Building R Packages in Windows

Eric M Reyes
(material from Friedrich Leisch)

Department of Statistics
North Carolina State University

NCSU Bayesian Seminar Series
01 September 2011
Why an R Package? (or SAS Macro for that Matter)
Why Package Your Code?

1. Makes your work accessible.
2. Contributes to general statistical knowledge.
3. Makes the output pretty.
4. Really...you are halfway there already!
Why Package Your Code?

1. Makes your work accessible.
Why Package Your Code?

1. Makes your work accessible.
2. Contributes to general statistical knowledge.
Why Package Your Code?

1. Makes your work accessible.
2. Contributes to general statistical knowledge.
3. Makes the output pretty.
Why Package Your Code?

1. Makes your work accessible.
2. Contributes to general statistical knowledge.
3. Makes the output pretty.
4. Really...you are halfway there already!
R Coding
R Functions

- R functions process input (usually data) and return a set of results.
- The “results” can be just about anything; so, think outside the box.
S3 vs. S4 Programming

**S3 Programming**
- Not really “object oriented,” but average user will not know the difference.
- Methods are really clever naming conventions.
- Usually dependent upon lists (that is, use $ to access “elements”).
- Currently the majority of R functions and packages.

**S4 Programming**
- Truly object oriented.
- Methods can perform automatic checks to prevent mistakes.
- Dependent on S4 objects (that is, use @ to access “slots”).
- The direction serious R programmers are moving.
Methods

- These functions process objects.
- Most commonly used to make the results look nice.
Creating R Package
What You Need

1. Your code gathered into one place.
2. Latest version of R.
3. (Windows Only) Ability to alter the PATH.
Steps to Creating R Package

1. (Windows Only, First Time) Change the PATH.
2. Place all your code in one place.
3. Set the working directory in R to the location of your files.
4. Create the package skeleton using `package.skeleton()`.
5. Alter the necessary files, especially help pages.
6. Check the package using `R CMD check`.
7. Build the package using `R CMD build`.
8. Install the package using `R CMD INSTALL`.
Changing the PATH in Windows

- **Note:** You CANNOT do this on the department machines.

1. **Control Panel** – > **System** – > **Advanced** – > **Environment Variables**

2. `C:\Rtools\bin; C:\Rtools\perl\bin; C:\Rtools\MinGW\bin; C:\Program files\R\R-2.12.1\bin\x64;`

3. This is dependent upon the version of R you are running and whether you have a 32-bit or 64-bit build.
While these steps are not required, they make things simple.

```r
> setwd("/path/to/files/")
```
Create Package Skeleton

1. `package.skeleton(name="NameOfPackage", code_files=c("file1.R", "file2.R"))`

2. When finished, directions pop up in R console, and a folder has been created with the name “NameOfPackage” and several files have been created within it.
Alter the Necessary Files

1. Change the DESCRIPTION file to have accurate information.
2. Change the help files for the package and each function (under the man folder).
3. Each file outlines what should be added and how.
   - Make the help files informative!
Checking and Building the Package

1. Open a command prompt and move to the directory with your package skeleton.

2. R CMD check NameOfPackage

3. Package must pass all checks to make it to CRAN (see the R documentation “Writing R Extensions” for tons of information).

4. R CMD build NameOfPackage

5. Now the package has a file that you can hand out to people.

6. R CMD INSTALL -l /path/to/library NameOfPackage

7. Now the package is installed in your library.

8. Alternatively in R use
   
   install.packages("NameOfPackage.tar.gz", repos=NULL, type="source")
Leisch F.

Creating R Packages: A Tutorial.