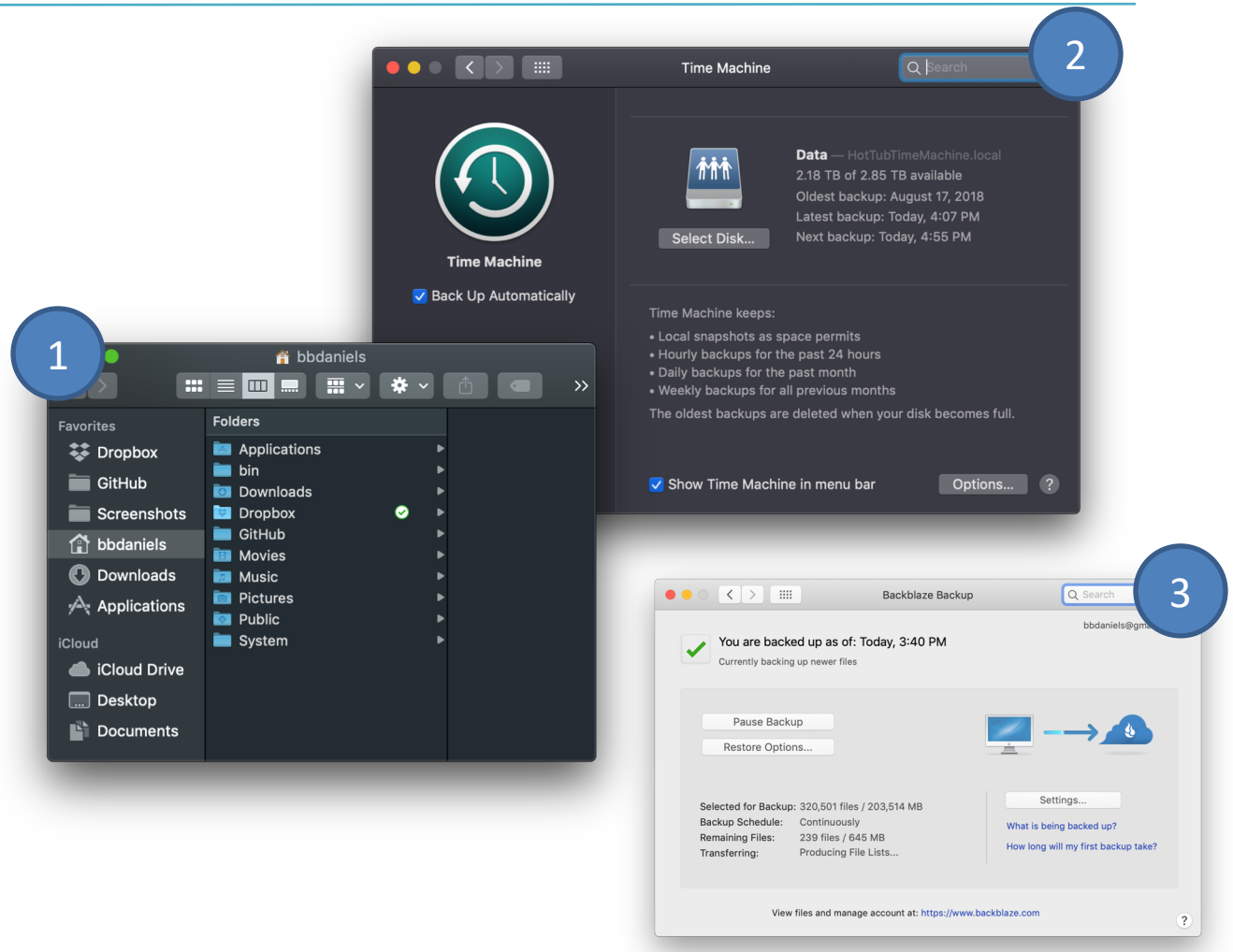
- Maintain a “code library” of your own good work and visit those of others
- Use DIME Wiki
- Write or borrow ado-files for common tasks
- Clean data *before* analysis files begin



Plan for mistakes.

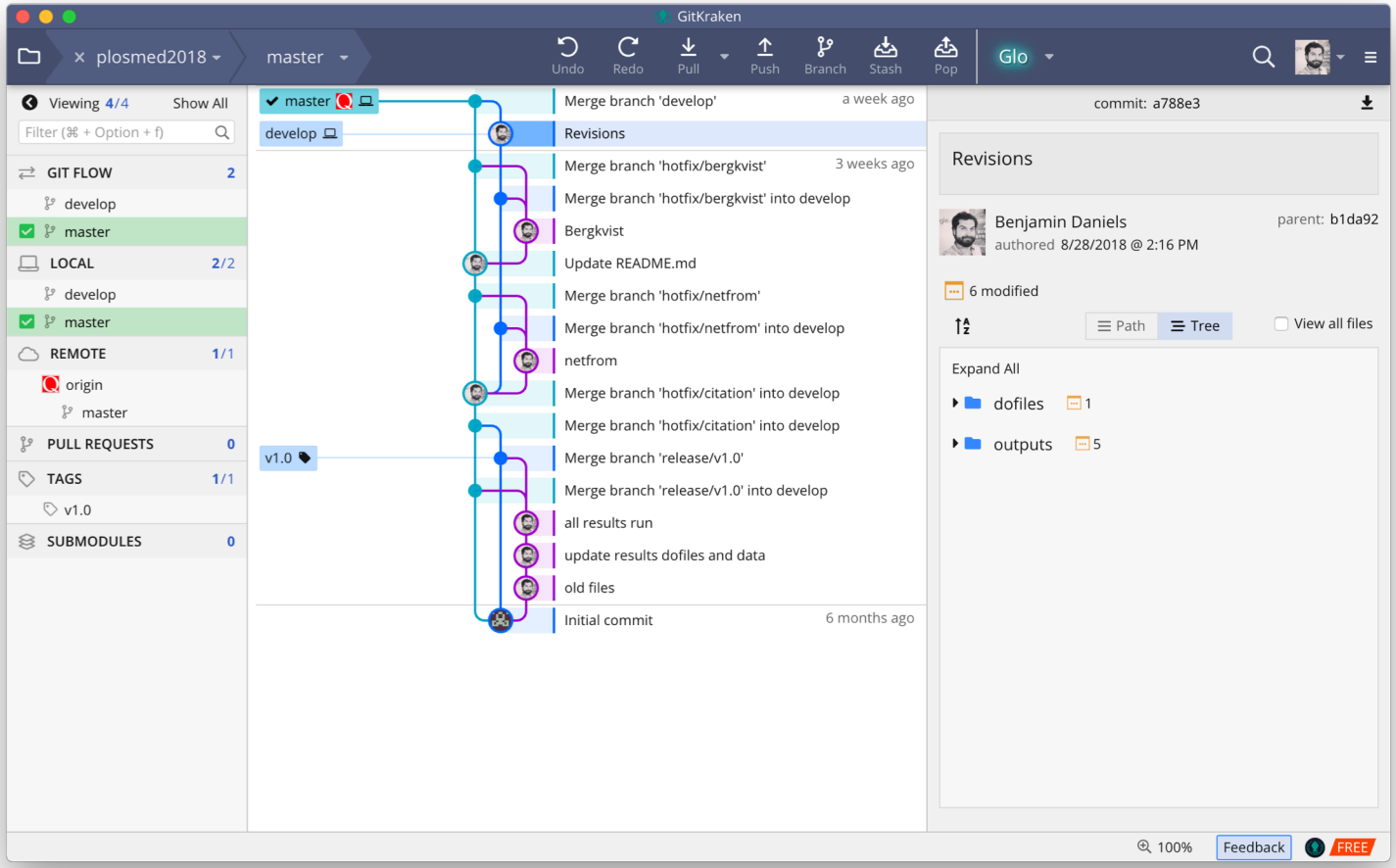
3-2-1-0 Backup Strategy:

- 3 total copies of your data
- 2 on different devices you own
- 1 copy off-site
- 0 Dropbox does not count



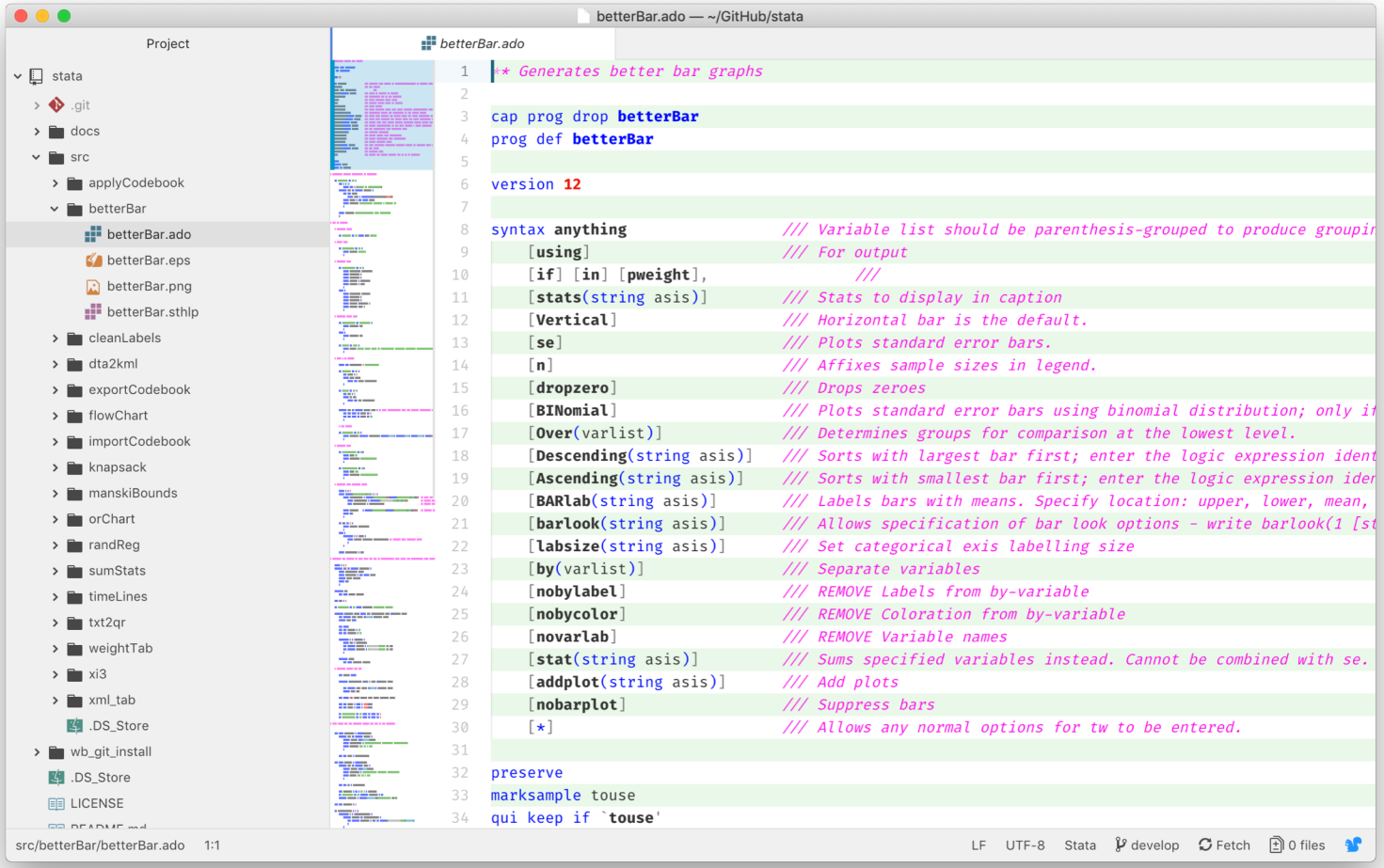
Optimize only when it works.

- First drafts are are rarely perfect
- _FINAL drafts *never* are
- So don’t re-generate stuff by hand all the time
- Nothing needs to look nice until you are ready to go to press
- Raw outputs need not be pretty
- At press time you can clean everything up and delete cruft



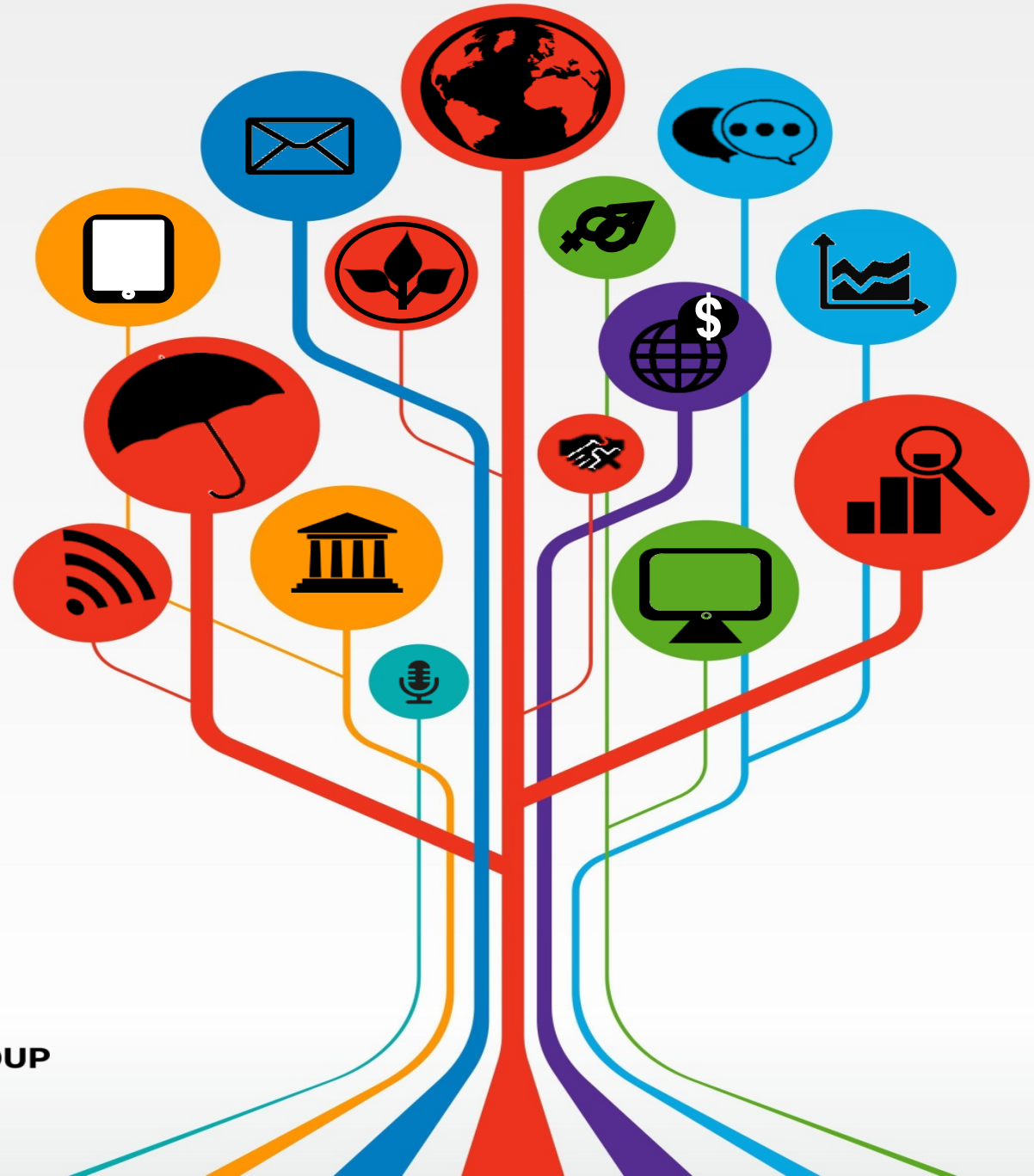
Document design and purpose, not mechanics.

- Don’t make us read the code!




```
** Generates better bar graphs

cap prog drop betterBar
prog def betterBar

version 12

syntax anything                 /// Variable list should be parenthesis-grouped to produce groupi
    [using]                     /// For output
    [if] [in] [pweight],                ///
    [stats(string asis)]        /// Stats to display in caption
    [Vertical]                  /// Horizontal bar is the default.
    [se]                        /// Plots standard error bars.
    [n]                         /// Affixes sample sizes in legend.
    [dropzero]                  /// Drops zeroes
    [BINomial]                  /// Plots standard error bars using binomial distribution; only i
    [Over(varlist)]             /// Determines groups for comparison at the lowest level.
    [Descending(string asis)]   /// Sorts with largest bar first; enter the logic expression ident
    [Ascending(string asis)]    /// Sorts with smallest bar first; enter the logic expression ide
    [BARlab(string asis)]       /// Labels bars with means. Specify location: upper, lower, mean,
    [barlook(string asis)]      /// Allows specification of bar look options - write barlook(1 [s
    [labsize(string asis)]      /// Set categorical exis labelling size
    [by(varlist)]               /// Separate variables
    [nobylabel]                 /// REMOVE Labels from by-variable
    [nobycolor]                 /// REMOVE Coloration from by-variable
    [novarlab]                  /// REMOVE Variable names
    [stat(string asis)]         /// Sums specified variables instead. Cannot be combined with se.
    [addplot(string asis)]      /// Add plots
    [nobarplot]                 /// Suppress bars
    [*]                         /// Allows any normal options for tw to be entered.

preserve
marksample touse
qui keep if `touse'
```

Fundamentals of Stata

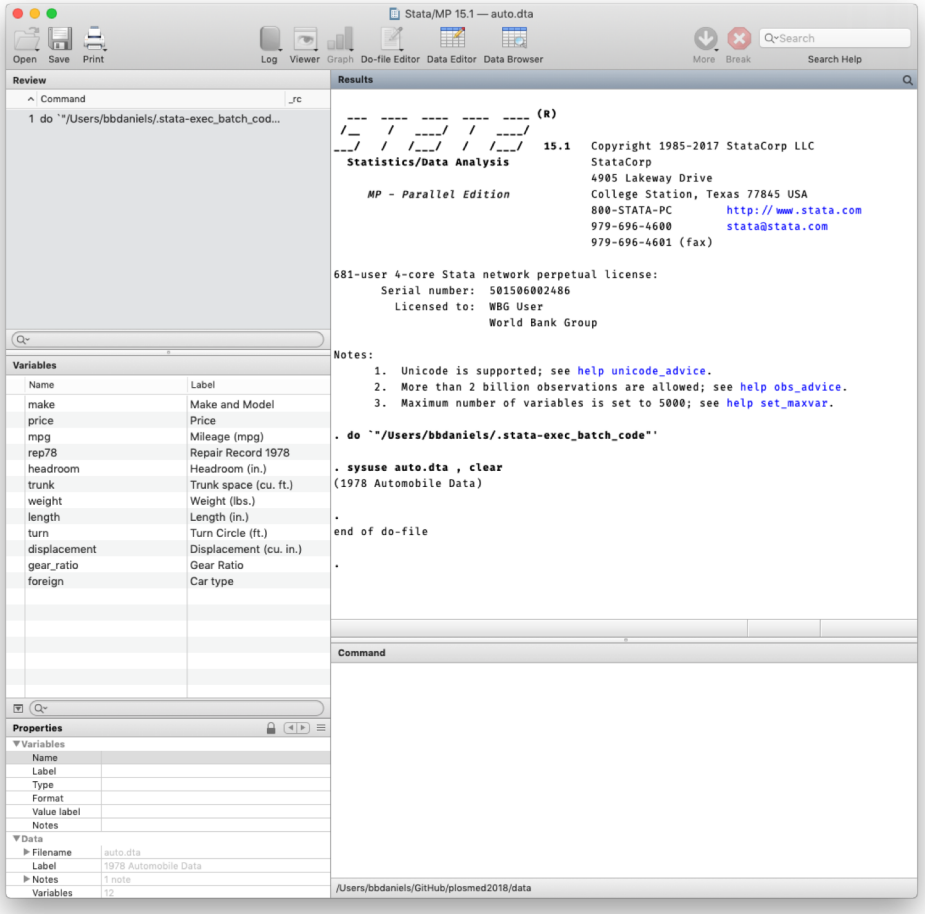


i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Scientific Computing with Stata

1. Systematic naming conventions
2. Subsetting your data
3. Handling repetitive tasks
4. [*foreach*] and [*forvalues*]
5. [*bys:*] and [*egen*]
6. Programs



- Review: [Manage Successful Impact Evaluations Stata Labs]
- Nick Cox. "[Speaking Stata: how to repeat yourself without going mad]".

Naming in Stata: datasets, dofiles, variables

- Names should [*order * , seq*] sensibly, so same rules as before
- Raw data should match questionnaire (including order) – many people use varnames like s01_q01
- Labels should be English only so that other code does not break, and should be in “Proper Case”

Variables

Name	Label
providerid	Provider ID
sp1_h_1	Duration of Cough
sp1_h_2	Sputum
sp1_h_3	Past TB
sp1_h_4	Family TB
sp1_h_5	Blood in Sputum
sp1_h_6	Cough Throughout Day
sp1_h_7	Fever
sp1_h_8	Fever Type
sp1_h_9	Family Symptoms
sp2_h_1	Duration of Cough
sp2_h_2	Sputum
sp2_h_3	Type of Doctor SP Saw
sp2_h_4	Medicine Taken
sp2_h_5	For How Long Medicines...
sp2_h_6	Family TB
sp2_h_7	Provider Saw X-Ray Film
sp2_h_8	Provider Read X-Ray Re...
sp2_h_9	Blood in Sputum
sp3_h_1	Duration of Cough

Constructed: more descriptive, still systematic

- Constructed data should stay sensible: crop_type crop_amount
- Note that this often means reversing English adjective order!

Utilities

- [*rename*] command takes wildcards (acceptable in Construct dofiles)
  - [*rename *_h_1 h01_**]
  - [*rename *_h_* h0*[2]_*[1]*]
- [*lookfor*] (try [*return list*] after)

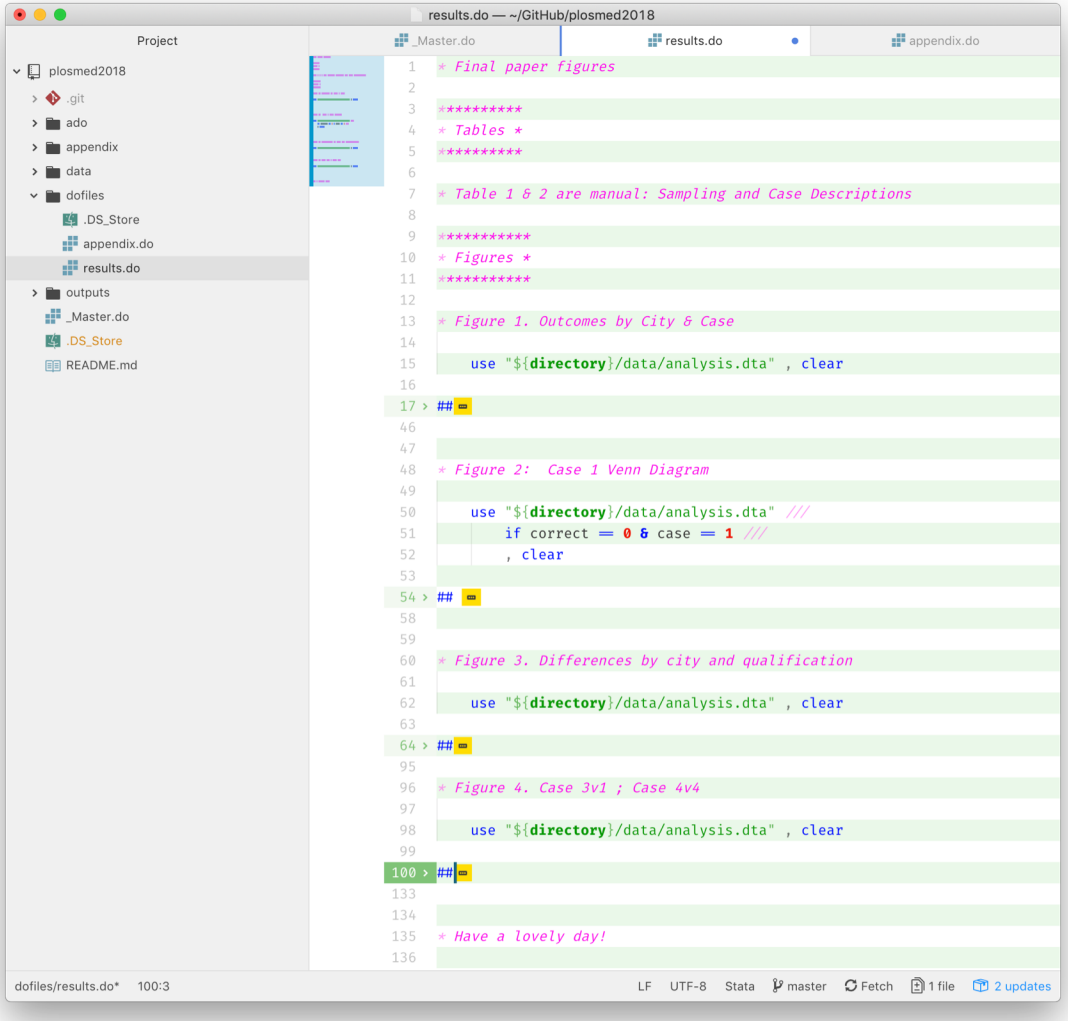
Variables

Name	Label
sp_id	SP ID
t_12	Ask to Return
t_12a	Return if Symptoms Persist
t_12b	Return if Symptoms Worsen
t_12c	Return to Get Medicines
t_12d	Return for Test Results
t_12e	Return After So Many Days
treat_cxr	Chest X-Ray
treat_refer	Referred Case
type	Sample Type
type_formal	MBBS Provider
visit	Visit
weight	City-Sample Weight
weight_city	Within-City-Case Weight

Subset your data; [*use*] often

Two major methods:

1. Create alternate datasets
   - "analysis_main.dta"
   - "analysis_main-long.dta"
   - "conf-neudc_prices-wide.dta"
   - "supplement_female-only.dta"
2. Subset with [*use*]
   - [*use *id* correct case*
     using "${dtFin}/analysis.dta"
     if case == 1, clear]
   - Takes varlist and if/in conditions

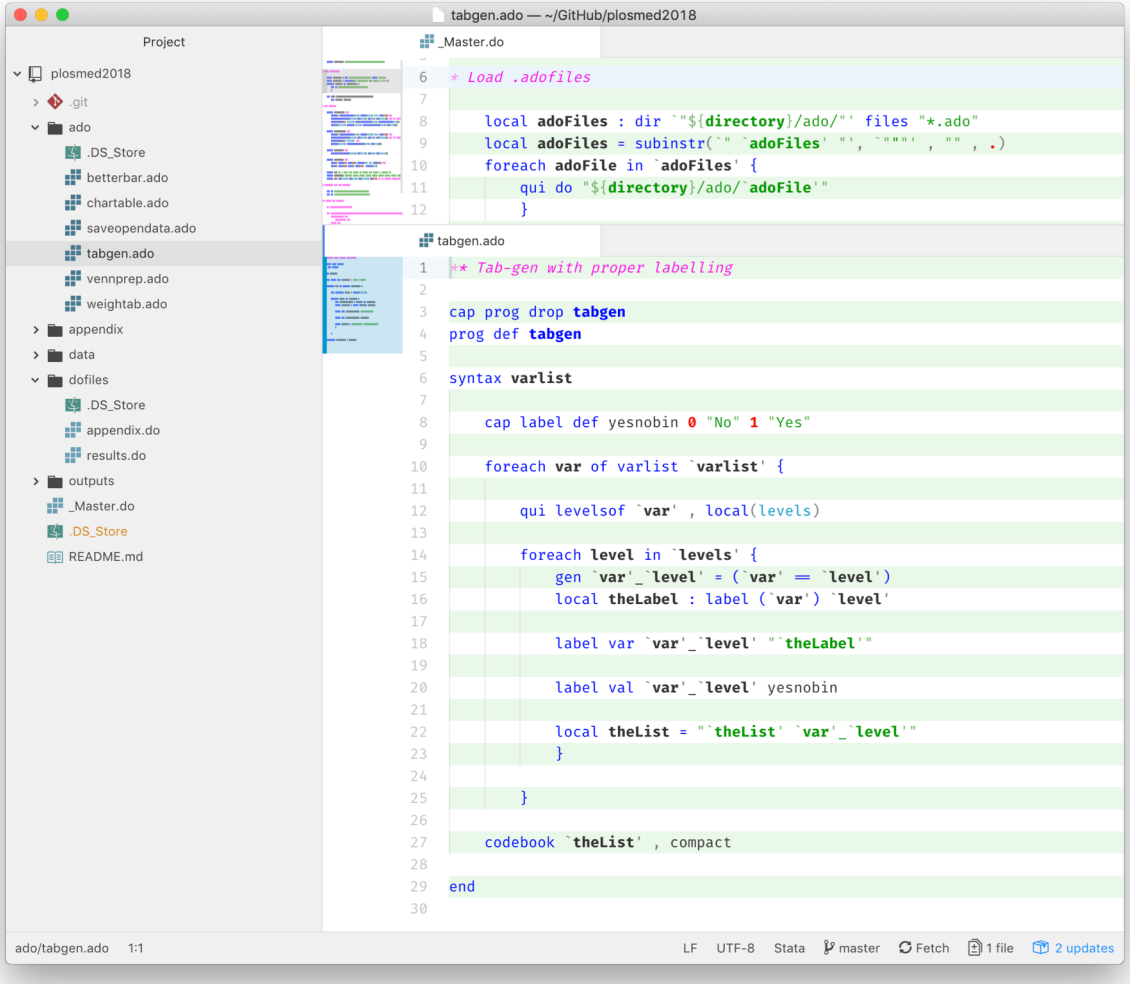


Repetitive tasks

Fool me thrice / rule of three:

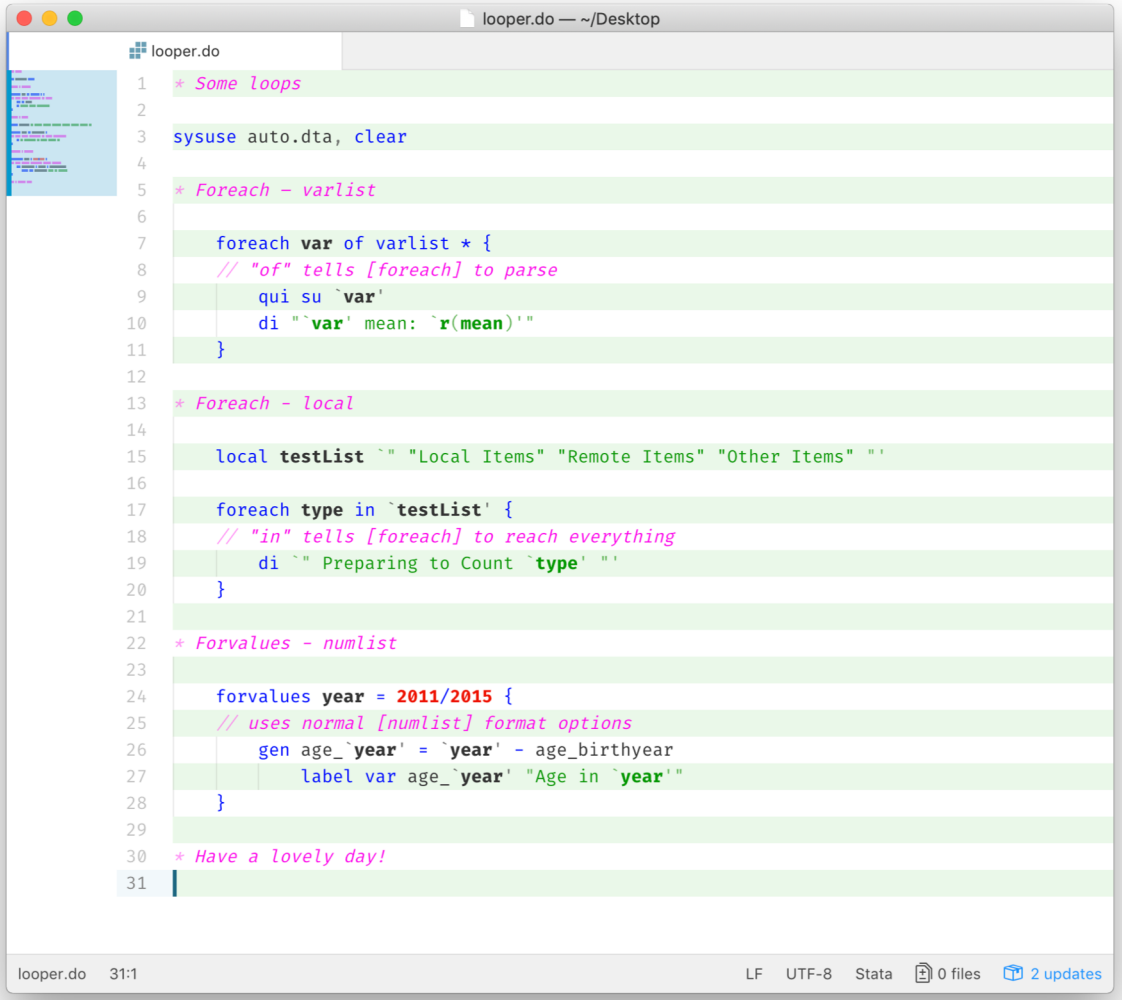
- The first time you do something, do it carefully and correctly
- The second time you do something, think about how you would generalize it
- The third time, implement the generalization

The fifth through 100th times, pat yourself on the back



[*foreach*] and [*forvalues*] loop commands

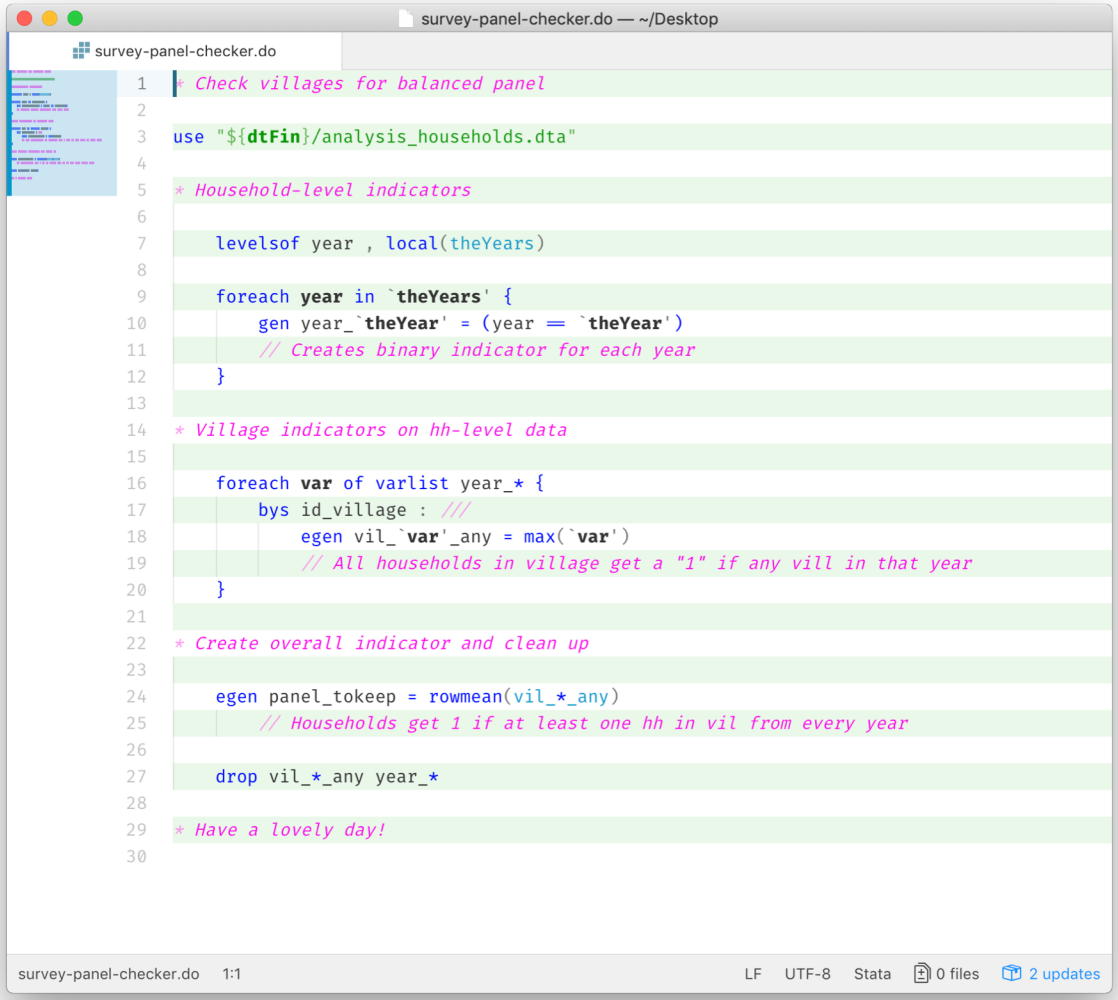
- Setting up loops: use a meaningful `index' local name
- Foreach replace `index' with the list items, in order
  - Variable names
  - Elements of a local
  - Whatever you want
- Be careful about hierarchy/indentation



[*bys:*] and [*egen*] use groups of observations

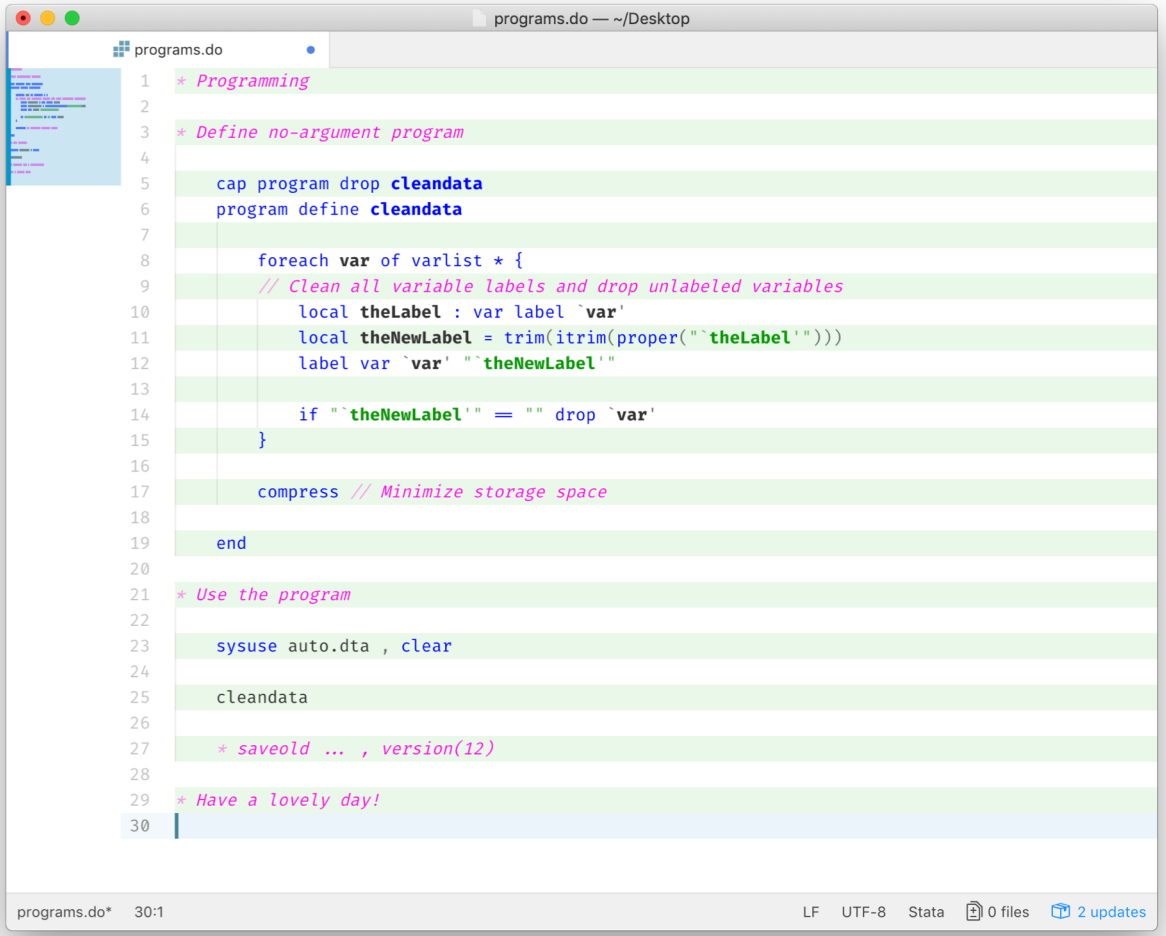
- You have long-format panel data of households covering several years.
- Which *villages* have at least one household in all survey years?

- [*collapse*] and [*merge*] would work with just two years, but an arbitrary number of years is harder…



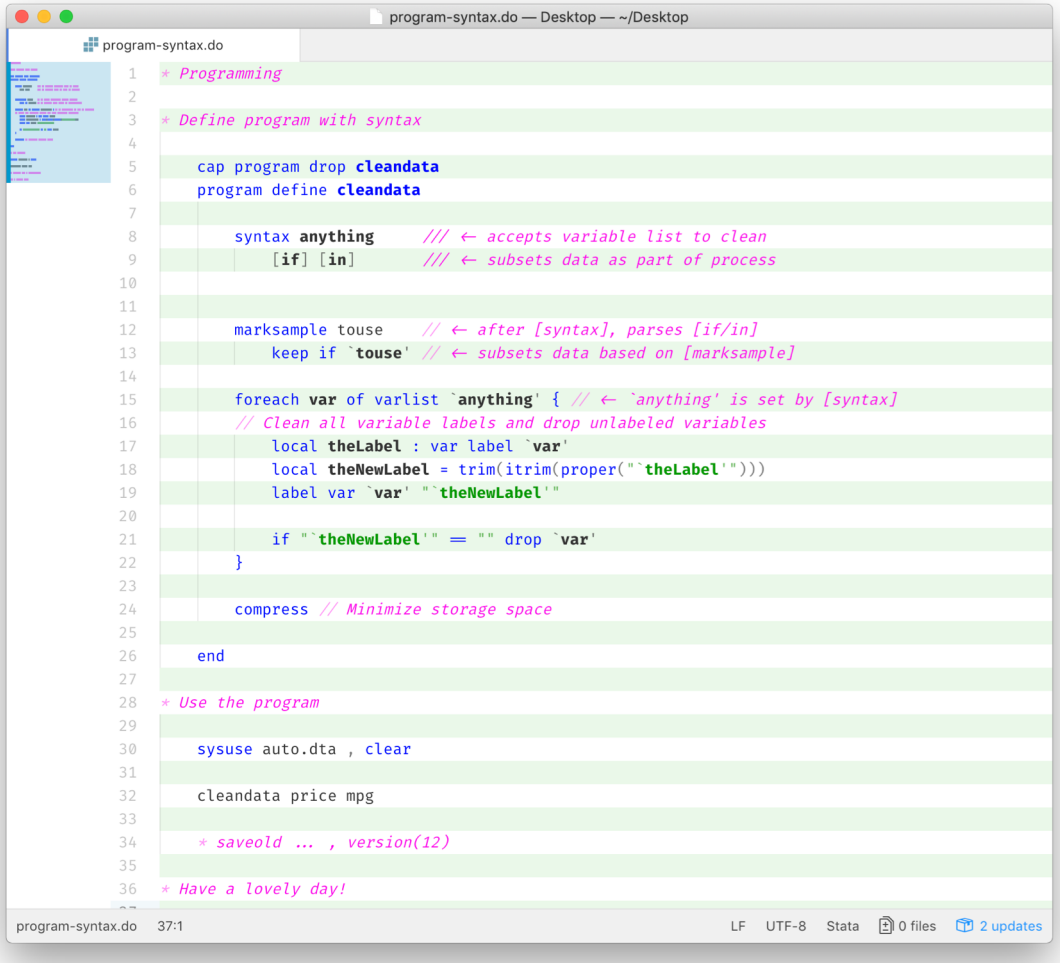
Generalizing with [*program*]

- [*program*] takes a loop-like code block and generalizes it into a command
- Allows you to call the same code block on demand
- Better than loops in that it allows multiple index-like entries: there are called “arguments” and “options”
- Can be further generalized into ado-files: more on that in future



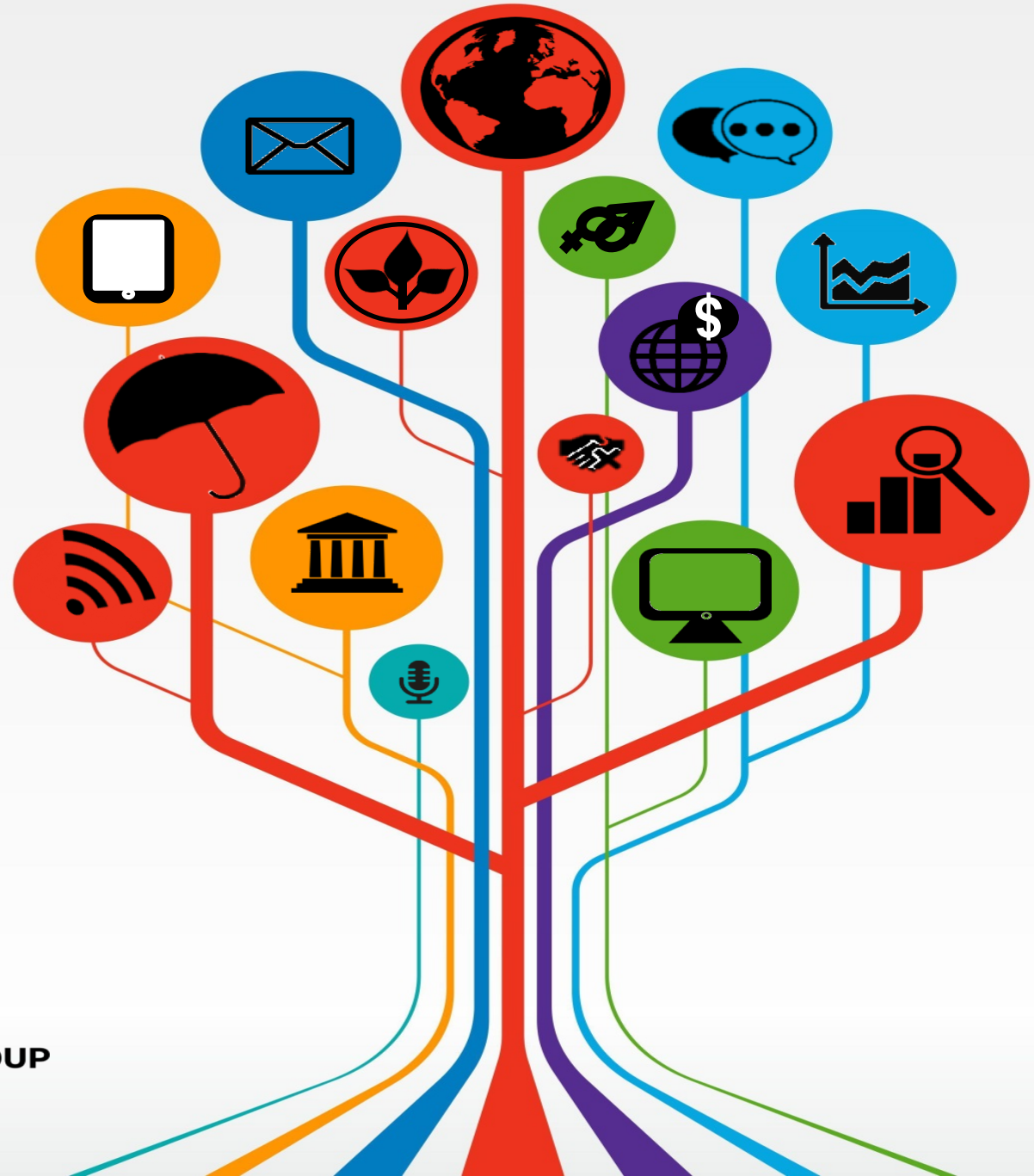
Generalizing with [*program*]

- [*program*] takes a loop-like code block and generalizes it into a command
- Allows you to call the same code block on demand
- Better than loops in that it allows multiple index-like entries: there are called “arguments” and “options”
- Can be further generalized into ado-files: more on that in future



Thank you!

worldbank.github.io/dimeanalytics



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP