Replication: Friedberg and Isaac (2022)

ECON 7670, Spring 2023
Instructor: Elliott Isaac

Name: ____

The intent of this assignment is to build upon previous work with Stata's variable generation, regression, and table and figure automation functions.

Grading

The grading rubric is below:

	5 points	2 points	1 point	0 points
Stata replication	I am able to completely replicate your log file by changing to my own directory and hitting "Do" without any additional work	I am able to completely replicate your log file by changing to my own directory and hitting "Do" after correcting a few mistakes	I am able to replicate parts of your log file after correcting several mistakes	I am not able to replicate your log file
Replication accuracy	▨	The output in your log file perfectly replicates the analysis and table and figure formatting	The output in your log file is close, but not identical to, the analysis and/or table and figure formatting	The output in your log file is very different than the analysis and/or table and figure formatting
LaTeX integration	I can perfectly construct a PDF from your LaTeX file by hitting "Compile" without any additional work	I can construct a PDF from your LaTeX file by hitting "Compile" after correcting a few mistakes	I can construct parts of a PDF from your LaTeX file by hitting "Compile" after correcting several mistakes	I am not able to construct a PDF from your LaTeX file

Submission

You will submit the following through Canvas:

1. Your .do file named lastname.do
2. Your .log file named lastname.txt
3. Your compiled LaTeX file named lastname.pdf

You will **not** submit the tables and figures you export from Stata. I will recreate your tables and figures by executing your .do file. You will also **not** submit the .tex file you use to compile your PDF (I have a copy). I will recreate your PDF by compiling the .tex file with the tables and figures your code generates.

Introduction

In our last replication assignment, we used estout and twoway commands to automatically build and export tables and figures in Stata that would be imported directly into LaTeX. This assignment builds upon our previous work by adding complexity to the tables and figures we create.

Part 1: Preparing Your Workspace

Concepts in this section:
- Directory
- File path
- Do-file
- Log file

Stata commands used in this section:
- cd (change directory)
- log using
- log close

Create a folder on your computer where you will store the raw data and all of your files for this assignment. This folder is called your **directory**. The location of your directory is called the **file path**. For example, my directory folder is called "clk2009_replication" and it is stored on my desktop, so the file path is:

```
/Users/elliottisaac/Desktop/fi2022_replication
```

In your directory, do the following:
1. Save your .do file as lastname.do
2. Save the dataset
3. Save your .tex file as lastname.do

In your .do file, do the following:
1. Copy your file path into the `cd` command
2. Change the name of the .log file to your own last name

Part 2: Building the Dataset

Stata commands in this section:

`use`
`gen` with `()` function (true/false)
`label variable`
`egen` with `rowmax` and `rowmin` functions

In your .do file, write code that:

1. Loads your dataset using the `use` command.
2. Generates the following variables:

Variable name	**Definition**
r_posincearn	Variable equal to 1 if *r_incearn* is greater than 0.
sp_posincearn	Variable equal to 1 if *sp_incearn* is greater than 0.
c_anychildren	Variable equal to 1 if *c_children* is greater than 0.
*c_incearn_split**	The fraction of couple income (*c_incearn*) that is earned by the higher earner. There is a *c_incearn_split* variable based on *r_incearn*, *sp_incearn*, and *c_incearn*; and *c_incearn_hat_split* variable based on *r_incearn_hat*, *sp_incearn_hat*, and *c_incearn_hat*. **Note**: Either *r_incearn* or *sp_incearn* may be larger. If *c_incearn* is 0 then *c_incearn_split* should be 0.
r_male	Variable equal to 1 if couple is male.
c_racecomp	Variable equal to 1 if *r_race* and *sp_race* are the same.
c_agemax	Variable equal to the age of the oldest partner.
c_agemin	Variable equal to the age of the youngest partner.
c_agediff	Variable equal to the age difference between the oldest and youngest partner.
c_edumax	Variable equal to the years of education of the most educated partner.
c_edumin	Variable equal to the years of education of the least educated partner.
c_edudiff	Variable equal to the difference in years of

	education between the most and least educated partner.
c_incearn_split_cond	Variable equal to *c_incearn_split*, defined only for observations with *c_incearn* greater than 0.
c_incearn_split_hat_cond	Variable equal to *c_incearn_hat_split*, defined only for observations with *c_incearn* greater than 0.
*c_incearn** multiples	Variables equal to *(c_incearn/10000)* to the 2, 3, 4, and 5 power.
*c_incearn_hat** multiples	Variables equal to *(c_incearn_hat/10000)* to the 2, 3, 4, and 5 power.

Part 3: Calculating summary statistics for Table 2

Stata commands in this section:

`summarize` with the `bysort`, `eststo`, and `estpost` prefixes

We previously used the `eststo: estpost summarize` prefixes and command to store the `summarize` output as a regression output in Stata’s memory so we can use it with `estout`.

Now we will add the `bysort` *`varlist`* prefix, which will allow us to compute summary statistics over separate groups (in this case, married or cohabiting couples). `bysort` will run the specified command separately for each distinct group within *varlist*.

In your .do file, write code that:

1. Calculates summary statistics for *r_posincearn*, *r_posincearn_hat*, *c_incearn*, *c_incearn_hat*, *c_incearn_split_cond*, *c_incearn_split_hat_cond*, *marriage_subsidy*, *marriage_subsidy_hat*, *fed_marriage_subsidy*, *fed_marriage_subsidy_hat*, *st_marriage_subsidy*, and *st_marriage_subsidy_hat* separately for married and cohabiting couples using the `bysort` *`groupvarname`*`: eststo: estpost summarize` command

The `bysort` prefix starts at the lowest value of *varlist* and works up, so it will first calculate summary statistics for cohabiting couples (*sscouple_mar* = 0) and save the result as est1, and will then do the same for married couples (*sscouple_mar* = 1) and save the result as est2.

Part 4: Running regressions and storing statistics for Table 3

Stata commands in this section:

- `reg` with `eststo` prefix and `vce(robust)` option
- `ivreg2` with `eststo` prefix and `partial`, `robust`, `ffirst`, and `savefirst` options
- `summarize` with `e(sample)` if statement
- `lincom`
- `eststo` with `add` option

The dependent variable is *sscouple_mar*, but the set of control variables differ. Note that the last specification uses a long list of variables that have already been coded for you as `local lassovars`. You can refer to this group of variables by typing `` `lassovars’ ``, but it is saved as a local which means that you must also execute the `local lassovars` code at the same time you want to call `` `lassovars’ `` (i.e., Stata “forgets” what `` `lassovars’ `` contains every time it stops running code, so you need to “remind” it each time).

OLS Regressions

In your .do file, write code that:

1. Runs the first OLS regression in Table 3, Column 1 using the `reg` command with `robust` standard errors.
2. Summarizes the outcome variable using the `summarize` command conditioning on having been in the sample.
3. Stores the regression output and the mean of the dependent variable using the `eststo` command with the `add` option (the mean of the dependent variable is stored as **r(mean)** after you run `summarize`).

Repeat steps 1-3 for the other OLS regressions. Since the only difference in coding for the different regressions is the control variables, see if you can use a `foreach` loop so that you only type the `reg`, `summarize`, and `eststo` commands once.

IV Regressions

We need to store additional statistics for the IV regressions, which come from the first stage estimates.

Storing First Stage F-statistics

`ivreg2` stores a matrix called *e(first)* that contains test statistics from the first stage regression. One of the statistics is the

Sanderson and Windmeijer (2016) F-statistic from the first stage coefficient, which we report at the bottom of Table 3. *e(first)* stores this statistic in the row labeled SWF (row 8) and the column labeled marriage_subsidy (column 1). You can see this number by typing `display e(first)[8,1]`, which tells Stata to display the entry in the *e(first)* matrix that is in row 8, column 1.

A few notes about referring to matrices:

- The *e(first)* matrix is the first stage matrix for the currently active regression output, so make sure the correct regression output is active.
- It is possible that Stata stores the statistic in a different row (not row 8), but it will still label it "SWF". The function `rownumb(e(first),"SWF")` will equal the row number in the *e(first)* matrix that is labeled "SWF", which you may also use to specify the row just in case the matrix stored it in a different row (this is good practice if you know the statistic you need and know how Stata labels it to make sure you capture the correct statistic every time regardless of the row it's stored in).

In your .do file, write code that:

1. Runs the first IV regression in Table 3, Column 2 using the `ivreg2` command with `robust` standard errors and the `savefirst` option.
2. Summarizes the outcome variable using the `summarize` command with the `e(sample)` if statement and saves it as a local (you will refer to this local later).
3. Stores the first stage F-statistic from the *e(first)* matrix as a local (you will refer to this local later).

Storing First Stage Coefficients, Standard Errors, and P-values

The first stage coefficients, standard errors, and p-values (and subsequent significance stars) must be captured from the first stage regression. Using the `savefirst` option with `ivreg2` will save the first stage regression as *_ivreg2_marriage_subsidy*.

We need the coefficient, standard error, and p-value (so we can convert it into significance stars). The easiest way to capture these from the same source is to type `lincom marriage_subsidy`, which effectively performs a test of whether the coefficient is equal to 0 (which what Stata displays by default in regression output). `lincom` stores the requested estimate as **r(estimate)**, stores the relevant standard error as **r(se)**, and stores the relevant p-value as **r(p)**.

I use a trick with `estout` to use the p-value to create significance stars in Stata. I use `if` and `else if` statements to define a local called star that is equal to 9999999999 if $0.05 < \mathbf{r(p)} \leq 0.10$, equal to 999999999 if $0.01 < \mathbf{r(p)} \leq 0.05$, equal to 99999999 if $\mathbf{r(p)} \leq 0.01$, and equal to 9999999 otherwise. Then I use the replace option in `estout` to:

```
substitute(9999999999 “***” 999999999 “**” 99999999 “*” 9999999 “ “)
```

In your .do file, write code that:

1. Restores the first stage regression.
2. Stores the first stage coefficient estimate from the lincom command as a local.
3. Stores the first stage coefficient standard error from the lincom command as a local.
4. Stores the first stage coefficient p-value from the lincom command as a local (following above definition).
5. Stores the regression output with the following additional statistics using the `eststo` command with the `add` option:
   a. Mean of the dependent variable
   b. First stage F-statistic
   c. First stage coefficient estimate
   d. First stage coefficient standard error
   e. First stage coefficient p-value

Repeat the above steps for the other IV regressions. Since the only difference in coding for the different regressions is the control variables, see if you can use a `foreach` loop so that you only type the `reg`, `summarize`, and `eststo` commands once.

Important:

The `savefirst` option saves over any existing first stage regressions, so the stored first stage regression you have will only be the most recently saved one. For this reason, you need to write code that will store the first stage statistics immediately after running the relevant regression before estimating the next IV regression.

Running the control function specification

The last IV specification in Table 3, column 6 uses a long list of variables that have already been coded for you as `local lassovars`. Running a standard IV regression with all those variables will take a very long time and may possibly not work. We are interested in controlling for them, but are not really interested in their coefficients. So you should use the `partial` option of `ivreg2` to include those variables without estimating the coefficients so the regression runs faster.

Part 5: Creating and exporting tables

Stata commands in this section:

- `estout`

List of estout options you probably need:

- `cells()`
- `stats()` with `fmt`, `labels`, and `layout` suboptions
- `keep()`
- `rename()`
- `substitute()`
- `varlabels()` with `elist` suboption
- `mlabels()`
- `mgroups()`
- `collabels()`
- `posthead()`
- `order()`
- `refcat()` with `below` suboption
- `indicate()`
- `prefoot()`
- `transform()`
- `varwidth()`
- `wrap`
- `style(tex)`
- `replace`

I have provided fi2022_replication_key.pdf, which displays what each of the tables and figures you create should look like.

We have already used `estout` to create a summary statistics table, so you should refer to the previous assignment instructions for tips on that if necessary.

The new aspects of `estout` for this assignment involve displaying additional statistics in Table 3. You will use the `stats` option to do this, which will also require the `fmt`, `labels`, and `layout` suboptions.

Tips for using `stats`:

`stats` need to be placed in the order they will appear. To insert a blank line between statistics you should type “blank” as a statistic (e.g., type `stats(`*stat1* `blank` *stat2*`)` to include a blank line between *stat1* and *stat2*.

Use the `layout` suboption to place statistics within the same cell (e.g., the coefficient and significance stars). Include them in quotes to place them in the same cell: `stats(stat1 stat2, layout(“@ @”)`, which prints *stat1* and *stat2* next to each other in the same cell in the same row.

Use the `layout` suboption to place statistics in parentheses or brackets: `stats(stat1 stat2, layout(“(@)” “[@]”)` prints *stat1* and *stat2* in separate rows and puts *stat1* in parentheses and *stat2* in brackets.

The `labels` suboption of `stats` only requires labels for the number of rows you create. In other words, if you type `stats(stat1 stat2 stat3, layout(“@ @” @)` so that *stat1* and *stat2* are printed in the same cell on a single row and *stat3* is printed on a new row, then you only need to define two labels (for two rows, instead of three labels for three statistics).

General tips:

Instead of typing “\$” into `estout`, type “@@” where you want the dollar sign to go and use the `substitute` option below to insert the dollar sign.

Use the substitute option to create significance stars from your p-value statistics and type a dollar sign: `substitute(@@ “\\$” 9999999999 “***” 999999999 “**” 99999999 “*” 9999999 “ “)`

If you use the `partial` option with `ivreg2`, as we do here for the control function, then Stata does not recognize that those variables included in `partial` are in the regression, meaning that the `indicate` option in `estout` will not recognize them. For this reason, I coded the “Control function” row of Table 3 manually using the `prefoot` option.

The *marriage_subsidy* coefficient is reported in $1,000s, but the variable in the dataset is measured in $1s. You can rescale the coefficient after-the-fact using the `transform` option of `estout`.

Use the `varwidth` and `wrap` options of `estout` to put variable labels on two lines. Spaces contained in variable labels count as characters that contribute to the `varwidth` threshold.

Part 6: Creating Figure B1a

Stata commands in this section:

- `reg`
- `predict` with and without `residuals` option
- `collapse` with *mean* and *count* statistics and `by()` option
- `replace`
- `twoway scatter` with `fweights` and `msize`, `mcolor`, and `msymbol` options
- `twoway lfit` with `fweights` and `lcolor` option
- `twoway qfit` with `fweights` and `lcolor` and `lpattern` options
- `graph export`

List of `twoway` options you probably need:

- `name()`
- `ytitle()`
- `ylabel()` with `format()` and `angle()` suboptions
- `yscale()`
- `xtitle()`
- `xlabel()` with `format()` and `angle()` suboptions
- `xscale()`
- `text()` with `size()` suboption
- `legend` with `order()` and `cols` suboptions
- `graphregion()`

The process to create Figure B1a is described in Friedberg and Isaac (2022) Appendix B, Section B.2, which involves estimating IV specification from Table 3, column 6 manually using residuals.

The dependent variable is still *sscouple_mar*, but the independent variable is *ms_hat*, which is the first stage fitted values.

In your .do file, write code that:

1. Estimates the first stage regression using `reg` with robust standard errors and predicts fitted values using `predict`
2. Estimates equation B1 from Appendix B.2 using `reg` with robust standard errors and predicts residuals using `predict` with the `residuals` option
3. Estimates equation B2 from Appendix B.2 using `reg` with robust standard errors and predicts residuals using `predict` with the `residuals` option

4. Estimates the regression using the marriage residual as the outcome variable and the marriage subsidy residual as the dependent variable using `reg` with robust standard errors
    - This confirms that you get the same coefficient as Table 3, column 6. You can also use this regression to capture the coefficient for the slope label in Figure B1b.

Residual bins

Observations in Figure B1a are grouped into $50 bins of the marriage subsidy residual. You still need to obtain the residuals yourself using the steps above, but I have provided the bins for you as the variable *residual_bin* to simplify things. Note that these are what the bins would be if you compute the residuals correctly.

Collapsing data

The data for Figure B1a are cell means of couples in each marriage subsidy residual bin. You can use the `collapse` command to collapse (i.e., aggregate) your data into these cells. The `by(varlist)` option of collapse will tell Stata to collapse within categories of *varlist*.

Specifying *mean* for some variables will tell Stata to collapse the data into mean values within the cell. Specifying *count* for some variables will tell Stata to collapse the data into the value of non-missing values of those variables.

In your .do file, write code that:

1. Collapses the data into *mean* values of the marriage residual and marriage subsidy residual and a *count* value of any other variable with no missing values by *residual_bin* using `collapse` with the `by(varlist)` option.
2. Adds 0.432 to the marriage residual
3. Constructs Figure B1a using `twoway`
4. Exports your figure as a .pdf file using `graph export`

Collapse tips:

You can use any variable that has no missing values for your *count* variable because you only need a count of how many observations are in each cell so you can use it as weights in the figure.

If you only specify the marriage residual, marriage subsidy residual, count variable, and marriage subsidy residual bin variables then you should end up with a dataset with 4 variables.

Part 7: Incorporating tables and figures into LaTeX

Download the `Student LaTeX Template.tex` file from Canvas. You can either open this in your own TeX editing software or open it with a text editor (e.g., Notepad in Windows or TextEdit in Mac) and copy/paste it to a blank document on Overleaf.com.

The `Student LaTeX Template.tex` file is already set up to directly import your tables and figures into the .tex file without you needing to change anything. I am providing this .tex file to you as a template for how you can set things up in the future.

A few things that this .tex file does that may be useful in the future:

1. Creates a bibliography using the biblatex package.
2. Expands the notes area underneath tables to be the page width even if the table is narrow.
3. Uses the `\input{}` and `\includegraphics{}` commands to pull tables and figures into the document without manually typing.
4. Uses the `\adjustbox{}` command to adjust the size of tables (not necessary in this case, but can be useful if you have very wide or tall tables to get them to fit on the page).

If you use your own TeX editing software on your computer:

1. Place the .tex file in your directory.
2. Make sure Stata is saving your table and figure output to your directory (not to a sub-folder).
3. Compile.

If you use Overleaf.com:

1. Copy the `Student LaTeX Template.tex` file into a blank document.
2. Upload your tables and figures to the project.
3. Compile.