R for Windows Users

Version 2.01

Ko-Kang Wang
Postgraduate (PGDipSci) Student
Department of Statistics
New Zealand

October 9, 2002

Acknowledgments

There are many people whom I would like to give my appreciation to, as without them, the existence of this document would have been impossible.

Firstly, The R Core team! For their efforts in the implementation of R. In particular, I would like to thank Prof. Brian Ripley, for his page on **Building R for Windows**, http://www.stats.ox.ac.uk/pub/Rtools/, which helped me to structure this document. I would also like to thank Dr. Ross Ihaka, for being my summer research supervisor for the past three year. It was Dr. Ihaka who first suggested me to write up something about using R under Windows. I also need to give my great appreciation to Dr. Paul Murrell, for reviewing this document and providing suggestions, as well as correcting my grammar.

I would also like to thank the many people from the `r-help` and `r-devel` mailing lists, for providing constructive comments, suggestions, and criticisms. In particular I would like to acknowledge Vito Muggeo, and Steve Wisdom for their valuable suggestions to improve this document.

Preface

Being kind of a lazy, or efficient (depends on your point of view) person, I often like to have a book that contains almost all the common things one needs to do, so I can refer to it all the time, rather than searching around to find a piece here and another piece there.

It is with this intention that I decided to write up this little handbook. The idea is to have things that a common R user, whose operating system is Windows, often encounters.

As the title suggests, this little handbook is specifically targeted at Windows environments. Everything described in it *should* work in all versions[1] of windows, however as I do not have the means to test them on ALL of the versions, I cannot guarantee this. If you find anything that does not work on a particular version of Windows, please do let me know.

It is also worth noting that a basic idea of the Windows environment is assumed. For example, I will **NOT** go and explain how you can find a file, or use a mouse, in Windows.

This document is still evolving, therefore it will be greatly appreciated if you can provide suggestions. Currently I am planning on adding in a chapter on CygWin support during summer (in the Southern Hemisphere) 2002/2003.

Enjoy reading...

Ko-Kang Wang
kwan022@stat.auckland.ac.nz

[1] That is, 9x/NT/2000/XP.

Contents

Chapter 1

Introduction

At the time of drafting up this handbook, the latest version of R is R-1.6.0. Therefore it is assumed that users will have R-1.6.0 or later installed.

There are already many sources of documentation on how to do statistics in R, for example, John Verzani's **Simple R** at http://www.math.csi.cuny.edu/Statistics/R/simpleR. Therefore in this handbook I will not attempt to explain how to do statistics, but rather I will go through some general areas, such as installing R on Windows (which is actually very straight forward), basic use (such as how to invoke R), compile R from source, and compile R packages...and so on.

In addition, when I wrote this handbook, I tried to word in a way that it can be understood by people are experienced Windows users, but have not used R much before; and people who are experienced Mac or UNIX users who may or may not have lots of experience with R, but need to use R under Windows in some (perhaps unfortunate) situations.

Any comments, positive or negative, may be sent to *kwan022@stat.auckland.ac.nz*
.

Chapter 2

Installation

The installation process for Windows is pretty straight forward, assuming you have at least installed some other application software on your current Windows platform.

2.1 Installing R Base

Firstly you will need to download the setup file from **CRAN**[1], http://cran.r-project.org/ or one of its mirror sites. Once you are at CRAN's homepage, you will see a table that looks like:

All Platforms

- Download the source code of the latest release (2002-06-17): R-1.6.0.tgz (or read what's new in the latest version).
- Sources of contributed packages
- Current patch set (daily snapshot): R-release.diff.gz.

Precompiled Binary Distributions (Base system and contributed packages)

- Alpha Unix (OSF/Tru64)
- Linux
- MacOS (System 8.6 to 9.1 and MacOS X)
- MacOS X (Darwin/X11)
- Windows (95 and later)

It should be obvious that you should click on **Windows (95 and later)**. Then you will see a list of folders, click on **base**. Now you have a choice, there are two flavours of R that you can choose:

rw1060.exe is a complete setup file (about 19MB), which is probably the easiest way to install R for most people.

[1] The Comprehensive R Archive Network

mini is a set of installation files that can be put onto floppy disks. You can put `miniR.exe` and `miniR-1.bin` on one floppy, and `miniR-2.bin` to `miniR-8.bin` on separate floppies.

Assuming you downloaded **rw1060.exe**, then you just need to run it by double clicking on the file.

The rest of the steps are trivial, as you pretty much just keep clicking on the NEXT button. It is worth noting that when you are asked to select the components that you wish to install, you may want to tick **Source Package Installation Files** if you wish to use **Rcmd**[2] later.

After the installation, you may wish to make a shortcut to **Rgui.exe** on your desktop for easy access in the future. The file can be found from `$R_HOME\bin`. `$R_HOME` refers to the directory where you installed R, for example,

`C:\Program Files\R\rw1051`.

2.2 Installing Packages

To get a package, the easiest way is to use Rgui. In Rgui, at the top you will see the standard pull-down menus, click on `Packages -> Install package from CRAN...` . You will then see a list of packages which you can select from.

[2] See Section 3.2.

Chapter 3

Running R

There are several ways to *run* R under Windows environment, namely **Rgui**, **Rcmd**, and **Rterm**. They are discussed in this chapter. Note that in order to run R smoothly, and a must if you want to proceed to 3.2 and 3.3, you need to put `$R_HOME\bin` in your `PATH` variable, which is explained in Chapter 4.2 if you don't know how to do it.

3.1 Rgui

Running Rgui is straight forward, assuming you have created the shortcut on your desktop as explain in 2.1. You simply need to double click on it, and you will be in.

It is an interactive mode, which means you input line by line and can see the immediate result(s) if you wish to.

3.1.1 Setting Working Directory

Though not necessary, it is a good idea to set your working directory before you start Rgui. To set it, you can right click on your Rgui shortcut, choose `Properties`, then in the field that is labeled **Target Location**, you can set it to the directory you want to work in. The default setting is `$R_HOME\bin`. There are several advantages in doing this, one is that your command history will automatically be set into your **current working directory**. Another advantage is when you need to read in or write out files, you do not need to specify the **absolute path**. Take a hypothetical example, suppose I have created a few functions and save them in `C:\Temp\foo.R`. If I have set the working directory to `C:\Temp\` before I start Rgui, then in Rgui I can just type:

```
source("foo.R")
```

Note that the `"foo.R"` is enclosed by **double quotes**!

Further suppose that I have another file called `foo.txt`, which is a tab delimited table[1], in `C:\Temp\`, that looks like:

[1] An ASCII formatted table like this is very common, and one can generate this easily from any spreadsheet programme. An ASCII formatted table is the best way to distribute your files as all analysis packages can read it in and convert into its own format, if necessary.

```
Index Values
    1    0.9
    2   -0.6
    3   -0.8
    4   -0.9
    5   -2.1
    6   -0.4
    7   -0.2
    8    0.3
    9    0.0
   10   -1.5
```

Then you can just do the following in Rgui:

```
fred <- read.table("foo.txt", header = TRUE)
attach(fred)
```

However, if you do not like the idea of setting and changing your working directory, you can use the **file.choose()** function. For example, suppose that I did not set the working directory to `C:\Temp\` before I opened Rgui, then I can just type:

```
source(file.choose())
```

This will bring up a window which allows you to browse through directories. You can then find your `foo.R` from this chooser window. In the **read.table()** example above, you can do:

```
fred <- read.table(file.choose(), header = TRUE)
```

to find your `foo.txt`.

3.1.2 Writing, Editing Commands

It is sometimes difficult to write your R commands directly in Rgui, as if you need to change or correct something later, you need to keep pressing the up/down arrow keys.

From my experience, I find it is best to write my R commands in a note-pad equivalent programme. Then copy a line or a block of lines into Rgui. This makes the editing much easier, and also makes it much easier to save your commands.

Another suggestion for Microsoft Word users: when pasting your R commands into Word, change the fonts to `Courier New`! This will make your code look much nicer (try it!).

3.2 Rcmd

Sometimes it may be time-consuming, especially for advanced users, to run in an interactive mode where you must input your commands line by line, or copy and paste them from a note-pad. It is especially so if you are doing something time-consuming, for example data mining.

For advanced users, and also recommended for beginners, it is a very good idea to type your commands in a note-pad equivalent programme. It makes debugging and correction much easier, and it is almost certain that you will need to make corrections, maybe just to correct a simple spelling mistake.

So assume you type the following commands in a note-pad:

```
x <- rnorm(1:10000)
y <- runif(1:10000)
summary(x)
summary(y)

postscript(‘‘random.ps’’, height = 5, width = 5.5,
           horizontal = FALSE)
plot(x, y, cex = .6,
     xlab = ‘‘Random Normal’’, ylab = ‘‘Random Uniform’’,
     main = ‘‘10000 Random Numbers’’)
dev.off()
```

and saved it as `random.R`. In order to run it all together, i.e. in a **BATCH** mode, you need to be in command mode (see Chapter 4.3). `cd` into the folder that you saved the file, then type:

```
Rcmd BATCH random.R
```

and the resulting output will be saved in `random.Rout`. You can see the outputs by opening `random.Rout` in a note-pad.

More optional parameters can be found by typing:

```
Rcmd BATCH --help
```

Running in BATCH mode, however, is not recommended for beginners. It is less intuitive and immediate feedback is not given (you need to see them by opening the Rout file).

3.3 Rterm

Rterm is for people who wish to run R in a DOS-prompt. Again you need to be in DOS-prompt (or command line mode), then type **Rterm** to get into R.

Chapter 4

Compile R Source

The *official version* of this chapter can be found from Prof. Brian Ripley's page, http://www.stats.ox.ac.uk/pub/Rtools/.

4.1 Preparation

There are several things that must be downloaded and installed first:

1. **ActivePerl**, http://www.activestate.com/Products/ActivePerl/.
2. **ActiveTcl**, http://www.activestate.com/Products/ActiveTcl/.
3. A set of tools bundled by *The R Core Team*, http://www.stats.ox.ac.uk/pub/Rtools/tools.zip.
4. **HTML Help Workshop**, http://www.microsoft.com/office/ork/xp/appndx/appa06.htm, for compiling CHTML files.
5. **MinGW**, http://www.mingw.org/
6. For LaTeX 2$_\varepsilon$ documentations, MikTeX http://www.miktex.org/.
7. For building Windows installer (rw1060.exe), http://jrsoftware.org/isinfo.php.
8. The `R` Source file, e.g. `R-1.6.0.tgz`, http://cran.r-project.org.
9. **Libpng**, **jpeg sources**, http://www.libpng.org/pub/png/libpng.html, ftp://ftp.uu.net/graphics/ respectively.
10. **PCRE**, ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/. (pcre-3.9.tar.gz)
11. **bzip2** Sources, http://sources.redhat.com/bzip2/. (bzip2-1.0.2.tar.gz).

Some of them, such as MikTeX, are very large to download. If you are as unfortunate as me, you will be on a 56kbps modem. If that is the case, you may wish to download them, MikTeXin particular, at night when you are asleep. It will be done when you wake up the next morning – hopefully!

To compile this version, R-1.6.0, you need a recent version of **MinGW**. You will be able to find version like `MinGW-2.0.0-3.exe` from MinGW's web site. It is the easiest way to get everything you need, as you will need the latest GCC (3.0.3) as well.

In fact, you can probably get away with not having some of the above software installed. For example, you won't be needing MikTEX ActiveTcl if you don't need any PDF manuals and Tcl support. You don't need the HTML Help Workshop, if you don't need CHTML files. However some of these may require you to edit the `MkRules` (Section 4.3). If you intend on re-distribution of your compiled version, you must have everything, including Tcl support, PDF documentation, ... and so on compiled.

4.2 Set `PATH` Variable

This section helps you to set up the **PATH** environmental variable in Windows. You need to add the full path of the directory to the PATH variable. Typically this full path looks something like `C:\bin`.

Your `PATH` should look something like:

```
C:\bin;C:\MinGW\bin;C:\Perl\bin\;C:\Tcl\bin;
C:\texmf\miktex\bin;
```

Where `C:\bin` is the directory (you may change the name if you wish) you unzipped the tools bundled by *The R Core Team*, this should appear in front of your `PATH`.

You should add other appropriate paths to suit your needs. For example, if you want to build the compiled HTML help files, then you need to add `hhc.exe` to your path.

If you have other paths in your `PATH`, simply append the paths at the end. Do NOT, unless you know what you are doing, delete any paths that already appeared in your `PATH`.

4.2.1 Windows 9x

Open the `AUTOEXEC.BAT` file and add or change the `PATH` statement as follows:

1. Start the system editor. Choose "Start", "Run" and enter `sysedit`, then click OK. The system editor starts up with several windows showing. Go to the window that is displaying `AUTOEXEC.BAT`.

2. Look for the `PATH` statement. (If you don't have one, add one.) If you're not sure where to add the path, add it to the right end of the `PATH`. For example, in the following `PATH` statement, the `AUTOEXEC.BAT` directory has been added at the right end:

   ```
   PATH = C:\WINDOWS;C:\WINDOWS\COMMAND;C:\BIN
   ```

   Capitalization doesn't matter. The `PATH` can be a series of directories separated by semi-colons (;). Microsoft Windows searches for programs in the `PATH` directories in order, from left to right. You should only have one bin directory under `C:\` in the path at a time (those following the first are ignored).

3. Restart the computer

4.2.2 Windows ME

To set the `PATH` permanently: From the start menu, choose programs, accessories, system tools, and system information. This brings up a window titled **Microsoft Help and Support**. From here, choose the tools menu, then select the system configuration utility. Click the environment tab, select `PATH` and press the edit button. Now add the paths to your `PATH`. After that, save the changes and reboot your machine when prompted

4.2.3 Windows NT, 2000, and XP

To set the `PATH` permanently: From the start menu, open the Control Panel, and from there, double click the System icon to open the System Control panel. In the System control panel, select the Advanced tab, and then click the **Environment Variables**; button. This will bring up a window in which you can edit system variables, including the `PATH` variable. After you've added the paths to your `PATH`, save the changes and reboot your machine when prompted.

4.3 Compiling R from Source

This is not compulsory and personally I do not recommend this unless you are a developer, or an experienced user who is not afraid to try this! If you are a normal end-user who simply wishes to get R running on your machine, then please read Section 2.1.

Assuming you have downloaded the official release version, `R-1.6.0.tgz`, then under DOS-Prompt (`Start -> Programs -> Dos Prompt` if you are using Windows 9x, or `Start -> Programs -> Accessories -> Command Prompt` if using Windows NT/2000/XP). If you can't find it, then under `Start -> Run...`, type **cmd** to get in. Once you are in the command prompt, cd into the directory you save the file in and type:

```
tar zxvf R-1.6.0.tgz
```

this unpacks the compressed tar-ball to `R-1.6.0` directory.

You will then need to uncompress `pcre-3.9.tar.gz` and `bzip2-1.0.2.tar.gz`. Then copy them to `R-1.6.0/src/extra/pcre`, and `R-1.6.0/src/extra/bzip2`, respectively.

Then cd into `R-1.6.0/src/gnuwin32`:

```
cd R-1.6.0/src/gnuwin32
```

You may need to edit the file `MkRules`. For example, it is most likely that you will need to edit the following "blocks":

```
## ============ configuration macros for building R ============

LEA_MALLOC=YES

# Set to YES and specify the path if you want to use the
```

```
# ATLAS BLAS.
USE_ATLAS=NO
ATLAS_PATH=/R/ATLAS/lib/WinNT_PII

# Where does 'HTML Help Workshop' live? (unused if compiled
# HTML help is not requested. Spaces allowed.)
HHWDIR=C:/Program Files/HTML Help Workshop

# Where does Tcl/Tk live? Spaces allowed.
TCL_HOME = C:/packages/Tcl # The path to where you install Tcl/Tk
TCL_VERSION = 83           # Your Tcl/Tk version.
```

and

```
## ====== configuration macros for building installers ======

# location where Inno Setup was installed
ISDIR=C:/packages/Inno
# location where zip files of binary packages are kept
PKGS=C:/R/Rdist/libs
```

Before you proceed, it is strongly recommended that you close all running programmes, including those running in the background, such as screen savers, anti-virus programme. This process will take quite a while.

Then you just need to type:

```
make
```

and wait for it to compile.

4.3.1 Building Bitmap Device Support

You will need to build the bitmap device support, `Rbitmap.dll`, manually.

You need to unpack the `libpng` and `jpeg` sources that you downloaded earlier. They must be unpacked into sub-directories of `R-1.6.0/src/gnuwin32/bitmap`. For example, assuming your `libpng-1.2.4.tar.gz` and `jpegsrc.v6b.tar.gz` are saved in the `R-1.6.0/src/gnuwin32/bitmap` directory, then you may type:

```
tar zxvf libpng-1.2.4.tar.gz
tar zxvf jpegsrc.v6b.tar.gz
mv libpng-1.2.4 libpng
```

Note that if your jpeg version is not 6b, you need to edit the `JPEGDIR` definition in `R-1.6.0/src/gnuwin32/bitmap/Makefile`.

Once you have done this, type:

```
make bitmapdll
```

under `R-1.6.0/src/gnuwin32/`.

4.3.2 Building Tcl/Tk Support

Before you continue with this chapter, you must make sure ActiveTcl has been installed, and the file `R-1.6.0/src/gnuwin32/MkRules` has been modified with the path to your Tcl directory.

Then you just need to type:

```
make tcl
```

under the directory `R-1.6.0/src/gnuwin32/`.

4.3.3 Building the Manuals

You may also wish to build the manuals (*An Introduction to R*, *Writing R Extensions*, ... and so on).

For PDF manuals of the manuals, type:

```
make docs
```

under `R-1.6.0/src/gnuwin32/`.

For info versions (not the Reference manuals), you need a version of `makeinfo 4.2`, which is included in the tool set provided by `The R Core Team`. Then cd into `R-1.6.0/doc` and type:

```
make -f Makefile.win info
```

For DVI versions of the manuals, type:

```
make -f Makefile.win dvi
```

under the directory `R-1.6.0/doc`.

For HTML versions of the manuals, type:

```
make -f Makefile.win html
```

under the directory `R-1.6.0/doc`.

4.3.4 Building the Installers

You need to have `Inno Setup 2.0.17` or later.

You also need a complete built `R`, including bitmap device support, Tcl support, reference manuals.

You also need, at least, all the recommended packages. The recommended packages are now come with `R-1.6.0.tar.gz`. SO all you need to do is type:

```
make recommended
```

under `R-1.6.0/src/gnuwin32/`, and check by:

```
make check-recommended
```

If you want to add in some extra packages, you will need to edit the:

```
# chapter for JR software installer.

RECOMMENDED=KernSmooth VR boot cluster foreign mgcv nlme rpart
survival grid lattice
EXTRA=
```

chapter in the file `R-1.6.0/src/gnuwin32/installer/Makefile`.

Then, under `R-1.6.0/src/gnuwin32/` type:

```
make
make distribution
make recommended
make fullcheck
make rinstaller
```

to make the distribution zip files.

You may need to edit `ISDIR` in the `MkRules` in `R-1.6.0/src/gnuwin32/`, which are the directory to your `Inno Setup`.

It will take quite a long time, especially the `make fullcheck`. As a benchmark, it took me more than 1 hour to run through the whole compilation process to get my final **rw1060.exe** installation file, on my Pentium 4, 1GHz laptop with 320MB RAM.

4.3.5 Notes to Inno Setup 3.x Users

At the time I wrote this document, `Inno Setup 3.0.3-beta` has been released. If you are using it, or a any Inno Setup 3.x versions, then you will need to make a little change to your **R.iss** under `R-1.6.0/src/gnuwin32/installer`. Line 12 (your **R.iss** may be different) says:

```
AlwaysCreateUninstallIcon = YES
```

you need to remove this line. `AlwaysCreateUninstallIcon` option is no longer supported in Inno Setup 3.x or later.

If you want this option, you can put something like:

```
Name: "{group}\R 1.6.0"; Filename: "{uninstallexe}"
```

into the `[Icons]` block.

Chapter 5

Build R Package

The official version of this chapter is *Writing R Extensions*, http://cran.r-project.org/manuals.html. If you will be writing documentation in R format, you should also read *Guidelines for Rd files*, http://developer.r-project.org/Rds.html.

5.1 Preparation

It is recommended that you have all the software listed in Chapter 4.1 installed first. Then set your `PATH` variable as described in Chapter 4.2. You must have a compiled working version of R installed, and have `$R_HOME\bin` placed in your `PATH`.

Now create a folder (for reference purpose, let's call it `foo`) that will contain your new package. The folder shall contain several sub-folders[1]. They are:

```
R
help
html
latex
man
```

Although not compulsory, it is strongly recommended that you set your working directory to be the `man` folder in `foo`. To do this, right click on the `Rgui.exe` icon (or its shortcut on the desktop if you have one), choose `Properties`, then put the `absolute path` of the `man` folder in the working directory text field.

This will ensure you can use:

```
> prompt(file)
```

to output a template of the `Rd` file into the `man` folder directory.

5.2 Documenting R Functions

Suppose I have written a function as follows:

[1] Refer to *Writting R Extensions* for more details

```
> my.layout <- function(m, n) {
+     par(mfrow = c(m, n))
+ }
```

And I want to write a `Rd` documentation for it. I can simply type:

```
> prompt(my.layout)
```

and R will output a template into the `man` folder, or whatever your current working directory happens to be.

The resulting output, named `my.layout.Rd` has the content as follows:

```
\name{my.layout}
\alias{my.layout}
%- Also NEED an '\alias' for EACH other topic documented here.
\title{ ~~function to do ... ~~ }
\description{
  ~~ A concise (1-5 lines) description of what the function does. ~~
}
\usage{
my.layout(m, n)
}
%- maybe also 'usage' for other objects documented here.
\arguments{
  \item{m}{ ~~Describe \code{m} here~~ }
  \item{n}{ ~~Describe \code{n} here~~ }
}
\details{
  ~~ If necessary, more details than the __description__  above ~~
}
\value{
  ~Describe the value returned
  If it is a LIST, use
  \item{comp1 }{Description of 'comp1'}
  \item{comp2 }{Description of 'comp2'}
  ...
}
\references{ ~put references to the literature/web site here ~ }
\author{ ~~who you are~~ }
\note{ ~~further notes~~ }

 ~Make other chapters like WARNING with \chapter{WARNING }{....} ~

\seealso{ ~~objects to SEE ALSO as \code{\link{~~fun~~}}, ~~~ }

\examples{
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--         or do  help(data=index)  for the standard data sets.

## The function is currently defined as
```

```
function(m, n) {
    par(mfrow = c(m, n))
  }
}
\keyword{ ~kwd1 }% at least one, from doc/KEYWORDS
\keyword{ ~kwd2 }% __ONLY ONE__ keyword per line
```

You can then modify this file. Please refer to *Writting R Extensions* for more details on how to edit this file.

5.3 Documenting Data Sets

Suppose I have a data frame called FG, which looks like[2]:

```
> FG
           FOO          GOO
1  -0.1646470  0.204260905
2  -1.2444621 -1.827261659
3  -0.8822063  0.499977046
4  -0.8628432  0.445908205
5  -0.5528232  0.732852476
6  -1.0124162  0.206794752
7  -1.6423894  2.322223652
8   1.1509125 -2.139585291
9   1.6897148 -0.006194043
10  0.5261203  0.613050939
```

I can then do:

```
> prompt(FG)
```

which created a file `FG.Rd` that has the contents:

```
\name{FG}
\alias{FG}
\non_function{}
\title{ ~~ 1-line description of the data frame ~~ }
\usage{data(FG)}
\description{
The \code{FG} data frame has 10 rows and 2 columns.
~~ Give a concise description here ~~
}
\format{
  This data frame contains the following columns:
  \describe{
    \item{FOO}{a numeric vector}
    \item{GOO}{a numeric vector}
  }
}
```

[2]Both FOO and GOO are simply some normal random numbers, generated for the purpose of showing this example.

```
\details{
 ~~ If necessary, more details than the _description_ above ~~
}
\source{
 ~~ reference to a publication or URL from which the data were obtained ~~
}
\references{
~~ possibly secondary sources and usages ~~
}
\examples{
data(FG)
## maybe str(FG) ; plot(FG) ...
}
\keyword{datasets}
```

You can then modify this file. Please refer to *Writting R Extensions* for more details on how to edit this file.

5.4 Compile the Package

You must have at least one `Rd` files in your `man` directory. If you just wish to test if you can compile a package, but have not got any documentations written yet, simply create a file called, say, `test.Rd`, and put it in the `man` directory.

Once you are ready to compile the package, open a DOS-prompt under Windows, then cd into the parent directory of `foo`, and type:

```
Rcmd build --binary foo
```

Or if you wish:

```
Rcmd build --binary --use-zip foo
```

which will zip your data sets and help files. Either command will produce a `foo_*.zip`, where `*` is the version of the package[3]. This file is a compiled package, which can be installed directly into R (Section 2.2).

Now, with the first command, once you installed your library, which by default is installed in `$R_HOME\library\mypack`, you will notice that under the `data` and `help` sub-directories, the files are un-zipped.

With the second line up there, i.e. with the `--use-zip` parameter, both the `data` and `help` sub-directories will show a zipped file in each of them. For example, take a look at `$R_HOME\library\base`. Under its `data` and `help` sub-directories, you will see a zip file.

To make it more clear, in the command prompt window (I assume you already have `$R_HOME\bin` in your PATH), type:

```
Rcmd build --help
```

you will then see that with the

```
Rcmd build --binary --use-zip-data
```

[3]The version is what you put in the `DESCRIPTION` file. Please refer to *Writing R Extensions*.

`foo` will produce a package such that, after installed, ALL the data files in the `data` sub-directories will be in a zip file. On the other hand, with

```
Rcmd build --binary --use-zip-help
```

will put all the files in the `help` sub-directory into a zip file. Finally, with

```
Rcmd build --binary --use-zip
```

is a combination of the above two.

It is usually a good idea to do `--use-zip` to save disk space.

Under the Windows environment it is not too uncommon to have folders with spaces in their names, e.g. `C:\Program Files\`. However when you try to compile your package, these *spaces* may give you problems. Therefore it is recommended, from my experience, that you copy the entire package folder (e.g. **foo**) into a folder that does not have any spaces in its name, e.g. `C:\Temp\`.

Chapter 6

Emacs Speaks Statistics (ESS)

ESS is an Emacs-Lisp interface to interactive statistical programming and data analysis languages, including: S dialects (such as R), LispStat dialects and SAS. **Emacs** is extremely powerful and is not just an *editor*, but also a *legend*.

6.1 Getting (X)Emacs and ESS

To get **ESS** to work you first need a working Emacs or **XEmacs**. If you wish to get XEmacs then you can download it from http://www.xemacs.org/Download/index.html. XEmacs has more graphical user interface than Emacs, and is a bit easier to install (therefore I will not discuss it here).

To get Emacs, Dr. Claus Dethlefsen[1] from the Department of Mathematical Sciences at Aalborg University has a very comprehensive web page explaining how to get it and install it, with LaTeXand other plug-ins. For completeness I will extract out the section that gets ESS to work.

The most important step is obviously to download Emacs, you can get it by going to http://ftp.sunsite.dk/pub/gnu/windows/emacs/latest/, it is recommended that you download the **fullbin**. If you wish to get AucTeX– the LaTeXplug-in, and Ispell – the spell checker, please refer to Claus's page for the detailed instructions.

You will also need to download ESS, which is available from CRAN (http://cran.r-project.org/other-software.html) under `Software -> Other`. You may wish to download the zip file if you are unfamiliar with the tar.gz files.

6.2 Installing ESS

Most of this section is quoted from Claus's web site.

To install Emacs:

- Unzip the **emacs-21.2-fullbin-i386.tar.gz** file to `C:\`[2] (1876 files). You must extract with the folders.

[1] http://www.math.auc.dk/~dethlef/Tips/introduction.html

[2] You may extract to elsewhere if you wish.

- Rename the `c:\emacs-21.2` directory to `C:\emacs`
- Run the file `C:\emacs\bin\addpm.exe`

Test: Open the **Start Menu/Programs/Gnu Emacs/Emacs**. Emacs should start up.

If you wish to use LaTeX, then at this stage it is best to go to Claus's web page for more instructions. Otherwise, you may follow the instructions below to simply get ESS working.

First of all, when Emacs starts it will read in the configuration file **.emacs**, you will need to specify a `HOME` variable that will contain the **.emacs** file. To create such a variable you simply follow the same instructions as in setting up PATH in Chapter 4.2, except now you will need to create a new name called **HOME**. It is recommended that you set it to `C:\`.

Now, extract (unzip) the ESS file that you downloaded into the folder `C:\emacs\site-lisp`. You must extract with the folder names. The files will be extracted under the sub-folder `ess-5.1.20`, rename it to `ess`.

Once this is done, open your favourite text editor, then type the following lines:

```
;; Set up ESS
(add-to-list 'load-path "c:/emacs/site-lisp/ess/lisp")
(require 'ess-site)
```

Note that you may need to change the path to where your emacs was installed.

Open the file `C:\emacs\site-lisp\ess\lisp\ess-site.el`, go to line 254 and delete the two semi-colons (which means comments) so it now reads:

```
(setq-default inferior-R-program-name "Rterm")  ; msdos systems
```

Be sure to have your `R_HOME\bin` in the `PATH`.

Claus suggested that to speed up ess, in Emacs, type

```
C-u 0 M-x byte-recompile-directory
```

and choose `C:\emacs\site-lisp\ess\lisp`.

If you have been using (X)Emacs in other operating systems then you will know that **C-u** means **Ctrl, u** keys pressed sequentially, while **M-x** means **Alt, x** keys. You may also know that you can type the Tab key after typing **byte-re** to get the `byte-recompile-directory`. But I will not explain any more about tricks in Emacs here.

6.3 ESS Quick Reference

This section was originally put together by Dr. Ross Ihaka. For the completeness of this chapter I have added it in.

Emacs commands are accompanied either by simultaneously holding down the *Control* key (indicated by `C-`) or the *Alt* key (indicated by `M-`). The abbreviation `TAB` means the *Tab* key and `RET` means the *Return* key.

Essential Commands

`M-x S`	start Splus
`M-x R`	start R
`C-c C-q`	quit from Splus or R
`C-c C-v`	open a help window
`TAB`	object name completion
`C-c C-u`	kill present command
`C-c C-w`	kill last word of present command

Command History

`M-p`	scroll backward through command history
`M-n`	scroll forward through command history
`M-r`	retrieve a previous command by pattern
`M-s`	retrieve a later command by pattern

Session Transcript

`C-c C-e`	jump to the end of the transcript
`C-c C-p`	jump to previous command in transcript
`C-c C-n`	jump to next command in transcript
`RET`	copy command to the input line and execute
`C-c RET`	copy command to the input line, don't execute
`M-RET`	execute command under cursor, skip to next
`C-c C-o`	execute command under cursor, skip to next
`C-x C-w`	save the current transcript as ...
`C-x C-s`	save the current transcript

Editing Objects

`C-c C-d`	dump an object into a buffer
`C-c C-l`	load the contents of a buffer
`TAB`	indent the line under the cursor
`M-C-q M-C-q`	indent the expression under the cursor

Sending Code to ESS

`C-c C-j`	send current line to the ESS process
`C-c M-j`	as above, then return to the process buffer
`C-c C-f`	send the current function to the ESS process
`C-c M-f`	as above, then return to the process buffer
`C-c C-r`	send the selected region to the ESS process
`C-c M-r`	as above, then return to the process buffer
`C-c C-b`	send the edit buffer to the ESS process
`C-c M-b`	as above, then return to the process buffer
`C-c C-n`	send line to the ESS process, step to next line

Chapter 7

Using GGobi with R (Rggobi)

7.1 Introduction

If you are often involved with multivariate data analysis, you will be familiar with terminologies like *Principle Component Analysis*, *Factor Analysis*, *Clustering*...and so on. However most of these techniques are trying to plot a multidimensional data set onto a 2 or 3 dimensional space, and hence we may lose information that are hidden in other dimensions.

If you have been using a UNIX-based system, you may be familiar, or may have heard of the free program called **XGobi**[1], http://www.research.att.com/areas/stat/xgobi/index.html#xgobi. Unfortunately setting up XGobi with Windows environment is a non-trivial experience, and involves you have a proper X-Server running on top of your Windows.

Recently there has been a new development called **GGobi**, http://www.ggobi.org/. At the time I write this chapter, GGobi is stable enough for a *normal user*.

7.2 Getting GGobi

Go the GGobi's web site, http://www.ggobi.org/. Under **Table of Contents** click on **Software Downloads**.

Download both the **Self-installing Windows binary** and **Rggobi for Windows**. The former is a stand-alone executable programme, which is GGobi itself. The later is the R plug-in, which if installed properly, one call call GGobi within R.

7.3 Installing GGobi

Once you downloaded the two files, you can install GGobi itself by double clicking on **ggobi.exe**. Follow the instructions and choose a place you wish it to be installed to.

[1] Allows viewing high-dimensional data.

Now you need to add the GGobi root directory (the directory that contains the installed *ggobi.exe*), into your PATH. Refer to Section 4.2 if you are unsure how to do this.

Installing the R package (RggobiWin_0.4-1.zip) is easy, simply follow the instructions in Section 2.2.

Once you get these steps done, you will need to add the directory that contains the Rggobi's dll files into your PATH. These files, under normal circumstances, are stored in `$R_HOME\library\Rggobi\libs`. Put the absolute address into your PATH, or copy these files into a directory that is already in your PATH variable.

7.4 Running Rggobi

In R, type:

```
library(Rggobi)
```

to load the library in (and check if you have installed properly).

Basic help files can be found by typing:

```
?ggobi
```

More detailed documentations can be found from http://www.ggobi.org/Documentation.html

For beginner, I would recommend *Manual (pdf)* and *Using R-GGobi Link (pdf)*.

Index