PREVENTING ERRORS & DEBUGGING LOOPS IN STATA

A community resource
created for SOC 561
by Rina James

OVERVIEW

This worksheet is meant to give you hands-on practice checking for errors and debugging loops. It focuses on practical applications of four Stata commands: **trace**, **assert**, **pause**, and **capture**.

PREVIOUS RESOURCES

The exercises presented here build off the information and examples presented in previous community resources. If you have not already familiarized yourself with these resources, I recommend doing so before starting. You can view them using the links below:

- Loops continued: preventing errors and debugging
- Loops continued: preventing and debugging errors, year 2

INSTRUCTIONS

Below I present three exercises aimed at accomplishing different data management tasks. Each exercise uses a loop (or loops) that contain an error. Use the previous community resources and other class materials to identify the error and identify how the four Stata commands this worksheet covers help in accomplishing the outlined task. Solutions are presented at the end of the worksheet.

GETTING STARTED

The problems below all use variables included in the 2016 cross-section of the General Social Survey (GSS). To download the data:

1. Visit: http://gss.norc.org/get-the-data/stata

2. Scroll down to the second box titled 'Download Individual Year Data Sets'
3. Click '2016'
4. Save the .zip file
5. Extract and open the data in Stata

PROBLEM #1A

The GSS contains 7 variables on abortion attitudes: **abany**, **abdefect**, **abhlth**, **abnomore**, **abpoor**, **abrape**, and **absingle**. **Abany** indicates whether a respondent supports the right to have an abortion for any reason. Logically, you expect that respondents who gave a 'yes' response for **abany** should have also answered 'yes' to the other questions asking whether a woman should have the right to an abortion in more specific circumstances. You want to test this expectation using the following loop:

```
foreach var in abdefect abhlth abnomore abpoor abrape
absingle {
     replace abmatch=1 if abany=1 & `var'!=1
     }
```

However, when you try to run the loop Stata gives you the following error message:

```
invalid syntax
```

Using one of the four commands emphasized in this worksheet, identify and correct the problem.

PROBLEM #1B

Rewrite the loop above so that it accomplishes the same check without creating a new variable.

PROBLEM #2

Now imagine that you want to construct a scale indicating a respondent's level of science knowledge. For this, you will use the variables **electron**, **evolved**, **hotcore**, **lasers**, **radioact**, **solarrev**, and **viruses**. After looking at the coding for these variables, you notice that ‘true’ responses are coded 1 and ‘false’ responses are coded 2, and that because of how the questions are worded 1 actually indicates correct science knowledge and 2 indicates incorrect knowledge for **electron**, **evolved**, and **hotcore**. Scales are easier to interpret when a higher value indicates a greater amount of what is being measured (in this case science knowledge), so you decide you will use a loop to recode those three variables so that ‘true’ is a higher value than ‘false.’

You know that best practice is to never recode existing variables, so you write the following loop to generate a set of new variables, recode them, and apply value labels for true or false:

```
foreach var in electron evolved hotcore {
     gen `var'1=`var'
     recode `var'1 (1=2)(2=1)(*=.)
     label def truefalse 1 False 2 True
     label val `var'1 truefalse
     }
```

However, when you try to run the loop, Stata generates the following output:

```
. foreach var in electron evolved hotcore {
  2.          gen `var'1=`var'
  3.          recode `var'1 (1=2)(2=1)(*=.)
  4.          label def truefalse 1 False 2 True
  5.          label val `var'1 truefalse
  6.          }
(1,861 missing values generated)
(electron1: 2867 changes made)
label truefalse already defined
```

Diagnose the problem, and identify which debugging or error prevention command you could use to make this loop work.

PROBLEM #3

Next, you want to create interaction terms for sex and self and family education, using a dummy variable indicating if a respondent, their parents, or their spouse has a 4-year college degree or greater.

First, you need to create dummy variables for education. Then you will create your interaction term and check your work. You write the following loops – the first to create the dummies, the second to generate the interaction term, and the third to check your work – and execute your do-file.

```
foreach var in codeg coldeg1 degree madeg padeg spdeg {
     gen `var'_dummy=0 if `var'!=.
     replace `var'_dummy=0 if `var'==2/7
     }

local i = 0
foreach var in codeg_dummy coldeg1_dummy ///
degree_dummy madeg_dummy padeg_dummy spdeg_dummy {
     local i=`i'+1
     gen edxsex`i'=`var'*sex
     }

local i = 0
foreach var in codeg_dummy coldeg1_dummy degree_dummy
madeg_dummy padeg_dummy spdeg_dummy {
     local i=`i'+1
     bysort sex: tab `var' edxsex`i'
     }
```

When you generate your tables, you see that everything has been coded as zero. Identify the best tools for troubleshooting and correct the error in the code.

SOLUTIONS

NOTE

In some cases, there is more than one way to adequately approach troubleshooting. The point of these exercises is to help you get practice approaching problems in your code – if the solution you would try is not exactly the same as that presented, simply ensure you're considering what approach is the most efficient.

PROBLEM #1

You may have been able to spot the syntax error here by sight. However, the best way to troubleshoot this using the above would be to use **set trace on.** The **trace** command presents a display for each line of the loop as Stata completes each evaluation. In this case, it shows which part of the syntax is invalid:

```
. set trace on

.
. foreach var in abdefect abhlth abnomore abpoor abrape
absingle {
  2.          replace abmatch=1 if abany=1 & `var'!=1
  3.          }
- foreach var in abdefect abhlth abnomore abpoor abrape
absingle {
- replace abmatch=1 if abany=1 & `var'!=1
= replace abmatch=1 if abany=1 & abdefect!=1
invalid syntax
```

This shows that the error lies in the **replace** portion of the code. Looking more closely (or referencing **help replace** if necessary) we can see that the **abany=1** condition is missing an equal sign. The correct code is:

```
foreach var in abdefect abhlth abnomore abpoor abrape
absingle {
     replace abmatch=1 if abany==1 & `var'!=1
     }
```

Bonus answer: Instead of creating a dummy variable, you could use **assert** to verify whether or not this condition was true using the following loop:

```
foreach var in abdefect abhlth abnomore abpoor abrape
absingle {
     assert abany==1 & `var'!=1
}
```

The results indicate that this assertion is not true for a large number of the observations.

PROBLEM #2

The error message generated when running this loop tells us that the problem is the **label def** line of the code. If you look at the bottom of the variable list in Stata, you will see that **electron1** and **evolved1** were created, but that the loop broke after recoding **evolved1**. Just like in Problem #1, you can use **trace** to identify which step in the iteration the loop breaks, but first you should drop the new variables and value labels that were successfully created before the loop broke:

```
drop electron1 evolved1
label drop truefalse

set trace on

foreach var in electron evolved hotcore {
     gen `var'1=`var'
     recode `var'1 (1=2)(2=1)(*=.)
     label def truefalse 1 False 2 True
     label val `var'1 truefalse
     }
```

The **trace** output (omitted) shows that the first two variables are being created and recoded, but that the loop breaks when it attempts to define the value labels after generating **evolved1**. This is because the value labels were already created during the loop's first iteration. You could simply remove this line from the loop and create the value label before running the loop, but an easier solution is to simply use the **capture** command to prevent the loop from breaking after the labels have been defined in the first iteration:

```
foreach var in electron evolved hotcore {
     gen `var'1=`var'
     recode `var'1 (1=2)(2=1)(*=.)
     capture label def truefalse 1 False 2 True
     label val `var'1 truefalse
     }
```

After adding **capture**, the loop runs correctly.

PROBLEM #3

This is a section of code with several loops, so the easiest way to check the error is to re-run after using pause to break up the chunks of code and review the results. I start with a pause after creating dummy variables:

```
foreach var in codeg coldeg1 degree madeg padeg spdeg {
     gen `var'_dummy=0 if `var'!=.
     replace `var'_dummy=1 if `var'==2/7
     }

pause on just created dummy variables

local i = 0
foreach var in codeg_dummy coldeg1_dummy ///
degree_dummy madeg_dummy padeg_dummy spdeg_dummy {
     local i=`i'+1
     gen edxsex`i'=`var'*sex
     }

local i = 0
foreach var in codeg_dummy coldeg1_dummy degree_dummy
madeg_dummy padeg_dummy spdeg_dummy {
     local i=`i'+1
     bysort sex: tab `var' edxsex`i'
     }
```

Pausing makes it apparent that something has gone wrong with the creation of the dummy variables; after running the first loop, Stata is displaying

```
(0 real changes made)
(0 real changes made)
(0 real changes made)
(0 real changes made)
(0 real changes made)
(0 real changes made)
```

Since there is no error message being generated, the problem must be with the way the replace command has been written. Some further investigation will show that the range has not been correctly specified for the replace command and that the / is meant for specifying ranges using recode. You could use | conditions for each value you want to include, but the simplest correction is to change the code from

```
     replace `var'_dummy=0 if `var'==2/7
```

to

```
     replace `var'_dummy=1 if inrange(`var’,2,7)
```

and rerun your code.