R Short Course Part 1
Introductory Topics
Session 1

Daniel Zhao, PhD

Sixia Chen, PhD

Department of Biostatistics and Epidemiology

College of Public Health, OUHSC

2/24/2020

The UNIVERSITY of OKLAHOMA
Health Sciences Center
College of Public Health

Outline

- Overview of the 5 sessions
- Pre-requisite requirements
- Session 1: Introduction to R
  - Lecture with hands-on practice (noon-1:30)
  - In-class exercise (1:30-2:00)

Overview of R Short Course Part 1

1. Introduction to R
2. Data Management
3. R Graphics
4. Statistical Calculation and Testing
5. Regression Analysis

Pre-requisite requirements

Sessions 1 – 4: One introductory undergraduate or graduate course in biostatistics.

Session 5: One introductory undergraduate or graduate course in biostatistics and at least one course in regression modeling.

Session 1: Introduction to R

1. Background
   a. Platforms
   b. History
   c. Books
   d. Online Resources
2. Getting Started
   a. R programming editor
   b. Running R code
   c. Help functions
3. Basic Data Structure
   a. Vectors
   b. Factors
   c. Arrays and Matrices
   d. Data frames
   e. Lists

Platforms to Use R

1. Installation on PC or Mac
   a. Windows: https://cran.r-project.org/bin/windows/base/
   b. Mac: https://cran.r-project.org/bin/macosx/
   c. UNIX system: Norman has a super computer
2. Cloud-based R using a web browser
   a. https://rstudio.cloud/. Free, simple to apply an account, can install packages
3. Microsoft R open
   a. https://mran.microsoft.com/open
   b. Multithreaded Performance

How is R used in workplace?

1. Me
   a. Primarily for statistical research, particularly run Monte Carlo simulations
   b. Statistical analysis if SAS does not have such procs. Eg, CART, Meta-Analysis
2. Others
   a. Academia: Use R with Knitr package to auto-generate reports with stat results including tables, figures, p-values, CIs, etc.
   b. Industry: Use R for Premarket Approval Application to Center for Devices and Radiological Health of the FDA

The History of R

- **S**: an interactive environment for data analysis developed at Bell Laboratories since 1976
- Exclusively licensed by AT&T/Lucent to Insightful Corporation, Seattle WA. Product name: "**S-plus**".
- **R** is a free implementation of the S language
- GNU General Public License (GPL)
  - can be used by anyone for any purpose
  - open source

R Books

1. Standard R installation comes with very good reference documents: R->Help->Manuals
2. Most S or Splus books
   - Modern Applied Statistics with S by Venables and Ripley
3. If you are SAS and/or SPSS users
   - R for SAS and SPSS users by Robert Muenchen
4. Many other R user books
   - https://www.r-project.org/doc/bib/R-books.html

R Online Resources

1. Search Engines with the right key words
2. UCLA resources to learn and use R
    1. http://www.ats.ucla.edu/stat/r/
3. http://stackoverflow.com/
4. CRAN Task Views allow you to browse packages by topic
    1. https://cran.r-project.org/web/views/
    2. Eg, click on cluster will get a list of all cluster analysis R packages

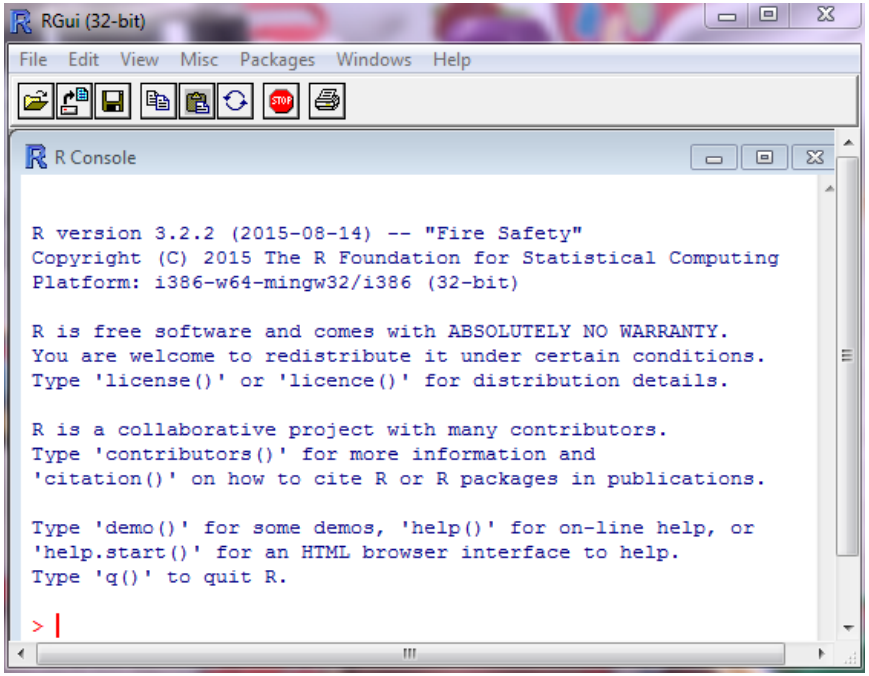
An Overview of R

R is an integrated software for data management, calculation, analysis, and graphics

1. data handling and storage facility
2. a suite of operators for calculations on arrays, in particular matrices
3. a large integrated collection of tools for data analysis
4. graphical facilities for data analysis and display
5. a well developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions, and input and output facilities.

Getting Started with R

1. Double click on R icon will bring up R GUI
2. Can type R commands in the R console, or
3. File – New Script, to open an R Editor, or
4. File – Open Script, to open an existing R script (similar to .sas)
5. R studio is a much better R editor



How to Run R Code

- For casual tasks, type R code after the prompt in R Console
- A more organized way to run R code is to open an R editor and type R code in the editor
- To run one line of code in the editor, do either of the following
  - Put cursor anywhere in the line, ctr-R
  - Put cursor anywhere in the line, right click anywhere in the R editor and choose “Run line or selection”
- To run multiple lines of code in the editor
  - Highlight the code you want to run
  - Right click anywhere in the R editor and choose “Run line or selection”
- Use up and down arrow to select previously run command.

Getting Help within R

- To get help on a specific function, say, mean

  > help(mean), or

  > ?mean

- To get help on functions with similar names,

  > ??mean

- To get help on a feature specified by special characters

  > help ("[[")

Naming Conventions for R objects

- R names can be used for variables or functions
- R is case sensitive, so a and A are different
- R names include letters, numbers, ‘.’, and ‘_’
- Must begin with a letter or ‘.’ , say x1, or .seed
- If a name starts with ‘.’ then the second character must not be digit (so .1a is wrong)
- Names are unlimited in length

Commands and Comments

- Commands are lines of R code
- Commands are separated by a semi-colon if they are on the same line, or by a new line
- Commands can be grouped together using braces {}, especially when define functions
- Comments start with a hashmark (#), everything to the end of the line is a comment
- To make multiple lines to be comments, you have to put a # at the beginning of each line

Set up a working directory

- Working directory
  - a folder on your computer that sets the default location of any files you read into R, or save out of R
- Get working directory
  - getwd()
- Set up a working directory
  - setwd("C:/Users/dzhao1/Desktop")
- The working directory is updated
  - getwd()

Save Output

- To divert output from the console to an external file, say record.txt

  > sink (“record.txt”)

- To restore the output to the console again

  > sink ()

An Example

```
x = 3+2
x
# To divert output from the console to an external file
sink ("record.txt")
x
# To restore the output to the console again
sink ()
x
```

Saving Objects and Commands

- The **workspace** is your current R working environment and includes any user-defined **objects** (vectors, matrices, data frames, lists, functions).
- To display the names of all objects in the workspace

  > objects()
- To remove certain objects

  > rm(x)
- At the end of each R session, when asked “save work space image”, click yes will
  - save all objects to a file called .RData
  - save all the command lines to a file called .Rhistory

Vectors and Assignment

- The simplest data structure in R is the numeric vector
- For example, to set up a vector x with 5 numbers, use any of the following R command

  > x = c(10, 5, 3, 6, 21)

  > x <- c(10, 5, 3, 6, 21)
- Use length function to get the length

  > length(x)
- What happens if we say?

  > y <- c(x, 0, x)

Generating Regular Sequences

- To generate a vector 1, 2,…, 30, use any of the following commands
  - > x = 1:30
  - > x1 = seq (from=1, to=30, by=1)
  - > x2 = seq (from=1, by=1, length=30)
- Another useful function is rep for replications
  - > y = rep (x, times=2)
    will put 2 copies of x end-to-end in y
  - > z = rep (x, each=2)
    will repeats each element of x two times before moving on to the next

Vector arithmetic

- Arithmetic operations on vectors are performed element by element
  > v = 2*x + 1
  > z = log (x) + sqrt (x)
- Common arithmetic functions: log, exp, sin, cos, tan, sqrt
- Other functions: min, max, range, length, sum, prod, mean, var, sd, sort
- These functions can be used on 2-dim matrices and data frames.

Logical Vectors

- The elements of a logical vector can have the values TRUE (T), FALSE (F), and NA
- Logical vectors are generated by conditions, eg

  > x = 1:4

  > y = (x > 2)

  the result is y = c(FALSE, FALSE, TRUE, TRUE)
- Logical operators:
  - <, <=, >, >=, == (equals), != (not equals)
  - & (and), | (or), ! (not)

Missing values

- Missing values (numeric or character) are represented by NA
- Function is.na gives a logical vector of the same size as x with value TRUE if and only if the corresponding element in x is NA

  > z <- c(1:3,NA); ind <- is.na(z)

- There is a second kind of "missing" values which are produced by numerical computation, the so-called Not a Number, NaN,

  > x = 0/0

- Similar to is.na, there is a function is.nan

Example on Missing Values

- Compute mean value

  > z <- c(1:3,NA)

  > mean(z) will return NA

- Remove missing values

  > mean(z, na.rm=TRUE)

  or

  > z1 = na.omit(z)

  > mean(z1)

Character vectors

- Character vector is a sequence of characters delimited by the double quote (or single quote)

  > x = c ("red ", "blue ")

- Character vectors may be concatenated into a vector by the c() function

  > y = c ("yellow ", "green"); z = c (x, y)

- The paste() function is used to create new character vectors

  > paste("red", "blue")

  > paste("red", "blue", sep="")

Index and Subset Vectors

1. A vector of positive integers (to be selected)

   > x = c (3, 5, NA)

   > y = x[1:2]
   The result is y=c(3, 5)

2. A vector of negative integers (to be dropped)

   > y = x[-3]
   The result is y=c(3, 5)

3. A logical vector: Values corresponding to TRUE in the index vector are selected and those corresponding to FALSE are omitted

   > z = !is.na(x)

   > y = x[z]
   The result is y=c(3, 5)

Types of Objects

The entities that R creates and manipulates are objects.

- Vectors
- Factors: Provide a way to handle categorical data
- Arrays/Matrices: multi-dim generalization of vectors
- Data Frames
  - matrix-like 2D data structures where the columns could be numerical and/or character variables
  - Most statistical tests are performed on data frames
- Lists
  - General form of vector whose elements can be vectors, matrices, data frames, lists
  - Provide a convenient way to return results of a stat computation
- Functions: provide a simple and convenient way to extend R

Factors

- A factor is a vector object used to specify a discrete grouping of the components of other vectors of the same length.
- An example on incomes of tax accounts from Australia states

```
> state = c("tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", "qld", "vic",
"nsw", "vic", "qld", "qld", "sa", "tas", "sa", "nt", "wa", "vic", "qld",
"nsw", "nsw", "wa", "sa", "act", "nsw", "vic", "vic", "act")
> statef = factor(state)
> levels (statef)
[1] "act" "nsw" "nt" "qld" "sa" "tas" "vic" "wa“
> incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, 61, 61,
61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46, 59, 46, 58, 43)
> incmeans = tapply(incomes, statef, mean)
```

Gives a mean vector with the components labeled by the levels

```
act     nsw     nt      qld     sa      tas     vic     wa
44.500 57.333 55.500 53.600 55.000 60.500 56.000 52.250
```

Arrays and Matrices (1)

- Arrays can be multi-dimentional and created by using array() function
- Focus on 2D matrices and matrix() function
- Create a matrix
  - > A = matrix(1:16, ncol=4, byrow=F)

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

- Subset a Matrix
  - > A[1,3]
  - > A[c(1,3),c(2,4)]
  - > A[1:3,2:4]
  - > A[1:2,]
  - > A[,1:2]
  - > A[-c(1,3),]
  - > A[-c(1,3),-c(1,3,4)]

Arrays and Matrices (2)

- Matrix multiplication (%*%)

  > x=matrix(1:4, ncol=2, byrow=T)

  > y=matrix(5:8, ncol=2, byrow=T)

  > z = x %*% y

- Binding matrices by rows (top-down)

  > z = rbind(x, y)

- Binding matrices by columns (left-right)

  > z = cbind(x, y)

- Functions on matrices: diag, solve, eigen, svd

Data Frames

- 2D rectangular data structures with rows as obs and columns as vectors/variables
- Different than matrices, column vectors of data frames can be numeric, logical, or character

Making Data Frames (1)

- If X and Y are two column vectors of the same length, then use Z = data.frame (X, Y)
  - mydata=data.frame(age=20:25 , weight=120:125, sex=c(rep("M",3),rep("F",3)))
- The following 3 variables do not exist: age, weight, sex
- You will need to either
  - Use mydata$age to get age
  - or attach(mydata) to get age
    - To undo attach, use detach() function

Making Data Frames (2)

- If x is a matrix, use y = as.data.frame(x) to coerce x to become a data frame
  - x=matrix(1:4, ncol=2, byrow=T)
  - y=as.data.frame(x)
- Use read.table() or read.csv() to read an entire data frame from an external file (.txt or .csv)

Lists

- An R list is an object consisting of an ordered collection of objects as its components.
- Provide a convenient way to return results of a stat computation
- A list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, a list, and so on

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

- Use Smith$name or Smith[[1]] to refer to “Fred”