A SHORT NOTE ON BUILDING R PACKAGES ON A MAC

IVY WANG AND HANNA JANKOWSKI

On Mac OS, the way to create a R package is similar to the steps on UNIX by using Applications → Utilities → Terminal.

Set-up.

If you haven't already, download and install TeX. You will also need to add `/usr/texbin` to the environment variable `PATH`. This is done by changing the `.profile` file in your home directory using an editor such as TextEdit or vi. Next, in a terminal window, run `. ./.profile` to make the shell reload the `.profile`. Type `echo $PATH` to double check that your changes have been incorporated.

In a terminal window, use `cd` to go to your home directory. This should look something like `/Users/user_name`. Next, create two directories, named `myrlibrary` and `myrpackages` (using the command `mkdir myrpackages`). `myrpackages` will contain the source files of your package, while `myrlibrary` will be where the package is built. Using a text editor of your choice (such as vi or emacs) create a file named `.Rprofile` which contains the single line `.libPaths("/Users/user_name/myrlibrary")`. This specifies that the directory `/Users/user_name/myrlibrary` is where your packages will be located.

Creating the Package Structure.

Go to your packages directory (`cd myrpackages`), and create a directory for your package (e.g. `mkdir testpac`, where `testpac` is the name of your package.). If necessary, change the permissions of this directory by typing `drwx r-x r-x`. This is so that the directory is accessible: `chmod 755 testpac`. Use `ls -l` to check the permissions.

In the directory `testpac` create a number of directories containing your R code, manuals, data etc. We only outline the necessary directories and files here. For a good description of what is needed, we direct the reader to [R], page 6, section **Create the directory tree.**

- Create 2 directories: `man` and `R`. The man directory will contain your R `.Rd` help files, and the R directory will contain the code files for your R functions (generally `.R`). If you also have datasets, create a directory in `testpac` called

Date: July 28, 2008.

`data` to put them in. If you also have compiled code files (C or Fortran files), create another directory in `testpac` named `src` to put them in.

- Put the datasets, R code and compiled code in the relevant directories.
- In the `man` directory, and create a `.Rd` file for each of your functions or datasets in the package (R code, C code and Fortran code). R has a function called `prompt` which can create a template manual file for you. This is much easier than working from scratch. To use `prompt`, in R, first source the function, say `testfun`, and type the command `prompt(testfun, file=testfun.Rd)` to create the template file in your current working directory. This file will have all of the desired sections, so you can simply edit as needed and copy over to the `man` directory.
- If you have C or Fortran code files (`.c` or `.f` files), create the file `firstlib.R` and place it in your `testpac/R` directory. The file should contain the following lines:

```
.First.lib <- function(lib, pkg)
{
library.dynam("testpac", pgk, lib)
cat("testpac 0.1-1 loaded\n")
}
```

  This file tells R to load in the compiled code contained in the `src` directory.
- Go to the main roots directory and create a file called `DESCRIPTION`. This file should contain the following (appropriately modified) lines:

```
Package:  testpac
Version:  0.0
Date:  2008-06-13
Title:  Testing 1 2 3
Author:  Ivy Wang, Hanna Jankowski
Maintainer:  Ivy Wang <ivywang@yorku.ca>
Depends:  R (>= 2.4)
Description:  This is a test package.
License:  GPL (version 2 or later)
```

Now the structure of the package is done. You will next need to make sure that all of your files and directories are accessible. Change the permissions (if necessary) of all created directories in `testpac` to `rwx r-x r-x (chmod 755 *)`. All files need to be readable; this can be changed by using `chmod 644 *` within each subdirectory.

Building the Package.

First, you will need to build the package. Go to the `myrpackages` directory and type `R CMD BUILD testpac`. This command should output a lot of information to the screen, and also create a zipped file of your package `testpac_0.1-1.tar.gz` in your `myrpackages` directory. Note that the underscored number will match whatever version you specified in your `DESCRIPTION` file. This comes in handy if you want to make changes to your files or add extra functions etc, so you can keep track of what you are doing and how each version changes. You have now (hopefully successfully) created an R package. You can also use `R CMD CHECK testpac` to locate any problems in the package structure.

Assuming that your package has been successully created, you will next want to install it. To do this, go into the `myrpackages` directory (where your package is) and type `R CMD INSTALL -l /Users/user_name/myrlibrary testpac_0.1-1.tar.gz`. You can check the `DESCRIPTION` file in the `/myrlibrary/testpac` directory for a built line to see if it has installed properly.

To see if the package works, start R from whichever directory you want, and type library(testpac).

Disclaimer. This note is a simple write-up of what worked for us. Please consult the CRAN website or an R expert for further details.

References

[CRAN] The Comprehensive R Archive Network. http://cran.r-project.org/

[R] Rossi, Peter. *Making R Packages Under Windows: A Tutorial.* Available from http://faculty.chicagogsb.edu/peter.rossi/research/bayes%20book/bayesm/Making%20R%20Packages%20Under%20Windows.pdf

Department of Mathematics and Statistics, York University
e-mail: ivywang@yorku.ca, hkj@mathstat.yorku.ca