Lab: Writing packages in R

Seth Falcon Robert Gentleman

January 18, 2006

1 Overview

In this lab you will build an R package called *DNAhelperUSER* from scratch (where USER is your login for your computer). The goal is to produce a working package, complete with proper documentation, that passes the `R CMD check` utility without any errors or warnings being reported.

If you have never built an R package from scratch before, you should be happy if you complete Section 5. Section 6 is for experienced developers interested in exploring how namespaces work with R packages. If you are comfortable building R packages, you should aim to move quickly so you have some time to work through Section 6.

2 A function for the package

To start with, the *DNAhelperUSER* package will contain a single function, `dnaComplement` that returns the complement of a given DNA sequence. For example, the call `dnaComplement("ACGT")` would return `"TGCA"`.

To implement `dnaComplement`, observe the following "vectorization" of `chartr`.

```
> dnaSeq <- "AAGGTTCC"
> comSeq <- "TTCCAAGG"
> ans <- chartr("ATGC", "TACG", dnaSeq)
> ans

[1] "TTCCAAGG"

> comSeq == ans

[1] TRUE
```

Exercise 1
Using the `chartr` example, write a `dnaComplement` function.

3 Package skeleton

R provides a function `package.skeleton` that according to the man page

> ... automates some of the setup for a new source package. It creates directories, saves functions and data to appropriate places, and creates skeleton help files and `README` files describing further steps in packaging.

Here we call `package.skeleton` to create the initial structure and templates for the *DNAhelperUSER* package.

```
> user <- Sys.info()["user"]
> pkgName <- paste("DNAhelper", user, sep = "")
> if (interactive()) {
+     package.skeleton(name = pkgName, list = "dnaComplement")
+ }
```

Exercise 2
Explore the directory created by `package.skeleton`*. Read the* `README` *files and then delete them.*

4 Manual pages: Editing Rd files

Inside the `DNAhelperUSER/man` subdirectory, you should find a template for the manual page documenting the `dnaComplement` function with the name `dnaComplement.Rd`. This file is in Rd (R documentation) format which is a LaTeX like markup that is documented in the *Writing R Extensions* manual.

Exercise 3
Use a text editor to complete the manual page for the `dnaComplement` *function. Add an example that demonstrates the use of the function.*

5 Building *DNAhelperUSER*

At a shell command prompt, build the *DNAhelperUSER* package like this:

```
R CMD build DNAhelperUSER
```

If you see any warning or error messages, try to correct them. Ask for help if you get stuck or are having trouble interpreting the warning and error messages.

Next, try installing the package:

```
R CMD INSTALL DNAhelperUSER
```

Again, if you see any warning or error messages, you will need to address them before continuing.

Finally, run check on the package and fix any errors or warnings that are reported:

```
R CMD check DNAhelperUSER_x.y.z.tar.gz
```

You can also run `check` on the package source directory:

```
R CMD check DNAhelperUSER/
```

Exercise 4
Cleanup the DNAhelperUSER package until it passes check without warnings and without errors.

Exercise 5
Install the DNAhelperUSER package, load it and try out the `dnaComplement` function. Try asking for help on the function and verify that your manual page is available.

Exercise 6
The example sections of all Rd files in the `man` subdirectory are executed during `R CMD check`. Modify the example in the manual page for `dnaComplement` by adding a call to `stop`. Rerun the check process; the example should fail.

6 Demonstrating namespace issues

In this section we will explore package namespaces. To start with, let's demonstrate how easy it is to break packages without a `NAMESPACE` file defined.

Exercise 7
Load the DNAhelperUSER package and then try redefining the `chartr` function like this:

```
chartr <- function(...) cat("This is my version of chartr\n")
```

Does the `dnaComplement` function still work?

Name collisions can be nasty. Adding a namespace to a package is one thing that a developer can do to make it more difficult for users to mess things up.

Advantages of a namespace for a package:

1. Your package won't be broken by "helper" functions that your users define in the global environment.

2. Your package won't be bothered by functions in other packages with the same name.

3. A namespace gives you the ability to clearly specify which functions are part of the public interface and which are private. R CMD check won't bother you for documentation on non-exported functions, although it is often still useful to give them some documentation.

There are some disadvantages:

1. You have to maintain the NAMESPACE file
2. It is less convenient to debug/develop packages with namespaces because you have to run `R CMD INSTALL` to be sure everything gets updated in the namespace properly. Although less convenient, you will be more certain that the behavior is really in your package and not a result of things hiding in your global environment.

Exercise 8
Add a NAMESPACE file to the package and verify that it protects against redefinition of `chartr`.