Introduction to Data Manipulation in R

Ryan Womack*

May 30, 2010

*Data and Economics Librarian, Rutgers University, New Brunswick, NJ, rwomack@rci.rutgers.edu

Abstract

This is a guide to basic data handling in R. First, basic file commands and R conventions are reviewed. Second, simple object creation and editing is illustrated. Third, importing and conversion of data files is illustrated with a real-life example, using the open data available from the World Bank. Fourth, data manipulation through cuts, subsets, transforms, and merges are demonstrated. Finally, advanced features available through additional packages are sketched.

This material was developed for the "Introduction to R" workshop, presented with Harrison Dekker, at the IASSIST Annual Conference, June 1, 2010, held at Cornell University, Ithaca, NY.

1 Preliminaries

All code is fully illustrated in this document. A file containing only the R commands (DataManipulation.R) is available on my personal home page. This may be useful for executing commands via cut-and-paste, or for review. The data files are created using live links to the World Bank website. If these are not functional for some reason, equivalent data files can also be found on my personal home page.

There are many useful references to R, but this document in particular relies on the author's study of three books, R for SAS and SPSS UsersRobert A. Muenchen [2009], Data Manipulation with RPhil Spector [2008], and Introductory Statistics with RPeter Dalgaard [2008].

If you do not already have R, obtaining and installing it is easy. Since R is open source software, licensed under the GPL, you can use it freely for just about anything except creating closed source software. Information about R is available at the R project site and the software itself is downloadable from CRAN, the Comprehensive R Archive Network, comprised of synchronized mirror sites around the world. Also, freedom means that you can install R in as many locations as you like: all of your public workstations, your web servers, your home machines, your netbook, USB drives, your friends' machines, ... you get the idea.

You can download Windows, Linux, or Mac versions. The Windows version of the base package is a self-contained executable containing all necessary files to get your R installation running. The contrib package contains additional modules, or *packages*, as they are known in R parlance. It is usually easier to download and install packages individually as you need them, as explained below.

In Linux, it is possible to install R from source, but it will usually be more convenient to wait for the latest version to be packaged for release in a major Linux distribution, such as Ubuntu or Fedora, and downloaded and installed using that distribution's tools. This simplifies the resolution of dependencies and staying current with updates. Rpms and .deb files are also available from CRAN, but may not always be in sync with the latest R version.

The Mac version is also available at CRAN as a downloadable package, although the author has no experience with it.

2 Getting Around in R

Once you have installed R, you are ready to run it - in Windows by clicking the R icon, or in Linux by simply typing R at the terminal prompt. Now you are presented with the most challenging part of your R

experience, the empty command line. What to do?

You can operate R entirely from the command line, entering text in interactive mode. For now, let's try typing some commands.

There are a few basic commands that will help you navigate your workspace. First, let's find out where we are.

Type

> getwd()

[1] "/home/ryan/R/data/IASSIST/Presentation/Code"

This command will show you the default path for your R files. Now type getwd, this time without the parentheses.

> getwd

```
function ()
.Internal(getwd())
<environment: namespace:base>
```

What you see now is the actual definition of the function in R. This is a nifty feature that gives you a clue to one of the primary characteristics of R. It is simple yet powerful at the same time. Typing any function name without its arguments will return the function itself. This becomes more interesting as you access functions created by other contributors in their packages, and can see exactly how their tools work. And you can use this functionality to easily modify existing functions and create your own. Any arguments to a function are enclosed within parentheses. With nothing inside the parentheses (), R will use the default values and settings for the function.

Note that R is case-sensitive so

> Getwd

will not work.

You can change working directory by typing

> setwd("pathname")

Within R, lots of Unix conventions are used, so paths are specified with a single forward-slash separator, even on Windows systems. So setwd("C:/Documents and Settings/username/My Documents") would be used to point to the My Documents directory.

You can list the objects in your workspace with ls(), and remove them with rm("objectname"). Notice that we don't have much in our workspace yet, but we will after we have created some objects.

We have talked about packages already. In order to use a package, you must install it. Let's do this for a couple of things that we will need.

```
> install.packages("Hmisc", dependencies = TRUE)
> install.packages("reshape", dependencies = TRUE)
> update.packages()
```

R will ask us which mirror we want to use. Choose your favorite country. The dependencies=TRUE option will check for other required packages and install those too. You probably want to do this, unless you are really fine-tuning your system or are a control freak! The final call to update the packages is how you would maintain your system. Empty parentheses will update all installed packages.

R will automatically locate packages that have been officially accepted into CRAN without trouble at all. Also, since R is flexible and powerful, it is also relatively easy to create your own packages with your own custom functions and data included, which you might distribute locally. For local packages, you'd have to specify an explicit path to where R could find the package.

You can type library() to see all of the available packages that have been installed on your system. The search() command will show what has been actively loaded in your current environment.

To load a specific package, use the library command again, but with an argument this time. We can now see that it is loaded with search().

Finally, even though we haven't yet created any data or output worth saving, we can learn how to save and load our workspace. A simple <code>save("objectname")</code> command will save a single item from a workspace (for example, a matrix that you created). To save the entire workspace, use the <code>save.image</code> command. One of the very useful features of R is the ability to save not only data and output files, but to save all functions and intermediate objects created in the course of a session as part of the workspace. A complete workspace of this kind is usually saved with the extension <code>.RData</code>.

```
> save.image("mydata.RData")
> load("mydata.RData")
```

3 R concepts

Let's create a small practice dataset to get our feet wet. Base R has a bare bones data editor and viewer. If you need more functionality in this area, some of the add-on GUIs and editors for R will let you do much more (RCmdr, RKWard). In Windows, you can use the Data Editor from the menu. Or, we can start the data editor from the command line. To do this, we must first create the data object that we will edit. Here's how we do it:

```
> testdata <- data.frame()
> testdata

data frame with 0 columns and 0 rows
```

We are telling R to create an object called testdata. The arrow (composed by typing the less-than sign and a hyphen) is the "assignment" operator and assigns to testdata the value of whatever comes after the arrow. You can reverse the arrows direction, or use the equals sign. However, the arrow is preferred in R because there are a few situations where the equals sign is syntactically ambiguous. The equals sign is routinely used for setting parameter values in functions, as you can see from the above example. So, stick with the <- arrow for assignment, unless you find it too troublesome. Here, we set up testdata as a data frame.

By typing testdata at the end, we are actually issuing a command equivalent to print(testdata). Rather than typing print(testdata), we just use the name as a shortcut.

A data frame is a special data construct in R, and is the closest equivalent to the typical rectangular dataset produced by SAS or SPSS and used for social science data. In R, the columns are called vectors, variables, or just columns, while the rows are cases, observations, or just rows. After we define some of the other R data structures, we will return for a revised definition of a data frame in R terms.

Let's fill in our data frame in the editor. Create 10 cases, changing the values and variable names to match the following table:

```
sex age
1
      М
         22
2
      F
         35
3
         43
4
      F
         52
5
      М
         58
6
      F
         23
```

```
7 M 36
8 F 46
9 M 39
10 F 31
```

We can also create simpler data objects. Let's create a vector that will contain 10 additional observations on our hypothetical subjects. Let's suppose this is the number of cups of coffee per day that each person drank yesterday.

```
> coffee = c(3, 1, 2, 5, 0, 2, 0, 1, 3, 2)
> coffee
[1] 3 1 2 5 0 2 0 1 3 2
```

Note that the [1] is an index number, telling us that R is printing beginning with item 1. This is more useful when working with larger datasets. We used the c() function, which stands for concatenate, to create our vector. This function can be used not only for individual numbers or strings, but can also be used to create a list of more complex objects. It is one of the basic building blocks of data structures in R.

Now we will use the mode() and class() functions to inspect the objects we have created so far. Every object in an R environment will have these characteristics.

```
> mode(coffee)
[1] "numeric"
> mode(testdata)
[1] "list"
> mode(testdata$age)
[1] "numeric"
> mode(testdata$sex)
[1] "numeric"
> class(coffee)
[1] "numeric"
> class(testdata)
[1] "data.frame"
> class(testdata$age)
[1] "factor"
> class(testdata$sex)
[1] "factor"
```

The mode of an object describes the nature of the contents of the object. For a vector the mode can be numeric, character, logical or factor. Numeric and character mean what you might expect. A logical vector contains a list of True or False values. We'll talk more about factors in just a moment. Note that coffee is numeric. The mode of testdata is "list", which means that it is composed of more than one sub-object. We can peek inside testdata by using the \$ symbol to access variables within testdata. The mode of age is numeric. The mode of sex is also numeric. That's a surprise! Didn't we type in "M" and "F"? We did, but R will automatically convert any character variables to factors. This is partly to save memory and partly to aid analysis.

A factor is an object with levels. In the case of sex, R has converted all the M's to 2 and all the F's to 1. We can see this by using the as.numeric command to force display of the numeric values:

> as.numeric(testdata\$sex)

[1] 2 1 2 1 2 1 2 1 2 1

We can also adjust how a factor is labeled. The labels are entered in alphabetical order. This is why R ranked females first (not that there couldn't be other reasons for that!) Note the change in the output.

```
> testdata$sex <- factor(testdata$sex, labels = c("Female", "Male"))
> testdata$sex
```

[1] Male Female Male Female Male Female Male Female Male Female Levels: Female Male

The class of an object describes its structure. Here we have a vector and a data frame. A *vector* is a one-dimensional list of entries of the same mode, and can be of arbitrary length. R will automatically set the vector to the least restrictive mode. So, mytest<-c("M", 1, 3) has mode "character". Other classes are *list*, a group of objects that can be of different modes; matrix, an object of dimension dim(x,y) whose elements must be of the same mode; and array, which is like a data frame but in higher dimensions. Now we can be more precise about a data frame too. The *data frame* is an object of dimension dim(x,y), whose elements can be of different modes, but whose rows all have the same length.

Essentially these data structures behave as you would expect, and R will gracefully handle many of the details automatically. Unlike other software, R does not mask these complexities entirely, and you will need to keep in mind that certain operations will only work with certain data structures. An understanding of data structures will help you to design and debug your R programs.

Let's do one final task with our practice dataset. We want to add the coffee data into the testdata to form a single combined data frame. A simple way to do this, which will work if the number of observations is the same, is to bind on an extra column with cbind. We must overwrite our existing testdata with the assignment operator in order for the changes to stick.

```
> testdata <- cbind(testdata, coffee)</pre>
```

> testdata

```
sex age coffee
            22
                      3
     Male
1
2
   Female
            35
                      1
3
            43
                      2
     Male
4
   Female
            52
                      5
                      0
5
     Male
            58
            23
                      2
6
   Female
7
                      0
     Male
            36
8
   Female
            46
                      1
9
     Male
            39
                      3
10 Female
```

We conclude by taking a look at the objects in our R space, with ls() and the useful summary command, which provides a snapshot of any dataset. We can also produce basic tables using the table command. Note how the different arguments produce different versions of the table.

```
> 1s()
[1] "age"
                  "coffee"
                                 "myPackages" "sex"
                                                             "testdata"
> summary(testdata)
                              coffee
     sex
                  age
                                  :0.00
Female:5
             22
                     :1
                          Min.
                          1st Qu.:1.00
Male:5
             23
                     : 1
```

```
31
                          Median:2.00
                    :1
             35
                    :1
                          Mean
                                 :1.90
                          3rd Qu.:2.75
             36
                    :1
             39
                                  :5.00
                    :1
                          Max.
             (Other):4
> table(coffee)
coffee
0 1 2 3 5
2 2 3 2 1
> table(coffee, sex)
      sex
coffee F M
     0 0 2
     1 2 0
     2 2 1
     3 0 2
     5 1 0
```

Now we've a done a little data entry and manipulation!

4 Importing and Converting Data

If the only way to get data was to type it in manually, we wouldn't get very far. Of course manual data entry builds character, but try convincing an undergraduate of that! Fortunately R provides convenient ways of importing data in a variety of formats.

For direct data input, there is an additional command called scan, which allows you to input data directly from console, from a file, or from a web site or other raw data source. The scan command takes the argument what to specify what is being entered.

For example we could have entered our coffee data with the following:

```
> coffee <- scan(what = "numeric")</pre>
```

Try it out. Entering a blank line stops the process. The scan command is useful to know about, but in most cases, you will want to import an already structured data file.

The read table command is the Swiss army knife of file importing in R, and can handle any kind of delimited file. In its simplest form, read table needs only a filename as an argument.

```
> importdata <- read.table("myfile.txt")</pre>
```

However, this will only work correctly if all of the default assumptions are met. R will correctly read a tab or space delimited file that has variable names in the header row and a first line with a length one shorter than subsequent lines (that is, a single blank cell in the upper left hand corner of the matrix). The default representation for a missing value is NA, and R will not correctly read SAS or SPSS files that use two tabs in a row to represent a missing value.

We can enter parameters to the command to adjust for any unique characteristics of our data. For example,

```
> importdata <- read.table("myfile.txt", header = TRUE, sep = ";",
+ row.names = "id", na.strings = "..", stringsAsFactors = FALSE)</pre>
```

This tells R to read a data file with a header row and semi-colon separated data. R will use the variable named "id" as the identifier for the rows in the R data frame. R will convert the .. characters in the original data to NA in R. We can also use parameters to force the reading of certain variables as characters or integers. Here we tell R not to convert strings into factors. In general, that is a good idea for things like names and addresses, where there is no real use for the "levels" that a factor representation of the variable would generate.

There are also functions read.csv, read.delim, read.csv2 (for euro-style separators (; and . replacing , in numbers) with reasonable default values for other typical files. There is no difference between these functions and read.table, except that using them saves the entry of several extra parameter options. Also, you can access data placed on the clipboard with read.table("clipboard", header=T).

When you are ready to export data from your R environment, you can use write.table, write.csv, and so on. These functions have all the same parameters as their read versions, but create delimited tables.

R also has a package called foreign which eases working with other data formats, such as SAS, SPSS, and Stata.

Here is an example of importing a SAS file, then an SPSS file. The documentation for foreign explains further options. We are also detaching the package once we are finished using it, just to keep our workspace tidy and avoid potential function conflicts.

```
> library(foreign)
> importdata <- sasxport.get("mydata.xpt")
> importdata <- spss.get("mydata.sav")
> detach(package:foreign)
```

You can try using the foreign package on any ICPSR dataset of your choosing, or try the Pew Foundations free SPSS data downloads.

Just as in the case of read.table and write.table, foreign allows you to write datasets in other formats using the write.foreign command.

5 World Bank Data

World Bank Open Data provides access to the full contents of many major datasets related to economic growth and human development collected by the Bank. As an example of the power of R in handling real world data, we will grab two complete datasets from this site, import them into R, then subset, transform, and merge the data into a compact file containing selected variables and countries of interest.

The World Bank's Data Catalog provides descriptions, downloads, and web interfaces to the data collections. We will use two of these datasets, Gender Statistics and the Millenium Development Indicators . In our example, we can think of a researcher who is interested in isolating variables associated with fertility and gender differences and comparing them with indicators of the availability of modern communications technology. For now, this researcher is only interested in the five most populous countries in the world: China, India, the United States, Indonesia, and Brazil. We will create a customized data extract for our hypothetical researcher.

Importing Data

First, we need to import the data from the website. Although there are other possibilities, the commadelimed .csv file is the most straightforward to use.

We import each dataset using the read.csv function. The defaults for read.csv usually work well, so we will just run it, with the file location specifying the direct link to the World Bank website. There is a bit of trick here to dealing with the zip file: if the file was not zipped, we could simply use the URL as the file location in the read.csv function. We also could have simply downloaded the file to our local PC and manually unzipped it outside of R, and then used read.csv on it.

Here is the code to download the *Gender Statistics* data file, unzip it, and read it from .csv format into an R data frame. Note that there are three separate .csv files in the zip archive, so we specifically reference the one that contains the main data. Finally, we will examine the head of the data frame just to make sure everything ran correctly.

```
> download.file("http://databank.worldbank.org/databank/download/Gender_Stats_csv.zip",
       "Gender.zip")
> unzip("Gender.zip")
> genderstats <- read.csv("Gender_Stats_Data.csv")</pre>
> head(genderstats)
  Country_code Country_Name Indicator_code
                                                                      Indicator_name
1
                  Afghanistan IC.FRM.FEMM.ZS Firms with female top manager (%)
2
            ARM
                      Armenia IC.FRM.FEMM.ZS Firms with female top manager (%)
3
            AZE
                   Azerbaijan IC.FRM.FEMM.ZS Firms with female top manager (%)
4
            BEN
                        Benin IC.FRM.FEMM.ZS Firms with female top manager (%)
            BFA Burkina Faso IC.FRM.FEMM.ZS Firms with female top manager (%)
6
                     Bulgaria IC.FRM.FEMM.ZS Firms with female top manager (%)
  X1960 X1961 X1962 X1963 X1964 X1965 X1966 X1967 X1968 X1969 X1970 X1971 X1972
                   NA
                         NA
                                NA
                                                                                NA
     NA
            NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
1
2
     NA
            NA
                   NA
                         NA
                                ΝA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
3
     NA
            NA
                   NA
                         NA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
4
                                NA
     NA
            NA
                   NA
                         NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
5
     NA
            NA
                   NA
                          ΝA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
6
     NA
            NA
                   NA
                         NA
                                ΝA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
  X1973 X1974
               X1975
                      X1976
                             X1977
                                    X1978
                                          X1979
                                                 X1980
                                                        X1981
                                                               X1982
                                                                     X1983 X1984 X1985
     NA
            NA
                   NA
                         NA
                                ΝA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
1
                          ΝA
                                NA
2
     NA
            NA
                   NA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
3
     NA
            NA
                   NA
                         NA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
4
                   NA
                         NA
                                NA
     NA
            NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
5
     NA
            NA
                   NA
                         NA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
                                                                                       NA
                         NA
                                                                                       NA
6
     NA
            NA
                   NA
                                NA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                         NA
                                                                                NA
  X1986 X1987 X1988
                      X1989
                                    X1991 X1992 X1993
                                                        X1994
                                                               X1995
                             X1990
                                                                     X1996 X1997 X1998
1
     NA
            NA
                   NA
                         NA
                                NA
                                       NA
                                              NA
                                                     NA
                                                           NA
                                                                  NA
                                                                         NA
                                                                                NA
2
     NA
            NA
                   NA
                         NA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                         NA
                                                                                NA
                                                                                       NA
                                                                  NA
3
     NA
                         NA
                                                                                NA
            NA
                   NA
                                NA
                                       NA
                                              NA
                                                    NA
                                                           NA
                                                                  NA
                                                                         NA
```

NA

X2017

X2004

NA

X2018 X2019

X2005

NA

NA

NA

NA

NA

NA

NA

NA

NA

NΑ

NA

NA

X2006

NA

NA

NA

NA

NA

NA

NA

NA

NA

NΑ

NA

NA

X2007

NA

NA

NA

NA

NΑ

NA

NA

X2008

0.75

NA

X2010 X2011

NA

X2009

4.72

NA 13.49

NA 27.50

NA 11.32

NA 25.75

X2020 X2021 X2022 X2023 X2024

NΑ

NA

NA

4

5

6

1 2

3

4

5

1

2

3

4

NA

NΑ

X1999

NA

X2012 X2013 X2014

X2000

NA

NA

NA

NA

NA

NA

NA

NA

NA

NΑ

NA

NA

X2001

NA

X2015

X2002

NA

ΝA

NA

NA

ΝA

NA

NA

NA

NA

NA

NA

NA

NA

X2016

X2003

```
6
     NA
           NA
                 NA
                                                NA
                                                                         NA
                        NA
                                          NA
                                                       NA
                                                             NA
                                                                   NA
  Now we will repeat the process to import the Millenium Development Indicators.
> download.file("http://databank.worldbank.org/databank/download/MDG_csv.zip",
      "MDI.zip")
> unzip("MDI.zip")
> MDstats <- read.csv("MDG_Data.csv")
> head(MDstats)
  Country.Code Country.Name
                                   Series.Code
                                AG.LND.FRST.K2
1
           ABW
                      Aruba
2
           ABW
                       Aruba
                                AG.LND.FRST.ZS
3
           ABW
                      Aruba
                                DT.ODA.ALLD.CD
4
           ABW
                       Aruba
                                DT.ODA.ODAT.CD
5
                      Aruba DT.ODA.ODAT.GN.ZS
           ABW
                       Aruba DT.ODA.ODAT.PC.ZS
6
           ABW
                                                                    Series.Name
1
                                                           Forest area (sq. km)
                                                  Forest area (% of land area)
3 Net official development assistance and official aid received (current US$)
                   Net official development assistance received (current US$)
4
5
                                                   Net ODA received (% of GNI)
6
                                     Net ODA received per capita (current US$)
                                    X1992
                                                 X1993
                                                                            X1995
         X1990
                      X1991
                                                               X1994
1 4.000000e+00 4.000000e+00 4.000000e+00 4.000000e+00 4.000000e+00 4.000000e+00
2 2.22222e+00 2.22222e+00 2.22222e+00 2.22222e+00 2.22222e+00 2.22222e+00
3 2.998000e+07 2.497000e+07 2.998000e+07 2.486000e+07 1.826000e+07 2.580000e+07
4 2.998000e+07 2.497000e+07 2.998000e+07 2.486000e+07 1.826000e+07 2.580000e+07
5 3.466232e+00 2.624562e+00 2.917643e+00 2.219643e+00 1.483004e+00
6 4.741195e+02 3.810817e+02 4.350223e+02 3.408328e+02 2.374049e+02 3.210912e+02
         X1996
                      X1997
                                    X1998
                                                  X1999
                                                                X2000
1 4.000000e+00 4.000000e+00 4.000000e+00
                                          4.000000e+00 4.000000e+00
2 2.22222e+00 2.22222e+00 2.222222e+00 2.22222e+00 2.22222e+00
3 1.953000e+07 2.507000e+07 1.126000e+07 -7.380000e+06 1.150000e+07
4 1.953000e+07 2.507000e+07 1.126000e+07 -7.380000e+06
            NA
                         NA
                                       NA
                                                                   NA
5
                                                      NΑ
6 2.350718e+02 2.940866e+02 1.293688e+02 -8.313900e+01
                                                                   NA
                       X2002
                                     X2003
                                                   X2004
          X2001
                                                             X2005
                                                                      X2006
  4.000000e+00 4.000000e+00 4.000000e+00
                                            4.000000e+00 4.000000 4.000000
  2.22222e+00 2.22222e+00 2.22222e+00
                                            2.22222e+00 2.222222 2.222222
3 -1.700000e+06 1.049000e+07 7.622000e+07 -1.132000e+07
                                                                NA
                                                                         NA
4
             NA
                           NA
                                        NA
                                                                NA
                                                                         NA
5
             NA
                          NΑ
                                        NA
                                                       NA
                                                                NA
                                                                         NA
6
             NA
                           NA
                                        NA
                                                       NA
                                                                NA
                                                                         NA
     X2007 X2008 X2009
1 4.000000
              NA
2 2.22222
              NA
                    NA
```

Note the file structure. The gender data is grouped by variable, with a row for each country-variable combination, and the columns holding all the yearly data. The MDI data is slightly different. It is grouped

3

4

5

6

NA

by country, with all variables for a single country in adjacent rows, and the yearly data again in columns. We could sort the data frames if we wanted them to display differently, but the layout will not matter for the select operations we will undertake next.

Important: R will assume that a matrix with a blank upper left cell (the first row has one fewer element than the rest of the matrix) is data with headers, and will use the first row as variable names. Most well-formatted data files will fit this convention, but if not, you may have to do more work at the import stage to specify variable names and the correct settings for import. Like other R functions, the read family of functions accepts many arguments.

Let's also inspect the objects to make sure they are what we expect them to be.

```
> mode(genderstats)
[1] "list"
> class(genderstats)
[1] "data.frame"
> mode(MDstats)
[1] "list"
> class(MDstats)
[1] "data.frame"
```

Selecting Rows/Observations

Our researcher is only interested in the five most populous countries. We have seen basic techniques like row and column indexing. Here's an example of the creation of a quick subset of the first 10 rows and columns of the MDI data, using simple numeric indexing of the matrix:

```
1
                                   AG.LND.FRST.K2
2
             ABW
                                   AG.LND.FRST.ZS
                         Aruba
3
             ABW
                         Aruba
                                   DT.ODA.ALLD.CD
4
             ABW
                         Aruba
                                   DT.ODA.ODAT.CD
5
             ABW
                         Aruba DT.ODA.ODAT.GN.ZS
6
             ABW
                         Aruba DT.ODA.ODAT.PC.ZS
7
             ABW
                         Aruba
                                   EN.ATM.CO2E.KT
8
                         Aruba
                                   EN.ATM.CO2E.PC
             ABW
9
                         Aruba ER.LND.PTLD.TR.ZS
             ABW
10
             ABW
                         Aruba
                                   ER.MRN.PTMR.ZS
```

```
Series.Name
1
                                                            Forest area (sq. km)
2
                                                   Forest area (% of land area)
3
   Net official development assistance and official aid received (current US$)
4
                    Net official development assistance received (current US$)
5
                                                    Net ODA received (% of GNI)
6
                                      Net ODA received per capita (current US$)
7
                                                              CO2 emissions (kt)
8
                                         CO2 emissions (metric tons per capita)
                         Terrestrial protected areas (% of total surface area)
9
```

```
10
                              Marine protected areas (% of total surface area)
          X1990
                       X1991
                                    X1992
                                                  X1993
                                                               X1994
  4.000000e+00 4.000000e+00 4.000000e+00 4.000000e+00 4.000000e+00
  2.22222e+00 2.22222e+00 2.22222e+00 2.22222e+00 2.22222e+00
3
  2.998000e+07 2.497000e+07 2.998000e+07 2.486000e+07 1.826000e+07
  2.998000e+07 2.497000e+07 2.998000e+07 2.486000e+07 1.826000e+07
4
  3.466232e+00 2.624562e+00 2.917643e+00 2.219643e+00 1.483004e+00
  4.741195e+02 3.810817e+02 4.350223e+02 3.408328e+02 2.374049e+02
7
  1.839328e+03 1.927264e+03 1.722080e+03 1.769712e+03 1.780704e+03
  2.908810e+01 2.941310e+01 2.498810e+01 2.426290e+01 2.315158e+01
8
9
             NA
                          NA
                                        NA
                                                     NA
                                                                  NA
             NA
                          NA
                                        NA
                                                     NA
                                                                  NA
10
          X1995
  4.000000e+00
2
  2.22222e+00
3
   2.580000e+07
4
  2.580000e+07
5
             NA
  3.210912e+02
6
7
  1.802688e+03
8
  2.243517e+01
9
             NA
             NA
10
```

We could pore over the original data and mark down the exact rows and columns we need, but that would be rather tedious. Instead we will use logical operators to select only those rows where the Country name matches those of interest. To do this, we will use the subset function. Also, while in practice we might peek at a data file an editor or spreadsheet program, we will examine the country names within R itself.

First we examine the variable names by studying the column labels.

> names(genderstats)

[1]	"Country_code"	"Country_Name"	"Indicator_code"	"Indicator_name"
[5]	"X1960"	"X1961"	"X1962"	"X1963"
[9]	"X1964"	"X1965"	"X1966"	"X1967"
[13]	"X1968"	"X1969"	"X1970"	"X1971"
[17]	"X1972"	"X1973"	"X1974"	"X1975"
[21]	"X1976"	"X1977"	"X1978"	"X1979"
[25]	"X1980"	"X1981"	"X1982"	"X1983"
[29]	"X1984"	"X1985"	"X1986"	"X1987"
[33]	"X1988"	"X1989"	"X1990"	"X1991"
[37]	"X1992"	"X1993"	"X1994"	"X1995"
[41]	"X1996"	"X1997"	"X1998"	"X1999"
[45]	"X2000"	"X2001"	"X2002"	"X2003"
[49]	"X2004"	"X2005"	"X2006"	"X2007"
[53]	"X2008"	"X2009"	"X2010"	"X2011"
[57]	"X2012"	"X2013"	"X2014"	"X2015"
[61]	"X2016"	"X2017"	"X2018"	"X2019"
[65]	"X2020"	"X2021"	"X2022"	"X2023"
[69]	"X2024"	"X2025"	"X2026"	"X2027"
[73]	"X2028"	"X2029"	"X2030"	"X2031"
[77]	"X2032"	"X2033"	"X2034"	"X2035"
[81]	"X2036"			

> names(MDstats)

```
[1] "Country.Code" "Country.Name" "Series.Code"
                                                                       "X1990"
                                                       "Series.Name"
                                      "X1993"
[6] "X1991"
                      "X1992"
                                                       "X1994"
                                                                       "X1995"
[11] "X1996"
                      "X1997"
                                      "X1998"
                                                       "X1999"
                                                                       "X2000"
[16] "X2001"
                      "X2002"
                                      "X2003"
                                                       "X2004"
                                                                       "X2005"
[21] "X2006"
                      "X2007"
                                      "X2008"
                                                       "X2009"
```

It looks like "Country_Name" and "Country.Name" are the columns we want to inspect. We'll look at the contents of those too, just to make sure we get the country names in their proper format. A dataset might abbreviate or alter names in ways we might not anticipate. We will use the \$ convention to refer to a variable within a data frame. Note that this table is also effectively gives us a count of the variables for which data is present for each country. To generate a simple listing, use levels.

```
> table(genderstats$Country_Name)
> table(MDstats$Country.Name)
> levels(genderstats$Country_Name)
> levels(MDstats$Country.Name)
```

In fact, there are no surprises in the names, so we can proceed to subset our data with them. We will use the logical operator == to represent equality. The double equal distinguishes it from the use of = as an assignment in R, and is a persistent source of typos in code! Other logical operators are and (&), or (|), not(!), and the usual >,<,>=,<=, and not equal to (!=). Important: the special operator is.na() is used to test for missing values, and is useful in many situations.

We'll assign our data subset to a new object, then inspect it to make sure everything ran smoothly. This is a fairly simple example, but we could mix and match selection by different criteria to our heart's content, as long as keep our Boolean logic straight.

Notice that since all of the levels are inherited from the parent object, the names of all other countries are still present, but with 0 data elements. In general, this won't cause any harm.

We can also compare the number of observations by running the following commands.

```
> row.names(genderstats)
> row.names(gscountry)
> row.names(MDstats)
> row.names(MDcountry)
```

Selecting Columns/Variables

Now we will further refine our subsets by selecting a few variables of interest. We could look over the variable names provided earlier, or examine documentation on the data to help make our choices. From Millenium Development Indicators, we will select three variables related to communications technology: Mobile Cellular Subscribers, Telephone Lines, and Internet Users. We'll choose a time period for which data is fairly complete, 2000-2008. To further simplify the file, we can drop everything the country code and variable codes.

We'll make use of subset as before, but now we add the select option to screen for the columns we want, and deposit the result in new data frames. It is also possible to use grep style pattern matching, but that takes us beyond the scope of this introduction. Note the use of the colon to indicate a range of columns in the select statement. And remember the double equals sign!

 $^{^{1}\}mathrm{The}$ voluminous output is suppressed in this document.

```
Series.Name == "Telephone lines (per 100 people)" | Series.Name ==
      "Internet users (per 100 people)", select = c(Country.Name,
      Series.Name, X2000:X2008))
> myMDI
                                                      Series.Name
                                                                       X2000
       Country.Name
1957
            Brazil Mobile cellular subscriptions (per 100 people) 13.3131882
1958
            Brazil
                                 Telephone lines (per 100 people) 17.7559180
                                  Internet users (per 100 people)
1959
            Brazil
                                                                  2.8706852
2683
             China Mobile cellular subscriptions (per 100 people) 6.7524918
2684
             China
                                 Telephone lines (per 100 people) 11.4702859
                                  Internet users (per 100 people)
2685
             China
                                                                   1.7819736
         Indonesia Mobile cellular subscriptions (per 100 people)
6631
                                                                   1.7874718
                                 Telephone lines (per 100 people)
6632
         Indonesia
                                                                   3.2456139
6633
          Indonesia
                                  Internet users (per 100 people)
                                                                   0.9255639
              India Mobile cellular subscriptions (per 100 people)
6733
                                                                  0.3521030
6734
              India
                                 Telephone lines (per 100 people)
                                                                   3.1927748
6735
              India
                                  Internet users (per 100 people) 0.5413796
15721 United States Mobile cellular subscriptions (per 100 people) 38.7983329
                                 Telephone lines (per 100 people) 68.2254058
15722 United States
                                  Internet users (per 100 people) 43.9448280
15723 United States
                              X2003
                                        X2004
                                                  X2005
                                                            X2006
          X2001
                    X2002
                                                                      X2007
     16.2718829 19.473151 25.544750 35.681357 46.331052 53.103451 63.633551
1958
     21.1881369 21.667014 21.596106 21.526240 21.417535 20.621026 20.723556
1959
      4.5284949 9.149425 13.207587 19.073673 21.022747 28.178380 30.884179
2683 11.3865629 16.089112 20.952576 25.833690 30.175652 35.167883 41.529116
2684 14.1815466 16.730865 20.393279 24.053855 26.880388 28.053423 27.744226
      2.6496835 4.615745 6.170444 7.252667 8.579043 10.601042 16.130450
2685
6631
      3.1341129 5.548753 8.656575 14.015965 21.399530 28.746095 41.566319
6632
      3.4695829 3.675473 3.771556 4.794043 6.162042 6.677399 8.692546
6633
      2.0186139 2.134136 2.387020 2.600286 3.602025 4.764813 5.786275
6733
      0.6334303 1.239700
                           3.165168 4.836434 8.235100 14.962005 20.770155
6734
      3.7324145 3.949875 3.945890 4.278699 4.584075
                                                        3.673598 3.504041
      0.6779836 1.581094 1.736192 2.037563 2.466693
                                                         2.901395 4.089663
15721 45.0747504 49.269642 55.329868 63.068258 72.066342 81.042220 87.291314
15722 67.1984903 65.756607 63.009610 60.635772 59.263889 56.126230 52.586130
15723 50.0989217 60.052769 63.100013 66.255455 69.619096 70.625502 73.591567
         X2008
1957 78.470712
1958
     21.430993
1959 37.519993
2683 48.407322
2684 25.728207
2685 22.496424
6631 61.834740
6632 13.362098
6633
      7.917479
6733 30.429882
6734
      3.324664
6735
      4.539613
15721 88.962705
15722 50.863266
15723 75.850161
```

> myMDI <- subset(MDcountry, Series.Name == "Mobile cellular subscriptions (per 100 people)" |

Let's move on to the Gender data. We'll choose a few more variables here, relating to fertility and education. The technique is identical to the first statement, although this one is a bit long due to all of the variables chosen.

```
> mygender <- subset(gscountry, Indicator_name == "GNI per capita, Atlas method (current US$)" |
      Indicator_name == "Expected years of schooling, female" |
      Indicator_name == "Expected years of schooling, male" | Indicator_name ==
      "Labor force, female (% of total labor force)" | Indicator_name ==
      "Adolescent fertility rate (births per 1,000 women ages 15-19)" |
      Indicator_name == "Fertility rate, total (births per woman)",
      select = c(Country_Name, Indicator_name, X2000:X2008))
> mygender
       Country_Name
377
             Brazil
387
              China
435
          Indonesia
437
              India
      United States
550
             Brazil
4125
4134
              China
4181
          Indonesia
4182
              India
4286
     United States
4323
             Brazil
4330
              China
4374
          Indonesia
4375
              India
4471
     United States
13362
             Brazil
13371
              China
13418
          Indonesia
13419
              India
13529 United States
             Brazil
15407
15417
              China
15466
          Indonesia
15467
              India
15578 United States
17455
             Brazil
17465
              China
17516
          Indonesia
17518
              India
17637 United States
                                                       Indicator_name
                         GNI per capita, Atlas method (current US$)
377
387
                         GNI per capita, Atlas method (current US$)
435
                         GNI per capita, Atlas method (current US$)
                         GNI per capita, Atlas method (current US$)
437
550
                         GNI per capita, Atlas method (current US$)
                                 Expected years of schooling, female
4125
4134
                                 Expected years of schooling, female
                                 Expected years of schooling, female
4181
                                 Expected years of schooling, female
4182
                                 Expected years of schooling, female
4286
```

```
4323
                                    Expected years of schooling, male
4330
                                    Expected years of schooling, male
                                    Expected years of schooling, male
4374
                                    Expected years of schooling, male
4375
4471
                                    Expected years of schooling, male
                        Labor force, female (% of total labor force)
13362
                        Labor force, female (% of total labor force)
13371
                        Labor force, female (% of total labor force)
13418
13419
                        Labor force, female (% of total labor force)
                        Labor force, female (% of total labor force)
13529
15407 Adolescent fertility rate (births per 1,000 women ages 15-19)
15417 Adolescent fertility rate (births per 1,000 women ages 15-19)
15466 Adolescent fertility rate (births per 1,000 women ages 15-19)
15467 Adolescent fertility rate (births per 1,000 women ages 15-19)
15578 Adolescent fertility rate (births per 1,000 women ages 15-19)
17455
                            Fertility rate, total (births per woman)
                            Fertility rate, total (births per woman)
17465
17516
                            Fertility rate, total (births per woman)
17518
                            Fertility rate, total (births per woman)
17637
                            Fertility rate, total (births per woman)
             X2000
                           X2001
                                         X2002
                                                       X2003
                                                                     X2004
377
       3870.000000
                     3310.000000
                                   3070.000000
                                                2970.000000
                                                              3330.000000
387
                                   1100.000000
                                                1270.000000
                                                              1500.000000
        930.000000
                     1000.000000
435
        580.000000
                      690.000000
                                    740.000000
                                                 840.000000
                                                              1020.000000
                                                               630.000000
437
        450.000000
                      460.000000
                                    470.000000
                                                  530.000000
550
      34410.000000
                    34830.000000
                                  35250.000000 37530.000000 41180.000000
4125
         14.815620
                       14.730028
                                     14.893334
                                                   14.315871
                                                                14.541594
4134
                 NA
                              ΝA
                                            NA
                                                   10.288279
                                                                        NA
4181
                 NA
                       10.958419
                                     11.206036
                                                                11.688224
                                                   11.478033
4182
          7.299020
                        7.342879
                                      7.599022
                                                    8.540262
                                                                 9.056418
4286
         15.674375
                       15.905855
                                     16.001070
                                                   16.224592
                                                                16.241715
4323
         14.195686
                       14.109089
                                     14.237610
                                                   13.597271
                                                                13.982430
4330
                 NA
                                                   10.411777
                                                                        NA
                              NA
4374
                       11.382223
                                     11.509705
                                                                12.028186
                 NA
                                                   11.840918
4375
          9.385997
                        9.352296
                                      9.453565
                                                    9.714072
                                                                10.271166
         14.768506
                                     14.886756
                                                                14.981555
4471
                       14.861889
                                                   14.990400
13362
         41.166286
                       41.304657
                                     41.960313
                                                   42.215680
                                                                42.546136
13371
         44.686145
                       44.679754
                                     44.664324
                                                   44.703913
                                                                44.675857
13418
         37.578699
                       37.403781
                                     37.226701
                                                   37.294869
                                                                37.287420
                       27.197584
                                                   27.038110
         27.299683
13419
                                     27.141606
                                                                26.980078
13529
         45.756985
                       45.854620
                                     45.808473
                                                   45.989230
                                                                45.866592
15407
         87.850000
                       87.110000
                                     86.370000
                                                   84.960000
                                                                82.880000
15417
          9.945500
                        9.888500
                                      9.831500
                                                    9.796300
                                                                 9.782900
15466
         49.878000
                       48.898800
                                     47.919600
                                                   46.669900
                                                                45.149700
15467
         87.893000
                       84.483000
                                     81.073000
                                                   78.245300
                                                                75.999900
         47.643000
                       45.819000
                                     43.995000
                                                   42.362900
15578
                                                                40.922700
17455
          2.364000
                        2.320000
                                      2.267000
                                                    2.208000
                                                                 2.143000
17465
          1.767000
                        1.762000
                                      1.759000
                                                    1.758000
                                                                 1.759000
17516
          2.453000
                        2.420000
                                      2.387000
                                                    2.354000
                                                                 2.319000
17518
          3.280000
                        3.210000
                                      3.140000
                                                    3.070000
                                                                 3.000000
                                                                 2.045000
17637
          2.056000
                        2.034000
                                      2.013000
                                                          NA
             X2005
                           X2006
                                         X2007
                                                      X2008
377
       3970.000000
                     4820.000000
                                   6060.000000
                                                7300.00000
                                                2940.00000
387
       1740.000000
                     2010.000000
                                   2410.000000
```

435	1170.000000	1300.000000	1520.000000	1880.00000
437	740.000000	820.000000	950.000000	1040.00000
550	43870.000000	45370.000000	46400.000000	47930.00000
4125	14.531284	NA	14.099942	NA
4134	NA	11.047391	11.352845	NA
4181	11.578087	NA	12.556157	NA
4182	9.320963	9.422804	9.755701	NA
4286	16.379382	16.374299	16.491325	NA
4323	13.858291	NA	13.452485	NA
4330	NA	10.781422	10.991157	NA
4374	12.075792	NA	12.839495	NA
4375	10.508036	10.627470	10.811553	NA
4471	14.954674	14.934308	15.021042	NA
13362	42.796536	43.114035	43.144333	43.49579
13371	44.715024	44.691393	44.730558	44.64665
13418	37.407735	37.728927	38.030060	38.40440
13419	27.162848	27.344949	27.526477	27.79151
13529	45.924360	45.841363	45.881637	46.09322
15407	80.800000	78.720000	76.640000	75.07000
15417	9.769500	9.756100	9.742700	9.74330
15466	43.629500	42.109300	40.589100	39.16770
15467	73.754500	71.509100	69.263700	67.11510
15578	39.482500	38.042300	36.602100	34.95750
17455	2.075000	2.007000	1.941000	1.88000
17465	1.759000	1.761000	1.762000	1.76500
17516	2.283000	2.245000	2.206000	2.16900
17518	2.930000	2.863000	2.798000	2.73800
17637	2.054000	2.100000	2.113200	2.10000

Merging Data

Now we'd like to combine the two datasets. To do this is quite simple. There are two functions, rbind (for row bind) and cbind (for column bind), that allow you to quickly paste together data objects. R is generally pretty good about matching observations and variables. Since we have similar variables in the columns, we simply want to add together our observations. We can try

```
mydata<-rbind(myMDI,mygender)</pre>
```

Oops! Our names do not match. Let's change them so that they do.

```
> names(mygender) <- c("Country.Name", "Series.Name", "X2000",
      "X2001", "X2002", "X2003", "X2004", "X2005", "X2006", "X2007",
      "X2008")
> names(mygender)
 [1] "Country.Name" "Series.Name"
                                    "X2000"
                                                    "X2001"
                                                                    "X2002"
 [6] "X2003"
                    "X2004"
                                    "X2005"
                                                    "X2006"
                                                                    "X2007"
[11] "X2008"
> mydata <- rbind(myMDI, mygender)</pre>
> head(mydata)
     Country.Name
                                                       Series.Name
                                                                       X2000
1957
           Brazil Mobile cellular subscriptions (per 100 people) 13.313188
1958
                                 Telephone lines (per 100 people) 17.755918
           Brazil
```

```
1959
           Brazil
                                 Internet users (per 100 people)
                                                                   2.870685
            China Mobile cellular subscriptions (per 100 people)
2683
                                                                   6.752492
2684
            China
                                Telephone lines (per 100 people) 11.470286
            China
2685
                                 Internet users (per 100 people)
                                                                   1.781974
                             X2003
         X2001
                   X2002
                                       X2004
                                                  X2005
                                                           X2006
                                                                    X2007
1957 16.271883 19.473151 25.544750 35.681357 46.331052 53.10345 63.63355
1958 21.188137 21.667014 21.596106 21.526240 21.417535 20.62103 20.72356
     4.528495 9.149425 13.207587 19.073673 21.022747 28.17838 30.88418
2683 11.386563 16.089112 20.952576 25.833690 30.175652 35.16788 41.52912
2684 14.181547 16.730865 20.393279 24.053855 26.880388 28.05342 27.74423
     2.649684
                4.615745 6.170444 7.252667 8.579043 10.60104 16.13045
        X2008
1957 78.47071
1958 21.43099
1959 37.51999
2683 48.40732
2684 25.72821
2685 22.49642
```

Using reshape and merge

It was a bit of a pain to have to reinput all of the variable names. Fortunately there is a package with several features to make data handling easier. That is Hadley Wickham's reshape. Let's load the package and try it out. Using reshape, we only have to indicate the changed variables.

```
> library(reshape)
> myChanges <- c(Country_Name = "Country.Name", Indicator_name = "Series.Name")
> mygender <- rename(mygender, myChanges)
> names(mygender)

[1] "Country.Name" "Series.Name" "X2000" "X2001" "X2002"
[6] "X2003" "X2004" "X2005" "X2006" "X2007"
[11] "X2008"
```

There is also another way to combine data frames, using merge². The merge function does more robust checking to make sure that data frames align, so it is preferable to rbind or cbind in more complex situations. The all=TRUE option is necessary to include all observations from each data frame. Without this, merge would select only observations whose variable values matched between the two data frames (an empty set in this case). Here we reverse the order of the combination so that we can see the change in the output [not displayed in the text].

```
> mydata <- merge(mygender, myMDI, all = TRUE)
> mydata
```

Using split

Another useful technique is to split the data. Any grouping variable can be used to separate a data frame into separate subframes. So, if we wanted to be able to easily access all of the data for one country, we could do the following:

```
> mysplit <- split(mydata, mydata$Country.Name, drop = TRUE)
> mysplit$Brazil
```

²You don't need reshape for this.

```
Country.Name
                                                                     Series.Name
        Brazil Adolescent fertility rate (births per 1,000 women ages 15-19)
1
                                           Expected years of schooling, female
2
        Brazil
3
        Brazil
                                             Expected years of schooling, male
4
        Brazil
                                      Fertility rate, total (births per woman)
5
        Brazil
                                    GNI per capita, Atlas method (current US$)
6
                                 Labor force, female (% of total labor force)
        Brazil
                                               Internet users (per 100 people)
7
        Brazil
                               Mobile cellular subscriptions (per 100 people)
8
        Brazil
9
                                              Telephone lines (per 100 people)
        Brazil
        X2000
                     X2001
                                 X2002
                                             X2003
                                                         X2004
                                                                    X2005
                 87.110000
                             86.370000
    87.850000
                                          84.96000
                                                      82.88000
                                                                 80.80000
1
2
    14.815620
                 14.730028
                             14.893334
                                          14.31587
                                                      14.54159
                                                                 14.53128
3
    14.195686
                 14.109089
                             14.237610
                                          13.59727
                                                      13.98243
                                                                 13.85829
4
                 2.320000
                              2.267000
     2.364000
                                           2.20800
                                                       2.14300
                                                                  2.07500
5
 3870.000000 3310.000000 3070.000000 2970.00000 3330.00000 3970.00000
6
    41.166286
                 41.304657
                             41.960313
                                          42.21568
                                                      42.54614
                                                                 42.79654
7
     2.870685
                 4.528495
                              9.149425
                                          13.20759
                                                      19.07367
                                                                 21.02275
                                          25.54475
8
    13.313188
                 16.271883
                             19.473151
                                                     35.68136
                                                                 46.33105
9
    17.755918
                 21.188137
                             21.667014
                                          21.59611
                                                     21.52624
                                                                 21.41754
       X2006
                  X2007
                              X2008
    78.72000
                76.64000
                           75.07000
1
2
                14.09994
          NA
                                 NA
3
          NA
                13.45249
                                 NA
4
     2.00700
                 1.94100
                            1.88000
5 4820.00000 6060.00000 7300.00000
6
    43.11403
               43.14433
                           43.49579
7
               30.88418
    28.17838
                           37.51999
8
    53.10345
               63.63355
                           78.47071
    20.62103
                20.72356
                           21.43099
```

The arguments to split are the data frame, the grouping variable or variables, and options. In this case, we dropped all of the countries for which there are no data from the mysplit data frame using the drop=TRUE argument.

Relabeling

We'd like to simplify some of the long names that were brought in with the data. Note that R does not require that variable names by unique, but it would be foolish and confusing to make use of this possiblity. This section is a bit of a hack. There are probably some better tools to do this, but we accomplish the relabeling by first converting the Series.Name from factor to character (otherwise names not in the previous list of levels will be rejected). Then we recursively search for the value we want to change, and use a created index vector, called **changes**, to overwite my changes onto Series.Name.

```
> mydata$Series.Name <- as.character(mydata$Series.Name)
> changes <- grep("Adolescent", mydata$Series.Name)
> mydata$Series.Name[changes] = "adfert"
> changes <- grep("schooling, female", mydata$Series.Name)
> mydata$Series.Name[changes] = "Fschool"
> changes <- grep("schooling, male", mydata$Series.Name)
> mydata$Series.Name[changes] = "Mschool"
> changes <- grep("Fertility", mydata$Series.Name)
> mydata$Series.Name[changes] = "totfert"
> changes <- grep("GNI", mydata$Series.Name)
> mydata$Series.Name[changes] = "GNI"
```

```
> changes <- grep("Labor", mydata$Series.Name)</pre>
> mydata$Series.Name[changes] = "Flabor"
> changes <- grep("Internet", mydata$Series.Name)</pre>
> mydata$Series.Name[changes] = "Internet"
> changes <- grep("Mobile", mydata$Series.Name)</pre>
> mydata$Series.Name[changes] = "Mobile"
> changes <- grep("Telephone", mydata$Series.Name)</pre>
> mydata$Series.Name[changes] = "Telephone"
> mydata$Series.Name
                                            "totfert"
 [1] "adfert"
                  "Fschool"
                               "Mschool"
                                                         "GNI"
                                                                       "Flabor"
 [7] "Internet"
                  "Mobile"
                               "Telephone"
                                            "adfert"
                                                         "Fschool"
                                                                       "Mschool"
[13] "totfert"
                  "GNI"
                               "Flabor"
                                            "Internet"
                                                         "Mobile"
                                                                       "Telephone"
[19] "adfert"
                               "Mschool"
                                                         "GNI"
                                                                       "Flabor"
                  "Fschool"
                                            "totfert"
[25] "Internet"
                  "Mobile"
                               "Telephone" "adfert"
                                                                       "Mschool"
                                                         "Fschool"
[31] "totfert"
                  "GNI"
                               "Flabor"
                                            "Internet"
                                                         "Mobile"
                                                                       "Telephone"
                                                                       "Flabor"
                  "Fschool"
                                                          "GNI"
[37] "adfert"
                               "Mschool"
                                             "totfert"
[43] "Internet"
                  "Mobile"
                               "Telephone"
```

Our new series names are adfert for Adolescent Fertility, Fschool for Female Years of Schooling, Mschool for Male Years of Schooling, totfert for Total Fertility, GNI for GNI per capita, Flabor for Female Percentage of the Labor Force, and Internet, Mobile, and Telephone for the number of users of each technology.

Transposing your data from wide to long format with reshape

Our data is in "wide" format, with a separate column for each year and a single row for each country/variable combination. We may want it in "long" format for some purposes. That is, the data should have separate rows for each yearly observation of the country/variable combination. Again, reshape can help with this. First, we create a vector of our years using a little technique that spares us from writing them out, and is useful for long sequences. Then we call reshape. We need the sep argument to tell reshape that the time variables are written without the . separator that reshape assumes as a default.

melt and cast

There is a further refinement to reshape that lets you do other things to the data. Using reshape as we did above uses some default assumptions that usually work, but can be restrictive. If we melt the data, we create a long format data frame that also has stored information that allows us to cast the data into alternative output forms. With cast, we specify the data frame to be operated on, and then a function. On the left side of the ~ are the variables to appear in the rows, and on the right, the variables to appear in the columns. In the following example, we also use | to divide the results into separate matrices for each series. The best way to get a feel for how cast works is to play around with moving different variables to the left or right of the ~. Also, in the example below, we insert a small step to remove the "X" from the years, convert them to numeric variables, and rename the variable from "variable" to "year".

```
> mymelt <- melt(mydata)
> head(mymelt)
```

```
Country.Name Series.Name variable
                                         value
        Brazil
                    adfert
                              X2000
                                      87.85000
1
2
                                      14.81562
        Brazil
                   Fschool
                              X2000
3
                   Mschool
                              X2000
                                      14.19569
       Brazil
4
        Brazil
                   totfert
                              X2000
                                       2.36400
5
                              X2000 3870.00000
        Brazil
                       GNI
        Brazil
                    Flabor
                              X2000
                                      41.16629
> mymelt$variable <- as.numeric(sub("X", "", mymelt$variable))</pre>
> names(mymelt)[3] <- "year"</pre>
> mycast <- cast(mymelt, year ~ Country.Name | Series.Name)
> head(mycast)
$adfert
  year Brazil China
                       India Indonesia United States
1 2000 87.85 9.9455 87.8930
                               49.8780
                                              47.6430
2 2001 87.11 9.8885 84.4830
                               48.8988
                                              45.8190
3 2002
       86.37 9.8315 81.0730
                               47.9196
                                              43.9950
4 2003 84.96 9.7963 78.2453
                              46.6699
                                              42.3629
5 2004 82.88 9.7829 75.9999
                               45.1497
                                              40.9227
6 2005 80.80 9.7695 73.7545
                               43.6295
                                              39.4825
       78.72 9.7561 71.5091
7 2006
                               42.1093
                                              38.0423
8 2007
       76.64 9.7427 69.2637
                               40.5891
                                              36.6021
9 2008 75.07 9.7433 67.1151
                               39.1677
                                              34.9575
$Flabor
         Brazil
                   China
                            India Indonesia United States
1 2000 41.16629 44.68614 27.29968 37.57870
                                                  45.75699
2 2001 41.30466 44.67975 27.19758 37.40378
                                                  45.85462
3 2002 41.96031 44.66432 27.14161 37.22670
                                                  45.80847
4 2003 42.21568 44.70391 27.03811 37.29487
                                                  45.98923
5 2004 42.54614 44.67586 26.98008 37.28742
                                                  45.86659
6 2005 42.79654 44.71502 27.16285 37.40774
                                                  45.92436
7 2006 43.11403 44.69139 27.34495 37.72893
                                                  45.84136
8 2007 43.14433 44.73056 27.52648 38.03006
                                                  45.88164
9 2008 43.49579 44.64665 27.79151 38.40440
                                                  46.09322
$Fschool
  vear
       Brazil
                   China
                            India Indonesia United States
1 2000 14.81562
                      NA 7.299020
                                                  15.67437
                                         NA
                      NA 7.342879 10.95842
2 2001 14.73003
                                                  15.90585
3 2002 14.89333
                      NA 7.599022 11.20604
                                                  16.00107
4 2003 14.31587 10.28828 8.540262 11.47803
                                                  16.22459
5 2004 14.54159
                      NA 9.056418 11.68822
                                                  16.24171
6 2005 14.53128
                      NA 9.320963 11.57809
                                                  16.37938
7 2006
             NA 11.04739 9.422804
                                         NA
                                                  16.37430
8 2007 14.09994 11.35284 9.755701
                                                  16.49133
                                   12.55616
9 2008
             NA
                      NA
                               NA
                                         NA
                                                        NA
$GNI
  year Brazil China India Indonesia United States
1 2000
         3870
                930
                      450
                                580
                                             34410
2 2001
         3310 1000
                      460
                                690
                                            34830
3 2002
         3070 1100
                                740
                                            35250
                      470
```

4 2003

2970 1270

530

37530

840

```
5 2004
         3330 1500
                      630
                               1020
                                             41180
6 2005
         3970
               1740
                                             43870
                      740
                               1170
7 2006
                                             45370
         4820
               2010
                      820
                               1300
8 2007
               2410
         6060
                      950
                               1520
                                             46400
9 2008
         7300
               2940
                     1040
                               1880
                                             47930
$Internet
  year
          Brazil
                     China
                               India Indonesia United States
1 2000
       2.870685 1.781974 0.5413796 0.9255639
                                                     43.94483
2 2001
       4.528495 2.649684 0.6779836 2.0186139
                                                     50.09892
3 2002 9.149425
                 4.615745 1.5810944 2.1341357
                                                     60.05277
4 2003 13.207587
                  6.170444 1.7361917 2.3870198
                                                     63.10001
5 2004 19.073673 7.252667 2.0375630 2.6002859
                                                     66.25545
6 2005 21.022747 8.579043 2.4666928 3.6020248
                                                     69.61910
7 2006 28.178380 10.601042 2.9013945 4.7648131
                                                     70.62550
8 2007 30.884179 16.130450 4.0896632 5.7862747
                                                     73.59157
9 2008 37.519993 22.496424 4.5396133 7.9174794
                                                     75.85016
$Mobile
 year
        Brazil
                    China
                               India Indonesia United States
1 2000 13.31319 6.752492 0.3521030 1.787472
                                                     38.79833
2 2001 16.27188 11.386563
                           0.6334303 3.134113
                                                     45.07475
3 2002 19.47315 16.089112
                           1.2397001 5.548753
                                                     49.26964
4 2003 25.54475 20.952576
                           3.1651676 8.656575
                                                     55.32987
5 2004 35.68136 25.833690
                          4.8364337 14.015965
                                                     63.06826
6 2005 46.33105 30.175652 8.2350996 21.399530
                                                     72.06634
7 2006 53.10345 35.167883 14.9620051 28.746095
                                                     81.04222
8 2007 63.63355 41.529116 20.7701548 41.566319
                                                     87.29131
9 2008 78.47071 48.407322 30.4298820 61.834740
                                                     88.96270
```

A final check on the data

In addition to summary and table, which we saw earlier, str provides a useful and compact summary of the structure of a data object. Let's try it out on the World Bank extract.

> str(mydata)

```
'data.frame':
                    45 obs. of 11 variables:
 $ Country.Name: Factor w/ 232 levels "Afghanistan",..: 27 27 27 27 27 27 27 27 27 41 ...
 $ Series.Name : chr "adfert" "Fschool" "Mschool" "totfert" ...
 $ X2000
               : num 87.85 14.82 14.2 2.36 3870 ...
               : num 87.11 14.73 14.11 2.32 3310 ...
 $ X2001
 $ X2002
               : num
                      86.37 14.89 14.24 2.27 3070 ...
               : num 84.96 14.32 13.6 2.21 2970 ...
 $ X2003
 $ X2004
               : num 82.88 14.54 13.98 2.14 3330 ...
               : num 80.8 14.53 13.86 2.08 3970 ...
 $ X2005
               : num 78.72 NA NA 2.01 4820 ...
 $ X2006
 $ X2007
               : num 76.64 14.1 13.45 1.94 6060 ...
 $ X2008
               : num 75.07 NA NA 1.88 7300 ...
> str(mylongdata)
'data.frame':
                    405 obs. of 5 variables:
 $ Country.Name: Factor w/ 232 levels "Afghanistan",..: 27 27 27 27 27 27 27 27 27 41 ...
 $ Series.Name : chr "adfert" "Fschool" "Mschool" "totfert" ...
```

6 Exporting output and saving the workspace

We have accomplished our goal of creating a useful extract from the World Bank data. We'll export the mydata file in .csv and SPSS format, using the obverse of the read functions used at the beginning:

```
> write.csv(mydata, "mydata.csv")
> library(foreign)
> write.foreign(mydata, datafile = "mydata.sav", codefile = "mydata.sps",
+ package = "SPSS")
```

And finally, save the workspace as an RData file. This stores all of the data structures that we have created in this session.

```
> save.image("mydata.RData")
```

Now you can truly say that you can handle data in R!

References

Peter Dalgaard. Introductory Statistics with R. Statistics and Computing. Springer, 2nd edition, 2008.

Phil Spector. Data Manipulation with R. Use R! Springer, 2008.

Robert A. Muenchen. R for SAS and SPSS Users. Statistics and Computing. Springer, 2009.