CHRISTOPHER PRENER, PH.D.
FALL, 2016

WEEK 11
LECTURE 12

QUANTITATIVE ANALYSIS

DOCUMENTING AND DEEP CLEANING DATA

AGENDA

1. Loops
2. Study Documentation
3. Dataset Metadata
4. Dataset Documentation
5. Deep Cleaning

1 LOOPS

AUTOMATING YOUR WORK REVIEW

- Automation helps
  - improve consistency,
  - manage repetitive tasks,
  - limit debugging,
  - and ultimately decrease time spent coding.

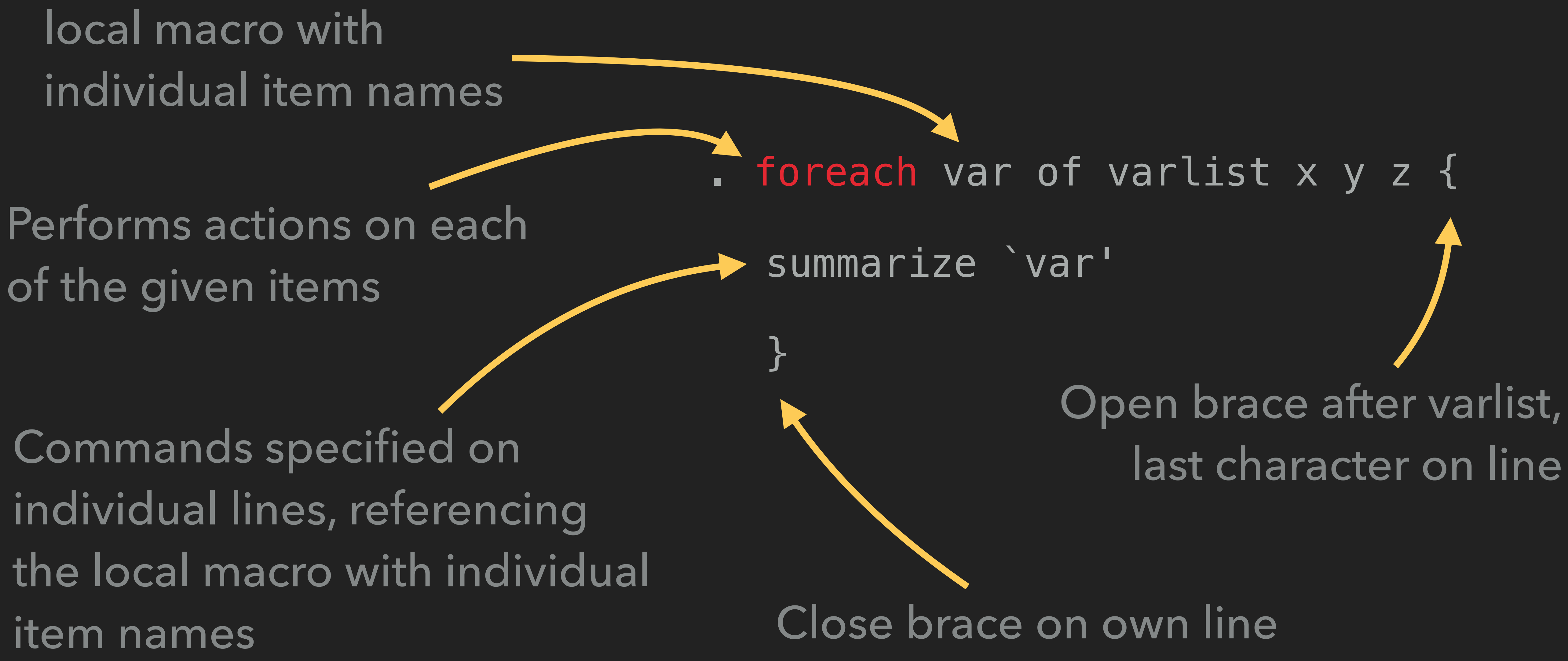
LOCAL MACROS REVIEW

```
// storing lists or other strings:

local macroName "string"
```

```
. local varlist "var1 var2 var3"
```

BASIC ANATOMY OF A LOOP



```
. foreach var of varlist x y z {
  summarize `var'
  }
```

local macro with individual item names

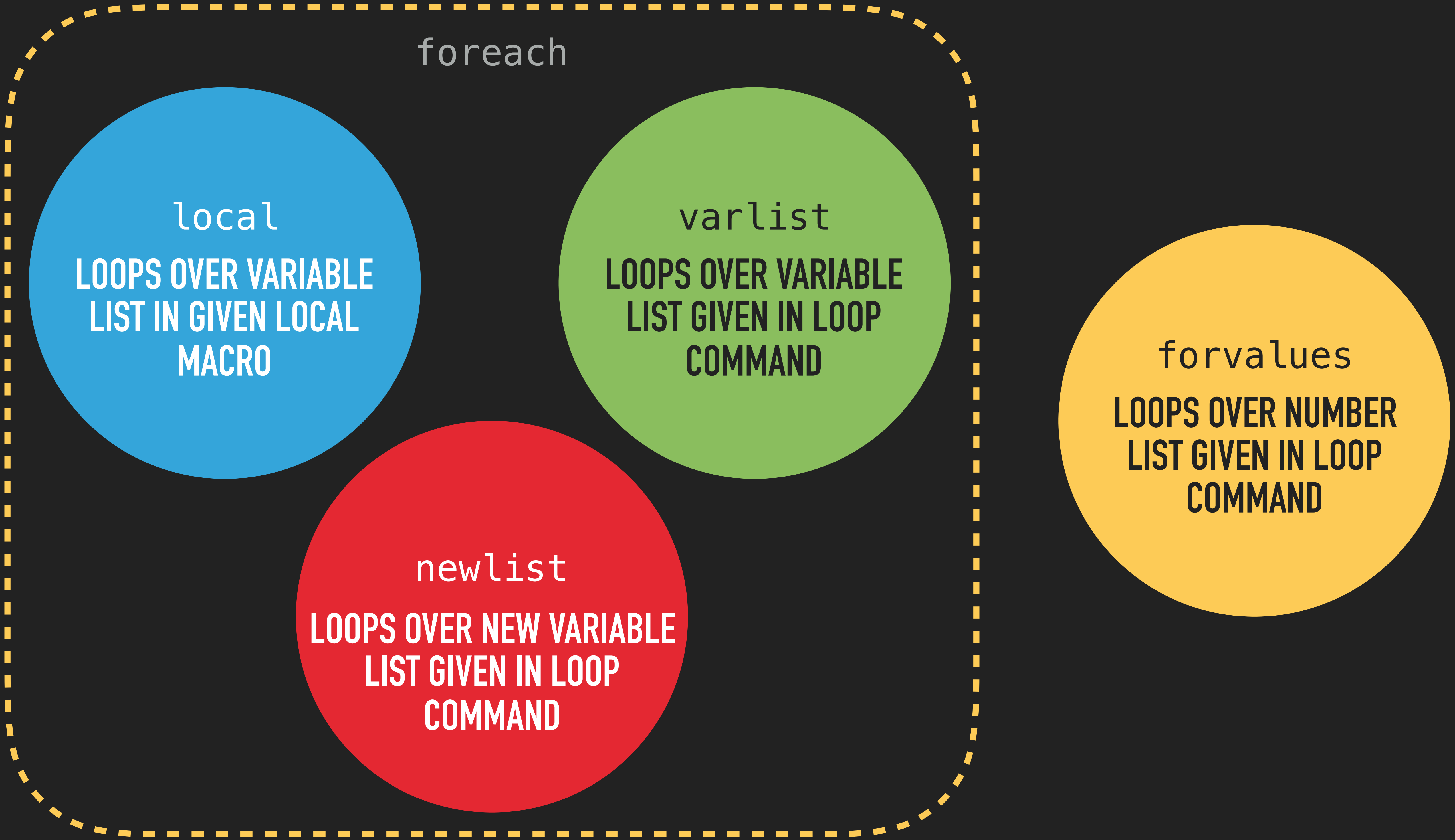
Performs actions on each of the given items

Commands specified on individual lines, referencing the local macro with individual item names

Open brace after varlist, last character on line

Close brace on own line

LOOPS - TYPES



foreach

local

LOOPS OVER VARIABLE LIST IN GIVEN LOCAL MACRO

varlist

LOOPS OVER VARIABLE LIST GIVEN IN LOOP COMMAND

newlist

LOOPS OVER NEW VARIABLE LIST GIVEN IN LOOP COMMAND

forvalues

LOOPS OVER NUMBER LIST GIVEN IN LOOP COMMAND

FOREACH LOOP – VARLIST

```
// action on given items

. foreach var of varlist mpg length weight {
  summarize `var'
  }
```

FOREACH LOOP - LOCAL

```
// action on given items in referenced local

. local coreVars "mpg length weight"
```

no back-tick/apostrophe

```
. foreach var of local coreVars {
   summarize `var'
   }

. describe `coreVars'
```

FOREACH LOOP – NEWLIST

```
// create given variables

. foreach var of newlist x y z {

  generate `var' = .

  }
```

FORVALUES LOOP

```
// action using the range of values

. forvalues i = 1/5 {

  display `i'

  }
```

FORVALUES LOOP

```
// action using the range of values

. forvalues i = 1/5 {
  display `i'
  }
1
2
3
4
5
```

FORVALUES LOOP

```
// action using the range of values

. forvalues i = 1(2)9 {

   display `i'

   }
```

FORVALUES LOOP

```
// action using the range of values

. forvalues i = 1(2)9 {

   display `i'

   }

1

3

5

7

9
```

USE CASE – CREATE PLOTS

```
// action using the range of values

. forvalues i = 1/5 {

   histogram mpg if rep78 == `i', scheme(s2mono)

   graph export fig`i'.png, as(png) width(800) height(600) replace

  }
```

PULLING IN A SECOND PIECE OF DATA

local macro with figure number

```
. local j = 5

. forvalues i = 1/5 {

   histogram mpg if rep78 == `i', scheme(s2mono)

   graph export fig`j'.png, as(png) width(800) height(600) replace

   local ++j

   }
```

reference local macro

increase local macro by increment of 1

PULLING IN A SECOND PIECE OF DATA

```
. local j = 5

. forvalues i = 1/5 {

   histogram mpg if rep78 == `i', scheme(s2mono)

   graph export fig`j'.png, as(png) width(800) height(600) replace

   local j = `j'+2

  }
```

increase local macro by increment of 2

PULLING IN A SECOND PIECE OF DATA

```
. local sourceVars "mpg weight length turn"

. local i = 1

. foreach var of newlist mpgBin weightBin lengthBin turnBin {

   local source : word `i' of `sourceVars'

   generate `var' = .

   quietly summarize `source'

   replace `var' = 0 if `source' <= r(mean)

   replace `var' = 1 if `source' > r(mean)

   local ++i

   }
```

PULLING IN A SECOND PIECE OF DATA

```
. local sourceVars "mpg weight length turn"

. local i = 1

. foreach var of newlist mpgBin weightBin lengthBin turnBin {

   local source : word `i' of `sourceVars'

   generate `var' = .

   quietly summarize `source'

   replace `var' = 0 if `source' <= r(mean)

   replace `var' = 1 if `source' > r(mean)

   local ++i

   }
```

PULLING IN A SECOND PIECE OF DATA

```
generate mpgBin = .
quietly summarize mpg
replace mpgBin = 0 if mpg <= r(mean)
replace mpgBin = 1 if mpg > r(mean)
generate weightBin = .
quietly summarize weight
replace weightBin = 0 if weight <= r(mean)
replace weightBin = 1 if weight > r(mean)
generate lengthBin = .
quietly summarize length
replace lengthBin = 0 if length <= r(mean)
replace lengthBin = 1 if length > r(mean)
generate turnBin = .
quietly summarize turn
replace turnBin = 0 if turn <= r(mean)
replace turnBin = 1 if turn > r(mean)
```

THIS CODE COMPLETES THE SAME TASK AS THE LOOP ON THE PREVIOUS SLIDE. THE LOOP IS 10 LINES LONG. THIS CODE IS 16 LINES LONG.

PULLING IN A THIRD PIECE OF DATA

```
. local sourceVars "mpg weight length turn”
. local labels "mileage weight length turn"
. local i = 1
. foreach var of newlist mpgBin weightBin lengthBin turnBin {
   local source : word `i' of `sourceVars'
   local label : word `i' of `labels'
   generate `var' = .
   quietly summarize `source'
   replace `var' = 0 if `source' <= `r(mean)'
   replace `var' = 1 if `source' > `r(mean)'
   label variable `var' "Binary measure of `label'"
   local ++i
   }
```

PULLING IN A THIRD PIECE OF DATA

```
. local sourceVars "mpg weight length turn”

. local labels "mileage weight length turn"

. local i = 1

. foreach var of newlist mpgBin weightBin lengthBin turnBin {

  local source : word `i' of `sourceVars'

  local label : word `i' of `labels'

  generate `var' = .

  quietly summarize `source'

  replace `var' = 0 if `source' <= `r(mean)'

  replace `var' = 1 if `source' > `r(mean)'

  label variable `var' "Binary measure of `label'"

  local ++i

  }
```

PULLING IN A THIRD PIECE OF DATA

```
. local sourceVars "mpg weight length turn”

. local labels `""miles per gallon" "weight" "length" “turn""'

. local i = 1

. foreach var of newlist mpgBin weightBin lengthBin turnBin {

   local source : word `i' of `sourceVars'

   local label : word `i' of `labels'

   generate `var' = .

   quietly summarize `source'

   replace `var' = 0 if `source' <= `r(mean)'

   replace `var' = 1 if `source' > `r(mean)'

   label variable `var' "Binary measure of `label'"

   local ++i

   }
```

SOME FINAL THOUGHTS

- Flat is better than nested - nested loops (and loops with nested if commands) are difficult to debug and should be avoided unless there is a significant advantage - i.e. convince Chris they are necessary!

- If the code to do the entire task is…

    - *shorter* or about the same length than the code that a loop needs to complete the task, don’t use a loop (simple is better than complex!).

    - *much longer* than the code that a loop needs to complete the same task, use a loop (complex is better than complicated!).

2 STUDY DOCUMENTATION

TYPES

VERSION CONTROL
SYSTEMS LIKE GITHUB PROVIDE A CONTAINER WHERE ALL CHANGES TO YOUR PROJECT ARE TRACKED

RESEARCH LOG
NOTES COVERING EACH PHASE OF THE DATA ANALYSIS PROCESS

META DICTIONARY
TRACKS ALL FILES WITHIN FOLDER HIERARCHY

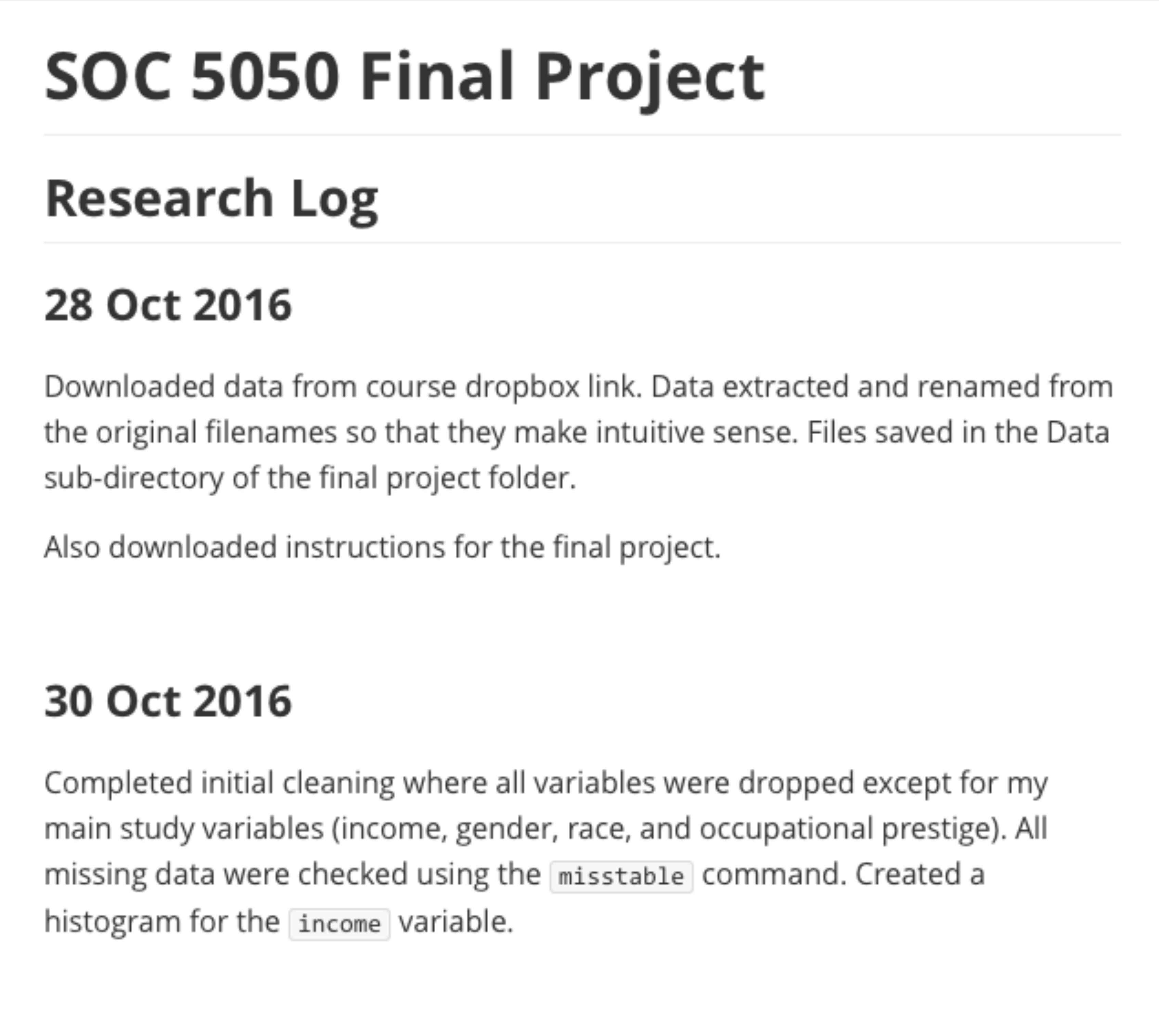
VERSION CONTROL

- Systems like GitHub, with commits, provide a timeline for the changes you make to your projects
- Commit messages and descriptions become a core piece of understanding how your project data have evolved over time
- With some advanced (i.e. command line) techniques, this commit history can be viewed and printed as part of your study's documentation
- Students get free GitHub account upgrades (private repos) - see https://education.github.com/pack

RESEARCH LOG

- Document each step in the research process (see Long 2009:39-43)
- Not just coding but literature review (search terms, areas for further analysis, analytical connections)
- If you are using GitHub, you can focus on higher level activities and changes to your data & code
  - If you are not using Git, the research log needs to be a guide to *every* change you are making to the data
- If you make changes without noting them, it can be difficult to work backwards or remember in a month or a year

RESEARCH LOG



SOC 5050 Final Project

Research Log

28 Oct 2016

Downloaded data from course dropbox link. Data extracted and renamed from the original filenames so that they make intuitive sense. Files saved in the Data sub-directory of the final project folder.

Also downloaded instructions for the final project.

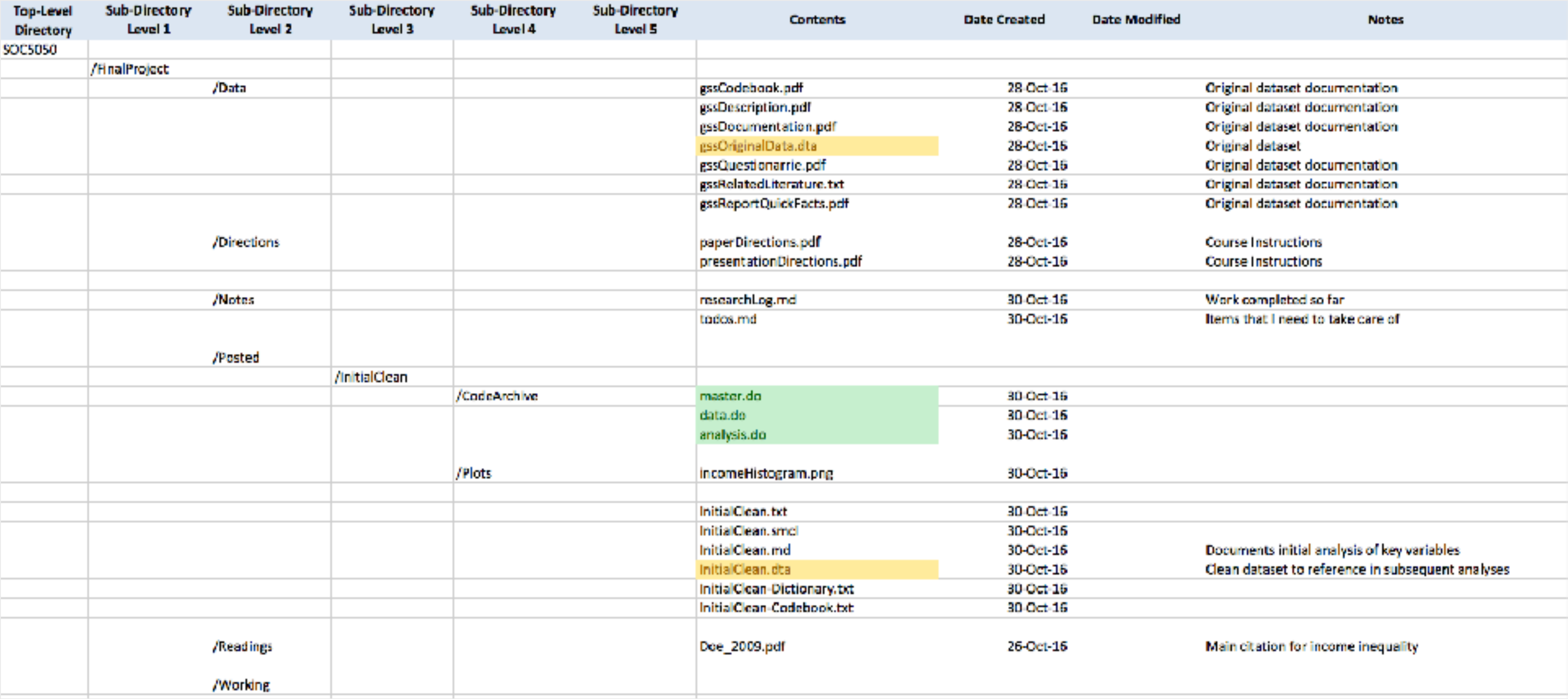
30 Oct 2016

Completed initial cleaning where all variables were dropped except for my main study variables (income, gender, race, and occupational prestige). All missing data were checked using the `misstable` command. Created a histogram for the `income` variable.

META DICTIONARY

- Long (2009) calls this a "Data Registry" (see Long 2009:44-45)
- Use this to visualize and track your directory structure
- If you are using GitHub, you can use this simply as an index for understanding where files are in your directory structure
    - If not using GitHub, you want to track changes in a more granular fashion

META DICTIONARY



Top-Level Directory	Sub-Directory Level 1	Sub-Directory Level 2	Sub-Directory Level 3	Sub-Directory Level 4	Sub-Directory Level 5	Contents	Date Created	Date Modified	Notes
SOC5050									
	/FinalProject								
		/Data				gssCodebook.pdf	28-Oct-16		Original dataset documentation
						gssDescription.pdf	28-Oct-16		Original dataset documentation
						gssDocumentation.pdf	28-Oct-16		Original dataset documentation
						gssOriginalData.dta	28-Oct-16		Original dataset
						gssQuestionarrie.pdf	28-Oct-16		Original dataset documentation
						gssRelatedLiterature.txt	28-Oct-16		Original dataset documentation
						gssReportQuickFacts.pdf	28-Oct-16		Original dataset documentation
		/Directions				paperDirections.pdf	28-Oct-16		Course Instructions
						presentationDirections.pdf	28-Oct-16		Course Instructions
		/Notes				researchLog.md	30-Oct-16		Work completed so far
						todos.md	30-Oct-16		Items that I need to take care of
		/Posted							
			/InitialClean						
				/CodeArchive		master.do	30-Oct-16		
						data.do	30-Oct-16		
						analysis.do	30-Oct-16		
				/Plots		incomeHistogram.png	30-Oct-16		
						InitialClean.txt	30-Oct-16		
						InitialClean.smcl	30-Oct-16		
						InitialClean.md	30-Oct-16		Documents initial analysis of key variables
						InitialClean.dta	30-Oct-16		Clean dataset to reference in subsequent analyses
						InitialClean-Dictionary.txt	30-Oct-16		
						InitialClean-Codebook.txt	30-Oct-16		
		/Readings				Doe_2009.pdf	26-Oct-16		Main citation for income inequality
		/Working							

3 DATASET METADATA

LABELS CAN BE ATTACHED TO DATASETS

```
// labels typically lay out the version and topic of the data
```

```
. sysuse auto.dta, clear

(1978 Automobile Data)
```

Dataset label

LABELS CAN BE ATTACHED TO DATASETS

```
// labels typically lay out the version and topic of the data


. sysuse auto.dta, clear

(1978 Automobile Data; Analysis Dataset)
```

LABELS CAN BE ATTACHED TO DATASETS

```
// labels typically lay out the version and topic of the data

// limited to 80 characters in length


. sysuse auto.dta, clear

(1978 Automobile Data; Analysis Dataset)
```

LABELS CAN BE ATTACHED TO DATASETS

```
label data “dataset label text”
```

```
. label data “auto analysis dataset – 28 Oct 2016”
```

NOTES CAN BE ATTACHED TO DATASETS

```
// notes at the dataset level can function in a similar way to
// GitHub commits


. notes _dta

_dta:
  1.  from Consumer Reports with permission
  2.  analysis version created 28 Oct 2016 by Chris
```

NOTES CAN BE ATTACHED TO DATASETS

```
notes _dta: dataset note text
```

```
. notes _dta: analysis version created 28 Oct 2016 by Chris
```

NOTES CAN BE ATTACHED TO VARIABLES

```
. notes varname
```

```
varname:

  1.   Text of note 1

  2.   Text of note 2
```

NOTES CAN BE ATTACHED TO DATASETS

```
notes varname: variable note text
```

```
. notes varname: Text of new note.
```

4 DATASET DOCUMENTATION

TYPES

VARIABLE INDEX
PROVIDES A QUICK REFERENCE TO VARIABLE NAMES AND LABELS TO FACILITATE WORKING WITH LARGER DATASETS

CODEBOOK
PROVIDES DETAILED TABLES AND NOTES FOR EACH VARIABLE IN THE DATASET

CREATING A VARIABLE INDEX

- Create your index in a separate text file
- This requires special a special log file

```
. quietly log using “$projName/$projName-Index.txt", text replace name(index)

. describe

. quietly log close index
```

CREATING A VARIABLE INDEX

```
Contains data from /Applications/Stata/ado/base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          12 Nov 2015 15:38
 size:         3,182                          (_dta has notes)
---------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
---------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
---------------------------------------------------------------------------------
Sorted by: foreign
```

CREATING A CODEBOOK

```
codebook [varlist] [, header notes]
```

```
. codebook, header notes

{output omitted}
```

```
. codebook mpg length weight, header notes

{output omitted}
```

CREATING A CODEBOOK

- Create your codebook in a separate text file
- This requires special a special log file

```
. quietly log using “$projName/$projName–CodeBook.txt", text replace ///
name(codebook)

. codebook, header notes

. quietly log close codebook
```

CREATING A CODEBOOK

```
              Dataset:  /Applications/Stata/ado/base/a/auto.dta
            Last saved:  12 Nov 2015 15:38

                 Label:  1978 Automobile Data
   Number of variables:  12
Number of observations:  74
                  Size:  3,182 bytes ignoring labels, etc.

_dta:
  1.  from Consumer Reports with permission
  2.  analysis version created 28 Oct 2016 by Chris

--------------------------------------------------------------------------------
make                                                              Make and Model
--------------------------------------------------------------------------------

                  type:  string (str18), but longest is str17

         unique values:  74                       missing "":  0/74

              examples:  "Cad. Deville"
                         "Dodge Magnum"
                         "Merc. XR-7"
                         "Pont. Catalina"

               warning:  variable has embedded blanks
```

5 DEEP CLEANING

WORKFLOW FOR DATA CLEANING

1. What variables do you need?
2. How is missing data handled? Recode as necessary…
3. Do variables need to be recoded? Recode as necessary…
4. Logic checks - should certain variables be missing or coded in specific way? Check and correct as necessary…

WE WANT TIDY DATA!!!

MISSING VALUES HANDLED CORRECTLY (AND CORRECTED**), NORMALITY OF VARIABLES CHECKED (AND CORRECTED**), VARIABLES RECODED AS NECESSARY INTO ANALYTICALLY USEFUL MEASURES.

** DOING THIS COMPLETELY IS BEYOND SCOPE OF THIS CLASS

CREATING DUMMY VARIABLES

```
tabulate varname, generate(groupName)
```

```
. tabulate rep78, generate(repair)

. describe repair1-repair5

              storage   display    value
variable name   type    format     label      variable label
---------------------------------------------------------------------------
repair1         byte    %8.0g                 rep78== 1.0000
repair2         byte    %8.0g                 rep78== 2.0000
repair3         byte    %8.0g                 rep78== 3.0000
repair4         byte    %8.0g                 rep78== 4.0000
repair5         byte    %8.0g                 rep78== 5.0000
```

MISSING DATA PITFALLS

```
. generate repBin = .

. replace repBin = 0 if rep78 <= 3

. replace repBin = 1 if rep78 > 3

. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         74    .4594595    .5017555          0          1
```

MISSING DATA PITFALLS

```
. generate repBin = .

. replace repBin = 0 if rep78 <= 3

. replace repBin = 1 if rep78 > 3 & rep78 < .

. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         69    .4202899    .4972216          0          1
```

MISSING DATA PITFALLS

```
. generate repBin = .

. replace repBin = 0 if rep78 <= 3

. replace repBin = 1 if rep78 > 3 & rep78 < .

. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         69    .4202899    .4972216          0          1
```

GENERALIZING THIS PROBLEM

- Under certain conditions, we expect variables to contain certain types of data.

- If a “parent” variable has missing data in a particular observation, any variable created from it should also have missing data in the same observations.

- If one variable implies that a respondent has a particular characteristic, that characteristic should follow through subsequent variables (i.e. identifying as Asian in a ‘race’ variable should be followed by being identified as ‘non-White’ in a recoded variable).

LOGIC CHECKS

- The `assert` command allows you to construct logic checks in your do-file - these are tests if one or more conditions are true.
- If no output is returned, the condition is true.
- If output is returned, it is false and your do-file will stop executing.

```
assert expression [if] [additionalExpressions]
```

```
. assert mpg < .
```

LOGIC CHECKS

```
. generate repBin = .

. replace repBin = 0 if rep78 <= 3

. replace repBin = 1 if rep78 > 3

. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         74    .4594595    .5017555          0          1
```

LOGIC CHECKS

```
. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         74    .4594595    .5017555          0          1

. assert repBin == . if rep78 == .
5 contradictions in 5 observations
assertion is false
r(9);
```

LOGIC CHECKS

```
. summarize rep78 repBin

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       rep78 |         69    3.405797    .9899323          1          5
      repBin |         69    .4202899    .4972216          0          1

. assert repBin == . if rep78 == .
```

DOCUMENT DETAILS

Document produced by Christopher Prener, Ph.D for the Saint Louis University course SOC 5050: QUANTITATIVE ANALYSIS - APPLIED INFERENTIAL STATISTICS. See the course wiki and the repository README.md file for additional details.