Tutorial 1 (Introdcution to R Programming)

Mr. Zebin Yang

Welcome to R!

- R website: https://cran.r-project.org
- R is a freely available programming language and is also the most widely used statistics programming language.
- The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions.
- R has the state-of-art graphics capability.

Intended Learning Outcomes

- Learn R's basic data structure and syntax.
- Capable to use R for simple data processing.

Practice

- datacamp

https://campus.datacamp.com/courses/free-introduction-to-r/

"Hello, World!" program

- R Command Prompt

It is easy to start your R command prompt by just typing commands at console. For example

```
myString <- "Hello, World!"
print(myString)
```

```
## [1] "Hello, World!"
```

"Hello, World!" program

- R Script File

Suppose we have a file "test.R" contatining the following codes.

```
myString <- "Hello, World!"
print(myString)
```

Then we can run this file in the command line by:

```
Rscript test.R
```

Loading Data

- Check & reset working directory

```
getwd()
```

```
## [1] "E:/HKU data/Courses@HKU/Stat3622/Tutorial/T1/R"
```

```
setwd("../") # redirecting to a new directory
```

- Load data from csv or txt file

```
df = read.csv("City_df.csv")
df = read.table("MyData.txt",sep=',',header=TRUE)
head(df, 2)
```

```
##        value category
## 1 -2.5456956        a
## 2  0.7440767        a
```

Loading Data

- R built-in dataset

```
head(iris, 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
```

If Else Control

An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

```
x <- c("what","is","truth")

if("Truth" %in% x) {
   print("Truth is found the first time")
} else if ("truth" %in% x) {
   print("truth is found the second time")
} else {
   print("No truth found")
}
```

```
## [1] "truth is found the second time"
```

Loops Control

- while loop

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

- for loop

Like a while statement, except that it tests the condition at the end of the loop body.

```
v <- c("Hello","loop")
for (cnt in seq(2, 4)) {
   print(v)
}
```

```
## [1] "Hello" "loop"
## [1] "Hello" "loop"
## [1] "Hello" "loop"
```

Function

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

- Function Name: This is the actual name of the function. It is stored in R environment as an object with this name.
- Arguments: An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.
- Function Body: The function body contains a collection of statements that defines what the function does.
- Return Value: The return value of a function is the last expression in the function body to be evaluated.

```
new.function <- function(a = 3, b = 6) {
   print(a * b)
}
new.function(a = 9,5)
```

```
## [1] 45
```

Packages
R packages are a collection of R functions, complied code and sample data. They are stored under a directory called "library" in the R environment.

```
#Get library locations containing R packages
.libPaths()
```

```
## [1] "C:/Program Files/R/R-3.5.3/library"
```

- Install package

install.packages("Package Name")

- Load Package to Library

library("package Name", lib.loc = "path to library")

Data Types

- Atomic data types

No.	Examples
Logical	TRUE, FALSE
Numeric	12.3, 5, 999
Integer	2L, 34L, 0L
Character	'a' , '"good", "TRUE", '23.4'
Complex	3 + 2i
Raw	"Hello" is stored as 48 65 6c 6c 6f

Data Types

- R-objects
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Data Frames

Vectors

Vector is the very basic the R-object, which hold elements of different classes.

- Use c() function

```
x = c("aa","nn","cc"); x
```

```
## [1] "aa" "nn" "cc"
```

```
x = 1:10; x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

- Use seq() function

```
seq(1, 9, by=2)  # From 1 to 19, with step 2
```

```
## [1] 1 3 5 7 9
```

```
seq(1, 19, length=10) # From 1 to 19, with length 10 (step = (19-1)/
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

Vectors

- Use rep() function

```
rep(1:4,2) # Repeated the whole vector twice
```

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

```
rep(1:4,c(2,1,2,1)) # Each element is repeated individually, and the
```

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

```
rep(1:4,each=2,len=10) # Each element is repeated individually, and t
```

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

```
rep(1:4,each=2,times=3) # The para "time" determines this procedure t
```

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

Vectors

- Basic operations of vectors

```
# Length
x = c(3:7)
length(x)
```

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

```
# Naming
names(x) = c(letters[1:length(x)])
x
```

```
## a b c d e
## 3 4 5 6 7
```

```
n = c(2,3,5); s = c("aa","bb","cc","dd","ee")
c(n, s) # Combine Vectors
```

```
## [1] "2"  "3"  "5"  "aa" "bb" "cc" "dd" "ee"
```

```
s[-4] #Exclude Elements
```

```
## [1] "aa" "bb" "cc" "ee"
```

Vectors

- Vector arithmetic

```
a = c(1,3,5,7)
b = c(1,2,4,8)
5*a
```

```
## [1]  5 15 25 35
```

```
a + b
```

```
## [1]  2  5  9 15
```

Vectors

- Vector arithmetic

```
a/b
```

```
## [1] 1.000 1.500 1.250 0.875
```

```
c(sum(a), mean(a), sd(a))
```

```
## [1] 16.000000  4.000000  2.581989
```

Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels.

```
data = c("East","West","East","North","North"); class(data)
```

```
## [1] "character"
```

```
factor_data = factor(data)
class(factor_data)
```

```
## [1] "factor"
```

```
levels(factor_data)
```

```
## [1] "East"  "North" "West"
```

Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it

```
# Create a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow
   list("green",12.3))
# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")
# Show the list.
print(list_data)
```

```
## $`1st Quarter`
## [1] "Jan" "Feb" "Mar"
##
## $A_Matrix
##      [,1] [,2] [,3]
## [1,]    3    5   -2
## [2,]    9    1    8
##
## $`A Inner list`
## $`A Inner list`[[1]]
## [1] "green"
##
## $`A Inner list`[[2]]
```

Lists

```
# Add element at the end of the list.
list_data[4] <- "New element"
print(list_data[4])
```

```
## [[1]]
## [1] "New element"
```

```
# Remove the last element.
list_data[4] <- NULL

list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")

# Convert the lists to vectors.
v1 <- unlist(list1)
v2 <- unlist(list2)
c(v1,v2)
```

```
## [1] "1"   "2"   "3"   "Sun" "Mon" "Tue"
```

Lists

- Merging Lists

```
# You can merge many lists into one list by placing all the lists ins
list1 <- list(1,2,3)
list2 <- list("Sun","Mon","Tue")
# Merge the two lists.
merged.list <- c(list1,list2)
print(merged.list)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] "Sun"
##
## [[5]]
## [1] "Mon"
##
```

Matrix

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

- Construct Matrices

```
A = matrix(c(2,4,3,1,5,7), #the data elements
       nrow =2, #number of rows
       ncol = 3, #number of columns
       byrow = TRUE) #fill by row first
rownames(A) = c("row1","row2")
colnames(A) = c("col1","col2","col3")
A
```

```
##      col1 col2 col3
## row1    2    4    3
## row2    1    5    7
```

Matrix

- Submatrix

```
A[,c(1,3)] #first and third columns
```

```
##      col1 col3
## row1    2    3
## row2    1    7
```

```
A[,-2] #all columns except the second
```

```
##      col1 col3
## row1    2    3
## row2    1    7
```

Matrix

- Combine existing matrices

```
B = matrix(c(2,4,3,1,5,7),nrow=3,ncol=2)
C = matrix(c(7,4,2),nrow=3,ncol=1)
D = matrix(c(6,2),nrow=1,ncol=2)

cbind(B,C) # Column Bind
```

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

```
rbind(B,D) # Row Bind
```

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

Matrix

- Useful matrix operations

```
x = matrix(1:4,2,2)
c(nrow(x), ncol(x))# dimensions of x; could also use dim(x)
```

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

```
c(rowSums(x), colSums(x), rowMeans(x), colMeans(x))# sums
```

```
## [1] 4.0 6.0 3.0 7.0 2.0 3.0 1.5 3.5
```

Arrays

Matrix is confined to two dimensions, and arrays can be of any number of dimensions.

```
column.names <- c("COL1","COL2","COL3")
row.names <- c("ROW1","ROW2","ROW3")
matrix.names <- c("Matrix1","Matrix2")
a <- array(c('green','yellow'),dim = c(3,3,2))
# Take these vectors as input to the array.
result <- array(c('green','yellow'),dim = c(3,3,2),dimnames = list(rc
print(result)
```

```
## , , Matrix1
##
##       COL1     COL2     COL3
## ROW1 "green"  "yellow" "green"
## ROW2 "yellow" "green"  "yellow"
## ROW3 "green"  "yellow" "green"
##
## , , Matrix2
##
##       COL1     COL2     COL3
## ROW1 "yellow" "green"  "yellow"
## ROW2 "green"  "yellow" "green"
## ROW3 "yellow" "green"  "yellow"
```

Arrays

```r
# Create two vectors of different lengths.
vector1 <- c(5,9,3)
vector2 <- c(10,11,12,13,14,15)

# Take these vectors as input to the array.
new.array <- array(c(vector1,vector2),dim = c(3,3,2))
print(new.array)
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    5   10   13
## [2,]    9   11   14
## [3,]    3   12   15
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    5   10   13
## [2,]    9   11   14
## [3,]    3   12   15
```

Arrays

- Calculations Across Array Elements

  apply(x, margin, fun)

  - x is an array.
  - margin is the name of the data set used.
  - fun is the function to be applied across the elements of the array.

```
# Use apply to calculate the sum of the rows across all the matrices.
result <- apply(new.array, c(1), sum)
print(result)
```

```
## [1] 56 68 60
```

Data Frame

Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical.

- Construct Data Frame

```
df = data.frame(col1 = c("this","is","text"), col2 = c(FALSE, TRUE, F
colnames(df) = c("Text", "isVerbal", "Length")
rownames(df) = c("Word1","Word2","Word3")
head(df)
```

```
##       Text isVerbal Length
## Word1 this    FALSE      2
## Word2   is     TRUE      0
## Word3 text    FALSE      4
```

- Convert a matrix to data.frame

```
cc = matrix(1:6,3,2)
df = as.data.frame(cc)
df
```

```
##   V1 V2
## 1  1  4
```

Data Frame

- print a data.frame

```
head(df, 6)
```

```
##   V1 V2
## 1  1  4
## 2  2  5
## 3  3  6
```

```
summary(df)
```

```
##        V1              V2
##  Min.   :1.0   Min.   :4.0
##  1st Qu.:1.5   1st Qu.:4.5
##  Median :2.0   Median :5.0
##  Mean   :2.0   Mean   :5.0
##  3rd Qu.:2.5   3rd Qu.:5.5
##  Max.   :3.0   Max.   :6.0
```

Data Frame

- Adding columns

```
df$newx = c("IT","HR","Finance")
head(df)
```

```
##   V1 V2    newx
## 1  1  4      IT
## 2  2  5      HR
## 3  3  6 Finance
```

- Removing columns

```
df$newx = NULL
head(df)
```

```
##   V1 V2
## 1  1  4
## 2  2  5
## 3  3  6
```

Thanks

- Have a good day!