Introduction to Computer Programming with R (FOR 6934)

Qing Zhao (School of Forest Resources & Conservation, qing.zhao@ufl.edu)
Daijiang Li (Department of Wildlife Ecology & Conservation, dli1@ufl.edu)
Denis Valle (School of Forest Resources & Conservation, drvalle@ufl.edu)

Basic Information

- Self introduction
- Why use R
  - Free
  - Powerful (and fast growing)
  - Programming skills are transferable
- Structure of lectures
- Homework and grading
- Office hour

Learning outcomes

- Data type and index
- Loops
- Customized functions
- Figures & tables
- Enjoy computer programming

Class one

- help() and several other basic functions
- Understand data types and structures
- Convert data types and structures

help() function

- You can learn to use all of the functions in R by using help() function

- You can also use the help() function to learn about data sets and packages

- Use it to explore the infinite potential of R

- Other useful functions

  - str()
  - apropos()

Sample code

Use help() to understand functions

```
help(mean)
help(apply)
help(ifelse)
```

Sample code

? is the same as help()

```
help(sum)
?sum
```

Sample code

Use help() to understand control-flow constructs

```
help("if")
help("else")
help("for")
```

Sample code

Use help() to understand data sets and packages

```
help(cars)
help(package="boot")
```

Sample code

str() and apropos()

```
str(mean)
```

```
## function (x, ...)
```

```
apropos("mean")
```

```
##  [1] ".colMeans"     ".rowMeans"     "colMeans"      "kmeans"
##  [5] "mean"          "mean.Date"     "mean.default"  "mean.difftime"
##  [9] "mean.POSIXct"  "mean.POSIXlt"  "rowMeans"      "weighted.mean"
```

Data types

Data type	Example
Logical	TRUE (T), FALSE (F)
Numeric (double)	4, 3.5, 1e2
Integer	2L, 1e2L
Complex	5+3i
Character	“Tom”, “1.2”, “FALSE”

Data structures

Data structure	Explanation
Vector	A set of values of a single type
Matrix	A two-dimensional rectangular data set of a single type of data
Array	A multiple dimensional data set of a single type of data
Data frame	A number of vectors of equal length (like a matrix). Each vector (column, also called variable) can have a different data type
List	An R object containing different types of data, functions, and even another list

Sample code

Create vectors with logical values

```
c(TRUE, TRUE, FALSE)
```

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

Sample code

Create vectors with numeric & integer values

```
c(5.2, 3.3, 4.9, 6.4, 8.1)
```

```
## [1] 5.2 3.3 4.9 6.4 8.1
```

```
c(1L, 2L, 3L)
```

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

```
1:3
```

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

Sample code

Create vectors with character values

```
c("red", "blue", "pink")
```

```
## [1] "red"  "blue" "pink"
```

```
c("3.2", "1.5", "7.3")
```

```
## [1] "3.2" "1.5" "7.3"
```

```
c("TRUE", "FALSE")
```

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

Sample code

Vectors do not allow for mixed types of values

```r
c(FALSE, 1.52, -2L, TRUE)
```

```
## [1]  0.00  1.52 -2.00  1.00
```

```r
c(FALSE, 1.52, -2L, TRUE, 'a')
```

```
## [1] "FALSE" "1.52"  "-2"    "TRUE"  "a"
```

Sample code

Name vectors

```r
a <- c(3.52, 5.73, 4.83)
a
```

```
## [1] 3.52 5.73 4.83
```

```r
b = c(TRUE, FALSE, TRUE, FALSE)
b
```

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

```r
c <- c("a", "b", "c") # '' also works, pick your style
c
```

```
## [1] "a" "b" "c"
```

Sample code

Create and name matrices

```r
mat1 <- matrix(c("dog", "cat", "horse", "gator"), ncol=2, nrow=2, byrow=FALSE)
mat1
```

```
##      [,1]  [,2]
## [1,] "dog" "horse"
## [2,] "cat" "gator"
```

```r
mat2 <- matrix(c("dog", "cat", "horse", "gator"), ncol=2, nrow=2, byrow=TRUE)
mat2
```

```
##      [,1]    [,2]
## [1,] "dog"   "cat"
## [2,] "horse" "gator"
```

Sample code

Create and name arrays

```r
arr <- array(1:30, dim=c(3,5,2))
arr
```

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

Sample code

Create and name data frames

```
animal <- c('dog', 'cat', 'horse', 'tortoise')
age <- c(15, 1, 3, 100)
large <- c(FALSE, FALSE, TRUE, TRUE)
dat <- data.frame(animal, age, large)
dat
```

```
##     animal age large
## 1      dog  15 FALSE
## 2      cat   1 FALSE
## 3    horse   3  TRUE
## 4 tortoise 100  TRUE
```

Sample code

Create and name lists

```
animal <- c('dog', 'cat')
age <- c(15, 1)
dat <- data.frame(animal, age)
lst <- list(animal=animal, age=age, dat=dat)
lst
```

```
## $animal
## [1] "dog" "cat"
##
## $age
## [1] 15  1
##
## $dat
##   animal age
## 1    dog  15
## 2    cat   1
```

Functions for identifying data types and structures

- class()
- typeof()
- str()

Functions for specific data types

- is.logical()
- is.numeric()
- is.integer()
- is.character()

Functions for specific data structures

- is.vector()
- is.matrix()
- is.array()
- is.data.frame()
- is.list()

Sample code

Identify data types and structure in vectors

```
a <- c(3.25, 6.73, 5.5)
class(a)
```

```
## [1] "numeric"
```

```
typeof(a)
```

```
## [1] "double"
```

```
str(a)
```

```
##  num [1:3] 3.25 6.73 5.5
```

Sample code

Identify specific data types in vectors

```
is.numeric(a)
```

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

```
is.character(a)
```

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

Sample code

Identify specific data structures in vectors

```
is.vector(a)
```

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

```
is.matrix(a)
```

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

Sample code

Identify data types and structures in matrices

```
b <- matrix(c('dog', 'cat', 'horse', 'gator'), nrow=2, ncol=2)
class(b)
```

```
## [1] "matrix"
```

```
typeof(b)
```

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

```
str(b)
```

```
##  chr [1:2, 1:2] "dog" "cat" "horse" "gator"
```

Sample code

Identify specific data types in matrices

```
is.character(b)
```

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

```
is.logical(b)
```

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

Sample code

Identify specific data structures in matrices

```
is.data.frame(b)
```

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

```
is.matrix(b)
```

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

Sample code

Identify data structures in data frames and lists

```
dat <- data.frame(animal=c('dog', 'cat'), age=c(15,1))
lst <- list(dat, fruit=c('apple', 'peach'))

class(dat)
```

```
## [1] "data.frame"
```

```
class(lst)
```

```
## [1] "list"
```

Some comments for data types and structures

- Always use class() to identify data structures first. It will give you the data type if it is a vector, or the data structure otherwise
- If it is a matrix or array, use typeof() to identify the data type in it
- typeof() does not make much sense for data frames and lists, since they allow multiple types of data
- Use class() and typeof() on the columns in a data frame or components in a list

Functions for converting data types

- as.logical()
- as.numeric()
- as.integer()
- as.character()

Functions for converting data structures

- as.vector()
- as.matrix()
- as.data.frame()

Sample code

Convert data to logical values

```
as.logical(c(1.5, -2.2, 1000, 0))
```

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

```
as.logical(c(1L, 0L, 5e2L, -3L))
```

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

```
as.logical(c('a', 'b', 'c'))
```

```
## [1] NA NA NA
```

Sample code

Convert data to numeric values

```
as.numeric(c(TRUE, TRUE, FALSE, TRUE))
```

```
## [1] 1 1 0 1
```

```
as.numeric(c('dog', 'cat', 'horse'))
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA NA NA
```

```
as.numeric(c('7.1', '-3.5', '1.2'))
```

```
## [1]  7.1 -3.5  1.2
```

Sample code

Convert data to integer values

```
as.integer(c(31.2, 43.9, -22.76))
```

```
## [1]  31  43 -22
```

```
as.integer(c(FALSE, TRUE))
```

```
## [1] 0 1
```

Sample code

Nested converting

```
as.logical(c('1.5','0.0','-3.2'))
```

```
## [1] NA NA NA
```

```
as.logical(as.numeric(c('1.5','0.0','-3.2')))
```

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

Sample code

Convert data to character values

```
as.character(c(TRUE,FALSE,TRUE))
```

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

```
as.character(c(1.5, 2.2, 7.6))
```

```
## [1] "1.5" "2.2" "7.6"
```

```
as.character(c(5L, 22L))
```

```
## [1] "5"  "22"
```

Sample code

Convert matrices to vectors

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

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

```
as.vector(mat)
```

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

Sample code

Convert vectors to a matrices

```
vec <- c(1, 3, 5, 7, 9, 11)
vec
```

```
## [1]  1  3  5  7  9 11
```

```
as.matrix(vec)
```

```
##      [,1]
## [1,]    1
## [2,]    3
## [3,]    5
## [4,]    7
## [5,]    9
## [6,]   11
```

Sample code

Convert vectors to a matrices, cont'd

```
vec <- c(1, 3, 5, 7, 9, 11)
vec
```

```
## [1]  1  3  5  7  9 11
```

```
matrix(vec, nrow=2, ncol=3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    3    7   11
```

Sample code

Convert matrices to data frames

```
mat <- matrix(1:12, ncol=6, nrow=2)
mat
```

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

```
as.data.frame(mat)
```

```
##   V1 V2 V3 V4 V5 V6
## 1  1  3  5  7  9 11
## 2  2  4  6  8 10 12
```

Summary

- Use help() and several other functions to learn about functions and data sets
- Understand data types and structures
- Convert data types and structures
- Learn by practicing

Thank you and see you next class