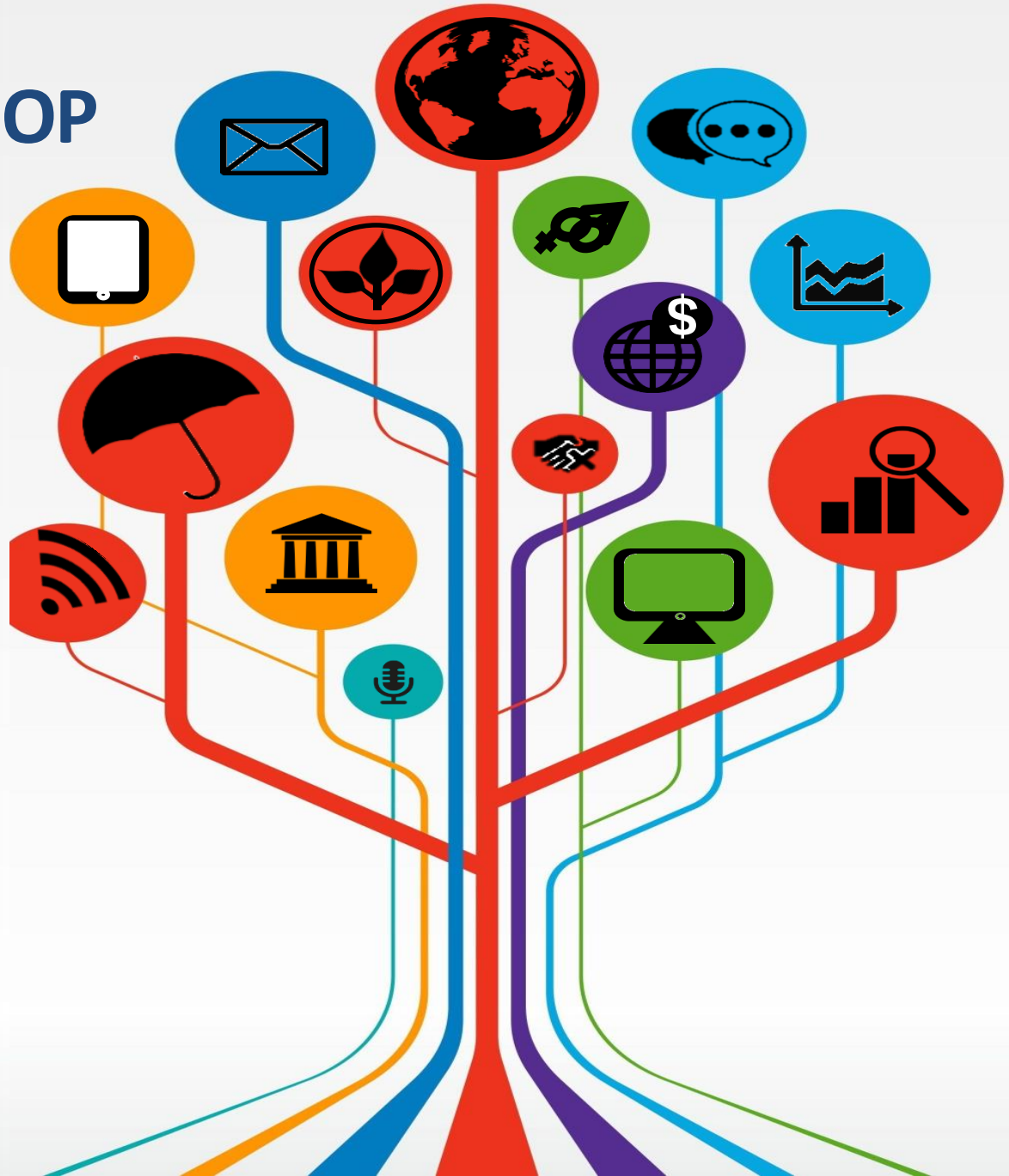
FIELD COORDINATOR WORKSHOP

Manage Successful Impact Evaluations

18 - 22 JUNE 2018
WASHINGTON, DC



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

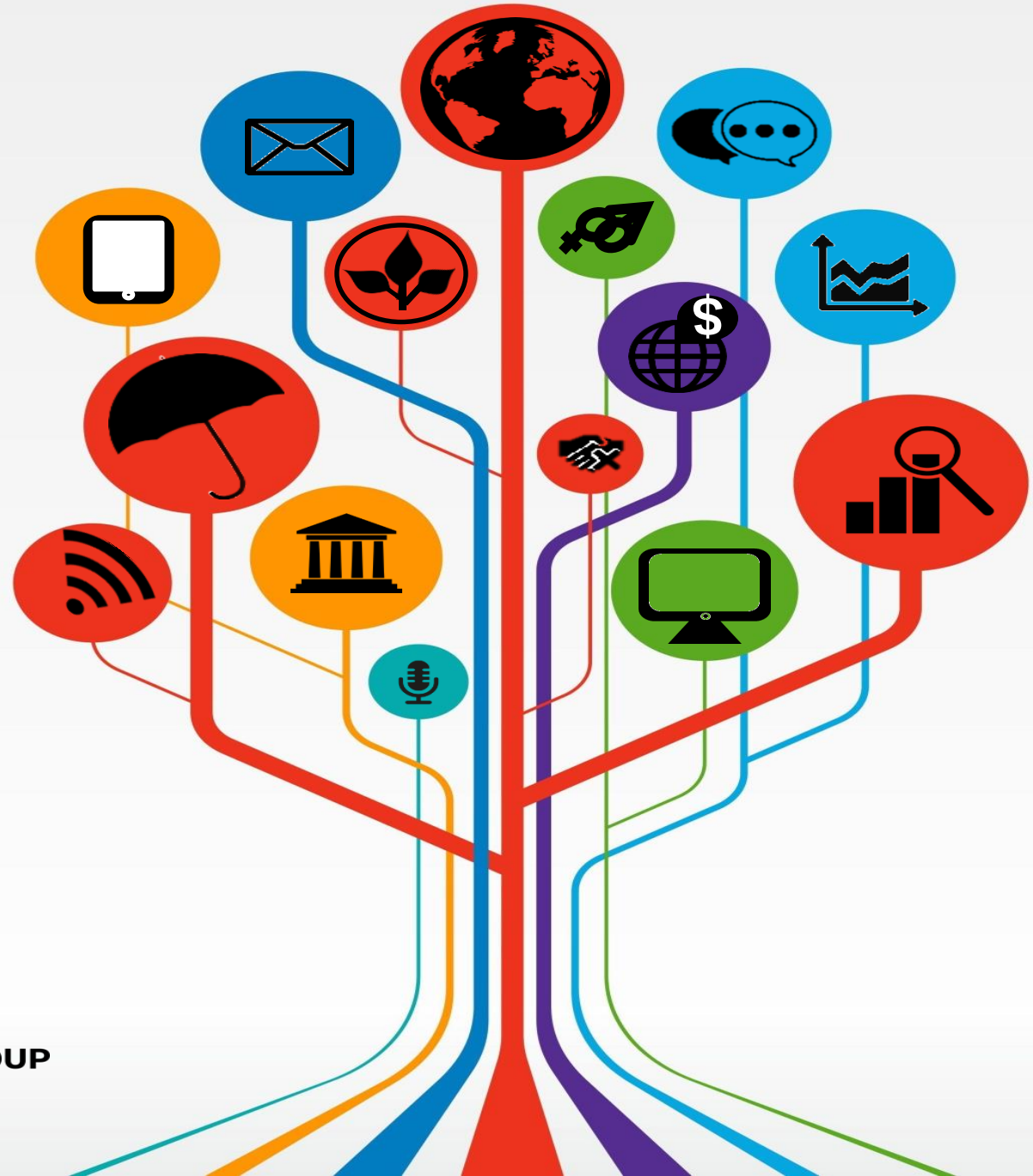
Stata Coding
for Reproducible Research

Stata Track 2

Prepared by **DIME Analytics**
DIMEAnalytics_Internal_Use_Only@worldbank.org

Presented by **Benjamin Daniels and Roshni Khincha**
bdaniels@worldbank.org | *rkhincha@worldbank.org*

http://www.worldbank.org/en/research/dime
June 19, 2018



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Introduction: Stata Coding

Stata coding is part of a reproducible research workflow.

- It should be easy to read and re-adapt
- This means in terms of *structure*, *syntax* and *style*
- Code should be modularized as much as possible
- Anything that might be used again can be saved as an “adofile”

Production

Collection → Analysis → Publication

SurveyCTO

Survey Forms

Raw Data

Dropbox

STATA

Raw Data

Cleaned Data & Outputs

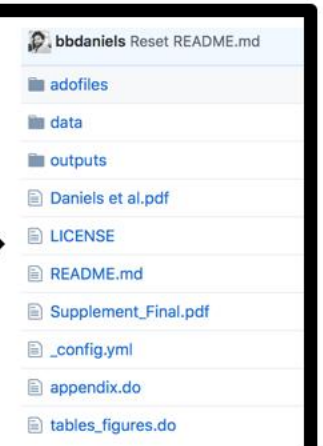
Dofiles

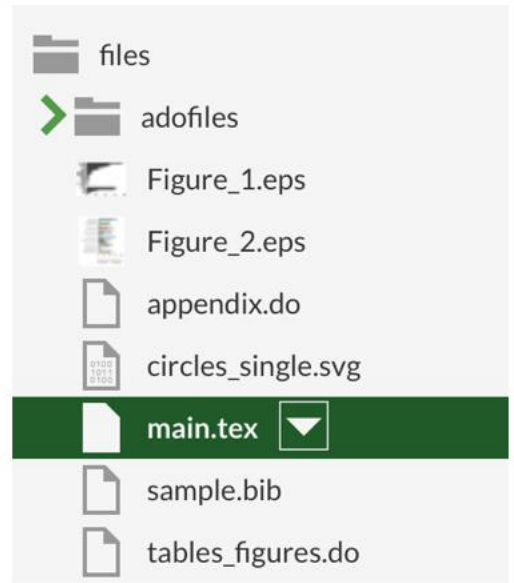
Overleaf

LaTeX Files and Bibliography

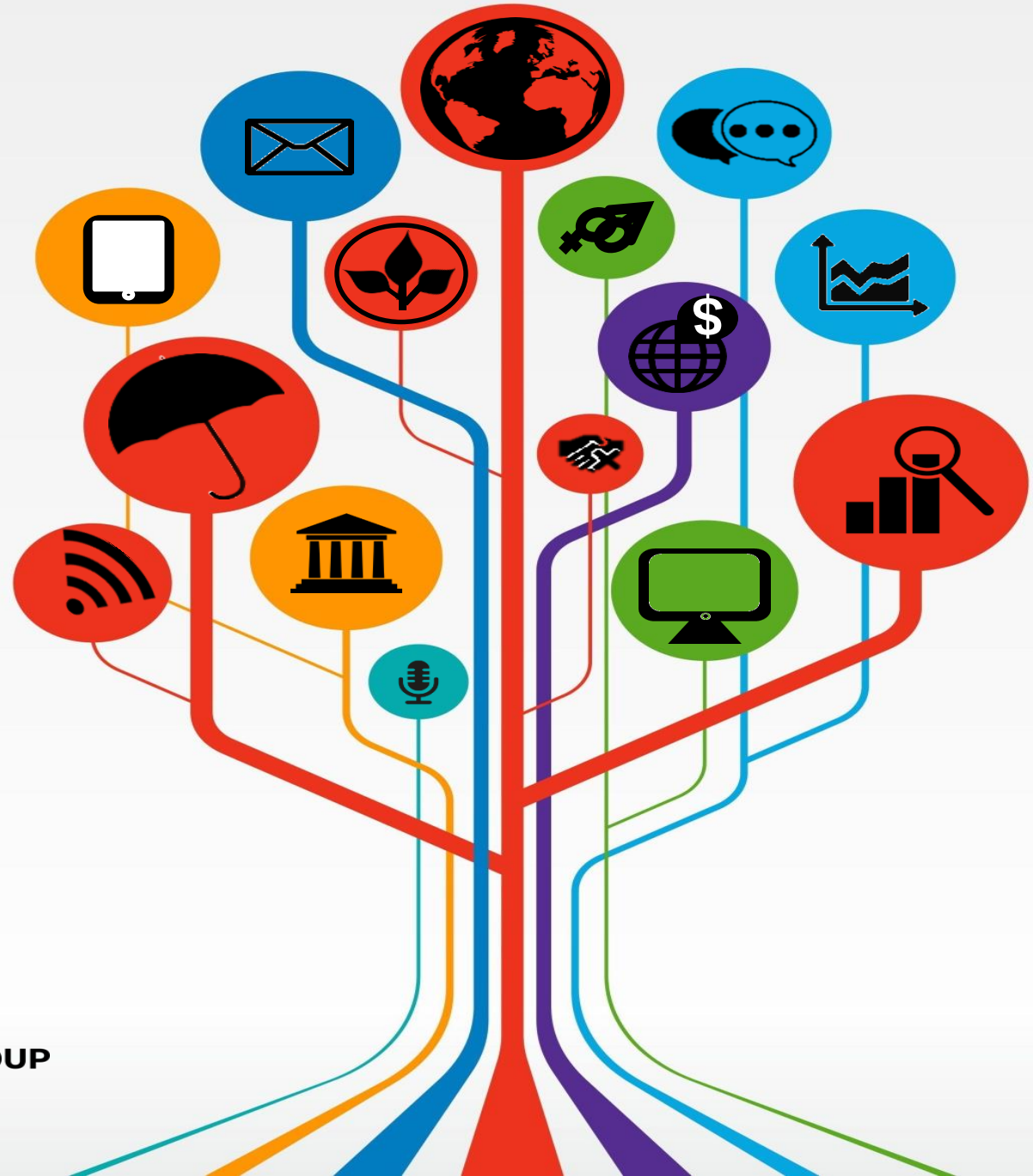
Public Release

GitHub





Stata Coding: Structure



i2i DIME
TRANSFORM DEVELOPMENT

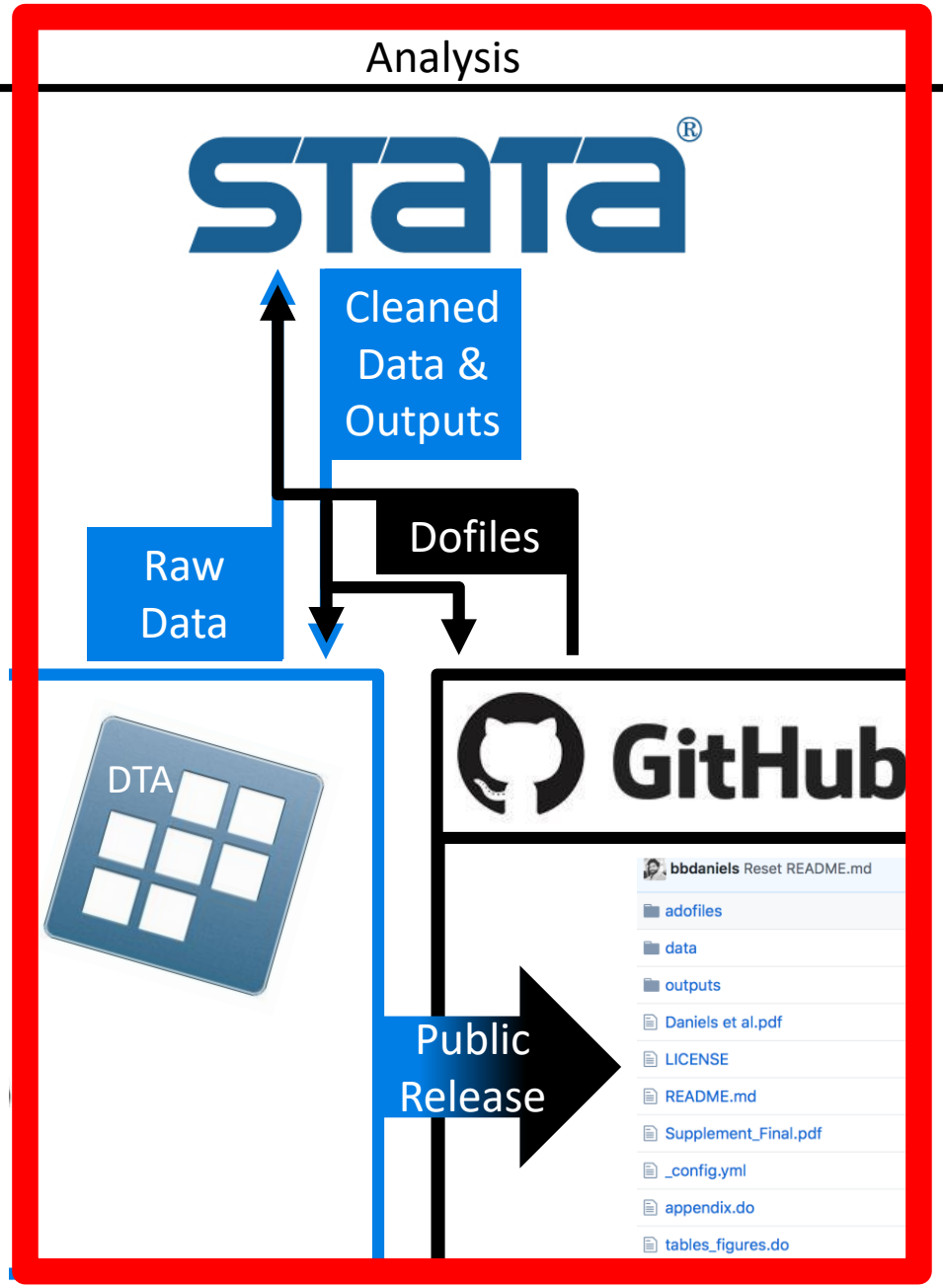
WORLD BANK GROUP

Production

Collection → Analysis → Publication

Stata structure is the environment your code lives in.

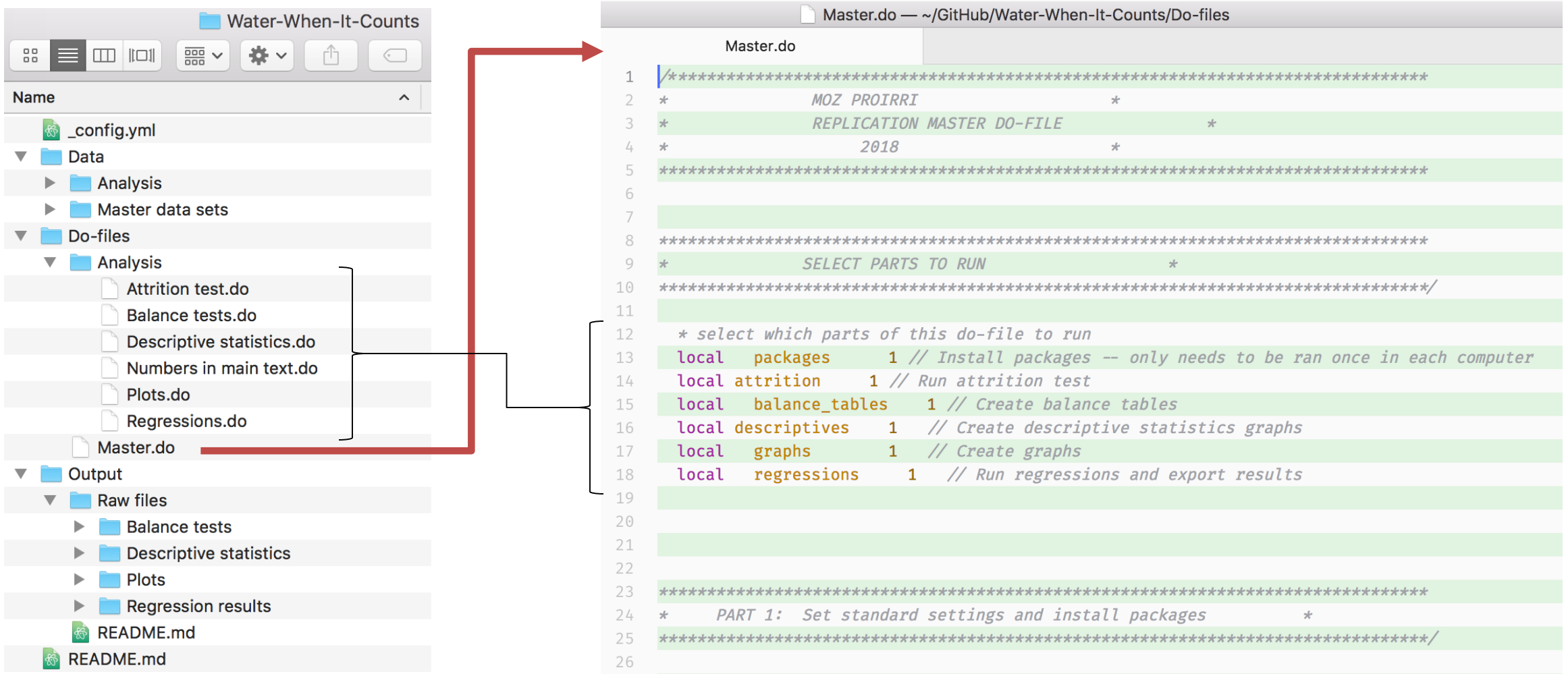
- Publishing a paper is not enough!
- Code and data to reproduce results is often required by Open Access agreements or journals themselves.
- And even if it isn't, others may want or need to use or reuse your code in the future, so it is good academic citizenship.



Modular organization is constant preparation

- It is much easier to maintain files **modularly** from the outset than to clean up “everything dofiles”.
- Cleaning is **separated** from analysis so that each analysis step begins with [*use*].
- The final product then only consists of keeping the analysis files that are used in the paper and **archiving** the rest.

What does this look like in practice?

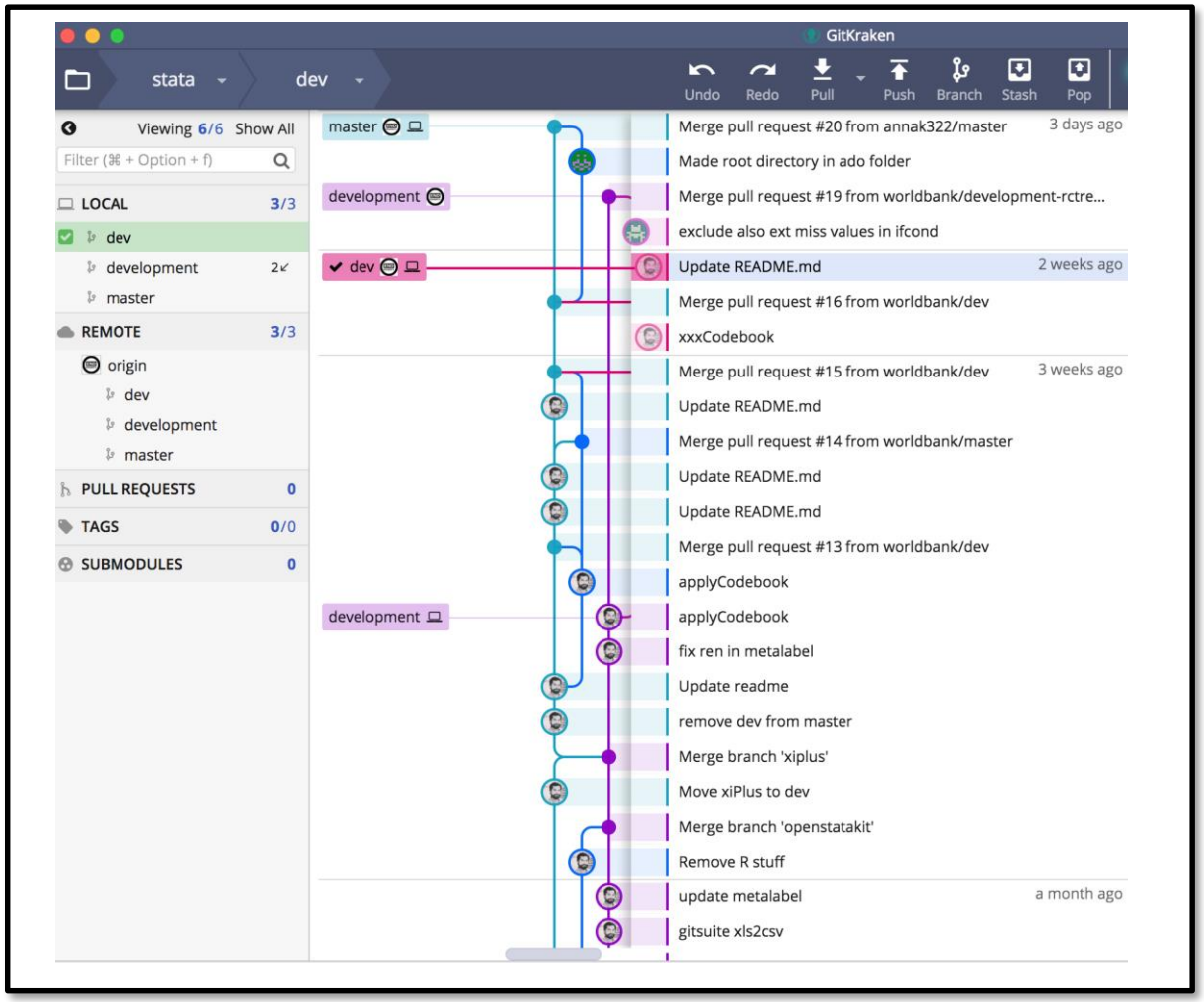


Modular data analysis with Git(Hub)

Git is a great workflow structure for this because components can be *added*, *deleted*, and *recovered* without information loss, even if you are working alone.

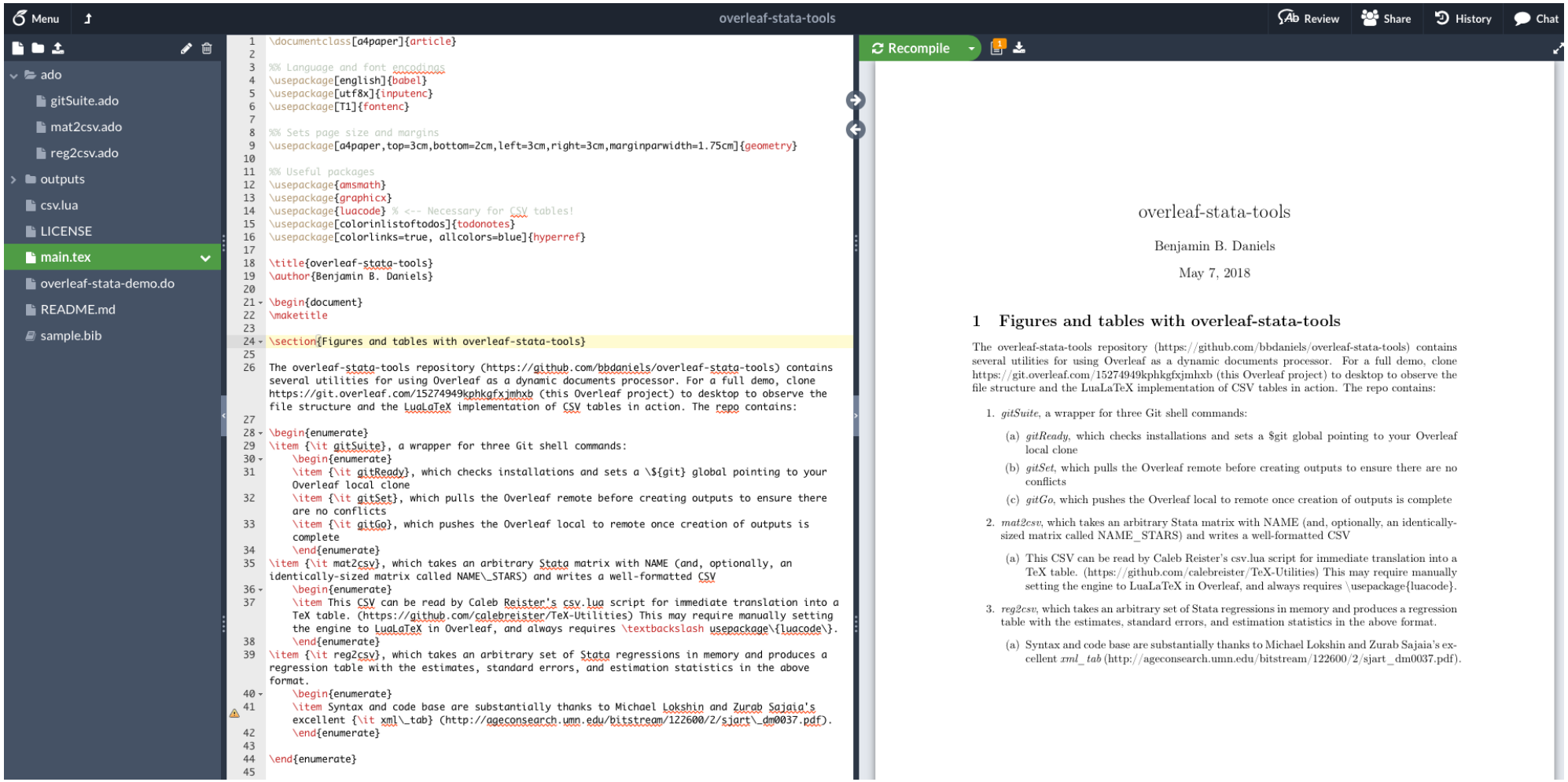
Every version of **everything** is available **everywhere.**

- **Version Control** – Analysts can always find "that code that did that one thing", and the current master version will have various experimental "branches" until they are confirmed functional and merged in.
- **Efficiency** – This allows simultaneous distributed editing of dofiles, text files, bibliographies, etc., but is not appropriate for Office/.dta/.pdf files.
- **Privacy** – Must use paid account (such as github.com/worldbank) to have "private" or "secret" folders.



Writing: LaTeX Documents (Overleaf + Git)

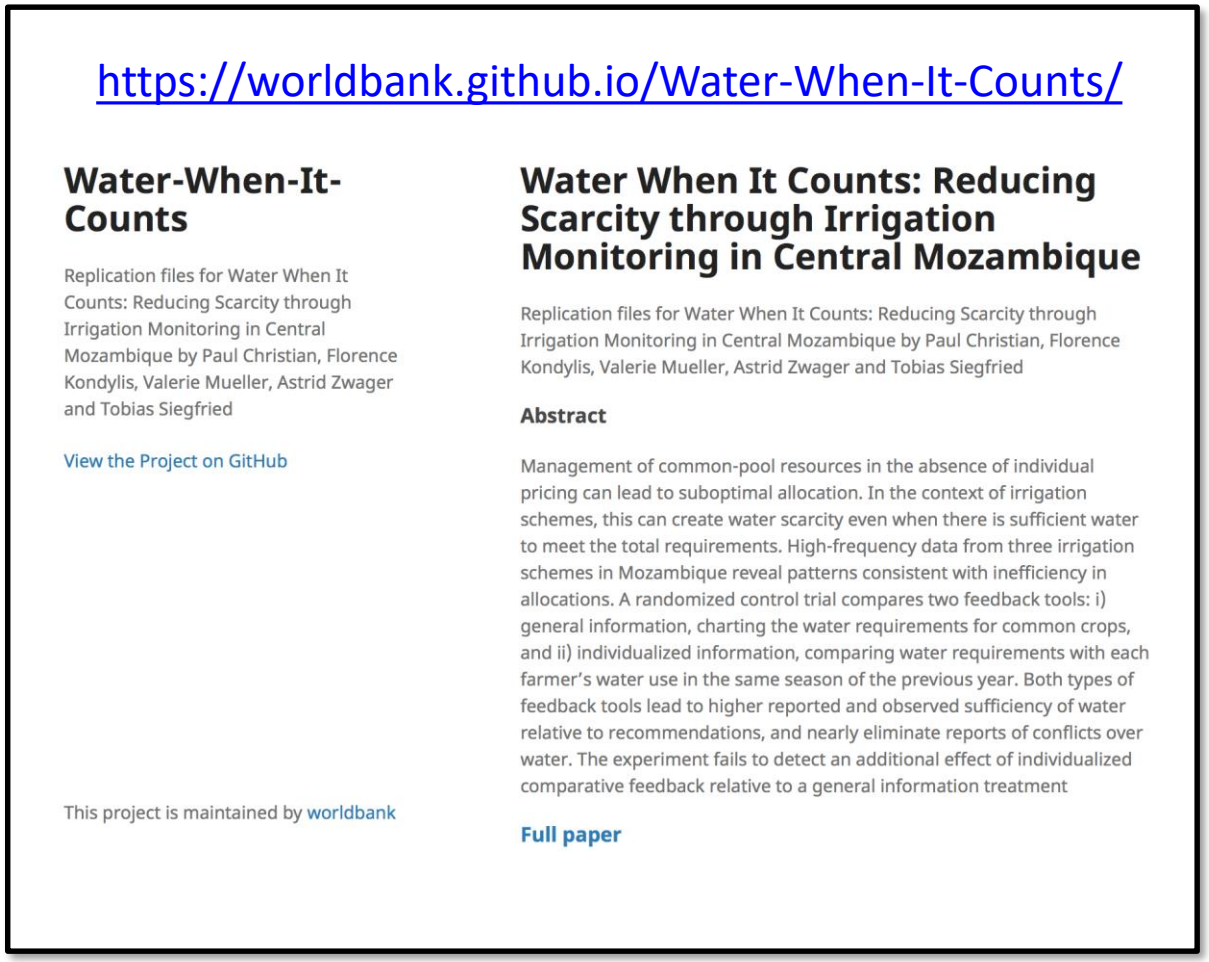
Overleaf is kept up to date via link to GitHub



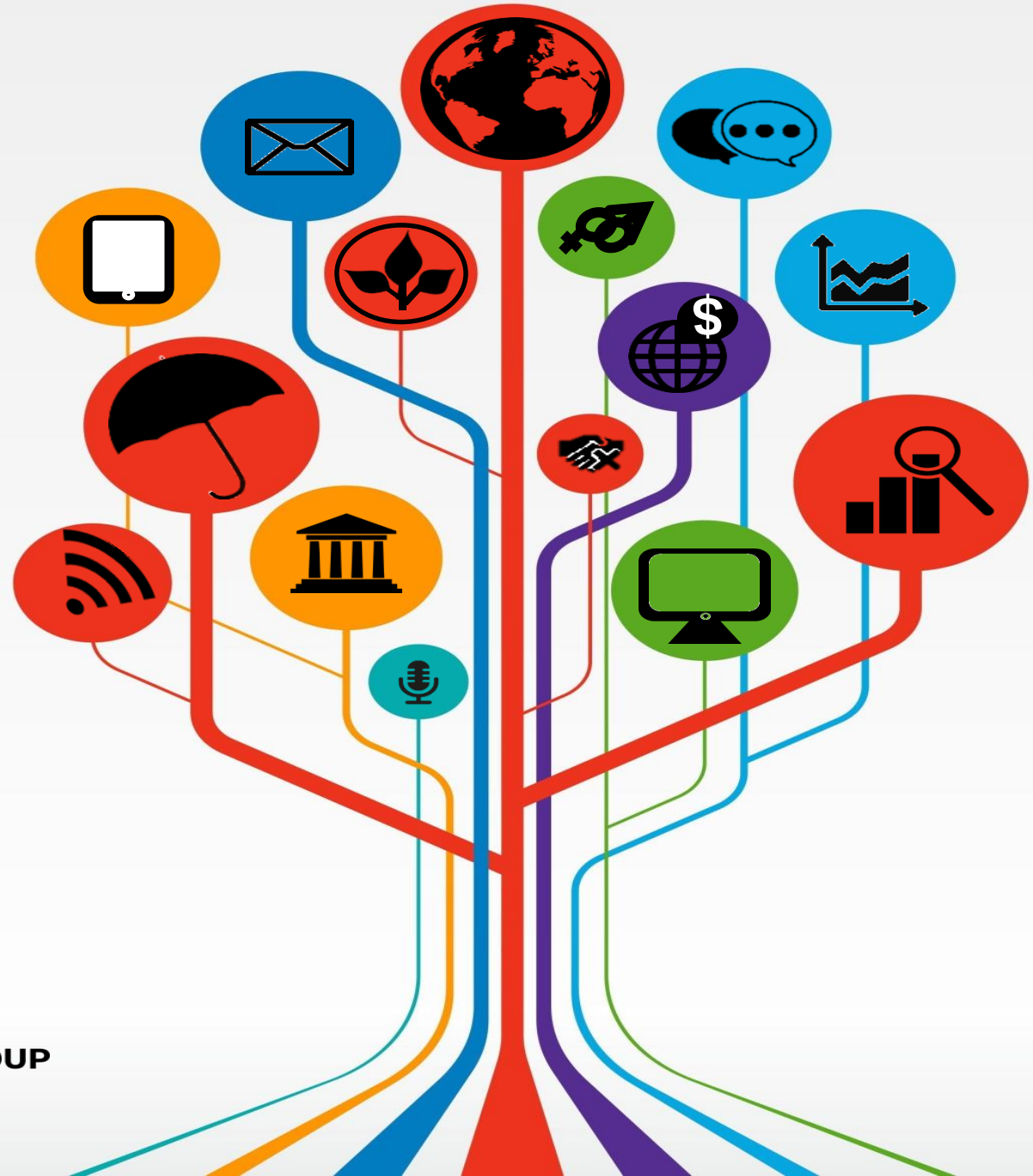
Publication: Public Release of Data + Code

Writing code on Git with collaboration public release in mind prepares the project for preservation and open access

- The version history answers seminar questions that start with “what happened when you tried….?”
- The final repository has reusable code that other projects will benefit from
- Picking up the same project to extend or replicate analyses later is a one-click process



Stata Coding: Syntax



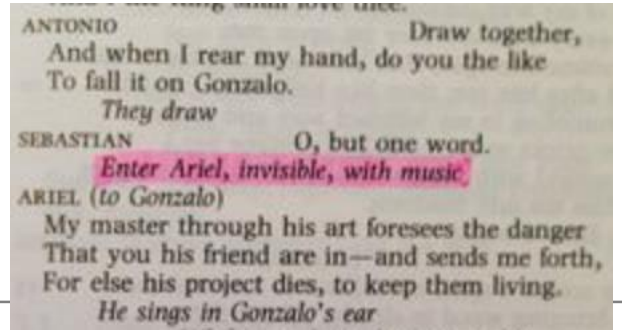
i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Stata syntax is the language of your code

- Stata code is a *script*, not a *program*
- Think of a play
  - You can *read the script* and get a good idea of what will happen when the instructions are followed
  - This is as important as the actual production (output) because it serves as a record of what you did
- Someone is going to want to read your code!
  - They want to see exactly how you got to the results
  - They want to do something similar but not identical in their own work

- More technical details:
  - Stata does not work like most ("object-oriented") programming languages
  - These (including R) treat "objects" or "functions" as the fundamental thing that is referenced by code
  - Stata is a *scripting* language for *econometrics*: its basic elements are observations and characteristics (what you know as variables but a programmer will not understand)
  - You cannot [*browse*] in R because a data.frame is not a basic object there


ANTONIO Draw together,
And when I rear my hand, do you the like
To fall it on Gonzalo.
They draw
SEBASTIAN O, but one word.
Enter Ariel, invisible, with music
ARIEL (*to Gonzalo*)
My master through his art foresees the danger
That you his friend are in—and sends me forth,
For else his project dies, to keep them living.
He sings in Gonzalo's ear

Functional syntax

- **``Stata quotes''**
  - Will break lots of programming tools because the backtick ` is a special character
  - Are really important to get right
  - `NAME' calls a local macro
  - local NAME "STRING" stores STRING in `NAME'
  - local NAME `" "STRING" "' stores "STRING" in `NAME'
  - local NAME `=2+2' stores 4 : this works anywhere, like yline(`=`beta'*`r(mean)'')

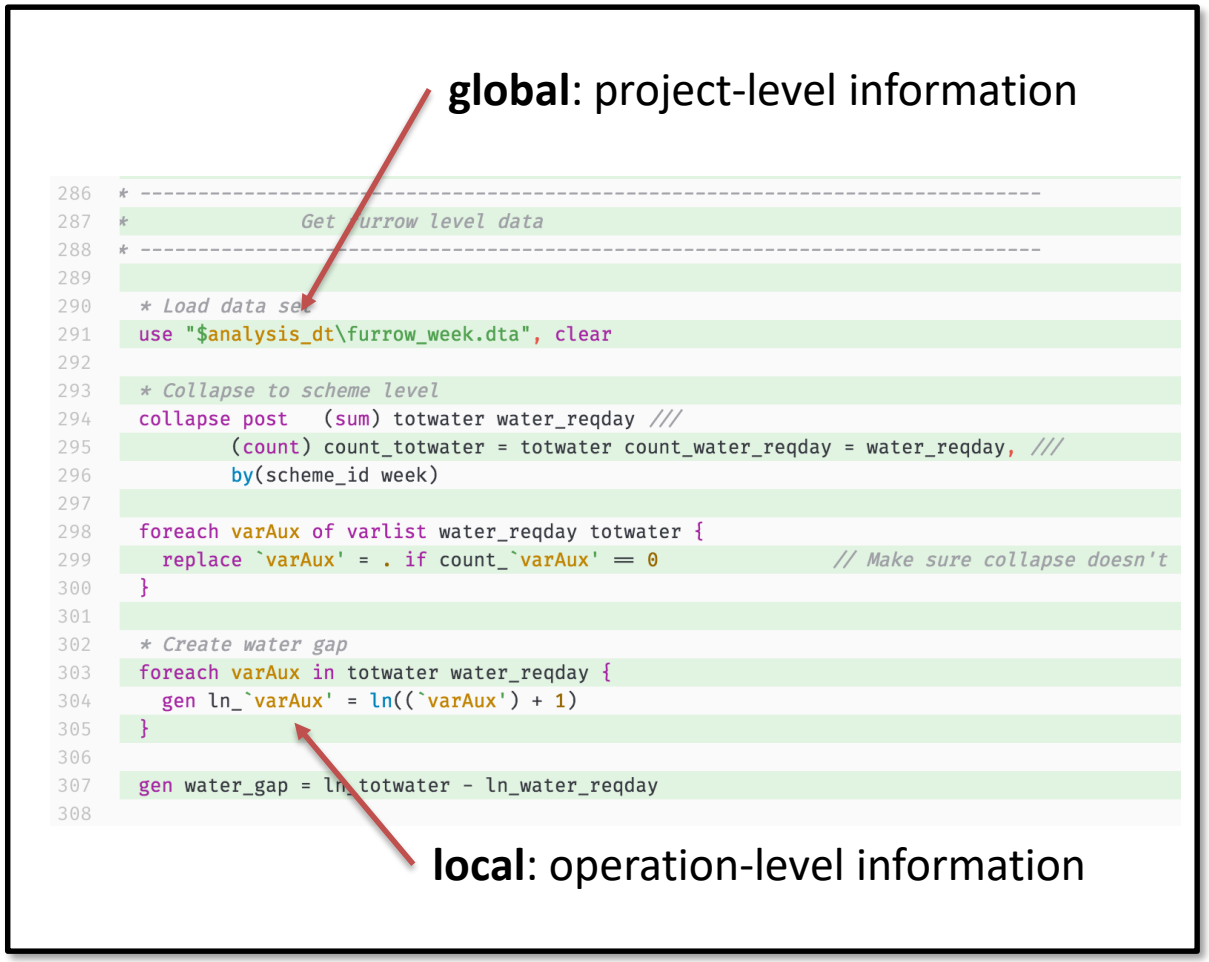
https://www.stata.com/help.cgi?quotes

- **Equals sign =**
  - When used with a macro, they evaluate what comes after
  - local NAME = min(2,3,4) stores 2 in `NAME'
  - local NAME = "min(2,3,4)" stores min(2,3,4) in `NAME'

- **Backslash **
  - *never* use in filepaths: /users/dropbox/
  - "Escapes" functional characters:
    - local NAME = "\`OTHERNAME'" stores `OTHERNAME', not the contents of the local `OTHERNAME'

Macros: `local' and ${global}

- "Macros" (this is what programmers call "variables") hold information within a Stata session
- Difference in *scope* – Use them appropriately according to scope
- Only define globals in the master do-file.
- Use locals everywhere else (varlists, loops, estimation results)… they are deleted after the dofile finishes



Macros: naming convention

- Always give a local or a global a name where the reader can tell what it represents.
- Especially in loops, abstract indices can become confusing quickly!
- To reiterate: Stata is a *scripting* language, not a *programming* language.

```
*Before
forvalues x = 1/3 {
    forvalues y = 1/10 {
        forvalues z = 1/6 {

            sum variable c1_harv_c`z'_s`x'_p`y'
        }
    }
}



*After
forvalues seasnum = 1/3 {
    forvalues plotnum = 1/10 {
        forvalues cropnum = 1/6 {

            sum variable c1_harv_c`cropnum'_s`seasnum'_p`plotnum'
        }
    }
}
```

Tips for usages of globals

- Only define globals in the main master do-file
- Usages of globals:
  - Root folders
  - Standardize conversion coefficients
  - Varlists commonly used across the project – for example list of controls included in multiple regressions

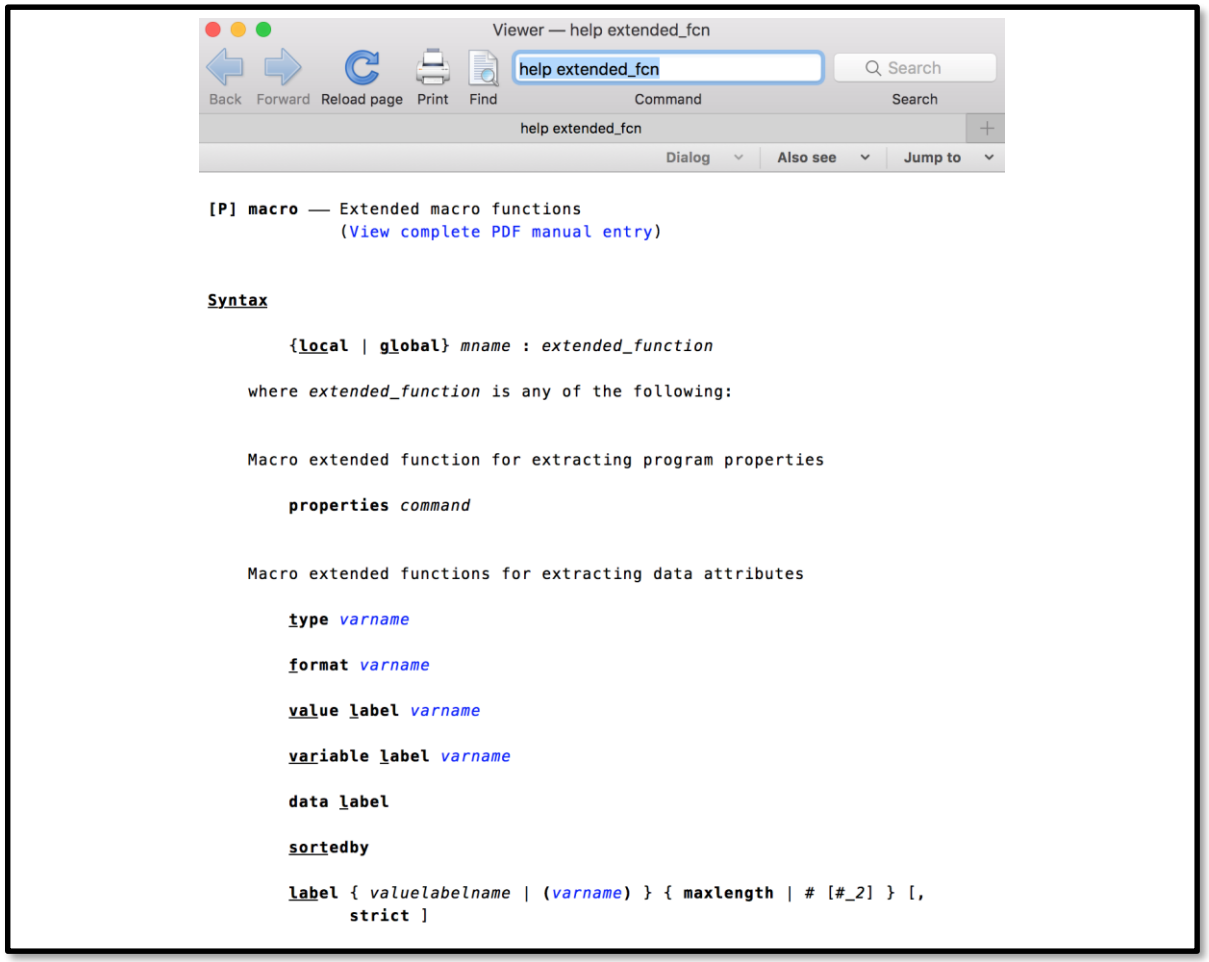
Tips for usages of locals

Use locals to shorten variable names and make them more explanatory

```
*Before
sum     c_harv_value                              if c_harv_q14_p`plotNum'_s`seasonNum' == 1
reg     c_harv_value fertilizer seedtype labor    if c_harv_q14_p`plotNum'_s`seasonNum' == 1



*After
local   grewRice     c_harv_q14_p`plotNum'_s`seasonNum'
sum     c_harv_value                              if `grewRice' == 1
reg     c_harv_value fertilizer seedtype labor    if `grewRice' == 1
```

“Extended functions” in Stata

- Because variables are the core of Stata, it knows a lot about them
- Extended functions access that information
  - [*h extended_fcn*]
- They pull information about variables (types, labels, etc), locals, and other Stata native objects into macros
- [*local theLabel : val lab foreign*]

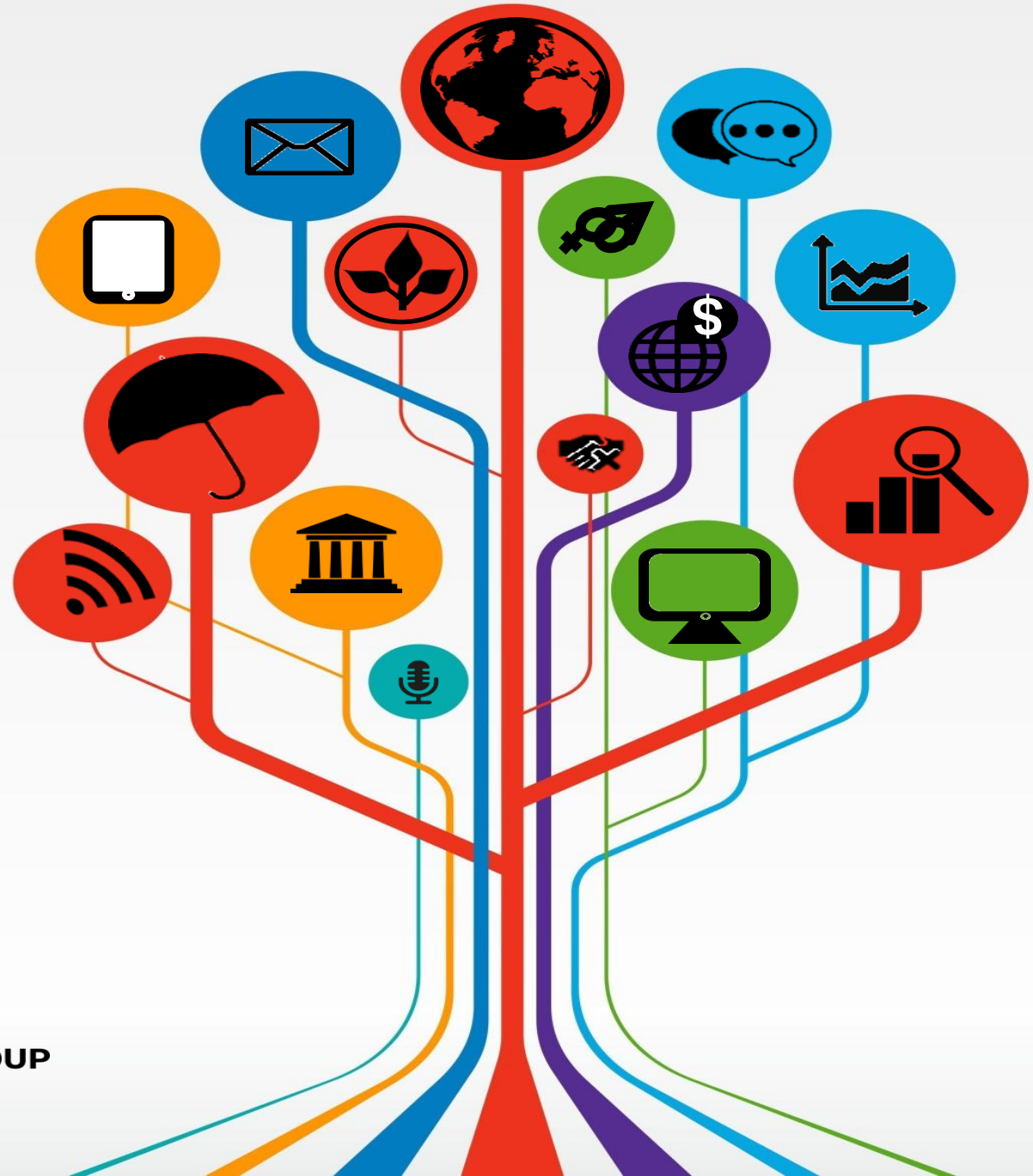


Stored results and [*return*], [*ereturn*], [*creturn*]

```
1
2  clear
3
4  sysuse auto
5
6  reg price mpg rep78 headroom
7    local mpgBeta= _b[mpg]
8    di "`mpgBeta'"
9
10 * return holds outputs
11
12   return list
13
14   mat results = r(table)
15   matlist results
16
17 * ereturn holds estimates
18
19   ereturn list
20   di "`e(N)'"
21   count if e(sample) == 1
22
```

```
23 * creturn holds system information
24
25   creturn list
26
27   foreach letter in `c(alpha)' {
28     di "`letter'"
29     }
30
31 * return is useful!
32
33   qui count
34   forvalues i = 1/`r(N)' {
35     local theName = make[`i']
36     di "`theName'"
37     }
38
39 * Need a blank line at the end
```

Stata Coding: Style

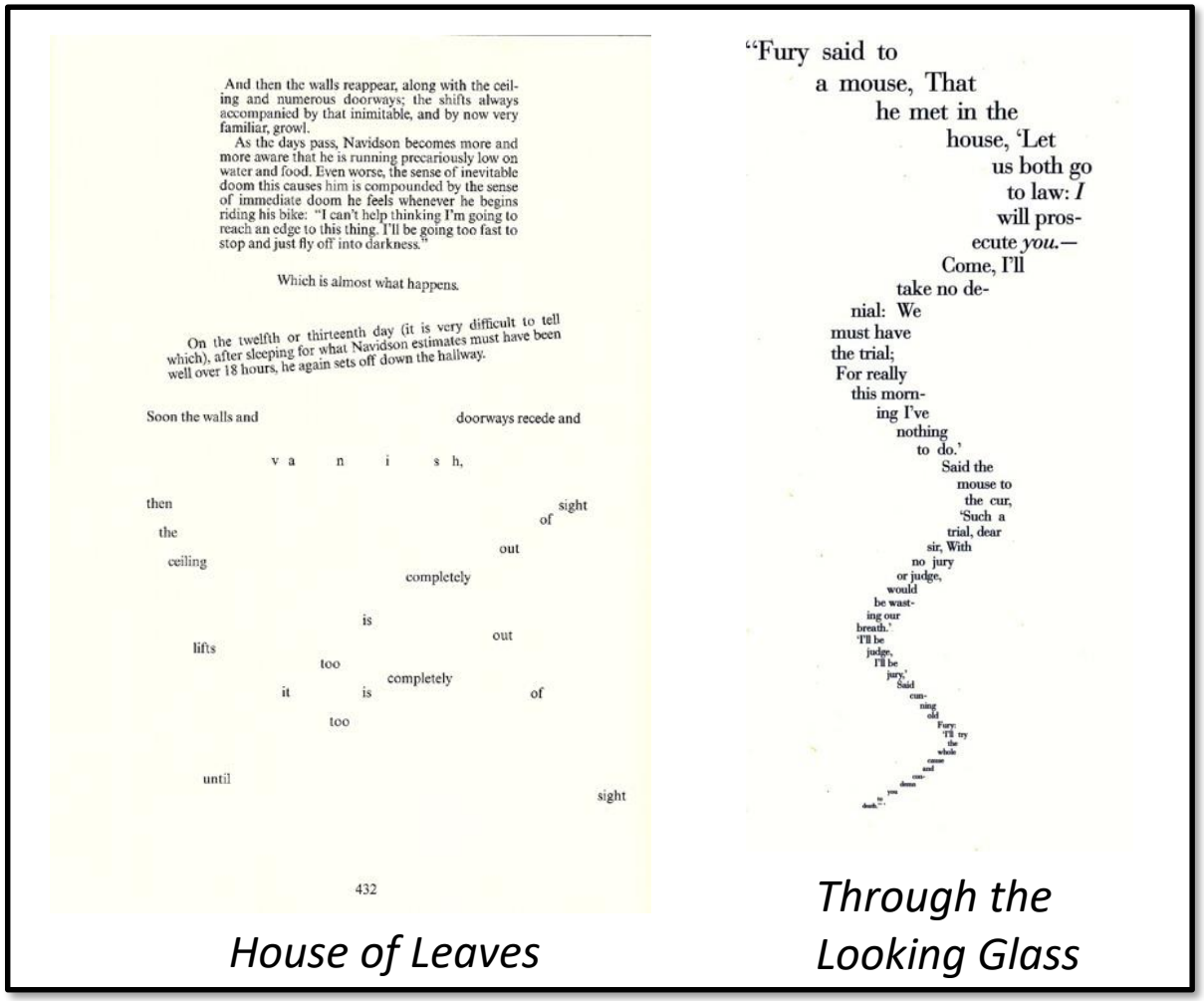


i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Stata style is the formatting of your code

- "Typesetting": the organization of the material induces a logic in the eye of the reader
- This can work *for* or *against* the meaning that the code conveys
- Simple but powerful elements: alignment, white space, line breaks, comments



And then the walls reappear, along with the ceiling and numerous doorways; the shifts always accompanied by that inimitable, and by now very familiar, growl.

As the days pass, Navidson becomes more and more aware that he is running precariously low on water and food. Even worse, the sense of inevitable doom this causes him is compounded by the sense of immediate doom he feels whenever he begins riding his bike: "I can't help thinking I'm going to reach an edge to this thing. I'll be going too fast to stop and just fly off into darkness."

Which is almost what happens.

On the twelfth or thirteenth day (it is very difficult to tell which), after sleeping for what Navidson estimates must have been well over 18 hours, he again sets off down the hallway.

Soon the walls and doorways recede and

v a n i s h,

then sight
of
the out
ceiling completely
is out
lifts
too completely
it is of
too

until sight

432

House of Leaves

"Fury said to
a mouse, That
he met in the
house, 'Let
us both go
to law: *I*
will pros-
ecute *you.*—
Come, I'll
take no de-
nial: We
must have
the trial;
For really
this morn-
ing I've
nothing
to do.'
Said the
mouse to
the cur,
'Such a
trial, dear
sir, With
no jury
or judge,
would
be wast-
ing our
breath.'
'I'll be
judge,
I'll be
jury,'
Said
cun-
ning
old
Fury:
'I'll try
the
whole
cause
and
con-
demn
you
to
death.'

Through the Looking Glass

Stata style makes your code “readable”

- **In general, spell out commands and variable names fully**
- This makes sure that:
  1. People in any language can tell what command you are using
  2. Variable lists don’t change based on ordering (*var1-var10* and *var_** are particularly bad practice)
  3. Code reads well in English as a narrative for your own sanity

White Space

- Stata does not distinguish between one empty space and many empty spaces, or one line break or many line breaks

- It makes a big difference to the human eye and we would never share a Word document, an Excel sheet or a PowerPoint presentation without thinking about white space – although we call it formatting

White space used for vertical alignment

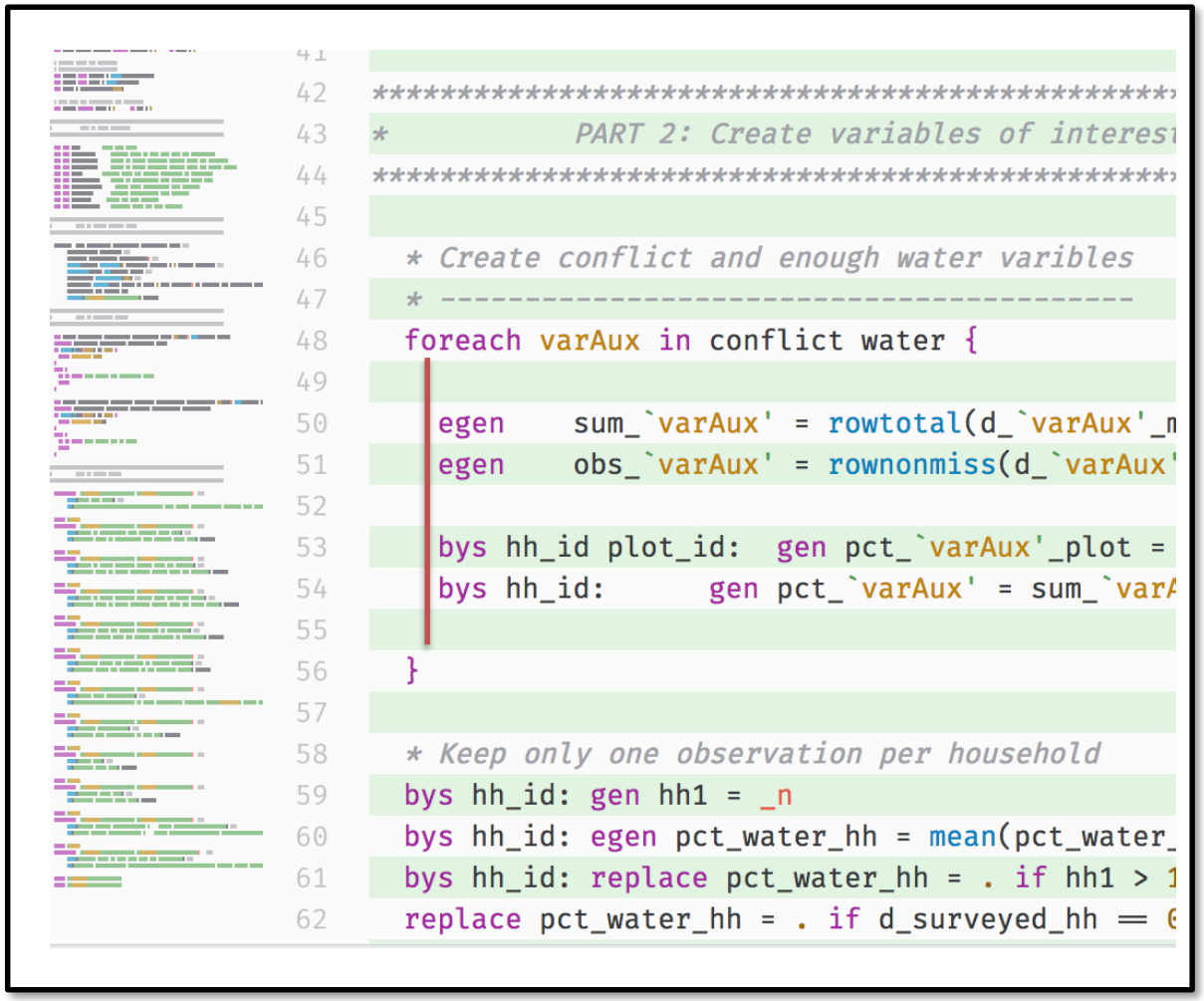
```
gen NoPlotDataBL = 0
replace NoPlotDataBL = 1 if c_plots_total_area >= .

gen NoHarvValueDataBL = 0
replace NoHarvValueDataBL = 1 if c_harv_value >= .

rename c_gross_yield c1_gross_yield
rename c_net_yield  c1_net_yield
rename c_harv_value c1_harv_value
rename c_total_earnings c1_total_earnings
rename c_input_spend c2_inp_total_spending
rename c_IAAP_harv_value c1_IAAP_harv_value
rename c_plots_total_area  c1_total_plotsize
rename  c1_cropPlotShare_??? c1_cropPlotShare_all_???

tempfile BL_append
save  `BL_append'
```

```
gen     NoPlotDataBL = 0
replace NoPlotDataBL = 1        if c_plots_total_area >= .

gen     NoHarvValueDataBL = 0
replace NoHarvValueDataBL = 1   if c_harv_value >= .

rename  c_gross_yield           c1_gross_yield
rename  c_net_yield             c1_net_yield
rename  c_harv_value            c1_harv_value
rename  c_total_earnings        c1_total_earnings
rename  c_input_spend           c2_inp_total_spending
rename  c_IAAP_harv_value       c1_IAAP_harv_value
rename  c_plots_total_area      c1_total_plotsize

rename  c1_cropPlotShare_???    c1_cropPlotShare_all_???

tempfile BL_append
save    `BL_append'
```

White space: Indentations suggest hierarchy

Makes code much more readable!

- Use for preserve/restore, loops and all other commands with curly brackets
- Very easy to see in “minimap” viewer (all advanced editors can display this – Atom is a particularly good one)



```
42  ************************************************
43  *           PART 2: Create variables of interest
44  ************************************************
45
46    * Create conflict and enough water varibles
47    * ---------------------------------------
48    foreach varAux in conflict water {
49
50      egen    sum_`varAux' = rowtotal(d_`varAux'_m
51      egen    obs_`varAux' = rownonmiss(d_`varAux'
52
53      bys hh_id plot_id:  gen pct_`varAux'_plot =
54      bys hh_id:      gen pct_`varAux' = sum_`varA
55
56    }
57
58    * Keep only one observation per household
59    bys hh_id: gen hh1 = _n
60    bys hh_id: egen pct_water_hh = mean(pct_water_
61    bys hh_id: replace pct_water_hh = . if hh1 > 1
62    replace pct_water_hh = . if d_surveyed_hh == 0
```

```
* Renaming pond variables
    forvalues pondNo = 1/3 {
        forvalues fishNo = 1/6 {
            foreach varname in ish_code ish_name
                _buy _buy_tk _harvest _stage f_h_
                f_s_u f_s_tk f_current f_ct_n f_
                m_consume m_c_n m_c_u m_sold m_s_

                rename pd`pondNo'_f`varname'`fishN

                if "`varname'" == "ish_code" {
                    rename d2_pd`pondNo'_f`fishNo
                    label var d2_pd`pondNo'_f`fish
                    label val d2_pd`pondNo'_f`fish
                }

                *Removing trailing underscores
                if substr("`varname'",length("`var
                    local varname = substr("`varna
                    rename d2_pd`pondNo'_f`fishNo
                }

                *Removing traili
                if substr("`varn
                    local varnam
                    rename d2_pd
                }
            }
        *Droping variables that
        drop d2_pd`pondNo'_fish_
        }
    }

*Dropping variables created when
*incorrectly added a 7th fish to
    drop d2_pd?_fish*7
```

Some text editors will also illustrate this alignment for you.

With comments, it points out the section of code

With logic and loops, it helps find matching braces

```
231  qui gen vargroup = 1
232
233  if `n_vargroups' > 1 {
234    forvalues i = 1/`n_vargroups' {
235      foreach varname in `vargroup_`i'' {
236        qui replace vargroup = `i' if regexm("`vargroup_`i''",varname)
237      }
238    }
239  }
```

Break up long rows of code

One should never have to scroll horizontally to be able to read code

Two recommended ways to break up lines:

1. [///]
   - Everything on the same row will be interpreted as a comment and the following row will be interpreted as if it was the same row
   - Good for breaking a long line of code into a few rows
2. [*#delimit ;*] and [*#delimit cr*]
   - Everything between *#delimit ;* and *#delimit cr* is executed as one line unless it is manually specified using a semicolon
   - Good for breaking a very long line of code into many rows

Example of row breaks

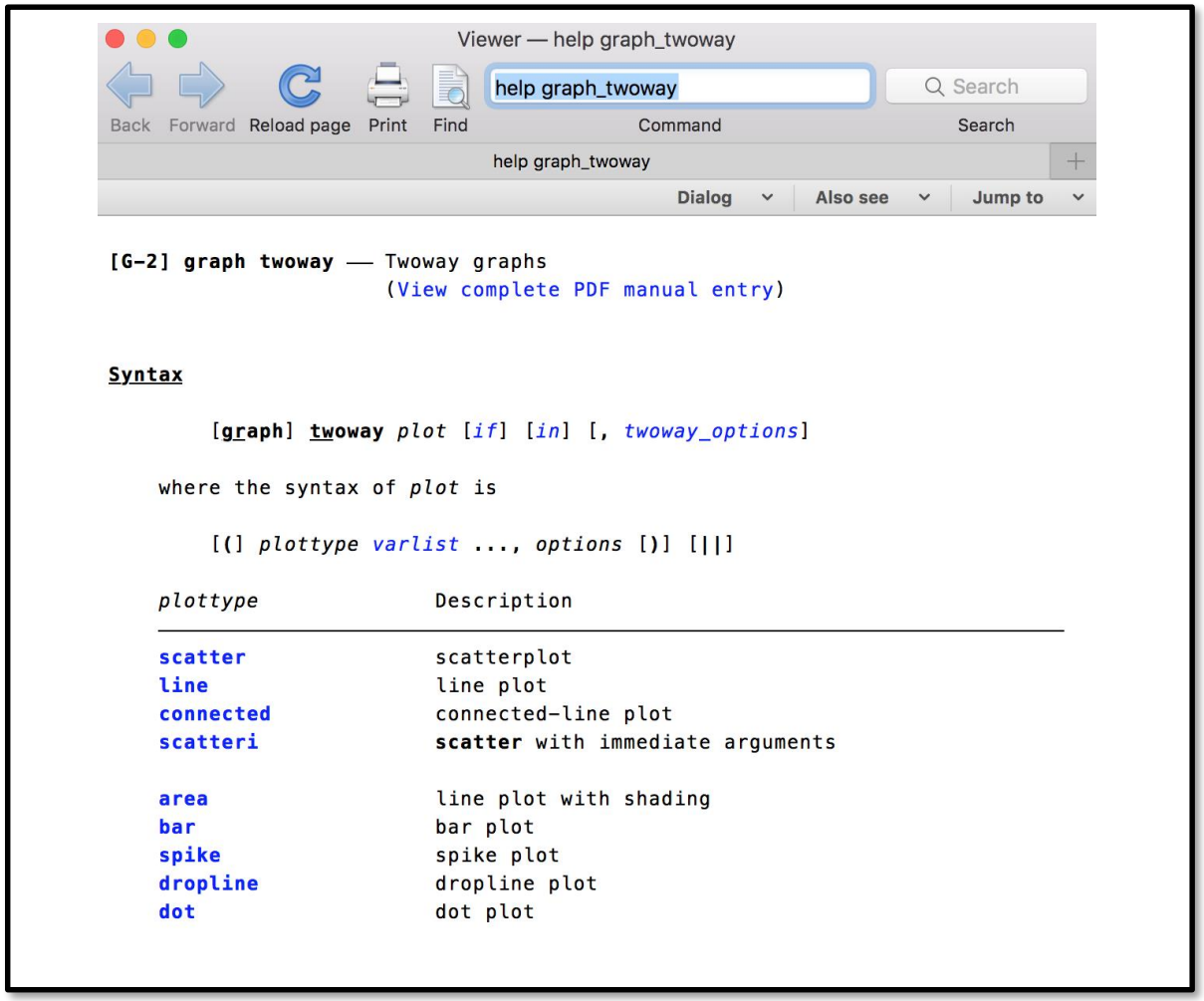
```
local cropcodes     101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116     ///
                    117 118 119 120 121 122 123 124 125 126 127 128 129 130 133 138     ///
                    139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154     ///
                    155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170     ///
                    171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186     ///
                    187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202     ///
                    203 204 205 207 208 209 210 211 212 213 214 215 216 217 218 219     ///
                    220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235     ///
                    236 237 238 239
```

```
* Code to export Graph
#delimit ;

histogram cons_bread, percent normal
  start(0) bin(10)
  bfcolor  ("178 0 80") blcolor("76 0 32")
  ytitle   ("Frequency")
  title    ("Food Security")
  xtitle   ("Number of days")
  subtitle ("Bread Consumption (All Sample)")
  note     ("Includes anyone in the HH who consumed bread last week");
  * Saving graph
  graph save "$outputs/Graph1_bread_consm.gph", replace;

#delimit cr;
```

Line ends with "delimiter"

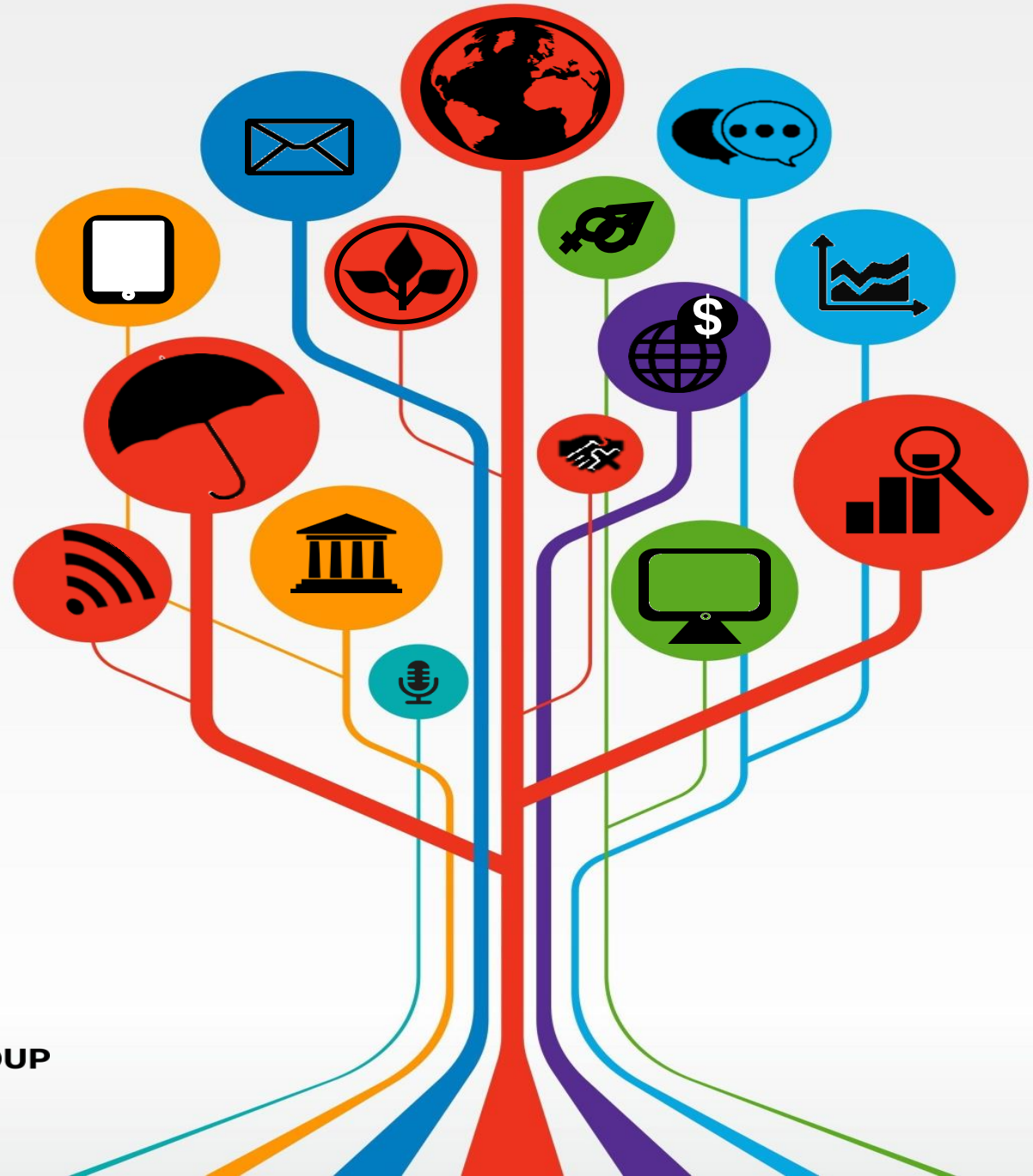
Documentation is your best friend!

- Throughout the rest of the training sessions (and your programming life), you will need [*help*]!
- Type [*help commandname*] in Stata at any time
- Or, google "commandname statalist" for the user-contributed help group
- Let's read this one together:



Resources

[Stata Cheat Sheets]
[SSC Stata Commands]
[UCLA Stata Tutorials]
[UCLA Visual Library]
[Stata Video Library]
[Speaking Stata Library]
[EGAP Methods Guides]



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP

Open access resources from the DIME Analytics team

Implementation Support

Import Commands

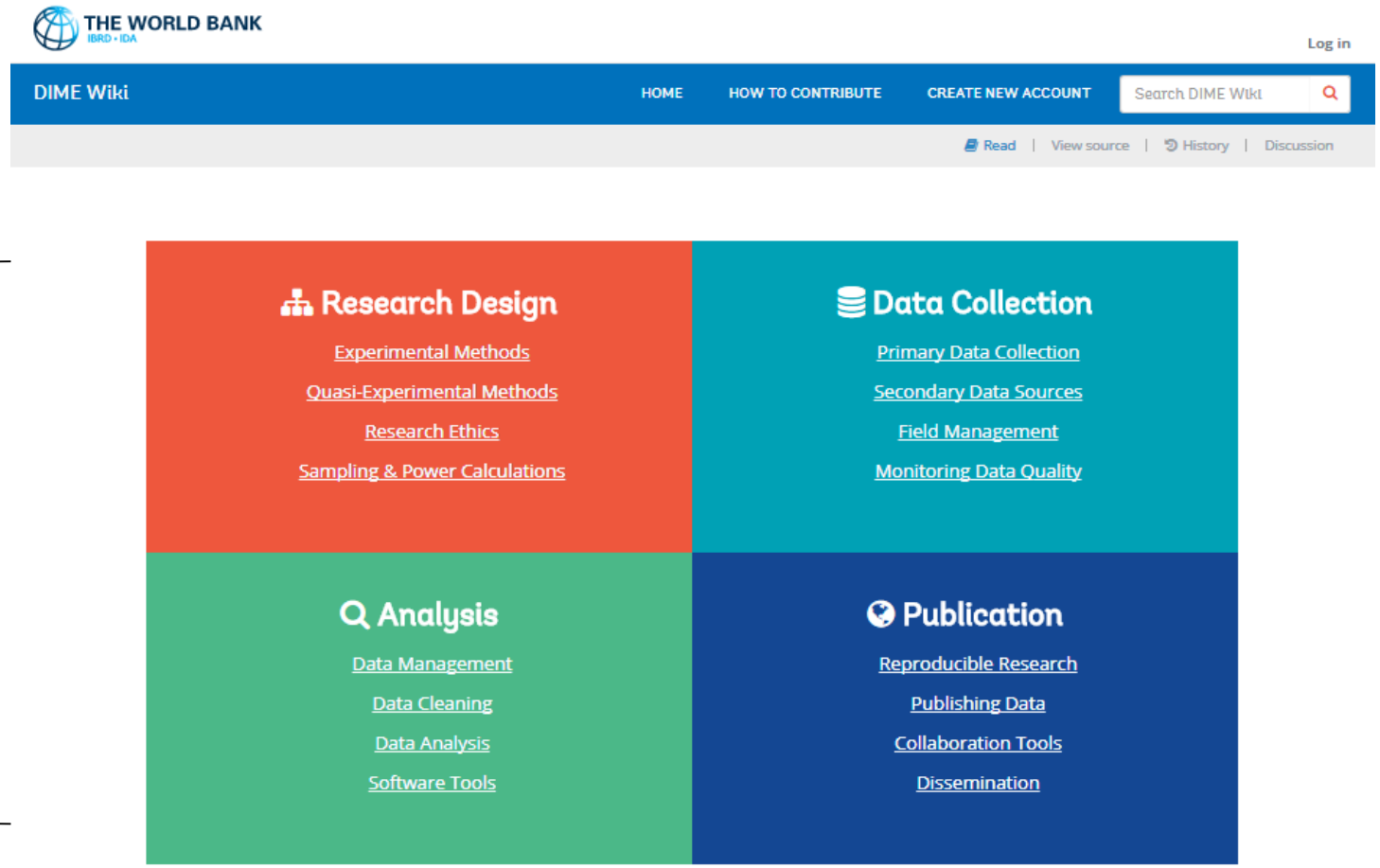
ietoolkit

Stata GitHub

LaTeX trainings

GitHub Trainings

DIME Wiki: structure mirrors open research workflow



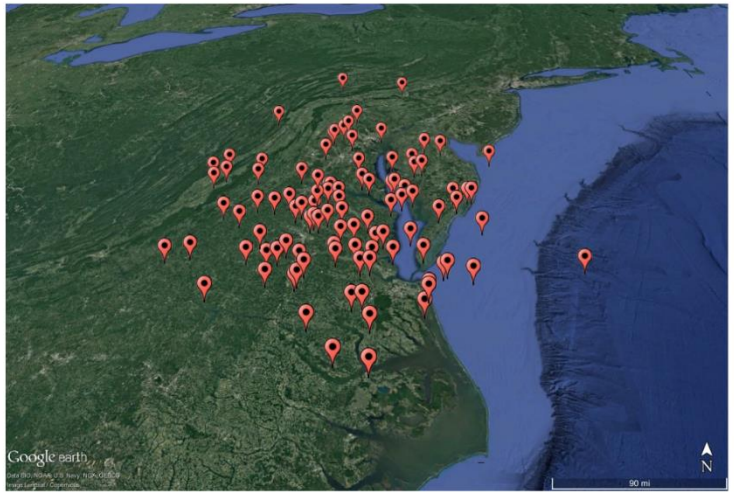
Cross-cutting resources highlight specific tools that support best practices

World Bank GitHub Repos

Code libraries and development repositories on GitHub allow collaborative improvement of software and open-source access to researchers everywhere.

dta2kml

`dta2kml` outputs decimal lat/lon coordinates into a KML file for visual exploration.




```
wb_git_install dta2kml
clear
set obs 100
gen lat = rnormal() +38
gen lon = rnormal() -77
dta2kml using demo.kml , lat(lat) lon(lon) replace
```

stata

Stata Commands for Data Management and Analysis

View the Project on GitHub

World Bank GitHub

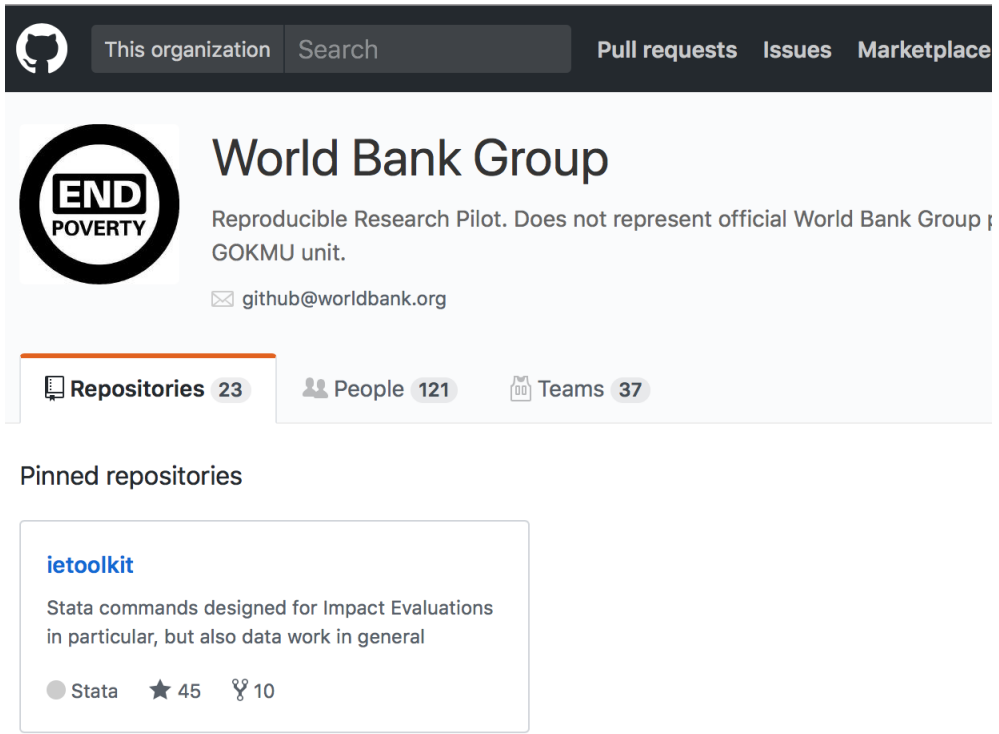
Other World Bank Repositories

- Impact Evaluations Toolkit
- Stata Visual Library
- Distributional Impact Analysis Toolkit

ietoolkit

Stata package routinizing common analytical tasks in IEs

Widely used in DIME, and endorsed by global research community



Reproducible research tools

LaTeX

- Create dynamic documents
- Export and update results transparently, without manual changes

DIME DYNAMIC DOCUMENTS TRAINING
Exercise 1

Luiza Andrade & Mrijan Rimal

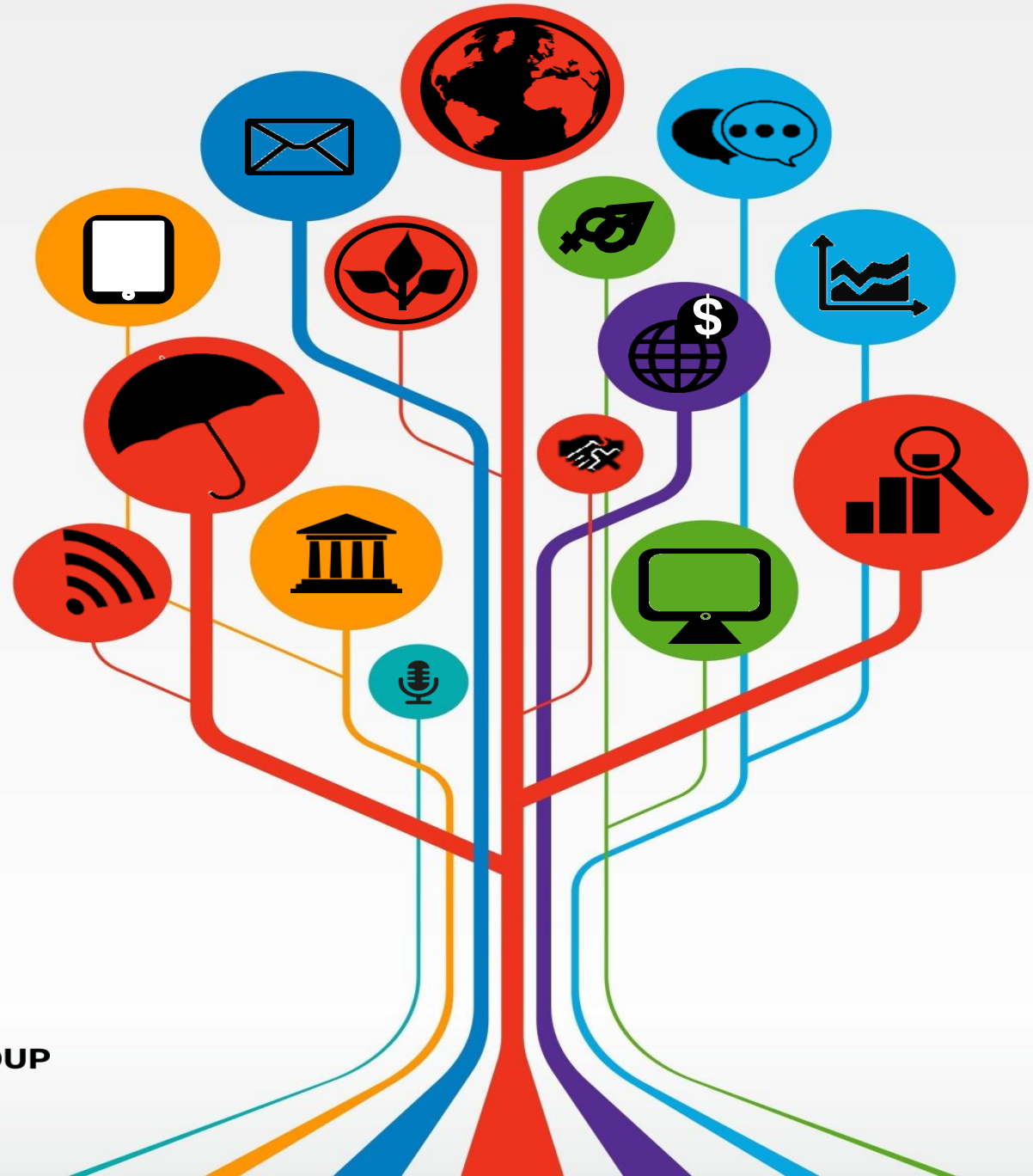
GitHub

- Transparent research with public codes
- Easier collaboration

Using GitHub for code collaboration

November 2, 2017

Thank you!



i2i DIME
TRANSFORM DEVELOPMENT

WORLD BANK GROUP