Stata SOS: Navigating Errors in Stata

Yung-Yu Tsai

Truman School of Government & Public Affairs
University of Missouri

October 27, 2023

Are you afraid of Stata errors?

Karolina Brodin

Join Date: Feb 2018
Posts: 1

Error r(111)
19 Feb 2018, 18:12

Hi,

I just started to learn Stata today and I can`t get my do.file to run.

I am trying to run http://masteringmetrics.com/wp-conte...9_hicompare.do

and I get the error code r(111):

Eva Brown

Join Date: Apr 2018
Posts: 2

1 group found, 2 required r(420); Error
27 Apr 2018, 16:13

Hi Everyone,

I'm running a lot of ttests and I keep getting this error whenever there aren't enough observations to test. code:

```
foreach region in list {
bysort ethnicity: ttest meanscore, by(survey)
bysort personofcolor: ttest meanscore, by(survey)
bysort gender: ttest meanscore, by(survey)
bysort cgp: ttest meanscore, by(survey)
bysort economicbackground: ttest meanscore, by(survey)
}
```

Shilpi Mukherjee

Join Date: Jun 2018
Posts: 1

Error r(100) - Varlist Required
24 Jun 2018, 19:29

Hello,
I am fairly new to Stata and am trying to write a code to understand the sequence of flights.My code is

```
set obs 509519
local count = 1
local slno = 1
gen sequence = _n
forval slno = 1(1)509517{
replace `sequence' = `count' in `slno'
while ((flightnum[`slno'] == flightnum[`slno' + 1]) && (tailnum[`slno'] == tailnum[`slno'+1])){
replace `sequence' = `count' in `slno'
replace `count' = `count' + 1
replace `slno' =`slno' + 1
}
```

Outline

1. Error vs. Warning: Why do we need to care about error?
2. Understanding: What do error messages and help files say?
3. Debugging: What if there are no clear error messages?
4. Handling: What if the errors are inevitable?
5. Fixing: What if the errors come from the author of the command?

Example Dataset

- Survey.xlsx
  - This is a fake data!
  - Has six sheets, from 2015 to 2020
  - Each sheet has nine variables: *id*, *year*, *age*, *sex*, *income*, *race*, *satisfaction*, *vote_senate* or *vote_house*, and *weight*
  - Odd years (2015, 2017, 2019) only have *vote_senate*, and even years (2016, 2018, 2020) only have *vote_house*
- Before importing data, remember to change the path

```
8 cd "/Users/yungyu/Dropbox/03 Teaching/Mizzou/
    Stata Error/material/data"
```

"Error" vs. "Warning"

- Let's first explore the dataset

```
14 import excel "Survey.xlsx", sheet(2015) clear
       first //import the data
15 sum //summarize everything
16 describe //describe everything
17 tab satisfaction
```

- We find *satisfaction* to be a numeric variable but loaded as a string variable

"Error" vs. "Warning"

- Let's make it into a numeric variable by "**destring**" it

```
20 destring satisfaction
```

- We get an error message

```
must specify either generate or replace option
r(198);
```

- Error
  - Comes with some message in red and an error code like r(198);
  - Your code would be stopped!
  - The command you just ran would not work
  - In most of cases, nothing would change

"Error" vs. "Warning"

- Let's rerun the code but add replace at the end

```
21 destring satisfaction, replace
```

- We get a warning message

```
satisfaction: contains nonnumeric characters; no replace
```

- Warning
  - Might give you some message in **red**, **blue**, or **black**
  - Your code would NOT be stopped
  - The command you just ran *might* or *might not*, or might *partly* work
  - Things *might or might not* change
  - There could probably be something wrong that you should notice

Some more examples on "Warning"

- Might fail to estimate the standard error

```
. margins, dydx(*) atmeans predict(outcome)
WARNING: variance matrix is nonsymmetric or highly singular
```

- Might result in biased estimation

```
. logit depvar indvar x1 x2
WARNING: Convergence not achieved.
```

```
. reghdfe depvar indvar, a(unitid year#region) cl(unitid) keepsing
WARNING: Singleton observations not dropped; statistical significance is biased
```

- Some results would not be reported

```
. ivreg2 yvar xvar (treat = iv), cl(cluster)
WARNING: estimated covariance matrix of moment conditions not of full rank.
overidentification statistic not reported.
```

Outline

Understanding: What do error messages and help files say?

Sometimes, you will be able to find a solution to fix the errors by:

- Read the error message or click the error code
- Carefully review the Stata help file

```
20 destring satisfaction
```

```
must specify either generate or replace option
r(198);
```

```
26 destring satisfaction, replace force
```

Do you know you can "**CLICK**" error code?

- Let's import 2016 data:

```
35 import excel "Survey.xlsx", sheet(2016) first
```

```
no; data in memory would be lost
r(4);
```

- If you click r(4); you would get the information below:

```
[P] error . . . . . Return code 4
no; dataset in memory has changed since last saved. You attempted to perform
a command that would substantively alter or destroy the data, and the data have
not been saved, at least since the data were last changed. If you wish to continue
anyway, add the clear option to the end of the command. Otherwise, save the
data first.
```

The error code would lead you to the solution(s)

[P] error . . . . . Return code 4

no; dataset in memory has changed since last saved. You attempted to perform a command that would substantively alter or destroy the data, and the data have not been saved, at least since the data were last changed. If you wish to continue anyway, (1) add the **clear** option to the end of the command. Otherwise, (2) **save** the data first.

- The message gives you the solution and also one reminder:
  - Add the **clear** option to the end of the command.
  - But, **save** the data first, if you do not want to destroy the data

```
37 save "Survey 2015.dta", replace //Let's save the
       data first
38 import excel "Survey.xlsx", sheet(2016) first
     clear //And then, add clear to the end of the
       initial command
```

Some more examples

```
40 by race: tab satisfaction //We want to see
     people's satisfaction level by their race
```

```
not sorted
r(5);
```

```
[P] error . . . . . Return code 5
not sorted
The observations of the data are not in the order required. To solve the problem,
use sort to sort the data then reissue the command; see help sort.
```

```
42 sort race
43 by race: tab satisfaction //We get what we want!
```

Some more examples

- r(109); type mismatch: change string to numeric (or vice versa)
- r(110); (variable) _____ already defined: change a name!
- r(111); variable _____ not found: check the spelling!
- r(170); unable to change to _____ (path): check the path name!
  - Be aware of the difference between Mac and Windows
  - In Mac, the direction of the slash ("\" vs. "/") matters
  - Use quote (" ") when the path has space(s)
  - Avoid the use of foreign language and special characters
- r(601); file _____ not found: check the path or file name!
- r(602); file _____ already exists: change a name or specify "replace"

Understanding help files

- Sometimes, the error message gives you a hint but not a solution
- You will be able to find the solution by reviewing the help file

```
47 tab race sex satisfaction //Let's explore the
     relationship between race, sex, and
     satisfaction; but we get an error
```

```
too many variables specified
r(3);
```

- We know we specified too many variables
- But how many are too many?
- And, what if we really want to see the relationship between all three variables?

Understanding help files

- Let's take a look at the help file

```
49 help tab
50 help tabulate twoway
```

Syntax

Two-way table
tabulate *varname1 varname2* [*if*] [*in*] [*weight*] [, *options*]

Two-way table for all possible combinations - a convenience tool
tab2 *varlist* [*if*] [*in*] [*weight*] [, *options*]

- The "*varname1 varname2*" part tells us the maximum number of variables is two
- The "**tab2**" command provides us an alternative way to work with three variables

Understanding help files: Structure

- On the top-right corner, you will see a "Jump to" button
- The structure of the help file could be organized into several parts:

Syntax	How to use the command with syntax (coding)
Menu	How to use the command with the user interface
Description	What is this command for?
Options	What are the available options?
Examples	Some examples
Stored results	What data would be stored after the command?
References	References in the help file; (Sometimes) If you want to cite the command, who should you cite for?

Understanding help files: Syntax

- The **underline** in the syntax means the available abbreviation
  - You can use **tabulate**, **ta**, **tab**, **tabul**

Syntax

Two-way table
tabulate *varname1 varname2* [*if*] [*in*] [*weight*] [, *options*]

- This rule also applies to *options*
  - "tab var1 var2, ch" would work as good as "tabulate var1 var2, chi2"

options	Description
Main	
chi2	report Pearson's chi-squared
exact[()]	report Fisher's exact test

Understanding help files: Options

- Everything inside the [squared brackets] is *optional*
  - This includes the [*if*], [*in*], [*weight*], and [*options*]
  - Everything before the comma is required, and everything after the comma is optional? NO!

```
57 help destring
```

destring [*varlist*], {**generate**(*newvarlist*)|**replace**} [*destring_options*]

- [*varlist*] is in the bracket → is OPTIONAL
- {**generate**|**replace**} is NOT in the bracket → is REQUIRED

```
58 destring satisfaction //This won't work
59 destring, replace //This will work
```

Understanding help files: Options

- There would be hyperlinks in the help file to help you navigate the options
- Some help files are more organized than others
- The order of the options does not matter
- If you get an error like Option not allowed or Option required, go check the help file and see what is required and what is available

Understanding help files: If, in, and weight

- Most commands allow for [*if*] and [*in*]
  - But some don't allow. For example: destring, merge
- Many commands allow for [*weight*]
  - But some don't allow. For example: generate, list
  - For those which allow, they allow weights in different ways
- There are four types of weights in Stata

62
```
help regress
```

aweights, **fweights**, **iweights**, and **pweights** are allowed; see weight.

63
```
help summarize
```

aweights, **fweights**, and **iweights** are allowed. However, **iweights** may not be used with the **detail** option; see weight.

Understanding help files: If, in, and weight

aweights, **fweights**, and **iweights** are allowed. However, **iweights** may not be used with the **detail** option; see weight.

```
65 sum age [aweight=weight] //This works
66 sum age [pweight=weight] //This doesn't work
67 sum age [iweight=weight] //This works
68 sum age [iweight=weight], detail //This doesn't
   works
```

- If you get an error like pweights not allowed, sometimes changing a weighting approach could work
- But you must be very careful! Sometimes weighting approach changes the results

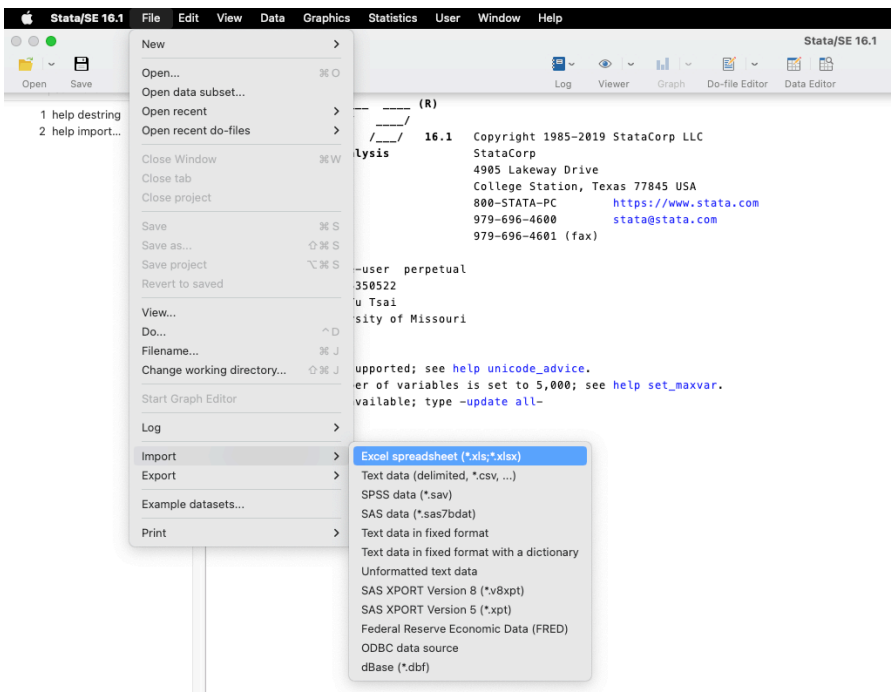
Understanding help files: Menu

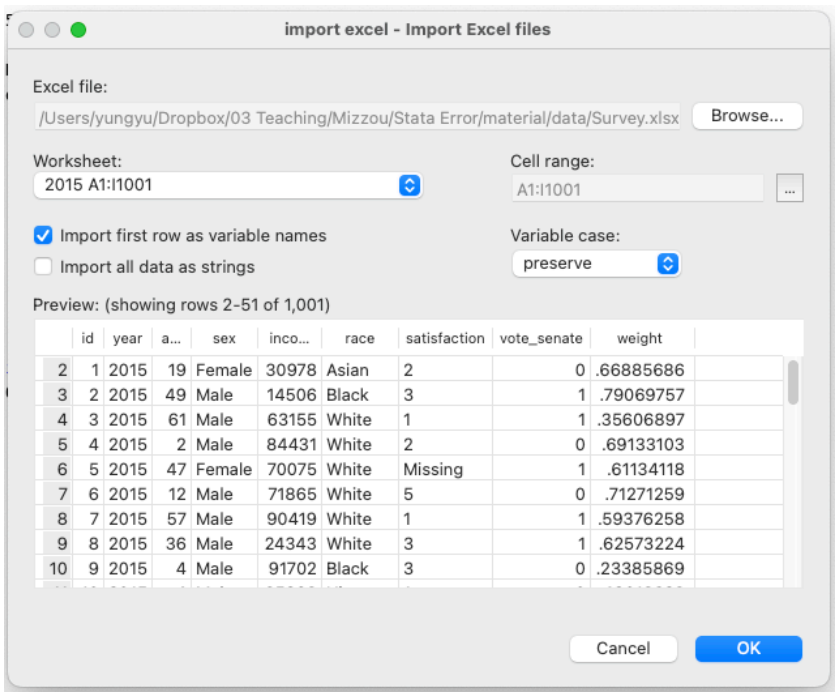
```
0 help import excel
```

import excel

File > Import > Excel spreadsheet (*.xls;*.xlsx)

Using the user interface menu is a helpful way to fix errors!





	id	year	a...	sex	inco...	race	satisfaction	vote_senate	weight
2	1	2015	19	Female	30978	Asian	2	0	.66885686
3	2	2015	49	Male	14506	Black	3	1	.79069757
4	3	2015	61	Male	63155	White	1	1	.35606897
5	4	2015	2	Male	84431	White	2	0	.69133103
6	5	2015	47	Female	70075	White	Missing	1	.61134118
7	6	2015	12	Male	71865	White	5	0	.71271259
8	7	2015	57	Male	90419	White	1	1	.59376258
9	8	2015	36	Male	24343	White	3	1	.62573224
10	9	2015	4	Male	91702	Black	3	0	.23385869

Outline

Observation issues

- Let's run a regression

```
76 reg vote_house age income satisfaction
```

```
no observations
r(2000);
```

- How about a correlation test?

```
77 corr vote_house age income satisfaction
```

```
(satisfaction ignored because string variable)
```

- How about summarize?

```
79 sum vote_house age income satisfaction
```

Observation issues

- In an analysis requiring numeric values, Stata would treat string variable as missing
- We need to **destring** the variable before include it in a regression analysis

```
81 destring satisfaction, replace force
82
83 reg vote_house age income satisfaction //Now
       it works!
```

Observation issues

- Let's combine the 2015 and 2016 data

```
85 append using "Survey 2015.dta" //Now let's
      append the 2015 data
86 corr vote_house vote_senate age income
      satisfaction //Let's run another correlation
      test; but we got an error
```

```
no observations
r(2000);
```

```
88 sum vote_house age income satisfaction
      vote_senate //Every variable has observations
```

Observation issues

- Stata considers **listwise** (**casewise**) when handing missing values.
- That is, the observations should have non-missing values for **ALL** variables!
- It takes forever to drop variables one by one to detect the problem
- We can use **misstable** to check which variables are causing the problems

Y	X_1	X_2	X_3
4	0.2	1.2	20
3	NA	1.2	21
2	0.3	1.1	16
2	0.4	1.1	17
1	0.5	2	18
2	0.4	2.1	18
NA	0.2	1.4	19
2	0.1	1.2	22
2	0.1	NA	NA

```
91 misstable patterns vote_house vote_senate age
     income satisfaction
```

Observation issues

Missing-value patterns
(1 means complete)

Percent	Pattern 1	2	3
<1%	1	1	1
49	1	1	0
48	1	0	1
2	0	0	1
1	0	1	0
100%			

Variables are (1) **satisfaction**
(2) **vote_house** (3) **vote_senate**

- How to read the table?
  1. There are three variables with missing values
  2. Only less than 1% have non-missing values among all three variables
  3. 49% of cases have missing in the third variables (vote_senate)
  4. 48% of cases have missing in the second variables (vote_house)
- Therefore, the problems were primarily caused by the second and third variables

Unbalanced quotes or parentheses

- When you have a really long code with a lot of parentheses

```
99 graph bar vote_house vote_senate, over(race,
     sort(vote_house) reverse gap(20)) bar(1,
     fcolor(maroon) lcolor(black) lwidth(medthin))
      bar(2,fcolor(forest_green) lcolor(black)
     lwidth(medthin) legend(order(1 "House" 2 "
     Senate") title(Turnout, size(median) color(
     black))) ylabel(0(0.2)1,angle(0) format(%4.1f
     )) title(Turnout by Race, size(medlarge))
     ytitle(Turnout) note(Source: A Fake Survey
     Data of 2015 and 2016)
```

parentheses do not balance

r(198);

Unbalanced quotes or parentheses

- Organize your code by using line breaks
  1. Use "///"

```
102 graph bar vote_house vote_senate, ///
103 over(race, sort(vote_house) reverse gap(20))
```

  2. Use "/* */"

```
114 graph bar vote_house vote_senate,/*
115 */over(race, sort(vote_house) reverse gap(20))
```

  3. Use #delimit to define the end of a command

```
126 #delimit ;
127 graph bar vote_house vote_senate,
128 over(race, sort(vote_house) reverse gap(20));
```

- Use ctrl + B or cmd ⌘ + B to find balanced parentheses

Unbalanced quotes or parentheses

- Use ctrl + B or cmd ⌘ + B to find balanced parentheses

```
104 bar(1,fcolor(maroon) lcolor(black) lwidth(
        medthin)) ///
```

- If it does not work, it means the balanced parenthesis is missing

```
105 bar(2,fcolor(forest_green) lcolor(black)
        lwidth(medthin) ///
```

A quick guide to loops

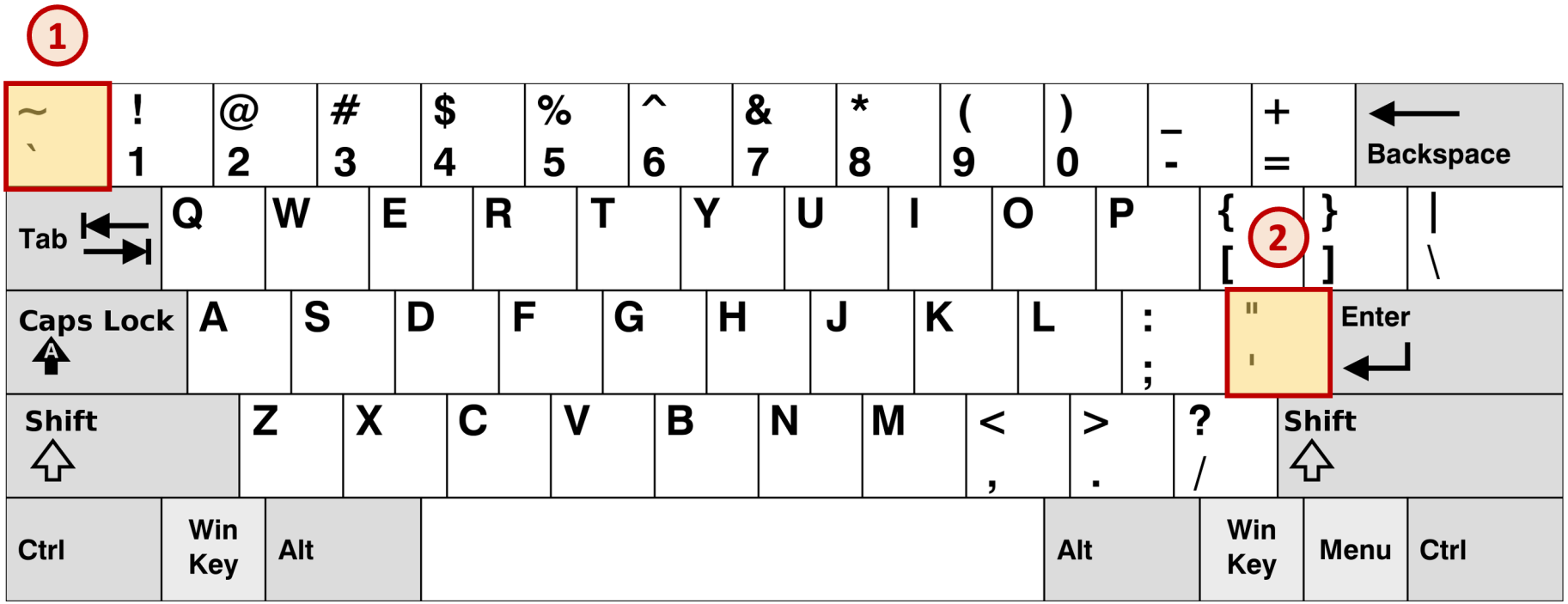
- Please turn to *Introduction_to_loop.do*

```
9  sysuse census, clear
```

```
21 forvalue i = 1(1)4{ //or "forv" as shortcut
22   sum pop medage death marriage divorce if
      region == `i'
23 }
```

```
36 foreach x in pop medage death marriage divorce{
37   tab region, sum(`x')
38 }
```

How to type 'x'


1
2
Backspace
Tab
Caps Lock
Enter
Shift
Shift
Ctrl
Win Key
Alt
Alt
Win Key
Menu
Ctrl

Unbalanced brackets in loops

- It's important to maintain indention
- Shortcut
  - Mac: Use tab →| or ctrl +] to indent right
  - Mac: Use ctrl + [to indent left
  - Windows: Use tab →| or ctrl + I to indent right
  - Windows: Use shift ⇧ + ctrl + I to indent left

```
141 foreach var in sex race {
142 encode `var', gen(`var'2)
143 foreach x in A B C{
144 foreach y in D E F{
145 forv z = 1(1)10{
146 {
147 }
148 }
149 }
150 }
```

Unbalanced brackets in loops

- Right Click > Preferences... > ☑ Indentation guide
- Use ctrl + B or cmd ⌘ + B to find balanced brackets

```
154 foreach var in sex race {
155   encode `var', gen(`var'2)
156   foreach x in A B C{
157     foreach y in D E F{
158       forv z = 1(1)10{
159       {
160       }
161     }
162   }
163 }
```

```
unexpected end of file
r(612);
```

Invalid syntax

The most tricky error!

```
invalid syntax
r(198);
```

Invalid syntax

- Common error 1: typo in command

```
169 label varialbe sex "Sex"
```

- Common error 2: extra or missing symbol

```
173 gen female = sex = "Female"
```

```
180 egen agegrp = cut(age),, group(10)
```

- Common error 3: missing space between code and comments

```
183 gen female = sex == "Female"// Some comments
      here
```

Invalid syntax

- Common error 4: missing quote for path with spaces

```
186 save Survey 2016.dta, replace
```

- Common error 5: use an undefined local value

```
190 forv i = 1(1)5{
191   gen sat`x' = satisfaction == `x'
192 }
```

```
195 sum satisfaction
196 forv i = `r(min)'(1)`r(max)'{
197   gen sat`i' = satisfaction == `i'
198 }
```

- Common error 6: use invalid variable name or expression

```
209 gen AVarWith@ = age > 5000%
```

Invalid syntax

- Solutions?
  - Be patient (Check and check again and again!)
  - Start from scratch. Don't copy and paste from other places (sometimes, there would be invisible symbols).
  - Seek help from others
  - Ask AI (such as ChapGPT) to debug for you

Error inside unshown details

- Now, we want to use a loop to clean the data for each year

```
216 forv i = 2015(1)2020{
217   import excel "Survey.xlsx", sheet(`i') clear first
218   destring satisfaction, replace force
219   foreach x in sex race{
220     encode `x', gen(`x'_encode)
221   }
222   foreach x in income age{
223     gen `x'2 = `x'^2
224   }
225   save "Survey_`i'.dta", replace
226 }
```

```
type mismatch
r(109);
```

- Stata provides no information on which variable of which year is causing the problem

- Tip 1: Use display, quietly, and noisily to make the progression of codes stands out

```
249 forv i = 2015(1)2020{
250   qui{
251   no dis "Clean data of year `i'"
252   import excel "Survey.xlsx", sheet(`i') clear first
253   destring satisfaction, replace force
254   foreach x in sex race{
255     no dis "- Encode variable `x'"
256     encode `x', gen(`x'_encode)
257   }
258   foreach x in income age{
259     no dis "- Generate squared term of variable `x'"
260     gen `x'2 = `x'^2
261   }
262   save "Survey_`i'.dta", replace
263   }
264 }
```

Error inside unshown details

- Tip 2: Use set trace on to show all details of outputs

```
270 set trace on //And then run the original code
```

```
– gen 'x'2 = 'x'^2
= gen age2 = age^2
type mismatch
r(109);
```

- If set trace on does not give you enough details, you can do set tracedepth #, where # is a number, the higher, the deeper

Outline

Ignore error

- Now let's fix the age variable issue in 2019

```
291 forv i = 2015(1)2020{
292   qui{
293   no dis "Clean data of year `i'"
294   import excel "Survey.xlsx", sheet(`i') clear first
295   destring satisfaction, replace force
296   foreach x in sex race{
297     no dis "- Encode variable `x'"
298     encode `x', gen(`x'_encode)
299   }
300   foreach x in income age{
301     no dis "- Generate squared term of variable `x'"
302     destring `x', replace force
303     gen `x'2 = `x'^2
304   }
305   save "Survey_`i'.dta", replace
306   }
307 }
```

Ignore error

Clean data of year 2020

- Encode variable sex

not possible with numeric variable

r(107);

- Age variable is numeric in most years but is a string variable in 2019
  - We put destring in the loop, and destring everything regardless of whether it is string or not
  - This is fine, because destring only gives us warning but not error
- Sex variable is a string variable in most years but is numeric in 2020
  - Why couldn't we just encode everything regardless of whether it is string or not?
  - Because encode gives us error when we try to encode a numeric variable
  - Couldn't we ask encode to behave like destring – only give us warning and not error?

Ignore error

- **capture**
  - **capture** executes command, suppressing all its output (including error messages)
  - The return code generated by command is stored in the scalar **_rc**
  - **capture** can be combined with { } to produce capture blocks
  - If you want **capture** to suppress only error but not output, combine it with **noisily**

```
323 capture no encode `x', gen(`x'_encode)
```

- **capture** only *suppress* error but not *fix* error
- You still need to check whether you need to manually fix the error!

Ignore error

- **capture** is useful when you know some of your models might fail
  - It might be fine because you only need the successful ones
  - You might want to revisit those models later and don't want them to break your loop

```
357 forv i = 2015(1)2020{
358   foreach y in vote_senate vote_house{
359     dis "Year `i', Dep. Var.: `y'"
360     logit `y' i.sex_encode i.race_encode age
      income satisfaction if year == `i'
361   }
362 }
```

```
no observations
r(2000);
```

Ignore error

```
366 eststo clear
367 forv i = 2015(1)2020{
368   foreach y in vote_senate vote_house{
369     dis "Year `i', Dep. Var.: `y'"
370     cap no logit `y' i.sex_encode i.race_encode
      age income satisfaction if year == `i'
371     eststo, title("`i'")
372   }
373 }
```

```
366 esttab, mtitle
367 est dir
```

Customize next step by error code

- Stata saved 12 models. Why?
- Stata's logic:
  1. Loop 1: Year 2015, Dep. Var.: vote_senate
     1. Successfully estimate the model
     2. Successfully store the result to eststo
  2. Loop 2: Year 2015, Dep. Var.: vote_house
     1. Fail to estimate the model because no observations
     2. eststo finding the last successfully estimated model to store
     3. Store the 2015 vote_senate results (this model being saved twice!)
- Could we run eststo only if the logit model succeed?

Customize next step by error code

```
386 if _rc == 0{
387   eststo, title("`i'")
388 }
```

- _rc == 0 means when there is no error occurs

Prevent error from the first beginning

- So far, we only fix the errors when they occur.
- Is there any way that we can prevent errors from the beginning?
- Especially if we know the error would occur if a given condition is not met

```
398 binscatter vote_senate age, rd(18) linetype(none)
```

```
command binscatter is unrecognized
r(199);
```

Prevent error from the first beginning

- We can check whether a command has already been installed before running it
- And if it's not, we can install it first

```
400 cap no which binscatter //check whether the
       command is installed
401 if _rc == 111{ //You can also use _rc != 0,
       error code of 0 means no error
402   ssc install binscatter
403 }
```

Prevent error from the first beginning

- Some other useful commands
- confirm

```
410 cap noi confirm file "figure"
411 if _rc != 0{
412   mkdir "figure"
413 }
```

```
417 confirm var vote_president
418 confirm numeric var sex
419 confirm string var age
```

- assert

```
423 assert sex == "Male" | sex == "Female" | sex == "
    Missing"
424 assert age > 17
```

Outline

1. Error vs. Warning: Why do we need to care about error?
2. Understanding: What do error messages and help files say?
3. Debugging: What if there are no clear error messages?
4. Handling: What if the errors are inevitable?
5. **Fixing: What if the errors come from the author of the command?**

Fixing errors from command but not user

- Sometimes the error is due to a *bug* of the command but not you
  - This usually happens on user-written command
  - But, here is an example of a bug of Stata build-in command

```
430 import delimited using "Survey_2017", clear
```

```
file _____ not found
r(601);
```

- Solutions:
  - Find a way to fix it out of the Stata (e.g., Excel or R)
  - Revise the command on your own
  - Report to the author (only when you are sure this is truly a bug but not your misuse of command)

Fixing bugs of command

- Let's find out the command of import delimited and fix it

```
442 which import_delimited
443 doedit "/Applications/Stata/ado/base/i/
    import_delimited.ado" //Revise this line to
    the path reported to you by "which
    import_delimited"
```

- In the *import_delimited.ado* file, edit the following lines

```
2   program define import_delimited2, rclass //
    change "import_delimited" to "
    import_delimited2"
```

```
143 pr.filename =  __import_check_using_file(
    filename, "") //delete ".csv" in the quote
```

Fixing bugs of command

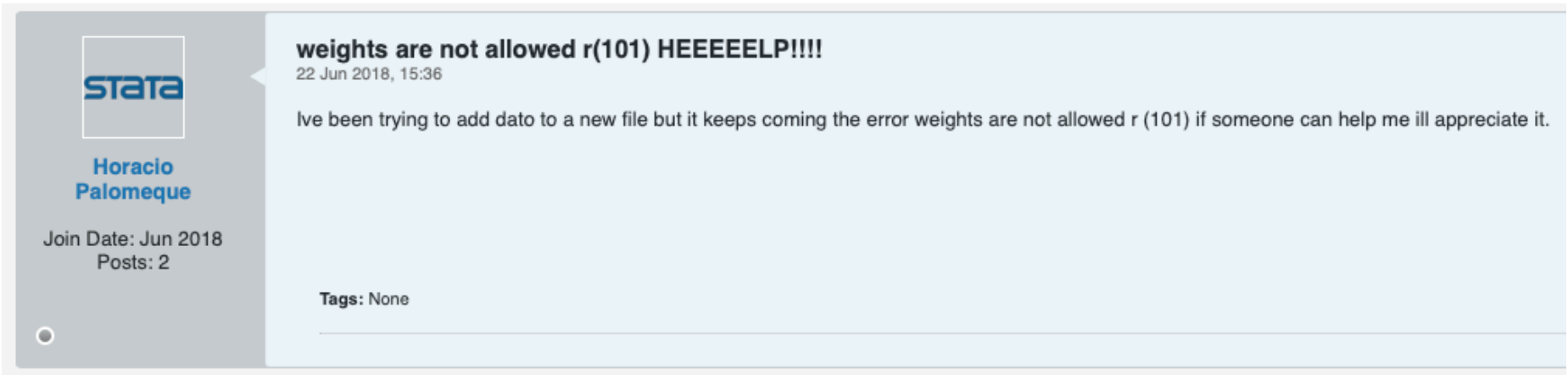
- Save this file as *import_delimited2.ado*
  - It is important that you must specify the ".ado" filename extension
- Now you can use this new command in your Stata:

```
450 import_delimited2 using "Survey_2017", clear //
      If errors occur, you might want to close and
      relaunch your Stata then re-run the code
```

How to seek help?

- Stata experts are not wizards!
  - Don't just say: this command does not work!
- Please provide enough information
  - The **complete and exact** codes you used
  - The **complete and exact** information that the Stata returned
  - Your system (Mac [Intel or M1 chip] or Windows), and your Stata version
  - (Sometimes might need) Your dataset
- Do not just provide a screenshot or photo
  - You don't want to waste the time of people who are helping you and expect them to manually type your codes into Stata to test
  - Some error is due to a missing or extra space, or a mistyped character. An eyeball check won't help.
  - It's okay to provide a screenshot in addition to your codes (typed)

A bad example



weights are not allowed r(101) HEEEEELP!!!!
22 Jun 2018, 15:36

Ive been trying to add dato to a new file but it keeps coming the error weights are not allowed r (101) if someone can help me ill appreciate it.

Tags: None

Horacio Palomeque

Join Date: Jun 2018
Posts: 2

- Without knowing what command you were using, it is impossible to detect the problem

Recap

1. Error vs. Warning: Why do we need to care about error?
2. Understanding: What do error messages and help files say?
3. Debugging: What if there are no clear error messages?
4. Handling: What if the errors are inevitable?
5. Fixing: What if the errors come from the author of the command?