WIDE Stata package: v.1.0

Gabriela Mathieu

June 12, 2020

Contents

1 Introduction

This document describes the process of automation and modularization of the WIDE code that allowed it to be turned into a Stata package. The updating of the code involved its modularization and refactoring. Thus the code was structured into different functions with specific tasks. The advantage of this change is evident in providing greater readability and reproducibility. At the same time, it allows future changes to be made more easily and reduces the risk of errors.

The WIDE package takes care of the following tasks: reading the original data from MICS (round 4, 5, and 6) and DHS, renaming variables, and standardizing the different data sets in order to work with a single database. Clean up the variables that have problems, recode variables, and build new variables. Finally, a single database is obtained to summarize the data for the different educational indicators of interest. Also, a testing function allows us to find indicator inconsistencies.

The package is stored at a Github repository and it includes A README file that contains a brief package's description and use.

2 Project plan

To obtain a more readable, robust and reusable code, it was structured in different functions, the structure of the folders where the different files are located was changed and the logic of the data was separated as much as possible reducing the hard-coding.

The code is automated and well documented, then someone else could run the same analysis or use a new datasets.

2.1 Modularization

The modularization involved, rewrite the code into short single-purpose functions to break down problem into bite size pieces. This allows reusing part of the code minimizing errors since it is not copied again but called through the function. In this sense, similar code blocks that used in different parts of the main-do are now part of a function.

The main stages of the process are: importing, cleaning, calculating and summarizing. Each of these stages is now a function whose name indicates the stage it performs: mics_read, mics_clean, mics_calculate, mics_summarize, etc.

The use of global macro is reduced and instead, local macro and temporal files are used to work with more than one file at a time. Stata's restriction to working with only one dataset in memory has been overcome by version 16.0, but for earlier versions, the only way is to use temporal files.

2.2 Name and style

In order to obtain a readable and tidy code, I have applied coding conventions, such as file organization, comments, naming conventions, programming practices. I've tried to be consistent throughout the package in naming, indentation, and other aspects of style. For example, I chose proper and expressive naming for functions because it is better to understand what the code does. I have also applied a consistent code style to make it easy to read: I have chosen to include spaces between a variable name and an operator and value.

The files are organized into directories (folders) that can help you easily categorize and find what you need (e.g. raw-data, data, output, etc.). Both the organization of the files and their names help the reproducibility of the code by helping others to understand how the package works quickly.

2.3 Documentation

In addition to this document, it is possible to access the functions in the repository as well as the README file. All the functions have comments to explain the code also comments throughout the code to explain the specific steps of the workflow. The functions and the auxiliary data are store in a private Github repository: https://github.com/rosavidarte/WIDE/.

2.4 Control Version Software

GitHub.com is a website that supports version control using git and it is a collaboration tool. Using GitHub also facilitates sharing the code with everybody or specific people privately. Git provides a way to create and track a "repository" for a project, i.e., a folder where all relevant files are kept. GitHub is a cloud-based platform to host git repositories, which allows us to store and manage our files and track changes. It is help to share and ease of replication and extension of the package by others, which further supports peer review.

3 Repository

The project folder it re-organized in a new tidy structure.

The github folder directory is like this:

```
folder_directory
├ README.me
└ .gitignore
   ├ programs
   │  └ package
   ├ auxiliary_data
   ├ output
   └ raw_data
```

The package includes the package folder and the auxiliary data folder. The raw data contains the raw microdata from MICS and DHS, those files could be download from the internet.

4 Raw data

Using this package requires to download the data from each source, DHS and MICS, from their website. You must register and login to have access to the datasets.

In both cases, we download a zip file containing the datasets. In the case of MICS for each country and year, we have a file that we place in a folder with the name of the country and inside it a folder with the name of the year. The dataset keeps the original name (e.g. hl). As we download more datasets we add year folders for that country. The dataset is in 'sav' format and we convert it to 'dta' with the command `usespss`.

As a rule to write the name of the countries we define:

- the first letter of the name will be capitalized.

- if the name of the country consists of more than one word each must be capitalized unless one of them is “and”.

We keep the DHS data the same way. We select the module Household Member Recode and in this case, there is the option to download them in different formats, we choose the option FL (Flat ASCII data).

The DHS filenames are more specific than the MICS filenames, e.g. HNPR61FL, where ‘HN’ is the country code, ‘PR’ is the survey module, ‘61’ is the round, and ‘FL’ is the file format. From DHS it is also necessary to download the (IR) module and the (MR).

MICS files (Household Member Recode - PR files) are not homogeneous between countries and years, changing the type of variable, supported values as well as the names, this variation occurs both between rounds and between countries. Therefore the process of cleaning up the data is a little more arduous than that of the DHS data which is quite standardized between countries and rounds.

For the proper functioning of the package the folder structure should be as follows:

```
folder_directory
└── ...
    └── raw_data
        ├── DHS
        │   ├── Afghanistan
        │   │   └── 2015
        │   │       ├── AFPR70FL.dta
        │   │       ├── AFIR70FL.dta
        │   │       └── AFMR70FL.dta
        │   └── ...
        └── MICS
            ├── ...
            └── TrinidadandTobago
                └── 2011
                    └── hl.dta
```

The raw data is preserved so it's not modified along with the analysis. The data outputs are kept separate from inputs so that you easily re-run your workflow as needed.

5 Tables

In the original code, cleaning the raw data required many lines of hard-coding. To reduce the code as much as possible and minimize errors, I have added tables with information regarding religion, ethnicity, educational duration, educational level and region. These tables are merged with the data from all countries and the original values are replaced with those corresponding to the respective tables. For example, to fix the *completion and out of school* indicators. This reduces the risk of introducing errors if some value changes because you should change only one value in the table.

The 'auxiliary_data' folder contains also two folders, one for MICS and other for DHS auxiliary tables:

On the other hand, I created auxiliary tables that, through a merge with the dataset, allow a systematic replacement of values in different variables (region, ethnicity, etc.).

```
folder directory
└── ...
    └── auxiliary_data
        ├── country_survey_year_uis.dta
        ├── UIS_duration_age_25072018.dta
        ├── month_start.dta
        ├── country_iso_codes_names.dta
        ├── filenames.xls
        ├── dhs_dictionary_setdoce.xls
        └── mics_dictionary_setdoce.xls
```

The following table describes the use of each table:

Table	Description
filenames	lists the file paths to be read
dhs_dictionary_setcode	selects the variables in each country dataset, standardizes the names and recodes several variables
mics_dictionary_setcode	selects the variables in each country dataset, standardizes the names and recodes several variables
country_iso_codes_names	adds the iso code3 variable
country_survey_year_uis	fixes survey year
current_school_year_MICS	fixes current school year
current_school_year_DHS	fixes current school year
dhs_adjustment	adjusts dates
dhs_interview_dates	adjusts dates
UIS_duration_age_*	adds levels duration
month_start	adds month start school

6 Functions

To improve the modularity of the code I have started to create functions for specific sub-tasks. This involves refactoring some of the original code, adding control structures, stop and warning functions to it. Using these functions will allow us to obtain a concise and robust main function called `widetable`.

The main changes I made to the MICS code are to separate it into 4 parts: reading, cleaning, calculating variables and summarizing. At the same time, inside each of these parts I have tried to simplify the code as much as possible and to separate the logic from the data. To do this, I introduced several Stata module commands that need to be installed and generated specific commands that I named with an underscore.

The “Package” folder contains some generic functions and specific programs for DHS and MICS (the ado files).

```
file_directory
└── ...
    └── package
        ├── widetable.ado
        ├── widetable.sthlp
        ├── widetests.ado
        ├── widetests.sthlp
        ├── dhs_read.ado
        ├── dhs_clean.ado
        ├── dhs_calculate.ado
        ├── dhs_summarize.ado
        ├── mics_read.ado
        ├── mics_clean.ado
        ├── mics_calculate.ado
        ├── mics_summarize.ado
        ├── replace_character.ado
        ├── replace_many.ado
        ├── standarize_output.ado
        └── dependency.do
```

6.1 Widetable function

The `wide` package simplifies reading, cleaning and summarizing WIDE. The main function of the package, `widetable`, imports DHS and MICS files, standardizes them and calculates educational variables. Finally, education indicators (access and completion) are obtained for each country and year of the survey, disaggregated by different variables of interest.

6.2 General porpouse functions

The general porpouse functiones are used also in MICS and DHS.

Function	Description
`replace_many`	replaces at once many values
`replace_character`	replaces several characters and accents that Stata does not recognize. This function is applicable to any string variable.
`standarize_output`	standarizes DHS and MICS output summary format

6.3 Reading function

In this part the data of each country is read, variables are selected according to the MICS dictionary where only those we are interested in keeping are included, the country and year_file variables are created, the names of variables and type are standardized.

Not always the same variable (according to its label) is called the same for different years or countries. The different names it takes are detailed in the dictionary and then in the standardization process where they are renamed.

A part of the data cleaning is needed in the reading.do because it is necessary to obtain a single set of data with the same name of variables, thus reducing the number of stored variables that occupy disk space but mainly occupy ram memory in reading and processing.

6.4 Cleaning function

In this part the values of the variables are standardized and for some variables, the values are replaced according to the auxiliary tables. These include region, religion, date, ethnicity, urban, sex, year, original education variables.

To simplify replacing one value with another, I created the replace_many function that replaces a "master" dataset value with a "using" dataset value as long as certain variables match.

6.5 Calculating education variables functions

The education variables are calculated depending on how the information is collected in each country. The calculate function computes the years of education by country and age group, computes the level reached in primary, lower secondary, upper secondary and educational gap, finally, computes if someone does not go to school.

6.6 Summarizing function

The summary function allows us to obtain the education indicators for each country by several variables of interest (region, religion, ethnicity, sex wealth, and location). The output is stored in a spreadsheet and on a SQL database for testing.

6.7 Testing function

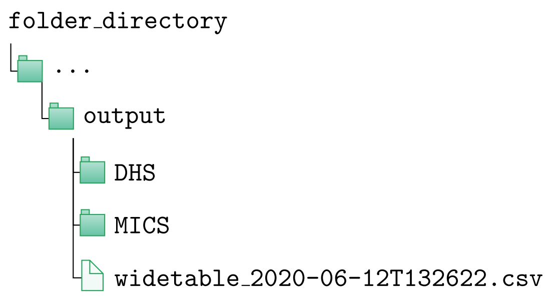
We can call a single database and execute SQL queries that automatically test the summarized data according to some parameters. For example, check that no percentage is less than 0 or more than 100. Warn about cases that take extreme values such as 0 or 100 and sudden changes between one year and another for the same country

The `widetests` function allow us to test the results. There are 15 pre-set tests.

nquery	Query description
1	checks extreme values (completion)
2	checks extreme values (completion)
3	checks extreme values (completion)
4	checks extreme values (completion by age group)
5	checks extreme values (completion by age group)
6	checks extreme values (completion by age group)
7	checks extreme values (completion by age group)
8	checks extreme values (school gap)
9	checks extreme values (school gap)
10	checks extreme values (school gap)
11	checks extreme values (out of school by level)
12	checks extreme values (out of school by level)
13	compares among education levels within countries
14	compares among education levels within countries
15	checks intertemporal indicator variation by country

7 Outputs

The main function `widetable` generates a spreadsheet with the indicators and is stored in the output folder.



8 Example

8.1 Installation

The ado files should be placed in the `c:\ado\personal\` folder and read by Stata from there. They should not be placed in the `c:\ado\plus\` folder (where packages downloaded from the Internet are located) because they may be deleted in an update.

When the package is hosted in a public repository, it can be installed directly from Stata. To install the latest version directly from Github, type in Stata:

```
github install username/wide
```

You must have the github package installed to type it into Stata: net install github, from ("https://haghish.github.io/github/"). This way of installation is better than using the net command as it automatically installs the package's dependencies.

8.2 Usage

The main function, `widetable`, has four mandatory arguments:

- `source`: indicates which source must use ('dhs','mics' or 'both'). The option 'both' includes the other two.
- `step`: indicates which process must run ('read', 'clean', 'calculate', 'summarize' or 'all'). The option 'all' includes all the above.
- `data_path`: indicates the raw data folder path.
- `output_path`: indicates the output table folder path.

Defining the folder path, it is recommended to use slash (/) as separator instead of backslash (\), regardless of the operating system. You can write the paths directly in the function or previously create a local macro:

```
* Defines the path folder in a absolute way (replace the dots)
local dpath /../WIDE/raw_data/
local opath /../WIDE/output/
```

This is a basic example which shows you how to use the widetable function:

```
widetable, source(both) step(all) data_path(`dpath') output_path(`opath')
```

The result is a table with the indicators that is saved in the 'output' folder in 'dta' and 'csv' format called 'WIDE_yyyy-mm-ddThhmmss', where *mm* refers to the month, *dd* to the day, *yyyy* refers to the year, **hh** refers to the houres, **mm** the minutes and **ss** the seconds.

Also, you can run the function without seeing the intermediate results by typing:

```
  quitely widetable, source(both) step(all) data_path(`dpath') output_path(`opath')
```

It is also possible to define some optional arguments that allow to work on a reduced number of files.

- `nf`: default value is 300. With this value all MICS and DHS files are read. To test the function it is recommended to use a value lower than 10.

- `country_name`: to evaluate the function for a given country.

- `country_year`: to evaluate the function for a given year. It requires defining the 'country_name'

The arguments `nf`, `country_name` and `country_year` are only used when step is "read".

To execute the query number 5:

```
widetests, table(widetable) nquery(5)
```

Alternatively, it is possible to make these queries outside Stata is useful the "widetests.sql" file.

9 Annex

9.1 Installation programs from SSC

The ssc command allows you to easily download a package. For example, when you type

```
. ssc install fs
```

The Stata modules that need to be installed are described in the dependency.do file.

Function	Description
`catenate`	a simple way to concatenate strings variables.
`fs`	to simplify the append without the need to make a loop.
`gtools`	to collapse variables with the `gcollapse` command faster than the `collapse` command
`replacestrvar`	a compact way to replace special characters. It is included in the replace_characters.ado.
`encodefrom`	a module that includes the renamefrom command
`rmfiles`	to remove files and folders
`sdecode`	to decode variables in a systematic way and in one step. It is no longer necessary to generate a new variable and then apply the decode command to it to then delete the original variable and rename the new variable.
`tosql`	to export a file to SQL
`tuples`	to select all possible tuples from a list
`usespss`	to load SPSS files