Reproducibility in Stata

Clare Clingain
November 14th, 2018

https://github.com/CClingain/StataReproducibility

What is reproducibility?

Code + Data

- Code is publicly available
- Can replicate results and check what tests were run
- De-identified data is publicly available
- Data structure

Software

- Open source software!!!!
- Stata, unfortunately, is proprietary, but is extremely common in the real world
- Track changes (Git)

Communication

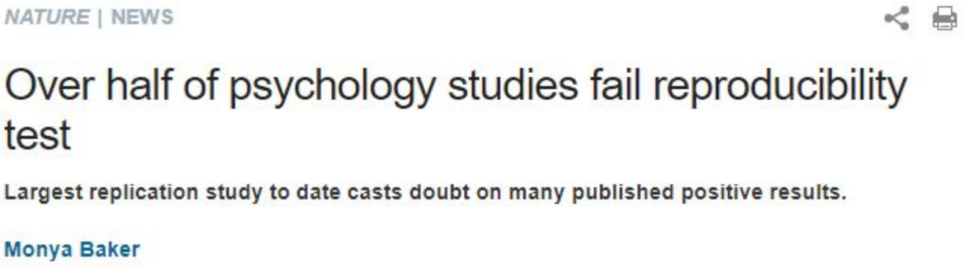
- Data Analysis Plan/Registration
- Peer-review
- Dissemination and Translation

Why should we care?

1. Preserves research integrity
2. Improves research quality
3. Basically lets us do science!
4. Makes your work more understandable to you
5. Makes your work more understandable to others

Reproducibility Crisis

Science
AAAS



NATURE | NEWS

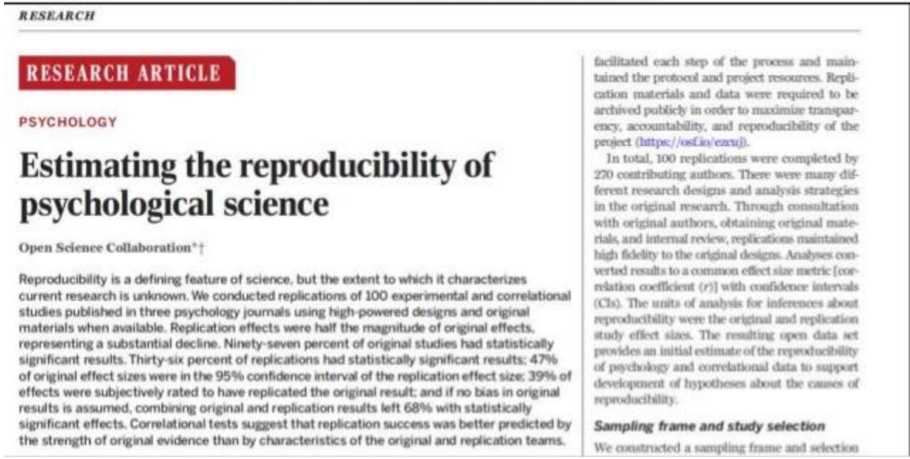
Over half of psychology studies fail reproducibility test

Largest replication study to date casts doubt on many published positive results.

Monya Baker

27 August 2015

The Economist
Britain's angry white men
How to do a nuclear deal with Iran
Investment tips from Nobel economists
Junk bonds are back
The meaning of Sachin Tendulkar
Economist.com
HOW SCIENCE GOES WRONG
99
Einsteinium



RESEARCH

RESEARCH ARTICLE

PSYCHOLOGY

Estimating the reproducibility of psychological science

Open Science Collaboration*†

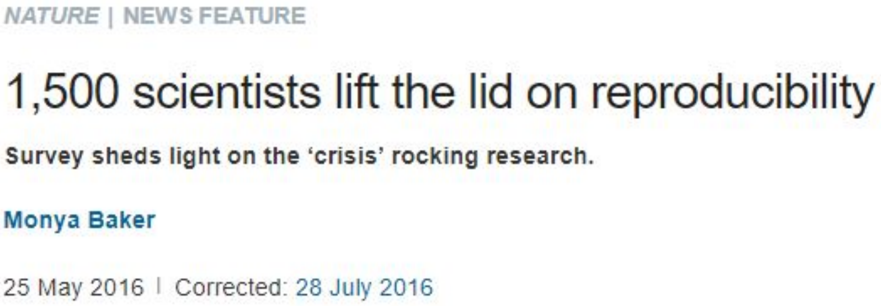
Reproducibility is a defining feature of science, but the extent to which it characterizes current research is unknown. We conducted replications of 100 experimental and correlational studies published in three psychology journals using high-powered designs and original materials when available. Replication effects were half the magnitude of original effects, representing a substantial decline. Ninety-seven percent of original studies had statistically significant results. Thirty-six percent of replications had statistically significant results; 47% of original effect sizes were in the 95% confidence interval of the replication effect size; 39% of effects were subjectively rated to have replicated the original result; and if no bias in original results is assumed, combining original and replication results left 68% with statistically significant effects. Correlational tests suggest that replication success was better predicted by the strength of original evidence than by characteristics of the original and replication teams.

facilitated each step of the process and maintained the protocol and project resources. Replication materials and data were required to be archived publicly in order to maximize transparency, accountability, and reproducibility of the project (https://osf.io/ezcuj).

In total, 100 replications were completed by 270 contributing authors. There were many different research designs and analysis strategies in the original research. Through consultation with original authors, obtaining original materials, and internal review, replications maintained high fidelity to the original designs. Analyses converted results to a common effect size metric [correlation coefficient (r)] with confidence intervals (CIs). The units of analysis for inferences about reproducibility were the original and replication study effect sizes. The resulting open data set provides an initial estimate of the reproducibility of psychology and correlational data to support development of hypotheses about the causes of reproducibility.

Sampling frame and study selection

We constructed a sampling frame and selection



NATURE | NEWS FEATURE

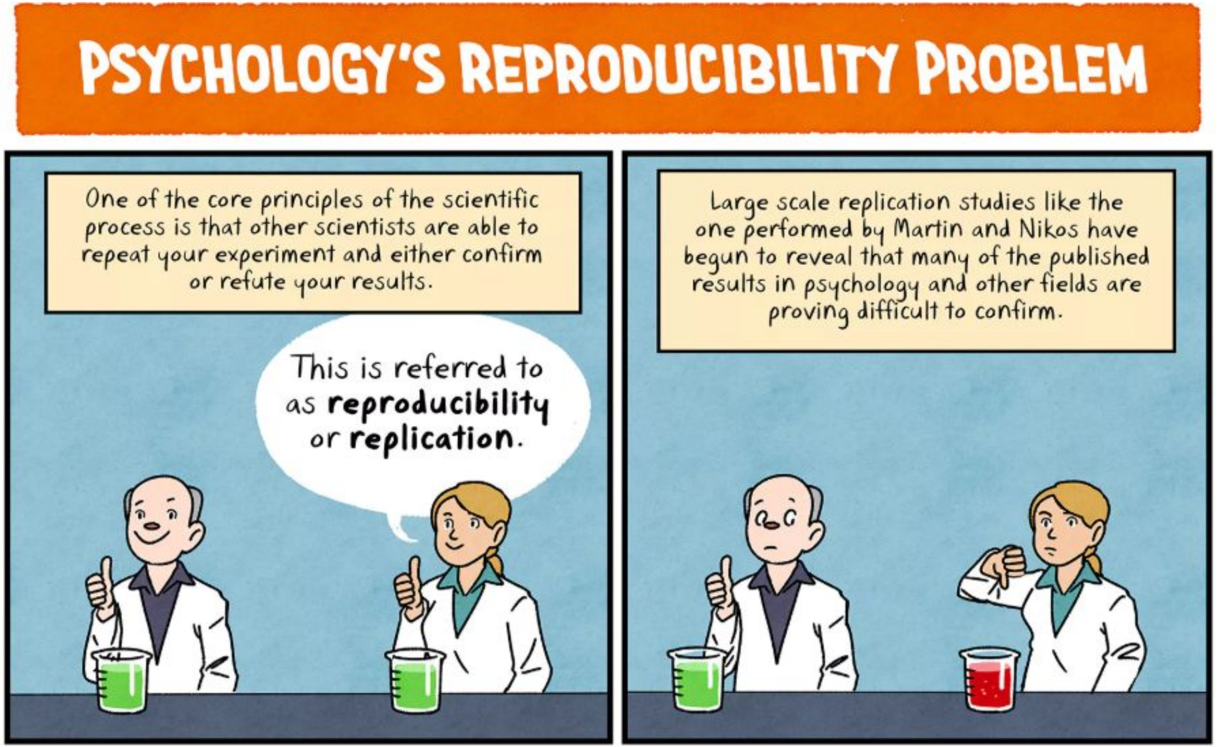
1,500 scientists lift the lid on reproducibility

Survey sheds light on the 'crisis' rocking research.

Monya Baker

25 May 2016 | Corrected: 28 July 2016

Reproducibility vs. Replicability


PSYCHOLOGY'S REPRODUCIBILITY PROBLEM
One of the core principles of the scientific process is that other scientists are able to repeat your experiment and either confirm or refute your results.
This is referred to as **reproducibility** or **replication**.
Large scale replication studies like the one performed by Martin and Nikos have begun to reveal that many of the published results in psychology and other fields are proving difficult to confirm.

Source for all cartoons: The Nib (2016) https://thenib.com/repeat-after-me

Reproducibility:
Same data, same code

Replicability:
Different data, (maybe) different code

Workshop Layout

Soft Coding in Stata

1. Learn how to use local variables
2. Test reproducibility of peers’ code

Export Analysis Results

1. Learn how to export tables to LaTeX
2. Learn how to export tables to Excel

Project Tracking + Management

1. Set up repository in GitKraken
2. Practice Commit/Push/Pull of changes in a Stata do-file
3. Learn how to write effective commit messages

Software Requirements

GitKraken: Windows/Mac
Git Bash: Windows only
Stata: Windows/Mac
MikTeX: Windows only
MacTeX: Mac only

GitKraken Installation

Windows

1. **Install Git Bash**

Download Git: https://gitforwindows.org/

Installation Guide: http://www.techoism.com/how-to-install-git-bash-on-windows/

2. **Install GitKraken**

Download + Installation Guide: https://support.gitkraken.com/how-to-install

Mac

1. **Install GitKraken**

Download + Installation Guide: https://support.gitkraken.com/how-to-install

GitKraken Shortcut

Show hidden folders

File Explorer → View → [] Hidden folders

Find the app

C Drive → Users → Your user → AppData → local → gitkraken → app-4.05 → gitkraken.app

Right click to send to desktop as shortcut

LaTeX Installation

Windows

1. **Install MikTeX**

Download Git: https://miktex.org/download

Installation Guide:
https://miktex.org/howto/install-miktex

Mac

1. **Install MacTeX**

Download + Installation Guide:
https://tug.org/mactex/mactex-download.html

Accessing Stata + Making Stata Do-files

Stata

1. Virtual Computer Lab

NYU Home → Academics → VCL

Link: https://nyu.apporto.com/

2. If you happen to have a copy on your computer… :)

Stata do-files

Since Stata is proprietary and the VCL may be a pain, we will use Notepad or Sublime Text to write our do-files.

1. Notepad or Notes built-in to computers
2. Sublime Text

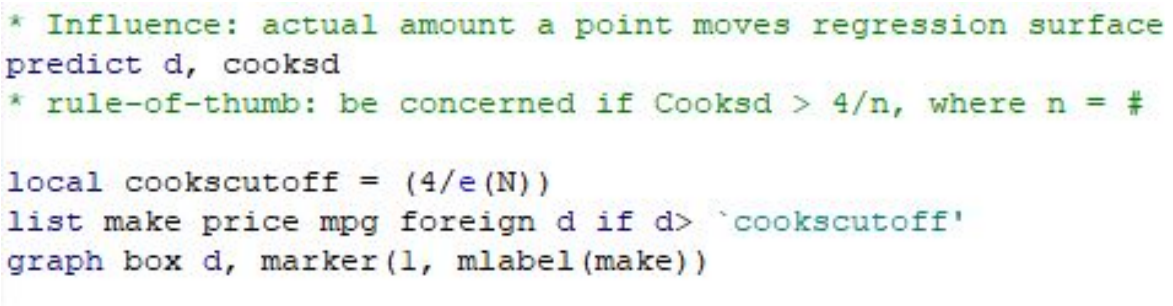
Windows/Mac Download + Installation Guide: https://www.sublimetext.com/3

Note: Sublime text has its advantages for code writing (highlighting syntax) and is easier to format

Soft Coding in Stata

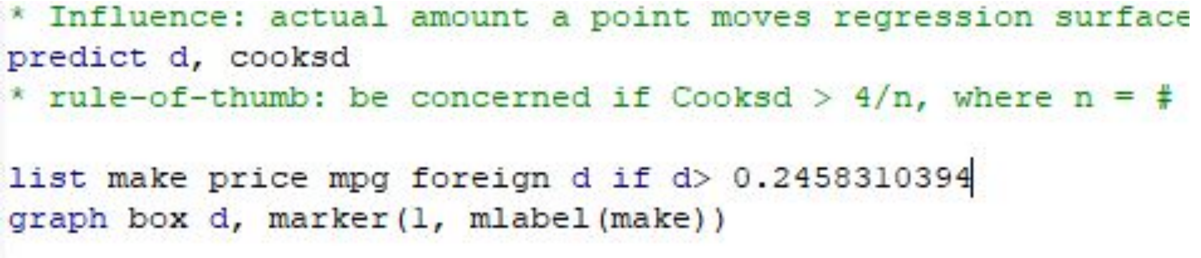
What is soft coding?

Soft coding is when you call on a series of values or variables in a programmatic way.



```
* Influence: actual amount a point moves regression surface
predict d, cooksd
* rule-of-thumb: be concerned if Cooksd > 4/n, where n = #

local cookscutoff = (4/e(N))
list make price mpg foreign d if d> `cookscutoff'
graph box d, marker(1, mlabel(make))
```

Hard coding is when you write in the value or variable by hand.



```
* Influence: actual amount a point moves regression surface
predict d, cooksd
* rule-of-thumb: be concerned if Cooksd > 4/n, where n = #

list make price mpg foreign d if d> 0.2458310394
graph box d, marker(1, mlabel(make))
```

Local Variables

- Local variables are defined in Stata by the **local** command
- You can call on a local variable you have made using Stata quotes `myvar’

```
summ age

local agemean = r(mean)

di `agemean’
```

- NOTE: any code that calls on a local variable must be **run at the same time** as the **local** command code that creates the variable!
- Otherwise, Stata “forgets” the local variable exists

What can be a local variable?

- A number

```
local meanage = r(mean)
```

- A string

```
local mytitle = “Demographic breakdown by county”
```

- A letter

```
local myexcelcol = “A”
```

- A list of variables

```
local demogvars age race sex ses
```

How to display a local variable

- If your local variable is a number

  di `meanage’

- If your local variable is a character

  di “`mytitle’”

- If your local variable is a list
  - Calling on the variable in code: `myvarlist’
  - Simply viewing the list as a string: di “`myvarlist’”

Create a “Example.do/txt/stmd” file

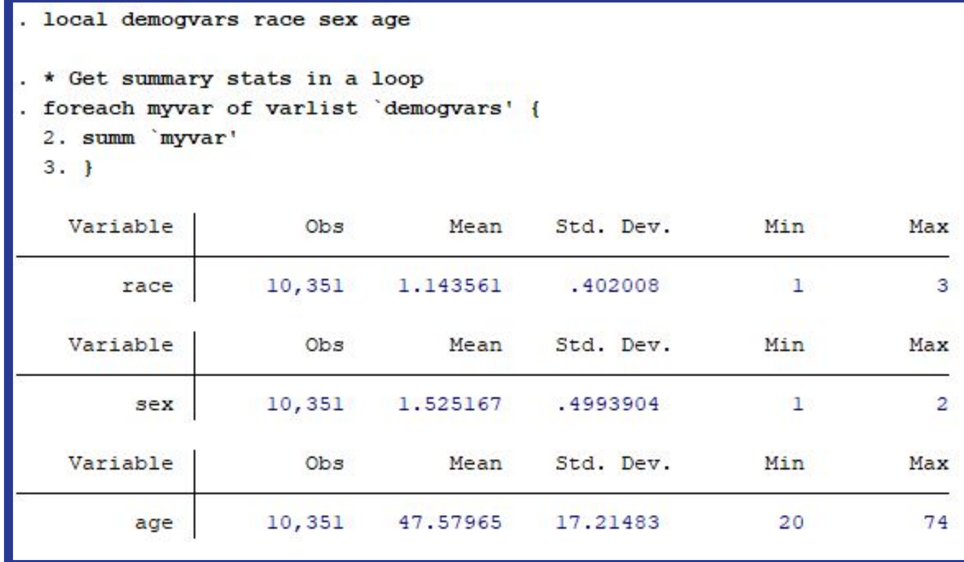
Add the code in the blue box

Note: update the quotes if you copy/paste!

```
* Load in the data

use
“http://www.stata-press.com/data/r14/nhanes2d.dta”, clear

codebook

* Create a local variable of demographics

local demogvars race sex age

* Get summary stats in a loop

foreach myvar of varlist `demogvars' {

summ `myvar'

}
```

Run the code in Stata

Note: if the code fails, check your quotes



```
. local demogvars race sex age

. * Get summary stats in a loop
. foreach myvar of varlist `demogvars' {
  2. summ `myvar'
  3. }

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        race |     10,351    1.143561     .402008          1          3

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         sex |     10,351    1.525167    .4993904          1          2

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         age |     10,351    47.57965    17.21483         20         74
```

Why go to all this trouble of local variables?

- Someone else can run your code
- Someone else can adapt your code to their data
- You can run your code if you use a new sample/subsample
- Other people can understand your code
  - Traceback for code process

So your code is REPRODUCIBLE

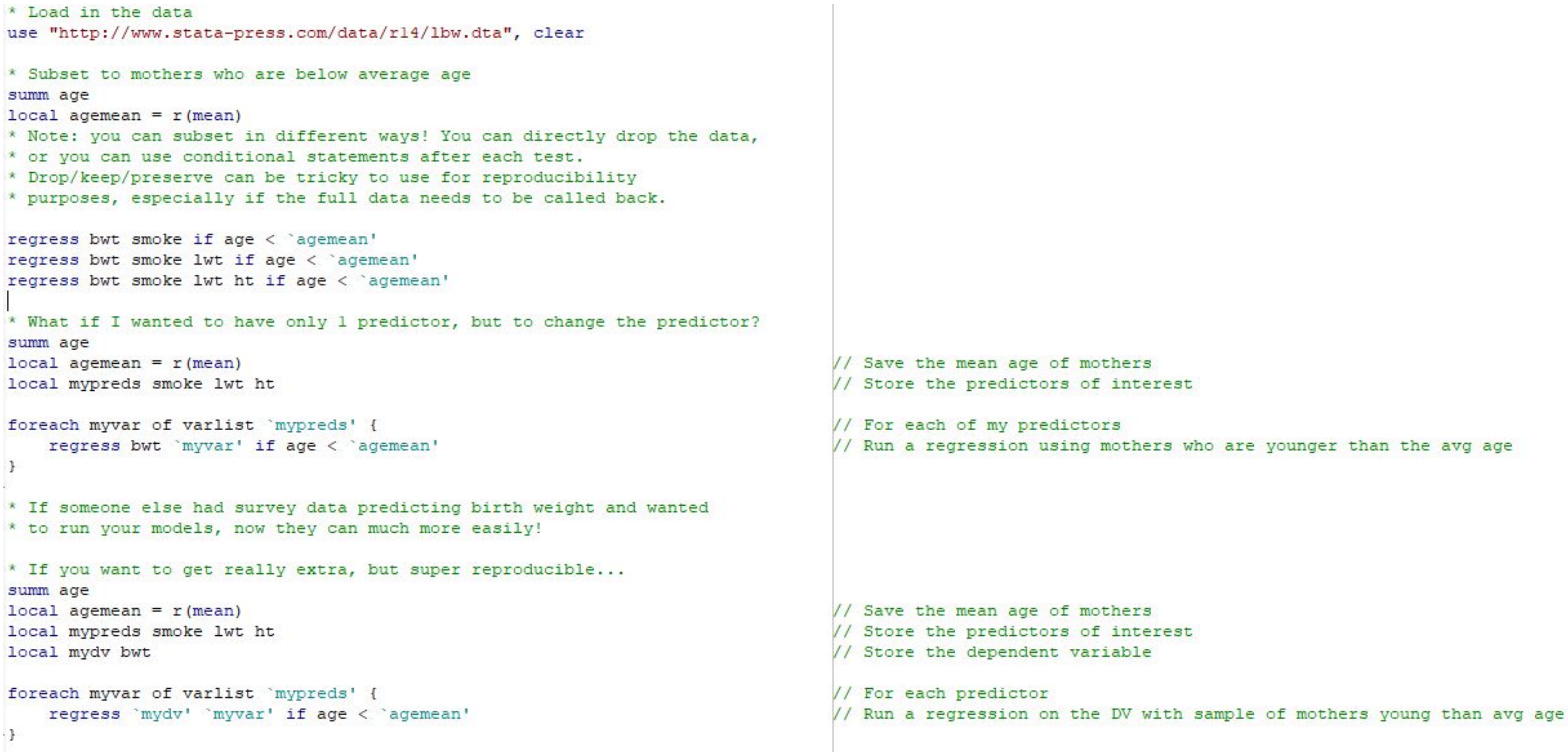
Practice!

If you are in an even row:

1. Use "http://www.stata-press.com/data/r14/lbw.dta", clear
2. Subset to mothers that are below the average age **using a local variable**
3. Run three regressions predicting birth weight
4. Switch code with your neighbor

If you are in an odd row:

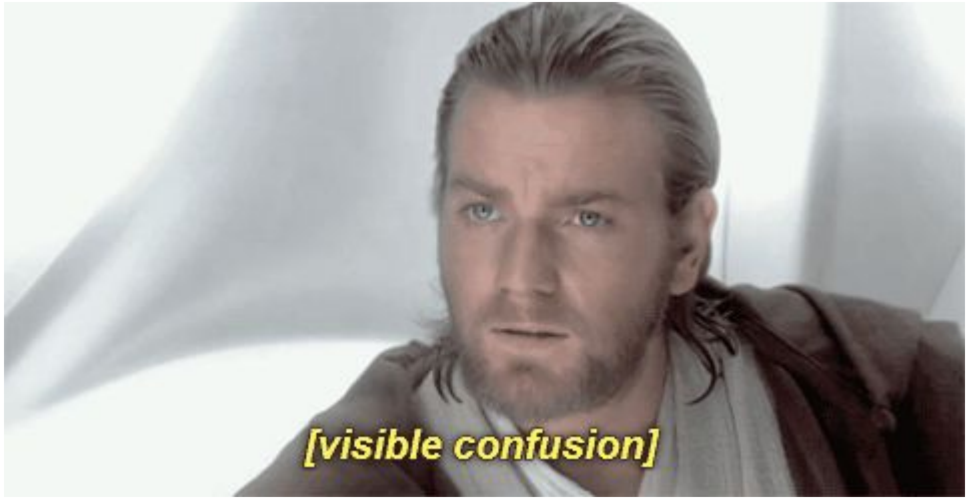
1. Use "http://www.stata-press.com/data/r14/lbw.dta", clear
2. Subset to mothers that are above the average age **using a local variable**
3. Run three regressions predicting birth weight
4. Switch code with your neighbor!



```
* Load in the data
use "http://www.stata-press.com/data/r14/lbw.dta", clear

* Subset to mothers who are below average age
summ age
local agemean = r(mean)
* Note: you can subset in different ways! You can directly drop the data,
* or you can use conditional statements after each test.
* Drop/keep/preserve can be tricky to use for reproducibility
* purposes, especially if the full data needs to be called back.

regress bwt smoke if age < `agemean'
regress bwt smoke lwt if age < `agemean'
regress bwt smoke lwt ht if age < `agemean'

* What if I wanted to have only 1 predictor, but to change the predictor?
summ age
local agemean = r(mean)                                                         // Save the mean age of mothers
local mypreds smoke lwt ht                                                      // Store the predictors of interest

foreach myvar of varlist `mypreds' {                                            // For each of my predictors
    regress bwt `myvar' if age < `agemean'                                      // Run a regression using mothers who are younger than the avg age
}

* If someone else had survey data predicting birth weight and wanted
* to run your models, now they can much more easily!

* If you want to get really extra, but super reproducible...
summ age
local agemean = r(mean)                                                         // Save the mean age of mothers
local mypreds smoke lwt ht                                                      // Store the predictors of interest
local mydv bwt                                                                  // Store the dependent variable

foreach myvar of varlist `mypreds' {                                            // For each predictor
    regress `mydv' `myvar' if age < `agemean'                                   // Run a regression on the DV with sample of mothers young than avg age
}
```

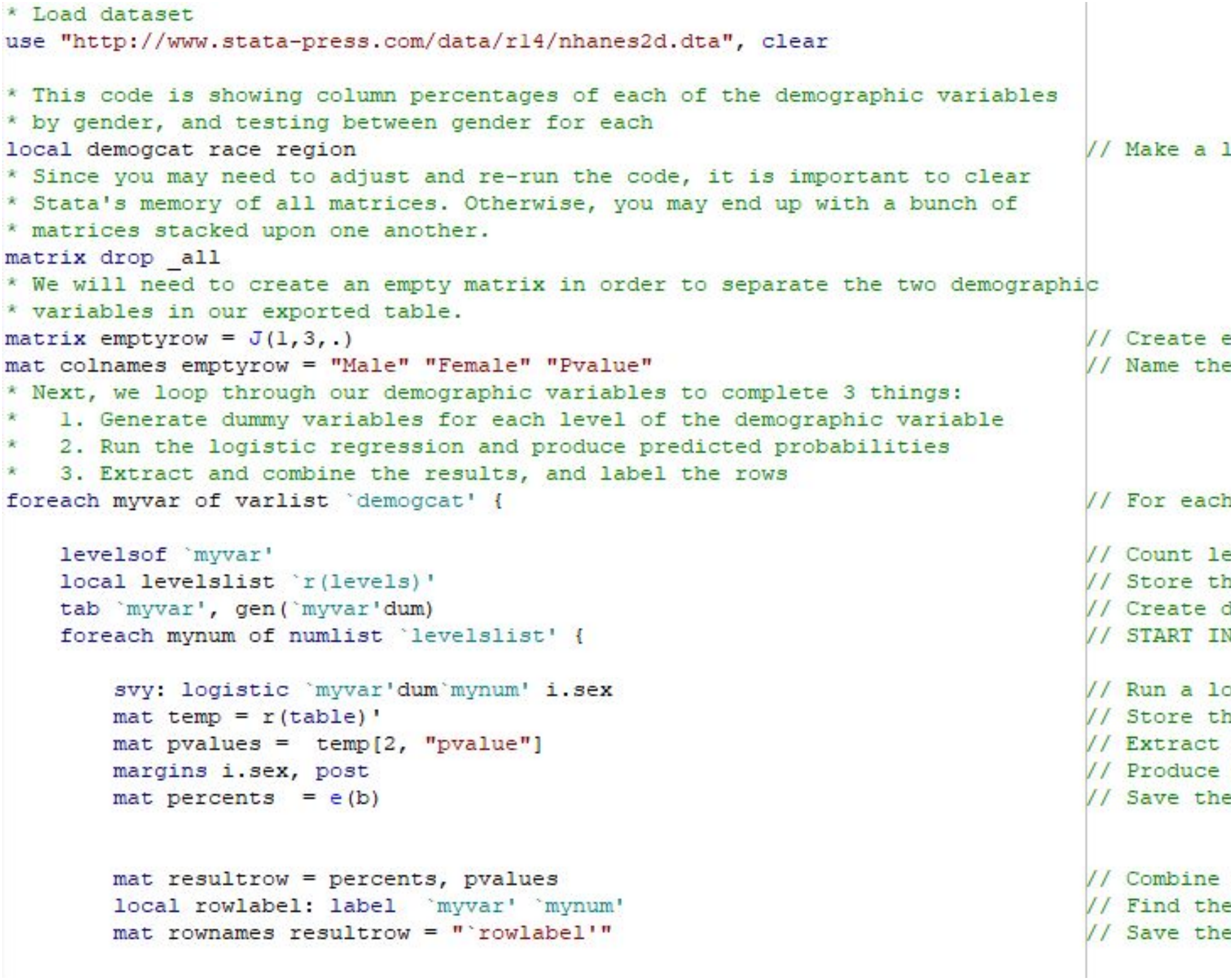
Thoughts?

This is a pretty simple example, but can you imagine if I gave you different data sets that had different variable names and different means...but I wanted the same models with the same subsample and the results to be saved in the same format?



Making your whole do-file reproducible

- Use local variables in for loops
  - List dependent and independent variables as local
  - Loop through each DV and IV to run your regression
  - Save the results!!!!!



```
* Load dataset
use "http://www.stata-press.com/data/r14/nhanes2d.dta", clear

* This code is showing column percentages of each of the demographic variables
* by gender, and testing between gender for each
local demogcat race region                                                      // Make a l
* Since you may need to adjust and re-run the code, it is important to clear
* Stata's memory of all matrices. Otherwise, you may end up with a bunch of
* matrices stacked upon one another.
matrix drop _all
* We will need to create an empty matrix in order to separate the two demographic
* variables in our exported table.
matrix emptyrow = J(1,3,.)                                                      // Create e
mat colnames emptyrow = "Male" "Female" "Pvalue"                                // Name the
* Next, we loop through our demographic variables to complete 3 things:
*   1. Generate dummy variables for each level of the demographic variable
*   2. Run the logistic regression and produce predicted probabilities
*   3. Extract and combine the results, and label the rows
foreach myvar of varlist `demogcat' {                                           // For each

    levelsof `myvar'                                                            // Count le
    local levelslist `r(levels)'                                                // Store th
    tab `myvar', gen(`myvar'dum)                                                // Create d
    foreach mynum of numlist `levelslist' {                                     // START IN

        svy: logistic `myvar'dum`mynum' i.sex                                   // Run a lo
        mat temp = r(table)'                                                    // Store th
        mat pvalues =  temp[2, "pvalue"]                                        // Extract
        margins i.sex, post                                                     // Produce
        mat percents  = e(b)                                                    // Save the


        mat resultrow = percents, pvalues                                       // Combine
        local rowlabel: label  `myvar' `mynum'                                  // Find the
        mat rownames resultrow = "`rowlabel'"                                   // Save the
```

Comment your code!

- Whenever you use local variables, make sure you comment what you are doing so that others can understand (and so that you can understand in the future)
- Comments are particularly important when you start using local variables in for loops
  - Keep track of nested levels
- Comments can be used to indicate which lines of code have to be changed for someone else to run your code
  - Generally, this is just **1 or 2 lines!**

A few comments

- Using local variables may not feel natural
  - We learn Stata via hard-coded commands with a sprinkling of local variables
- Local variables are a **must** for dealing with text data in Stata
- Helpful for internal use when you constantly run the same analyses on different data

Export Analysis Results

Saving results as tables

Excel + LaTeX

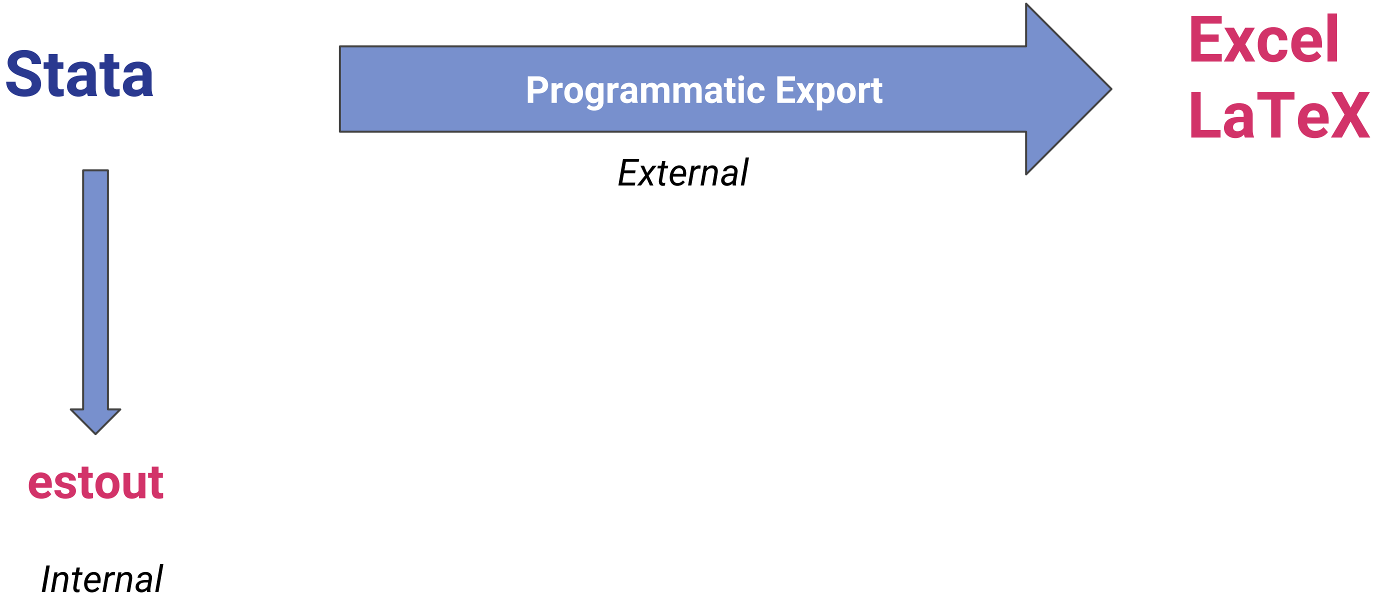
The old days...

Why???

- No traceback to how you got a number
- Human error
- Time consuming

So that this is not your reaction when a referee, colleague, anyone asks you where you got that number and you can't find it anywhere in your output

Best practices...



Stata

Programmatic Export

External

Excel

LaTeX

estout

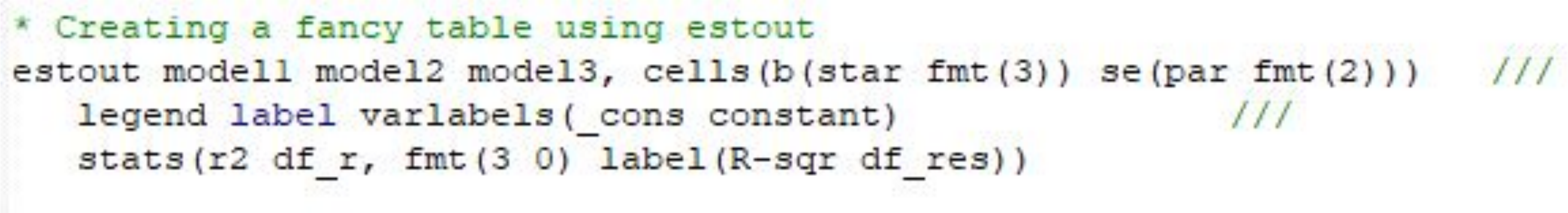
Internal

Quick Comparison

	estout	**putexcel**	**texdoc**
Built in to Stata?	✘	✓	✘
Requires another program?	✘	✓	✓
Easy to use?	✓	✓	✘
Coding intensive?	✘	✓	✓

Estout -- a handy Stata package

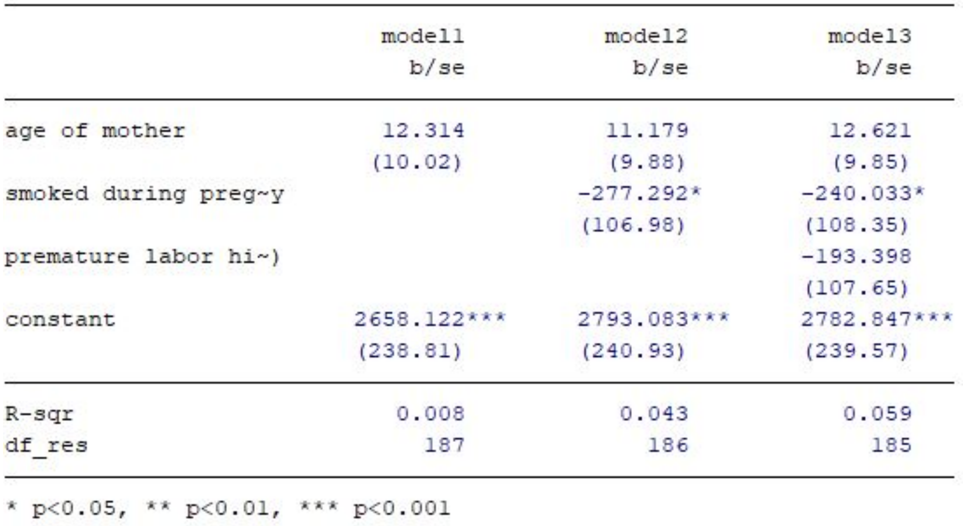
- Prints out a table within your Stata output
- Requires that you store results after each model
    - regress bwt age
    - est store [*name the model*]



```
* Creating a fancy table using estout
estout model1 model2 model3, cells(b(star fmt(3)) se(par fmt(2)))   ///
   legend label varlabels(_cons constant)                   ///
   stats(r2 df_r, fmt(3 0) label(R-sqr df_res))
```

Estout table

Low birth weight data



```
--------------------------------------------------------------------
                          model1          model2          model3
                            b/se            b/se            b/se
--------------------------------------------------------------------
age of mother             12.314          11.179          12.621
                         (10.02)          (9.88)          (9.85)
smoked during preg~y                    -277.292*       -240.033*
                                        (106.98)        (108.35)
premature labor hi~)                                    -193.398
                                                        (107.65)
constant                2658.122***     2793.083***     2782.847***
                        (238.81)        (240.93)        (239.57)
--------------------------------------------------------------------
R-sqr                      0.008           0.043           0.059
df_res                       187             186             185
--------------------------------------------------------------------
* p<0.05, ** p<0.01, *** p<0.001
```

Note: code can be found on Github!

putexcel: Package linking Stata to Excel

Command to initialize export location:

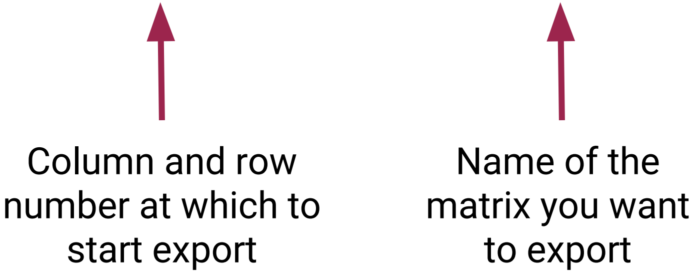
putexcel set “StataRepro/StataReproducibility/Export_Tables.xlsx”, sheet (“Regression Table 1”) modify

If the Excel file doesn’t exist, putexcel creates it for you!

putexcel: Package linking Stata to Excel

Command to export data:

putexcel A1 = matrix(*matrix_name*)



Column and row number at which to start export

Name of the matrix you want to export

Hint: Column and row can be specified with local variables

Practice: Exporting a simple regression

Excel sheet must be CLOSED for code to run

1. Run your regression command (**regress bwt smoke**)
2. Look at the **return list**
3. Save the matrix *r(table)* as the following:
   a. **mat temp = r(table)’**
   b. Note: we need to transpose the matrix so that the data is read column-wise
4. Display the matrix (**mat list temp**)
5. Set working directory to your repository
   a. **cd “C:/Users/Clare/Documents/StataReproducibility”**
6. Send to your Excel sheet:
   a. **putexcel set "Export_Tables.xlsx", sheet("Regression Table 1") modify**
   b. **putexcel A1 = matrix(temp)**

Wait -- this looks ugly!

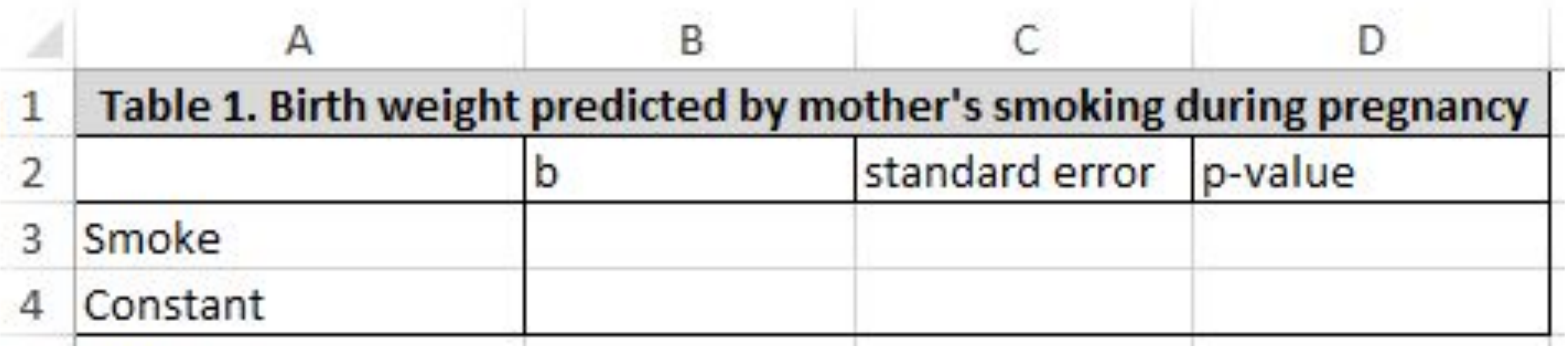
- This is what putexcel will give you if you do absolutely no formatting before exporting your matrix

Let’s try this again

```
putexcel A5 = matrix(temp), rownames
```

***Make sure you close the Excel sheet before running command**

Building a table shell



	A	B	C	D
1	**Table 1. Birth weight predicted by mother's smoking during pregnancy**			
2		b	standard error	p-value
3	Smoke			
4	Constant			

How do I get only the data that I want for my table shell?

Aside: Matrix Manipulation in Stata

How to extract all the rows/columns of a matrix

1… means all of the values starting at the first

How to extract a column from a matrix

mat pvalues = temp[1…,**”pvalue”**]

How to extract up to a certain row/column of a matrix

mat bweights = temp[1…, **1..”se”**]

Exporting to a table shell

```
* Set the export sheet to the table shell sheet
putexcel set “Export_Tables.xlsx”, sheet(“Regression Shell Table 1”) modify
* Run the regression
regress bwt smoke
* Get the initial results
mat temp = r(table)’
* Extract b-weights and standard errors
mat bweights = temp[1...,1.."se"]
* Extract p-values
mat pvalues =  temp[1…,”pvalue”]
* Save as one big matrix
mat final = bweights , pvalues
* Export to the correct cell
putexcel B3 = matrix(final)
```

Table Shell Export: Results

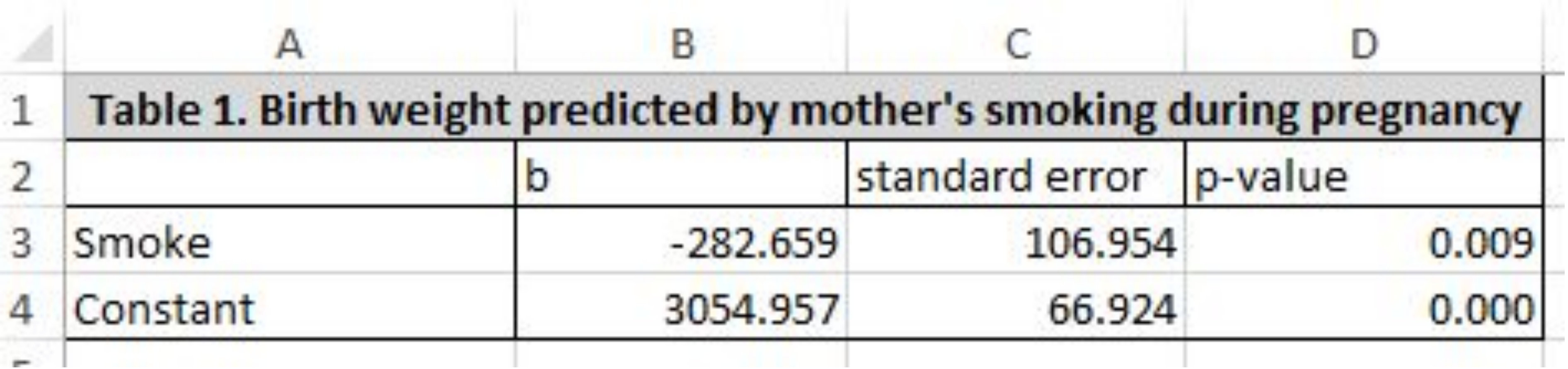


Table 1. Birth weight predicted by mother's smoking during pregnancy

	b	standard error	p-value
Smoke	-282.659	106.954	0.009
Constant	3054.957	66.924	0.000

Note: you can set Excel to display *x* decimal places, but still retain the information

Good practice

- Add the date and time of last export to your code
- Choose an arbitrary cell that is nowhere near your data, but is accessible to scroll to

```
putexcel A30=("$S_TIME  $S_DATE")
```

- Have a section of your do-file just for exporting

Exporting more than one model

Low birth weight data

Nested Models

Model 1: regress bwt smoke

Model 2: regress bwt smoke ht

Multiple Models

1. Build the table shell
2. Write the code
3. Export to the table shell

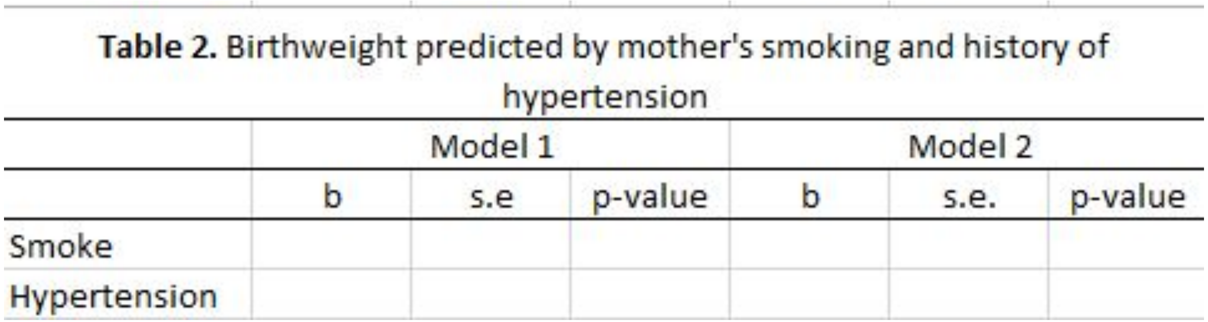


Table 2. Birthweight predicted by mother's smoking and history of hypertension

	Model 1			Model 2		
	b	s.e	p-value	b	s.e.	p-value
Smoke						
Hypertension						

Tip: making the table shell first can help you visualize how you need to write the code

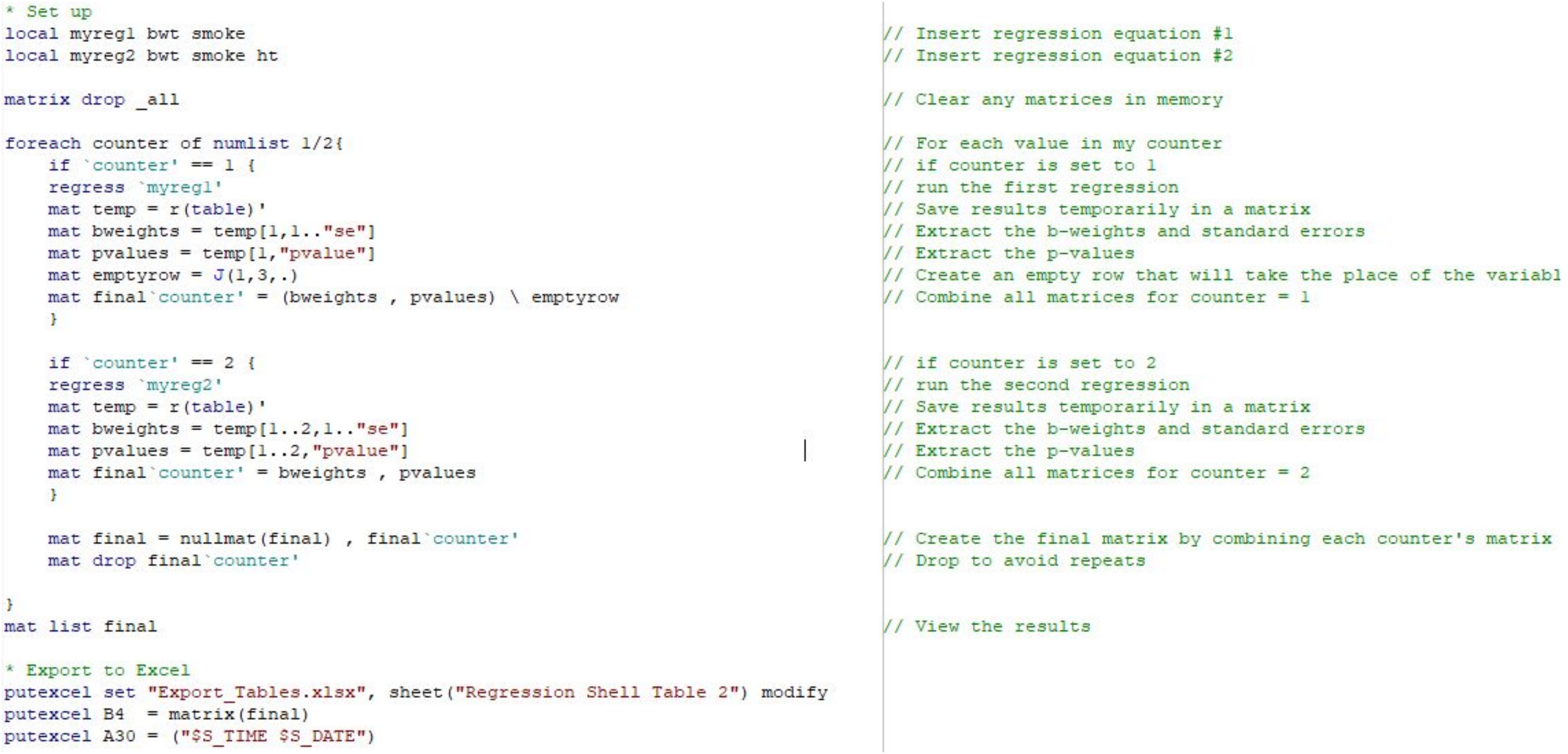
Easiest way to think about this...

- Save our two regression equations as locals
  - local myreg1 bwt smoke
  - local myreg2 bwt smoke ht
- Create a counter such that
  - At value 1, we want to run our first regression
  - At value 2, we want to run our second regression
- At each value we want to
  - Extract the b-weights and p-values for the variables
- Combine the two results matrices into one matrix for exporting

Aside: For loops in Stata

3 different loops:

```
foreach myvar of varlist `demogvar’ { do something }

foreach mynum of numlist 1/2 { do something }

forval i = 1/2 { do something }
```



```
* Set up
local myreg1 bwt smoke                                                          // Insert regression equation #1
local myreg2 bwt smoke ht                                                       // Insert regression equation #2

matrix drop _all                                                                // Clear any matrices in memory

foreach counter of numlist 1/2{                                                 // For each value in my counter
    if `counter' == 1 {                                                         // if counter is set to 1
    regress `myreg1'                                                            // run the first regression
    mat temp = r(table)'                                                        // Save results temporarily in a matrix
    mat bweights = temp[1,1.."se"]                                              // Extract the b-weights and standard errors
    mat pvalues = temp[1,"pvalue"]                                              // Extract the p-values
    mat emptyrow = J(1,3,.)                                                     // Create an empty row that will take the place of the variabl
    mat final`counter' = (bweights , pvalues) \ emptyrow                        // Combine all matrices for counter = 1
    }

    if `counter' == 2 {                                                         // if counter is set to 2
    regress `myreg2'                                                            // run the second regression
    mat temp = r(table)'                                                        // Save results temporarily in a matrix
    mat bweights = temp[1..2,1.."se"]                                           // Extract the b-weights and standard errors
    mat pvalues = temp[1..2,"pvalue"]                                           // Extract the p-values
    mat final`counter' = bweights , pvalues                                     // Combine all matrices for counter = 2
    }

    mat final = nullmat(final) , final`counter'                                 // Create the final matrix by combining each counter's matrix
    mat drop final`counter'                                                     // Drop to avoid repeats

}
mat list final                                                                  // View the results

* Export to Excel
putexcel set "Export_Tables.xlsx", sheet("Regression Shell Table 2") modify
putexcel B4  = matrix(final)
putexcel A30 = ("$S_TIME $S_DATE")
```

Nested table shell results

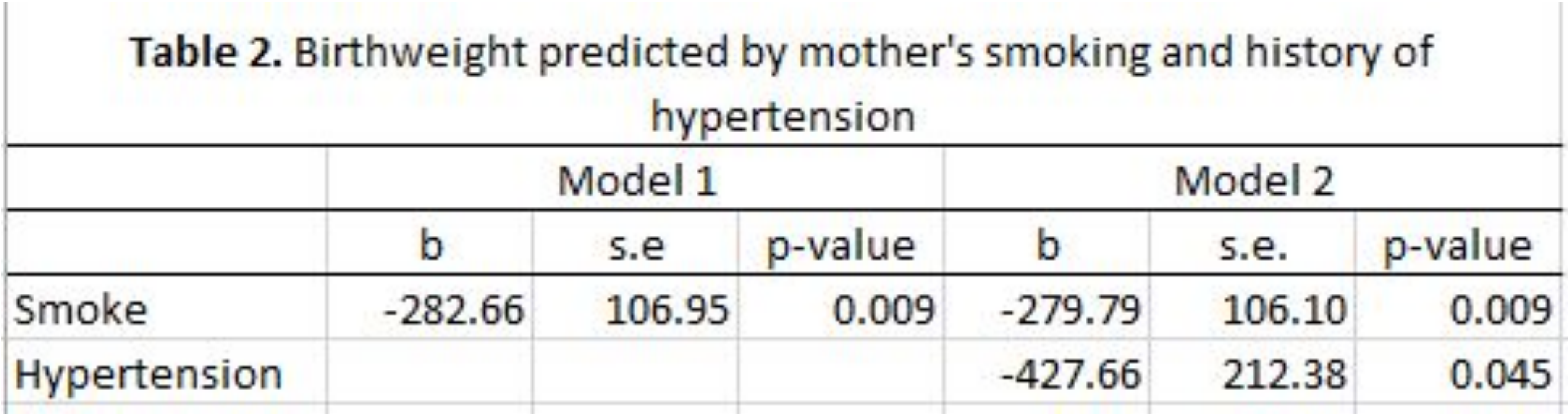


Table 2. Birthweight predicted by mother's smoking and history of hypertension

	Model 1			Model 2		
	b	s.e	p-value	b	s.e.	p-value
Smoke	-282.66	106.95	0.009	-279.79	106.10	0.009
Hypertension				-427.66	212.38	0.045

Note: if you want spaces between variables in the table, you can add empty rows to your matrix

texdoc: Package linking Stata to LaTeX

Install texdoc:

```
capture which texdoc

if _rc==111 ssc install texdoc.pkg
```

texdoc: Package linking Stata to LaTeX

Command to initialize export location:

```
texdoc init Textable, replace
```

If the LaTeX file doesn’t exist, texdoc creates it for you!

texdoc: Package linking Stata to LaTeX

Commands to set LaTeX formatting:

```
tex \documentclass{article}

tex \usepackage{stata}

tex \begin{document}

tex \section{Table 1}
```

texdoc: Package linking Stata to LaTeX

Commands to add Stata output to LaTeX:

```
…
tex \section{Table 1}
texdoc stlog texlog
regress bwt smoke
texdoc stlog close
tex \end{document}
texdoc close
```

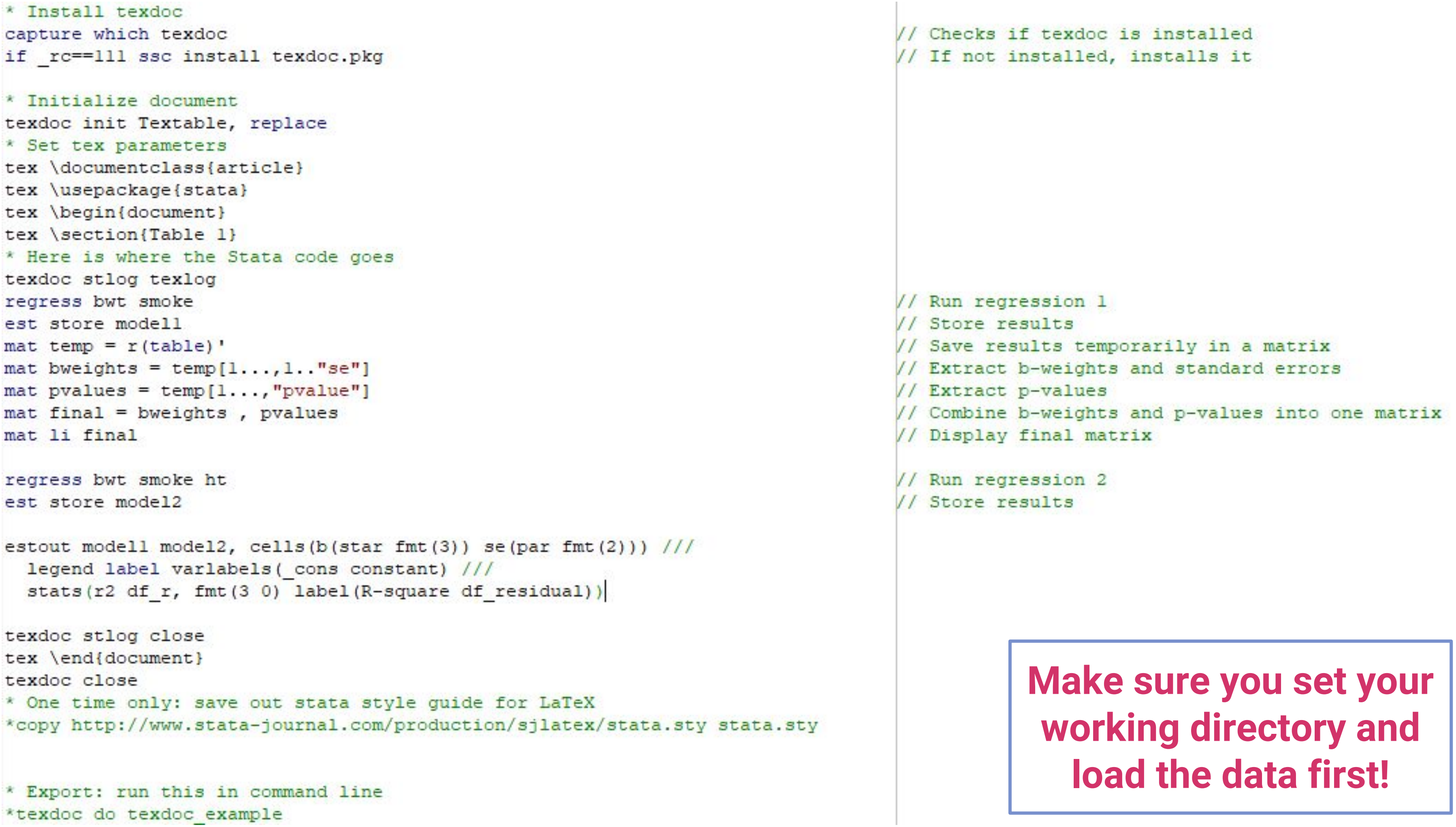
texdoc: Package linking Stata to LaTeX

Commands to export to LaTeX:

```
texdoc do texdoc_example
```

***This will create the Tex file to create the output and tables**

*** MUST be in separate doc or run in command line**



```
* Install texdoc
capture which texdoc                                     // Checks if texdoc is installed
if _rc==111 ssc install texdoc.pkg                       // If not installed, installs it

* Initialize document
texdoc init Textable, replace
* Set tex parameters
tex \documentclass{article}
tex \usepackage{stata}
tex \begin{document}
tex \section{Table 1}
* Here is where the Stata code goes
texdoc stlog texlog
regress bwt smoke                                        // Run regression 1
est store model1                                         // Store results
mat temp = r(table)'                                     // Save results temporarily in a matrix
mat bweights = temp[1...,1.."se"]                        // Extract b-weights and standard errors
mat pvalues = temp[1...,"pvalue"]                        // Extract p-values
mat final = bweights , pvalues                           // Combine b-weights and p-values into one matrix
mat li final                                             // Display final matrix

regress bwt smoke ht                                     // Run regression 2
est store model2                                         // Store results

estout model1 model2, cells(b(star fmt(3)) se(par fmt(2))) ///
  legend label varlabels(_cons constant) ///
  stats(r2 df_r, fmt(3 0) label(R-square df_residual))

texdoc stlog close
tex \end{document}
texdoc close
* One time only: save out stata style guide for LaTeX
*copy http://www.stata-journal.com/production/sjlatex/stata.sty stata.sty


* Export: run this in command line
*texdoc do texdoc_example
```

Make sure you set your working directory and load the data first!

Estout and texdoc

- Use in conjunction to send estout output to LaTeX

```
esttab using example.tex, label nostar title(Regression table\label{tab1})
```

Creates a new file
with this name

Note: must have stored models first

Note 2: Must add \documentclass and \begin + \end commands

Estout + texdoc presents

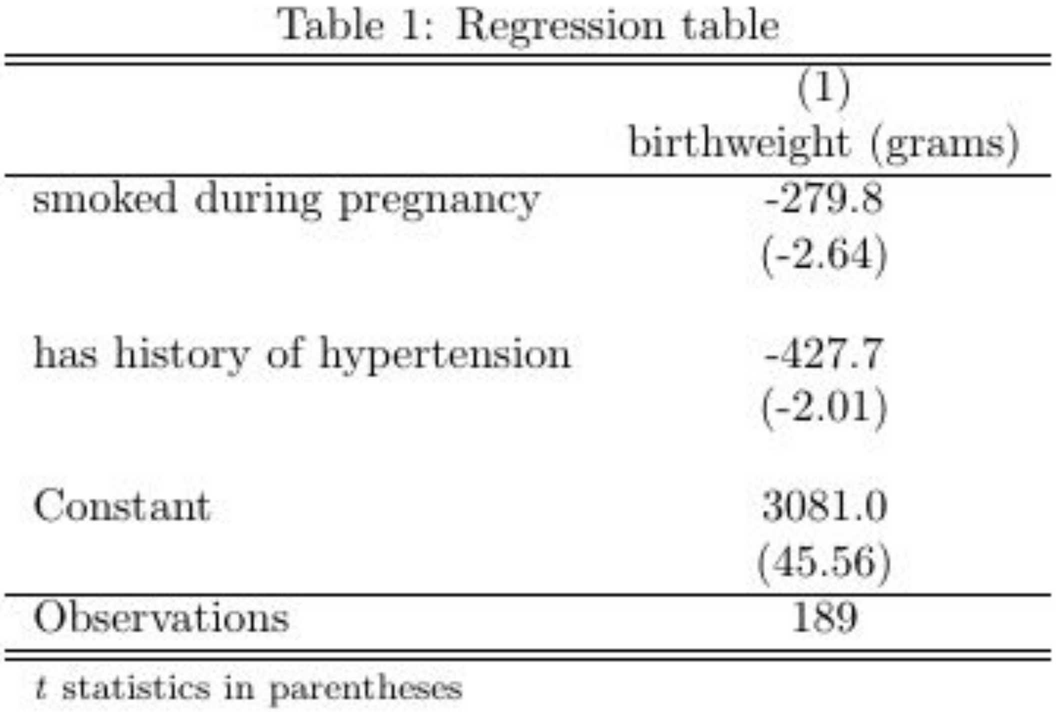


Table 1: Regression table

	(1)
	birthweight (grams)
smoked during pregnancy	-279.8
	(-2.64)
has history of hypertension	-427.7
	(-2.01)
Constant	3081.0
	(45.56)
Observations	189

t statistics in parentheses

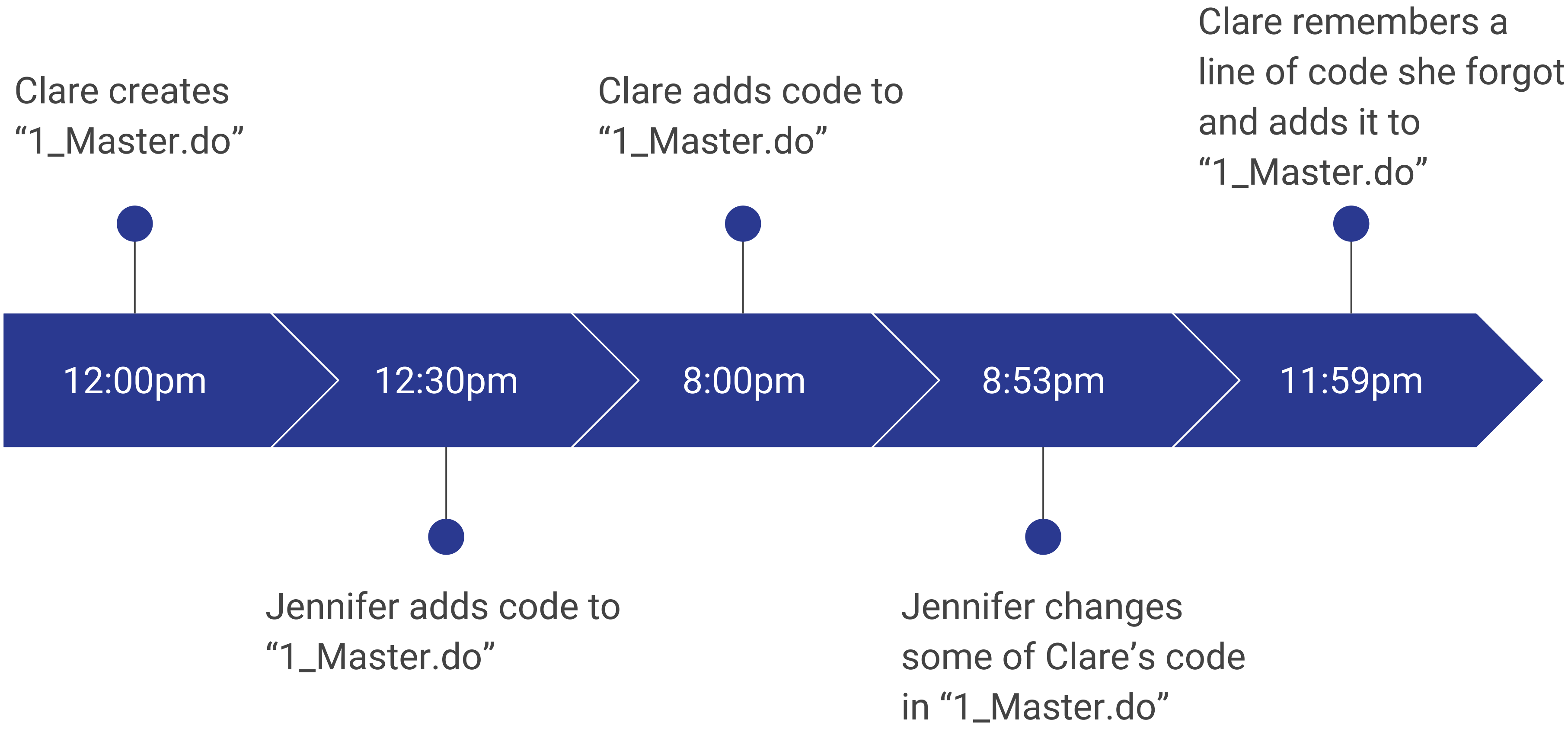
Comments on texdoc

- Texdoc is not an easy package to use!
- Good for keeping track of everything, but that can also be done internally in Stata.
- Lots of code...lots of places for things to go wrong...

Use whichever method works best for you, but don’t be afraid of learning curves

Most robust option: putexcel

Project Tracking + Management



12:00pm

Clare creates “1_Master.do”

12:30pm

Jennifer adds code to “1_Master.do”

8:00pm

Clare adds code to “1_Master.do”

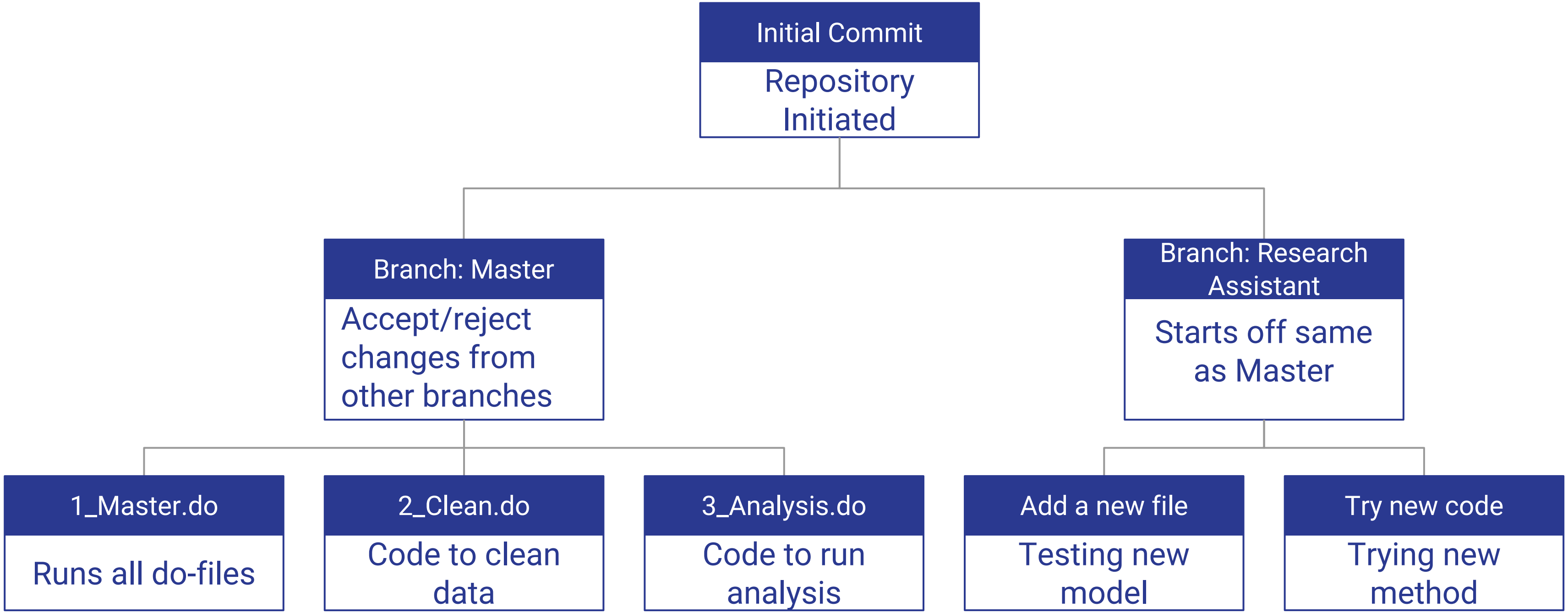
8:53pm

Jennifer changes some of Clare’s code in “1_Master.do”

11:59pm

Clare remembers a line of code she forgot and adds it to “1_Master.do”

GitKraken Branches



Initial Commit

Repository Initiated

Branch: Master

Accept/reject changes from other branches

Branch: Research Assistant

Starts off same as Master

1_Master.do

Runs all do-files

2_Clean.do

Code to clean data

3_Analysis.do

Code to run analysis

Add a new file

Testing new model

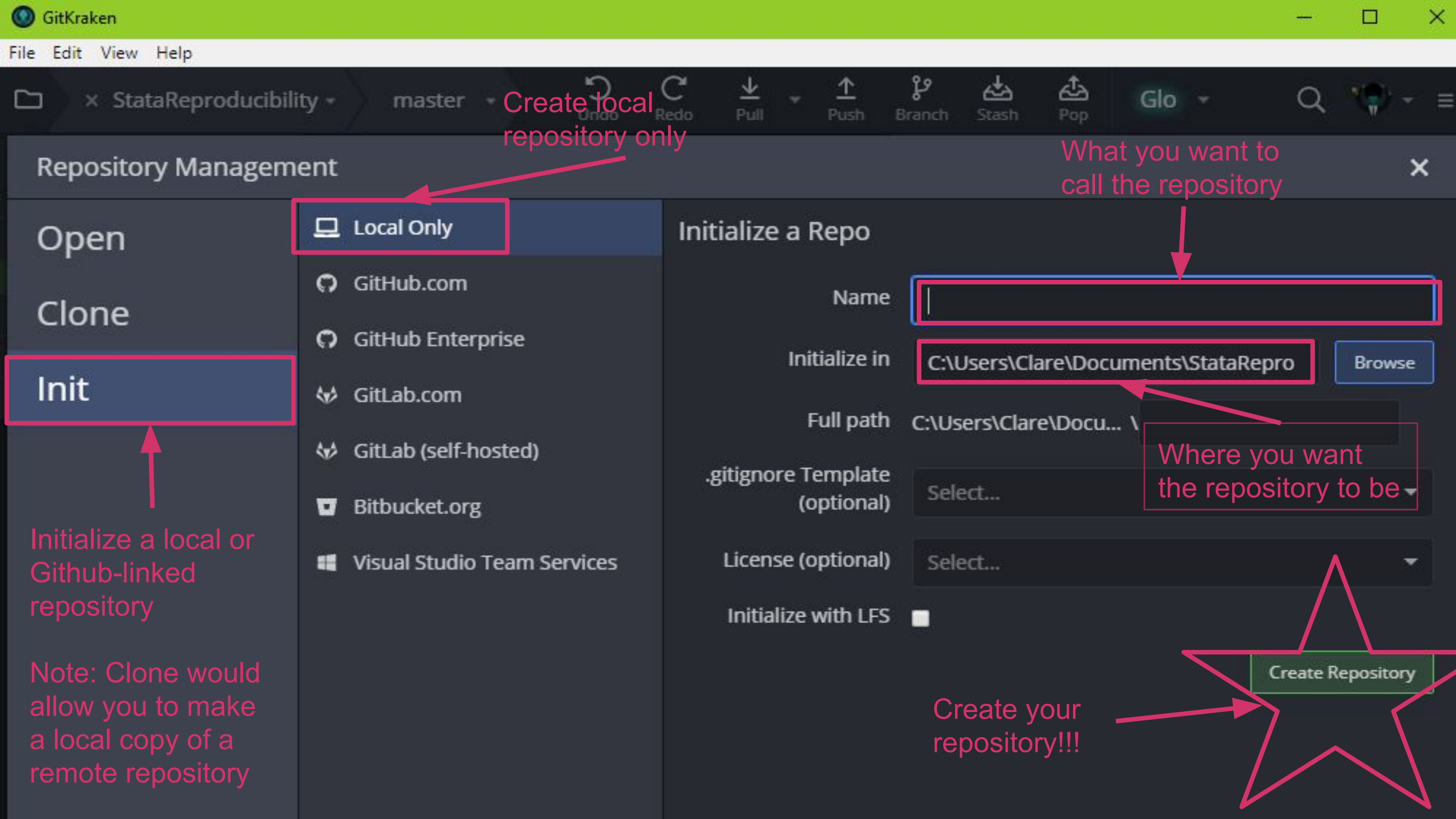
Try new code

Trying new method

Create your first repository!

Note: We will not link to a remote repository, although this is very common

1. Open up GitKraken
2. Create an account
3. Get ready for instructions...



Initialize a Repo

Name

Initialize in C:\Users\Clare\Documents\StataRepro

Full path C:\Users\Clare\Docu... \

.gitignore Template (optional) Select...

License (optional) Select...

Initialize with LFS □

Create local repository only

What you want to call the repository

Where you want the repository to be

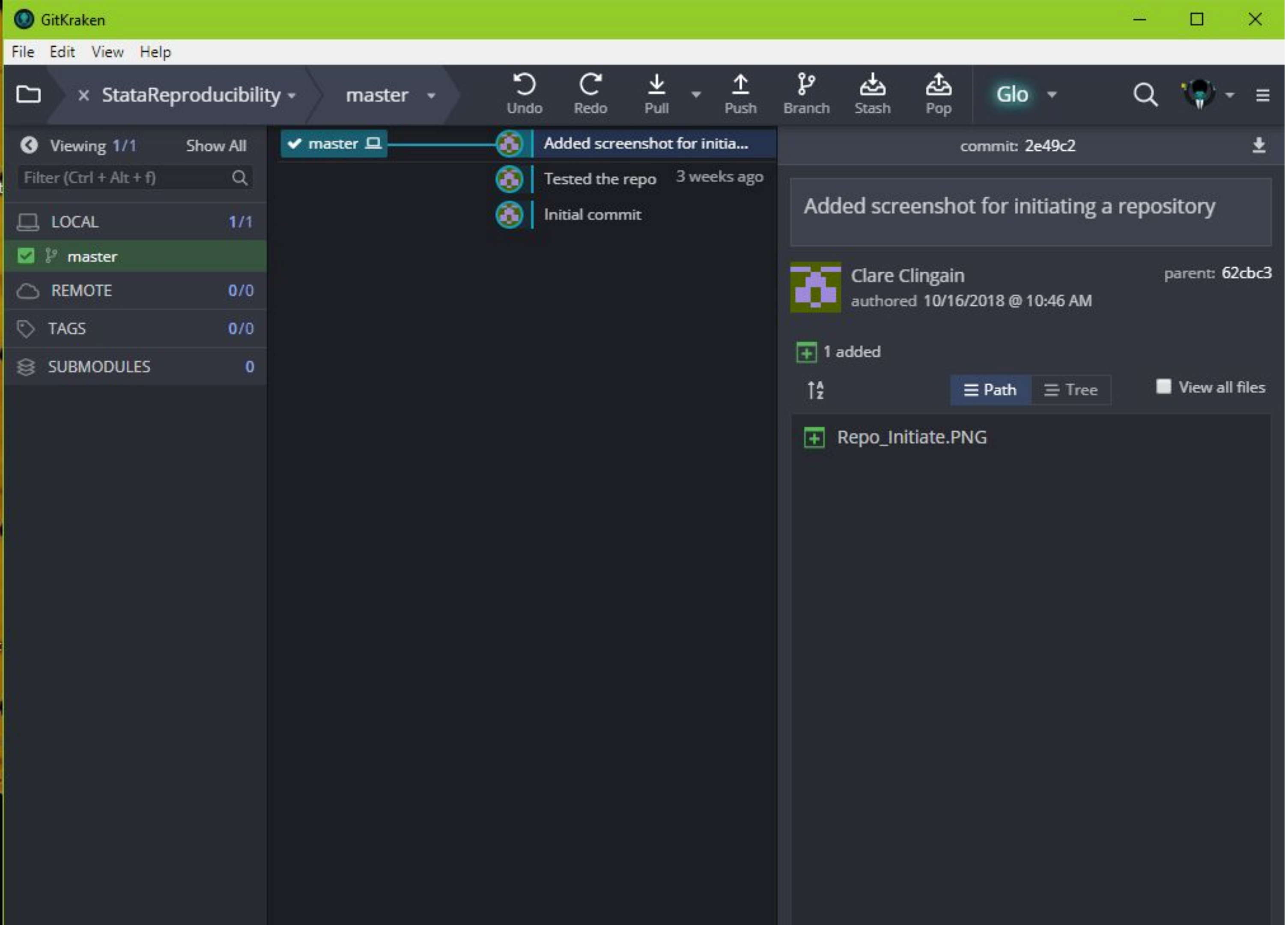
Initialize a local or Github-linked repository

Note: Clone would allow you to make a local copy of a remote repository

Create your repository!!!

Create a Stata Reproducibility Workshop Repo

1. Select “Init”
2. Select “Local only”
3. Name the repository “StataReproducibility”
4. Set file path to “C:/Users/[insert your user]/Documents”
5. Click “Create Repository”



Added screenshot for initia...

Tested the repo 3 weeks ago

Initial commit

commit: 2e49c2

Added screenshot for initiating a repository

Clare Clingain
authored 10/16/2018 @ 10:46 AM
parent: 62cbc3

1 added

Repo_Initiate.PNG

Navigation

Left column: active working directory

LOCAL: master

Middle column: commit tree

Right column: staging and committing area

Top bar: functions

Functions

Pull: get changes from remote repository (i.e., changes made by others)

Push: send your changes to the remote repository

Branch: create a new branch in your repository

Stash: store changes you have made without committing them

Pop: retrieve stashed changes

Stage: set up files for commit

Commit: save your changes (necessary step for pushing)

Add a do-file to the repository

If you are using Stata

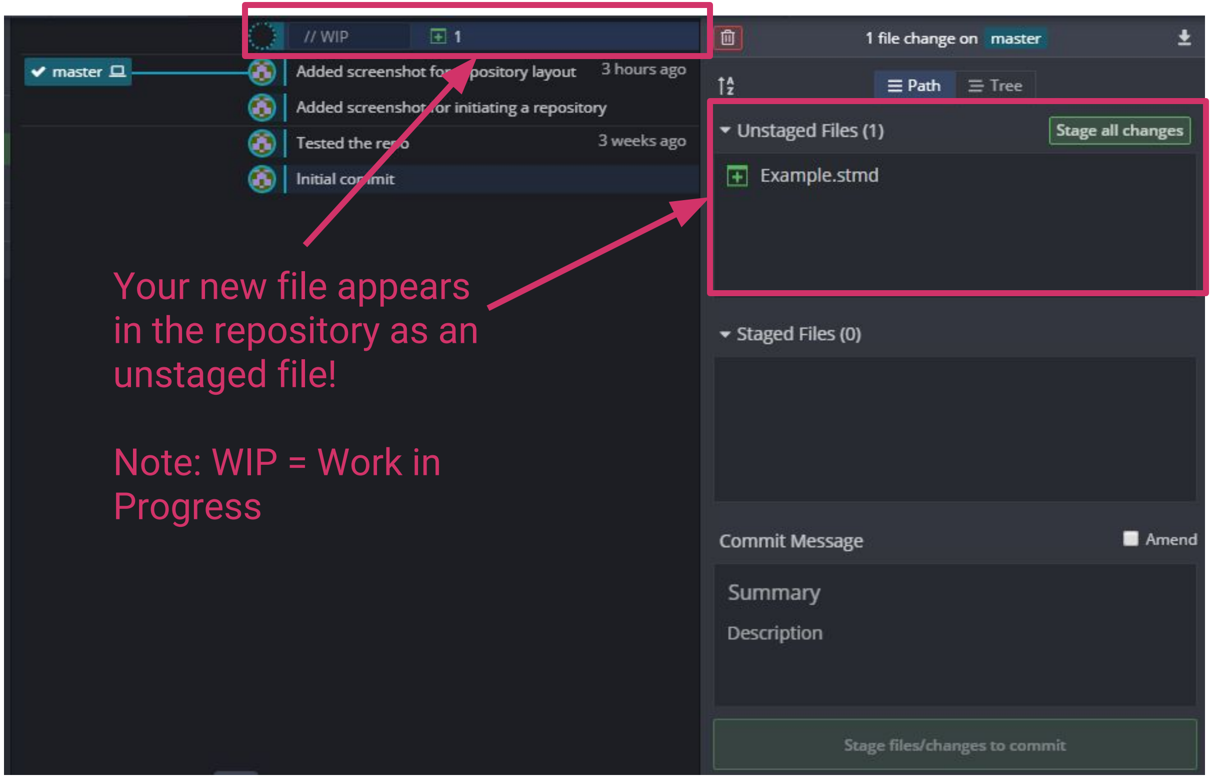
1. Move do-file “Example.do” to the repository folder

If you are using Sublime

1. Move “Example.stmd” OR “Example.txt” file to the repository folder

If you are using Notepad

1. Move “Example.txt” file to the repository folder

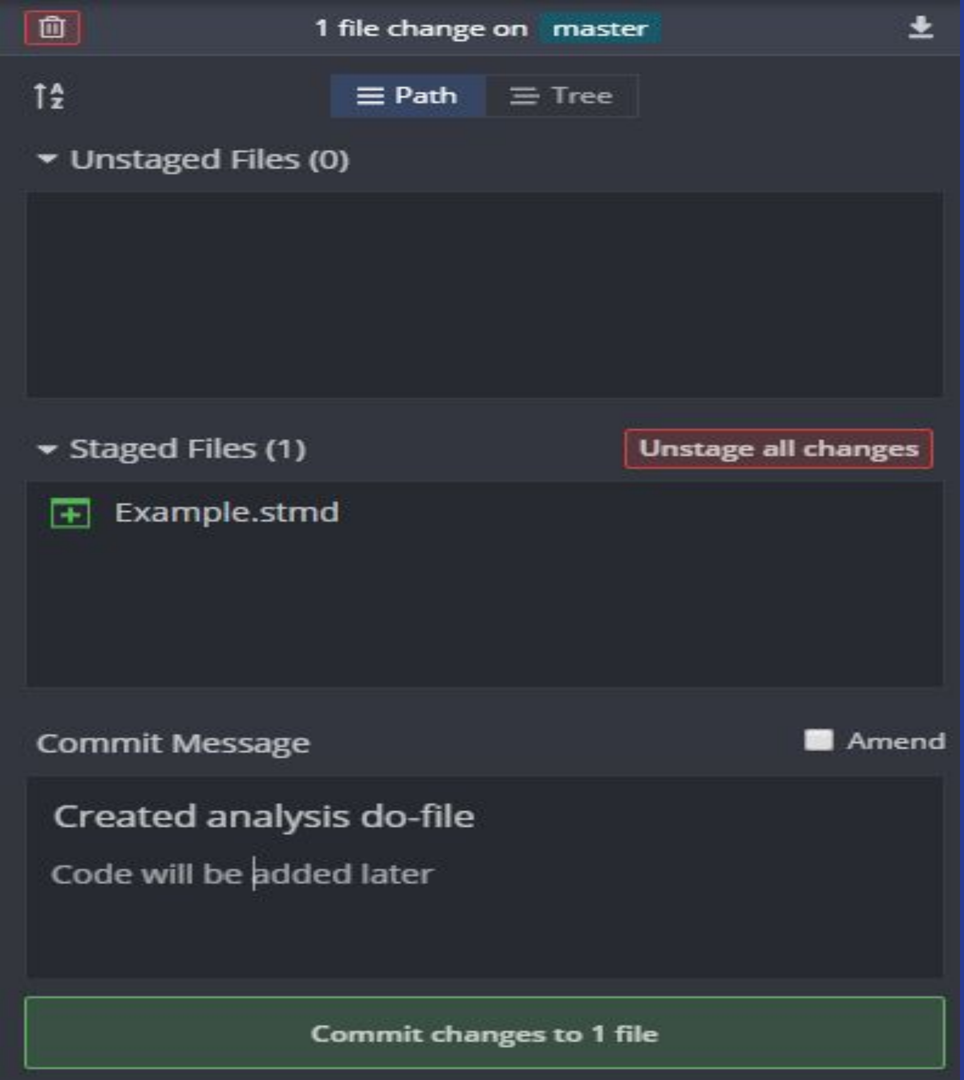


Your new file appears in the repository as an unstaged file!

Note: WIP = Work in Progress

Commit messages

- Should start with verbs
- Summary section: Quick verb-started sentence about what was done
- Description sentence: Any details, notes, or questions you have for other members of the repository to see.



Created analysis do-file

Code will be added later

1. Select “Stage File”
2. Write “Created analysis do-file” to the Commit Summary
3. Write “Code will be added later” to the Commit Description
4. Click “Commit changes to 1 file”

Check your commit tree!

Push/Pull

- If we had a remote repository connected to our local repository, this is where we would first **pull** any changes from the remote repository, and then **push** our changes to the remote repository
- **Always pull before you push to avoid conflicts!!!**
- If two people try to make changes to the same file or the same line of code, this can create a **conflict**.
    - GitKraken has a nice way to “cherry pick” the changes you want if a conflict occurs between two versions of the file

Commit before you make any “experimental” changes to your code!

Stata Reproducibility Assignment

Assignment Requirements

Prompt: Imagine that you have been approached by the Center for Disease Control (CDC) to run statistical analyses on the National Health and Nutrition Examination Survey II (NHANES II), and to compile a report of your findings in table format. The CDC requests that the research process be reproducible so that other researchers can replicate your results and any critics about table results can be more easily shut down.

Data: Center for Disease Control's National Health and Nutrition Examination Survey II (NHANES II)

Assignment Requirements

Tasks:

1. Subset your data based on one demographic feature (eg. gender, race, region, etc.)
2. Create and/or recode 3 variables
3. Run one descriptive analysis of at least 4 variables
4. Export a descriptive table (via estout, putexcel, texdoc -- your choice)
5. Run two predictive analyses: either a nested regression or two models with same predictors, different outcomes
6. Create reproducible code that will run your regressions and export the results into a table shell

Assignment Requirements

Submission:

Please submit the following documents via NYU Classes:

1. A **clean and readable** do-file containing all your code
2. A 1-2 page PDF write-up of your research process containing your final tables
3. An Excel file with your table shell

Thank you!

Contact: clc586@nyu.edu