Reading Foreign Files

by Duncan Murdoch

One of the first tasks in any statistical analysis is getting the data into a usable form. When your client gives you notebooks filled with observations or a pile of surveys and you control the data entry, then you can choose the form to be something convenient. In R, that usually means some form of text file, either comma- or tab-delimited. Through the **RODBC** and various SQL packages R can also import data directly from many databases.

However, sometimes you receive data that has already been entered into some other statistical package and stored in a proprietary format. In that case you have a couple of choices. You might be able to write programs in that package to export the data in an R-friendly format, but often you don't have access or familiarity with it, and it would be best if R could read the foreign file directly. The **foreign** package is designed for exactly this situation. It has code to read files from Minitab, S, S-PLUS, SAS, SPSS and Stata. In this article I'll describe how to import data from all of these packages. I wrote the S and S-PLUS parts, so I'll describe those in most detail first. Other credits for the **foreign** package go to Doug Bates, Saikat DebRoy, and Thomas Lumley.

For more information on data import and export, I recommend reading the fine manual "R Data Import/Export" which comes with the R distribution.

Reading S and S-PLUS files

Both S and S-PLUS usually store objects (which, as most R users know, are very similar to R objects) in individual files in a data directory, as opposed to the single workspace files that R uses. Sometimes the file has the same name as the object, sometimes it has a constructed name, e.g., '__12'. The mapping between object name and file name is complicated and version dependent. In S-PLUS 2000 for Windows, the file '__nonfi' in the data directory contains a list of object names and corresponding filenames. Other versions use different mechanisms. Sometimes (e.g., in library sections) all objects are stored in one big file; at present the **foreign** package cannot read these.

The **foreign** function `read.S()` can be used to read the individual binary files. I have tested it mainly on S-PLUS 2000 for Windows, but it is based on code that worked on earlier versions as well. It may not work at all on later versions. (This is a continuing problem with reading foreign files. The writers of those foreign applications frequently change the formats of their files, and in many cases, the file formats are unpublished. When the format changes, **foreign** stops working until someone works out the new format and modifies the code.)

`read.S()` takes just one argument: the filename to read from. It's up to you to figure out which file contains the data you want. Because S and R are so similar, `read.S` is surprisingly successful. It can generally read simple vectors, data frames, and can occasionally read functions. It almost always has trouble reading formulas, because those are stored differently in S and R.

For example, I have an S-PLUS data frame called `Bars`:

```
> Bars
    type time      y1      y2
  1    0    1 -0.5820 24.7820
  2    0    2  0.5441 23.6559
317    0    7 -1.1925 25.3925
319    0    8  1.4409 22.7591
631    0   13 -3.4194 27.6194
633    0   14  0.7975 23.4025
738    0   19 -0.3439 24.5439
740    0   20  0.6580 23.5420
```

At the beginning of the session, S-PLUS printed this header:

```
Working data will be in
F:\Program files\sp2000\users\murdoch\_Data
```

so that's the place to go looking for `Bars`: but there's no file there by that name. The '__nonfi' file has these lines near the end:

```
"Bars"
"__107"
```

which give me the correct filename, '__107'. Here I read the file into R:

```
> setwd('F:/Program files/sp2000/users')
NULL
> read.S('murdoch/_Data/__107')
    type time      y1      y2
1      0    1 -0.5820 24.7820
2      0    2  0.5441 23.6559
317    0    7 -1.1925 25.3925
319    0    8  1.4409 22.7591
631    0   13 -3.4194 27.6194
633    0   14  0.7975 23.4025
738    0   19 -0.3439 24.5439
740    0   20  0.6580 23.5420
```

(The only reason to use `setwd` here is to fit into these narrow columns!) We see that `read.S` was successful; only the printing format is different.

Of course, since I have access to S-PLUS to do this, it would have been much easier to export the data frame using `dump()` and read it into R using `source()`. I could also have exported it using `data.dump` and restored it using **foreign**'s `data.restore`, but that has the same limitations as `read.S`.

Reading SAS files

SAS stores its files in a number of different formats, depending on the platform (PC, Unix, etc.), version, etc. The `read.xport` function can read the SAS transport (XPORT) format. How you produce one of these files depends on which version of SAS you're using. In older versions, there was a `PROC XPORT`. Newer versions use a "library engine" called `sasv5xpt` to create them. From the SAS technical note referenced in the `?read.xport` help page, a line like

```
libname xxx sasv5xpt 'xxx.dat';
```

creates a library named `xxx` which lives in the physical file 'xxx.dat'. New datasets created in this library, e.g. with

```
data xxx.abc;
  set otherdata;
```

will be saved in the XPORT format.

To read them, use the `read.xport` function. For example, `read.xport('xxx.dat')` will return a list of data frames, one per dataset in the exported library. If there is only a single dataset, it will be returned directly, not in a list. The `lookup.xport` function gives information about the contents of the library.

Reading Minitab files

Minitab also stores its data files in several different formats. The **foreign** package can only read the "portable" format, the MTP files. In its current incarnation, only numeric data types are supported; in particular, dataset descriptions cannot be read.

The `read.mtp` function is used to read the files. It puts each column, matrix or constant into a separate component of a list.

Reading SPSS and Stata files

The **foreign** functions `read.spss` and `read.dta` can read the binary file formats of the SPSS and Stata packages respectively. The former reads data into a list, while the latter reads it into a data frame. The `write.dta` function is unique in the **foreign** package in being able to *export* data in the Stata binary format.

Caveats and conclusions

It's likely that all of the functions in the **foreign** package have limitations, but only some of them are documented. It's best to be very careful when transferring data from one package to another. If you can, use two different methods of transfer and compare the results; calculate summary statistics before and after the transfer; do anything you can to ensure that the data that arrives in R is the data that left the other package. If it's not, you'll be analyzing programming bugs instead of the data that you want to see.

But this is true of any data entry exercise: errors introduced in data entry aren't of much interest to your client!

Duncan Murdoch
University of Western Ontario
murdoch@stats.uwo.ca

Maximally Selected Rank Statistics in R

by Torsten Hothorn and Berthold Lausen

Introduction

The determination of two groups of observations with respect to a simple cutpoint of a predictor is a common problem in medical statistics. For example, the distinction of a low and high risk group of patients is of special interest. The selection of a cutpoint in the predictor leads to a multiple testing problem, cf. Figure 1. This has to be taken into account when the effect of the selected cutpoint is evaluated. Maximally selected rank statistics can be used for estimation as well as evaluation of a simple cutpoint model. We show how this problems can be treated with the **maxstat** package and illustrate the usage of the package by gene expression profiling data.

Maximally selected rank statistics

The functional relationship between a quantitative or ordered predictor X and a quantitative, ordered or censored response Y is unknown. As a simple model one can assume that an unknown cutpoint μ in X determines two groups of observations regarding the response Y: the first group with X-values less or equal μ and the second group with X-values greater μ. A measure of the difference between two groups with respect to μ is the absolute value of an appropriate standardized two-sample linear rank statistic of the responses. We give a short overview and follow the notation in Lausen and Schumacher (1992).

The hypothesis of independence of X and Y can be formulated as

$$H_0 : P(Y \leq y | X \leq \mu) = P(Y \leq y | X > \mu)$$

for all y and $\mu \in \mathbb{R}$. This hypothesis can be tested as