STA122 Lab Session # 3: Writing functions in R

Course Instructor: Prof. Merlise Clyde
Teaching Assistant: Debdeep Pati (dp55@stat.duke.edu)

January 27, 2009

1 Lists

An R list is an object consisting of an ordered collection of objects known as its components. There is no particular need for the components to be of the same mode or type, and, for example, a list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, and so on. Here is a simple example of how to make a list:

```
> Lst <- list(name="Fred", wife="Mary", no.children=3,
child.ages=c(4,7,9))
```

Components are always numbered and may always be referred to as such. Thus if Lst is the name of a list with four components, these may be individually referred to as `Lst[[1]],etc.`If further, `Lst[[4]]`is a vector subscripted array then `Lst[[4]][1]`is its first entry. If Lst is a list, then the function length(Lst) gives the number of (top level) components it has. Components of lists may also be named, and in this case the component may be referred to either by giving the component name as a character string in place of the number in double square brackets, or, more conveniently, by giving an expression of the form

```
> name$component_name
```

for the same thing. This is a very useful convention as it makes it easier to get the right component if you forget the number. So in the simple example given above: `Lst$name`is the same as `Lst[[1]]`and is the string "Fred", `Lst$wife`is the same as `Lst[[2]]`and is the string "Mary", `Lst$child.ages[1]`is the same as `Lst[[4]][1]`and is the number 4. When the concatenation function c() is given list arguments, the result is an object of mode list also, whose components are those of the argument lists joined together in sequence.

```
> list.ABC <- c(list.A, list.B, list.C)
```

Recall that with vector objects as arguments the concatenation function similarly joined together all arguments into a single vector structure. In this case all other attributes, such as dim attributes, are discarded.

2 Functions in R

- A function is defined by an assignment of the form

  ```
  > name <- function(arg_1, arg_2, ...) {expression}
  ```

The expression is an R expression, (usually a grouped expression), that uses the arguments, arg i, to calculate a value. The value of the expression is the value returned for the function. A call to the function then usually takes the form name($expr_1, expr_2, \ldots$) and may occur anywhere a function call is legitimate. 10.1 Simple examples As a first example, consider a function to calculate the two sample t-statistic, showing all the steps. This is an artificial example, of course, since there are other, simpler ways of achieving the same end. The function is defined as follows:

- ```
  > twosam <- function(y1, y2) {
  n1 <- length(y1);
  n2 <- length(y2)
  yb1 <- mean(y1);
  yb2 <- mean(y2)
  s1 <- var(y1);
  s2 <- var(y2)
  s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
  tst <- (yb1 - yb2)/sqrt(s*(1/n1 + 1/n2))
  tst }
  ```

  With this function defined, you could perform two sample t-tests as following:

  ```
  > x<-rnorm(100,0,1)
  > y<-rnorm(100,2,1)
  > data <-data.frame(sam1=x,sam2=y)
  > tstat <- twosam(data$sam1, data$sam2);
   tstat
  ```

- After writing the function and saving it under the name twosam.R, you can use the function later by typing

  ```
  > source("twosam.R")
  > twosam(data$sam1, data$sam2)
  ```

2.1 Example showing the use of ... in function argument

Consider the function hist.with.normal(rnorm(200)). The hist function has been slightly extended so that it now uses the empirical mean and standard deviation of the data instead of just 0 and 1.

```
> hist.with.normal <- function(x, xlab=deparse(substitute(x)),...)
{
   h <- hist(x, plot=F, ...)
   s <- sd(x)
   m <- mean(x)
   ylim <- range(0,h$density,dnorm(0,sd=s))
   hist(x, freq=F, ylim=ylim,
xlab=xlab, ...)
  curve(dnorm(x,m,s), add=T)
}
```
```

Notice the use of a default argument for xlab. If xlab is not specified, then it is obtained from this expression which evaluates to a character form of the expression given for x, that is, if you pass rnorm(100) for x, then the x label becomes rnorm(100). Notice also the use of a ... argument which collects any additional arguments and passes them on to hist in the two calls.

3 Example of saving Plots

```
vets = read.table("case0302.csv", header=T,sep=",") names(vets)
attach(vets)
# allows one to refer to variables by name within a dataframe
postscript("AO-hist.ps",horizontal=F,
           width=8,height=4,pointsize=14,fg="orange",bg="transparent")
  par(mfrow=c(1,2))
  hist(DIOXIN, xlab="Dioxin Concentration (ppt)",
       ylab="Density",prob=T, main="Default Number of Bins",col="orange")
  hist(DIOXIN, xlab="Dioxin Concentration (ppt)", ylab="Density",
       prob=T, main="50 Bins", breaks=50,col="orange")
dev.off()
```

4 Writing Output to a file

Having finished running a program, you can then output a data array or matrix to a text or other file type using the write.table() command. As an example, the code

```
write.table(mydata,file="filename.dat")
```

would write the data in the object mydata to the file `filename.dat`.This command is especially useful for outputting, say, draws from a posterior distribution drawn using Metropolis-Hastings algorithms.