Readme for AEJ Compartamos source code

This readme provides information for replicating the results of the Compartamos paper.

Many readers may be interested solely in replicating the main OLS regression results of Tables 2-8. For this, simply run the do-file `Main/Compartamos AEJ tables 2-8.do` in Stata 11.2 or higher. The dataset `Main/data/analysis_data_AEJ_pub.dta` is used.

There are two other sets of primary results: the descriptive tables Table 1 and Appendix Table 1 and the quantile regression results. These require more code, as well as steps to set up the Stata environment; Stata 13 is also required. These source code files are stored in `Source`. Thus, there are two subdirectories: `Source`, for the source code for the descriptive and quantile results, and `Main`, which contains the simple do-file for the main regression results, as well as do-file inputs and outputs.

The variable labels of `analysis_data_AEJ_pub.dta` are designed to match the tables and figures. For more information about a variable's definition, see the Data Appendix.

Dataset variable names

Variables whose names begin with `BE_` are the baseline equivalent of an endline variable: `BE_varname` is the baseline version of `varname`. `BE_` variables have had their missing values replaced with `0` in preparation of the regressions. Many `BE_` variables are defined only for the panel sample and are missing (`0`) outside the panel.

A variable whose name matches the pattern `BE_*_mdum` is a dummy variable that indicates whether the corresponding baseline equivalent (`BE_*`) variable is missing. In preparation of the regressions, the missing values of baseline equivalents are replaced with `0`. When these variables are included in regressions, their corresponding missingness dummies are included, as well.

The remainder of this readme describes the steps required to run the files in `Source`.

User-written programs

Type the following in Stata to install SSC commands used in the Compartamos do-files:

```
ssc install fastcd
ssc install estout
ssc install winsor
ssc install eclplot
ssc install charlist
ssc install outreg2
```

Now set up `fastcd` to work on your computer as follows:

```
* Change the working directory to the location of Source on your computer.
cd ...
c cur comprado

* Change the working directory to the location of Main on your computer.
cd ..
c cur compra
```

After this, the command `c comprado` will change the working directory to `Source`; likewise for `c compra` and `Main`.

`fastcd` is the name of the SSC package, not the command itself; the command is named `c`. To change the working directory, type `c` in Stata, not `fastcd`. To view the help file, type `help fastcd`, not `help c`. `cd` is a rare command in Compartamos do-files; `c` is used wherever possible.

These user-written programs are also saved in `Source/ado/External`. If a user-written program has changed since the time of this readme, the version we used can be found there.

Working directory

Set the working directory to `Source` by typing `c comprado` after setting up `fastcd`. Use relative paths to refer to files in `Source`.

The names of directories and datasets outside `Source` (i.e., in `Main`) are stored in globals defined by `header.do`. `header.do` is a do-file that completes initialization common across do-files: defining shared macros, compiling Mata functions, and so on. To use files outside `Source`, use the globals; do not change the working directory. `header.do` assumes that the working directory has already been set to `Source`. If you can run `c comprado` followed by `include header` without receiving an error message, you should be set up and ready to run the remaining do-files in `Source`.

Which do-file do I run?

Many do-files call other do-files, which are not meant to be run on their own. It is important that you run the correct do-files. For Table 1, run `Source/Analysis/tables_descriptive/Table 1.do`. For Appendix Table 1, run `Source/Analysis/tables_descriptive/Appendix Table 1.do`. For the quantile results, first run `Source/Analysis/Quantile/graphs_quantile_regs.do`, then run `Source/Analysis/Quantile/graphs_quantile.do`. The former creates datasets of quantile results; the latter uses those datasets to produce the graphs found in the paper. Separating these tasks means that you can modify the graphing do-file without having to rerun the quantile regressions.

All other do-files in `Source`, including `header.do`, are subroutines that are run by the four do-files listed above; there is no need to run them yourself. There is also no need to move the ado-files from `Source/ado` to a system directory such as `PERSONAL` or `PLUS`; `header.do` ensures that the do-files will find these files. Further, there is no need to compile the `.mata` files in

`Source/ado`; again, `header.do` does this.

Visual Basic

Table formatting is implemented in Excel using VBA: see `Source/bas`. Do-files create `.csv` files that are formatted using VBA for the paper. For instance, `Table 1.do` creates a specially formatted .csv file that the `.bas` files convert to easily readable tables.

To run the `.bas` files, copy `Source/bas/FormatTables/FormatTables parameters sheet example.csv` to a new Excel workbook, naming the worksheet `Parameters`. Replace the absolute file and directory paths listed in the cells of the sheet so that they match your computer. Next, add `Source/bas/ExternalModules.bas` to a new VBA module; do not add the other `.bas` files. Replace this line:

```
Const DIR_COMPRADO = "...\Source\"
```

so that it equals the absolute path of the `Source` directory. The path must end in a directory separator (\ for Windows, / for Mac).

For convenience, we have added an `.xlsm` file for which the `Parameters` sheet and `ExternalModules.bas` have been added: see `Main/results/Tables/Format Compartamos tables.xlsm`. You must still modify the file and directory paths as described above.

Enumerated types

See this (http://www.stata-journal.com/sjpdf.html?articlenum=pr0040) Stata Journal article for a description of macro use in Mata. This article also motivated us to try to implement enumerated types in Stata. For the purposes of Compartamos, an enumerated type is a one-to-one map of names to positive integers, where each association constitutes a value of the type. The values of the integers do not really matter, because the values of enumerated types are referred to solely by their names.

The values of enumerated types are stored in locals. The name of the local is the concatenation of the name of the enumerated type and the name of the value. The value of the local is the integer associated with the value. For example:

```
local CatCredit 1
```

`` `CatCredit' `` represents a single value of the enumerated type `Cat` (representing outcome categories). The value is associated with the name `Credit` and the integer `1`.

Locals whose names begin with the name of an enumerated type are reserved for that type: they store the values of the type and nothing more. Further, the names of locals used for enumerated types are strictly CamelCase. Other than locals used for Mata, no other multi-word local names are CamelCase, in order to minimize the probability of name conflicts.

All source code for enumerated types is housed in `Source/ado/Enumerated types`. Adding enumerated types to a do-file requires `include header` followed by `enumtypes`.

General programs for enumerated types

Use `enumtypes` and `evalname` for general use and management of enumerated types. Also see the Mata functions `evalname()` and `enumtype()`.

Type-specific conversion programs

Most use of enumerated types is restricted to the corresponding locals, for example, `` `SampleEndline' ``. However, sometimes it is necessary to convert a specific enumerated type to another object. For instance, for the `Sample` type, you may wish to get an `if` qualifier expression that returns `1` if an observation belongs to the sample and `0` otherwise; for `` `SampleEndline' ``, this would be `survey == "Endline"`.

Conversion programs in `Source/ado/Enumerated types` serve this purpose. In the case of the `Sample` type, the program `sampexp` takes a `Sample` enumerated value as an input, and returns the associated `if` qualifier expression: `sampexp `SampleEndline'` returns the relevant expression in `r(exp)`.

While not a program, `Set category properties.do` also converts a `Cat` enumerated value to other objects. `Set specification properties.do` takes a string analysis specification name and returns related values.