Introduction to R
UCLA/CCPR
February 2005

Exercises (Part 2)

1. Read data from various sources into data frames:

    - Directly from the keyboard.
    - Using the data editor `fix()`.
    - From a text file in which the data values are delimited by white space.
    - From an SPSS data set (in a `.sav` or `.por` file) or from some other source supported by the `foreign` package..

2. If you have access to MySQL, PostgreSQL, or another database-management system that can communicate via ODBC, attempt to access a database from the DBMS using the `RODBC` package and the functions in the file `Fox-odbc-functions.R` (on the course web site).

    Notes:

    - This is potentially a time-consuming exercise, but if you routinely work with very large data sets, you might find the investment worthwhile.
    - If you wish to install a DBMS on your Windows computer, I'd recommend MySQL (for its simplicity and ease of installation under Windows). MySQL is available at *www.mysql.com*, and is free.
    - In installing MySQL on my laptop, I could not get the current version of the MySQL ODBC driver for Windows to work properly with RODBC; a slightly older version worked fine, however. If you encounter this problem, I can give you the older driver. I understand that the RODBC package will be changed to work around a bug in the new MySQL driver.
    - For purposes of experimentation, you can easily save an R data frame as a table in your data base using the `sqlSave()` function in the `RODBC` package. For example, having read the `Prestige` data frame into memory, `sqlSave(channel, Prestige)` will save the data frame in the table named `prestige` in the data base accessed via `channel`. See `?sqlSave` for details.

3. Explore the properties of various kinds of objects:

    - Create a character vector, a numeric vector, a logical vector, a character matrix, a numeric matrix, a factor, a data frame, a list, and a function.
    - Apply each of the following functions to these objects: `length()`, `class()`, `mode()`, and `attributes()`.
    - Look at the help files for each of these functions – e.g., `?length`.
    - What did you learn?

4. R has a number of "coercion" functions, prefixed with `as.`, and a number of "predicate" functions, prefixed with `is.`: for example, `is.matrix` and `as.matrix`.

    - Get a complete list of these functions via the commands `apropos("^is\\.")` and `apropos("^is\\.")`. *Note*: The quoted arguments to `apropos()` are "regular expressions" – a powerful notation for searching text that will be familiar to Unix users; see `?regex` for how regular expressions are used in R.
    - Using the objects created in the previous exercise, experiment with (for example) the coercion functions `as.matrix` , `as.vector`, and `as.character`, and with the predicates `is.vector` and `is.character`. What did you learn?

5. A general alternative to attaching a data frame to the search path is to use the `with()` function, the first argument of which can be a data frame, and the second argument an expression to be evaluated within the "environment" of the data frame. For example, having read the `Prestige` data frame into memory, but not having attached it to the search path, `with(Prestige, mean(education))` calculates the mean of the variable `education` in the `Prestige` data frame.

    - Perform some computations using `with()`.
    - Look up the help file for `with` – `?with`.
    - What are the advantages and disadvantages of using `with()` instead of attaching a data frame to the search path?