Statistics 452: Statistical Learning and Prediction

Chapter 2: Statistical Learning, R supplement

Brad McNeney

2017-09-01

What is R?

- ▶ R is an open-source environment for statistical computing and graphics.
- ▶ Started in the mid-1990's at Auckland University
- ▶ Now maintained by a team of experts called the R Development Core Team
- ▶ A "packages" system allows any user to bundle R code, data and examples together.
  - ▶ Load packages with `library()`
- ▶ R and R packages are distributed through the Comprehensive R Archive Network (CRAN).
- ▶ SFU has a CRAN mirror at `http://cran.stat.sfu.ca`

What does "environment" mean?

- R is a fully-functioning programming environment with all the usual constructs, such as
  - conditionals (if-then-else),
  - loops
  - user-defined functions.
- In addition there are built-in facilities for
  - data input, storage, manipulation, and output
  - optimization, matrix computation, etc.,
  - random number generation,
  - data analysis and graphics.
- "Base" R is good, but it is the package system that makes R great.

Starting R

- ▶ Start R by starting RStudio.
- ▶ The "Console" window is where you can type your commands.
- ▶ However, it is good practice to open an R script, type your commands in the script, and then submit the commands to the R console.
  - ▶ `Session -> Set Working Directory` to set the working directory
  - ▶ `File -> New File -> R Script` to open a new R script
  - ▶ type your commands into the script
  - ▶ put your cursor on the line you want to submit and hit `Ctrl-enter`
- ▶ Save your script for later use.
- ▶ More on the RStudio interface at `https://support.rstudio.com/hc/en-us/sections/200107586-Using-RStudio`

R Cheatsheets

- Use Google to find one that works for you.
- This one looks OK to me: `http://github.com/rstudio/cheatsheets/raw/master/source/pdfs/base-r.pdf`

R objects

- In R, data structures and functions are all referred to as "objects".
- Objects are created with the assignment operator <-; e.g., x <- 1.
  - The objects a user creates from the R console are contained in the user's workspace, called the global environment.
  - Use ls() to see a list of all objects in the workspace.
  - Use rm(x) to remove object x from the workspace.

R Data Structures

- Focus on four common data structures: atomic vectors, lists, matrices and data frames.
- Atomic vectors and lists are 1d, while matrices and data frames are 2d objects
- R has no true scalars; e.g., in x<-1, x is a vector of length one.
- R also has an array data structure for higher dimensional elements that we will not discuss.
- Use str() to see the structure of an object

Vectors

- Vectors can be either atomic or list

    - atomic vectors must be comprised entirely of logical, integer, double (numeric) or character elements
    - lists can be comprised of multiple data types

- Data vectors can be created with `c()` or `list()`:

```
avec <- c(50,200,77)
lvec <- list(50,200,77,c("grey","thin"))
```

Combining vectors

- ▶ Use c() to combine vectors

```r
c(avec,c(100,101))
```

```
## [1]  50 200  77 100 101
```

```r
c(lvec,TRUE)
```

```
## [[1]]
## [1] 50
##
## [[2]]
## [1] 200
##
## [[3]]
## [1] 77
##
## [[4]]
## [1] "grey" "thin"
##
## [[5]]
## [1] TRUE
```

Factors

- ▶ The statistical concept of a factor is important in experimental design.
- ▶ Factors are implemented in R as atomic vectors with attributes `class` and `levels`:

```
trt <- factor(c("drug1","placebo","placebo","drug2"))
attributes(trt)
```

```
## $levels
## [1] "drug1"   "drug2"   "placebo"
##
## $class
## [1] "factor"
```

```
str(trt)
```

```
##  Factor w/ 3 levels "drug1","drug2",..: 1 3 3 2
```

- ▶ The levels are coded numerically (1, 2 and 3) with assigned labels ordered alphabetically ("drug1", "drug2" and "placebo")

Subsetting vectors and extracting elements

▶ Subset with [or by name:

```r
lvec[c(1,3)] # same as lvec[c("age","height")]
```

```
## [[1]]
## [1] 50
##
## [[2]]
## [1] 77
```

▶ Extract individual elements with [[, or $ for named objects:

```r
lvec[[4]]
```

```
## [1] "grey" "thin"
```

```r
lvec$hair
```

```
## NULL
```

Subsetting and assignment

- You can combine subsetting and assignment to change the value of vectors

```
avec
```

```
## [1]  50 200  77
```

```
avec[2] <- 210
avec
```

```
## [1]  50 210  77
```

Matrices and data frames

- Though both 2d objects, matrices and data frames are different enough that we will need to discuss them separately.
- The elements of a matrix must all be of the same type.
- Data frames are essentially lists where each list element has the same length. Thus data frames can include columns of varying type.

Matrices

- Matrices can be created with the `matrix()` function as in

```
A <- matrix(1:4,nrow=2,ncol=2)
A
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

- Here 1:4 is the same as c(1,2,3,4)
- The default is to read the data vector into the matrix column-by-column. To read row-by-row instead use the byrow=TRUE argument:

```
A <- matrix(1:4,nrow=2,ncol=2,byrow=TRUE)
A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

Combining matrices

- Combine matrices with rbind() and cbind():

```r
rbind(A,matrix(c(5,6),nrow=1,ncol=2))
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

```r
cbind(A,A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    1    2
## [2,]    3    4    3    4
```

Subsetting matrices

- ▶ Subset with [and a comma to separate rows from columns:

```
A[1,1]
```

```
## [1] 1
```

```
A[1,]
```

```
## [1] 1 2
```

```
A[,1]
```

```
## [1] 1 3
```

- ▶ When a subsetting operation leads to a vector, the dimension of the object is "dropped" from 2 to 1. To prevent this use drop=FALSE:

```
A[1,,drop=FALSE]
```

```
##      [,1] [,2]
## [1,]    1    2
```

Extracting elements from matrices

```
A[1,1]
```

```
## [1] 1
```

Data frames

- ▶ Data frames (class `data.frame`) are the usual way to store data in R.
  - ▶ Rows are intended to be observational units, columns variables
  - ▶ Implemented as a list (columns are list elements), but also behave like a matrix in terms of combining and subsetting.
- ▶ Create with `data.frame`:

```
set.seed(1)
n <- 4
x <- 1:n; y <- rnorm(n,mean=x,sd=1) # multiple commands separated by ;
dd <- data.frame(x=x,y=y) # like making a list
str(dd)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ x: int  1 2 3 4
##  $ y: num  0.374 2.184 2.164 5.595
```

Subsetting and combining data frames

- ▶ Can subset columns like a list:

```
dd$x
```

```
## [1] 1 2 3 4
```

- ▶ Can subset columns/rows and combine like matrices; e.g.,

```
dd[1:2,]
```

```
##   x         y
## 1 1 0.3735462
## 2 2 2.1836433
```

```
zz = data.frame(z=runif(4))
cbind(dd,zz)
```

```
##   x         y          z
## 1 1 0.3735462 0.62911404
## 2 2 2.1836433 0.06178627
## 3 3 2.1643714 0.20597457
## 4 4 5.5952808 0.17655675
```

Logical operators

- ! is NOT
- & and && are AND, with & acting vector-wise and && acting on scalars
- | and || are OR, with | acting vector-wise and || acting on scalars
- Make sure you understand the following:

```
x <- c(TRUE,TRUE,FALSE); y <- c(FALSE,TRUE,TRUE)
!x ; x&y ; x&&y ; x|y ; x||y
```

```
## [1] FALSE FALSE  TRUE
```

```
## [1] FALSE  TRUE FALSE
```

```
## [1] FALSE
```

```
## [1] TRUE TRUE TRUE
```

```
## [1] TRUE
```

Relational operators

- Relational operators can be used to compare values in atomic vectors
    - See `help("Comparison")`
- `>` is greater than, `>=` is greater than or equal
- `<` is less than, `<=` is less than or equal
- `==` is equal and `!=` is not equal
- Make sure you understand the following:

```
x <- 1:3; y <- 3:1
x>y ; x>=y ; x<y ; x<=y ; x==y ; x!=y
```

```
## [1] FALSE FALSE  TRUE
```

```
## [1] FALSE  TRUE  TRUE
```

```
## [1]  TRUE FALSE FALSE
```

```
## [1]  TRUE  TRUE FALSE
```

```
## [1] FALSE  TRUE FALSE
```

```
## [1]  TRUE FALSE  TRUE
```

Subsetting vectors with logical expressions

- Can subset with logicals and [:

```
avec
```

```
## [1]  50 210  77
```

```
avec>100
```

```
## [1] FALSE  TRUE FALSE
```

```
avec[avec>100]
```

```
## [1] 210
```

```
avec[avec>50 & avec<100]
```

```
## [1] 77
```

Subsetting matrices with logical expressions

- ▶ Can also subset matrices, but results may not be as expected:

```
A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

```
A>1
```

```
##       [,1] [,2]
## [1,] FALSE TRUE
## [2,]  TRUE TRUE
```

```
A[A>1] # coerces to a vector
```

```
## [1] 3 2 4
```

Missing values

- ▶ R has a special data code for missing data: `NA`
- ▶ Test for and set missing values with `is.na()`

```
avec
```

```
## [1]  50 210  77
```

```
is.na(avec)
```

```
## [1] FALSE FALSE FALSE
```

```
is.na(avec) <- 2
avec
```

```
## [1] 50 NA 77
```

R functions: Example

```
f <- function(x) {
  return(x^2)
}
f
```

```
## function(x) {
##   return(x^2)
## }
```

Reading Data: Native format

- Use `save()` to save R objects to an "R Data" file.
    - `save.image()` is short-hand to save all objects in the workspace

```
x <- rnorm(100); y <- list(a=1,x=x)
save(x,y,file="test.RData") # Or .rda, or ...
```

- Load R Data files into the workspace with `load()`.

```
load("test.RData")
file.remove("test.RData")
```

```
## [1] TRUE
```

Reading Table Format Files

- `read.table()` is the main function for reading tabular data from plain-text files.
  - `read.csv()` and `read.delim()` are basically `read.table()` with defaults for reading comma- and tab- delimited files.
- `write.table()`, `write.csv()` and `write.delim()` are the analogous functions for writing tabular data

```
write.table(matrix(1:9,3,3),file="test.txt")
test <- read.table("test.txt")
file.remove("test.txt")
```

```
## [1] TRUE
```

```
test
```

```
##   V1 V2 V3
## 1  1  4  7
## 2  2  5  8
## 3  3  6  9
```

Reading files from a URL

- ► load(), read.table(), etc. can read data from a URL.

```
baseURL <- "http://people.stat.sfu.ca/~mcneney/Teaching/Stat452/"
rdURL <- url(paste0(baseURL,"Data/PorschePrice.rda"))
load(rdURL)
head(PorschePrice)
```

```
##    Price Age Mileage
## 1  69.4   3    21.5
## 2  56.9   3    43.0
## 3  49.9   2    19.9
## 4  47.4   4    36.0
## 5  42.9   4    44.0
## 6  36.9   6    49.8
```

```
csvURL <- url(paste0(baseURL,"Data/PorschePrice.csv"))
PorschePrice <- read.csv(csvURL)
```

stringsAsFactors

- ▶ Reading columns that include characters in as factors is controlled by a global option in your R session called `stringsAsFactors`, set to `TRUE` by default.
- ▶ If you want to set to `FALSE` for an R session type `options(stringsAsFactors = FALSE)` into the Console.
- ▶ An alternative is to over-ride the default in the call to `read.table()`:

```r
exURL <- url(paste0(baseURL,"Data/Ex1_1_4.txt"))
ex2 <- read.table(exURL,header=TRUE,sep="\t",
                  stringsAsFactors=FALSE)
```

Viewing Data: print(), View() and edit()

- ▶ print() prints R objects
  - ▶ This function is "generic", meaning that it will try to find the specific function to print specific objects (e.g., print.data.frame).
- ▶ View() launches a new window (or RStudio tab) to view a data frame and edit() launches a data editor.

Graphics

- "Base" graphics in R is good, but ggplot() is better.
- We could spend a lot of time on ggplot(), but will just learn what we need as we go.
- ggplot2 cheatsheet at [https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf]
- Wickham (2009) ggplot2: Elegant graphics for data analysis, Chapters 4 and 5.
- Chang (2012) R graphics cookbook. Available at [http://www.cookbook-r.com/Graphs/]