R_Code.R

User02

Tue Mar 20 14:33:24 2018

```
#Assignment Operators#
x <- 2
x

## [1] 2

x = 2
x

## [1] 2

# Assigning character to x #
x <- "IMS"
x

## [1] "IMS"

x = "IMS"
x

## [1] "IMS"

# Arithmatic addition operator #
a <- 5
b <- 6
a + b

## [1] 11

# Arithmatic subtraction operator #
a <- 5
b <- 6
a - b

## [1] -1

# Arithmatic multiplication operator #
a <- 5
b <- 6
a * b

## [1] 30

# Arithmatic division operator #
a <- 5
```

```r
b <- 6
a / b
```

```
## [1] 0.8333333
```

```r
# Arithmatic exponentioation operator #
a <- 5
b <- 6
a^b
```

```
## [1] 15625
```

```r
# Arithmatic modulo operator #
a <- 5
b <- 6
a%%b
```

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

```r
# Relational greater than Operator#
a <- 5
b <- 6
a > b
```

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

```r
# Relational greater than equal to Operator#
a <- 5
b <- 6
a >= b
```

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

```r
# Relational less than Operator#
a <- 5
b <- 6
a < b
```

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

```r
# Relational less than equal to Operator#
a <- 5
b <- 6
a <= b
```

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

```r
# Relational equal to Operator#
a <- 5
b <- 6
a == b
```

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

```r
# Relational not equal to Operator#
a <- 5
b <- 6
a != b
```

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

```r
# Logical Operators #
# Check AND operator with z = 1 #
z <- 1
(z > 2) & (z > 5)
```

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

```r
# Check AND operator with z = 5 #
z <- 5
(z > 4) & (z < 7)
```

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

```r
# Logical Operators #
# Check OR operator with z = 1 #
z <- 1
(z > 2) | (z > 5)
```

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

```r
# Check OR operator with z = 5 #
z <- 5
(z > 4) | (z < 7)
```

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

```r
# Logical Operators #
# Check NOT operator with z = 1 #
z <- 1
!((z > 2) & (z > 5))
```

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

```r
# Logical Operators #
# Check Logical AND operator with z = 1 #
z <- 1
(z > 2) && (z > 5)
```

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

```r
# Check Logical AND operator with z = 5 #
z <- 5
(z > 4) && (z < 7)
```

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

```r
# Check Logical OR operator with z = 1 #
z <- 1
(z > 2) || (z > 5)
```

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

```r
# Check Logical OR operator with z = 5 #
z <- 5
(z > 4) || (z < 7)
```

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

```r
# Different Data type #
# character data types #
a <- "IMS"
#print class of a #
print(class(a))
```

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

```r
# Numeric data types #
a <- 15.4
#print class of a #
print(class(a))
```

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

```r
# Integer data types #
a <- 2L
#print class of a #
print(class(a))
```

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

```r
# Logical data types #
a <- TRUE
# print class of a #
print(class(a))
```

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

```r
# complex data types #
a <- 4+5i
# print class of a #
print(class(a))
```

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

```r
#as.function() and is.function() #
# write a numeric vector #
x <- c(1,5,7,6,2,4,8)
x
```

```
## [1] 1 5 7 6 2 4 8

# check the type of the vector #
is.numeric(x)

## [1] TRUE

# Convert a numeric vetor to factor #
as.factor(x)

## [1] 1 5 7 6 2 4 8
## Levels: 1 2 4 5 6 7 8

# Write a character vector #
y <- c("BA","FM","CFA","CIMA","DS")
# check the type of the vector #
is.character(y)

## [1] TRUE

# Convert a character vector to factor #
as.factor(y)

## [1] BA   FM   CFA  CIMA DS
## Levels: BA CFA CIMA DS FM

# Vector #
# Assigning a numeric vector to workshop #
workshop <- c(1,2,1,2,1,2,1,2)
workshop

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

class(workshop)

## [1] "numeric"

# Assigning a character vector to gender including NA #
gender <- c("f","f","f",NA,"m","m","m","m")
gender

## [1] "f" "f" "f" NA  "m" "m" "m" "m"

class(gender)

## [1] "character"

x = c(1,2,3)
x

## [1] 1 2 3

# Vector replication #
rep(1:3, times = 2)
```

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

```
# Vector Operations #
#Numeric vector of continuous sequence #
x = c(1:5)
x
```

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

```
#Numeric vector of continuous sequence along with additional numbers#
y=c(1:5,10,20)
y
```

```
## [1]  1  2  3  4  5 10 20
```

```
# class of vector of mixed datatype #
a <- c(1,4,2,"a",3+5i,TRUE)
a
```

```
## [1] "1"    "4"    "2"    "a"    "3+5i" "TRUE"
```

```
class(a)
```

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

```
# Combining Vectors #
x = c(1:5)
x
```

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

```
z = c("aa","bb","cc","dd","ee")
z
```

```
## [1] "aa" "bb" "cc" "dd" "ee"
```

```
a <- c(x,z)
a
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "aa" "bb" "cc" "dd" "ee"
```

```
#Vector Arithmatic #
#Create x vector #
x = c(1:5)
# Multiply x by 5 #
5*x
```

```
## [1]  5 10 15 20 25
```

```
# Subtract x from 10 #
10-x
```

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

```r
# Add 15 to x #
15+x
```

```
## [1] 16 17 18 19 20
```

```r
# Arithmetic Operations using vector #
a <- c(1,5,9)
b <- c(3,7,11)
# addition operator #
a + b
```

```
## [1]  4 12 20
```

```r
# Subtraction operator #
a - b
```

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

```r
# Multiplication operator #
a * b
```

```
## [1]  3 35 99
```

```r
# Division operator #
a/b
```

```
## [1] 0.3333333 0.7142857 0.8181818
```

```r
# Exponentiation operator #
a^b
```

```
## [1]           1       78125 31381059609
```

```r
# Modulo operator #
a%%b
```

```
## [1] 1 5 9
```

```r
#  Relational Operations using vector #
a <- c(1.5,3.8,6.7)
b <- c(2.1,8.9,4.1)
# Greater than Operator #
a > b
```

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

```r
# Greater than equal to Operator #
a >= b
```

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

```r
# Less than Operator #
a < b
```

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

```
# Less than equal to Operator #
a<= b
```

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

```
# Equal to Operator #
a==b
```

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

```
# Not equal to Operator #
a!=b
```

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

```
# Logical Operations using vector #
a <- c(-1.5,3.8,6.7, TRUE)
b <- c(2.1,8.9,-4.1, FALSE)
# AND Operator #
a&b
```

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

```
# OR operator #
a|b
```

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

```
# NOT operator #
!a
```

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

```
# Logical AND Operator #
a&&b
```

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

```
# Logical OR Operator #
a||b
```

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

```
#Matrix#
mymatrix <- matrix(c(1,1,5,1,2,1,4,1,2,2,4,3,3,NA,3,4), nrow =4, ncol = 4,
byrow = TRUE)
mymatrix
```

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

```
## [3,]    2    2    4    3
## [4,]    3   NA    3    4
```

```
# Element of 4th row and 2nd column is accessed#
mymatrix[4,2]
```

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

```
# Element of 1st and 3rd row and all columns are accessed #
mymatrix[c(1,3),]
```

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

```
# Element of 4th row and all columns are accessed#
mymatrix[4,]
```

```
## [1]  3 NA  3  4
```

```
# Elements of all rows and 3rd column is accessed#
mymatrix[,3]
```

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

```
is.matrix(mymatrix)
```

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

```
# Element of 3rd row and 2nd column is deleted#
#mymatrix[-3,-2]
# Dimensions of matrix #
dim(mymatrix)
```

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

```
# Adding columns in matrix #
mymatrix1<- cbind(mymatrix, c(1:4))
mymatrix1
```

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

```
# Adding rows in matrix #
mymatrix2<- rbind(mymatrix1, c(1:5))
mymatrix2
```

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

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

```
# Give names to the matrix #
dimnames(mymatrix) <- list(c("row1", "row2", "row3","row4"),
c("col1","col2","col3","col4"))
mymatrix
```

```
##      col1 col2 col3 col4
## row1    1    1    5    1
## row2    2    1    4    1
## row3    2    2    4    3
## row4    3   NA    3    4
```

```
# Access element of rows using row and column names of matrix #
mymatrix["row2","col4"]
```

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

```
# Transpose of matrix #
t(mymatrix)
```

```
##      row1 row2 row3 row4
## col1    1    2    2    3
## col2    1    1    2   NA
## col3    5    4    4    3
## col4    1    1    3    4
```

```
#dataframe#
#Create a vectors of all types #
n=c(2,3,5)
s=c("aa","bb","cc")
b=c(TRUE,FALSE,TRUE)
#create a dataframe usig vectors #
df=data.frame(n,s,b)
df
```

```
##   n  s     b
## 1 2 aa  TRUE
## 2 3 bb FALSE
## 3 5 cc  TRUE
```

```
#List#
# Create a list of numeric types #
my_list <- list(1:4,8,9)
my_list
```

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

```
## [1] 8
##
## [[3]]
## [1] 9

# create a list with different data types #
x = list(1, "a", TRUE, c(1:8))
x

## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1 2 3 4 5 6 7 8

# Create a list on iris data #
list(iris[1:5,1:2],iris[11:17,3:4],iris[30:36,1:4])

## [[1]]
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
##
## [[2]]
##    Petal.Length Petal.Width
## 11          1.5         0.2
## 12          1.6         0.2
## 13          1.4         0.1
## 14          1.1         0.1
## 15          1.2         0.2
## 16          1.5         0.4
## 17          1.3         0.4
##
## [[3]]
##    Sepal.Length Sepal.Width Petal.Length Petal.Width
## 30          4.7         3.2          1.6         0.2
## 31          4.8         3.1          1.6         0.2
## 32          5.4         3.4          1.5         0.4
## 33          5.2         4.1          1.5         0.1
## 34          5.5         4.2          1.4         0.2
## 35          4.9         3.1          1.5         0.2
## 36          5.0         3.2          1.2         0.2
```

```r
#getwd Setwd#
# Used to get a current working directory #
getwd()
```

```
## [1] "C:/Users/User02/Google Drive/Business Analytics/Business Analytics
Video/Moodle Upload/4. Introduction to R/Class Room PPT"
```

```r
# Used to set a working directory #
setwd("C:/Users/User02/Google Drive/Business Analytics/Business Analytics
Video/Moodle Upload/4. Introduction to R/Class Room PPT/")
# Used to read a .csv file #
urban_pop <- read.csv("urbanpop.csv")
# Shows first 5 observations of each column #
head(urban_pop)
```

```
##          country   X1960       X1961       X1962      X1963      X1964
## 1    Afghanistan  769308  814923.049  858521.698  903913.86  951225.94
## 2        Albania  494443  511802.780  529438.851  547376.75  565571.75
## 3        Algeria 3293999 3515147.548 3739963.007 3973289.13 4220987.01
## 4 American Samoa      NA   13660.298   14165.797   14758.93   15396.42
## 5        Andorra      NA    8723.921    9700.346   10748.38   11865.86
## 6         Angola  521205  548265.046  579695.370  612086.70  645261.59
##        X1965      X1966
## 1 1000582.35 1058743.47
## 2  583982.89  602512.17
## 3 4488175.64 4649105.24
## 4   16044.82   16693.11
## 5   13052.75   14216.81
## 6  679109.12  717833.40
```

```r
# Shows last 5 observations of each column #
tail(urban_pop)
```

```
##                   country   X1960      X1961     X1962      X1963
## 204               Vietnam 5107221 5329816.33 5558754.7 5795307.61
## 205 Virgin Islands (U.S.)   18080   19722.55   21488.8   23436.02
## 206                 Yemen  475203  494623.59  524818.5  556070.43
## 207                Zambia  551011  601586.61  653480.6  708251.74
## 208              Zimbabwe  472675  504363.25  537348.4  572056.07
## 209           South Sudan      NA  287795.80  292120.9  296594.19
##          X1964      X1965      X1966
## 204 6040676.77 6295975.61 6574579.81
## 205   25572.38   27842.76   30528.18
## 206  588186.39  621007.03  658653.50
## 207  765969.16  826729.58  903398.07
## 208  608445.60  646509.12  692932.86
## 209  301196.05  305915.63  310762.87
```

```r
# Shows 5 point summary of data #
summary(urban_pop)
```

```
##            country        X1960               X1961        
##  Afghanistan   :  1   Min.   :     3378   Min.   :     1028  
##  Albania       :  1   1st Qu.:    88978   1st Qu.:    70644  
##  Algeria       :  1   Median :   580675   Median :   570159  
##  American Samoa:  1   Mean   :  4988124   Mean   :  4991613  
##  Andorra       :  1   3rd Qu.:  3077228   3rd Qu.:  2807280  
##  Angola        :  1   Max.   :126469700   Max.   :129268133  
##  (Other)       :203   NA's   :11                             
##       X1962               X1963               X1964        
##  Min.   :     1090   Min.   :     1154   Min.   :     1218  
##  1st Qu.:    74974   1st Qu.:    81870   1st Qu.:    84953  
##  Median :   593968   Median :   619331   Median :   645262  
##  Mean   :  5141592   Mean   :  5303711   Mean   :  5468966  
##  3rd Qu.:  2948396   3rd Qu.:  3148941   3rd Qu.:  3296444  
##  Max.   :131974143   Max.   :134599886   Max.   :137205240  
##                                                             
##       X1965               X1966        
##  Min.   :     1281   Min.   :     1349  
##  1st Qu.:    88633   1st Qu.:    93638  
##  Median :   679109   Median :   735139  
##  Mean   :  5637394   Mean   :  5790281  
##  3rd Qu.:  3317422   3rd Qu.:  3418036  
##  Max.   :139663053   Max.   :141962708  
##                                         
```

```
View(urban_pop)
# Shows the structure of the data #
str(urban_pop)
```

```
## 'data.frame':    209 obs. of  8 variables:
##  $ country: Factor w/ 209 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10
...
##  $ X1960  : int  769308 494443 3293999 NA NA 521205 21699 15224096 957974
24996 ...
##  $ X1961  : num  814923 511803 3515148 13660 8724 ...
##  $ X1962  : num  858522 529439 3739963 14166 9700 ...
##  $ X1963  : num  903914 547377 3973289 14759 10748 ...
##  $ X1964  : num  951226 565572 4220987 15396 11866 ...
##  $ X1965  : num  1000582 583983 4488176 16045 13053 ...
##  $ X1966  : num  1058743 602512 4649105 16693 14217 ...
```

```
# Main functions #
#length#
# Gives the length of the vector #
y=c("aa","bb","cc","dd","ee")
length(y)
```

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

```
#Paste #
# used to concatenate vectors after converting to character#
```

```r
fname="Joe"
lname="Smith"
paste(fname, lname)
```

```
## [1] "Joe Smith"
```

```r
#Mode#
#Get or set the type or storage mode of an object#
x <- c(1,5,4,7,2,1,5,3,1)
y <- mode(x)
y
```

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

```r
#WHICH#
#Returns the position of the element #
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
which(letters=="s")
```

```
## [1] 19
```

```r
z <- c(6,5,-3,7)
which(z*z > 9)
```

```
## [1] 1 2 4
```

```r
# Order #
x1 <- c(1,5,4,7,2,1,5,3,1)
# This willgive the position of the number #
order(x1, decreasing = FALSE)
```

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

```r
# This will give the actual number from the vector #
x1[order(x1, decreasing = FALSE)]
```

```
## [1] 1 1 1 2 3 4 5 5 7
```

```r
# merge #
# Make a data frame mapping story numbers to titles
story <- read.table(header=TRUE, text='
                    storyid  title
                    1       lions
                    2      tigers
                    3       bears
                    ')

# Make another data frame with the data and story numbers (no titles)
data <- read.table(header=TRUE, text='
```

```
                  subject storyid rating
                  4       1    6.7
                  4       2    4.5
                  4       1    3.7
                  3       2    3.3
                  1       2    4.1
                  2       1    5.2
                  ')

# Merge the two data frames
merge(story, data, "storyid")

##   storyid  title subject rating
## 1       1  lions       4    6.7
## 2       1  lions       4    3.7
## 3       1  lions       2    5.2
## 4       2 tigers       4    4.5
## 5       2 tigers       3    3.3
## 6       2 tigers       1    4.1

#### Loops ####
#If loop #
# Assign 17 to num_views #
num_views <- 17
# Apply if loop to check the number of views greater than 15 #
if(num_views > 15) {
  print("This show is popular!")
}

## [1] "This show is popular!"

# if else loop #
# Control structure for num_views
# Assign 14 to num_views #
num_views <- 14
# Apply ifelse loop to check the number of views greater than 15 #
# if the condition is TRUE if loop is executed otherwise else loop will be
executed #
if (num_views > 15) {
  print("This show is popular!")
}else{
  print("This show is not popular!")
}

## [1] "This show is not popular!"

# if else if #
# Assign 13 to num_views #
num_views = 13
if (num_views > 15) {
  print("This show is popular!")
```

```
} else if (num_views <= 15 & num_views > 10) {
    print("This show is average!")
} else {
  print("This show is not popular!")
}
```

```
## [1] "This show is average!"
```

```
# for loop#
# Create a vector #
x <- c(2,5,3,9,8,11,6, 4,8,9,1,3,4,6,7,5,21,12,13,14,10)
# initiate count with 0 #
count <- 0
for (i in x)
{
  if(i %% 2 == 0)
    count = count+1
}
print(count)
```

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

```
#While loop#
# Assign the cut off speed to speed variable #
speed <- 84
# apply while loop to check the condition #
while (speed > 30 ) {

  print("Slow down!")
  speed = speed-5
}
```

```
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
## [1] "Slow down!"
```

```
print(speed)
```

```
## [1] 29
```

```
#Apply functions #
# Take a matrix discussed in previous topic #
mymatrix
```

```
##      col1 col2 col3 col4
## row1    1    1    5    1
## row2    2    1    4    1
## row3    2    2    4    3
## row4    3   NA    3    4
```

```
# Use apply function to take sum of column #
colSum <- apply(mymatrix,2,sum)
colSum
```

```
## col1 col2 col3 col4
##    8   NA   16    9
```

```
# lapply function #
# create a list with different data types #
mylist = list(x = 1:15, y = c(TRUE,FALSE,TRUE,TRUE,FALSE))
# lapply functions to take mean #
lapply(mylist, mean)
```

```
## $x
## [1] 8
##
## $y
## [1] 0.6
```

```
# sapply function to take mean #
sapply(mylist, mean)
```

```
##   x   y
## 8.0 0.6
```

```
#Dplyr package#
#install dplyr package #
#install.packages("dplyr")
#load dplyr and datasets package #
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(datasets)
# load inbuild dataset mtcars using data() function #
data(mtcars)
# View loaded dataset using View() function #
```

```r
View(mtcars)
# Apply select function from dplyr package #
select_mtcars <- select(mtcars, cyl, drat)
select_mtcars
```

```
##                     cyl drat
## Mazda RX4             6 3.90
## Mazda RX4 Wag         6 3.90
## Datsun 710            4 3.85
## Hornet 4 Drive        6 3.08
## Hornet Sportabout     8 3.15
## Valiant               6 2.76
## Duster 360            8 3.21
## Merc 240D             4 3.69
## Merc 230              4 3.92
## Merc 280              6 3.92
## Merc 280C             6 3.92
## Merc 450SE            8 3.07
## Merc 450SL            8 3.07
## Merc 450SLC           8 3.07
## Cadillac Fleetwood    8 2.93
## Lincoln Continental   8 3.00
## Chrysler Imperial     8 3.23
## Fiat 128              4 4.08
## Honda Civic           4 4.93
## Toyota Corolla        4 4.22
## Toyota Corona         4 3.70
## Dodge Challenger      8 2.76
## AMC Javelin           8 3.15
## Camaro Z28            8 3.73
## Pontiac Firebird      8 3.08
## Fiat X1-9             4 4.08
## Porsche 914-2         4 4.43
## Lotus Europa          4 3.77
## Ford Pantera L        8 4.22
## Ferrari Dino          6 3.62
## Maserati Bora         8 3.54
## Volvo 142E            4 4.11
```

```r
# Apply filter function from dplyr package #
filter(mtcars, mpg > 20)
```

```
##     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 5  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## 6  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## 7  32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
```

```
## 8  30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## 9  33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## 10 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## 11 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## 12 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## 13 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## 14 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

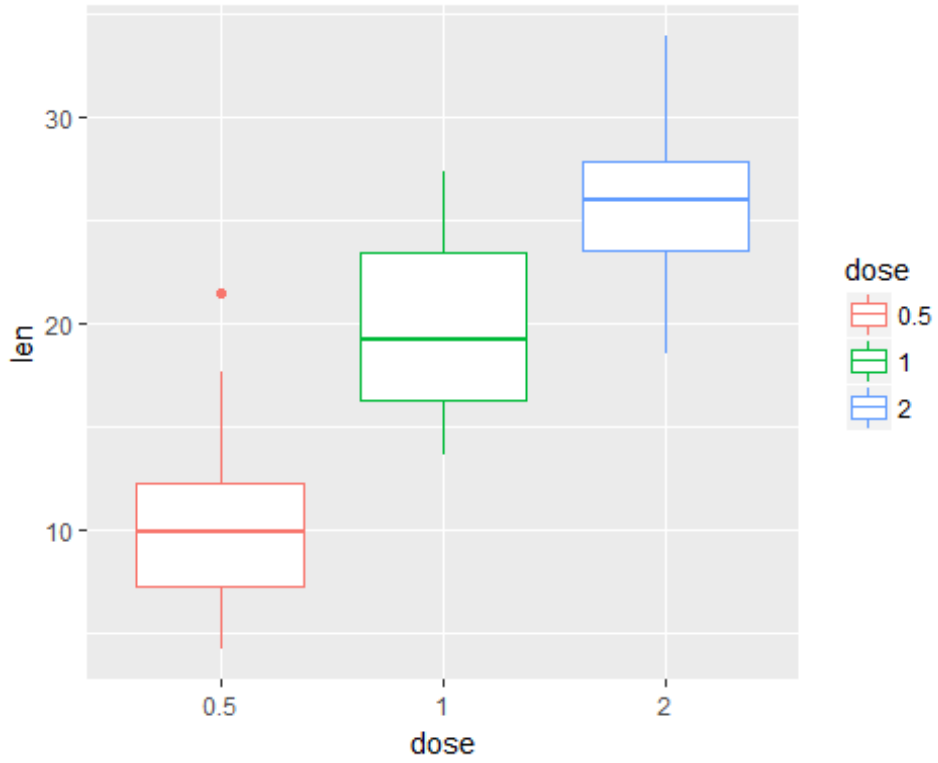
```
# Apply arrange function from dplyr package #
arrange(mtcars, mpg,wt)
```

```
##     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## 2  10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## 3  13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## 4  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## 5  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## 6  15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## 7  15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## 8  15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## 9  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## 10 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## 11 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## 12 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## 13 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## 14 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## 15 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 16 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## 17 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## 18 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## 19 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 20 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 21 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
## 22 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 23 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## 24 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 25 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## 26 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## 27 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## 28 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## 29 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## 30 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## 31 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## 32 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

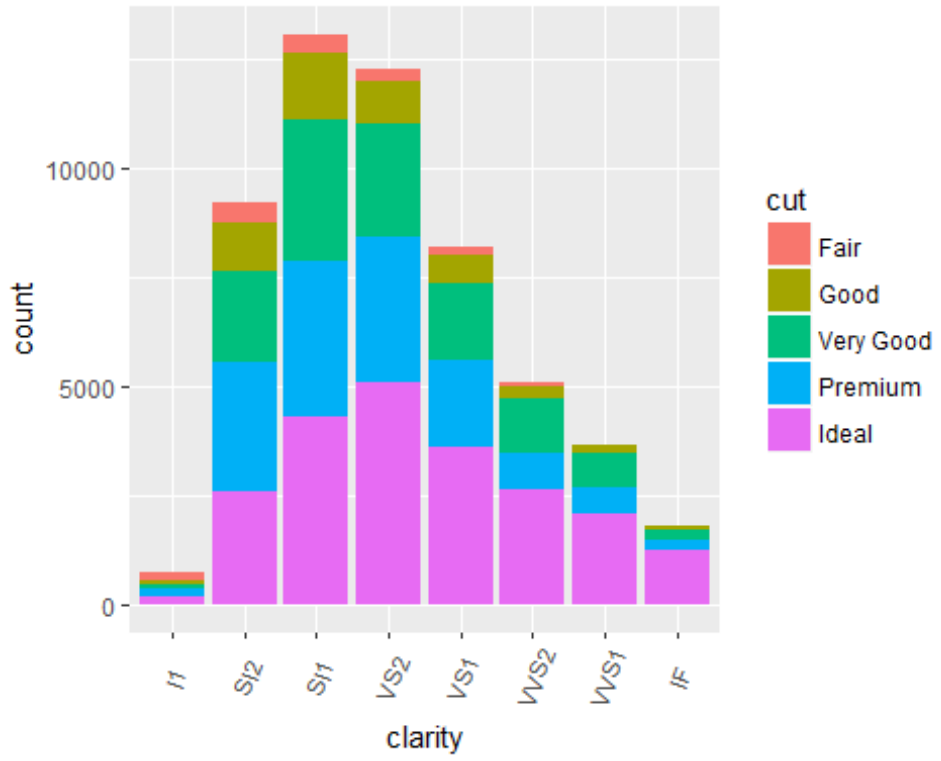
```
# ggplot2 #
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

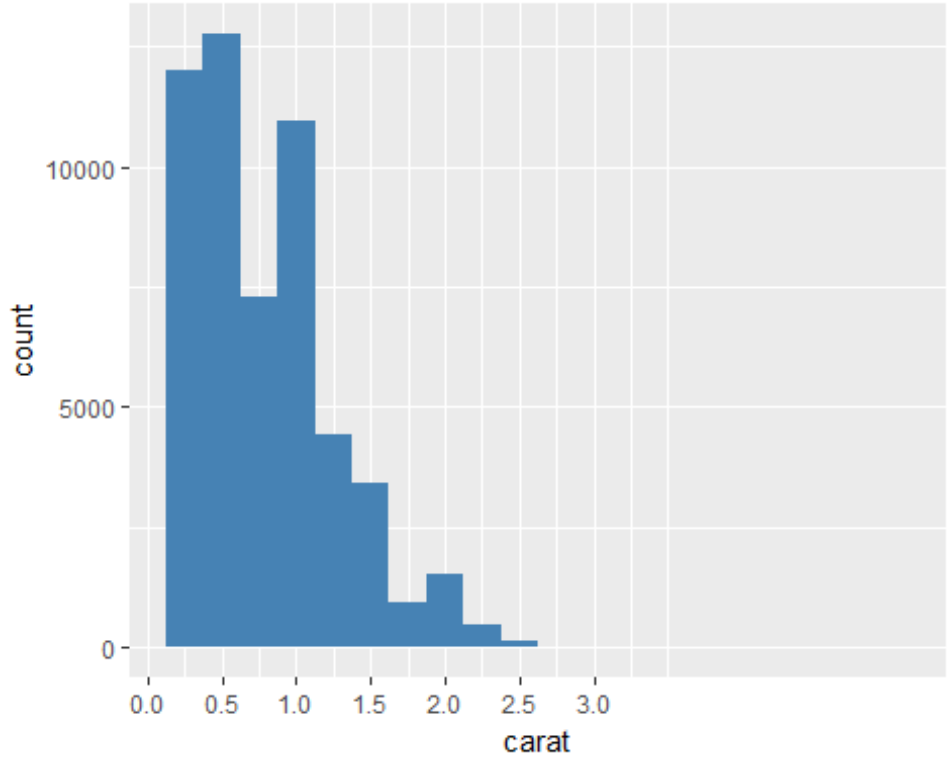
```
## The following object is masked from 'package:dplyr':
##
##     combine

# load a inbuilt dataset #
df <- ToothGrowth
# convert variable to factor #
df$dose <- as.factor(df$dose)
# Show the first five observations of the data #
head(df)

##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5

# plot a boxplot using ggplot function #
model<- ggplot(df, aes(x = dose, y = len, color = dose)) + geom_boxplot()
model
```




```
#Plot a barplot using ggplot function #
bp <- ggplot(diamonds, aes(clarity, fill = cut)) + geom_bar()
+theme(axis.text.x = element_text(angle = 70, vjust = 0.5))
bp
```




```
# Plot a histogram using ggplot function #
ggplot(diamonds, aes(x = carat)) + geom_histogram(binwidth = 0.25, fill =
'steelblue')+scale_x_continuous(breaks=seq(0,3, by=0.5))
```



```r
# readr package #
# install readr package #
#install.packages("readr")
# load readr package #
library(readr)
# load urbanpop dataset using read_csv #
read_csv("urbanpop.csv")
```

```
## Parsed with column specification:
## cols(
##   country = col_character(),
##   `1960` = col_integer(),
##   `1961` = col_double(),
##   `1962` = col_double(),
##   `1963` = col_double(),
##   `1964` = col_double(),
##   `1965` = col_double(),
##   `1966` = col_double()
## )

## # A tibble: 209 x 8
##                country   `1960`       `1961`       `1962`       `1963`
##                  <chr>    <int>        <dbl>        <dbl>        <dbl>
##  1         Afghanistan   769308   814923.049   858521.698   903913.86
##  2             Albania   494443   511802.780   529438.851   547376.75
##  3             Algeria  3293999  3515147.548  3739963.007  3973289.13
##  4      American Samoa       NA    13660.298    14165.797    14758.93
##  5             Andorra       NA     8723.921     9700.346    10748.38
##  6              Angola   521205   548265.046   579695.370   612086.70
##  7 Antigua and Barbuda    21699    21635.051    21664.200    21740.74
##  8           Argentina 15224096 15545222.590 15912120.020 16282345.35
##  9             Armenia   957974  1008597.321  1061426.399  1115612.32
## 10               Aruba    24996    28139.757    28532.729    28763.12
## # ... with 199 more rows, and 3 more variables: `1964` <dbl>,
## #   `1965` <dbl>, `1966` <dbl>
```