The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go “beyond the Stata manual” in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

For more information on the *Stata Journal*, including information for authors, see the web page

http://www.stata-journal.com

The *Stata Journal* is indexed and abstracted in the following:

- Science Citation Index Expanded (also known as SciSearch®)
- CompuMath Citation Index®

**Copyright Statement**: The *Stata Journal* and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp LP. The contents of the supporting files (programs, datasets, and help files) may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the *Stata Journal*.

The articles appearing in the *Stata Journal* may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the *Stata Journal*.

Written permission must be obtained from StataCorp if you wish to make electronic copies of the insertions. This precludes placing electronic copies of the *Stata Journal*, in whole or in part, on publicly accessible web sites, file servers, or other locations where the copy may be accessed by anyone other than the subscriber.

Users of any of the software, ideas, data, or other materials published in the *Stata Journal* or the supporting files understand that such use is made without warranty of any kind, by either the *Stata Journal*, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the *Stata Journal* is to promote free communication among Stata users.

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press. Stata and Mata are registered trademarks of StataCorp LP.
kountry: A Stata utility for merging cross-country data from multiple sources

Rafal Raciborski
Department of Political Science
Emory University
Atlanta, GA
rafal.raciborski@emory.edu

Abstract. This article describes kountry, a data-management command that can be used to translate one country-coding scheme into another, to recode country names into a “standardized form”, and to generate geographic-region variables. Users can build a custom dictionary through a helper command, kountryadd, that “teaches” kountry new name variations. The dictionary can be protected from an accidental overwriting through two helper commands: kountrybackup and kountryrestore.

Keywords: dm0038, kountry, kountryadd, kountrybackup, kountryrestore, country names, country-coding schemes, geographical, region

1 Introduction

Social scientists working with cross-country data know how frustrating and time consuming it is to put together information from multiple sources. Various datasets use various country-coding schemes, which makes the process of merging information tedious and prone to error. As an example, depending on the source, Switzerland can be coded 146, 225, 756, CH, CHE, SWTZ, SWZ, and SZ. The kountry command is designed to simplify this data-management step. In particular, kountry can convert country names into “standardized names” and from one coding scheme into another. Moreover, you can “train” kountry to remember new long-country-name variations. In addition, kountry can generate geographical region variables, which can be used in statistical analysis to control for regional effects. All the features are described in detail below.

2 The kountry command

2.1 Syntax

\texttt{kountry country\_var, from(dataset\_name|other) [to(dataset\_name) geo(geo\_option) marker stuck]}

\texttt{country\_var} is the variable to be converted into another coding scheme. \texttt{country\_var} can be character or numeric.
2.2 Options

from(dataset_name | other) is required and specifies the dataset where country.var comes from. Use dataset_name only if country.var is an actual country code. Supported dataset names are listed in section 2.3. If country.var contains “long” country names, use the keyword other. In either case, a new variable called NAMES_STD will be generated.

to(dataset_name) specifies what coding scheme dataset_name is to be converted to. Supported dataset names are listed in section 2.3. This option generates a new variable called _VAR_, where VAR is a capitalized dataset_name keyword. For example, if the user specifies to(marc), the new variable will be called _MARC_.

geo(geo_option) creates a geographical region variable called GEO. Supported geo_options are listed in section 2.4.

marker generates a variable called MARKER that takes on the value of 1 if country.var was standardized successfully, and 0 otherwise.

stuck is useful when you want to convert from long country names to a coding scheme. This option is described in more detail in section 3.3.

(Continued on next page)
Merging cross-country data from multiple sources

2.3 Country-name coding schemes

`from()` and `to()` accept the following `dataset_names`:

<table>
<thead>
<tr>
<th>Dataset</th>
<th><code>dataset_name</code></th>
<th>character</th>
<th>numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlates of War</td>
<td>cowc</td>
<td>cown</td>
<td></td>
</tr>
<tr>
<td>EUGene</td>
<td>cowc</td>
<td>cown</td>
<td></td>
</tr>
<tr>
<td>International Crisis Behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMF</td>
<td></td>
<td></td>
<td>imfn</td>
</tr>
<tr>
<td>ISO 3166 alpha-2</td>
<td>iso2c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO 3166 alpha-3</td>
<td>iso3c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO 3166 numeric</td>
<td>iso3n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>McClelland(^1)</td>
<td>mcc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARC (Library of Congress)</td>
<td>marc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARGene</td>
<td></td>
<td>cown</td>
<td></td>
</tr>
<tr>
<td>Militarized Interstate Disputes</td>
<td></td>
<td>cown</td>
<td></td>
</tr>
<tr>
<td>National Material Capabilities</td>
<td>capc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penn World Table</td>
<td>iso3c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polity IV</td>
<td></td>
<td>cown</td>
<td></td>
</tr>
<tr>
<td>World Bank</td>
<td>iso3c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNCTAD</td>
<td>unc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UN Stats</td>
<td></td>
<td>iso3n</td>
<td></td>
</tr>
<tr>
<td>Long country names</td>
<td>other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The keyword `other` cannot be specified when `from()` and `to()` are used together. This is because there is no one-to-one mapping from `from(other)` to `to()`. The mapping between the available `dataset_names` is not always perfect either. For example, Belarus is coded `bw` in `marc` but was coded `bwr` before June 1992. So if you use the `to(marc)` option, Belarus will be assigned the most recent code, `bw`. The lack of one-to-one mapping is notorious for post-Soviet states, the former Yugoslavia, and other countries that changed their official name. I suggest you use the `marker` option and tabulate the resulting country-name variable to check for any trouble spots.

For consistency, Prussia and the Federal Republic of Germany are both recoded to Germany; Korea and South Korea, to South Korea; the USSR, the Soviet Union, and the Russian Federation, to Russia; and Serbia and Serbia/Montenegro, to Yugoslavia.

---

1. The McClelland country codes are from the World Event/Interaction Survey (WEIS) Project, 1966–1978, ICPSR Study No. 5211, and they should not be mistaken for the National Material Capabilities dataset, which uses the Correlates of War country codes. A modified McClelland coding scheme is used by the Kansas Event Data System Project. I thank Philipp Fuerst for pointing this out.
2.4 Geographical region schemes

`geo()` accepts the following keywords:

<table>
<thead>
<tr>
<th>geo_option</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cow</td>
<td>Correlates of War “home region”</td>
</tr>
<tr>
<td>marc</td>
<td>MARC (Library of Congress) regions</td>
</tr>
<tr>
<td>men</td>
<td>Middle East regions, “narrow”</td>
</tr>
<tr>
<td>meb</td>
<td>Middle East regions, “broad”</td>
</tr>
<tr>
<td>sov</td>
<td>makes a separate post-Soviet region</td>
</tr>
<tr>
<td>un</td>
<td>UN Stats regions, “broad”</td>
</tr>
<tr>
<td>undet</td>
<td>UN Stats regions, “detailed”</td>
</tr>
</tbody>
</table>

`men` regions are identical to `un` regions, with the exception of the following countries reclassified to the Middle East region:

Bahrain, Egypt, Iran, Iraq, Israel, Jordan, Kuwait, Lebanon, Oman, Palestine, Qatar, Saudi Arabia, Syria, Turkey, United Arab Emirates, and Yemen.

`meb` is identical to `men`, with the exception of adding the following countries to the Middle East region: Afghanistan, Algeria, Djibouti, Libya, Mauritania, Morocco, Pakistan, Somalia, Sudan, Tunisia, and Western Sahara.

`ssov` is identical to `men`, with the exception of the following countries reclassified as being post-Soviet: Armenia, Azerbaijan, Belarus, Estonia, Georgia, Kazakhstan, Kyrgyz Republic, Latvia, Lithuania, Moldova, Russia, Tajikistan, Turkmenistan, Ukraine, and Uzbekistan.

3 Examples

3.1 Merging on standardized names

Imagine you have one cross-national dataset with the IMF coding scheme and another cross-national dataset with the World Bank coding scheme. Here is what the two datasets may look like:

<table>
<thead>
<tr>
<th>imfcode</th>
<th>wbcde</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>AFG</td>
</tr>
<tr>
<td>914</td>
<td>ALB</td>
</tr>
<tr>
<td>612</td>
<td>DZA</td>
</tr>
<tr>
<td>614</td>
<td>AGO</td>
</tr>
<tr>
<td>312</td>
<td>AIA</td>
</tr>
<tr>
<td>311</td>
<td>ATG</td>
</tr>
<tr>
<td>213</td>
<td>ARG</td>
</tr>
<tr>
<td>314</td>
<td>ABW</td>
</tr>
</tbody>
</table>

2. The `un` keyword does not create a Middle East region.
Merging cross-country data from multiple sources

Assuming you have the IMF and World Bank codebooks in front of you, you can merge the two datasets together the hard way by coding something like the following:

```
. use imfdata
. generate str3 wbcode = ""
(8 missing values generated)
. replace wbcode = "AFG" if imfcode == 512
  (1 real change made)
. replace wbcode = "ALB" if imfcode == 914
  (1 real change made)
...
. replace wbcode = "ABW" if imfcode == 314
  (1 real change made)
. sort wbcode
. save imfdata, replace
  file imfdata.dta saved
. use wbdata, clear
. sort wbcode
. merge wbcode using imfdata
```

Or you can use the `kountry` command:

```
. use imfdata, clear
. kountry imfcode, from(imfn)
The command has finished.
The new variable is named NAMES_STD.

. sort NAMES_STD
. save imfdata, replace
  file imfdata.dta saved
. use wbdata, clear
. kountry wbcode, from(iso3c)
The command has finished.
The new variable is named NAMES_STD.

. sort NAMES_STD
. merge NAMES_STD using imfdata
```
In the latter case, the result is

<table>
<thead>
<tr>
<th>imfcode</th>
<th>wbcode</th>
<th>NAMES_STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>AFG</td>
<td>Afghanistan</td>
</tr>
<tr>
<td>914</td>
<td>ALB</td>
<td>Albania</td>
</tr>
<tr>
<td>612</td>
<td>DZA</td>
<td>Algeria</td>
</tr>
<tr>
<td>614</td>
<td>AGO</td>
<td>Angola</td>
</tr>
<tr>
<td>312</td>
<td>AIA</td>
<td>Anguilla</td>
</tr>
<tr>
<td>311</td>
<td>ATG</td>
<td>Antigua and Barbuda</td>
</tr>
<tr>
<td>213</td>
<td>ARG</td>
<td>Argentina</td>
</tr>
<tr>
<td>314</td>
<td>ABW</td>
<td>Aruba</td>
</tr>
</tbody>
</table>

### 3.2 Converting between coding schemes

You can also convert directly from one coding scheme into another, without generating the NAMES_STD variable. Starting with the two datasets as above, you may code

```
. use imfdata, clear
. kountry imfcode, from(imfn) to(iso3c)
```

You are converting from imfn to iso3c....

The command has finished. The new variable is named _ISO3C_

```
. rename _ISO3C_ wbcode
. sort wbcode
. save imfdata, replace
file imfdata.dta saved
. use wbdata
. sort wbcode
. merge wbcode using imfdata
```

The result is

<table>
<thead>
<tr>
<th>imfcode</th>
<th>wbcode</th>
<th>wbcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>AFG</td>
<td></td>
</tr>
<tr>
<td>914</td>
<td>ALB</td>
<td></td>
</tr>
<tr>
<td>612</td>
<td>DZA</td>
<td></td>
</tr>
<tr>
<td>614</td>
<td>AGO</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>AIA</td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>ATG</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>ARG</td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>ABW</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Help, I am stuck!

The stuck option is useful when you need to convert long country names into a coding scheme. You cannot code \texttt{kountry country\_var, from(other) to(dataset\_name)} because there is no one-to-one correspondence between long country names and a coding scheme. Even if you convert \texttt{country\_var} to \texttt{NAMES\_STD}, you still cannot code \texttt{kountry NAMES\_STD, from(other) to(dataset\_name)}. In other words, you are stuck.

The syntax with the stuck option is as follows:

\begin{verbatim}
 kountry country\_var, from(other) stuck [marker]
\end{verbatim}

As a result, a new variable called \_ISO3N\_ is created, and it contains iso3n country codes. From there, you can convert further to a coding scheme of your choice. The reason I chose iso3n is that this scheme covers more countries than alternative coding schemes.

4 kountryadd or when kountry fails

4.1 Preliminaries: The keyword other

When you specify the \texttt{from(other)} option, kountry makes a call to \texttt{k\_other.ado}. That file contains variations of country names, by no means exhaustive, as you will find out sooner or later.

However, before calling \texttt{k\_other.ado}, kountry asks Stata to perform the following operations on your \texttt{country\_var}: \texttt{country\_var} is converted to lowercase; it is stripped of all periods, commas, dashes, double blanks, ampersands, parentheses, square brackets, and \texttt{thes}; and, finally, leading and trailing blanks are removed. This reduces somewhat the variety of country names; for example, after such operations, \texttt{USA}, \texttt{U.S.A.}, \texttt{The USA}, and \texttt{USA, The} are all converted to \texttt{usa}. Only then does \texttt{k\_other.ado} convert \texttt{usa} to the standardized form \textit{United States}. For example, imagine a dataset with country names like this:

\begin{verbatim}
cnames
  Antigua & Barbuda
  Abura
  B.A.H-A.M.As, The
  Falkland Islands (Malvinas)
  Hong Kong, (China)
  Korea, (South) Republic of
  Myanmar [ex-Burma]
  Virgin Islands, U.S.
\end{verbatim}
Because recoding the names by hand is tedious, we proceed in the following way:

```
. kountry cnames, from(other) marker
```

The command has finished.
The new variables are named NAMES_STD and MARKER.

```
. list NAMES_STD MARKER
+-----------------+-----------------+
| NAMES_STD       | MARKER          |
|-----------------+-----------------|
| Antigua and Barbuda | 1              |
| abura           | 0              |
| Bahamas         | 1              |
| Falkland Islands| 1              |
| Hong Kong       | 1              |
| South Korea     | 1              |
| Myanmar         | 1              |
| United States Virgin Islands | 1  |
+-----------------+-----------------+
```

All the country names except one have been standardized successfully, a very pleasing result indeed. We can fix the failed name by typing

```
. replace NAMES_STD = "Aruba" if NAMES_STD == "abura"
```

but there is a better way.

### 4.2 Enter kountryadd

When you get tired of typing

```
. replace NAMES_STD = "Aruba" if NAMES_STD == "abura"
```

over and over again, you may consider using kountryadd. kountryadd is a helper file in the kountry family that adds alternative country name spellings to the main kountry command. The syntax is

```
kountryadd "failed name" to "standardized name" add
```

where failed name is the country name that failed to standardize, and standardized name is a country name spelled exactly as in the list of standardized country names.\(^3\) Double quotes are required. For example, to let kountry know that in the future we wish to convert any instances of “abura” into “Aruba”, we type

```
. kountryadd "abura" to "Aruba" add
```

---

\(^3\) The list can be obtained by typing help kountrynames in Stata.
Now `kountry` is “trained” to recognize the new name variation:

```
. drop NAMES_STD MARKER
. kountry cnames, from(other) marker
```

The command has finished.
The new variables are named `NAMES_STD` and `MARKER`.

```
. list NAMES_STD MARKER

NAMES_STD       MARKER
----------------------
        Antigua and Barbuda 1
            Aruba 1
          Bahamas 1
     Falkland Islands 1
              Hong Kong 1
          South Korea 1
            Myanmar 1
United States Virgin Islands 1
```

It does not matter how failed name is entered because `kountryadd`, just like `kountry`, will convert the name to lowercase and strip all periods, commas, dashes, double blanks, ampersands, parentheses, square brackets, and thes. For example, coding

```
. kountryadd "nerlands" to "Netherlands" add
```

is equivalent to coding:

```
. kountryadd "Netherlands, The" to "Netherlands" add
```

In either case, `kountryadd` will return a message that looks something like this:

```
The following line has been added to c:\ado\plus/k/k_other_extras.txt:
    replace NAMES_STD = "Netherlands" if NAMES_STD == "nerlands"
```

### 4.3 `kountrybackup` and `kountryrestore`

`kountryadd` saves your corrected spellings in files called `k_other_extras.txt` and `k_other_extras.ado`. However, the `kountry` command does not come preloaded with `k_other_extras.ado`. When you install `kountry`, you get, among other files, a “blank” `k_other_extras.txt`. When you use `kountryadd` for the first time, Stata creates `k_other_extras.ado` from the text file, adding a line with the corrected spellings. Subsequent calls to `kountryadd` add extra spellings to the text file and overwrite the existing ado-file. Over time, the file can grow substantially in size, reflecting hundreds of alternative spellings. Unfortunately, if the user chooses to update or reinstall `kountry`, the dictionary will be overwritten with the default blank text file, and the very first call to `kountry` will overwrite the ado-file as well.
One option is to copy the text file to a safe location, reinstall `kountry`, and then replace the newly installed `k_other_extras.txt` with the old one.

An alternative is to use two helper commands, `kountrybackup` and `kountryrestore`. The syntax is simply

```
kountrybackup [, replace]
```

and

```
kountryrestore
```

`kountrybackup` creates a `.bak` file that is a copy of `k_other_extras.txt`, while `kountryrestore` replaces the newly installed blank text file with the backup file.

Thus, if you plan to update `kountry`, you can issue the command

```
. kountrybackup, replace
```

reinstall `kountry`, and type

```
. kountryrestore
```

This is all you have to do.

## 5 Further `kountry` internals

### 5.1 The stuck option

When you specify the `stuck` option, `kountry` invisibly converts your `country_var` to `NAMES_STD`, and then converts `NAMES_STD` to `_ISO3N_`. There is a one-to-one mapping from `NAMES_STD` to `_ISO3N_`. However, there is still no guarantee that the final output will be correct. This is why I recommend specifying `stuck` with `marker`. `marker` gives you a peek at how good or bad the internal conversion step was. You can code

```
. kountry myvar, from(other) stuck marker
```

```
. tabulate myvar if MARKER == 0
```

If the tabulation returns any entries, you should recode the offending names by hand or fix them with `kountryadd` and then rerun `kountry` with the `stuck` option.

The `stuck` option can get painfully slow with large datasets because it calls three ado-files with many `replace ... if` lines. Because `if` conditions are evaluated line-by-line, Stata will take a while to finish. For example, a computer with an AMD Athlon 64 processor 3800+ with 1 GB of RAM took 512 seconds to process a dataset with 1.2 million observations, whether the `marker` option was specified or not.
5.2 A remark about from() and to()

When from() and to() are specified together, Stata is asked to call merge, which grabs the required variable from the kountry.dta dataset. If you installed kountry with a web-aware Stata, the dataset should reside in your ado/plus/k folder or directory. If Stata cannot find the dataset, make sure to move kountry.dta manually to the ado/plus/k location. To see the path to the PLUS directory, type sysdir in Stata.

kcountry can get very slow with large datasets when from() and to() are specified together. This is because kcountry calls preserve, which causes Stata to preserve your data in memory, open the dataset with country codes, sort on the to() variable, then restore your data, and finally perform the merger. For example, a computer with an AMD Athlon 64 processor 3800+ with 1 GB of RAM took 175 seconds to process a dataset with 1.2 million observations, and specifying marker extended the time to 218 seconds.

6 Conclusion

I hope that the kountry command offers a useful addition to data-management tools. kcountry automates the process of manual recoding of hundreds of country names or codes, as well as “geographic region” variables. You have a choice of converting from one coding scheme to another or of converting to the “standardized” form of country names. The command is easily extendable, even to those with rudimentary Stata programming skills, and it is easy to add new country name variations via the kountryadd command.

I also hope that kountry will instill good data-management practice, especially in those who are just being introduced to data collection and analysis. From my experience, beginner data analysts are eager to work with the data that are provided to them, but are usually helpless when it comes to incorporating outside information into a dataset. I hope kountry will provide a welcome alternative to hours of typing raw data into a spreadsheet.

7 Acknowledgments

I thank Philipp Fuerst for testing the command and an anonymous referee for helpful suggestions.

About the author

Rafal Raciborski is a graduate student in the Department of Political Science at Emory University. His interests include political economy, comparative welfare states, applied econometrics, and statistical computing.