The Stata Journal (2008)
8, Number 3, pp. 390–400

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

`kountry` *country_var*`, from(`*dataset_name*`|other)` [`to(`*dataset_name*`)`
`geo(`*geo_option*`)` `marker` `stuck`]

country_var is the variable to be converted into another coding scheme. *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_name*s 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_name*s 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_option*s 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*)

2.3 Country-name coding schemes

`from()` and `to()` accept the following *dataset_name*s:

	dataset_name	
Dataset	character	numeric
Correlates of War	`cowc`	`cown`
EUGene	`cowc`	`cown`
International Crisis Behavior		`cown`
IMF		`imfn`
ISO 3166 alpha-2	`iso2c`	
ISO 3166 alpha-3	`iso3c`	
ISO 3166 numeric		`iso3n`
McClelland[1]	`mcc`	
MARC (Library of Congress)	`marc`	
MARGene		`cown`
Militarized Interstate Disputes		`cown`
National Material Capabilities	`capc`	
Penn World Table	`iso3c`	
Polity IV		`cown`
World Bank	`iso3c`	
UNCTAD	`unc`	
UN Stats		`iso3n`
Long country names	`other`	

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_name*s 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:

geo_option	description
`cow`	Correlates of War "home region"
`marc`	MARC (Library of Congress) regions
`men`	Middle East regions, "narrow"
`meb`	Middle East regions,"broad"
`sov`	makes a separate post-Soviet region
`un`	UN Stats regions, "broad"
`undet`	UN Stats regions, "detailed"

`men` regions are identical to `un` regions, with the exception of the following countries reclassified to the Middle East region:[2] 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.

`sov` 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:

imfcode
512
914
612
614
312
311
213
314

wbcode
AFG
ALB
DZA
AGO
AIA
ATG
ARG
ABW

2. The `un` keyword does not create a Middle East region.

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

imfcode	wbcode	NAMES_STD
512	AFG	Afghanistan
914	ALB	Albania
612	DZA	Algeria
614	AGO	Angola
312	AIA	Anguilla
311	ATG	Antigua and Barbuda
213	ARG	Argentina
314	ABW	Aruba

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

imfcode	wbcode
512	AFG
914	ALB
612	DZA
614	AGO
312	AIA
311	ATG
213	ARG
314	ABW

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 `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 *country_var* to `NAMES_STD`, you still cannot code `kountry NAMES_STD, from(other) to(`*dataset_name*`)`. In other words, you are stuck.

The syntax with the `stuck` option is as follows:

`kountry` *country_var*`, from(other)` st`uck` [`marker`]

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 `from(other)` option, `kountry` makes a call to `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 `k_other.ado`, `kountry` asks Stata to perform the following operations on your *country_var*: *country_var* is converted to lowercase; it is stripped of all periods, commas, dashes, double blanks, ampersands, parentheses, square brackets, and *the*s; and, finally, leading and trailing blanks are removed. This reduces somewhat the variety of country names; for example, after such operations, *USA*, *U.S.A.*, *The USA*, and *USA, The* are all converted to *usa*. Only then does `k_other.ado` convert *usa* to the standardized form *United States*. For example, imagine a dataset with country names like this:

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.

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 *the*s. 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.

`kountry` can get very slow with large datasets when `from()` and `to()` are specified together. This is because `kountry` 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. `kountry` 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.