Data Cleaning

Luiza Andrade, Kristoffer Bjarfekur & Benjamin Daniels

DIME Analytics – World Bank

November 7, 2017

Overview

Overview

Introduction

- This presentation will show you best practices to clean survey data
- At DIME, we have large teams collaborating on the same codes and data sets
- Standardizing data set formats is important to reduce the cost of moving from one project to another

Think reproducible

The end goal of our best practice guidelines is to ensure that all the research we develop is reproducible

- This means transparency, accountability...
- ... and an easier workflow
- Research reproducibility means we need to share our data
- It also means the people we share it with should be able to understand it

Overview

Team roles on data cleaning

Field Coordinator/Research Assistants

- No one will look at the data as much as the RA/FC
- Irregularities in the data that the RA/FC does not identify will often never be discovered
- Your role is to gather descriptive information on why something is irregular
- You can suggest how to solve the irregularities, but should spend most of your time identifying and describing them
- Never make any decision on data reformatting without talking to PIs

Team roles on data cleaning

Principal Investigators

- Decides which irregularities will be corrected and how
- PIs completely depend on FCs/RAs to identify irregularities and get the information to make the best call (of course, this all comes up when the PI starts writing up the analysis)

You've got the data. Now what?

1. **Cleaning data:** means formatting the data set to make the it more understandable and documenting the knowledge you have about the data (you will soon forget)
    - Encode variables
    - Label variables
    - Label missings
    - Rename variables
2. **Constructing variables:** means editing the data set to remove data points that will bias the analysis or make it invalid in any other way
    - Standardize units
    - Treat outliers

In practice, the two are often done simultaneously, so that you're working at the same time on a cleaning and a construct do-file, but it is useful to think of them separately

Overview

Data collection and importing

- For this presentation, we will assume you've already collected and imported your data
- When the data is being collected, high-frequency checks should be run daily, so any mistakes in the data can be corrected
- Part of the high-frequency checks is checking for duplicated IDs and variables consistency. If this is done properly, your data cleaning will be a lot easier
- If the data import from Survey CTO is done correctly, variable and value labels will also be imported, which will also make the data cleaning process shorter

De-identification

- Once the data collection is over and all data is imported, the data set should be de-identified
- That means all variables that contain personally identifiable information should be saved in an encrypted folder and a data set without those variables, generated
- Individuals names are the most obvious example of PII, but it also includes contact information, GPS coordinates, etc
- If these variables are needed for analysis, then the data set needs to be de-identified before being publicly released

Unique ID

The first thing you want to look for every single time you open a new data set for the first time is

1. Unit of observation
2. Uniquely and fully identifying ID variable

Before you separate the identifiable from the de-identified data, make sure you know how to cross both using the unique ID

Unique ID

Unit of observation check list when opening a new data set:

- What does each row represent?
- Which variable do you think is the main ID uniquely identifying each row?

Stata commands for testing that the variable indeed is uniquely and fully identifying

- `isid`
- `codebook`

Unique ID

HHID	Village	District	HH number	HH head	HHH Age
022501	25	2	1	Andrew	52
022502	25	2	2	Patrick	48
023207	32	2	7	Charles	29
023205	32	2	5	Jeffrey	37
012501	25	1	1	Walter	48
011103	11	1	3	Anne	26
011205	12	1	5	Lawrence	61
024502	45	2	2	Dennis	45
024501	45	2	1	Nancy	41

Unique ID

Clinic ID	Clinic Number	District	Patient	Age
02452	542	2	Andrew	52
02543	543	2	Patrick	48
02156	156	2	Charles	29
01152	152	1	Jeffrey	37
01152	152	1	Walter	49
01238	238	1	Anne	26
01122	122	1	Lawrence	61
02122	122	2	Dennis	45
02122	122	2	Nancy	41

Overview

Label variables

When cleaning a data set, you should make sure that all variables are properly labeled, so that it is easy to understand what each variable represents:

- Check all variables have variable labels (in English)
- Variable labels should explain what the variable is and, if that's the case, what unit it is in
- Labels cannot be longer than 80 characters

Note that survey data imported properly to Stata should not require this!

Encode variables

The clean data set should contain no string variables, except for

1. Proper nouns that are not categories
2. Digits with leading zeros or long IDs (over 15 digits)

That means categorical string variables must be made numeric with value labels

Encode variables

- Check all that all categorical variables have value labels (in English)
- Best practice is to use `encode` with both the `label` and the `noextend` options.

Example

```
encode dist_name, generate(dist_id) label(district) noextend
```

- Other useful commands: `label define`, `label value`, `label dir`, `label list`, `labelbook`

Note that survey data imported properly to Stata should not require this!

Label missings

- We use codes like -88, -9,-777 to represent different reasons for missing data such as dont know, declined to answer etc
- These values need to be removed since they will otherwise bias the means
- If we change them all to missing, we will lose information
- Use extended missing values to keep the information but still tell Stata to treat them like missing

Tip

The command `summarize` is efficient to find codes for missing answer as they stand out as negative values (that is why we use negative values as missing answer codes)

Label missings

- Regular missing value in Stata: .
- Extended missing values in Stata: .a, .b, .c etc.
- Use each missing value to represent the same reason for missing data across the project
  - .a = question not applicable
  - .b = dont know
  - .m = not in monitoring data
- Replace this

```
sum HH_income if employment != .
```

  With this

```
sum HH_income if employment < .
sum HH_income if !missing(employment)
```

Renaming variables

Do not change the names of variables coming from a survey, even if you do not like the naming conventions used in the questionnaire. There are two exceptions for this:

1. **Identifying roster variable**: In a loop over, for example, crops where a variable like harvest is asked for each crop and automatically named, harvest_1, harvest_2 etc., then those can be renamed to harvest_crop1, harvest_crop2 etc., or harvest_c1, harvest_c2 etc.
2. **Roster number**: if a household cultivated several crops, but will only be asked about the 5 most important ones, then their codes will not correspond to the crop codes, but to their importance. So harvest_c1 means the harvested quantity of the most important crop, not of the crop whose code is 1. This may be changed to reflect the regular code for each crop

Check variables consistency

- Check that values are consistent across variables
- For example, if an individual is male, then he cannot be pregnant
- This kind of inconsistency should usually be corrected during the high-frequency checks, but often times there's no time when the enumerators are in the field to identify and correct all of them
- So if you find any issues, create flag variables that identify observations with inconsistent values

Identify and document outliers

- We do not want our results to be driven by a few individuals. For example, if the village leaders get all benefits
- There is no exact rule for what is an outlier. Ask if your PI for preference of specific rule
- Identifying outliers often comes down to common sense. Can the outlier be explained by typos, Especially common when selecting units from multiple choice lists?
- RAs should try to identify as many discrepant values as possible, even at the cost of not correcting them

Useful commands

- `sum detail`
- `tabulate`
- `inspect`
- `assert`
- `histogram`

Overview

Add metadata as notes

- Variable labels must be short and self-explanatory
- That means they must often be different from the survey question itself
- However, you can add any other information to the variable as a note
- Examples of relevant information are question wording, value constraints and relevance conditions
- Use the `notes` command to add this information to your variable

Recategorize values listed as "others"

- Categorical variables usually have an open-ended "other" option that is saved as a string
- Answers that appear frequently in the open-ended question can be included as a new category in the categorical variable
- That is usually done during the pilot or the high-frequency checks, but it is possible that there are still relevant categories left out

Drop variables from survey

- Some variables are created to be used within the survey and for survey checks
- That is the case of most calculate fields, as well as notes and duration variables
- Variables that are not part of the questionnaire itself may be dropped from the clean data set

Ordering variables

- It is recommended the variables in the final data set follow the some order as in the questionnaire
- If you created new variables during the data cleaning, for example to change roster codes, they will probably be out of order
- You may want to reorder those variables so the data set is easier to read and to compare to the questionnaire

Overview

Saving files

- During the data cleaning process, you might have saved multiple intermediate files, for example if you cleaned long modules separately to make your code more readable
- After cleaning your data and merging it back together, you'll want to save a final cleaned data set, containing all variables from your survey
- This new data set will probably be quite heavy. Use `compress` to save your variables in the most economic format
- It's often desirable to save your data set in a previous Stata version, so other members of your team will not have version conflicts. To do this, use `saveold`

Naming files

- Make sure all output files, datasets and others are clearly and uniquely labeled, i.e.: “desc_stats_tmt_only.xls” “input_plan_adm_data.dta”
- It’s often desirable to have the names of your data sets and do-files linked, so it is easy to understand which do-files is creating which data set, such as “merge.do” and “merged.dta” or “cleaning.do” and “clean.dta”
- Do not use _v1, _v2 etc. for any final files. This leads to bugs in do-files that depend on these files when a new versions is added.
- It’s ok to use _v1, _v2 etc. for old versions of files if you **really** need to keep an archive

The clean data set

- At the end of this process, you should have a data set that is essentially the same as the one you downloaded from the server
- The main difference is that the clean data set should be easier to understand for anyone that's opening it for the first time
- Other than that, it should be very similar to the raw data and entirely comparable to the questionnaire
- This is the data that we will publish in the Microdata Catalog

Overview

Construct stage

- The data set is now clean and ready to be used
- The RA is very familiar with it and has identified all possible issues in the data
- The RA should discuss with the PI what is the best way to correct these issues
- In the construct stage, the data set will actually be modified to solve such issues
- The next few slides will discuss the most common of those issues and possible solutions

Standardize units

- Quantity variables should be converted into standardized units, so all the answers are comparable
- That means one unit for weight (usually kg), length/distance (usually meter), etc.
- Often a difficult task due to usage of local non-standardized units
- Use field team to collect as much information about local units and then decide on a conversion rate together with your PI
- Set all conversion globals in the master do-file, so they're easily accessible

Treating outliers

- When you were first exploring your data set, you identified and documented outliers in different variables
- These outlier values can affect our estimations, so we have to treat them
- The goal, however, is to deal with them while losing as little information as possible
- How these values will be corrected is ultimately the PI's decision

Treating outliers

Here are the most common ways of dealing with outliers:

1. Deleting observations is almost never desired as it loses a lot of information in variables that were correct
2. Deleting specific data point (i.e. replace to missing) is frequently used, but the observation would be dropped in regressions using that variable
3. Winsorization is a method were observations are not dropped, neither from the data set or from regressions

Treating outliers

- This method assumes that an upper tail outlier is still some large number
- We set all values larger than the a selected percentile to that percentile
- The outliers will still be a large number, but it will no longer greatly bias the estimators
- Can be applied in both upper and lower tail

Example

```
winsor revenue, gen(revenue_w) p(.01) highonly
```

Thank you!