1. Commonly Used R Objects in Scientific Computing

Cheng Peng

Lab Note for MAT325 Numerical Analysis

Contents

1 Intrdoction

R was initially created by group of statisticians for data analysis (for free). In the past decade, many people from other disciplines contributed to the continuous development of this program. It is among the top programming language in data science and machine learning and widely used to perform a variety of non-statistical tasks, including data processing, information visualization, data mining, and scientific computing, etc.

During the semester, I will write series of short notes on R to implement numerical algorithms to be covered in the course. I will also write one or two lab notes to cover optimization problems in machine learning and data science.

The following three books focus on using R for scientific computing. You can find one of them as a reference when you make programs for this class.

1. **Introduction to scientific programming and simulation using R**. This book can be found from internet.
2. **Mastering Scientific Computing with R**. WCU library has this eBook. You can access this book using the following link. https://ebookcentral.proquest.com/lib/wcupa/detail.action?pq-

origsite=primo&docID=1936749

3. **Using R Numerical Analysis in Science and Engineering**. You can also find this from internet.

For those who programmed in MATLAB, you can check the following page to see the back-to-back syntax comparison between R and MATLAB https://mathesaurus.sourceforge.net/octave-r.html.

R is case sensitive!.

2 Vectors and Matrices

Vectors and matrices are two major R objects that will be used frequently in numerical analysis. This section outlines the definition and utilization of vectors and matrices.

2.1 Vectors

A vector is a collection of *like* elements without dimensions. The vector element or elements must be of the same types of data (either **character**, **numeric**, or **logical**).

2.1.1 Definition

An R vector defined by a built-in R function `c()` (**c** stands for concatenate). The following are examples of basic types of vectors.

```
intVec = c(1,3,6,7)                    # vector of integers
charVec = c('One','Two','Three')       # vector of characters, string vector
logiVec = c(FALSE, TRUE)               # logical vectors
singe.Val.Vec = c("convergent")        # single element character vector
emptyVec = NULL                        # empty vector / null vector
##
intVec                                 # type the of the name of intVec
```

```
## [1] 1 3 6 7
```

2.1.2 Use of Vector Index

One can access elements in a vector through index using square bracket `[idx]`. Similar to MATLAB, **R index starts from 1**!

```
exampleVec = c(1, 2, 3, 2, 7, 9, 11, 15, 7, 2)  # use this vector as an example
###
exampleVec[7]             # extract the 7th element in the vector
```

```
## [1] 11
```

```
exampleVec[c(1,2,3)]      # extract the first three elements, c(1,2,3) is the vector of indexes
```

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

```
exampleVec[-7]            # drop the 7th element from the vector
```

```
## [1]  1  2  3  2  7  9 15  7  2
```

```
exampleVec[-c(1,2,3)]     # drop the first three elements
```

```
## [1]  2  7  9 11 15  7  2
```

```
duplVec = exampleVec      # duplicate an existing vector and rename it
## The following code replaces elements with NEW elements
```

```
duplVec[c(9,10)] = c(99,100)   # replace elements 7, 4 with 99 and 100 respectively.
duplVec                  # display the modified vector
```

```
##  [1]   1   2   3   2   7   9  11  15  99 100
```

2.1.3 Operations Between Vectors

The following examples show the operations commonly used in error analysis.

```
A = c(5, 2, 3, 7, 5, 1, 9)
B = c(3, 2, 4, 8, 2, 7)
## next we define different new vectors using A and B
new01 = c(A,B)    # concatenate A and B
new01             # display new01
```

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

```
new02 = A-5       # subtract 5 from individual element in A
new02
```

```
## [1]  0 -3 -2  2  0 -4  4
```

```
new03 = A^2       # square each individual element in A
new03
```

```
## [1] 25  4  9 49 25  1 81
```

```
new04 = 2*A       # multiply each element of A by 2
new04
```

```
## [1] 10  4  6 14 10  2 18
```

2.1.4 Shortcuts for Defining Vectors

There are shortcuts to define patterned vectors (sequence). The following are few examples.

```
seq.vec = seq(1, 100, by = 2)       # This defines a sequence: 1, 3, 5, 7, ..., 99. by = jump
seq.vec
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
## [26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

```
seq.vec01 = seq(1, 99, length = 5)  # This defines a sequence with 5 numbers that are equally spaced be
seq.vec01
```

```
## [1]  1.0 25.5 50.0 74.5 99.0
```

```
seq.vec02 = rep(1, 100)             # This defines a sequence with 100 1s.
seq.vec02
```

```
##   [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 1 1 1 1 1 1 1 1 1 1
##  [38] 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 1 1 1 1 1 1 1 1 1
##  [75] 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 1 1 1 1 1 1 1 1 1
```

```
seq.vec03 = 3:10                    # This sequence: 3,4,5,6,7,8,9,10.
seq.vec03
```

```
## [1]  3  4  5  6  7  8  9 10
```

```
seq.vec04 = letters                 # letters is a built-in vector of all lower case letters
seq.vec04
```

```
##  [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"
```

```
seq.vec05 = LETTERS                 # LETTERS is a built-in vector of all uppercase letters
seq.vec05
```

```
##  [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[1:7]                        # first 7 uppercase letters.
```

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

2.2 Matrices

R matrices are two dimensional table indexed by two subscripts using $[i, j]$, where i = index of row of the matrix and j = index of the column of the matrix. On can access the matrix using index $[i, j]$. The following are some examples of matrices

```
vec0 = 1:36
m01 = matrix(vec0, ncol = 9, byrow = TRUE)   # this defines a 4x9 matrix, the cells were filled from ve
m01
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    1    2    3    4    5    6    7    8    9
## [2,]   10   11   12   13   14   15   16   17   18
## [3,]   19   20   21   22   23   24   25   26   27
## [4,]   28   29   30   31   32   33   34   35   36
```

```
m02 = matrix(vec0, nrow = 6, byrow = FALSE)  # this defines a 6x6 square matrix, cells were filled from
m02
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    7   13   19   25   31
## [2,]    2    8   14   20   26   32
## [3,]    3    9   15   21   27   33
## [4,]    4   10   16   22   28   34
## [5,]    5   11   17   23   29   35
## [6,]    6   12   18   24   30   36
```

```
m03 = matrix(ncol = 5, nrow = 6)        # this defined a 5x6 empty matrix
m03
```

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

```
m03[4,5] = 99                           # replace the element in row 4 and column 5 with 99.
m03
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   NA   NA   NA   NA   NA
## [2,]   NA   NA   NA   NA   NA
## [3,]   NA   NA   NA   NA   NA
## [4,]   NA   NA   NA   NA   99
```

```
## [5,]   NA   NA   NA   NA   NA
## [6,]   NA   NA   NA   NA   NA
```

3 Built-in Mathematical Functions and Operators

R has built in most of the commonly used mathematical functions and important scalars. The following is a partial list.

3.1 Arithmetic Operators

`+` – addition

`-` – subtraction

`*` – multiplication

`/` – division

`^` – raise to the power of

3.2 Basic Mathematical Functions

`abs()` - absolute value

`sqrt()` - square root

`round()` - rounding function

`ceiling()` - rounding up

`floor()` - rounding down

`sign()` - sign of a number

`exp()` - natural base exponential function

`log()` - natural base logarithmic function

`log10()` - base 10 logarithmic function

3.3 Trigonometry

`sin()` – sine

`cos()` – cosine

`tan()` – tangent

`asin()` – sine inverse

`acos()` – cosine inverse

`atan()` – tangent inverse

3.4 Linear Algebra

`+` – element-wise addition

`-` – element-wise subtraction

`*` – element-wise multiplication

`/` – element-wise division

`%*%` – matrix multiplication

`t()` – transpose

`eigen()` – eigenvalues and eigenvectors

`solve()` – inverse of matrix

`rbind()` – combines vectors of observations horizontally into matrix class

`cbind()` – combines vectors of observations vertically into matrix class

3.5 Output Functions

`print()` –

`sprintf()` – C

4 Basic R Objects