Basic R Programming II

Teerasak E-kobon

Department of Genetics, Faculty of Science, KU

Contact: fscitse@ku.ac.th Room 4508

Contents

1. Basic R programming
2. R functions
3. Biological data types
4. Data input in R
5. Basic data analysis in R (Practical)

Basic R programming

1. Type function name for more details
2. To check data type: typeof(), is.integer(), is.character(),
   is.double(), is.numeric(), is.function(), is.matrix(), …
3. To change data type: as.character(), as.integer(), as.double(),
   as.numeric(), …
4. The : is used for a range of data. 6:3 = 6,5,4,3
5. letters are standard variable in R for A-Z characters.
6. sample() is a random sampling function

```
sample(1:100, 6)          # random 6 numbers from 1-100
```

Basic R programming (Con.)

7. x = “I love Biology.” # assignment

   x # to call a variable for a value inside

   print(x) # display the value in x variable on screen

8. [ ] and [[ ]] use for subsetting and indexing the data

9. paste() function used for concatenating the character/string data with a separation sign (sep = “ ”)

   paste(“I”, “love”, “Biology”, “.”)

10. Conditional operators &, && and, | , | | or, == equal, != not equal, ! not

Basic R programming (Con.)

11. @ and $ used to access sub-variables or sub-objects or components
12. ~ used for a formula
13. append() function for combining strings

    append("I", "love", "Biology", ".")

14. Subsetting/indexing

```
x = 11:22
x[3]
x[c(1,4,6)]
x[3:7]
x[-2]
x+10
x[x>15]
y = letters[1:20]
```

Basic R programming (Con.)

14. Subsetting/indexing (Con.)

```
y[3] = “bird”          # subsetting and assignment
which(CONDITION)
which(y == “g”)

a1 = c(1,2,3,4,5)
a2 = c(4,5,6,7,8,9,10)
match(a1,a2)           # compare two variables
a1 %in% a2             # do the same as above
intersect(a1,a2)
setdiff(a1,a2)
union(a1,a2)
```

Basic R programming (Con.)

15. Function writing:

sq1 = function (x, y) return (x*x)

sq1: function name

function(): a function to build function

x,y: arguments/parameters of the function

return: to give back output

(x*x): expressions/codes/scripts to be executed

if more than one line, put in { } and separated with ;

Basic R programming (Con.)

16. Flow control/recursion/iteration

for (VARIABLE in SEQ) {EXPRESSION}

while (CONDITION) {EXPRESSION}

repeat {EXPRESSION}

Example:

```
for (I in 1:10) print(i)

i = 5

while (i != 0) {print(i);
                i = i+1}

repeat {print(i); i = i-1; if(i==3) {break}}
```

Basic R programming (Con.)

17. Flow control/recursion/iteration (Con.)

```
j=10
repeat {print(j);
        j = j-1;
        if(j==3) {break}
        }
```

*** Tidy the program by indentation and newline.

Basic R programming (Con.)

18. Conditional statement

```
if (CONDITION == TRUE) {EXPRESSION 1}
        else {EXPRESSION 2}
animal = “ant”
if (animal == “ant”){print(“This is an ant.”)} else
        {print(“This is not an ant!!”)}
ifelse(x%%2 == 0, “YES”, “NO”)
```

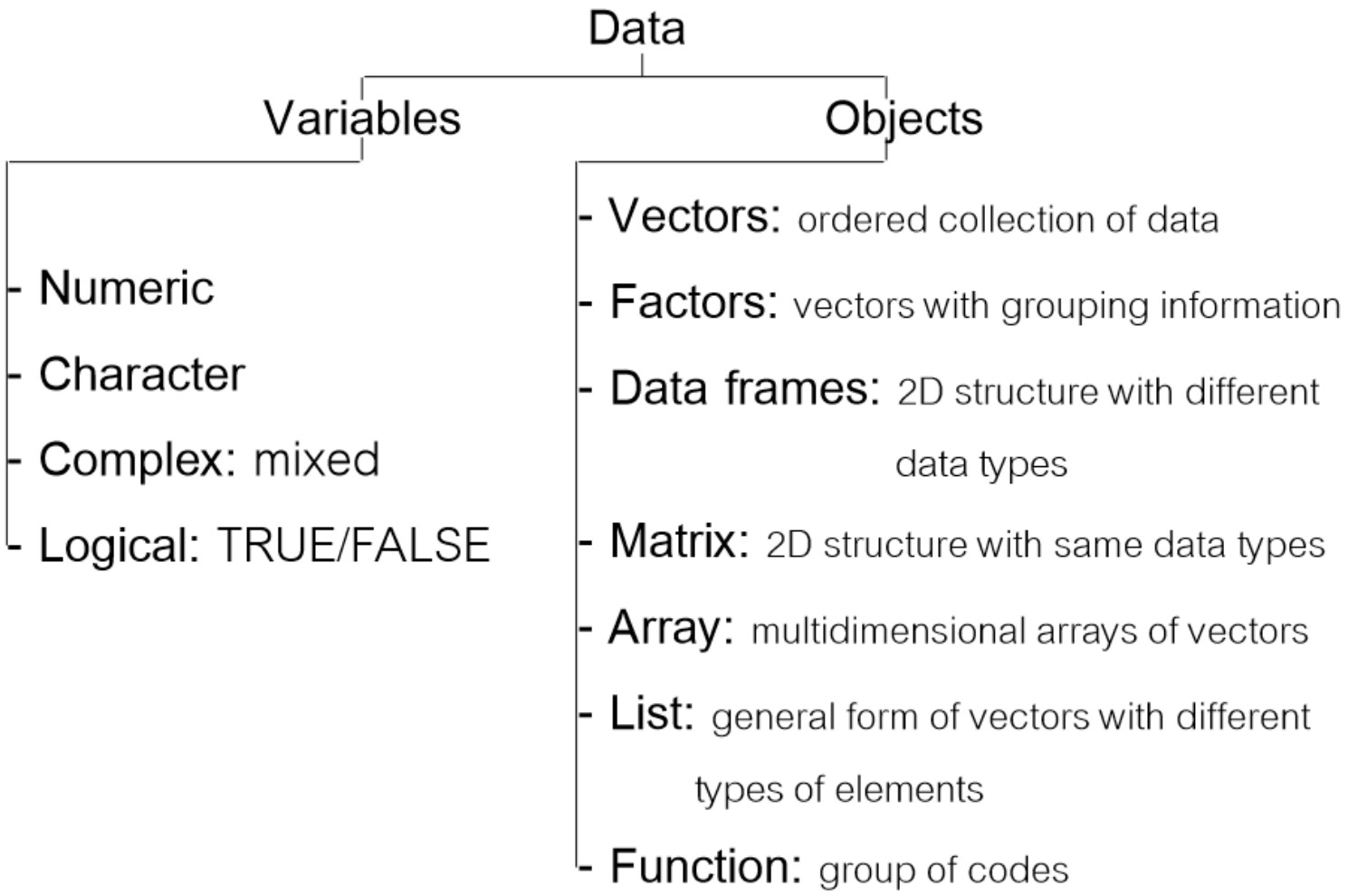
Basic R programming (Con.)

19. Other useful functions

```
library(LIBRARY NAME)
example(FUNCTION)
search()
ls()                    # show variables
objects()               # show objects
str()                   # show type and structure of the variable
class()                 # show the class
summary()               # show basic statistic of the variable/data
attributes(OBJECT) # show attributes of an object
dir()                   # show file directory
getwd()                 # check the current directory
setwd()                 # set the current directory as default
table()                 # summarize data in the variable/counting
merge()                 # merge data frames
```

Data types in R

- numeric, categorical, ordinal, time series,…



Variable naming

1. Easy to remember and relate to the data

2. Dot . Can use to separate the name

3. Should not have these signs in the variable names

£, $, %, ^, *, +, -, ( ), [ ], #, !, ?, <, >

There are special characters/operators in R

4. Small and capital letters are different:

Ant not same as ANT and ant

Data input in R

1: Input a single value (variable or constant) by keyboard typing

```
a = 14
```

2: Input multiple values using function, c()

```
wing <- c(59, 55, 53.5, 55, 55)
```

*** use indexing [ ], range :, sum(), min(), max(), mean()

3: Combine values from multiple variables using c(), cbind() for column, rbind() for row

4: Combine data using vector() function

```
wing <- vector(length = 5)
```

Data input in R

5: Combine data using matrix() function for data of the same types

```
BirdMatrix <- matrix(nrow = 5, ncol = 4)
BirdMatrix[ ,1] <- c(59, 55, 53.5, 55, 55)
BirdMatrix[ ,2] <- c(22.3, 19.7, 20.8, 20.3, 21.5)
BirdMatrix[ ,3] <- c(31.2, 30.4, 30.6, 30.3, NA)
BirdMatrix[ ,4] <- c(9.5, 13.8, 14.8, 15.2, 15.7)
colnames(BirdMatrix) <- c("wing", "leg", "head", "weight")
as.matrix()
is.matrix()
```

Data input in R

6: Combine different data types using data.frame() function

- suitable for data in table format

```
BirdDframe <- data.frame(W = wing,
                +          L = leg,
                +          H = head,
                +          We = weight)
BirdDframe$W                                  # to see sub-data
```

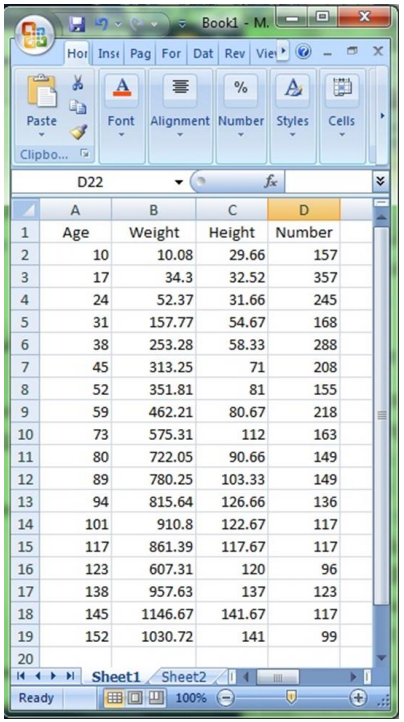
7: Combine more complex data using list()

```
D1 <- c(1, 2, 3)
D3 <- matrix(nrow = 2, ncol = 2)
D3[ ,2] <- c(35, 39)
```

```
D2 <- c("Dog", "Cat", "Bat")
D3[ ,1] <- c(24, 26)
ListX <- list(D1 = D1, D2 = D2, D3 = D3)
```

Large data import to R

1. Import data in the tab-delimited format (.txt)



	A	B	C	D
1	Age	Weight	Height	Number
2	10	10.08	29.66	157
3	17	34.3	32.52	357
4	24	52.37	31.66	245
5	31	157.77	54.67	168
6	38	253.28	58.33	288
7	45	313.25	71	208
8	52	351.81	81	155
9	59	462.21	80.67	218
10	73	575.31	112	163
11	80	722.05	90.66	149
12	89	780.25	103.33	149
13	94	815.64	126.66	136
14	101	910.8	122.67	117
15	117	861.39	117.67	117
16	123	607.31	120	96
17	138	957.63	137	123
18	145	1146.67	141.67	117
19	152	1030.72	141	99
20				

- read.table()
- Use NA for missing data
- avoid using names that contain symbols such as ?, $,%, ^, &, *, (,),-, #, ?,,,<,>, /, |, \, [,] ,{, and }
- Column for variables
- Row for samples
- Rice <- read.table(file = “C:\\Rfolder\\Rice.txt”, header = TRUE)

Large data import to R

1. Import data in the tab-delimited format (.txt)

Several functions can use to process the imported data.

```
setwd("C:\\Rfolder\\")
Rice <- read.table(file = "Rice.txt", header = TRUE)
names(Rice)
str(Rice)
Rice$Age
Rice[ ,4]
mean(Rice$Age)
```

Large data import to R

1. Import data in the tab-delimited format (.txt)

Several functions can use to process the imported data.

```
attach(Rice)                                    # begin
Age
mean(Age)
detach(Rice)                                    # finish
unique(Rice$Age)
RiceAge <- Rice[Rice$Age <= 45, ]               # select data
RiceAge <- Rice[Rice$Age <= 45 & Rice$Weight > 50, ]
```

Conditional & Boolean Operators

- Conditional operators:

  ==, >, <, >=, <=, != (not equal)

- Boolean operators: & (and), | (or)

Large data import to R

2. Import data in the excel format (.csv) using read.csv() and

   read.csv2()

3. Read each line in the file using readLines() function

   readLines("FILE PATH", n = -1)

   unlink("FILE PATH") # tidy up

4. Read in data via keyboard typing with scan()

   mycolors = scan(,what='')

   *** press enter twice to finish.

Data exporting in R

1. Use write.table() function

Variable to be writen

File name/ directory

Separation charater

```
> write.table(RiceMerged, file = "RiceMerged.txt", sep = " ",
quote = FALSE, append = FALSE, na = "NA")
```

Exclude "" sign

Write as a new file

Not append

Indicate that there Are missing values

2. Write .csv file using write.csv() function

```
write.csv(MyData, file = "MyData.csv",row.names=FALSE)
```

References

Teetor, P. 2011. R Cookbooks. O'Reilly Media, US.

Crawley, M.J. 2005. Statistics: An introduction using R. Wiley, US.

Crawley, M.J. 2012. The R book. 2nd Eds. Wiley, US.