Using R – Foundation

MING-CHANG LEE

Department of Information Management
Yu Da College of Business

October 6, 2006

Email: alan9956@ydu.edu.tw
Web: http://web.ydu.edu.tw/~alan9956/
Outline

1. Introduction
2. Data Manipulation
3. Descriptive Statistics
4. Graphics
1. Introduction

- R for statistical analysis and graphics - Ihaka Gentleman, 1996.

- R is similar to the S language that was developed at AT&T Bell Laboratories by Rick Becker, John Chambers and Allan Wilks.

- Versions of R are available:
  - Microsoft Windows
  - Linux
  - Unix
  - Macintosh OS X (10.4.4)
Features

- An effective data handling and storage facility.
- A suite of operators for calculations on arrays, in particular matrices.
- A large, coherent, integrated collection of intermediate tools for data analysis.
- Graphical facilities for data analysis and display either on-screen or on hardcopy.
- A well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.
How R works

- An interpreted language (Not a compiled language).
- An object-oriented language – variables, data, functions, result are stored in the forms of objects.
R function

- Arguments:
  data, formulate, expressions.

- Packages of function:
  C:/Program Files/R/R-2.3.1/library
  C:/Program Files/R/R-2.3.1/library/base
> : The command line prompt
Examples:

```r
x <- c(1:100)
y <- rnorm(100)*100
hist(y)
test.model <- lm(y ~ x)
test.model
plot(x,y)
library(help="graphics")
```

# On-line help:

```r
? ...  
? rnorm  
? plot  
# Information on package 'base'
library(help="base")
```
2. Data Manipulation

2.1 Data Manipulation Introduction
2.2 Generating Data
2.3 Creating Objects
2.4 Import/Export Data
2.1 Data Manipulation Introduction

- Objects have two intrinsic attributes:
  - mode - the basic type of the elements of the object.
    1. Numeric
    2. Character
    3. Logical (FALSE or TRUE).
    4. Complex
  - Length - the number of elements of the object.

```r
x <- c(1:2)
mode(x)
# [1] "numeric"
length(x)
# [1] 2
```
Name of object

Assign operator:

The name of an object must start with a letter (A-Z and a-z) and can include letters, digits (0-9), and dots (.)

R discriminates for the names of the objects the uppercase letters from the lowercase ones. (i.e. x and X can name two distinct objects)

TRY !

A <- "WEPA"; compar <- TRUE; z <- 3+4i
mode(A); mode(compar); mode(z)
[1] "character"
[1] "logical"
[1] "complex"
Special number

- R correctly represents non-finite numeric values:
  - $+\infty : \text{Inf}$
  - $-\infty : -\text{Inf}$
- NaN: Not a number

```r
x <- 5/0
x
[1] Inf
exp(x)
[1] Inf
exp(-x)
[1] 0
x - x
[1] NaN
```
## Type of Objects

<table>
<thead>
<tr>
<th>object</th>
<th>modes</th>
<th>several modes possible in the same object?</th>
</tr>
</thead>
<tbody>
<tr>
<td>vector</td>
<td>numeric, character, complex or logical</td>
<td>No</td>
</tr>
<tr>
<td>factor</td>
<td>numeric or character</td>
<td>No</td>
</tr>
<tr>
<td>array</td>
<td>numeric, character, complex or logical</td>
<td>No</td>
</tr>
<tr>
<td>matrix</td>
<td>numeric, character, complex or logical</td>
<td>No</td>
</tr>
<tr>
<td>data.frame</td>
<td>numeric, character, complex or logical</td>
<td>Yes</td>
</tr>
<tr>
<td>ts</td>
<td>numeric, character, complex or logical</td>
<td>Yes</td>
</tr>
<tr>
<td>list</td>
<td>numeric, character, complex, logical, function, expression, ...</td>
<td>Yes</td>
</tr>
</tbody>
</table>

PS: ts – time series
2.2 Generating Data

- Regular sequences
  - `c()`
  - `seq()`
  - `scan()`
  - `rep()`
  - `sequence()`
  - `gl()`
  - `expand.grid()`
- Constants
- Missing values

- Random sequences
  - refer “4. Probability Distribution”
Joining (concatenating) vectors: \texttt{c}

- \texttt{c( \ldots )}: Join these numbers together in to a vector.

```r
x <- c(2,3,5,2,7,1)
x
[1] 2 3 5 2 7 1
y <- c(10,15,12)
y
[1] 10 15 12
z <- c(x, y)
z
[1] 2 3 5 2 7 1 10 15 12
```
Subsets of Vectors

# Specify the numbers of the elements that are to be extracted:
x <- c(3,11,8,15,12)  # Assign to x the values 3, 11, 8, 15, 12
x[c(2,4)]  # Extract elements (rows) 2 and 4
[1] 11 15

# Use negative numbers to omit elements:
x[-c(2,3)]
[1] 3 15 12

x>10  # This generates a vector of logical (T or F)
[1] F T F T T
x[x>10]
[1] 11 15 12

# vectors have named elements:
c(ALAN=100, SERENA=2000, ANDY=300, ALPHA=400)[c("ALAN","ANDY")]
ALAN ANDY
  100  300
Regular sequences: seq, scan

- Regular sequence of integers: `seq(from, to, steps)`
- Combine Values into a Vector or List function: `c()`
- Using keyboard to input data: `scan()`

```
x1 <- 1:100
x2 <- 100:1
x3 <- seq(1,10, 0.5)
x4 <- seq(length=9, from=1, to=5)
x5 <- c(1,2,2.5,6,10)
x6 <- scan()
1: 1
2: 2
3: 3
4: 5
5: 
Read 4 items
```
Regular sequences: rep, sequence

- creates a vector with all its elements identical:
  rep( )

- creates a series of sequences of integers each ending by the numbers given as arguments:
  sequence( )

```
rep(1,30)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
sequence(5)
[1] 1 2 3 4 5
sequence(c(5,2))
[1] 1 2 3 4 5 1 2
sequence(c(5,9))
[1] 1 2 3 4 5 1 2 3 4 5 6 7 8 9
```
Generate levels (factors): `gl`

- `gl()`: Generating regular series of factors.
- `gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)`
  - `n`: An integer giving the number of levels (or glass).
  - `k`: An integer giving the number of replications.
  - `length`: An integer giving the length of the result.
  - `labels`: An optional vector of labels for factor levels.
  - `ordered`: The result is ordered or not.
Example: `gl()`

```r
# gl(3, 5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
Levels: 1 2 3

# gl(3, 5, length=30)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
Levels: 1 2 3

# gl(2, 6, label=c("Male", "Female"))
[1] Male Male Male Male Male Male Female Female Female Female Female
Levels: Male Female
```

- **TRY**

```r
x <- gl(3, 3, label=c("優良", "普通", "加油"), length=27)
x
class(x)
```
Generate data frame: expand.grid

- `expand.grid(arguments)`: arguments including:
  - Vectors
  - factors
  - list
Example: `expand.grid()`

```r
x <- expand.grid(h=c(160, 165, 170), w=c(50, 60), sex=c("M", "F"))
print(x)
   h  w sex
 1 160 50 M
 2 165 50 M
 3 170 50 M
 4 160 60 M
 5 165 60 M
 6 170 60 M
...
12 170 60 F
class(x)
[1] "data.frame"
```
## Constants

- LETTERS
- letters
- month.abb
- month.name
- pi

```r
TRY

x <- LETTERS
y <- x[-c(2:10)]
length(x)
length(y)
z <- month.name
z
```
'NA' is a logical constant of length 1 which contains a missing value indicator.

```r
x <- c(pi, 1, 2)
x
[1] 3.141593 1.000000 2.000000
x[2] <- NA
x
[1] 3.141593 NA 2.000000
is.na(x[2])
[1] TRUE
is.na(x[1])
[1] FALSE
# To replace all NAs by 0, use
x[is.na(x)] <- 0
x
[1] 3.141593 0.000000 2.000000
```
2.3 Creating objects

- list
- vector
- factor
- array
- matrix
- data.frame
Creating objects: list

- Function to construct, coerce and check for all kinds of R lists.

```r
data()  # list all available data sets
cars
  speed dist
1   4   2
2   4  10
3   7   4
...
49  24 120
50  25  85
pts <- list(x=cars[,1], y=cars[,2])
plot(pts)
```
Creating objects: vector

- Vector produces a vector of the given length and mode.

vector (mode , length)

- mode specified as argument: numeric – 0 ;
  logical – FALSE ; character – “ “

```
x <- vector(mode="numeric", length=1000000)
is.vector(x)
[1] TRUE
x <- c("Taiwan", "China", "USA")
is.vector(x)
[1] TRUE
```
Creating objects: `factor`

- A factor includes not only the values of the corresponding categorical variable, but also the different possible levels of that variable (even if they are present in the data).

```r
define_factor <- function(x, 
                          levels = sort(unique(x), na.last = TRUE), 
                          labels = levels, 
                          exclude = NA, 
                          ordered = is.ordered(x) )
```

- `x`: a vector of data, usually taking a small number of distinct values
- `levels` specifies the possible levels of the factor (by default the unique values of the vector `x`),
- `labels` defines the names of the levels,
- `exclude` the values of `x` to exclude from the levels,
- `ordered` is a logical argument specifying whether the levels of the factor are ordered.
Example: factor

```r
factor(1:3)
[1] 1 2 3
Levels: 1 2 3
factor(1:3, levels=1:5)
[1] 1 2 3
Levels: 1 2 3 4 5
factor(1:3, labels=c("A", "B", "C"))
[1] A B C
Levels: A B C
x <- factor(letters[1:6], label="YDU")
x
[1] YDU1 YDU2 YDU3 YDU4 YDU5 YDU6
Levels: YDU1 YDU2 YDU3 YDU4 YDU5 YDU6
class(x)
[1] "factor"
```
Creating objects: array

- Creates or tests for arrays.
  - `array(data = NA, dim = length(data), dimnames = NULL)`
  - `as.array(x)`
  - `is.array(x)`

```r
x <- array(letters)
class(x)
[1] "array"
dim(x)
[1] 26
```
Example: array

```r
x <- array(letters)
class(x)
[1] "array"
dim(x)
[1] 26

x <- array(1:3, c(2,4))
x
[1,]  1  3  2  1
[2,]  2  1  3  2
dim(x)
[1] 2 4
length(x)
[1] 8
x[1, ] # select row 1
```
Creating objects: matrix

- `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`
- `as.matrix(x)`
- `is.matrix(x)`
Example: matrix

```
matrix.data <- matrix(
  c(1,2,3,4,5,6),
  nrow = 2,
  ncol = 3,
  byrow=TRUE,
  dimnames = list(c("row1", "row2"), c("C1", "C2", "C3"))
)

matrix.data
  C1 C2 C3
row1 1  2  3
row2 4  5  6
```
Creating objects: data.frame

- A data frame is the type of object normally used in R to store a data matrix.
- A list of variables of the same length, but possibly of different types (numeric, factor, character, logical, . . . ).

- `data.frame(..., row.names = NULL, check.rows = FALSE, check.names = TRUE)`
Example 1: `data.frame`

```r
x <- 1:4; n <- 10; M <- c(10, 35); y <- 2:4
data.frame(x, n)
  x n
1 1 10
2 2 10
3 3 10
4 4 10
data.frame(x, M)
  x M
1 1 10
2 2 35
3 3 10
4 4 35
TRY !
data.frame(x, y)
z <- data.frame(var1= rnorm(5), var2=LETTERS[1:5])
```
Example2: data.frame

data(cars)
help(cars)
class(cars)
[1] "data.frame"
cars
   speed dist
  1   4   2
  2   4  10
...

# TRY! How to add row names (e.g., Row1, Row2,...)
x
   speed dist
Row1   4   2
Row2   4  10
...
2.4 Import/Export Data

# Create a data directory C:\Program Files\R\R-2.3.1\data
# Set working directory
workpath <- "C:/Program Files/R/R-2.3.1/data"
setwd(workpath)
# Get working directory
getwd()
# Create a text file C:\Program Files\R\R-2.3.1\data\r_input.txt
# Import dataset
score1 <- read.table(file="r_input.txt", header= TRUE)
score1
# Add new column data for mid_term
mid_term <- matrix(c(60,80,65,85,80,90,99), nrow=7, ncol=1, byrow=FALSE,
                 dimnames = list(c(),c("mid_term")))
mid_term
# Merge two data.frame( score1 and mid_term)
score2 <- data.frame(score1, mid_term)
score2
# Export dataset
write.table(score2 , file= "r_output.txt", sep = "\t", append=FALSE, row.names=
            FALSE, col.names = TRUE, quote= FALSE)
3. Descriptive Statistics

3.1 Operators
3.2 Mathematical Functions
3.3 Accessing Data
3.4 Descriptive Statistics
### 3.1 Operators

#### Operators

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Comparison</th>
<th>Logical</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>! x</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>&gt; x &amp; y</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
<td>&lt;= x &amp;&amp; y</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
<td>&gt;= x</td>
</tr>
<tr>
<td>^</td>
<td>power</td>
<td>== x</td>
</tr>
<tr>
<td>%%</td>
<td>modulo</td>
<td>!= xor(x, y)</td>
</tr>
<tr>
<td>%/%</td>
<td>integer division</td>
<td></td>
</tr>
</tbody>
</table>

PS: The following characters are also operators for R:
- `$`
- `[`
- `]`
- `[[`
- `::`
- `?
- `<-.`
### 3.2 Mathematical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sum(x)</code></td>
<td>sum of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>prod(x)</code></td>
<td>product of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>max(x)</code></td>
<td>maximum of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>min(x)</code></td>
<td>minimum of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>which.max(x)</code></td>
<td>returns the index of the greatest element of <code>x</code></td>
</tr>
<tr>
<td><code>which.min(x)</code></td>
<td>returns the index of the smallest element of <code>x</code></td>
</tr>
<tr>
<td><code>range(x)</code></td>
<td>id. than c(min(x), max(x))</td>
</tr>
<tr>
<td><code>length(x)</code></td>
<td>number of elements in <code>x</code></td>
</tr>
<tr>
<td><code>mean(x)</code></td>
<td>mean of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>median(x)</code></td>
<td>median of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>var(x)</code></td>
<td>variance of the elements of <code>x</code> (calculated on n -1)</td>
</tr>
<tr>
<td><code>cor(x)</code></td>
<td>correlation matrix of <code>x</code> if it is a matrix or a data frame (1 if <code>x</code> is a vector)</td>
</tr>
<tr>
<td><code>var(x, y)</code></td>
<td>covariance between <code>x</code> and <code>y</code>, or between the columns of <code>x</code> and those of <code>y</code></td>
</tr>
<tr>
<td><code>cor(x, y)</code></td>
<td>linear correlation between <code>x</code> and <code>y</code>, or correlation matrix if they are matrices or data frames</td>
</tr>
</tbody>
</table>
### Mathematical Functions (cont.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>round(x, n)</code></td>
<td>rounds the elements of <code>x</code> to <code>n</code> decimals.</td>
</tr>
<tr>
<td><code>ceiling(x)</code></td>
<td>returns a numeric vector containing the smallest integers not less than <code>x</code>.</td>
</tr>
<tr>
<td><code>floor(x)</code></td>
<td>returns a numeric vector containing the largest integers not greater than <code>x</code>.</td>
</tr>
<tr>
<td><code>rev(x)</code></td>
<td>reverses the elements of <code>x</code>.</td>
</tr>
<tr>
<td><code>sort(x)</code></td>
<td>sorts the elements of <code>x</code> in increasing order.</td>
</tr>
<tr>
<td></td>
<td>To sort in decreasing order: <code>rev(sort(x))</code>.</td>
</tr>
<tr>
<td><code>rank(x)</code></td>
<td>ranks of the elements of <code>x</code></td>
</tr>
<tr>
<td><code>log(x, base)</code></td>
<td>computes the logarithm of <code>x</code> with base &quot;base&quot;</td>
</tr>
<tr>
<td><code>choose(n, k)</code></td>
<td>computes the combinations of <code>k</code> events among <code>n</code> repetitions</td>
</tr>
<tr>
<td></td>
<td>[= \frac{n!}{(n-k)! \cdot k!}]</td>
</tr>
<tr>
<td><code>sample(x, size)</code></td>
<td>resample randomly and without replacement. The option <code>replace = TRUE</code></td>
</tr>
<tr>
<td></td>
<td>allows to resample with replacement.</td>
</tr>
</tbody>
</table>
3.3 Accessing Data

how many elements?  
ith element  
all *but* ith element  
first $k$ elements  
last $k$ elements  
specific elements.  
all greater than some value  
bigger than or less than some values  
which indices are largest

| length(x) |
|---|---|
| x[2] ($i = 2$) |
| x[-2] ($i = 2$) |
| x[1:5] ($k = 5$) |
| x[(length(x)-5):length(x)] ($k = 5$) |
| x[c(1,3,5)] (First, 3rd and 5th) |
| x[x>3] (the value is 3) |
| x[ x< -2 | x > 2] |
| which(x == max(x)) |
3.4 Descriptive Statistics

summary(score2)
score2[2]

score2$Quiz1

quiz1 <- score2$Quiz1

mean(quiz1)

max(quiz1)

min(quiz1)

std(quiz1) # error function

# solution 1
sqrt( sum( (quiz1 - mean(quiz1))^2 /(length(quiz1)-1))) #

std=10.80123

# solution 2
std = function(x) sqrt(var(x))

std(quiz1) # same as solution 1 # TRY sd(quiz1)
4. Graphics

4.1 Graphical device
4.2 Plot
4.3 Bar charts
4.4 Pie charts
4.5 Box-and-whisker plot
4.6 Stem-and-Leaf plot
4.1 Graphical device

- The result of a graphical function is sent to a graphical device.
  - Graphical window
  - File.

- There are two kinds of graphical functions:
  1. *High-level plotting functions* which create a new graph.
  2. *Low-level plotting functions* which add elements to an already existing graph.
Graphical devices

- Open a graphical window:
  `x11()` or `windows()`

- List of available graphical devices:
  `dev.list()`

- Show/Change the active device:
  `dev.cur()`, `dev.set(3)`

- Close the active device:
  `dev.off()`, `dev.off(2)`
4.2 Plot( )

- plot(x, y) # Same as “plot(y ~ x)”
- type:
  - p : point,
  - l: line,
  - b: both
- pch: controls the type of symbol, either an integer between 1 and 25, or any single character within “ “
- col: controls the colour of symbols. e.g., “red”
- xlab = "string"
- ylab = "string"
- main = "string"
- sub = "string"
- cex: a value controlling the size of texts and symbols with respect to the default
- lwd: a numeric which controls the width of lines
Figure 2: The plotting symbols in R (pch=1:25). The colours were obtained with the options col="blue", bg="yellow", the second option has an effect only for the symbols 21–25. Any character can be used (pch="*", "?", ".", ...).
Example: plot

data()
data(cars) # Speed and Stopping Distances of Cars
plot(cars) # x-axis: speed; y-axis: dist
plot(cars$dist, cars$speed)
plot(cars, type="b")
Example: plot (cont.)

```r
> cl <- colors()
> cl
[1] "white" ...

plot(cars,
     type="b",
     pch=5,
     col="red",
     xlab="Speed(mph)",
     ylab="Stop distance(ft)",
     main="Speed and Stopping Distances of Cars",
     sub = "Figure 1: Plotting demonstration")
```
4.3 Bar charts – barplot()

```r
CarArrived <- table(NumberOfCar <- rpois(100, lambda=5))
CarArrived

0  1  2  3  4  5  6  7  8  9 10 11 13
1  4  4 10 19 21 11 16  3  3  5  2  1

barplot(CarArrived)
barplot(CarArrived, col=rainbow(14))
```
4.4 Pie charts – `pie()`

```
pie.sales <- c(0.14, 0.30, 0.26, 0.15, 0.10, 0.05)  # Sales ratio

names(pie.sales) <- c("Taipei1", "Taipei2", "Taipei3", "Taichung", "Kao", "Other")  # Sales area

pie(pie.sales)  # default colours

pie(pie.sales, col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))  # set colour

pie(pie.sales, density = 10, clockwise=TRUE)  # The density of shading lines
```
4.5 Box-and-whisker Plot – boxplot()

# Produce box-and-whisker plot(s) of the given (grouped) values.
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100), T5 = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
boxplot(data.frame(mat), main = “boxplot”)
4.6 Stem-and-Leaf Plot – stem( )

mat = scan()
1: 2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5
21:
Read 20 items
stem(mat)

The decimal point is 1 digit(s) to the right of the |

0 | 000222344568
1 | 23446
2 | 38
3 | 1
References


http://cran.r-project.org/doc/contrib/usingR-2.pdf

http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

http://cran.r-project.org/doc/manuals/R-intro.pdf


6. The R Manuals – Documentation Manuals
http://cran.r-project.org/
THANKS

Q & A