6.2 Applying `reshape` repeatedly

The problem: are your data the wrong shape?[4] That is, are they not organized in the structure that you need to conduct the analysis you have in mind? Data sources often provide the data in a structure quite suitable for presentation but very clumsy for statistical analysis. One of the key data management tools Stata provides is `reshape` ([D] **reshape**). If you need to modify the structure of your data, you should surely be familiar with `reshape` and its two functions: `reshape wide` and `reshape long`. In some cases, you may have to apply `reshape` twice to solve a particularly knotty data management problem.

As a first example, consider this question, posed on Statalist, by an individual who has a dataset in the wide form:

country	tradeflow	Yr1990	Yr1991
Armenia	imports	105	120
Armenia	exports	90	100
Bolivia	imports	200	230
Bolivia	exports	80	115
Colombia	imports	100	105
Colombia	exports	70	71

He would like to reshape the data into the long form:

country	year	imports	exports
Armenia	1990	105	90
Armenia	1991	120	100
Bolivia	1990	200	80
Bolivia	1991	230	115
Colombia	1990	100	70
Colombia	1991	105	71

We must exchange the roles of years and tradeflows in the original data to arrive at the desired structure, suitable for analysis as `xt` data. This can be handled by two successive applications of `reshape`:

```
. clear
. input str8 country str7 tradeflow Yr1990 Yr1991
      country  tradeflow     Yr1990     Yr1991
  1. Armenia imports 105 120
  2. Armenia exports 90 100
  3. Bolivia imports 200 230
  4. Bolivia exports 80 115
  5. Colombia imports 100 105
  6. Colombia exports 70 71
```

4. This recipe is adapted from Stata Tip 45 (Baum and Cox (2007)). I am grateful to Nicholas J. Cox for his contributions to this Stata Tip.

```
  7. end
. reshape long Yr , i(country tradeflow)
(note: j = 1990 1991)
Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                        6   ->      12
Number of variables                   4   ->       4
j variable (2 values)                     ->   _j
xij variables:
                           Yr1990 Yr1991   ->   Yr
-----------------------------------------------------------------------------
```

This transformation swings the data into long form with each observation identified by `country`, `tradeflow` and the new variable `_j`, taking on the values of year. We now perform `reshape wide` to make imports and exports into separate variables:

```
. rename _j year
. reshape wide Yr, i(country year) j(tradeflow) string
(note: j = exports imports)
Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                       12   ->       6
Number of variables                   4   ->       4
j variable (2 values)         tradeflow   ->   (dropped)
xij variables:
                                     Yr   ->   Yrexports Yrimports
-----------------------------------------------------------------------------
```

Transforming the data to wide form once again, the `i()` option contains `country` and `year` as those are the desired identifiers on each observation of the target dataset. We specify that `tradeflow` is the `j()` variable for `reshape`, indicating that it is a `string` variable. The data now have the desired structure. Although we have illustrated this double-reshape transformation with only a few countries, years and variables, the technique generalizes to any number of each.

As a second example of successive applications of `reshape`, consider the World Bank's *World Development Indicators* (WDI) dataset.[5] Their extract program generates a comma-separated-value (CSV) database extract, readable by Excel or Stata, but the structure of those data hinders analysis as panel data. For a recent year, the header line of the CSV file is:

```
"Series code","Country Code","Country Name","1960","1961","1962","1963",
"1964","1965","1966","1967","1968","1969","1970","1971","1972","1973",
"1974","1975","1976","1977","1978","1979","1980","1981","1982","1983",
"1984","1985","1986","1987","1988","1989","1990","1991","1992","1993",
"1994","1995","1996","1997","1998","1999","2000","2001","2002","2003","2004"
```

That is, each row of the CSV file contains a *variable* and *country* combination, with

5. `http://econ.worldbank.org`

the columns representing the elements of the timeseries.[6]

Our target dataset structure is that appropriate for panel-data modeling with the variables as columns and rows labeled by country and year. Two applications of `reshape` will again be needed to reach the target format. We first `insheet` ([D] **insheet**) the data and transform the triliteral country code into a numeric code with the country codes as labels:

```
. insheet using wdiex.raw,comma names
. encode countrycode, generate(cc)
. drop countrycode
```

We then must deal with the fact that the timeseries variables are named `var4-var48`, as the header line provided invalid Stata variable names (numeric values) for those columns. We use *rename* ([D] **rename**) to change `v4` to `d1960`, `v5` to `d1961` and so on, as described in Section 3.6. We use a technique for macro expansion, involving the equals sign, by which an algebraic expression may be evaluated within a macro. In this case, the target variable name contains the string `1960`, `1961`, ..., `2004`:

```
. forvalues i=4/48 {
.   rename v`i' d`=1956+`i''
. }
```

We now are ready to carry out the first `reshape`. We want to identify the rows of the reshaped dataset by both countrycode (`cc`) and `seriescode`, the variable name. The `reshape long` will transform a fragment of the WDI dataset containing two series and four countries:

```
reshape long d, i(cc seriescode) j(year)
(note: j = 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 197
> 3 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
> 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 20
> 04)
Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                        7   ->     315
Number of variables                  48   ->       5
j variable (45 values)                    ->   year
xij variables:
                  d1960 d1961 ... d2004   ->   d
-----------------------------------------------------------------------------
```

```
list in 1/15
```

	cc	seriesc~e	year	countryname	d
1.	AFG	adjnetsav	1960	Afghanistan	.
2.	AFG	adjnetsav	1961	Afghanistan	.
3.	AFG	adjnetsav	1962	Afghanistan	.
4.	AFG	adjnetsav	1963	Afghanistan	.
5.	AFG	adjnetsav	1964	Afghanistan	.
6.	AFG	adjnetsav	1965	Afghanistan	.

6. A variation occasionally encountered will resemble this structure, but with time periods in reverse chronological order. The solution below can be used to deal with that problem as well.

7.	AFG	adjnetsav	1966	Afghanistan	.
8.	AFG	adjnetsav	1967	Afghanistan	.
9.	AFG	adjnetsav	1968	Afghanistan	.
10.	AFG	adjnetsav	1969	Afghanistan	.
11.	AFG	adjnetsav	1970	Afghanistan	-2.97129
12.	AFG	adjnetsav	1971	Afghanistan	-5.54518
13.	AFG	adjnetsav	1972	Afghanistan	-2.40726
14.	AFG	adjnetsav	1973	Afghanistan	-.188281
15.	AFG	adjnetsav	1974	Afghanistan	1.39753

The rows of the data are now labeled by year but one problem remains: all variables for a given country are stacked vertically. To unstack the variables and put them in shape for `xtreg` ([XT] **xtreg**), we must carry out a second `reshape` which spreads the variables across the columns, specifying `cc` and `year` as the i variables and j as `seriescode`. As that variable has string content we use the `string` option.

```
reshape wide d, i(cc year) j(seriescode) string
(note: j = adjnetsav adjsavCO2)
Data                                long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                       315   ->     180
Number of variables                    5   ->       5
j variable (2 values)        seriescode   ->   (dropped)
xij variables:
                                       d   ->   dadjnetsav dadjsavCO2
-----------------------------------------------------------------------------

order cc countryname

tsset cc year
       panel variable:  cc (strongly balanced)
        time variable:  year, 1960 to 2004
```

After this transformation, the data are now in shape for `xt` modeling, tabulation or graphics.

As illustrated here, the `reshape` command can transform even the most inconvenient data structure into the structure needed for your research. It may take more than one application of `reshape` to get there from here, but it can do the job.