Objects and Classes in R

Learn about Object-Oriented Programming (OOP) along with R's objects, different classes like S3 and S4, along with its construction, creating its generic function with examples and many more.

Jun 2020 · 9 min read

Olivia Smith
Senior developer at CMARIX TechnoLabs. Writes about trending technologies like AI & ML

TOPICS

R Programming

Data Science

Object-Oriented Programming (OOP) is a programming paradigm in where different methods are used to design software around data or objects rather than using functions. A method is just a function, talked about in the OOP context. It is object-oriented because all the processing revolves around the objects and fields. Every object has different attributes and behavior.

In Object-Oriented programming, the whole program can be divided into different classes to quickly understand the program execution. R is a functional programming language, and OOP will help manage large system problems. OOP used to manage GUI applications, most likely web-applications. Object-Oriented Programming is good for building tools for data analysis but bad for data analysis itself.

Systems of OOP

There are mainly two major systems of OOP, which are described below:

- **S3 Classes:** These let you overload the functions.
- **S4 Classes:** These let you limit the data as it is quite difficult to debug the program

Note: These classes will be discussed in detail in the later section.

Objects in R

Objects are the instance of the class. Also, everything in R is an object and to know more look at **Data types in R.** They also can have their attributes like class, attributes,dimnnames, names, etc.

For example, objects are assigned a value using <-, consider the following piece of code where 'a1' and 'a2' are the variables where the numeric value is assigned to it.

```
a1 <- 10
a2 <- 20
```

Explain code

OpenAI

An object can be assigned a set of numbers where 'a3' is the object containing a vector of different numbers.

```
a3 <- c(10,20,30)
```

Explain code OpenAI

Here **c** contains the numbers,10, 20, 30 into a vector.

Let's take a real-life example like a student, which has attributes of name, age, class, and gender, the teacher, has a name, age, and gender.

```
st_name <- "andrew"
st_age <- 16
st_class <- 10
st_gender <- "male"

t_name <- "angelina"
t_age <- 26
t_gender <- "female"

student <- c(st_name, st_age, st_class, st_gender)
teacher <- c(t_name, t_age, t_gender)


print(student)
print(teacher)
```

Explain code OpenAI

```
[1] "andrew" "16"     "10"     "male"
[1] "angelina" "26"       "female"
```

Explain code OpenAI

The above code gives the following output:

```
"andrew" "16" "10" "male"
"angelina" "26"  "female"
```

Explain code OpenAI

In the above code, you can see the example where there are different types of objects. The 'student' and 'teacher' object, also called a variable, hold type 'character' and 'double.'

Class in R

Class is the blueprint that helps to create an object and contains its member variable along with the attributes.

As discussed earlier in the previous section, there are two classes of R, S3, and S4.

S3 Class

These classes help in overloading functions by splitting them into generic and methods.

functions = generic + methods

S3 is very different from other programming languages like C#, Java, etc. Because it's straightforward to implement, it uses only the first argument to dispatch. That's why it doesn't require too much knowledge as a programmer's perspective.

S3 works by defining a strict naming convention of generic dot class.

```
generic.class
```

Explain code OpenAI

The arguments of methods should contain arguments of generic, and method should include a **...arg**.

Constructing along with Example of S3 Class

You will be demonstrated to construct an S3 class in the following example, which shows the creation of the list where components are being passed along with proper class name followed by the result to be printed.

The following example shows 'studentBio' contains a list where components are passed as a list where 'student_name,' 'student_age,' and 'student_contact' are being given. The class is named 'Student Info,' and 'studentBio' is the newly created class whose values are displayed below.

```
studentBio <- list(studentName = "Harry Potter", studentAge = 19, studentCon t=
class(studentBio) <- "StudentInfo"
studentBio
```

Explain code OpenAI

```
$studentName
[1] "Harry Potter"

$studentAge
[1] 19

$studentContact
[1] "London"

attr(,"class")
[1] "StudentInfo"
```

Explain code OpenAI

The above code gives the output as below:

```
$studentName
 "Harry Potter"

$studentAge
19
```

```
$studentContact
"London"

attr(,"class")
"StudentInfo"
```

Explain code OpenAI

If you are not comfortable about writing function in R, then read a **A Tutorial on Using Functions in R!**

Creating your Own Generic Method in S3

You will be implementing your generic function below by using the following code.

The code below shows the generic function called 'contact' where the function passing with the object is assigned to it. Also, 'UseMethod' is used with the generic function called 'contact' is passed where 'dispatching' of method occurs, which is the process where the result varies based on the arguments passed.

```
contact <- function(object) {
UseMethod("contact")
}
```

Explain code OpenAI

Let's make a method for the particular class that you've created earlier named 'StudentInfo.' The method 'contact' followed by dot notation along with the class 'StudentInfo' is used where the function body contains the access of the attribute 'studentContact' is done by using '$.'

```
contact.StudentInfo <- function(object) {
cat("Your contact is", object$studentContact, "\n")
}
contact(studentBio)
```

Explain code OpenAI

```
Your contact is London
```

Explain code OpenAI

The above code uses object$studentContact , where the object passed as an argument to function along with '$' to access attribute 'studentContact,' which results in output as below.

```
Your contact is London
```

Explain code OpenAI

S4 Class

S4 Class is stricter, conventional, and closely related to Object-Oriented concepts. The classes are represented by the formal class definitions of S4. More specifically, S4 has setter and getter functions for methods and generics. As compared to the S3 class, S4 can be able to facilitate multiple dispatches.

Let us see the above-discussed classes of R by creating them and discuss with an example.

An S4 class is defined using the set class function.

```
setClass()
```

Explain code OpenAI

Constructing along with Example of S4 Class

As discussed earlier, the S4 class is defined by the setClass() method. You will use the setClass() method to create and define the S4 class. We will use a subroutine to verify that the data is consistent (validation) and also set the default values (the paradigm).

Following is the code to define the class and its slots:

The following is used to define a class, also called 'prototype', where the slots are defined where the class name is 'employee', name is a character, id as a numeric, and contact as a character.

```
setClass("employee", slots=list(name="character", id="numeric", contact="cha te
```

Explain code OpenAI

The following code can create the specific instance or objects of S4.

Let's create an object using 'new' with the class name inside as 'employee' with the slots like name, id, and contact are filled with a value.

```
obj <- new("employee",name="Steven", id=1002, contact="West Avenue")
obj
```

Explain code OpenAI

```
An object of class "employee"
Slot "name":
[1] "Steven"

Slot "id":
[1] 1002

Slot "contact":
[1] "West Avenue"
```

Explain code OpenAI

```
An object of class "employee"
Slot "name":
"Steven"

Slot "id":
1002

Slot "contact":
"West Avenue"
```

Explain code OpenAI

The above output shows that the object created was of class "employee" and the Slot with "name" called "Steven" along with Slot "id" with value 1002 and Slot "contact" with value is

"West Avenue".

Function and commands

For checking whether an object is an S4 object or not following two commands can be used:

The command 'is.object(obj)'checks whether a variable or instance refers to an object or not.

```
is.object(obj)
```

Explain code OpenAI

TRUE

The above code gives the following output as below:

```
TRUE
```

Explain code OpenAI

The command 'is.S4(obj)'checks, whether a variable or instance is an S4 object or not.

```
isS4(obj)
```

Explain code OpenAI

TRUE

The above code gives the following output as below:

```
TRUE
```

Explain code OpenAI

Accessing and Modifying the slot value

Accessing the slot

The slot can be accessed using the "@" instead of "$".

'obj@name' is used where the object name along with the slot name is to access the name 'Steven' and is similarly done to 'obj@id'.

```
obj@name
obj@id
```

Explain code OpenAI

'Steven'

1002

The above code gives the following output:

```
'Steven'
1002
```

Explain code OpenAI

Modifying the slot

Let's modify the slot value just by using the assignment operator("<-") where the 'obj@contact' is assigned with a new value as "North Avenue".

```
obj@contact <- "North Avenue"
obj
```

Explain code OpenAI

```
An object of class "employee"
Slot "name":
[1] "Steven"

Slot "id":
[1] 1002

Slot "contact":
[1] "North Avenue"
```

Explain code OpenAI

The above code gives the following output:

```
An object of class "employee"
Slot "name":
"Steven"

Slot "id":
1002

Slot "contact":
"North Avenue"
```

Explain code OpenAI

The above output shows that the Slot "contact" has changed it's value previously "West Avenue" to "North Avenue".

Alternatively, there is an option to use 'slot()' to change the value.

'slot(obj, "id")' indicates the slot "id" value to ve changed with the assignment of 1004, and the result is printed with 'obj'.

```
slot(obj,"id") <- 1004
obj
```

Explain code OpenAI

```
An object of class "employee"
Slot "name":
[1] "Steven"

Slot "id":
[1] 1004

Slot "contact":
[1] "North Avenue"
```

Explain code OpenAI

```
An object of class "employee"
Slot "name":
"Steven"

Slot "id":
1004

Slot "contact":
"North Avenue"
```

Explain code OpenAI

The above output shows that the Slot "id" has changed its value previously 1002 to 1004.

Creating your own Generic Method in S4

Let's create a function that helps print the output according to our needs, where the 'setMethod()' and the generic function called 'show()' is used. Also, all the slots' value is accessed by '@', and their names are printed out. Finally, the output is obtained with the help of 'obj'.

```
setMethod("show",
"employee",
function(object) {
cat("Name:",object@name, "\n")


)
obj
```

Explain code OpenAI


```
Name: Steven
Id: 1004
Contact: North Avenue
```

Explain code OpenAI

```
Name: Steven
Id: 1004
Contact: North Avenue
```

Explain code OpenAI

The above output shows that the results are printed as in the form of Slot names along with respective Slot values.

Congratulations

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you've learned about Object-Oriented Programming(OOP) along with R's objects, different classes like S3 and S4, along with its construction, creating its generic function with examples and many more.

If you would like to learn more about R, check DataCamp's courses:

Object-Oriented Programming with S3 and R6 in R

TOPICS