iefieldkit: Stata commands for primary data collection and data cleaning

Kristoffer Bjärkefur, Luíza Cardoso de Andrade, and Benjamin Daniels
July 11, 2019

Development Impact Evaluation (DIME)
The World Bank
A brief introduction to iefieldkit

iefieldkit is designed to apply many of the lessons from the last presentation to other common tasks in the DIME data collection workflow.

- Working with primary data from developing contexts.
- Large teams with many projects and diverse skillsets.
- Standardization of easy tasks adds value and avoids error.
Data collection is a process that has traditionally suffered from low levels of documentation, standardization, and replicability.

iefieldkit is meant to bring that mindset to these tasks, which are often partially conducted by staff with little or no Stata skills.

We think it is a great example of using Stata to bring ideals of standardization and replicability to one of our core tasks that is usually considered less technical.
A brief introduction to iefieldkit

Data collection requires lots of analytical work that we don’t necessarily want to keep in Stata dofiles. iefieldkit provides a start-to-finish data collection and cleaning workflow that is self-documenting.

Core commands:

- ietestform ensures that ODK surveys are Stata-optimized
- ieduplicates & iecompdup identify and resolve duplicates in data
- iecodebook uses spreadsheet codebooks to clean or append data

All iefieldkit commands automatically output human- and machine-readable spreadsheet documentation as a functional part of the intended workflow.

https://dimewiki.worldbank.org/iefieldkit
ietestform
We believe that it is best to have automated quality control in place, even before data is ready for Stata. Stata is a very convenient tool for this purpose because our teams are already familiar with it. This idea can extend to any workflow involving structured non-Stata components.

- Open Data Kit (ODK) is a common data collection software in the field
- Many of our teams use SurveyCTO, a proprietary variant of ODK, and almost all our teams use Stata for data analysis
- But ODK data isn’t naturally prepared for Stata, and Stata doesn’t know what ODK data can look like
- Therefore it is very easy to make “non-errors” in ODK programming that are time-consuming and challenging to fix for Stata after the data is already collected
ODK data collection (and proprietary implementations like SurveyCTO) are common in primary data collection.

- Structured “pseudo-code” in spreadsheet format is built into survey
- Material is both human and machine-readable
- Lots of options for controlling data format
**BUT...** the survey forms are primarily built and operated by field staff or survey firms, not by Stata coders!

So we designed `ietestform` to read the survey definition file and give instructions on best practices and likely errors that are easier to fix during survey design than after data collection.

**Major tests for Stata optimization:**

- All variable names are Stata-compliant, including auto-generated ones in rosters and other dynamic fields
- All variables use multi-language support to create a “Stata” variable label that is *not* the full text of the question
- All value labels are Stata-compliant
ietestform: Generating a flags report

Simple syntax:

```python
ietestform
, surveyform("/path/to/survey.xlsx")
report("/path/to/report.csv")
```

- CSV format supports version control in Git
- Flags report likely errors
- Sometimes functionality may be desired, so you do not necessarily want an “empty” report
Additional syntax checks ensure machine-compatibility after import. All flags are linked with a complete explanation for the practice on https://dimewiki.worldbank.org/ietestform.

- All groups and loops open and close correctly
- No leading or trailing spaces in fields
- No repeated values or value labels, and no unused values in value labelling
ieduplicates & iecompdup
• Primary data coming in from the field can be very messy!
• Cleaning raw data and doing quality assurance is time-sensitive: it has to be done while the survey team is still on site
• Entries with duplicated identifier variables are particularly bad: they prevent the team from knowing the results of other quality checks, and therefore from efficiently implementing things like followup surveys
• Therefore there is a huge value to our team for having a standardized and pre-programmed process for handling duplicates coming in from the field

Additional challenges in this phase include interfacing with non-technical staff in the field; and creating documentation of the resolution of issues.
ieduplicates implements a standard self-documenting workflow using Excel data output and input. The command outputs a report of duplicates into Excel, and the user responds in pre-defined ways to each flagged observation.

- Run ieduplicates on the raw data. If there are no duplicates, you are done!
- If there are duplicates in the Excel report, analyze them using Stata and/or field records to find out the correct resolution.
- Enter the resolutions on the corresponding observations in the report outputted by ieduplicates.
- After entering the corrections, save the report in the same location with the same name.

Why Excel? Because it is easier for everyone to read and understand when there are large numbers of information to process, rather than de-coding Stata code.
On the first run, `ieduplicates` does two main tasks:

- Lists all duplicates in data into a file called `iedupreport.xlsx` and backs up a dated copy
- Removes all copies of duplicates from the data so other quality-assurance tasks like back-checks can be performed on unambiguous portion of data
ieduplicates: Correcting duplicates in data

ieduplicates expects standardized, structured responses to the observations flagged, so that they can be written and read quickly by any staff.
On *future* runs, *ieduplicates* will first apply all corrections in the current version of the duplicates report to the raw data – accept as correct, drop, or change ID.

- Run *ieduplicates* on the raw data again. The corrections you have entered will be applied, and only duplicates that are still not resolved are removed this time. Note that the raw data is unchanged, and therefore the report leaves a record of how all duplicates were resolved in the creation of the final dataset.
- Repeat these steps every time you get new data. Our recommendation is that this is done every day that you have new data.
iecompdup: Analyzing duplicates in data

Used on the raw data, iecompdup will return basic information about how duplicate observations are the same or different (with the relevant information stored in return for programming of reports). Naturally there is no way to fully automate the resolution process, but we look for three main groups:

Case 1. Double submission of the same observation, with the same data. 
   Resolution: Keep only one of the entries.

Case 2. Double submission of the same observation, with different data. 
   Resolution: Return to field team for audit.

Case 3. Incorrect ID variable. 
   Resolution: Return to field team to obtain correct ID.
**iecompdup: Analyzing duplicates in data**

Syntax:

```
iecompdup idvariable, id(idvalue)
```

Notes:

- Only accepts pairwise comparisons; any help on reporting about larger groups would be appreciated!
- No other output or documentation; intended to encourage careful review and documentation in the main ieduplicates report
iecodebook
Three tasks for reproducible data construction

- **Data cleaning**: `iecodebook apply`
  Reads an Excel codebook that specifies renames, recodes, variable labels, and value labels, and applies them to the current dataset.

- **Dataset combination**: `iecodebook append`
  Reads an Excel codebook that specifies how variables should be harmonized across two or more datasets - rename, recode, variable labels, and value labels - applies the harmonization, and appends the datasets.

- **Data documentation**: `iecodebook export`
  Creates an Excel codebook that describes the current dataset, and optionally produces an export version of the dataset with only variables used in specified dofiles.

https://dimewiki.worldbank.org/iecodebook
iecodebook apply runs an arbitrary number of rename, recode, and label commands in a single line of code.

- Operates on dataset in memory
- Commands in structured spreadsheet for future reference
-Eliminates repetitive coding

Syntax:

```bash
iecodebook apply
using "/path/to/codebook.xlsx"
```
iecodebook apply: Setting up a template

// Load data
sysuse auto.dta, clear

// Create cleaning template
iecodebook template
    using "/path/to/codebook.xlsx"

The template subcommand sets up the spreadsheet based on the data in memory.
A step back

The codebook is nothing more than a structured way to write common Stata commands outside of Stata. Why did we decide to spend time implementing this additional layer of abstraction?

- These are easy tasks: any Stata user can write `rename`, `recode`, and `label`
- But doing it over and over again is extremely boring, and demands attention to detail (such as the ordering of the commands) that can cause silly errors
- Development datasets are often really large and messy (such as a dataset received from a partner agency)
- The goal of the Excel codebook is therefore to allow users to input large amounts of information quickly;
- and to make sure that information is structured so that other users can review it efficiently
### icodebook apply: Filling out the template

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>label</td>
<td>type</td>
<td>choices</td>
<td>name:current</td>
<td>label:current</td>
<td>type:current</td>
<td>choices:current</td>
<td>recode:current</td>
</tr>
<tr>
<td>2</td>
<td>survey</td>
<td>(Ignore this placeholder, but do not delete it. Thanks!)</td>
<td>float</td>
<td>yesno</td>
<td>make: Make and Model</td>
<td>str18</td>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
<tr>
<td>3</td>
<td>price</td>
<td>Price</td>
<td>float</td>
<td>Price</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>5</td>
<td>mpg</td>
<td>Mileage (mpg)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>7</td>
<td>rep78</td>
<td>Repair Record 1978</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>8</td>
<td>headroom</td>
<td>Headroom (in.)</td>
<td>float</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>9</td>
<td>trunk</td>
<td>Trunk space (cu. ft.)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>8</td>
<td>weight</td>
<td>Weight (lbs.)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>9</td>
<td>length</td>
<td>Length (in.)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>11</td>
<td>turn</td>
<td>Turn Circle (ft.)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>13</td>
<td>displacement</td>
<td>Displacement (cu. in.)</td>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
</tr>
<tr>
<td>14</td>
<td>domestic</td>
<td>Domestic Make and Model</td>
<td>yesno</td>
<td></td>
<td>foreign: Car type</td>
<td>byte</td>
<td>origin</td>
<td>(0=1)(1=0)</td>
<td></td>
</tr>
</tbody>
</table>

**Survey**

**Choices**

**Choices_current**
// Load data
sysuse auto.dta, clear

// Apply cleaning template
iecodebook apply
using "/path/to/codebook.xlsx"

Simply changing template to apply in the command gives the basic syntax.

The drop option removes all un-named variables. The missingvalues() option specifies extended missing value codes for the whole dataset.
### DIME Training Baseline

<table>
<thead>
<tr>
<th>Field</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (required)</td>
<td>What is your name?</td>
<td></td>
</tr>
<tr>
<td>quest (required)</td>
<td>What is your quest?</td>
<td></td>
</tr>
<tr>
<td>airspeed (required)</td>
<td>What is the average airspeed of an unladen swallow?</td>
<td></td>
</tr>
<tr>
<td>color (required)</td>
<td>What is your favorite color?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Blue</td>
</tr>
<tr>
<td>3</td>
<td>Green</td>
</tr>
</tbody>
</table>

### DIME Training Endline

<table>
<thead>
<tr>
<th>Field</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp_name (required)</td>
<td>What is your name?</td>
<td></td>
</tr>
<tr>
<td>resp_quest (required)</td>
<td>What quest are you on?</td>
<td></td>
</tr>
<tr>
<td>resp_color (required)</td>
<td>Among the following, which is your favorite color:</td>
<td></td>
</tr>
<tr>
<td>swallow_speed (required)</td>
<td>What is the average airspeed of an unladen African swallow?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Blue</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
</tr>
<tr>
<td>4</td>
<td>Green</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0-25 km/h</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>26-50 km/h</td>
</tr>
<tr>
<td>3</td>
<td>51-75 km/h</td>
</tr>
<tr>
<td>4</td>
<td>76-100 km/h</td>
</tr>
</tbody>
</table>
iecodebook append runs an arbitrary number of rename, recode, and label commands on two or more datasets with the intention of harmonizing them. It then runs an append command on the harmonized data.

- Operates on datasets on disk
- All data structures in one spreadsheet for future reference

// Create codebook template
iecodebook template
"baseline.dta" "endline.dta"
using "/path/to/codebook.xlsx",
surveys(Baseline Endline)

The template subcommand sets up the spreadsheet based on multiple datasets on disk. The surveys() option names the datasets in the template and is required.
## iecodebook append: Setting up a spreadsheet template

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>name</td>
<td>label</td>
<td>type</td>
<td>choice</td>
<td>year</td>
<td>nameBaseline</td>
<td>labelBaseline</td>
<td>typeBaseline</td>
<td>choicesBaseline</td>
<td>recodeBaseline</td>
</tr>
<tr>
<td>survey</td>
<td>ignore</td>
<td>str</td>
<td>name</td>
<td>year</td>
<td>What is your name?</td>
<td>What is your quset?</td>
<td>What is the average airspeed of an unladen swallow?</td>
<td>What is your favorite color?</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>What is your name?</td>
<td>str?</td>
<td>quest</td>
<td>str16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>airspeed</td>
<td>What is the average airspeed of an unladen swallow?</td>
<td>byte</td>
<td>color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>What is your favorite color?</td>
<td>byte</td>
<td>color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resp_name</td>
<td>What is your name?</td>
<td>str7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resp_quest</td>
<td>What quest are you on?</td>
<td>str16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resp_color</td>
<td>Among the following, which is your favorite color?</td>
<td>byte</td>
<td>resp_color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>swallow_speed</td>
<td>What is the average airspeed of an unladen African swallow?</td>
<td>byte</td>
<td>swallow_speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### icodebook append: Filling out the template

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>label</td>
<td>type</td>
<td>choices</td>
<td>name:Baseline</td>
<td>label:Baseline</td>
<td>type:Baseline</td>
<td>choices:Baseline</td>
<td>recode:Baseline</td>
<td>name:Endline</td>
<td>label:Endline</td>
<td>type:Endline</td>
<td>choices:Endline</td>
<td>recode:Endline</td>
</tr>
<tr>
<td>2</td>
<td>survey</td>
<td>(Ignore this placeholder, but: float)</td>
<td>yes/no</td>
<td>name</td>
<td>What is your nut?</td>
<td>resp_name</td>
<td>What is your name? str?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>resp_name</td>
<td>Name</td>
<td>quest</td>
<td>quest</td>
<td>What is your quest?</td>
<td>resp_quest</td>
<td>What quest are you? str?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>swallow_speed</td>
<td>Guessed Speed</td>
<td>swallow_speed</td>
<td>airspeed</td>
<td>What is the average byte</td>
<td>swallow_speed</td>
<td>What is the average byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>resp_color</td>
<td>Favorite Color</td>
<td>resp_color</td>
<td>color</td>
<td>What is your favorite color</td>
<td>resp_color</td>
<td>Among the following colors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Survey Details:**
- **Survey Name:** Name
- **Questions:**
  - What is your name?
  - What is your quest?
  - What is the average byte?
  - What is your favorite color?
// Append the datasets
icodebook template
"baseline.dta" "endline.dta"
using "/path/to/codebook.xlsx",
surveys(Baseline Endline)

Simply changing template to append in the command gives the basic syntax.

All un-named variables are removed by default; nodrop cancels this. The missingvalues() option specifies extended missing value codes for the whole dataset.
Function 1: Just create the codebook for documentation

Function 2: Trim dataset based on variables used in dofiles:

- Reads your dofiles
- Keeps only the variables that are used in analysis
- Creates a minimal codebook
- Rewards good syntax – you must:
  Spell variable names completely
  Avoid wildcards or lists: * ? -

Syntax:

```makefile
iecodebook export [if] [in]
         using "/path/to/codebook.xlsx"
```

and optionally:

```makefile
, trim(
    "/path/to/dofile1.do"
    ["/path/to/dofile2.do"]
    [...] )
```
Thank you!

For more information about iefieldkit, contact us or visit our online resources:

dimeanalytics@worldbank.org

dimewiki.worldbank.org/iefieldkit

github.com/worldbank/iefieldkit