INTRODUCTION
to
R

Types of statistical software

- command-line software
- GUI-based software
- hybrid types (both command-line and GUI)

Well-known statistical software

- SAS
- SPSS
- Minitab
- Statgraphics
- S-Plus
- R

Why R?

- free
- language almost the same as S
- maintained by top quality experts
- available on all platforms
- continuous improvement

Available through www.r-project.org

R Overview

- Versions of R exist of Windows, MacOS, Linux and various other Unix flavors
- R was originally written by Ross Ihaka and Robert Gentleman, at the University of Auckland, New Zealand during 1990s.
- It is an implementation of the S language, which was principally developed by John Chambers
- In 1998, the Association for Computing Machinery gave John Chambers its Software Award. His citation reads:

"S has forever altered the way people analyze, visualize, and manipulate data ... It is an elegant, widely accepted, and enduring software system, with conceptual integrity."

- Language for statistical computing and data analysis
- Freely available under GPL v2
- Extensive library support
- Programming paradigms
  - procedural
  - functional
  - object-oriented
- General matrix computation (similar to Matlab)
- R can run interactively
  - Statements converted to machine instructions as they are encountered
  - This is much more flexible, but also slower

RStudio: An Integrated Development Environment (IDE) for R is similar to the standard RGui, but more user friendly.

RStudio Windows / Tabs	Description
Console Window	location were commands are entered and the output is printed
Source Tabs	built-in text editor
Environment Tab	interactive list of loaded R objects
History Tab	list of key strokes entered into the Console
Files Tab	file explorer to navigate C drive folders
Plots Tab	output location for plots
Packages Tab	list of installed packages
Help Tab	output location for help commands and help search window
Viewer Tab	advanced tab for local web content

Running R

- Command Line
  - Just type R
  - The R command prompt comes up

    > .....
- With a GUI
  - R Studio
  - R Commander

R Function Libraries

- Implement many common statistical procedures
- Provide excellent graphics functionality
- A convenient starting point for many data analysis projects

- **R Programming Language**
  - Interpreted language

To start, we will review

- Syntax and common constructs
- Function definitions
- Commonly used functions

- **Interactive R**
  - R defaults to an interactive mode
  - A prompt “>” is presented to users
  - Each input expression is evaluated and a result returned

R as a Calculator

```
>1 + 1      # Simple Arithmetic
[1] 2
>2 + 3 * 4  # Operator precedence
[1] 14
>3 ^ 2      # Exponentiation
[1] 9
 > exp(1)    # Basic mathematical functions are available
 [1] 2.718282
 > sqrt(10)
 [1] 3.162278
>pi         # The constant pi is predefined
 [1] 3.141593
 > 2*pi*6378  # Circumference of earth at equator (in km)
 [1] 40074.16
```

Outline

- Variables and Vectors
- Factors
- Arrays and Matrices
- Data Frames
- Functions and Conditionals
- Graphical Procedures

Normal Variables

- We can use <- as the assignment operator in R
- **>** x <- 4
  (set x to 4)
- For printing the value of x
- **> x**
  [1] 4
- OR, **>** print(x)
  [1] 4

A Numeric Vector

- Simplest data structure
  - Numeric vector
  - **>** v <- c(1,2,3)
  - **<-** is the assignment operator
  - **c** is the list concatenation operator
- To print the value, *v*
  - Type : **>** v
  - Output: [1] 1 2 3

A vector is a full fledged variable

- Let us do the following:
- **>** 1/v
  [1] 1.0000000 0.5000000 0.3333333
- **>** v + 2
  [1] 3 4 5
- We can treat a vector as a regular variable
- For **example**, we can have:
  - **>** v1 <- v / 2
    > v1
    [1] 0.5 1.0 1.5

Creating a vector with vectors

- **>** v <- c (1,2,3)

  > v

  [1] 1 2 3

  > vnew <- c (v,0,v)

  > vnew

  [1] 1 2 3 0 1 2 3

The **c** operator concatenates all the vectors

Functions on Vectors and Complex Numbers

- If *v* is a vector
- Here, are a few of the functions that take vectors as inputs:
  mean(v), max(v), sqrt(v), length(v), sum(v), prod(v), sort (v) (in ascending order)
- **>** x <- 1 + 1i

  > y <- 1i

  > x * y

  [1] -1+1i

Generating Vectors

- Suppose we want a vector of the form: (1,2,3,... 100)
- We do not have to generate it manually.
- We can use the following commands:
  > v <- 1:100
  OR
  > v <- seq(1,100)
- *seq* takes an additional argument, which is the difference between consecutive numbers:
  - seq (1,100,10) gives (1,11,21,31 ... , 91)
- rep (2,5) generates a vector (2, 2, 2, 2, 2)

Boolean Variables and Vectors

- R recognizes the constants: TRUE, FALSE
  - TRUE corresponds to 1
  - FALSE corresponds to 0
- We can define a vector of the form:
  - v <- c (TRUE, FALSE, TRUE)
- We can also define a logical vector
  - Can be created with logical operators: <, <=, >=, ==, !=, & and I

```
> v <- 1:9 > 5
> v
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
TRUE  TRUE  TRUE
```

String Vectors

- Similarly, we can have a vector of strings
  - > vec <- c (“f1”, “f2”, “f3”)

```
> vec
[1] "f1" "f2" "f3“
```

- The paste function can be used to create a vector of strings

```
>paste(1:3, 3:5,sep="*")
[1] "1*3" "2*4" "3*5"
```

It takes two vectors of the same length, and an optional argument, *sep*. The i^{th} element of the result string, contains the i^{th} elements of both the arguments, separated by the string specified by *sep*.

Additional Features

- nrow (mat) → Number of rows in the matrix
- ncol (mat) → Number of columns in the matrix

Feature	Function
Eigen Values	eigen
Singular Value Decomposition	svd
Least Squares Fitting	lsfit
QR decomposition	qr