

# R\_Basics

- Vedanti Borate

## Installation and Pre-requisites

To set up and configure R programming using RStudio, follow these steps:

1. **Install R:** First, you need to install R on your system. You can download it from the Comprehensive R Archive Network (CRAN) at <https://cran.r-project.org/> (<https://cran.r-project.org/>). Choose the version that matches your operating system (Windows, MacOS, Linux), download it, and follow the installation instructions.
2. **Install RStudio:** RStudio is an integrated development environment (IDE) for R. It provides a user-friendly interface and tools to help you use R more effectively. You can download RStudio from <https://www.rstudio.com/products/rstudio/download/> (<https://www.rstudio.com/products/rstudio/download/>). Again, choose the version that matches your operating system, download it, and follow the installation instructions.
3. **Configure RStudio:** After installing RStudio, you can configure it to suit your needs. Here are some basic configurations you might want to make:
  - **Set your working directory:** This is the directory where RStudio will look for files and save your work. You can set it by going to Tools → Global Options → General → Default working directory .
  - **Choose your CRAN mirror:** This is the server from which you'll download packages. You can set it by going to Tools → Global Options → Packages → CRAN mirror .
  - **Manage your packages:** R packages are collections of functions and data sets developed by the community. You can install, update, and manage your packages by going to Tools → Install Packages .
4. **Start using RStudio:** Now you're ready to start using RStudio. You can write your R code in the script editor, and then run it by clicking the Run button or pressing Ctrl+Enter (or Cmd+Enter on a Mac). The results will appear in the console.

## Commenting in R

R doesn't support Multi-line and Documentation comments. It only supports single-line comments drafted by a '#' symbol. Multi-line commenting : select lines > control/command + shift + C

# R Operators

## Arithmetic Operators

```
vec1 <- c(0, 2)
vec2 <- c(2, 3)

cat("Addition =", vec1 + vec2)
```

```
## Addition = 2 5
```

```
cat("Subtraction =", vec1 - vec2)
```

```
## Subtraction = -2 -1
```

```
cat("multiplication =", vec1 * vec2)
```

```
## multiplication = 0 6
```

```
cat("division =" ,vec1 /vec2)
```

```
## division = 0 0.6666667
```

```
cat("division remainder=" ,vec1 %% vec2)
```

```
## division remainder= 0 2
```

```
cat("power=", vec1 ^ vec2)
```

```
## power= 0 8
```

## Logical Operators

```
cat("power=", vec1 & vec2)
```

```
## power= FALSE TRUE
```

```
cat("power=", vec1 | vec2)
```

```
## power= TRUE TRUE
```

```
cat("power=", !vec1)
```

```
## power= TRUE FALSE
```

```
#alternatives  
cat("and op=", vec1[1] & vec2[1])
```

```
## and op= FALSE
```

```
cat("and and op=", vec1[1] && vec2[1])
```

```
## and and op= FALSE
```

```
cat("OR", vec1[1] || vec2[1])
```

```
## OR TRUE
```

## Assignment Operators

```
vec1 <- c(2:5)
c(2:5) ->> vec2
vec3 <<- c(2:5)
vec4 = c(2:5)
c(2:5) -> vec5
```

```
cat ("vector 1 :", vec1, "\n")
```

```
## vector 1 : 2 3 4 5
```

```
cat("vector 2 :", vec2, "\n")
```

```
## vector 2 : 2 3 4 5
```

```
cat ("vector 3 :", vec3, "\n")
```

```
## vector 3 : 2 3 4 5
```

```
cat("vector 4 :", vec4, "\n")
```

```
## vector 4 : 2 3 4 5
```

```
cat("vector 5 :", vec5)
```

```
## vector 5 : 2 3 4 5
```

# Data types

```
# Logical  
print(class(TRUE))
```

```
## [1] "logical"
```

```
# Integer  
print(class(3L))
```

```
## [1] "integer"
```

```
# Numeric  
print(class(10.5))
```

```
## [1] "numeric"
```

```
# Complex  
print(class(1+2i))
```

```
## [1] "complex"
```

```
# Character  
print(class("12-04-2020"))
```

```
## [1] "character"
```

```
#type verification  
#is.data_type(object)  
print(is.numeric("12-04-2020"))
```

```
## [1] FALSE
```

```
#type conversion  
#as.data_type(object)  
print(as.numeric(TRUE))
```

```
## [1] 1
```

## Variable methods

```
#finding datatype of variables  
print(class("Hi"))
```

```
## [1] "character"
```

```
#listing variables in workspace  
print(ls())
```

```
## [1] "vec1" "vec2" "vec3" "vec4" "vec5"
```

```
#remove variables  
#rm(variable)  
  
#global local variable  
global = 5  
display = function() {  
  global=2  
  
}  
display()  
print(global) #displays outside value
```

```
## [1] 5
```

## Reading Input

```
#var1 = readline(prompt = "Enter your name : ");  
#d = scan(what = double())
```

## Printing Output

```
x <- "GeeksforGeeks"  
#single variable  
x
```

```
## [1] "GeeksforGeeks"
```

```
#string and variable  
print(x)
```

```
## [1] "GeeksforGeeks"
```

```
cat("cat method ",x,9)
```

```
## cat method  GeeksforGeeks 9
```

```
paste0("paste method ",x,9)
```

```
## [1] "paste method GeeksforGeeks9"
```

```
#displaying message
message(x, " is best")
```

```
## GeeksforGeeks is best
```

```
#printing decimals
x <- 15 / 7
print(x, digits = 3)
```

```
## [1] 2.14
```

## Control Statments

### IF-else

```
# if(expression){
#     statements
#     ....
#     ....
# }
var='abc'
if (var == 'abc'){
    print(var)
}else {
    print("not found")
}
```

```
## [1] "abc"
```

### for loop



```
# for(value in vector){  
#     statements  
#     ....  
#     ....  
# }  
  
for (val in (1:5)){  
    print(val)  
}
```

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

## while loop

```
# while(expression){  
#     statement  
#     ....  
#     ....  
# }  
  
i=1  
while(i<10){  
    print(i^2)  
    i =i+ 1  
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
## [1] 36
## [1] 49
## [1] 64
## [1] 81
```

## repeat loop

```
# repeat {
#   statements
#   ....
#   ....
#   if(expression) {
#     break
#   }
# }
x = 1

repeat{
print(x)
x = x + 1
if(x > 5){
  break
}
}
```

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