Programming in R

A. Michelle Lawing
Ecosystem Science and Management
Texas A&M University
College Station, TX 77843
alawing@tamu.edu
Schedule for today

• My presentation
  – Review
  – New stuff

• Model Selection and Bootstrapping presentation

• Model Selection and Bootstrapping demonstration

• BREAK

• Model Selection and Bootstrapping tutorial
Review: Probability and Distribution

- R makes it easy to draw probability distributions to demonstrate statistical concepts

- Functions for each distribution follow a particular format

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dname()</td>
<td>Density or probability function</td>
</tr>
<tr>
<td>pname()</td>
<td>Cumulative density function</td>
</tr>
<tr>
<td>qname()</td>
<td>Quantile function</td>
</tr>
<tr>
<td>rname()</td>
<td>Random deviates</td>
</tr>
</tbody>
</table>
Review: Probability and Distribution

<table>
<thead>
<tr>
<th>Normal</th>
<th>Uniform</th>
<th>Student t</th>
</tr>
</thead>
<tbody>
<tr>
<td>dnorm(0)</td>
<td>dunif(0)</td>
<td>dt(0,20)</td>
</tr>
<tr>
<td>pnorm(0)</td>
<td>punif(0)</td>
<td>pt(0,20)</td>
</tr>
<tr>
<td>qnorm(0)</td>
<td>qunif(0)</td>
<td>qt(0,20)</td>
</tr>
<tr>
<td>rnorm(1)</td>
<td>runif(1)</td>
<td>rt(1,20)</td>
</tr>
</tbody>
</table>
Comparison of Distributions

Density vs. Probability

Distributions
- df=1
- df=3
- df=8
- df=30
- norm
- unif

x value

<table>
<thead>
<tr>
<th>Distributions</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>norm</td>
<td>3</td>
</tr>
<tr>
<td>unif</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
Comparison of t Distributions

Distributions

- df=1
- df=3
- df=8
- df=30
- norm
Normal Distribution

\[ P(-0.674 < x \text{ values} < 0.674) = 0.5 \]
Student T Distribution Two-Tailed

\[ P\left( -1.44 > x \text{ values} < 1.44 \right) = 0.15 \]
Programming in R
Getting Help

• There are various control functions that are used to establish the control flow

help(Control)
Control Flow

Description

These are the basic control-flow constructs of the R language. They function in much the same way as control statements in any Algol-like language. They are all reserved words.

Usage

if(cond) expr
if(cond) cons.expr else alt.expr

for(var in seq) expr
while(cond) expr
repeat expr
break
next

Arguments

cond  A length-one logical vector that is not NA. Conditions of length greater than one are accepted with a warning, but only the first element is used. Other types are coerced to logical if possible, ignoring any class.

var  A syntactical name for a variable.

seq  An expression evaluating to a vector (including a list and an expression) or to a pairlist or NULL. A factor value will be coerced to a character vector.

expr, cons.expr, An expression in a formal sense. This is either a simple expression or a so called compound expression, usually of the form { expr1 alt.expr ; expr2 }.

Remarks

break breaks out of a for, while or repeat loop; control is transferred to the first statement outside the inner-most loop. next halts the processing of the current iteration and advances the looping index. Both break and next apply only to the innermost of nested loops.

Note that it is a common mistake to forget to put braces ({ ... }) around your statements, e.g., after if(...) or for(....). In particular, you should not have a newline between 1 and else to avoid a syntax error in entering a if...else construct at the keyboard or via source. For that reason, one
Executing a file

• A set of commands can be saved in a file and executed from the command line in R

```r
source("http://michellelawing.info/file.R")

#source a file, url, or other connection

help(source)
```
Executing a file

```r
source("http://michellelawing.info/file.R", echo = T)
source("http://michellelawing.info/file.R", print.eval = T)
source("http://michellelawing.info/file.R", print.eval = F)
```

TIP: Open R from right clicking on your script file
Where is the file?

```r
source("notThere.R")  #sometimes R can’t find the file
#R searches the current director

getwd()       #returns the working directory

dir()         #lists the files in the wd

setwd()       #change wd
```
if statement

```r
x <- 1

if( x < 5 ) {
  x <- x + 1
}
```

#assign x a number

#if x is less than 5

#then increment by 1
else statement

```r
x <- 5

#assign x a number

if( x < 5 ) {
  x <- x + 1
  #if x is less than 5
  #then increment by 1
} else {
  x <- x - 1
  #in all other cases
  #decrement by 1
}
```
chain ifs together

```r
x <- 5  #assign x a number

if( x < 5) {
  x <- x + 1  #if x is less than 5
} else if (x == 5){
  x <- x + 1  #or if x equals 5
} else {
  x <- x - 1  #in all other cases
}
```
for statements

```r
for(i in 1:10) {
  print(i)
}

for(i in seq(0, 1, by = 0.1)) {
  print(i)
}

for(i in 10:5) {
  print(i)
}
```

#loop the temp var i 10 times
#print i each loop

#loop the i from 0 to 1 by 0.1
#print i each loop

#loop i from 10 to 5
#print i each loop
Random Walk

```r
#assign the number of times steps
tt <- 200

#assign the starting value
y <- 0

for(i in 2:tt) {
  #loop i from 1 to tt
  y[i] <- rnorm(1, mean = y[i - 1])
    #draw a random number
}

plot(1:tt, y, pch = 16, type = "l")
```

TIP: You don’t have to initialize arrays in R.
while statement

x <- 1
#assign a starting value
cntr <- 0
#set the counter to 0

while(x < 3) {
  x <- rnorm(1)
  #draw random number
  cntr <- cntr + 1
  #increment the counter
}

cntr
#print the number of draws
Other useful statements

• **repeat**
  - Same as while, except evaluates at least once
  - Need to define stopping point

• **break**
  - break stops the execution of the current loop

• **next**
  - next skips the remaining statements within a loop and starts the next looping cycle
functions

```r
foo <- function (a, b) {
  x <- runif(10, a, b)
  mean(x)
}

foo(-1, 1)  #execute function with min and max input
#the last expression in the function is returned
```

functions – return

foo <- function (a, b) {
  x <- runif(10, a, b)
  return(list(mu = mean(x), std = sd(x)))
}

#use return statement

foo(-1, 1) #returns list
functions – return multiple

```r
foo <- function (a, b) {
  x <- runif(10, a, b)
  return(list(a = a, b = b, mu = mean(x), std = sd(x)))
}
#use return statement

foo(-1, 1)  #returns list and input
```
functions – return plot

```r
foo <- function (a, b) {
  x <- runif(10, a, b)
  hist(x)
  return(list(a = a, b = b, mu = mean(x), std = sd(x)))
}
#use return statement

foo(-1, 1)  #returns list and input
```
Exercise - Programming

Create a function that plots a random walk by using `rnorm`. The function should read in the initial trait value and the number of time steps.