STAT 408
Applied Regression Analysis

Miles Xi

Department of Mathematics and Statistics

Loyola University Chicago

Fall 2022

Introduction to R Programming Language: Part 2

Data Structure

- Data structure in R is the data type containing more than one element

- Major data structures in R
  - Vector
  - List
  - Matrix
  - Array
  - Data frame
  - Factor

Vector

- A vector is a list of items with the same type

- c() function combines items to a vector; separate the items by a comma

```
# Vector of strings
fruits <- c("banana", "apple", "orange")

# Print fruits
fruits
```

Vector

- Create a vector that combines numerical values

```
# Vector of numerical values
numbers <- c(1, 2, 3)

# Print numbers
numbers
```

- Use the “:” operator to create a vector with numerical values

```
# Vector with numerical values in a sequence
numbers <- 1:10
numbers
```

Vector

- A vector of logical values

```
log_values <- c(TRUE, FALSE, TRUE, FALSE)
log_values
```

- Use the length() function to find out how many items a vector has

```
fruits <- c("banana", "apple", "orange")
length(fruits)
```

Vector

- Use the sort() function to sort items in a vector

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")
numbers <- c(13, 3, 5, 7, 20, 2)

sort(fruits)          # Sort a string
sort(numbers)         # Sort numbers
```

Vector

- Use vector_name[index] to access the vector items

```
fruits <- c("banana", "apple", "orange")
# Access the first item (banana)
fruits[1]

# Access the first and third item (banana and orange)
fruits[c(1, 3)]

# Access all items except for the first item
fruits[-1]

# Access the first two items
fruits[1:2]
```

Vector

- Change the item in the vector

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")

# Change "banana" to "pear"
fruits[1] <- "pear"

# Print fruits
fruits
```

Vector

- To repeat items, use the rep() function:

```
# repeat each value
repeat_each <- rep(c(1,2,3), each = 3)
repeat_each

# repeat the whole sequence
repeat_times <- rep(c(1,2,3), times = 3)
repeat_times

# repeat each value independently:
repeat_indepent <- rep(c(1,2,3), times = c(5,2,1))
repeat_indepent
```

Vector

- Use the seq() function to make a flexible sequenced vector

```
numbers <- seq(from = 0, to = 100, by = 20)
numbers
```

- Use %in% operation to check if one item is in the vector

```
0%in%numbers
```

Factor

- Factor is a vector with categorical items

```
# Create a factor
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz",
"Rock", "Jazz"))

# print the unique values of factors
levels(music_genre)
```

Factor

- The “levels” argument inside the factor() function can manually set the levels

```
music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz",
"Rock", "Jazz"), levels = c("Classic", "Jazz", "Pop", "Rock", "Other"))

levels(music_genre)
```

- Other operations of factor is similar to vector

Matrix

- A matrix is a two-dimensional data type with columns and rows
- The elements in the matrix must have the same type

```
# Create a 3 by 2 numerical matrix
thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
thismatrix

# Create a 2 by 2 string matrix
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol =
2)
thismatrix
```

Matrix

- Access the matrix items by using matrix_name[row_index, col_index]

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape",
"pineapple", "pear", "melon", "fig"), nrow = 3, ncol = 3)

thismatrix[1, 2]   # element at first row and second column

thismatrix[2, ]   # elements from second row

thismatrix[, 2]   # elements from second column

thismatrix[c(1,2),]   # elements from first two rows

thismatrix[,2:3]   # elements from last two columns
```

Matrix

- Use cbind() function to add additional columns in a matrix

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape",
"pineapple", "pear", "melon", "fig"), nrow = 3, ncol = 3)

newmatrix <- cbind(thismatrix, c("strawberry", "blueberry", "raspberry"))
newmatrix
```

- Use rbind() function to add additional rows in a matrix:

```
newmatrix <- rbind(thismatrix, c("strawberry", "blueberry", "raspberry"))
newmatrix
```

Matrix

- Use cbind() and rbind() functions can also combine two matrices

```
Matrix1 <- matrix(c("apple", "banana", "cherry", "grape"), nrow = 2, ncol = 2)
Matrix2 <- matrix(c("orange", "mango", "pineapple", "watermelon"), nrow = 2,
ncol = 2)

# Combine by rows
Matrix_Combined_row <- rbind(Matrix1, Matrix2)
Matrix_Combined_row

# Combine by columns
Matrix_Combined_column <- cbind(Matrix1, Matrix2)
Matrix_Combined_column
```

Matrix

- Use dim() function to find the number of rows and columns in a matrix

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
dim(thismatrix)
```

- Use length() function to find the total number of elements in a matrix

```
length(thismatrix)
```

Matrix

- Use for loop to iterate each element in a matrix

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol =
2)

# double loop
for (rows in 1:nrow(thismatrix)) {
  for (columns in 1:ncol(thismatrix)) {
    print(thismatrix[rows, columns])
  }
}
```

Data Frame

- A data frame is similar to a two-dimensional matrix, but its columns can have different data types

```
# Create a data frame with Training, Pulse, and Duration as column names
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)
```

Data Frame

- Use the summary() function to show the summary statistic for each column

```
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)
summary(Data_Frame)
```

Data Frame

- Use single brackets [ ], double brackets [[ ]] or $ to access columns from a data frame

```
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)
# access first column
Data_Frame[,1]
Data_Frame[["Training"]]
Data_Frame$Training
```

Data Frame

- Use rbind() function to add new rows in a data frame:

```
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)
# Add a new row
New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))
```

Data Frame

- Use cbind() function to add new columns in a data frame:

```
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

# Add a new column with a name Steps
New_col_DF <- cbind(Data_Frame, Steps = c(1000, 6000, 2000))
```

List

- A list is similar to a vector but can contain items of different types

```
# Create a list containing strings, numbers, vectors and logical values
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)
list_data
```

List

- The list elements can be given names and they can be accessed using these names

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Access the first element of the list.
print(list_data[[1]])

# Access the list element using the name of the element.
print(list_data$A_Matrix)
```

List

- Use append() function to add an item to the end of the list

```
thislist <- list("apple", "banana", "cherry")
append(thislist, "orange")

# add "orange" to the list after "banana" (index 2)
append(thislist, "orange", after = 2)
```

List

- Loop through the list items by using a for loop

```
thislist <- list("apple", "banana", "cherry")

for (x in thislist) {
  print(x)
}
```

- The same operation applies to vector

Array

- Array is the data type with more than two dimensions (tensor)
- The elements in the array must have the same type
- Array is not very common in statistical analysis, but very popular in engineering

Package

- R packages are a collection of R functions
- R packages provide functionality beyond basic R
- R packages need to be installed first and imported before being used

```
# install package ggplot2
install.packages('ggplot2’)

# import ggplot2
library(ggplot2)

# check documents
?ggplot2
```

Working Directory

- Working Directory is the folder we save the data file
- By default, we read and write data from the work directory

```
# Get and print current working directory.
print(getwd())

# set working directory
setwd("C:/Users/mxi1/Desktop")

# read a cvs file from working directory
data <- read.csv('pima.csv')
class(data)
```