Data Handling

T. Eiting, M. Rosario, C.-Y. Kuo

09.14.2012

1 Introduction

1.1 Welcome

This document covers how to get up and running with R (very briefly) and the basics of starting an R session, bringing in data, assigning variables, and other basic R syntax. If you are following along, we recommend that you type in the code yourself, so that you become familiar with writing R code rather than relying on copying and pasting.

1.2 Getting & Installing R

If you do not already have the R software, please download it from CRAN at: http://www.r-project.org/. To improve use-ability, you may wish to use a script editor. We recommend using Rstudio. Rstudio is nice because it generally integrates seamlessly with R without much setup required.

2 Data Handling

2.1 Setting Up Your Session

When you first start a new R session or a new script, you should make sure to clear your working memory. This will ensure that you do not have any data, objects, variables, etc. in your workspace. This is important because, as you write more and more code in a given session, you do not want to wind up using the same variable name more than once or to over-write earlier variables whose values you were hoping to keep in working memory. This will become more apparent as your experience with R grows.

```
rm(list = ls())
```

To double-check that you have cleared the list, you may just type the following.

```
ls()
```

Next we want to make sure that we are using the correct directory to read in files and to write new files to. We can do this with the next chunk of code.

```
# set your working directory
setwd("D:/R")

# check to make sure your working directory is correct
getwd()
```

Now that we have our working directory assigned correctly, let's begin to setup our workspace with packages we will need. Packages are extensions of the basic R code that provide additional functionality. Usually each package is united around a common theme. For example, the vegan package provides many

functions designed for community ecology. Consequently, it is a good package to use when you need to carry out various multivariate statistical techniques. You can see the full list of packages here.

You (should) only need to install packages one time. Here's how.

```
# you will need to set the CRAN mirror
install.packages("vegan")
```

To actually use a package in your R session, you will need to load it into your working memory.

```
# loads package into your current session
library("vegan")

# also loads package - designed for use within functions
require("vegan")
```

2.2 Loading and Manipulating Data

Now let's move on to bringing in and manipulating data. The code you use to bring in files varies slightly depending upon what type of file you're going to import. To import a *.txt file in your current working directory, use the following.

```
geospiza <- read.table("geospiza.txt", header = T)
```

The read.table syntax allows you to import many different file types. For example, we can also read in a *.csv file with this function.

```
anolis <- read.table("anolis.missing.csv", header = T, sep = ",",
    na.strings = "")
```

To ensure this last file was read in correctly, let's use the following code to print out information about our data table. The print out is useful for checking that all columns and rows were imported, as well as to take note of any assumptions R has made about the data in the import process.

```
str(anolis)

## 'data.frame': 23 obs. of  4 variables:
##  $ species : Factor w/ 23 levels "aln","alu","an",..: 17 10 5 1 18 13 7 20 21 19 ...
##  $ logSVL  : num  3.63 5.05 4.27 4.02 3.84 ...
##  $ logSSD  : num  -0.00512 0.08454 0.24703 0.24837 0.09844 ...
##  $ ecomorph: Factor w/ 6 levels "crown-giant",..: 6 1 4 4 2 6 1 1 5 2 ...
```

We could also use the following syntax to examine the first few lines of the imported data.

```
head(anolis)

##   species logSVL   logSSD    ecomorph
## 1      oc  3.630 -0.00512        twig
## 2      eq  5.050  0.08454 crown-giant
## 3      co  4.272  0.24703 trunk-crown
## 4     aln  4.024  0.24837 trunk-crown
## 5      ol  3.839  0.09844  grass-bush
## 6      in  3.733  0.06137        twig
```

We can use slightly different syntax to import the anolis file. This function is basically equivalent to the read.table() function, but assumes the data is in *.csv format.

```
anolis2 <- read.csv("anolis.missing.csv", na.strings = "")
```

To double-check that this performed as well as the previous syntax to read in the anolis file, use the following syntax again.

```
str(anolis2)

## 'data.frame': 23 obs. of  4 variables:
##  $ species : Factor w/ 23 levels "aln","alu","an",..: 17 10 5 1 18 13 7 20 21 19 ...
##  $ logSVL  : num  3.63 5.05 4.27 4.02 3.84 ...
##  $ logSSD  : num  -0.00512 0.08454 0.24703 0.24837 0.09844 ...
##  $ ecomorph: Factor w/ 6 levels "crown-giant",..: 6 1 4 4 2 6 1 1 5 2 ...
```

If you have any questions about the options to include in your syntax, just access the help file as follows.

```
# type in a question mark followed by the function name for
# documentation
`?`(read.csv)
```

You will have noticed that we have been using arrows to assign the data we read in to variable names. You can also use equal signs. Run the following code to prove it to yourself.

```
anolis.new <- read.table("anolis.missing.csv", header = T,
    sep = ",", na.strings = "")
head(anolis.new)

##   species logSVL   logSSD    ecomorph
## 1      oc  3.630 -0.00512        twig
## 2      eq  5.050  0.08454 crown-giant
## 3      co  4.272  0.24703 trunk-crown
## 4     aln  4.024  0.24837 trunk-crown
## 5      ol  3.839  0.09844  grass-bush
## 6      in  3.733  0.06137        twig
```

Note that the anolis.new file is thus the same as the original anolis file. We used an equal sign read in this file, as opposed to the arrow. Otherwise the syntax was the same. That said, use the arrow syntax whenever possible, because in some very old versions of R (and old packages), the arrow and equal-sign syntax are actually not equitable.

Stay tuned for our next document that covers more about data handling and manipulation. Happy R-ing!