**Loops & Control Structures**

**If else in R**

x = -1;

if ( x > 0)

{

cat (x, "is a positive number")

} else

{

cat (x, "is a negative number")

}

**Nested If else in R**

if(x < 0 ) {

cat (x ,"is negative number ")

} else if (x >0) {

cat(x,"is a positive number")

} else

printf(x ,"is equal to 0")

**ifelse() Function**

Most functions in R take vector. This vectorization is easier than applying the same function to each element in the vector.

a = c(5,7,2,9)

ifelse(a %% 2 == 0,"even","odd")

**SWITCH STATEMENT**

A switch statement is a selection control mechanism that allows the value of an expression to change the control flow of program execution. A switch statement is a selection control mechanism that allows the value of an expression to change the control flow of program execution. We can have any number of cases. There is no default case.

**1) Based on Index**

If the cases are values like a character vector, and the expression is evaluated to a number than the expression's result is used as an index to select the case.

x =switch (4, "first","second","third","fourth")

print (x)

**2) Based on Matching Value**

When the cases have both case value and output value like ["case\_1"="value1"], then the expression value is matched against case values. If there is a match with the case, the corresponding value is the output.

val1 = 6

val2 = 7

val3 = "s"

result = switch(

val3,

"a"= cat("Addition =", val1 + val2),

"d"= cat("Subtraction =", val1 - val2),

"r"= cat("Division = ", val1 / val2),

"s"= cat("Multiplication =", val1 \* val2),

"m"= cat("Modulus =", val1 %% val2),

"p"= cat("Power =", val1 ^ val2)

)

**LOOPS**

**For loop**

fruits<- c("apple", "banana", "cherry")  
for (x in fruits) {  
  print(x)  
}

**Repeat Loop**

result <**-** c("Hello World")

i <**-** 1

repeat {

**print**(result)

   i <**-** i **+** 1

**if**(i >5) {

**break**

   }

}

**While Loop**

i <- 1  
while (i < 6) {  
  print(i)  
  i <- i + 1  
}

**Loop Control Statements**

Loop control statements are also called jump statements. Loop control statements change their execution from their normal sequence.

**break statement**

x <- 1:5

for (val in x) {

if (val == 3){

break

}

print(val)

}

**Next statement**

Next statement in R is used to skip any remaining statements in the loop and continue the execution of the program.

val = 6:12

for (i in val)

{

if (i == 8)

{

# test expression

next

}

print(i)

}

**FUNCTIONS & PACKAGES IN R**

**Functions without Arguments**

x <- function() {  
  print("Hello World!")  
}

**Functions with Arguments, Named Arguments & Default Arguments**

**power <- function(x,y=5) {**

**result = x^y**

**cat ( "\n", result)**

**}**

**power(2,3)**

**power(y=3,x=2)**

**power(2)**

**Recursive Functions**

**rec\_fac <- function(x){**

**if(x==0 || x==1)**

**{**

**return(1)**

**}**

**else**

**{**

**return(x\*rec\_fac(x-1))**

**}**

**}**

**rec\_fac(6)**

**PACKAGES**

Packages in [R Programming language](https://www.geeksforgeeks.org/introduction-to-r-programming-language/) are a set of R functions, compiled code, and sample data. These are stored under a directory called “library” within the R environment. By default, R installs a group of packages during installation. Once we start the R console, only the default packages are available by default. Other packages that are already installed need to be loaded explicitly to be utilized by the R program that’s getting to use them.

To check the library locations containing R Packages

.libPaths()

To get the list of all packages installed using the following command

library()

To get all the packages loaded in R environment, use the following command

search()

**Installing a Package**

There are two ways to install R Package. One is installing directly from CRAN directory and another is downloading to the local system and installing it manually.

**To install directly from CRAN**

install.packages(“reshape”)

**To install from the local system**

**Eg**

install.packages(“E:/reshape\_3.98.zip”,repos = NULL, type = “source”)

**Loading a Package**

Before a package can be used in the code, it must be loaded to the current R environment. We also need to load a package that is already installed but not available in the current environment. To load a package

**library(“package\_name”, lib.loc = “path to library”)**