CEQ Manual for writing Do Files

April 2017

Contents

1 Introduction

This manual is intended to assist the construction of an Stata dofile for a CEQ analysis. Below are the sections that tend to fall in a CEQ do file by order of which they appear. Note that not every section falls in each do file as some depend on the specific function of the do file. This manual was based on the set of do files that contributed to the project "Taxes and Social Spending in Brazil" but can be followed to construct any CEQ do file.

2 Folder Orgnanization

When working on a project it is important to have files organized in folders that are clearly named. In a typical project for CEQ you will have the following folders:

2.1 /adofiles

User-written programs called by the do files are stored here. In the master dofile you will need to create a path to this folder for the user-written functions to work.

2.2 /data

Raw data to be cleaned and processed.

2.3 /documentation

Original documentation of the raw data is in documentation/. For example, this could be the a survey that was administered for the anaylsis.

2.4 /dofiles

The do files to replicate results in Stata.

2.5 /graphs

Graphs produced by the do files and used in the paper.

2.6 /logs

Logs generated by the do files. The logs capture everything that appeared on the Stata console while the log and do file were running.

2.7 /proc

The processed data produced by the do files.

3 Do File Organization

3.1 Do Files

One important thing to do in order to keep the project organized is to construct do files with the purpose of performing a specific function or closely related functions. For example, one do file might prepare the data for the analysis while another is dedicated to developing graphs. This way it will be much easier to find any bugs or make quick adjustments to the analysis.

Once you have constructed do files with specific purposes for the project, it is common practice to have a `0\_master.do` do file that can call each of the other do files. In the `0\_master.do` establish the a global directory, which is used to access the sub folders described above and create a path to the /adofiles folder so that the user-written functions can be accessed. Thus when someone is trying to replicate the results the `0\_master.do` is the only place where anything needs to be changed as long as your folder structure within the base directory matches that of the original.

3.2 Within a Do File

While each do file varies in the sections it contains there are certain sections that consistently appear in do files. The appear according to the following order:

- README
- Directories (0_master.do only)
- Preliminaries (0_master.do only)
- Log
- Locals
- Do (0_master.do only)
- Prelimiary Programs
- Read and Clean Data
- Data Preparation
- Final Cleaning
- Save
- Wrap Up

4 Format of Section Titles

Each section title follows the same format. The entire title is uppercase and is bordered by asterisks, which is demonstrated below

```
**********************
** [SECTION TITLE] **
**********************
```

5 Introduction to Do File

At the top of the do file, include the title of the do file followed by the authors including a contact and then the date of the last revision. Each piece of information occupies its own line.

The introduction follows the following format:
Title
dofile created by [authors], email
last revised [date]

Below, at the top of the code sample, is an example of an introduction to a do file.

```
/*******************************************************************************
POF 2008-2009 DATA PREPARATION
do file created by Sean Higgins, shiggins@tulane.edu
last revised Apr 26, 2015

**************
** READ ME: **
**************
1. These do files were created for the following papers;
       if using the data resulting from this do file please
       cite all of the following:

Higgins, Sean and Nora Lustig. 2016. "Can a Poverty-Reducing and Progressive Tax
  and Transfer System Hurt the Poor?" Journal of Development Economics 122, 63-75.
  doi:10.1016/j.jdeveco.2016.04.001

Higgins, Sean, Nora Lustig, Whitney Ruble, and Timothy M. Smeeding. 2015.
  "Comparing the Incidence of Taxes and Social Spending in Brazil and the United
  States." Review of Income and Wealth. doi:10.1111/roiw.12201

Higgins, Sean and Claudiney Pereira. 2014. "The Effects of Brazils Taxation
  and Social Spending on the Distribution of Household Income." Public Finance
  Review 42, 346-367.

*******************************************************************************/
```

6 README

In the example posted above you can also see the README section, which is the first section of any do file. This is where one must describe the reason for the do file. Here one usually lists the papers for which the paper was created, doing so by listing the papers' citations. Finally, this section also includes any information the user must know before running the do file.

7 Directories

Directories use global variables to set up paths to folders that will be used to access and save data throughout the analysis. They also load in user written ado files that will be used in throughout the project. Directories are generally established in the master do file only and in this manner they are passed on to the other do files that are called.

```
*****************
** DIRECTORIES **
*****************
global base = "C:/Dropbox/Submissions/JDE_2015" // !change! if different computer
adopath ++ "$base/adofiles" // load in user-written ado files used by these programs
```

8 Preliminaries

Here the user establishes any requirements needed to run the file. For example, the user would declare the version of the Stata being used.

```
******************
** PRELIMINARIES **
******************
version 11
```

9 Log

This marks the first step towards running the actual content of the do file. Begin by running the user written time function and setting a local variable to the name of the do file. These will both be used in the title of the log. Then stop any log that may have been running beforehand using capture log close and initiate a new log file to be stored in the logs sub folder of the main project folder. This log file will capture all the subsequent processes in the file. Once the log has been initialized, display the date and time, generated by the time function, and the current working directory.

```
*********
** LOG **
*********
time
local project 1_pof_dataprep
capture log close
log using "$base/logs/‘project’_‘time’.log", text replace
di "‘c(current_date)’ ‘c(current_time)’"
pwd
```

10 Locals

In this section declare local variables to store values that will be needed later on in the do file. In the master do file the user uses local variables to indicate which do files to run.

```
************
** LOCALS **
************
local xlist BC SA1 SA2 norent // scenarios:
       // Benchmark case (contributory pensions as deferred income, not government transfer)
       // Sensitivity analysis 1 (contributory pensions as government transfer, contributions as tax)
       // Sensitivity analysis 2 (special circumstances pensions and contributory pensions as deferred income)
       // Benchmark case excluding imputed rent
// For variable labels:
local l_BC "benchmark case"
local l_SA1 "sensitivity analysis 1"
local l_SA2 "sensitivity analysis 2"
local l_norent "benchmark case excluding imputed rent"

// Age groups
local ages cri adol adulto mayor
local l_cri "Child"
local l_adol "Adolescent"
local l_adulto "Adult"
local l_mayor "Elderly"
```

11 Do

The Do section is specific to the master do file. Conditioned on the local variables declared in the locals section this section runs the do files with more specific functions for the project. Before calling a do file make sure to list the inputs and the outputs of the file.

```
********
** DO **
********

if `3_fiscal_impoverishment' do "$base/dofiles/3_fiscal_impoverishment.do"

** 4. 4_robustness // ROBUSTNESS CHECKS INCLUDED IN FOOTNOTES OF PAPER
       // Produces numbers that are cited in the paper's robustness checks
       ** INPUTS
       **  proc/pof2.dta
       ** OUTPUTS
       **  Numbers used in paper printed as Stata output
       ** ADO FILES REQUIRED
       **  -time- (Sean Higgins)
```

12 Preliminary Programs

Any programs that will be utilized through the file should be written here towards the beginning of the do file.

```
*************************
** PRELIMINARY PROGRAMS **
*************************
// Create dummy variables for purchase types in expenditure data
capture program drop create_purchase_types
program define create_purchase_types, rclass
       syntax anything
       gen byte boughtforUC=(aquisicao==1 | aquisicao==3 | aquisicao==5)
       gen byte boughtforother=(aquisicao==2 | aquisicao==4 | aquisicao==6)
       gen byte donated=(aquisicao==7)
       gen byte retirada=(aquisicao==8)
       gen byte trade=(aquisicao==9)
       gen byte ownproduction=(aquisicao==10)
       gen byte outra=(aquisicao==11)
       foreach p of local anything {
              gen double v_`p' = 0
              format v_`p' %16.2f
              replace v_`p' = valor_an if `p'==1
              by code_uc: egen double vuc_`p' = sum(v_`p')
              format vuc_`p' %16.2f
              local keeplist `keeplist' vuc_`p' // iteratively create list of variables to
                     // keep below
       }
       return local keeplist `keeplist'
end
```

13 Read and Clean Data

Load in and clean the raw data files that are stored in the /data sub folder. If one has difficulty in this section one should be able to refer to the documents in the /document sub folder to clarify the structure of the data. Use the the infix command to specify how the data will be read into Stata. After loading in the data label and format the variables accordingly. Here the user puts the data into the right structure for the analysis. For example, the user will address the coding of missing values and outliers. Below is an example of using infix and a dicitonary to read in a text file.

```
** REGISTRO:  CONDIES  DE VIDA - POF6 (tipo_reg=04)
** (SUBJECTIVE WELL-BEING)
clear
#delimit ;
infix
           tipo_reg                          1-2   /* tipo de registro // type of register */
              UF                             3-4   /* codigo da unidade federativa // state code */
           num_seq                           5-7   /* numero sequencial // sequential number in household */
           num_dv                            8     /* digito verificador do sequencial // verification digit */
           num_dom                           9-10  /* numero do domicilio // dwelling number */
   double  code_dom                          3-10   /* unique dwelling ID */
           num_uc                            11     /* numero da unidade de consumo // household number */
   double  code_uc                           3-11  /* unique household ID */
           num_informante                    12-13  /* numero do informante // respondent number */
   double  code_pessoa                       3-13  /* unique person ID */
           estratogeo                        14-15  /* estrato geografico // geographic strata */
           renda_subj                        44     /* codigo da renda familiar // family income code */
           qtd_alimento_subj                 45     /* quantidade de alimento // quantity of food */
           tipo_alimento_subj                46     /* tipo de alimento // type of food */
       using "$base/data/POF0809/T_CONDICOES_DE_VIDA_S.txt"
;
#delimit cr
```

Here is an exmaple of the processes one may peform when cleaning the recently loaded data.

```
label var deflator      "deflator"
label var produto_imp   "imputed product"
format weight1 %14.8f
format weight2 %14.8f
format code_dom %8.0f
format code_uc %9.0f
format code_produto %7.0f
format valor %11.2f
format deflator %5.2f
drop if num_item==99901
replace valor=0 if valor==999999.99 // this is how missing values are coded
gen double valor_an=valor*deflator*anualizacao
format valor_an %16.2f
sort code_uc
create_purchase_types `purchase_types'
local keeplist `r(keeplist)'
tempfile despesav
save `despesav', replace
by code_uc: drop if _n>1 // keep only one observation per household
       // (faster than duplicates drop)
keep code_uc `keeplist'
foreach p of local purchase_types {
       rename vuc_`p' vuc_`p'_51 // 51 because this data set corresponds to
              // box 51 of the POF questionnaire
}
tempfile despesa_51
save `despesa_51', replace
```

14 Data Preparation

Once the user has loaded and cleaned all the raw data she must prepare the data such that it is in the right format for the analysis. This includes ensuring an observation does not include a external member of the household and that the data is at the household level. It also entails merging together the separate pieces of household data to construct the processed data set.

```
**********************
** DATA PREPARATION **
**********************
use `pessoas'
sort code_uc
by code_uc: drop if _n>1 // keep only one observation per household
       // (faster than duplicates drop)
count
keep code_uc weight2
duplicates report code_uc
merge 1:1 code_uc using `condicoes'
       ** 670 not matched from master (prob didn't fill out POF6)
       ** DON'T DROP THE NON MATCHES
gen byte noPOF6=(_merge==1)
drop _merge
merge 1:1 code_uc using `despesa_0609'
       ** 1048 not matched from master DO NOT DROP
foreach x of local purchase_types {
       replace vuc_`x'_0609=0 if vuc_`x'_0609==.
}
```

15 Data

Load in the processed data.

```
**********
** DATA **
**********
use "$base/proc/pof1.dta", clear
```

16 Final Cleaning

Keep the relevant variables for the analysis and perform the final labeling of values and variables that may be necessary.

```
********************
** FINAL CLEANING **
********************
label var i "Include (do not drop) dummy"
describe
keep UF-renda_subj i BF_imp pc_BolsaFamilia pc_total_transfers pc_total_taxes Y*_* y*_*
describe
```

17 Save

Save the processed databases and store them in the /proc sub folder.

```
**********
** SAVE **
**********
save "$base/proc/pof1.dta", replace
```

18 Wrap Up

Close the log file and exit Stata.

```
*************
** WRAP UP **
*************
log close
exit
```

19 Coding Practices

The following section will cover coding practices that are expected of any CEQ researcher.

19.1 Indentation

In some programming languages such as Python, it necessary to indent after certain statements for the program to function. The most common incidents of this is after for statements and if statements. While this is not necessary for a do file to run, it tidies the code and greatly assists other researchers in understanding the do file. After an if statement, for or while statement, first line break, and anywhere else deemed necessary the Stata Programmer should indent the next line of code. Below is an example of using indentation in loops and conditional statements.

```
foreach x in taxes transfers {
	gen pc_total_`x' = 0
	foreach var of local `x' {
		if strpos("`var'","vuc_") {
			local suffix = subinstr("`var'","vuc_","",.)
			cap drop pc_`suffix'
			gen pc_`suffix' = `var'/members
			local var pc_`suffix'
		}
		replace pc_total_`x' = pc_total_`x' + `var'
	}
}
```

19.2 Line Breaks

When multiple line breaks will be needed for a command, the practice is to use #delmitr ; as demonstrated below.

```
#delimit ;}
local purchase_types
	boughtforUC
	boughtforother
	donated
	retirada
	trade
	ownproduction
	outra
;
local need_later
	despesa90
	despesa12
	despesa7
	despesao
	despesav
	despesa_ind
	bensduraveis
	domicilio
	pessoas
;
#delimit cr
```

If there is only one long line that is needed to be split then it is better to use \\\ to perform the line break.

19.3 Temprary Files and Variables

Throughout the many stages of building do files for a project one will find it useful create temporary databases. Unless this additional information is necessary for tasks in a future do file it is recommended that the programmer use `tempfile` when creating these databases so that they are immediately erased once the program ends. Below is an example of how this can be done.

```
tempfile pessoas // full data set of individual-level characteristics
tempfile headetc // this will be a data set with just two variables:
       // the individual ID number and the variable indicating their household
       // position (head of household, spouse of head, etc.)
save 'pessoas', replace
keep code_pessoa head_etc
       ** 377 have head_etc>=6 (boarders, domestic servants, and their families; to be dropped)
sort code_pessoa
save 'headetc', replace
```

This can also be done with variables. When working with a data set, one may need to use a variable to assist the task at hand but not need them permanently in the data set. Using `tempvar` can easily facilitate this.

19.4 Commenting

For interpretability, one should comment their do file as much possible. Before a command or after a command on the same line write your comment following the //.