R Programming

Installing and Creating Packages

R Packages

R consists of a base level of functionality together with a set of contributed libraries which provide extended capabilities.

The key idea is that of a *package* which provides a related set of software components, documentation and data sets.

Packages can be installed into R, or into a user's account.

You need special rights to install into a system version of R.

Installing Your Own Packages

You can install R packages using the `install.packages` command.

```
> install.packages("xtable")
```

If you are not using your own personally installed version of R you will not be able to install the package where R normally keeps them.

On Linux, this will be detected automagically and a personal package space will be set up for you (assuming that you respond "`y`" when offered the chance to set one up.

You will also need to specify which archive to retrieve the package from.

Obtaining R Packages

There are currently over 3000 packages available in the CRAN archives.

The major CRAN archives of interest to us are the CRAN master site in Austria

`cran.r-project.org`

and the local mirror site

`cran.stat.auckland.ac.nz`

As part of the package installation process you will be asked to specify which archive to download from.

You should specify `cran.stat.auckland.ac.nz`.

Downloading Package Sources Manually

To get a package from CRAN, visit the CRAN site using a web browser.

Click on the *Packages* link.

Click on the link to the package you are interested in.

Download the package source using *shift-click*.

The package called `foo` will be downloaded as something like `foo_version.tar.gz`, where `version` is the current version number.

Examining Package Sources

Supposing that you have downloading `foo_version.tar.gz`, you can unpack the sources using the shell command:

```
$ tar zxf foo_version.tar.gz
```

This will create a directory `foo-version` which contains the source code, documentation and datasets for the packages.

You can cd to the directory and examine the files using commandline commands or you can use the graphical filesystem browser to do the same thing.

Installing Downloaded Packages

Type the command:

```
$ R CMD INSTALL foo_version.tar.gz
```

R will then install the package, taking care of any technicalities for you, including:

1. Compilation of Fortran/C/C++ code
2. Creation of code libraries
3. Creation of documentation
4. Installation

Using Installed Packages

When R is running, simply type:

```
> library(foo)
```

This adds the R functions in the library to the search path.

You can now use the functions and datasets in the package and inspect the documentation.

The Search Path

R maintains a list of places which is searches for the values of variables (including both data sets and functions).

The contents of the search list can be obtained by using the `search` function.

```
> search()
 [1] ".GlobalEnv"         "package:foo"
 [3] "package:stats"      "package:graphics"
 [5] "package:grDevices" "package:utils"
 [7] "package:datasets"   "package:methods"
 [9] "Autoloads"          "package:base"
```

Packages are attached after the global environment, which is where variables created by top-level assignments are created.

Examining Package Contents

The contents of attached packages can be obtained by using the `objects` function (or using the older `ls` function).

The commands

```
> objects(pos=2)
```

or

```
> objects("package:foo")
```

will return a vector containing the names of the variables providing values in the package.

Creating Your Own Packages

The process of creating your own packages is fully described in the document *Writing R Extensions* which you can find in the Manuals area of CRAN.

The manual tends to be technical and can be tough going for newbies.

Fortunately, for packages which do not include compiled Fortran/C/C++ code, the process is relatively easy.

Step 1: Write the Functions

This is the time-consuming part of package development. To illustrate the process we'll look at a simplified version of the dotchart functions and data written for assignment 6.

We'll assume the functions and data are already in place in the R session we are running.

```
> objects()
[1] "dotchart" "llines"   "speakers"
```

Step 2: Create a Package Skeleton

The R function `package.skeleton` can be used to create a basic package structure.

```
> package.skeleton("dotchart",
          c("dotchart", "llines", "speakers"))
Creating directories ...
Creating DESCRIPTION ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
Further steps are described in
    './dotchart/Read-and-delete-me'.
```

The first argument to `package.skeleton` is the name of the package and the second gives the names of the objects to be included in the package.

Package Structure

Running `package.skeleton` creates a directory called `dotchart`.
Using the shell command `ls`, we can peek inside the `dotchart` directory.

```
$ ls dotchart
DESCRIPTION  R  Read-and-delete-me  data  man
```

The file `Read-and-delete-me` contains instructions on how to complete the process of building the package.

`R`, `data` and `man` are directories which contain the package's functions, data and documentation.

The `DESCRIPTION` file contains a package description which must be filled in.

The DESCRIPTION File

The DESCRIPTION file contains an outline which needs to be modified to produce a complete description of the package.

```
Package: dotchart
Type: Package
Title: What the package does (short line)
Version: 1.0
Date: 2009-05-26
Author: Who wrote it
Maintainer: Who to complain to <yourfault@somewhere.ne
Description: More about what it does
             (maybe more than one line)
License: What license is it under?
LazyLoad: yes
```

The DESCRIPTION File

An completed version of the DESCRIPTION file might appear as follows.

```
Package: dotchart
Type: Package
Title: Plotting Dotcharts
Version: 1.0
Date: 2009-05-26
Author: Ross Ihaka
Maintainer: Ross Ihaka <ihaka@r-project.org>
Description: This is a small demonstration package
        which gives an example of how to produce
        an R package.
License: GPL2
LazyLoad: yes
```

Writing Documentation

The files in the package `man` directory are documentation files for the functions and datasets in the package.

```
$ ls dotchart/man
dotchart-package.Rd  llines.Rd
dotchart.Rd          speakers.Rd
```

In this case, there are manual entries for the `dotchart` and `llines` functions, the `speakers` data set and the package as a whole.

Initially these files contain just a skeleton of the documentation and an indication of how it should be filled in.

The *emacs* ESS package contains support for editing these files.

Documentation Format

The lines in the R documentation (Rd) files looks very much like LaTeX. For example, one of the first sections in the file appears as follows.

```
\title{
%%  ~~function to do ... ~~
}
```

This is the title which appears at the top of the help file for this function.

In this case an appropriate title section might be the following.

```
\title{
The height of lines of text
}
```

Documentation Format

The next section the documentation file is the description section.

A comment in the section indicates that it should give a short description of what the function does.

A suitable section might be

```
\description{
This function computes the height of \code{x}
lines of text in units suitable for the
\code{height} and \code{width} arguments of
the layout function.
}
```

Notice the use of the `\code{}` macros to indicate that these are fragments of R code.

Completing the Documentation

In general all parts of the documentation files which contain comments of the form

```
%%  ~Describe the value returned
```

indicate the kind of thing which is supposed to appear at that point in the file.

Just replace the comment by the appropriate text.

The section “Marking Text” in the *Writing R Extensions* shows you what kind of markups are supported.

Installing the Package

Once you have completed the `DESCRIPTION` file and the various documentation files in the package, you can try installing it.

This is done by typing the (shell) command

```
$ R CMD INSTALL dotchart
```

from the directory which contains the top-level package directory.

Assuming the package installs correctly, you can now test it by starting R and typing the expression

```
> library(dotchart)
```

Building a Package Source Distribution

Once you've tested your package, you may want to prepare it for distribution.

This is done by issuing the command

```
$ R CMD build dotchart
```

in the directory containing the top-level package directory.

This should produce a file with a name of the form `dotchart-1.0.tar.gz` which is ready to be installed using a command such as

```
$ R CMD INSTALL dotchart-1.0.tar.gz
```

Packages for Windows

Source packages can be installed under the Linux/Unix and MacOS operating systems as described previously.

The situation under Windows is rather different.

Windows provides a very primitive development environment and to install packages from source requires the installation of several hundred megabytes of additional compilers and other utilities.

As an alternative it is possible to transfer installed packages from Linux to Windows (provided that they don't contain compiled C/C++/Fortran).

Transfering Packages from Linux to Windows

Use the following steps:

1. Install the package under Linux.
2. Go to the private directory where you install your packages. (This is the what the shell variable `R_LIBS` points to.)
3. To create a Windows version of the `dotchart` library issue the command

   ```
   $ zip dotchart.zip dotchart
   ```

4. The resulting file `dotchart.zip` can be transfered to Windows and installed there using the menus in the GUI version of R.

Writing Package Vignettes

A vignette is an Sweave document which provides additional information about an R package.

They are typically used to provide a longer description of how to use a package than a simple manual entry.

To provide vignettes you will need to make a subdirectory `inst/doc` inside the top-level package directory.

Place the Sweave source for the vignette in the `inst/doc`.

When the `R CMD build` command is executed, the vignette source is turned into a PDF file which is included in the package distribution. When that package is installed, the vignette is installed too.

Viewing Vignettes

The R command

```
> vignette()
```

lists the names of all the vignettes it is possible to view.

If there is a vignette called `dotcharts`, it can be viewed by typing the R command

```
> vignette("dotcharts")
```