Contents

Introduction to R

- R is a language and environment for statistical computing and graphics.
- R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, . . .) and graphical techniques, and is highly extensible.
- One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed.
- R offers plenty of options for loading external data, including Excel, Minitab, SAS and SPSS files.
- Here you can download R and RStudio

Basics

- After R is started, there is a console awaiting for input. At the prompt `>`, you can enter numbers and perform calculations.
- The assignment operators are the left arrow with dash `<-` ,`->`and equal sign `=`.
- The character `#` marks the beginning of a comment.All characters until the end of the line are ignored.

```
# Example
4+5
```

```
## [1] 9
```

```
x <- 5
print(x)
```

```
## [1] 5
```

Create a vectors

`c` *function*

- R functions are invoked by its name, followed by the parenthesis and arguments. The function `c` is used to combine three numeric values into a vector

The command `c(1,2,3,4,5)` combines the numbers 1,2,3,4 and 5 to a vector.

```
c(1,2,3,4,5)
```

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

Basic operations

R's basic operators have the following precedence (listed in highest-to-lowest order)

`^` exponentiation

`- +` unary minus and plus

`:` sequence operator

`%/% %%` integer division, remainder

`* /` multiplication, division

`+ -` addition, subtraction

```r
2^3^2
```

```
## [1] 512
```

```r
(2^3)^2
```

```
## [1] 64
```

```r
2^(3^2)
```

```
## [1] 512
```

```r
sqrt(2)
```

```
## [1] 1.414214
```

Seq operator and seq function

The expression $n_1 : n_2$, generates the sequence of integers from n_1 to n_2

```r
# print the numbers 1 to 15
1:15
```

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

```r
# specifies interval and increment
seq(2,8,by=2)
```

```
## [1] 2 4 6 8
```

```r
# specifies interval and the number of elements
seq(0,1,length=11)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Generating sequences of letters-lower case alphabets

```r
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```r
letters[5:10]
```

```
## [1] "e" "f" "g" "h" "i" "j"
```

sequence of uppercase alphabets

```r
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
LETTERS[2:6]
```

```
## [1] "B" "C" "D" "E" "F"
```

Repeats-the command `rep`

```
rep(2, times = 5)
```

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

```
rep(1:3, times = 4)
```

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

```
rep(1:3, each=2, times = 4)
```

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

Matrix

- Matrices are important objects in any calculation. A matrix is a rectangular array with p rows and n columns.
- The parameter `nrow` defines the row number of a matrix.
- The parameter `ncol` defines the column number of a matrix.
- The parameter data assigns specified values to the matrix elements.

`matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`

In R, a 4 × 2-matrix X can be created with a following command:

```
x <- matrix(data = c(1:8), nrow = 4, ncol = 2)
print(x)
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

One can access a single element of a matrix with `x[i,j]`:

```
x[3, 2]
```

```
## [1] 7
```

We can get specific properties of a matrix:

```
dim(x)
```

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

```
nrow(x)
```

```
## [1] 4
```

```
ncol(x)
```

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

Assigning a specified number to all matrix elements:

```
y <- matrix(data = 3, nrow = 2, ncol = 2)
print(y)
```

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

Construction of a diagonal matrix, here the identity matrix of a dimension 2:

```
diagonal <- diag(3, nrow = 2)
diagonal
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    0    3
```

Transpose of a matrix X: X'

```
z <- t(x)
z
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
```

Basic matrix operation

```
a <- matrix(1:9, nrow = 3, ncol = 3)

b <- matrix(11:19, nrow = 3, ncol = 3)

a+b
```

```
##      [,1] [,2] [,3]
## [1,]   12   18   24
## [2,]   14   20   26
## [3,]   16   22   28
```

```
a-b
```

```
##      [,1] [,2] [,3]
## [1,]  -10  -10  -10
## [2,]  -10  -10  -10
## [3,]  -10  -10  -10
```

```
a/b
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.09090909 0.2857143 0.4117647
## [2,] 0.16666667 0.3333333 0.4444444
## [3,] 0.23076923 0.3750000 0.4736842
```

Multiplication of a matrix with a constant

```
# Note: x is already defined
x*5
```

```
##      [,1] [,2]
## [1,]    5   25
## [2,]   10   30
## [3,]   15   35
## [4,]   20   40
```

Matrix multiplication: operator `%*%`

- `crossprod()` computes x^TÖy

Note: Command `crossprod()` executes the multiplication faster than the conventional method with t(a)%*%a

```
a <- matrix(1:6, nrow = 3, ncol = 2)
a
```

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

```
b <- matrix(11:16, nrow = 2, ncol = 3)
b
```

```
##      [,1] [,2] [,3]
## [1,]   11   13   15
## [2,]   12   14   16
```

```
a %*%b
```

```
##      [,1] [,2] [,3]
## [1,]   59   69   79
## [2,]   82   96  110
## [3,]  105  123  141
```

```
crossprod(t(a), b)
```

```
##      [,1] [,2] [,3]
## [1,]   59   69   79
## [2,]   82   96  110
## [3,]  105  123  141
```

Access to rows, columns or submatrices

```
x <- matrix(1:9, nrow = 3, ncol = 3)
x
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
x[2, ]
```

```
## [1] 2 5 8
```

```
x[1:2,2:3]
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    5    8
```

Sorting

`sort` function sorts the values of a vector in ascending order (by default) or descending order.

```r
sort(c(20,50, 10, 30, 90,70, 80), decreasing = FALSE)
```

```
## [1] 10 20 30 50 70 80 90
```

```r
sort(c(20,50, 10, 30, 90,70, 80), decreasing = TRUE)
```

```
## [1] 90 80 70 50 30 20 10
```

Practice

1. Create `a` sequence from 1 to 6
2. Create `b` sequence from 7 to 12
3. Create matrix for `a` and `b`.
4. Calculate the basic operation for the two matrix.
5. Arange the `b` sequence in descending order.

Logical operator

```r
a <- 5
b <- 3

# Equal to
print(a == b)
```

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

```r
# Not equal to
print(a != b)
```

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

```r
# Greater than
print(a > b)
```

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

```r
# Less than or equal to
print(a <= b)
```

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

```r
# Less than
print(a < b)
```

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

```r
# Greater than or equal to
print(a >= b)
```

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

```r
x <- c(1, 0, 3, 5)
y <- c(0, 3, 3, 2)

# Element-wise logical AND
print(x & y)
```

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

```r
# Element-wise logical OR
print(x | y)
```

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

```r
# Element-wise logical NOT on x
print(!x)
```

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

```r
# Comparison: x is greater than y
print(x > y)
```

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

Data Frames

- In a data frame, we can combine variables of equal length,with each row in the data frame containing observations on the same unit.
- Variables in a data frame may be numeric (numbers) or categorical (characters or factors).
- Basic Syntax: `data_frame_name <- data.frame(column1 = vector1, column2 = vector2, ...)`

```r
id <- c(101, 102, 103)
name <- c("Alice", "Bob", "Charlie")
gender <- factor(c("Male", "Female", "Female"))
full_time <- c(TRUE, FALSE, TRUE)
salary <- c(50000, 60000, 55000)

# Creating the data frame
employee_data <- data.frame(ID = id, Name = name,Gender = gender,
                            FullTime = full_time, Salary = salary)

# Print the data frame
print(employee_data)
```

```
##    ID    Name Gender FullTime Salary
## 1 101   Alice   Male     TRUE  50000
## 2 102     Bob Female    FALSE  60000
## 3 103 Charlie Female     TRUE  55000
```

Arithmetic mean

Ungrouped data Find the mean value for 55,68,72,79,90,63,85,77,64,82, 55, 66, 89, 78, 67.

```r

data <- c(55,68,72,79,90,63,85,77,64,82, 55, 66, 89, 78, 67)
data
```

Manual method

```
##  [1] 55 68 72 79 90 63 85 77 64 82 55 66 89 78 67
```

```r
avg <- sum(data)/length(data)
print(avg)
```

```
## [1] 72.66667
```

```
mean(data)
```

Using R command

```
## [1] 72.66667
```

Grouped data - Discrete frequncy distribution

```
x <- table(data)
x
```

Manual method

```
## data
## 55 63 64 66 67 68 72 77 78 79 82 85 89 90
##  2  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
fx <- unique(data)*x
fx
```

```
## data
##  55  63  64  66  67  68  72  77  78  79  82  85  89  90
## 110  68  72  79  90  63  85  77  64  82  66  89  78  67
```

```
N <- sum(x)
avg <- sum(fx)/N
avg
```

```
## [1] 72.66667
```

```
mean(data)
```

Using R command

```
## [1] 72.66667
```

Practice

1. max(c(62,83,44,75)^ -c(9,-3)) / min(c(52,62,71,85)^c(2,3)) - prod(c(1,2,1,2) ^ c(1,2))+max(c(12,13,14,15)^c(2,3)) ?
2. X1 <- c(123,258,318,624), X2 <- sqrt(X1 ^3) +X1/X1^2-X1**(1/2)
3. X<-matrix(nrow=3,ncol=3,data= c(10,20,30,40,50,60,70,80,90),byrow=F) then X[,2]?
4. X<-matrix(nrow=3, ncol=3, data = c(10,20,30,40,50,60,70,80,90) , byrow=F) then X[2:3,2:3] ?
5. sqrt(abs(seq(-6,6, by = 3)))?
6. x <- c(10, 75, 20, 35, 30, 40, 180, 50, 60, 27, 70, 67, 80, 50, 39, 120) x[(x>50)]? x[(x - 20 > 40)] x[(x^2 + 10 > 50)]