R Tutorial Basic

Md Delwar Hossain

2022-11-05

R Tutorial

1. R Syntax

```r
"Hello World"
```

```
## [1] "Hello World"
```

```r
5
```

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

```r
6
```

```
## [1] 6
```

```r
7
```

```
## [1] 7
```

```r
5 + 5
```

```
## [1] 10
```

```r
print("Delwar")
```

```
## [1] "Delwar"
```

```r
for (x in 1:10) {
  print(x)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

2. R Comments

```r
7 + 3 #This is a comment
```

```
## [1] 10
```

```r
"Delwar" # Another Comment
```

```
## [1] "Delwar"
```

3. R Variables

```r
name <- 'Delwar'
age <- 30
name
```

```
## [1] "Delwar"
```

```r
age
```

```
## [1] 30
```

4. Concatenate Elements

```r
paste(name, age)
```

```
## [1] "Delwar 30"
```

```r
text <- "I Love my Society"
paste("Ok ", text)
```

```
## [1] "Ok  I Love my Society"
```

```r
num1 <- 5
num2 <- 10

num1 + num2
```

```
## [1] 15
```

5. Multiple Variable

```
x <- y <- z <- 10
x
```

```
## [1] 10
```

```
y
```

```
## [1] 10
```

```
z
```

```
## [1] 10
```

6. Legal Variable Names

```
myvar = "Delwar"
my_var = "Delwar"
myVar = "Delwar"
MYVAR = "Delwar"
myvar2 = "Delwar"
.myvar = "Delwar"
```

7. Data Types

```
x <- 20
class(x)
```

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

```
y <- "Salay"
class(y)
```

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

```
i <- 9 + 4i
class(i)
```

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

```
z <- TRUE
class(z)
```

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

```
t = 10L
class(t)
```

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

```
c = 5i
class(c)
```

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

8. Type Conversion

```
#You can convert from one type to another type with the following functions
x <- 4L
y <- 5
class(x)
```

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

```
class(y)
```

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

```
a <- as.numeric(x)
b <- as.integer(y)

class(a)
```

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

```
class(b)
```

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

9. R Math

```
10 + 5
```

```
## [1] 15
```

```
10 - 4
```

```
## [1] 6
```

10. R Built in math function

```r
max(5, 10, 15)
```

```
## [1] 15
```

```r
min(5, 10, 15)
```

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

```r
sqrt(16)
```

```
## [1] 4
```

```r
abs(-10) # return positive number
```

```
## [1] 10
```

```r
ceiling(1.6)
```

```
## [1] 2
```

```r
ceiling(1.3)
```

```
## [1] 2
```

```r
floor(1.6)
```

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

```r
floor(1.3)
```

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

11. R String

```r
str = 'I am Delwar'
str1 <- 'I am HP'
str
```

```
## [1] "I am Delwar"
```

```r
str1
```

```
## [1] "I am HP"
```

```r
str2 <- 'I am writing a letter,
I am doing my duties,
You can buy someting
Learn R programming'

str2
```

```
## [1] "I am writing a letter,\nI am doing my duties,\nYou can buy someting\nLearn R programming"
```

```r
cat(str2)
```

```
## I am writing a letter,
## I am doing my duties,
## You can buy someting
## Learn R programming
```

12. String Length

```r
nchar(str)
```

```
## [1] 11
```

```r
nchar(str2)
```

```
## [1] 85
```

13. Check a string

```r
grepl('am', str2)
```

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

```r
grepl('not', str2)
```

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

```r
grepl('buy', str2)
```

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

14. Escape character to avoid error

```
str <- 'I am not working "hard", in my job'
str
```

```
## [1] "I am not working \"hard\", in my job"
```

```
cat(str)
```

```
## I am not working "hard", in my job
```

15. R Boolean

```
10 > 9
```

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

```
10 == 0
```

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

```
10 < 9
```

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

```
a <- 10
b <- 20
a > b
```

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

```
if (a > b) {
  print(a)
} else{
  print(b)

}
```

```
## [1] 20
```

16. R Oprators

```
a = 10
b = 20

a + b
```

```
## [1] 30
```

```r
a - b
```

```
## [1] -10
```

```r
a * b
```

```
## [1] 200
```

```r
a / b
```

```
## [1] 0.5
```

```r
a %% b
```

```
## [1] 10
```

```r
a^b
```

```
## [1] 1e+20
```

17. R Comparison Operators

```r
a == b
```

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

```r
a != b
```

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

```r
a > b
```

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

```r
a < b
```

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

```r
a >= b
```

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

```r
a <= b
```

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

18. R Logical Operators

```
a & b
```

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

```
a && b
```

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

```
x = 1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

19. R if. . . .else

```
a <- 5
b<- 10

if (a > b) {
  print("A is Greater than B")
}else{
  print("B is Greater than A")
}
```

```
## [1] "B is Greater than A"
```

20. elseif

```
a <- 5
b<- 10

if (a > b) {
  print("A is Greater than B")
}else if(a == b){
  print("A is equal to B")
}else{
  print("None")
}
```

```
## [1] "None"
```

21. Nested elseif

```r
x <- 41

if (x > 10) {
  print("Above ten")
  if (x > 20) {
    print("and also above 20!")
  } else {
    print("but not above 20.")
  }
} else {
  print("below 10.")
}
```

```
## [1] "Above ten"
## [1] "and also above 20!"
```

22. Use of AND/OR

```r
a <- 200
b <- 33
c <- 500

if (a > b & c > a) {
  print("Both conditions are true")
}
```

```
## [1] "Both conditions are true"
```

```r
if(a>b | c>a){
    print("At least one of the conditions is true")
}
```

```
## [1] "At least one of the conditions is true"
```

23. R While Loops

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

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

```r
i <- 1
while (i<6) {
  print(i)
  i <- i+1
  if(i==4){
    break
  }
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

With the next statement, we can skip an iteration without terminating the loop:

```r
i <- 1
while (i<6) {
  i <- i+1
  if(i==3){
    next
  }
  print(i)
}
```

```
## [1] 2
## [1] 4
## [1] 5
## [1] 6
```

```r
 dice <-1

while (dice<=6) {
  if(dice<6){
    print("Not Allowed")
  }else{
    print("Allowed")
  }
  dice<- dice + 1
}
```

```
## [1] "Not Allowed"
## [1] "Not Allowed"
## [1] "Not Allowed"
## [1] "Not Allowed"
## [1] "Not Allowed"
## [1] "Allowed"
```

24. R For Loop

```
for (x in 1:5) {
  print(x)
}
```

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

```
fruits <- list("apple","banana","cherry")
for (i in fruits) {
  print(i)
}
```

```
## [1] "apple"
## [1] "banana"
## [1] "cherry"
```

```
fruits <- list("apple","banana","cherry")
for (i in fruits) {
  if(i== "banana"){
    break
  }
  print(i)
}
```

```
## [1] "apple"
```

```
fruits <- list("apple","banana","cherry")
for (i in fruits) {
  if(i== "banana"){
    next
  }
  print(i)
}
```

```
## [1] "apple"
## [1] "cherry"
```

```
dice <- 1:6
for (x in dice) {
  if(x==6){
    print(paste("The dice number is", x, " Six"))
  }else{
    print(paste("The dice number is", x, "Not Six"))
  }
}
```

```
## [1] "The dice number is 1 Not Six"
## [1] "The dice number is 2 Not Six"
## [1] "The dice number is 3 Not Six"
## [1] "The dice number is 4 Not Six"
## [1] "The dice number is 5 Not Six"
## [1] "The dice number is 6  Six"
```

25. Nested For Loops

```r
adj <- list(1:5)
name <- list("Delwar","Rakib","Akbar","Karim","Rahim")
for (i in adj) {
  for(j in name){
    print(paste(i,j))
  }
}
```

```
## [1] "1 Delwar" "2 Delwar" "3 Delwar" "4 Delwar" "5 Delwar"
## [1] "1 Rakib" "2 Rakib" "3 Rakib" "4 Rakib" "5 Rakib"
## [1] "1 Akbar" "2 Akbar" "3 Akbar" "4 Akbar" "5 Akbar"
## [1] "1 Karim" "2 Karim" "3 Karim" "4 Karim" "5 Karim"
## [1] "1 Rahim" "2 Rahim" "3 Rahim" "4 Rahim" "5 Rahim"
```

26. R Functions

```r
my_function <- function(){
  print("Hi")
}
my_function()
```

```
## [1] "Hi"
```

```r
my_function <- function(fname){
  print(fname)
}
my_function("Delwar")
```

```
## [1] "Delwar"
```

```r
my_function("Peter")
```

```
## [1] "Peter"
```

```r
my_function("Shuvo")
```

```
## [1] "Shuvo"
```

```
my_function <- function(fname,age){
  print(paste(fname,age))
}
my_function("Delwar",29)
```

```
## [1] "Delwar 29"
```

```
my_function("Peter",27)
```

```
## [1] "Peter 27"
```

```
my_function("Shuvo",26)
```

```
## [1] "Shuvo 26"
```

27. Nested Functions

There are two ways to create a nested function

```
#call a function within another function

nested_function <- function(x,y){
  a <- x+y
  return(a)
}
nested_function(3,4)
```

```
## [1] 7
```

```
nested_function(nested_function(3,4),nested_function(3,4))
```

```
## [1] 14
```

```
#write a function within a function
out <- function(x){
  inner <- function(y){
    a<- x+y
    return(a)
  }
  return(inner)
}
result <- out(5)
result(3)
```

```
## [1] 8
```

28.Function Recursion

```r
tri_function <- function(k){
  if(k>0){
    res <- k+tri_function(k-1)
    print(res)
  }else{
    res = 0
    return(res)

  }
}
tri_function(6)
```

```
## [1] 1
## [1] 3
## [1] 6
## [1] 10
## [1] 15
## [1] 21
```

29. Global Variable

Can be used by everyone both inside and outside

```r
txt <- "Awesome"
my_function <- function(){
  paste("R is", txt)
}
my_function()
```

```
## [1] "R is Awesome"
```

```r
txt <- "Awesome"
my_function <- function(){
  txt = "Delwar"
  paste("R is", txt)
}
my_function()
```

```
## [1] "R is Delwar"
```

```r
txt
```

```
## [1] "Awesome"
```

```r
txt <- "Awesome"
my_function <- function(){
  txt <<- "Delwar"  #global assignment operator
  paste("R is", txt)
}
my_function()
```

```
## [1] "R is Delwar"
```

```
txt
```

```
## [1] "Delwar"
```

R Data Structure

1. R Vectors

```
fruits <- c("banana","apple","orange")
fruits
```

```
## [1] "banana" "apple"  "orange"
```

```
x = c(1,2,3,4,5)
x
```

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

```
x = c(1:5)
x
```

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

```
z = 4:10
z
```

```
## [1]  4  5  6  7  8  9 10
```

```
# Vector with numerical decimals in a sequence
numbers1 <- 1.5:6.5
numbers1
```

```
## [1] 1.5 2.5 3.5 4.5 5.5 6.5
```

```
# Vector with numerical decimals in a sequence where the last element is not used
numbers2 <- 1.5:6.3
numbers2
```

```
## [1] 1.5 2.5 3.5 4.5 5.5
```

```
# Vector of logical values
log_values <- c(TRUE, FALSE, TRUE, FALSE)
log_values
```

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

```r
#vector length
f = c(20:30)
length(f)
```

```
## [1] 11
```

```r
#Sort a vector
x = c(4,7,4,12,5,8,3)
sort(x)
```

```
## [1]  3  4  4  5  7  8 12
```

```r
#access a vector
x[2]
```

```
## [1] 7
```

```r
#access multiple elements
x[c(2,6)]
```

```
## [1] 7 8
```

```r
#change an item
x = c(4,7,4,12,5,8,3)
x[1] = 1
x
```

```
## [1]  1  7  4 12  5  8  3
```

```r
#repeat vector
x = rep(c(1,2,3), each=3)
x
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

```r
x = rep(c(1,2,3), times=3)
x
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```r
x = rep(c(1,2,3), times = c(2,3,4))
x
```

```
## [1] 1 1 2 2 2 3 3 3 3
```

```r
#generating sequence vector
num = 1:10
num
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
num = seq(from = 0, to= 10, by = 2)
num
```

```
## [1]  0  2  4  6  8 10
```

2. R List

```r
i = list("apple","banana","grape")
i
```

```
## [[1]]
## [1] "apple"
##
## [[2]]
## [1] "banana"
##
## [[3]]
## [1] "grape"
```

```r
#access list
i = list("apple","banana","grape")
i[2]
```

```
## [[1]]
## [1] "banana"
```

```r
#change the value
i = list("apple","banana","grape")
i[2]= "palm"
i
```

```
## [[1]]
## [1] "apple"
##
## [[2]]
## [1] "palm"
##
## [[3]]
## [1] "grape"
```

```r
i = list("apple","banana","grape")
length(i)
```

```
## [1] 3
```

```r
#check items if exists
i = list("apple","banana","grape")
"banana" %in% i
```

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

```
"cake" %in% i
```

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

```
#add list items
i = list("apple","banana","grape")
append(i,"orange")
```

```
## [[1]]
## [1] "apple"
##
## [[2]]
## [1] "banana"
##
## [[3]]
## [1] "grape"
##
## [[4]]
## [1] "orange"
```

```
i = list("apple","banana","grape")
append(i,"orange", after = 1)
```

```
## [[1]]
## [1] "apple"
##
## [[2]]
## [1] "orange"
##
## [[3]]
## [1] "banana"
##
## [[4]]
## [1] "grape"
```

```
#remove list items
i = list("apple","banana","grape")
j = i[-1]
j
```

```
## [[1]]
## [1] "banana"
##
## [[2]]
## [1] "grape"
```

```
#range of index
l = list('a','b','r','t')
l
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "b"
##
## [[3]]
## [1] "r"
##
## [[4]]
## [1] "t"
```

```
(l)[2:4]
```

```
## [[1]]
## [1] "b"
##
## [[2]]
## [1] "r"
##
## [[3]]
## [1] "t"
```

```
#join two list
l3 = c(i,l)
l3
```

```
## [[1]]
## [1] "apple"
##
## [[2]]
## [1] "banana"
##
## [[3]]
## [1] "grape"
##
## [[4]]
## [1] "a"
##
## [[5]]
## [1] "b"
##
## [[6]]
## [1] "r"
##
## [[7]]
## [1] "t"
```

```
length(l3)
```

```
## [1] 7
```

3. R Matrics

4. R Arrays

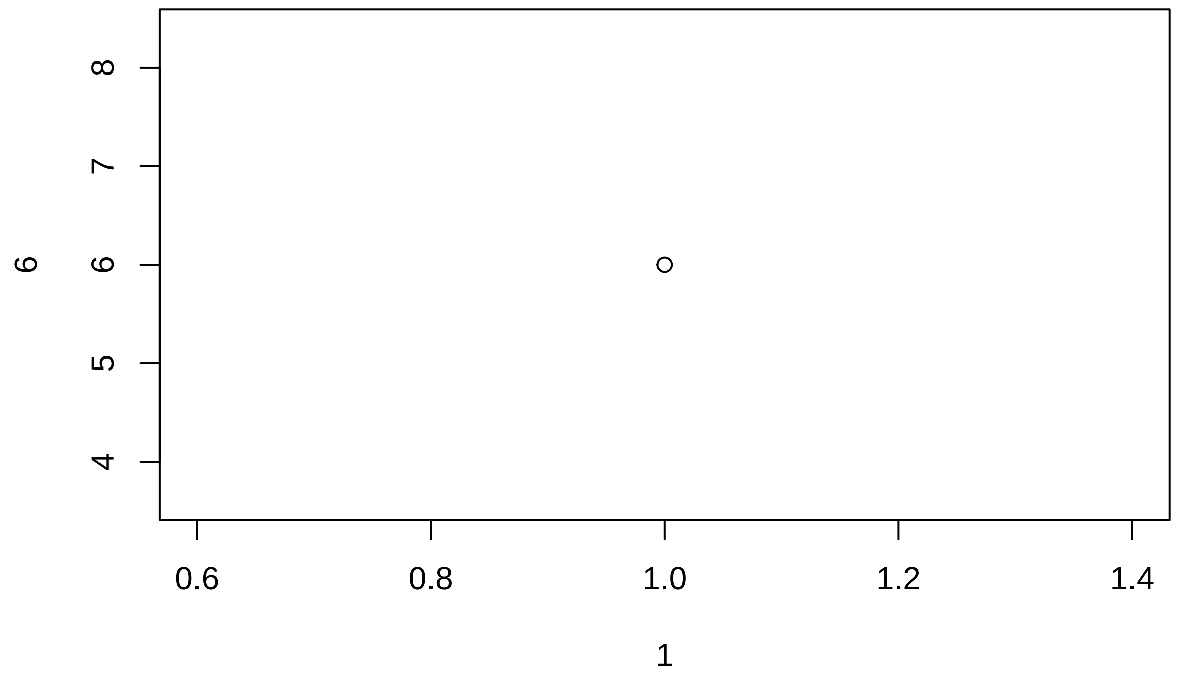
5. R Data Frames

6. Factors

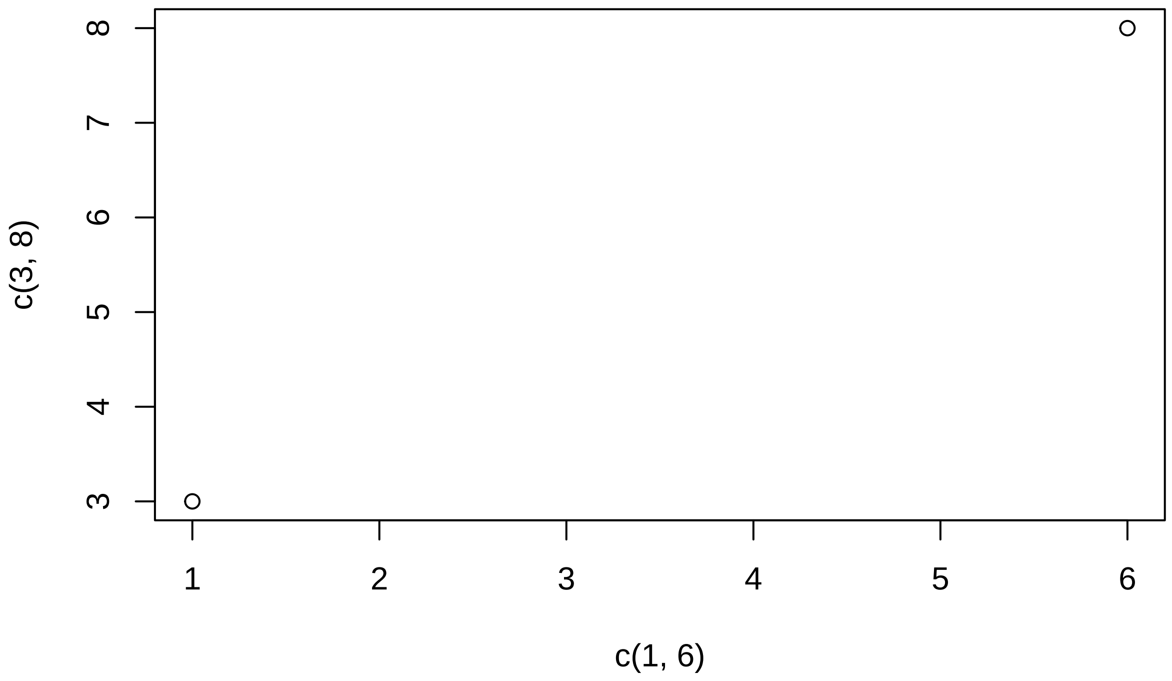
R Graphics

1. R Plot

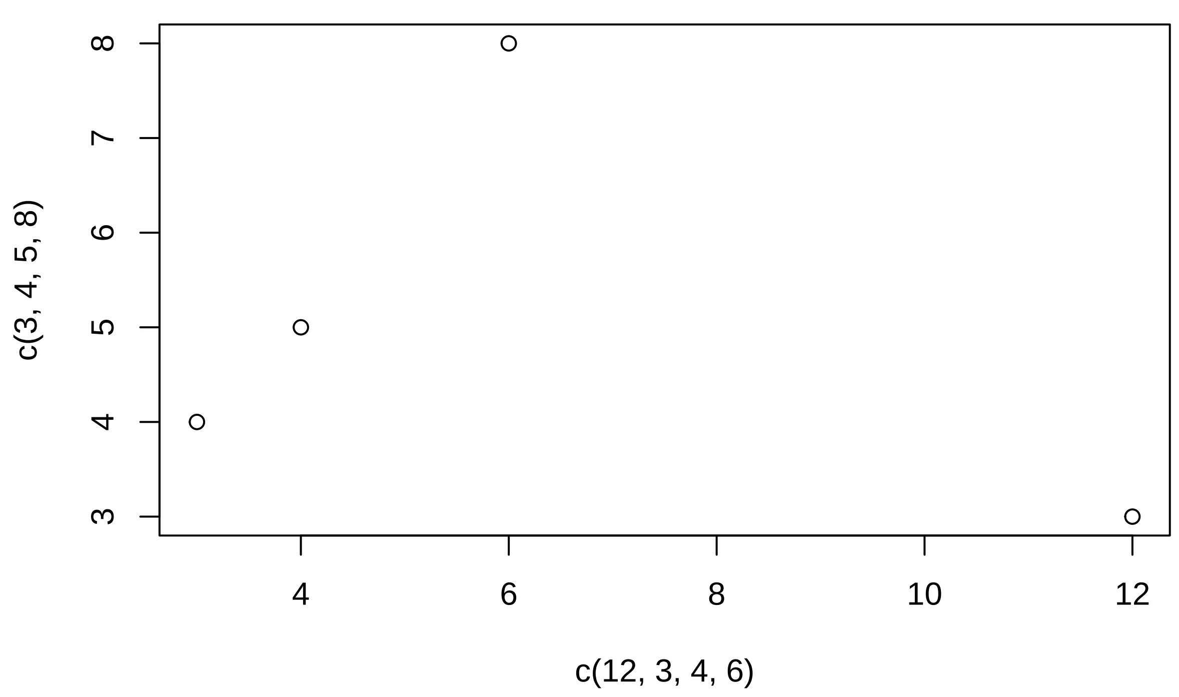
```
plot(1,6)
```



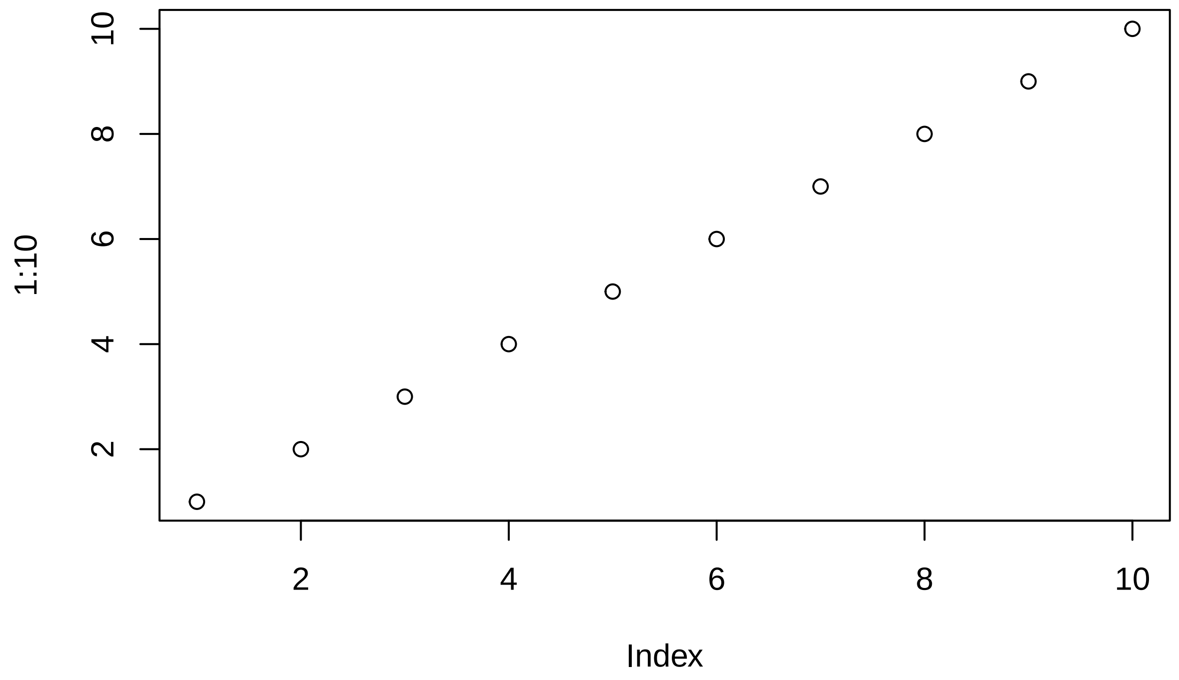
```
plot(c(1,6),c(3,8))
```



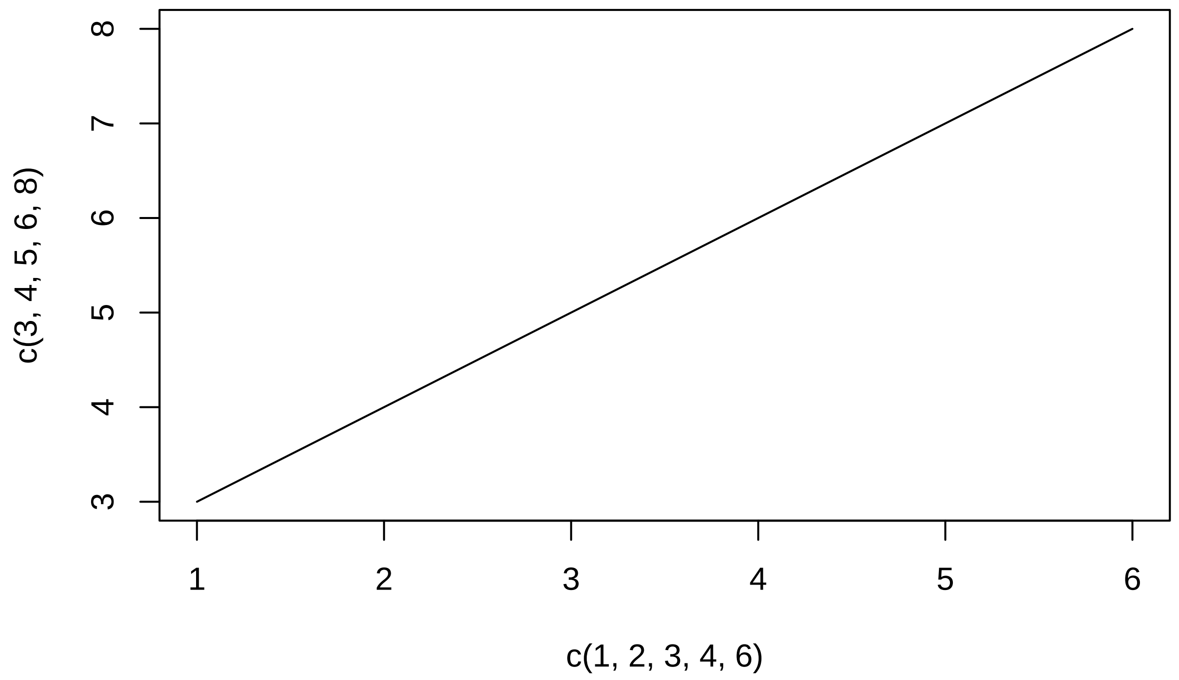
```
plot(c(12,3,4,6),c(3,4,5,8))
```



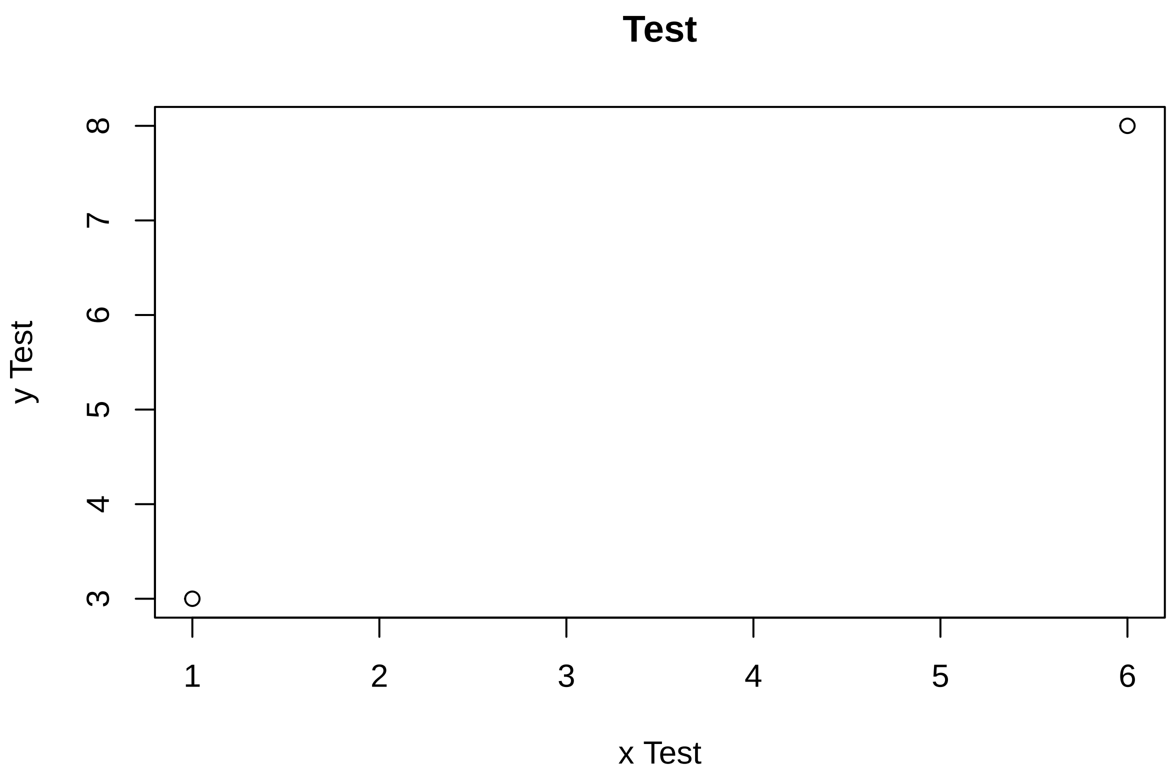
```
plot(1:10)
```



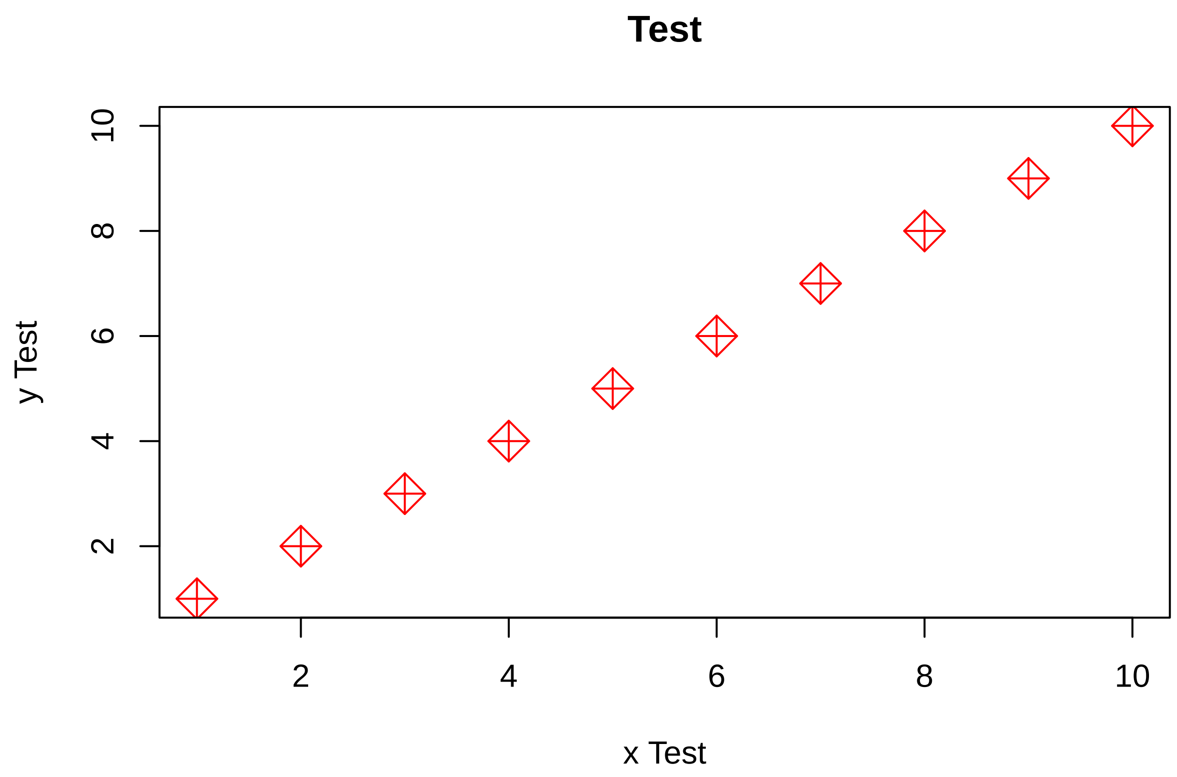
```r
plot(c(1,2,3,4,6),c(3,4,5,6,8), type = "l")
```



```
plot(c(1,6),c(3,8), main = "Test", xlab = "x Test", ylab = "y Test")
```

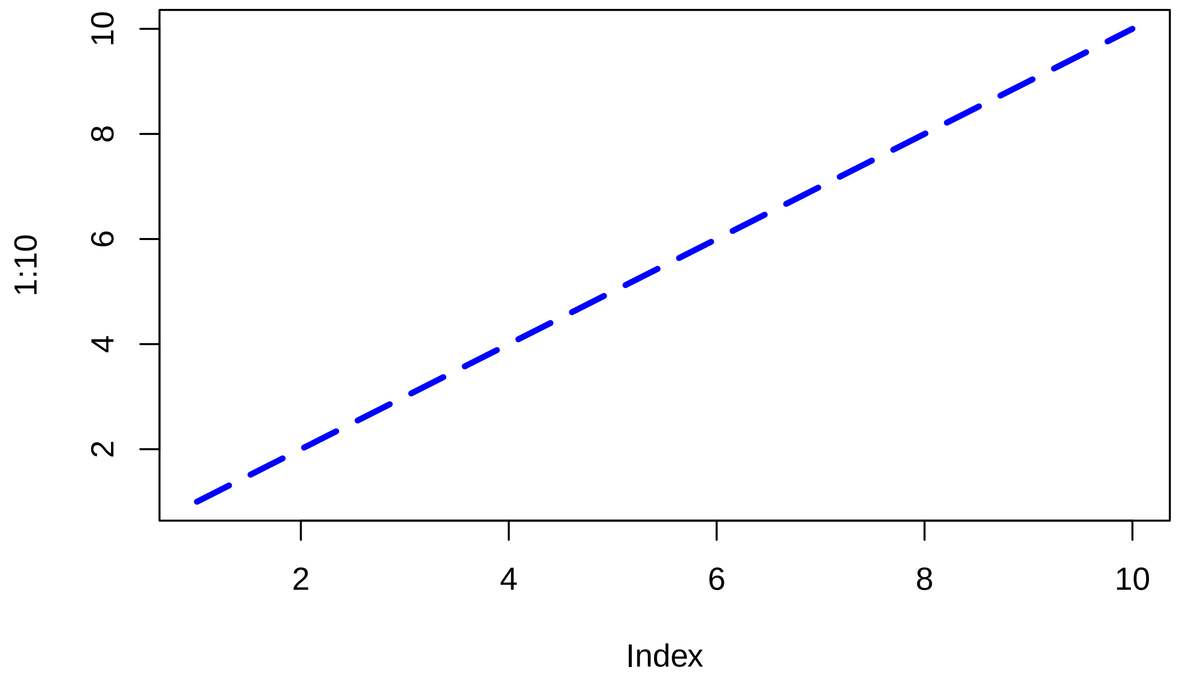


```
plot(1:10,col= "red",cex=2,pch=9, main = "Test", xlab = "x Test", ylab = "y Test")
```

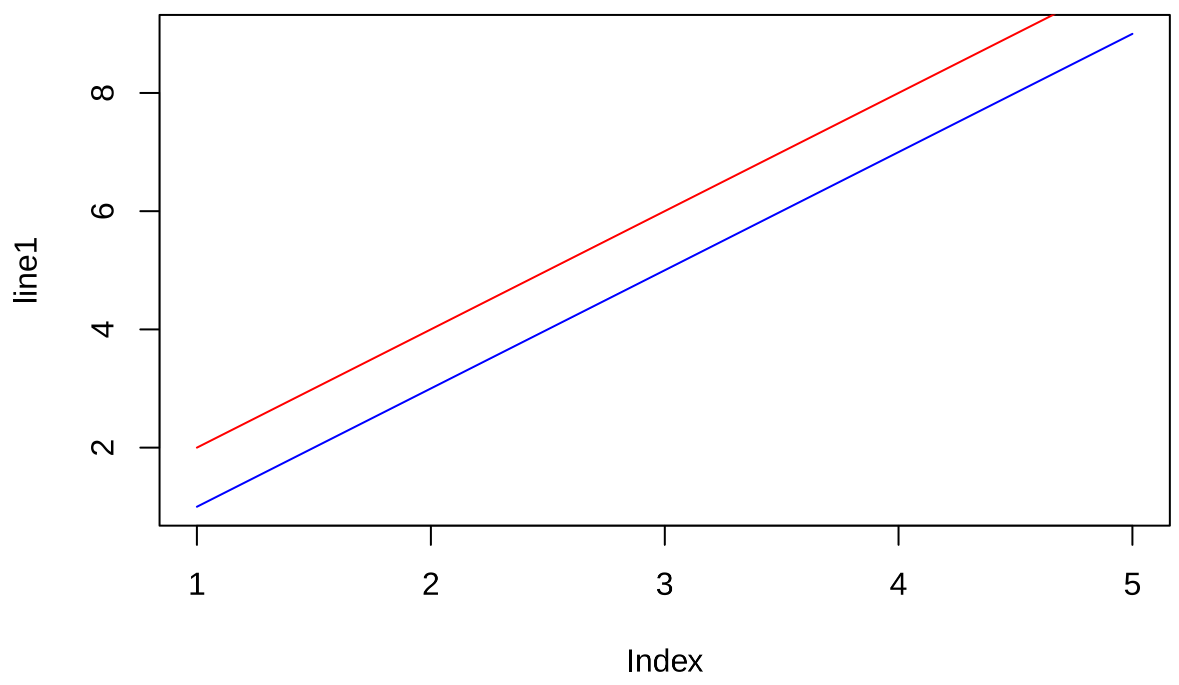


2. R Line

```
plot(1:10, type = "l", col ="blue", lwd = 3, lty = 5)
```

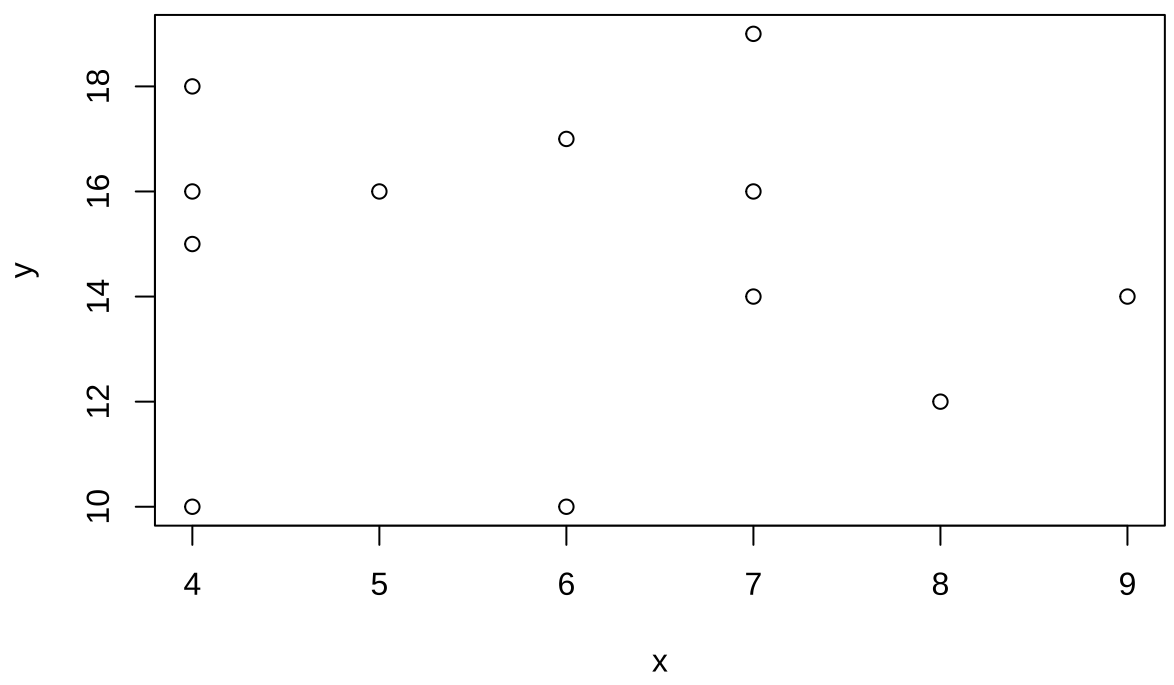


```
line1 = seq(from = 1, to = 10, by=2)
line2 = seq(from =2, to= 10, by =2)
plot(line1, type = "l", col ="blue")
lines(line2,type = "l",  col = "red")
```

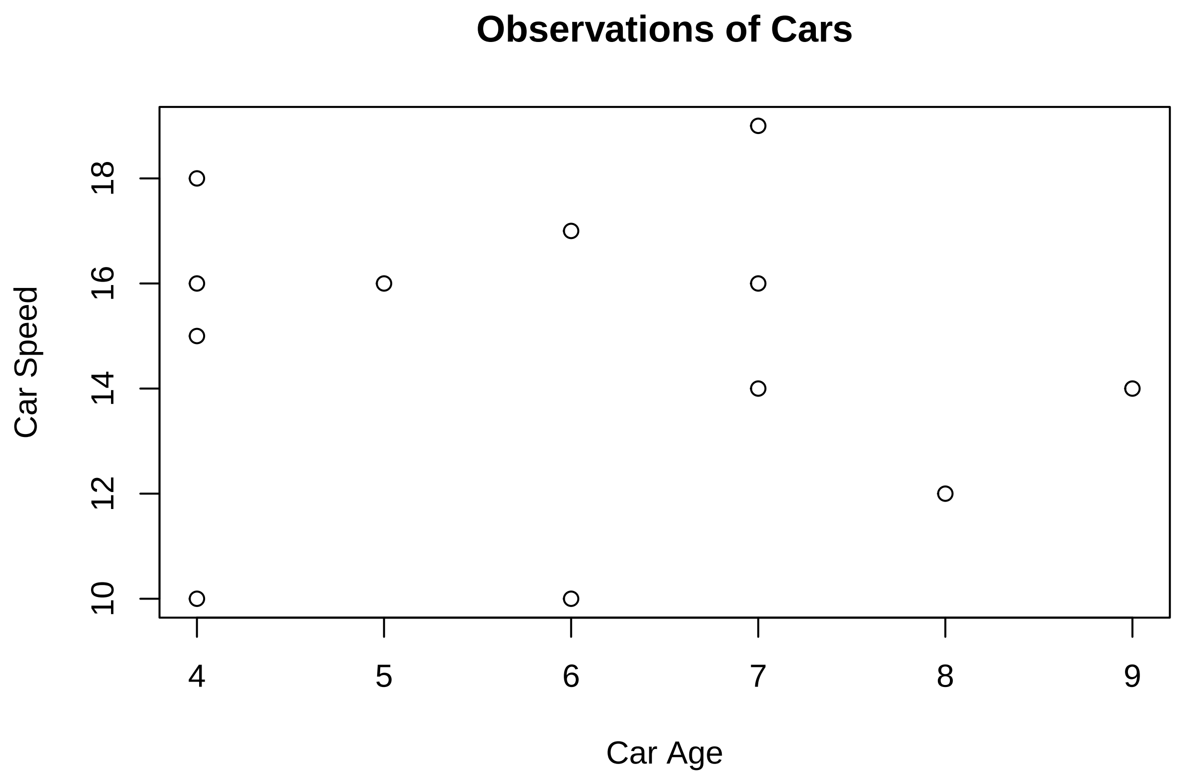


3. R Scatterplot

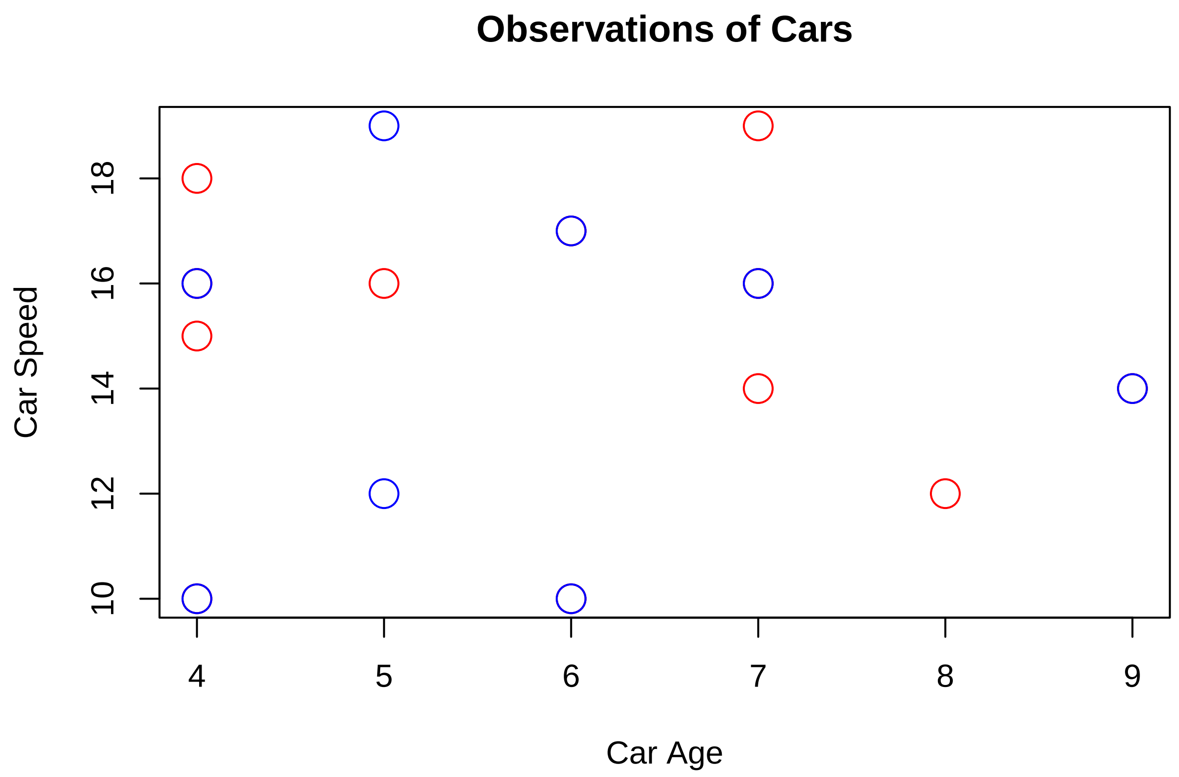
```
x <- c(4,5,7,8,4,4,6,7,4,6,7,9)
y <- c(18,16,14,12,15,16,17,19,10,10,16,14)
plot(x,y)
```



```r
x <- c(4,5,7,8,4,4,6,7,4,6,7,9)
y <- c(18,16,14,12,15,16,17,19,10,10,16,14)
plot(x,y, main = "Observations of Cars", xlab = "Car Age", ylab = "Car Speed")
```

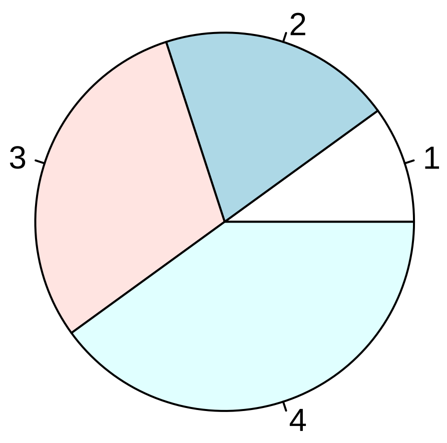


```
x1 <- c(4,5,7,8,4,4,6,7,4,6,7,9)
y1 <- c(18,16,14,12,15,16,17,19,10,10,16,14)

x2 <- c(1,2,3,5,1,4,6,5,4,6,7,9)
y2 <- c(18,16,14,12,15,16,17,19,10,10,16,14)
plot(x1,y1, main = "Observations of Cars", xlab = "Car Age", ylab = "Car Speed", col = "red", cex= 2)
points(x2,y2, col = "blue", cex= 2)
```

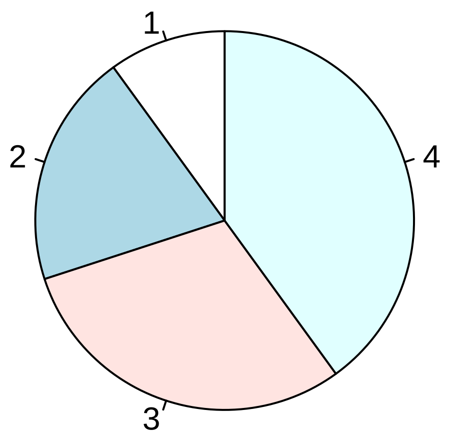


4. R Pie Charts

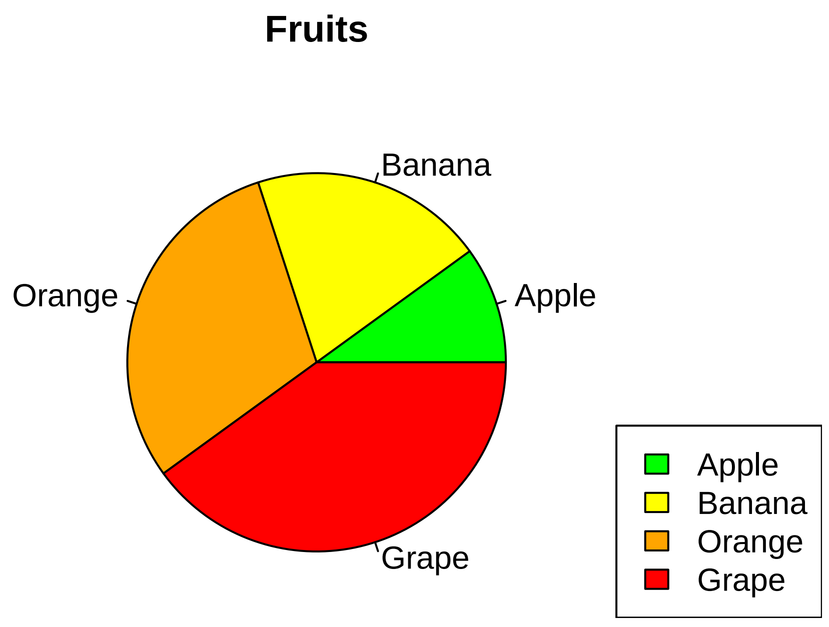
```
x <- c(10,20,30, 40)
pie(x)
```



```
x <- c(10,20,30, 40)
pie(x, init.angle = 90)
```

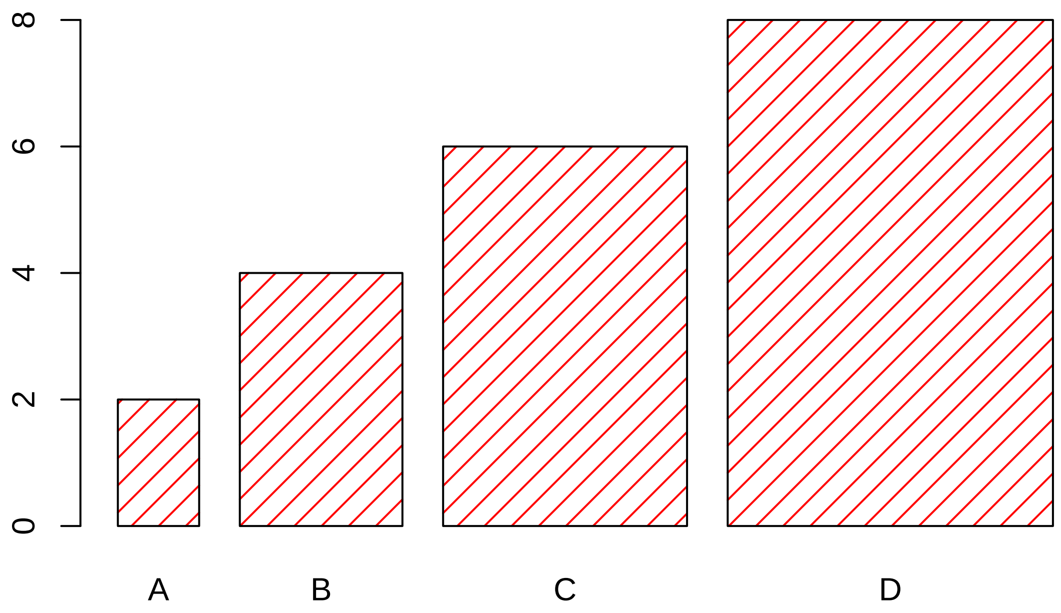


```r
x <- c(10,20,30, 40)
label = c("Apple","Banana", "Orange","Grape")
color = c("green","yellow","orange","red")

pie(x, labels = label, main = "Fruits", col = color)

legend("bottomright", label, fill = color)
```

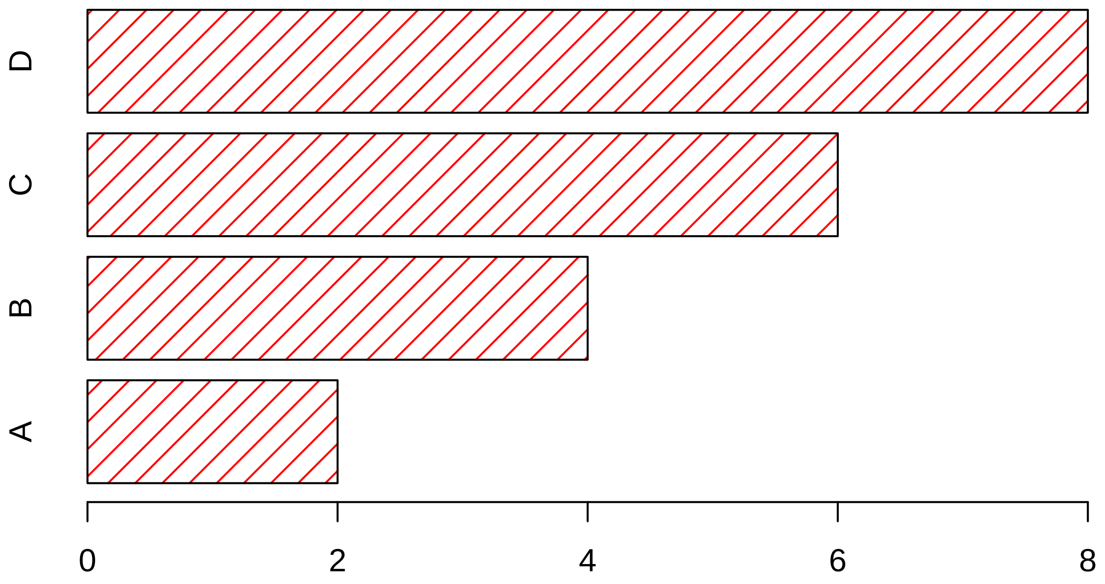


5. R Bars

```
x <- c("A","B","C","D")
y <- c(2,4,6,8)
barplot(y, names.arg = x, col = "red", density = 10, width = rep(y))
```



```
x <- c("A","B","C","D")
y <- c(2,4,6,8)
barplot(y, names.arg = x, col = "red", density = 10, horiz = TRUE)
```


D
C
B
A
0
2
4
6
8