Kalinga University

Faculty of Computer Science & Information Technology

Course- BCAAIML
Subject- R Programming
Subject Code – BCAAIML401 **Sem- IV**

Unit 4

S3 Classes – S4 Classes – Managing your objects – Input/output – accessing keyboard and monitor – reading and writing files – accessing the internet – String Manipulation – Graphics – Creating Graphs – Customizing Graphs – Saving Graphs to files – Creating Three-Dimensional plots.

In **R Programming**, object-oriented programming can be used with different classes such as **S3** and **S4** classes. R also provides extensive functionality for managing objects, input/output operations, manipulating strings, and creating and customizing graphics. Here's a breakdown of the various topics you're interested in:

S3 and S4 Classes in R

R provides two primary systems for object-oriented programming: **S3** and **S4** classes.

S3 Classes

S3 is a simpler, informal system used in R. It allows you to define your own class by simply assigning an object to a class and writing methods for it.

- **Creating an S3 object**: You can create an S3 object by assigning it a class.

  ```
  person <- list(name = "John", age = 25)
  class(person) <- "person"
  ```

- **Creating an S3 method**: Methods are defined by naming them according to the class of the object.

  ```
  print.person <- function(x) {
    cat("Name:", x$name, "Age:", x$age, "\n")
  }
  ```

- **Calling the method**:

```
print(person)
```

S4 Classes

S4 is a more formal and rigorous system for defining objects and methods. It allows you to specify the types of arguments and the class structure.

- **Creating an S4 class**: You define the class with setClass.

  ```
  setClass("person", slots = c(name = "character", age = "numeric"))
  ```

- **Creating an instance of an S4 object**:

  ```
  john <- new("person", name = "John", age = 25)
  ```

- **Creating an S4 method**: Define methods for an S4 object using setMethod.

  ```
  setMethod("show", "person", function(object) {
    cat("Name:", object@name, "Age:", object@age, "\n")
  })
  ```

- **Calling the method**:

  ```
  show(john)
  ```

Managing Your Objects

Objects in R can be stored and managed using lists, data frames, and custom classes (as shown above). You can access and modify object attributes using $ for lists and @ for S4 objects.

Input/Output in R

R allows easy interaction with the console (keyboard) and files (monitor, text files, etc.).

Accessing the Keyboard (User Input)

You can take user input from the keyboard using the readline() function.

```
name <- readline(prompt = "Enter your name: ")
cat("Hello, ", name, "!\n")
```

Writing to Monitor (Output)

You can print information to the console using cat() or print().

```
cat("Hello World!\n")
print("This is a print statement")
```

Reading and Writing Files

- **Reading from a file**:

  ```
  data <- read.csv("file.csv")
  ```

- **Writing to a file**:

  ```
  write.csv(data, "output.csv")
  ```

- **Reading a text file**:

  ```
  lines <- readLines("file.txt")
  ```

- **Writing to a text file**:

  ```
  writeLines(lines, "output.txt")
  ```

Accessing the Internet in R

R provides functions for downloading data from the internet, such as download.file() and url(), and packages like **httr** or **RCurl**.

- **Downloading a file from the internet**:

  ```
  download.file("https://example.com/data.csv", "data.csv")
  ```

- **Reading data from a URL**:

  ```
  data <- read.csv(url("https://example.com/data.csv"))
  ```

String Manipulation

R provides a variety of functions for working with strings, such as nchar(), substr(), strsplit(), and regular expressions.

- **String length**:

  ```
  nchar("Hello World")
  ```

- **Subsetting a string**:

  ```
  substr("Hello World", 1, 5)
  ```

- **Splitting a string**:

  strsplit("Hello,World", ",")

- **Regular expressions**:

  gsub("world", "R", "Hello world!")

Graphics in R

R provides several packages for creating and customizing graphs. The primary graphics system is base R plotting, and additional functionality is available through packages like **ggplot2**.

Creating Graphs

- **Basic Plot**:

  plot(x = 1:10, y = rnorm(10), type = "b", main = "Basic Plot", xlab = "X-axis", ylab = "Y-axis")

- **Histograms**:

  hist(rnorm(100), main = "Histogram", xlab = "Values")

- **Boxplot**:

  boxplot(rnorm(100), main = "Boxplot", ylab = "Values")

Customizing Graphs

You can customize graphs by adding titles, labels, changing colors, and modifying other parameters.

- **Adding titles and labels**:

  plot(x = 1:10, y = rnorm(10), main = "Customized Plot", xlab = "X-axis", ylab = "Y-axis", col = "blue")

- **Adding grid lines**:

  plot(x = 1:10, y = rnorm(10))
  grid()

- **Adding multiple plots in one window**:

  par(mfrow = c(1, 2))

```
plot(1:10)
plot(rnorm(10))
```

Saving Graphs to Files

You can save plots to various formats (e.g., PNG, PDF, JPG).

- **Saving to a PNG file**:

```
png("plot.png")
plot(x = 1:10, y = rnorm(10))
dev.off()  # To close the PNG device
```

- **Saving to a PDF file**:

```
pdf("plot.pdf")
plot(x = 1:10, y = rnorm(10))
dev.off()
```

Creating Three-Dimensional Plots

R allows for 3D plotting using the persp() function in base R, or by using specialized packages like **plotly** or **rgl**.

- **3D plot using persp()**:

```
x <- seq(-5, 5, length = 50)
y <- seq(-5, 5, length = 50)
z <- outer(x, y, function(x, y) { sin(sqrt(x^2 + y^2)) })
persp(x, y, z, main = "3D Surface Plot")
```

- **3D Plot with plotly**:

```
library(plotly)
plot_ly(x = ~rnorm(100), y = ~rnorm(100), z = ~rnorm(100), type = "scatter3d", mode =
"markers")
```