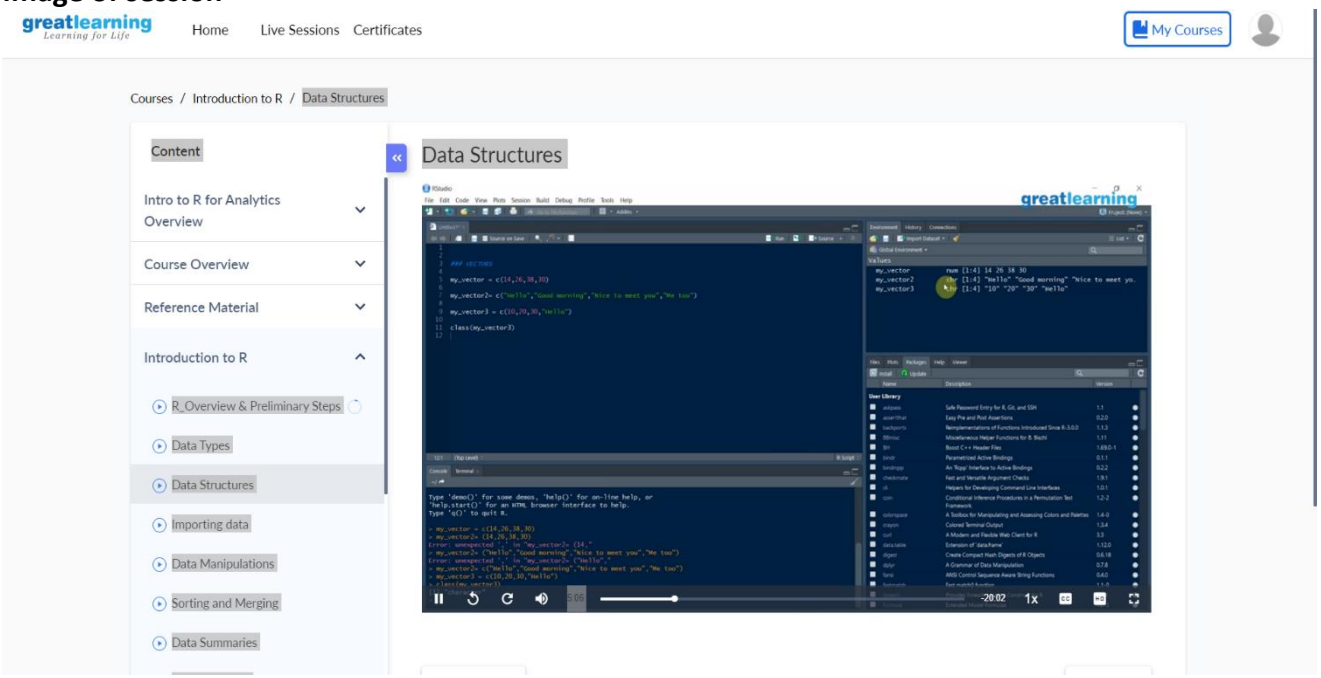
DAILY ASSESSMENT FORMAT

Date:	10 JUNE 2020	Name:	PAVITHRAN S
Course:	R PROGRAMMING	USN:	4AL17EC068
Topic:	DATA STRUCTURES IN R.	Semester & Section:	6TH B
Github Repository:	Pavithran		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

A variable provides us with named storage that our programs can manipulate. A variable in R can store an atomic vector, group of atomic vectors or a combination of many Robjects. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Variable Name	Validity	Reason
var_name2.	valid	Has letters, numbers, dot and underscore
var_name%	Invalid	Has the character '%'. Only dot(.) and underscore allowed.

2var_name	invalid	Starts with a number
.var_name, var.name	valid	Can start with a dot(.) but the dot(.)should not be followed by a number.
.2var_name	invalid	The starting dot is followed by a number making it invalid.
_var_name	invalid	Starts with _ which is not valid

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using **print()** or **cat()** function. The **cat()** function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2 <- c("learn","R")

# Assignment using rightward operator.
c(TRUE,1) -> var.3

print(var.1)
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
```

When we execute the above code, it produces the following result −

```
[1] 0 1 2 3
var.1 is  0 1 2 3
var.2 is  learn R
var.3 is  1 1
```

Note − The vector c(TRUE,1) has a mix of logical and numeric class. So logical class is coerced to numeric class making TRUE as 1.

Data Type of a Variable

In R, a variable itself is not declared of any data type, rather it gets the data type of the R - object assigned to it. So R is called a dynamically typed language, which means that we can change a variable's data type of the same variable again and again when using it in a program.

```
var_x <- "Hello"
cat("The class of var_x is ",class(var_x),"\n")

var_x <- 34.5
cat("  Now the class of var_x is ",class(var_x),"\n")
```

```
var_x <- 27L
cat("   Next the class of var_x becomes ",class(var_x),"\n")
```

When we execute the above code, it produces the following result −

```
The class of var_x is  character
   Now the class of var_x is  numeric
      Next the class of var_x becomes  integer
```

Finding Variables

To know all the variables currently available in the workspace we use the **ls()** function. Also the ls() function can use patterns to match the variable names.

```
print(ls())
```

When we execute the above code, it produces the following result −

```
[1] "my var"     "my_new_var" "my_var"     "var.1"
[5] "var.2"      "var.3"      "var.name"   "var_name2."
[9] "var_x"      "varname"
```

Note − It is a sample output depending on what variables are declared in your environment.

The ls() function can use patterns to match the variable names.

```
# List the variables starting with the pattern "var".
print(ls(pattern = "var"))
```

When we execute the above code, it produces the following result −

```
[1] "my var"     "my_new_var" "my_var"     "var.1"
[5] "var.2"      "var.3"      "var.name"   "var_name2."
[9] "var_x"      "varname"
```

The variables starting with **dot(.)** are hidden, they can be listed using "all.names = TRUE" argument to ls() function.

```
print(ls(all.name = TRUE))
```

When we execute the above code, it produces the following result −

```
[1] ".cars"        ".Random.seed" ".var_name"    ".varname"
".varname2"
[6] "my var"       "my_new_var"   "my_var"       "var.1"        "var.2"
[11]"var.3"        "var.name"     "var_name2."   "var_x"
```

Deleting Variables

Variables can be deleted by using the **rm()** function. Below we delete the variable var.3. On printing the value of the variable error is thrown.

```
rm(var.3)
print(var.3)
```

When we execute the above code, it produces the following result −

```
[1] "var.3"
```

```
Error in print(var.3) : object 'var.3' not found
```

All the variables can be deleted by using the **rm()** and **ls()** function together.

```
rm(list = ls())
print(ls())
```

When we execute the above code, it produces the following result −

```
character(0)
```