John Miyamoto
Email: jmiyamot@u.washington.edu

February 16, 2006
http://faculty.washington.edu/jmiyamot

Getting Started with the R Statistical Programming Language

NOTE: R is a freeware statistical program. See the R homepage (http://www.r-project.org/) for the terms of its use. The homepage also has useful information about the history of the R-project, how to use R, how to write R programs, and much more.

Instructions for Downloading R.

These instructions assume that R version 2.2.1 is the current state-of-the-art version of R for Windows. If there is a more recent version, go ahead and use it.

1. Go to http://www.r-project.org/ or http://cran.us.r-project.org/welcome.html. Either should work.
2. Click on CRAN (Comprehensive R Archive Network). CRAN will ask which of various mirrored website you want to be connected to. Choose one close to where you are. (In Seattle, I usually choose UCLA or Wisconsin.)
3. Under "Precompiled Binary Distributions", click on the operating system for your computer, e.g., MacOS (System 8.6 to 9.1 and MacOS X) or Windows (95 and later). DO NOT download the R source code unless you are a computer expert who wants to work with the code.
4. For either Macos or Windows, there are two components to the program. Base refers to the main program. Contrib refers to special sets of functions (R calls them "packages") that have been written by R-users for special purposes. Everyone needs to have the Base program, but you only need packages if you have a need for its specific purpose. E.g., there is a repeated measures package called nlme that has multilevel modeling functions.
5. For Windows users, download the R-2.2.1 program file. This is all you need to install the R base program.
   * If you are short of hard drive space, you can download the files miniR.exe and miniR-1.bin to miniR-6.bin. If you need to, you can put miniR.exe and miniR-1.bin on one floppy, and miniR-2.bin to miniR-6.bin on separate floppies. This is a small installation, containing text and compiled HTML help files, and the Introduction to R and Data Import/Export manuals in PDF.
   * The file ReadMe.R Version# contains installation instructions if you require a more customized installation.
6. For Windows users, go back to the window with the Contrib option (to start from the R-Homepage, click on CRAN; click on R-Binaries; click on Windows; click on Contrib). Now you can click on the names of any packages that you want to download. Over time, you will hear about various packages that you may need to use in your work. It is easy to use this procedure to download zip files that contain various packages. Put the package zip files in a temporary directory. I recommend that you download at least the 'foreign' package; it has functions for reading data from SPSS or SAS files (among others).
7. It may be convenient to download the R Manuals (see instructions below) while you are still connected to CRAN.
8. It may be useful to look at the FAQ for Windows and Mac users. Go to http://cran.r-project.org/, and click on FAQs. Choose the R Windows FAQ or the R MacOS X FAQ.

I'm afraid that I don't use a Mac, Unix or Linux systems, so I don't have any tips about what to download for these operating systems. You can probably get help with a Mac, Unix or Linux installation at CSSCR.

Installing R on a Windows System:

The files README and RW-FAQ documents contain instructions for installing R.

- To install R, simply apply the Add/Remove programs utility (available on the Control Panel) to R installation file (named something like 'R-2.2.1-win32.exe'.
- After the program has been installed, you will want to put an icon (short-cut) to the Rgui.exe program file on your desktop. Rgui.exe is contained in the ...\R Version#\bin directory. Rgui.exe runs the interactive version of R. Right mouse click on the Rgui.exe, create a shortcut to R, and then copy or cut and paste this shortcut to the Windows Desktop (or any other place you might want to keep the icon).
- You may also want to install some R packages. First, download the zip file for the package from CRAN (see step 6 above). *To Install a R Package:* When you unzip the zip file for the package, make ...\R Version#\library the target directory (where unzip is instructed to place the files). The unzip program will automatically create directories under ...\R Version#\library for the individual packages. You can delete the zip files after you have installed the packages. When you run R, you need to give the command, **`library(package.name)`**, where "package.name" is the name for the particular package. For example, to load the 'foreign' package, you must give the R command, **`library(foreign)`**. Below is a list of some of the many useful R packages.
  * The foreign package (foreign.zip) contains a function, read.spss, that allows R to read SPSS data files.
  * The MASS package (vr.zip) contains many useful functions and data sets for Venables & Ripley's excellent textbooks about S computing (also R). This package is automatically installed when you install the base package (the main R program), so you don't need to download it or install it individually.
  * The NLME package (nlme.zip) contains functions and data for repeated measures anova and multilevel modeling. This package is automatically installed when you install the base package (the main R program), so you don't need to download it or install it individually.
  * Note: It is very easy to update R packages, i.e., get the latest versions of packages that you have previously installed on your computer. To update your R packages, do the following: (i) connect to the internet; (ii) run R; (iii) give the command, update.packages(), to R. The update.packages function will automatically update every package that you have installed on your computer.

Setting the Startup Directory for R:

The startup director is the directory that contains the data objects that you want to work on in your current R session. For example, suppose you are working on two projects; the data files for your dissertation project are contained in a directory (folder) called c:\diss and the data for your job are contained in a directory called c:\job. When you want to use R to analyze your dissertation data, you want to start R with c:\diss as the startup directory; and when you want to use R to analyze data from your job, you want to start R with c:\job as the startup directory.

- To set the startup directory, click the right mouse button on the R program icon. Choose "properties". Click on the Shortcut tab, and type the path and name of the directory that contains the files that you want to work on. You can choose any directory as the startup directory.
- *Example:* Suppose as suggested above that you are working on two projects, a dissertation project with files in c:\diss, and a job project with files in c:\job. A standard way to deal with this is to make two copies (or shortcuts) of the R program icon. (Right mouse click on any existing R program icon or shortcut; select Create Shortcut; do this twice if you want two shortcuts to R.) Now edit the properties of these icons, setting one so that it starts in c:\diss\data and the other so that it starts in c:\job\data. Click on the appropriate icon when working on the corresponding project. In this way, R objects that are created for purposes

of analyzing the dissertation data (in c:\diss\data) will not get confused with R objects that are created for purposes of analyzing job data (in c:\job\data).

Suppose you want to work on the data in c:\diss\data, and you have created an R shortcut (icon) that starts in this directory. Double click on the R icon that starts in c:\diss\data. As it starts up, R automatically looks for a file called **.Rdata** in c:\diss\data. This file, **.Rdata**, contains any functions or data objects that were created and saved in previous R-sessions that started in c:\diss\data. If there is no **.Rdata** file in c:\diss\data, e.g., because this is the first time you have started R in this directory, then R automatically creates a **.Rdata** file. When you initially start R, the **.RData** file in the startup directory is loaded into the computer's working memory and is given the name, **.GlobalEnv**. **.GlobalEnv** can be thought of as the workspace for your current R session. If you create additional objects during your R session, these objects will be added to **.GlobalEnv**, but they will not be permanently saved until you issue a command that saves the workspace. You can use a **save.image** command to save the current workspace to **.Rdata** (see the R-Help page for the **save.image** and **save** functions). In addition, when you quit R, you will be queried whether you want to save your current workspace to c:\diss\data. If you choose to save your current workspace, then all functions and objects in your current workspace will be loaded the next time you start R in c:\diss\data.

How does R find the functions and objects (e.g., data) that are needed in an analysis?

Whenever you enter an R-expression at the R-prompt, R looks for functions or objects that correspond to the functions or objects that are referred to in the expression. For example, suppose you type the following at the R-prompt:

```
>X <- Y + Z
```

R will look for objects called "**Y**" and "**Z**", and try to assign to "**X**" the sum of the values assigned to "**Y**" and "**Z**". To find the objects "**Y**" and "**Z**", R looks through a series of files for objects with these names. This series of files is called the *search path* for R. You can see the current search path by giving the R command, **search()**. For example on my computer, **search()** produces the following information:

```
> search()
[1] ".GlobalEnv"            "file:c:/r/jmm/.RData" "package:ctest"
[4] "package:mass"          "package:nls"          "Autoloads"
[7] "package:base"
```

This output tells us that **.GlobalEnv** is in position 1 (the top position on the search path). The files in positions 2 - 6 are occupied by other files that contain useful functions. Position 7 contains the base package which constitute the default basic set of functions that define the R programming language. The specific search path will differ on different computers, or even on the same computer in different R-sessions; later I will explain how the user can control which files will be placed on the search path.

If the expression, "**X <- Y + Z**", is entered at the R-prompt, R looks for objects corresponding to "**Y**" and "**Z**". R looks first in **.GlobalEnv**, second in the file **c:/r/jmm/.RData**, third in the file **ctest**, fourth in the file **mass**, etc. R assigns to each expression the first object with a matching name that it finds in this search. For example, suppose that there is an object named "**Y**" in **ctest** where "**Y**" has the value 4, there is also an object

named "**Y**" in **mass** where it has the value 10, and there is no object named "**Y**" in any other file along the search path. Y will be assigned the value 4 because this is the value of Y in the earliest file on the search path. Suppose that there is an object named "**Z**" in **.GlobalEnv** where it has the value 2, there is also an object named "**Z**" in **package:nls** where it has the value 15, and there is no object named "**Z**" in any other file along the search path. Then Z will be assigned the value 2 because this is the value of Z in the earliest file on the search path. The expression "**X <- Y + Z**" results in assigning the number 6 to "**X**" (because 6 = 4 + 2). Furthermore the object called **X** will be placed in **.GlobalEnv** (if there previously existed an object called **X** in **.GlobalEnv**, that object would be replaced by the **X** whose value is 2).

In general, whenever an R-expression is entered at the R-prompt, R looks for functions and objects that correspond to functions and objects that are referenced in the expression. R will always interepret such references according to the first function or object of the same name that occurs along the search path.

The **attach**, **library**, and **require** functions can be used to place sets of R objects on the search path. The **detach** function is used to remove files from the search path. Suppose that **c:\diss\data** is the startup directory, and you want to access the objects in a file, **c:\rfiles\previous.rda**. You can place it in position 2 of the search path with the function, **attach("c:/rfiles/previous.rda", pos=2)**. Note that the "\" symbol must be changed to a "/" symbol when indicating directory paths to R. Once **c:\rfiles\previous.rda** is attached to the search path, then functions and other objects in **previous.rda** can be used in the current R program. The **library** and **require** commands are used to attach packages to the search path. See the online documentation for further information about **library**, **require** and **detach**.

Setting the Startup Configuration of R:

You don't have to create a startup configuration for yourself because R has a default configuration that works fine. In the long run, however, you will discover that there are certain settings that you prefer to have as your personal defaults. This section describes how to set these defaults (to the best of my current understanding). See Section 10.8 ("Customizing the environment") of *An Introduction to R* for a fuller, and no doubt more accurate description of the initialization process for R.

When R starts up, it first looks in the file **...\library\base\R\Rprofile.site** for initial program settings; it runs any R commands that it finds in this file. (The symbol "..." refers to the directory that contains the R program; on my computer it is **C:\Program Files\R\R-2.2.1** but it will be different on different computers.) Next R looks in the file **...\etc\Rprofile.site** for additional program settings, and runs any R commands that it finds in this file. Finally R looks for a function called **.First** that may be stored in one of the positions along the search path. It executes the first **.First** function that it finds on the search path, starting from **.GlobalEnv** and ending at the last environment on the search path. Remember, if **c:\data** is the start up directory for the current R session, then R initially looks for a file called **.RData** in c:\data and loads the functions and objects in **.RData** into the current workspace (called **.GlobalEnv**). Thus, if R finds a **.First** function in **.GlobalEnv** at start up, this happens because you created a **.First** function in a previous R session and saved it to the **.RData** file c:\data. If there is no **.First** function in **.GlobalEnv**, then R will look for a **.First** function in the second position, then in the third position, ... As stated, R executes the

first **.First** function that it finds along the search path. If there is no **.First** function along the search path, then no **.First** function will be executed.

The standard way to set the initial configuration of R is to create a **.First** function that defines the desired initial settings of R, and to save this **.First** function to a file on the initial search path of R. The user creates the **.First** function or modifies an existing **.First** function in order that the R program has the desired initial configuration. For example, the following code creates a **.First** function that is a simplification of the one I use (the function is shown on the left; an explanation of the function is shown on the right):

Table 1

`.First <- function() {`	Begins the definition of the **.First** function.
`options(continue = "& ", digits = 8)`	Sets the continuation character to & and the default number of digits after the decimal place to 8.
`attach("C:/R/jmfuns.rda", pos = 2)`	Puts C:\r**jmfuns.rda** on the search path in position 2. Note that a "\" in Windows corresponds to a "/" in R because of R's Unix ancestry.
`attach("data.rda", pos=2)`	Looks in the startup directory for a file called data.rda. If it finds it, it attaches it in position 2 (jmfuns.rda gets bumped to position 3).
`library(foreign)`	This command places the foreign package on the search path. The foreign package contains functions that allow the user to read data from other statistical packages, e.g., from SPSS.
`}`	Ends the function definition.

I want the **.First** function to run whenever I start R because it sets program parameters the way that I like them. These settings become my defaults - they can be overridden whenever I like. If you use R, you will learn what configuration you like, and you can make this configuration your default by defining your own **.First** function.

In order to have the **.First** file run at startup, you need to have **.First** in an appropriate place. Here are two ways to do this.

Method I: Create a separate **.First** function in every directory in which you start up R, and save this function to the **.RData** file for this directory[1]. This method works but it is inconvenient. For example, if you want to make the same change to every **.First** function in every directory that you work in, there is no easy way to do this.

Method II (my preferred method): Method II is slightly elaborate to set up in the first place, but it is very convenient once the necessary configuration has been created. The purpose of Method II is to create a default search path that looks like:

[1] See the section "*Setting the Startup Director for R"* for an explanation of how to set the directory in which R starts.

Table 2

```
> search()
 [1] ".GlobalEnv"          "file:data.rda"    "file:c:/r/jmfuns.rda"
 [4] "package:methods"     "package:stats"    "package:graphics"
 [7] "package:grDevices"   "package:utils"    "package:datasets"
[10] "Autoloads"           "package:base"
```

.GlobalEnv is the current workspace as determined by the startup directory[1]. I use **.GlobalEnv** as a temporary workspace - objects that I want to retain permanently are kept in **c:/diss/data.rda** or **c:/r/jmfuns.rda**.

data.rda is a user-created file that I use to store data objects for this particular project, e.g., data frames, matrices, vectors, and results of analyses. The file 'data.rda' is stored in the startup directory[1]. When working on a different project, I start R with a different startup directory, and a different file of data objects (also called 'data.rda') would be attached in this position.

c:/r/jmfuns.rda contains general purpose functions that I have written for my personal use. When I create new functions or data objects, they are initially stored in **.GlobalEnv**. If I want to save data objects permanently, I move them from **.GlobalEnv** to **c:/diss/data.rda**. If I want to save functions permanently, I move them from **.GlobalEnv** to **c:/r/jmfuns.rda**. This is useful because these function then become available whenever I compute with R (on my own computer), not just when working on this particular project. When I work on a different project, I have a different file of data objects in the second position, but I keep my standard 'jmfuns.rda' function in the third position. Functions or object that have no lasting value are left in **.GlobalEnv**. These files can be saved for temporary use in the **.Rdata** file of the startup directory, and they can be reused in the short term but not the long term.

methods, stats, graphics, grDevices, utils, datasets
These are the packages that R Version 2.2.1 loads by default.

Autoloads See the R-Help documentation.

base The base package contains the base set of functions for R. R won't run without these functions.

The rationale behind this organization is that in any analysis, one creates many functions or data objects that are temporarily useful but are not of lasting value. In the long run, it is confusing to have these objects mixed in with other functions or objects that are of permanent value. Therefore I keep data objects of permanent value to the particular project in a **data.rda** file, and I keep functions that I have written for use in any project in **c:/r/jmfuns.rda**. By keeping objects and functions that have permanent value in files that are separate from **.GlobalEnv**, I avoid the mistake of accidentally deleting them or overwriting them. Of course, I need a function that lets me move new objects or functions from **.GlobalEnv** to **data.rda** or **c:/r/jmfuns.rda** and a function that lets me delete objects from **data.rda** and **c:/r/jmfuns.rda**. These functions (**move** and **rm.sv**) are described below.

The good news is that it is easy to create the code that creates the default search shown in Table 2. The bad news is that this code involves a few, superficially mysterious manoeuvers. If we simply stored the **.First** function defined in Table 1 in the **.Rdata** file of the startup directory, then R would start with the following initial search path:

Table 3

```
> search()
 [1] ".GlobalEnv"                "package:methods"          "package:stats"
 [4] "package:graphics"          "package:grDevices"        "package:utils"
 [7] "package:datasets"          "package:foreign"          "file:data.rda"
[10] "file:e:/r/jmm/jmfuns.rda" "Autoloads"                "package:base"
```

Note that **data.rda** and **jmfuns.rda** follow all of the default-loaded packages. This configuration is created because a separate automatic initialization function, **.First.sys**, loads the set of default R-packages. Since **.First.sys** always runs after **.First**, there is no obvious way to reposition **data.rda** and **jmfuns.rda** ahead of the default R-packages. Fortunately, there is a not so obvious way to accomplish this, as shown next. (Note: The "#" symbol in R code has the effect that all symbols on the same line that occur to the right of "#" are ignored by R.)

Step 1. In order to have the **.First** function run every time R starts up, you need to edit a file called **Rprofile.site** in the directory …**\R\R-Version#\etc**. Upon the initial installation of the R program, this **Rprofile.site** is inactive (has no effect on the running of the program) because every line of the file is commented out ("#" is at the beginning of each line; "#" tells R to ignore this line.) Use a word processor or editor to add the following line to the Rprofile.site file: **attach("c:/r/myfuns.rda", pos=2)**. This line can go anywhere in the file; just make sure that you don't put a "#" at the beginning of this line. The effect of this line is to place **c:\r\myfuns.rda** on the search path. (When you first add this line to the **Rprofile.site** file, the **c:\r\myfuns.rda** file may not yet exist; not to worry - we will create it at Step 4 below.

Step 2: After editing **Rprofile.site**, you need to save it to the …**\R\R-Version#\etc** directory. Be sure that the word processor does not add an extension to the file name, i.e., turn "Rprofile.site" into "Rprofile.site.doc" - R will not recognize the latter name. Also, be sure that you save the **Rprofile.site** file as a text file, not as a document file in the format of the word processor.

Step 3. Run R. Create the following .First function. The function is defined in the left panel of Table 4; the explanation of the code is given in the right panel.

Table 4

`.First <- function()  {`	# Begin the function definition.
`curr.rdefault.packages <-` `    options("defaultPackages")[[1]]`	# Assign the names of the packages that # R currently adopts as the default # packages to a variable called # 'curr.rdefault.packages'.

`for (i in 1:length(curr.rdefault.packages))`
`    library(curr.rdefault.packages[i],`
`    pos=2, character.only=T)`
	# This 'for' loop loads these packages
	# onto the search path. Note that we are
	# forcing this to happen here so that we
	# can put the 'data.rda' and 'myfuns.rda'
	# files ahead of this position on the
	# search path.

```
library("foreign")
library("MASS")
```
Loads the 'foreign' and 'MASS'
packages onto the search path.
This simply reflects my personal
habit of using these packages.
The user can use the 'library' function
to load any packages as a user-defined
default. Leave this blank if the user
prefers not to load any packages by
default (other than the ones that are
assumed in any standard R session.)

```
attach("c:/r/myfuns.rda", pos=2)
```
This code positions 'myfuns.rda' in the
second position on the search path.
Note that

```
 if (!any(tolower(list.files()) == "data.rda") )
{   File.Description <- paste(getwd(),
"/data.rda contains data objects for this project",
    sep="")
save(File.Description, file="data.rda") }
```
If there exists a file called 'data.rda' in
the startup directory, this code has no
effect. If no such file exists, this code
creates a string variable called
'File.Description' and saves it to
'data.rda' (thereby creating this file).
Later you can change the contents of
'File.Description' to give it a more
informative description of 'data.rda'.

```
attach("data.rda", pos=2)
```
This command attaches 'data.rda' to
the search path.

```
LngthS <- length(search())
Dir2nd <- max((1:LngthS)
    [search()=="file:e:/r/jmm/myfuns.rda"] )
detach(pos=Dir2nd)
```
Because of the procedure that is
implemented at Step 4 below, there
will be two occurrences of 'myfuns.rda'
on the search path. This code simply
detaches that last occurrence of
'myfuns.rda'.

The remaining code simply reflects my
somewhat idiosyncratic preferences.
They can be omitted if desired.

```
options(continue = "& ", digits = 8)
```
I prefer to use "& " as the continuation
character, and to display 8 digits after
the decimal place.

```
options(chmhelp=TRUE)
```
This option requests compiled HTML
help.

```
options(editor=
    "C:/Program Files/UEdit70/UEDIT32.EXE")
```
Ultraedit is my preferred editor.

```
par(pch=16, bty="l", xaxs="i", yaxs="i")
palette(value=
    c(colors()[c(24, 503, 255, 128, 109, 142,
              652, 1)]))
dev.off()
```
These are simply my preferred settings
for graphs.

```
tm.t <- confl.jm(all.nm=T)
tm.f <- confl.jm(all.nm=F)
```
Creates some variables to be used
below.

```
cat("Welcome to R for Windows.")                    # Produces my preferred user feedback
cat("\n\n")                                         # at startup.
cat("Current .GlobalEnv = ")
print(getwd())
cat("Current search path is:\n")
print(search())
cat("\nDefault colors are: ")
print(palette())
cat('\nThe following ".*" files have naming conflicts:
\n')
print(tm.t[ !( tm.t  %in%  tm.f ) ])
cat('\nThe following non-".*" files have naming
conflicts:  \n')
print(confl.jm(all.nm=F))
rm(tm.t, tm.f)                                      # Deletes the temporary variables.
        }  #End of function definition              # End of function definition
```

Step 4. Once you have created the **.First** function, save it to the file in which you want to save your personal set of functions. For example, if **c:\r\myfuns.rda** is the path and file name in which you wish to store your functions, enter the code:

```
save(.First, file = "c:/r/myfuns.rda")
```

WARNING: If **c:\r\myfuns.rda** already exists, its contents will be overwritten with **.First** when you give the **save** command. To avoid this, you should first **attach("c:/r/myfuns.rda")**, then move **.First** to **c:/r/myfuns.rda** using the **move** command that is described at the end of this document. The **move** command automatically saves the attached file after moving an object.

This completes the configuration for Method II. With this configuration, the **.First** function in **c:\r\myfuns.rda** will always run at startup no matter what is the startup directory[2]. Additional modifications can be made to the **.First** function in **c:\r\myfuns.rda** to change the preferred initial configuration of R. I have created two functions, **move** and **rm.sv**, that make it easier to work with this configuration. The **move** function moves objects from **.GlobalEnv** to any other environment along the search path and saves this environment to the associated file. It also deletes the object from **.GlobalEnv**. A typical application of **move** is to move a newly defined function from **.GlobalEnv** to the file that contains the user's defined functions, e.g., **c:\r\myfuns.rda** in the example given above. In this way, the function will be available on future occasions when R is started in other directories. In addition, by deleting the function from **.GlobalEnv**, one avoids having multiple copies of the function in different directories, which can be confusing if the function is revised over time.

The **rm.sv** function removes (deletes) an object from an environment on the search path that is not the topmost environment, i.e., not **.GlobalEnv**, and saves that environment to its associated file. If the save operation were not carried out, the object would still be deleted from that environment during the current R session, but it would reappear in that environment if the associated file were attached to the search path in some future R session. The definitions of **move**

[2] The only exception to this is that if there exists a **.First** in the **.Rdata** file of the startup directory, then this function will run at startup and not any other **.First** function.

and **`rm.sv`** are given below. It is also possible to download these functions from http://faculty.washington.edu/jmiyamot/gsdat/dfiles.htm#Rnotes.

Microsoft Word Macros for Writing R Code

It is easier to use R if you use a programming editor like Emacs[3] or UltraEdit[4] while writing the code. I have written a number of macros for Microsoft Word that facilitate the writing of R code in Word. These macros and an explanation of how to use them are contained in a separate document called Rmacros.doc.

Instructions for Downloading R Manuals.

The manuals will be saved to your hardrive as Acrobat pdf files. Put them in whatever directory will be convenient for you.

1. Go to http://www.r-project.org/. Click on Manuals.
2. Click on *An Introduction to R*. This will automatically download an R reference manual.
3. Click on Contributed.
4. Download: ``*Using R for Data Analysis and Graphics*'' by John Maindonald (PDF [695kB], and the data sets and scripts that are available at JM's homepage.
5. Download: ``*Notes on the use of R for psychology experiments and questionnaires*'' by Jonathan Baron and Yuelin Li, and the ``*R reference card*'' by Jonathan Baron.

R-FAQs:

Go to http://www.ci.tuwien.ac.at/~hornik/R/R-FAQ.html.

Documentation for Using R

The R-program comes with documentation through its help system. You can also get free documentation from several users of R. Go to http://cran.r-project.org/. Under the Documentation headings, click on Contributed. You will see the following list of contributed documentation.

``**Using R for Data Analysis and Graphics**'' by John Maindonald (PDF [702kB], data sets and scripts are available at John Maindonald's homepage (http://room.anu.edu.au/~johnm/).

``**R for Beginners / R pour les débutants**'' by Emmanuel Paradis, an introduction in English (PDF [152kB]) and French (PDF [280kB]).

"Practical Regression and Anova using R" by Julian Faraway (PDF [1MB], data sets and scripts are available at the book homepage). This is a concise yet reasonably comprehensive description of regression and anova computations in R.

``**Kickstarting R (version 1.2)**'' compiled by Jim Lemon, a short introduction in English as HTML files: donload as gzipped TAR [64kB] or ZIP [81kB]; or browse directly.

---

[3] Emacs is a well known programming editor. ESS is a version of Emacs that is tailor made for use with statistical programs. You can get more information and download the software for free from http://www.usc.edu/isd/doc/statistics/help/multiuse/ESS.shtml or http://www.stat.math.ethz.ch/ESS/.

[4] Go to http://www.idmcomp.com/ for information about this programming editor. Click on Downloads to download a free trial version of this software. You have to pay $30 if you continue to use it for more than 45 days.

``**Notes on the use of R for psychology experiments and questionnaires**'' by Jonathan Baron and Yuelin Li (HTML [116kB], PDF [235kB]).

``**R reference card**'' by Jonathan Baron (PDF [58kB], LaTeX source [5kB]).

``**Einführung in S**'' by Günther Sawitzki (PDF [884kB]), lecture notes (in German) for a 4-5 day introductory course in programming in the S language for students with basic knowledge in probability theory. See also the StatLab Heidelberg S page for more information.

A Spanish translation of ``**An Introduction to R**'' by Andrés González and Silvia González (PDF file [660kB], Texinfo sources)

I find R for Beginners, Baron's R for Psychology Experiments, and the R Reference Card to be especially helpful. Maindonald's, Faraway's and Lemon's notes are also useful.

Appendix I
The move and rm.sv Functions

The code for the move and rm.sv functions can be downloaded directly from http://faculty.washington.edu/jmiyamot/gsdat/dfiles.htm#Rnotes.

The **`move`** function moves objects from **`.GlobalEnv`** to any other environment along the search path and saves this environment to the associated file. It also deletes the object from **`.GlobalEnv`**. A typical application of **`move`** is to move a newly defined function from **`.GlobalEnv`** to the file that contains the user's defined functions, e.g., **`c:\r\myfuns.rda`** in the example given above. In this way, the function will be available on future occasions when R is started in other directories. In addition, by deleting the function from **`.GlobalEnv`**, one avoids having multiple copies of the function in different directories, which can be confusing if the function is revised over time. By default, the **`move`** is not carried out if an object with the same name exists in the destination directory. The default can be overridden (**`replace=`**T). The move function also applies to a character vector of object names. The move function is defined as follows:

```
move <- function(x, dest = "data", pos=NA, replace.object=F)  {
        if (dest == "data") dir <- "data.rda"

# The next line assumes that user-defined functions are stored in "e:/r/jmm/jmfuns.rda".
# Change this to whatever the current user prefers as the default location of functions.

        if (dest == "fun") dir <- "e:/r/jmm/jmfuns.rda"
        if (is.na(pos)) dirL <- paste("file:", dir, sep="") else dirL <- search()[pos]
        if (mode(x) != "character")  name <- deparse(substitute(x)) else name <- x
# Assign the correct number to tmpos = pos, if it is not already assigned.
        if (is.na(pos)) {
                tmp <- search() == dirL
                if (sum(tmp) < 1) stop(message=paste(dir, "not in search path."))
                if (sum(tmp) > 1)  {
                        cat("There is more than one directory named",
                                dir, "in the search path.\n",
                                "Set the destination with the pos argument.\n")
                        stop(message = "Execution of move terminated.")
```

```
                    } #end of if (sum(tmp) > 1)
                tmpos <- (1:length(search()))[tmp]
                }       else tmpos <- pos                   #End of if (is.na(pos))
# Assign the correct directory name to dirN if it is not already assigned.
        tms <- search()[tmpos]
        dirN <- substring(tms, 6, nchar(tms))
#-------------------------------------------------
# The next if carries out the move, if this is possible.
        e.test <- NULL;  for (i in 1:length(name))
                e.test <- c(e.test, !exists(name[i], env=pos.to.env(tmpos)) )
        if ( all(e.test) | replace.object)   {
                for (i in 1:length(name)) assign(name[i], get(name[i], envir=.GlobalEnv),
                        pos=tmpos)
                save(list=objects(pos=tmpos, all=T), file=dirN )
                rm(list=name, envir=.GlobalEnv)
                }       else    {   #end if, start else
#-------------------------------------------------
# The next code gives warning messages if the move could not be carried out.
# Case I: The target directory was specified by dir.
                if (is.na(pos)) {
                cat("\n No movement of object was carried out!\n Object(s) ",
                name[!e.test], " exists in ", dir,".\n\n",
                'Add "replace.object=T" or "repl=T to the move command in order to replace',
                name[!e.test],"\nin", dir,".\n\n") } else
                {
# Case II: The target directory was specified by pos.
                cat("\n No movement of object was carried out!\n Object(s) ",
                name[!e.test], " exists in pos =", pos,".\n\n",
                'Add "replace.object=T" or "rep = T" to the move command in order to \n',
                'replace', name[!e.test],"in pos =", pos,".\n\n") }
        } #end of case where object exists in destination directory
######### Documentation below this line ######################
# The function "move" puts an R object in another environment on the search
# path, and  deletes it from the current top environment on the search path,
# i.e., deleted it from .GlobalEnv.  By default, the move is not carried out
# if an object with the same name exists in the destination directory.  The
# default can be overridden (replace.object=T).  The move function also applies
# to a character vector of object names.
# x = object to be moved;  or a character vector of object names.
# if (dest == "data"), then the move takes the object to the data.rda file.
# if (dest == "fun"), then the move takes the object to the jmfuns.rda file.
# pos = position number of the target directory;
# pos is set to NA if target directory is specificed by dir.  Specify dest or pos
# but not both.  The setting of pos overrides the setting of dest.
# dir = the name of target R-environment;  it is determined by dest or pos.
# The default setting for dir is dir="data.rda" (the default data storage location).
# Note that dir has the form, dir="e:/....", and not, dir="file:e:/....".
# Note also that the target must be a file, not a package.
```

```
# dirL = file:dir
# dirN = dir if pos was not specified:  else, dirN = directory name corresponding to pos
# replace.object = T to replace same name object in target directory
} #end of function definition ------------------------------------------------
```

To use the move function, specify x to be the object to be moved or a character vector of object names to be moved. Change **`dir="c:/r/myfuns.rda"`** in the function definition to whichever file you wish to be the default destination of the move operation. The destination environment of the move can be specified either by **`pos=k`** where k = the position number of the destination on the search path, or as an environment name, as in **`dir="c:/r/myfuns.rda"`**. If both **`pos`** and **`dir`** are specified, the specification of **`pos`** overrides the specification of **`dir`**.

EXAMPLES	EXPLANATION
`move(x1)`	Moves the object x1 from .GlobalEnv to the default destination, which is **`data.rda`** file in the startup directory.
`move(x1, pos=3)`	Moves the object x1 from .GlobalEnv to the environment in position 3 of the search path.
`move(c("x1","x2","x3") )`	Moves the objects, x1, x2, and x3, from .GlobalEnv to the default destination, which is **`data.rda`** file in the startup directory.
`move(c("x1","x2","x3"), pos = 3 )`	Moves the objects, x1, x2, and x3, from .GlobalEnv to the environment in position 3 of the search path.

The **`rm.sv`** function removes (deletes) an object from an environment on the search path and saves that environment to its associated file. If the save operation were not carried out, the object would still be deleted from that environment during the current R session, but it would reappear in that environment if the associated file were attached to the search path in some future R session. The rm.sv function also applies to the objects named by a character vector of object names.

```
rm.sv <- function(x, dir="data.rda", pos=NA)  {
        if (is.na(pos)) dirL <- paste("file:", dir, sep="") else dirL <- search()[pos]
        if (mode(x) != "character")  name <- deparse(substitute(x)) else name <- x

# Assign the correct number to tmpos = pos, if it is not already assigned.
        if (is.na(pos)) {
                tmp <- search() == dirL
                if (sum(tmp) < 1) stop(message=paste(dir, "not in search path."))
                if (sum(tmp) > 1)  {
                        cat("There is more than one directory named",
                                dir, "in the search path.\n",
                                "Set the destination with the pos argument.\n")
                        stop(message = "Execution of deletion terminated.")
                        } #end of if (sum(tmp) > 1)
                tmpos <- (1:length(search()))[tmp]
                }       else tmpos <- pos                   #End of if (is.na(pos))
```

```
# Assign the correct directory name to dirN if it is not already assigned.
        tms <- search()[tmpos]
        dirN <- substring(tms, 6, nchar(tms))
#-------------------------------------------------
# The next if carries out the deletion, if this is possible.
        e.test <- NULL;  for (i in 1:length(name))
                e.test <- c(e.test, exists(name[i], env=pos.to.env(tmpos)) )
        if ( all(e.test))   {
                remove(list=name, pos=tmpos)
                save(list=objects(pos=tmpos, all=T), file=dirN )
                 }  else   {   #end if, start else
#-------------------------------------------------
# The next code gives warning messages if the deletion could not be carried out.
# Case I: The target directory was specified by dir.
                if (is.na(pos)) {
                cat("\n No deletion of object was carried out!\n Object(s) ",
                name[!e.test], " does not exist in ", dir,".\n\n") } else
                {
# Case II: The target directory was specified by pos.
                cat("\n No deletion of object was carried out!\n Object(s) ",
                name[!e.test], " does not exist in ", dir,".\n\n")
        } } #end of case where object(s) do not exist in destination directory
######### Documentation below this line ######################
# The function "rm.sv" delete an R object from another environment on the search
# path, and then saves that environment to its associated file.  The rm.sv function
# also applies to the objects specified by a character vector of object names.
# x = object to be deleted;  or a character vector of object names.
# pos = position number of the directory containing the object;
# pos is set to NA if directory is specificed by dir.
# dir = the name of R-environment containing the object;
# By default, dir = "data.rda" (in the current active directory)
# Specify pos or dir, but not both.
# Note that dir has the form, dir="c:/....", and not, dir="file:c:/....".
# Note also that the directory must be a file, not a package.
# dirL = file:dir
# dirN = dir if pos was not specified:  else, dirN = directory name corresponding to pos
} #end of function definition
```

To apply **`rm.sv`**, **`x`** = object to be deleted or character vector of names of objects to be deleted; **`pos`** = position number of the target directory; **`pos`** is set to NA if target directory is specificed by **`dir`**; **`dir`** = the name of target R-environment; specify **`pos`** or **`dir`**, but not both; if **`pos`** and **`dir`** are both specified, the specification of **`pos`** overrides the specification of **`dir`**.

EXAMPLES	EXPLANATION
`rm.sv(x1, pos=2)`	Deletes (removes) the object **`x1`** from the environment in position 2 of the search path.
`rm.sv(c("x1","x2","x3"), pos=2)`	Deletes (removes) the objects **`x1`**, **`x2`**, and **`x3`** from the environment in position 2 of the

	search path.