



SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

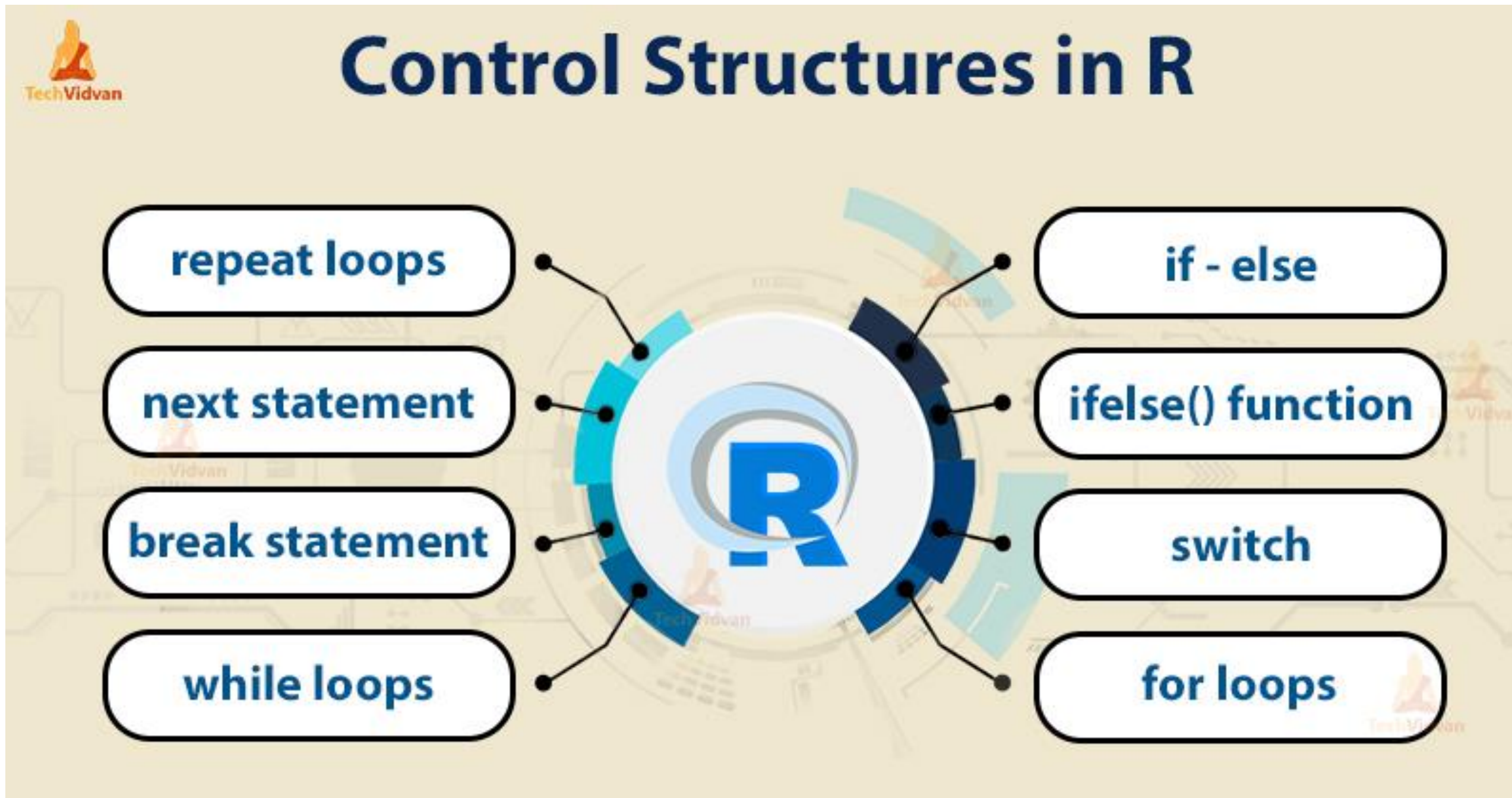


DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME :19CS407 DATA ANALYTICS WITH R
II YEAR /IV SEMESTER

Unit 4- R PROGRAMMING BASICS

Topic : Control Structures





Control Statements in R Programming



✓ Control statements are expressions used to control the execution and flow of the program based on the conditions provided in the statements. These structures are used to make a decision after assessing the variable. In this article, we'll discuss all the control statements with the examples.

- if condition
- if-else condition
- for loop
- nested loops
- while loop
- repeat and break statement
- return statement
- next statement



if condition



✓ This control structure checks the expression provided in parenthesis is true or not. If true, the execution of the statements in braces {} continues.

✓ **Syntax:**

```
if(expression)
```

```
{
```

```
Statements
```

```
...
```

```
....
```

```
}
```



if condition



✓ Example:

```
x <- 100
```

```
if(x > 10){  
  print(paste(x, "is greater than 10"))  
}
```

Output:

```
[1] "100 is greater than 10"
```



if-else condition



It is similar to if condition but when the test expression in if condition fails, then statements in else condition are executed.

Syntax:

```
if(expression){  
    statements  
    ....  
    ....  
}  
else{  
    statements  
    ....  
    ....  
}
```



if-else condition



Example:

```
x <- 5
```

```
# Check value is less than or greater than 10
```

```
if(x > 10){  
  print(paste(x, "is greater than 10"))  
}else{  
  print(paste(x, "is less than 10"))  
}
```

Output:

```
[1] "5 is less than 10"
```



for loop



It is a type of loop or sequence of statements executed repeatedly until exit condition is reached.

Syntax:

```
for(value in vector){  
    statements  
    ....  
    ....  
}
```




for loop



Example:

```
x <- letters[4:10]
```

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

Output:

```
[1] "d"  
[1] "e"  
[1] "f"  
[1] "g"  
[1] "h"  
[1] "i"  
[1] "j"
```



Nested loops

Nested loops are similar to simple loops. Nested means loops inside loop. Moreover, nested loops are used to manipulate the matrix.

Example:

```
# Defining matrix
m <- matrix(2:15, 2)

for (r in seq(nrow(m))) {
  for (c in seq(ncol(m))) {
    print(m[r, c])
  }
}
```

Output:

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
[1] 12
[1] 14
[1] 3
[1] 5
[1] 7
[1] 9
[1] 11
[1] 13
[1] 15
```



while loop



- ✓ while loop is another kind of loop iterated until a condition is satisfied. The testing expression is checked first before executing the body of loop.

Syntax:

```
while(expression)
{
Statement
....
...
}
```



while loop



Example:

```
x = 1
# Print 1 to 5
while(x <= 5){
print(x)
x = x + 1
}
```

Output:

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```



Repeat loop and break statement

repeat is a loop which can be iterated many number of times but there is no exit condition to come out from the loop. So, break statement is used to exit from the loop. break statement can be used in any type of loop to exit from the loop.

Syntax:

```
repeat {  
    statements  
    ....  
    ....  
    if(expression) {  
        break  
    }  
}
```



Repeat loop and break statement



Example:

```
x = 1
```

```
# Print 1 to 5
```

```
repeat{
```

```
  print(x)
```

```
  x = x + 1
```

```
  if(x > 5){
```

```
    break
```

```
  }
```

```
}
```

Output:

```
[1] 1
```

```
[1] 2
```

```
[1] 3
```

```
[1] 4
```

```
[1] 5
```



return statement

return statement is used to return the result of an executed function and returns control to the calling function.

Syntax:

```
return(expression)
```



return statement

Example:

Checks value is either positive, negative or zero

```
func <- function(x){  
  if(x > 0){  
    return("Positive")  
  }else if(x < 0){  
    return("Negative")  
  }else{  
    return("Zero")  
  }  
}
```

func(1)

func(0)

func(-1)

Output:

[1] "Positive"

[1] "Zero"

[1] "Negative"



return statement

Next statement is used to skip the current iteration without executing the further statements and continues the next iteration cycle without terminating the loop.

Example:

```
# Defining vector
x <- 1:10
# Print even numbers
for(i in x){
  if(i%%2 != 0){
```

```
    next #Jumps to next loop
  }
  print(i)
}
```

Output:

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```



Assessment 1





References



1. João Moreira, Andre Carvalho, Tomás Horvath – “A General Introduction to Data Analytics” – Wiley -2018
2. <https://www.geeksforgeeks.org/control-statements-in-r-programming/>

Thank You