**DSA**

**IMT Ghaziabad Term – 2, 2020-22**

**Individual Assignment:**

Submitted by-Harshit Arora (200103063)

Sumitted to-Prof. (Dr.) Sridhar Vaithianathan,

Associate Professor (Analytics).

IMT - Hyderabad.

Date:19/12/20

Harshit Aroa-200103063-DSA Individual Assignment

Harshit Arora

18/12/2020

## Some of the Arithmetic Operations

# Simple Arithematic Calculations  
  
x = 10  
y = 15  
## Addition  
x+y

## [1] 25

## Substraction  
x-y

## [1] -5

## Multiplication  
x\*y

## [1] 150

## Division  
x/y

## [1] 0.6666667

## Square Root  
sqrt(x)

## [1] 3.162278

## Exponential  
x^y

## [1] 1e+15

exp(x)

## [1] 22026.47

log(x, base=exp(1))

## [1] 2.302585

log10(x)

## [1] 1

help("log") # Utilizing R Help

## starting httpd help server ... done

factorial(x)

## [1] 3628800

cos(x)

## [1] -0.8390715

abs(x)

## [1] 10

## Some of the DATA TYPES

#Numeric   
#Character  
x = 10  
class(x)

## [1] "numeric"

i = 5L # L - Integer  
class(i)

## [1] "integer"

is.integer(i)

## [1] TRUE

is.numeric(x)

## [1] TRUE

s = "Harshit Arora"  
class(s)

## [1] "character"

#Logical - TRUE (1) and FALSE (0)  
  
TRUE \* 5

## [1] 5

FALSE \* 5

## [1] 0

K = TRUE  
class(K)

## [1] "logical"

is.logical(K)

## [1] TRUE

#Date - Starting Date (1970) - Numeric Value.  
  
date1 = as.Date("2015-08-20")  
  
date1

## [1] "2015-08-20"

class (date1)

## [1] "Date"

as.numeric(date1)

## [1] 16667

#POSIXct - Date and Time  
date2 = as.POSIXct("2015-08-20 17:42")  
date2

## [1] "2015-08-20 17:42:00 IST"

class(date2)

## [1] "POSIXct" "POSIXt"

as.numeric(date2)

## [1] 1440072720

## Some of the Functions

getwd() # Get Working Directory

## [1] "C:/Users/Dell/Downloads"

# Functions in R  
divider = function(x,y) {  
 result = x/y  
 print(result)  
}  
divider(50,25)

## [1] 2

divider (100,25)

## [1] 4

# Multiplication  
multiply = function(a,b){  
 result = a \* b  
 print (result)  
}  
multiply(23,25)

## [1] 575

multiply (19,20)

## [1] 380

## Some of the Vectors & Arrays

# CONCATENATION AND ARRAYS  
f <- c(1,2,3,4,5)  
f = c(1,2,3,4,5)  
f

## [1] 1 2 3 4 5

f+4

## [1] 5 6 7 8 9

d = f / 4  
d

## [1] 0.25 0.50 0.75 1.00 1.25

f+d

## [1] 1.25 2.50 3.75 5.00 6.25

f = c(1,2,3,4,5)  
  
# Listing and Deleting Objects (Variables)  
ls()

## [1] "d" "date1" "date2" "divider" "f" "i"   
## [7] "K" "multiply" "s" "x" "y"

rm (a)

## Warning in rm(a): object 'a' not found

rm (list = ls())  
  
  
# Vector - R is called as Vectorized language.  
  
v = c(1,2,3,4,5)  
s = v\*2  
s

## [1] 2 4 6 8 10

# A vector is collec tion of elements, all of same type.  
# A vector canot be of mixed type.  
  
# Vector Operation  
d = v-2  
d

## [1] -1 0 1 2 3

f = v /2  
f

## [1] 0.5 1.0 1.5 2.0 2.5

sqrt(f)

## [1] 0.7071068 1.0000000 1.2247449 1.4142136 1.5811388

## Some of the Vector Operations

# Creating Vectors  
# The most common way to create a Vector is using 'c' [combine]  
x = c(1,2,3,4,5,6,7,8,9,10)  
x

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

# Vector Operations  
x\*3 # multiplies each element by 3; No loops necessary!

## [1] 3 6 9 12 15 18 21 24 27 30

x+2

## [1] 3 4 5 6 7 8 9 10 11 12

x-3

## [1] -2 -1 0 1 2 3 4 5 6 7

x/4

## [1] 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50

x^2

## [1] 1 4 9 16 25 36 49 64 81 100

sqrt(x)

## [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427  
## [9] 3.000000 3.162278

# colon (:) operation - Sequencing  
# Creates sequence of Numbers in either direction!  
1:10 #(: - Through)

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

10:1

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

-2:3

## [1] -2 -1 0 1 2 3

5:-7

## [1] 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7

# More on Vector Operations ... Two vectors  
# create two vectors of equal length  
x = 1:10  
y = -5:4  
x + y # Add

## [1] -4 -2 0 2 4 6 8 10 12 14

x-y

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

x\*y

## [1] -5 -8 -9 -8 -5 0 7 16 27 40

x/y

## [1] -0.2 -0.5 -1.0 -2.0 -5.0 Inf 7.0 4.0 3.0 2.5

x^y

## [1] 1.000000e+00 6.250000e-02 3.703704e-02 6.250000e-02 2.000000e-01  
## [6] 1.000000e+00 7.000000e+00 6.400000e+01 7.290000e+02 1.000000e+04

# check the length of each vector  
length(x)

## [1] 10

length(y)

## [1] 10

# Unequal length vectors  
x+c(1,2) # Shorter vector gets recycled!

## [1] 2 4 4 6 6 8 8 10 10 12

x+c (1,2,3)# If Longer vector is not "multiple" of shorter vector, a warning is given!

## Warning in x + c(1, 2, 3): longer object length is not a multiple of shorter  
## object length

## [1] 2 4 6 5 7 9 8 10 12 11

# Comparison also work on vector!  
x <= 5

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

x<y

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

# Vector Comparison - "any" and "all"  
x = 10:1  
y = -4:5  
any(x<y)

## [1] TRUE

all(x<y)

## [1] FALSE

# The "nchar" function also acts on each element of vector.  
q = c("Hockey","Football","Baseball","Curlin","Rugby","Lacrosse",  
 "Basketball","Tennis","Cricket","Soccer")  
q

## [1] "Hockey" "Football" "Baseball" "Curlin" "Rugby"   
## [6] "Lacrosse" "Basketball" "Tennis" "Cricket" "Soccer"

nchar(q)

## [1] 6 8 8 6 5 8 10 6 7 6

nchar(y)

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

?nchar()  
  
# Subscripting:Accessing "individual elements" in vector is done using square brackets []  
x[1]

## [1] 10

x[1:2]

## [1] 10 9

x[c(1:5,9)]

## [1] 10 9 8 7 6 2

# Give Names to Vector!  
c(One = "a", Two = "y", Last = "r") # Name-Value pair

## One Two Last   
## "a" "y" "r"

# You can Name the vector after creating vector as well!  
w = 1:3  
names(w) = c("a","b","c")  
w

## a b c   
## 1 2 3

## Some of the Data Frames

# Creating a Dataframe from vectors  
  
x = 10:1  
y = -4:5  
q = c("Hockey","Football","Baseball","Curlin","Rugby","Lacrosse",  
 "Basketball","Tennis","Cricket","Soccer")  
theDF = data.frame(x,y,q) # this would create a 10x3 data.frame with x, y and q as variable names  
theDF

## x y q  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

# Assigning Names  
theDF = data.frame (First=x, Second =y, Sport = q)  
theDF

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

# Checking the dimensions of the DF.  
nrow(theDF)

## [1] 10

ncol(theDF)

## [1] 3

dim(theDF)

## [1] 10 3

names (theDF)

## [1] "First" "Second" "Sport"

names(theDF)[3]

## [1] "Sport"

rownames(theDF)

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

# Head and Tail  
head(theDF)

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse

head(theDF, n=7)

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball

tail(theDF)

## First Second Sport  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

class(theDF)

## [1] "data.frame"

# Accessing Individual Column using $  
theDF$Sport # gives the third column named Sport

## [1] "Hockey" "Football" "Baseball" "Curlin" "Rugby"   
## [6] "Lacrosse" "Basketball" "Tennis" "Cricket" "Soccer"

# Accessing Specific row and column  
theDF[3,2] # 3rd row and 2nd Column

## [1] -2

theDF[3,2:3] # 3rd Row and column 2 thru 3

## Second Sport  
## 3 -2 Baseball

theDF[c(3,5), 2]# Row 3&5 from Column 2;

## [1] -2 0

# since only one column was selected, it was returned as vector and hence no column names in output.  
  
# Rows 3&5 and Columns 2 through 3  
theDF[c(3,5), 2:3]

## Second Sport  
## 3 -2 Baseball  
## 5 0 Rugby

theDF[ ,3] # Access all Rows for column 3

## [1] "Hockey" "Football" "Baseball" "Curlin" "Rugby"   
## [6] "Lacrosse" "Basketball" "Tennis" "Cricket" "Soccer"

theDF[ , 2:3]

## Second Sport  
## 1 -4 Hockey  
## 2 -3 Football  
## 3 -2 Baseball  
## 4 -1 Curlin  
## 5 0 Rugby  
## 6 1 Lacrosse  
## 7 2 Basketball  
## 8 3 Tennis  
## 9 4 Cricket  
## 10 5 Soccer

theDF[2,]# Access all columns for Row 2

## First Second Sport  
## 2 9 -3 Football

theDF[2:4,]

## First Second Sport  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin

theDF[ , c("First", "Sport")]# access using Column Names

## First Sport  
## 1 10 Hockey  
## 2 9 Football  
## 3 8 Baseball  
## 4 7 Curlin  
## 5 6 Rugby  
## 6 5 Lacrosse  
## 7 4 Basketball  
## 8 3 Tennis  
## 9 2 Cricket  
## 10 1 Soccer

## Some of the Factors

# Factor Vectors - Ordinal data [Ordered Categorical]  
# Factors are important concept in R, esp. when building models  
  
q2 = c(q,"Hockey","Lacrosse","Hockey","Water Polo","Hockey","Lacrosse")  
q2

## [1] "Hockey" "Football" "Baseball" "Curlin" "Rugby"   
## [6] "Lacrosse" "Basketball" "Tennis" "Cricket" "Soccer"   
## [11] "Hockey" "Lacrosse" "Hockey" "Water Polo" "Hockey"   
## [16] "Lacrosse"

class(q2)

## [1] "character"

as.numeric(q2)

## Warning: NAs introduced by coercion

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

class(q2)

## [1] "character"

# Converting "q2" to factor!  
q2\_F = as.factor(q2)  
q2\_F # notice the "Levels" info in the output!

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer Hockey Lacrosse   
## [13] Hockey Water Polo Hockey Lacrosse   
## 11 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Water Polo

# 11 Levels - 10 Distinct Names from "q" and one (Water polo) from "q2"   
# The "levels" of a factor are the unique values of that factor variable.  
# Technically R is giving "unique integer" to each distinct names, See below  
as.numeric(q2\_F)# IN the O/P --> Notice "6" = "Hockey"

## [1] 6 5 1 4 8 7 2 10 3 9 6 7 6 11 6 7

# Ordered Levels and Un-ordered Levels  
# Factors can drastically reduce the size of the variable...  
# ... because they are storing only unique values!  
factor(x=c("High School","College","Masters","Doctrate"),  
 levels = c("High School","College","Masters","Doctrate"),  
 ordered = TRUE)

## [1] High School College Masters Doctrate   
## Levels: High School < College < Masters < Doctrate

## Missing, NA & Null

# NA - Missing data - Missing Value  
z = c(1,2,NA,8,3,NA,3)  
z = c(1,2,NA,8,3,NA,3)  
z

## [1] 1 2 NA 8 3 NA 3

# "is.na" tests each element of a vector for missingness  
is.na(z)

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

#Another example  
z\_char = c("Hockey", NA ,"Cricket")  
z\_char

## [1] "Hockey" NA "Cricket"

is.na(z\_char)

## [1] FALSE TRUE FALSE

z= c(1,NULL,3)  
z

## [1] 1 3

x = c(1,NA,3)  
x

## [1] 1 NA 3

# Notice, here the "NULL" didnot get stored in "z", infact "z" has only length of 2!  
length(z)

## [1] 2

length(x)

## [1] 3

# Assigning NULL and checking!  
d = NULL  
is.null(d)

## [1] TRUE

## Example of Operations on Data Frames

# Creating a Dataframe from vectors  
  
x = 10:1  
y = -4:5  
q = c("Hockey","Football","Baseball","Curlin","Rugby","Lacrosse",  
 "Basketball","Tennis","Cricket","Soccer")  
theDF = data.frame(x,y,q) # this would create a 10x3 data.frame with x, y and q as variable names  
theDF

## x y q  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

str(theDF)# Very important - Str - Structure

## 'data.frame': 10 obs. of 3 variables:  
## $ x: int 10 9 8 7 6 5 4 3 2 1  
## $ y: int -4 -3 -2 -1 0 1 2 3 4 5  
## $ q: chr "Hockey" "Football" "Baseball" "Curlin" ...

q = as.factor(q)  
  
# Assigning Names  
theDF = data.frame (First=x, Second =y, Sport = q)  
theDF

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

# Checking the dimensions of the DF.  
nrow(theDF)

## [1] 10

ncol(theDF)

## [1] 3

dim(theDF)

## [1] 10 3

names (theDF)

## [1] "First" "Second" "Sport"

names(theDF)[3]

## [1] "Sport"

rownames(theDF)

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

# Head and Tail  
head(theDF)# First 6 rows with all columns

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse

head(theDF, n=10)

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

tail(theDF)# last six rows with all columns

## First Second Sport  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

class(theDF)

## [1] "data.frame"

# Accessing Individual Column using $  
theDF$Sport # gives the third column named Sport

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer   
## 10 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Tennis

# Accessing Specific row and column  
theDF[3,2] # 3rd row and 2nd Column

## [1] -2

theDF[3,2:3] # 3rd Row and column 2 thru 3

## Second Sport  
## 3 -2 Baseball

theDF[c(3,5), 2]# Row 3&5 from Column 2;

## [1] -2 0

# since only one column was selected, it was returned as vector and hence no column names in output.  
  
# Rows 3&5 and Columns 2 through 3  
theDF[c(3,5), 2:3]

## Second Sport  
## 3 -2 Baseball  
## 5 0 Rugby

theDF[ ,3] # Access all Rows for column 3

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer   
## 10 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Tennis

theDF[ , 2:3]

## Second Sport  
## 1 -4 Hockey  
## 2 -3 Football  
## 3 -2 Baseball  
## 4 -1 Curlin  
## 5 0 Rugby  
## 6 1 Lacrosse  
## 7 2 Basketball  
## 8 3 Tennis  
## 9 4 Cricket  
## 10 5 Soccer

theDF[2,]# Access all columns for Row 2

## First Second Sport  
## 2 9 -3 Football

theDF[2:4,]

## First Second Sport  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin

theDF[ , c("First", "Sport")]# access using Column Names

## First Sport  
## 1 10 Hockey  
## 2 9 Football  
## 3 8 Baseball  
## 4 7 Curlin  
## 5 6 Rugby  
## 6 5 Lacrosse  
## 7 4 Basketball  
## 8 3 Tennis  
## 9 2 Cricket  
## 10 1 Soccer

theDF[ ,"Sport"]# Access specific Column

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer   
## 10 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Tennis

class(theDF[ ,"Sport"])

## [1] "factor"

theDF["Sport"]# This returns the one column data.frame

## Sport  
## 1 Hockey  
## 2 Football  
## 3 Baseball  
## 4 Curlin  
## 5 Rugby  
## 6 Lacrosse  
## 7 Basketball  
## 8 Tennis  
## 9 Cricket  
## 10 Soccer

class(theDF["Sport"]) # Data.Frame

## [1] "data.frame"

theDF[["Sport"]]#To access Specific column using Double Square Brackets

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer   
## 10 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Tennis

class(theDF[["Sport"]]) # Factor

## [1] "factor"

theDF[ ,"Sport", drop = FALSE]# Use "Drop=FALSE" to get data.fame with single sqaure bracket.

## Sport  
## 1 Hockey  
## 2 Football  
## 3 Baseball  
## 4 Curlin  
## 5 Rugby  
## 6 Lacrosse  
## 7 Basketball  
## 8 Tennis  
## 9 Cricket  
## 10 Soccer

class(theDF[ ,"Sport", drop = FALSE]) # data.frame

## [1] "data.frame"

theDF[ ,3, drop = FALSE]

## Sport  
## 1 Hockey  
## 2 Football  
## 3 Baseball  
## 4 Curlin  
## 5 Rugby  
## 6 Lacrosse  
## 7 Basketball  
## 8 Tennis  
## 9 Cricket  
## 10 Soccer

class(theDF[ ,3, drop = FALSE]) # data.frame

## [1] "data.frame"

# To see how factor is stored in data.frame  
newFactor = factor(c("Pennsylvania","New York","New Jersey","New York","Tennessee","Massachusetts","Pennsylvania","New York"))  
newFactor

## [1] Pennsylvania New York New Jersey New York Tennessee   
## [6] Massachusetts Pennsylvania New York   
## Levels: Massachusetts New Jersey New York Pennsylvania Tennessee

# model.matrix(~newFactor -1)  
# ? model.matrix()

## Operations on matrix

# A matrix (plural matrices) is a rectangular array or table of numbers, symbols, or expressions...  
#..., arranged in rows and columns.(i.e.) 2-Dimensional Array  
  
# Similar to data.frame(RxC) and also similar to Vector  
# Matrix - Element by element operations are possible  
  
A = matrix(1:10, nrow=5)# Create a 5x2 matrix  
B = matrix(21:30, nrow=5)#Create another 5x2 matrix   
C = matrix (21:40, nrow=2)#Create another 2x10 matrix  
  
A

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

B

## [,1] [,2]  
## [1,] 21 26  
## [2,] 22 27  
## [3,] 23 28  
## [4,] 24 29  
## [5,] 25 30

C

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 21 23 25 27 29 31 33 35 37 39  
## [2,] 22 24 26 28 30 32 34 36 38 40

nrow(A)

## [1] 5

ncol(A)

## [1] 2

dim(A)

## [1] 5 2

# Add Them  
A+B

## [,1] [,2]  
## [1,] 22 32  
## [2,] 24 34  
## [3,] 26 36  
## [4,] 28 38  
## [5,] 30 40

# Multiply Them (Vector Multiplication!)  
A

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

B

## [,1] [,2]  
## [1,] 21 26  
## [2,] 22 27  
## [3,] 23 28  
## [4,] 24 29  
## [5,] 25 30

A\*B # A = 5x2 and B = 5x2

## [,1] [,2]  
## [1,] 21 156  
## [2,] 44 189  
## [3,] 69 224  
## [4,] 96 261  
## [5,] 125 300

#See if the elements are equal  
A == B

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

# Matrix Multiplication(MM. A is 5x2. B is 5x2. B-transpose is 2x5  
A %\*% t(B)

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 177 184 191 198 205  
## [2,] 224 233 242 251 260  
## [3,] 271 282 293 304 315  
## [4,] 318 331 344 357 370  
## [5,] 365 380 395 410 425

# Naming the Columns and Rows   
colnames(A)

## NULL

rownames(A)

## NULL

colnames(A)= c("Left","Right")  
rownames(A)= c("1st","2nd","3rd","4th","5th")  
colnames(B)

## NULL

rownames(B)

## NULL

colnames(B)= c("First","Second")  
rownames(B)= c("One","Two","Three","Four","Five")  
colnames(C)

## NULL

rownames(C)

## NULL

colnames(C) = LETTERS [1:10]  
rownames(C) = c("Top", "Bottom")  
  
# Matrix Multiplication. A is 5x2 and C is 2x10  
dim(A)

## [1] 5 2

dim(C)

## [1] 2 10

t(A)

## 1st 2nd 3rd 4th 5th  
## Left 1 2 3 4 5  
## Right 6 7 8 9 10

A %\*% C

## A B C D E F G H I J  
## 1st 153 167 181 195 209 223 237 251 265 279  
## 2nd 196 214 232 250 268 286 304 322 340 358  
## 3rd 239 261 283 305 327 349 371 393 415 437  
## 4th 282 308 334 360 386 412 438 464 490 516  
## 5th 325 355 385 415 445 475 505 535 565 595

## Multi Dimentional Arrays

theArray = array(1:12, dim=c(2,3,2))# Total Elements = R x C x OD  
theArray

## , , 1  
##   
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6  
##   
## , , 2  
##   
## [,1] [,2] [,3]  
## [1,] 7 9 11  
## [2,] 8 10 12

theArray [1, ,]# Accessing all elements from Row 1, all columns, all outer dimensions & build C x OD (R x C)

## [,1] [,2]  
## [1,] 1 7  
## [2,] 3 9  
## [3,] 5 11

theArray[1, ,1]# Accessing all elements from Row 1, all columns, first outer dimension

## [1] 1 3 5

theArray[, ,1]# Accessing all rows, all columns, first outer dimension

## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6

# Array with Four Outer Dimensions (OD)  
theArray\_4D = array(1:32, dim=c(2,4,4))  
theArray\_4D

## , , 1  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 1 3 5 7  
## [2,] 2 4 6 8  
##   
## , , 2  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 9 11 13 15  
## [2,] 10 12 14 16  
##   
## , , 3  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 17 19 21 23  
## [2,] 18 20 22 24  
##   
## , , 4  
##   
## [,1] [,2] [,3] [,4]  
## [1,] 25 27 29 31  
## [2,] 26 28 30 32

theArray\_4D [1, ,]

## [,1] [,2] [,3] [,4]  
## [1,] 1 9 17 25  
## [2,] 3 11 19 27  
## [3,] 5 13 21 29  
## [4,] 7 15 23 31

theArray\_4D[1, ,1]

## [1] 1 3 5 7

theArray[, ,1]

## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6

## Lists

# Lists - Stores any number of items of any type.  
# List can contain all numerics or characters or...  
#...a mix of the two or data.frames or recursively other lists.  
  
# Lists are created with the "list" function.  
# Each argument in "list" becomes an element of the list.  
  
list(1,2,3)# creates a three element list

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

list(c(1,2,3))# creates a single element(vector with three elements)

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

list3 = list(c(1,2,3), 3:7)# create two element list  
# first is three elements vector, next is five element vector.  
list3

## [[1]]  
## [1] 1 2 3  
##   
## [[2]]  
## [1] 3 4 5 6 7

# The same can be written as  
(list3 = list(c(1,2,3), 3:7))

## [[1]]  
## [1] 1 2 3  
##   
## [[2]]  
## [1] 3 4 5 6 7

# Two Element list  
# First element is data.frame and next is 10 element vector  
list(theDF, 1:10)# theDF is already created in previous exercise!

## [[1]]  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer  
##   
## [[2]]  
## [1] 1 2 3 4 5 6 7 8 9 10

# Three element list  
list5 = list(theDF, 1:10, list3)  
list5

## [[1]]  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer  
##   
## [[2]]  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## [[3]]  
## [[3]][[1]]  
## [1] 1 2 3  
##   
## [[3]][[2]]  
## [1] 3 4 5 6 7

#Naming List (similar to column name in data.frame)   
names(list5)= c("data.frame", "vector","list")  
names(list5)

## [1] "data.frame" "vector" "list"

list5

## $data.frame  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer  
##   
## $vector  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $list  
## $list[[1]]  
## [1] 1 2 3  
##   
## $list[[2]]  
## [1] 3 4 5 6 7

#Naming using "Name-Value" pair  
list6 = list(TheDataFrame = theDF, TheVector = 1:10, TheList = list3)  
names(list6)

## [1] "TheDataFrame" "TheVector" "TheList"

list6

## $TheDataFrame  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer  
##   
## $TheVector  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $TheList  
## $TheList[[1]]  
## [1] 1 2 3  
##   
## $TheList[[2]]  
## [1] 3 4 5 6 7

# Creating an empty list  
(emptylist = vector(mode="list", length =4))

## [[1]]  
## NULL  
##   
## [[2]]  
## NULL  
##   
## [[3]]  
## NULL  
##   
## [[4]]  
## NULL

# Accessing individual element of a list - Double Square Brackets  
# specify either element number or name  
list5[[1]]

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

list5[["data.frame"]]

## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer

list5[[1]]$Sport

## [1] Hockey Football Baseball Curlin Rugby Lacrosse   
## [7] Basketball Tennis Cricket Soccer   
## 10 Levels: Baseball Basketball Cricket Curlin Football Hockey ... Tennis

list5[[1]][,"Second"]

## [1] -4 -3 -2 -1 0 1 2 3 4 5

list5[[1]][,"Second", drop = FALSE]

## Second  
## 1 -4  
## 2 -3  
## 3 -2  
## 4 -1  
## 5 0  
## 6 1  
## 7 2  
## 8 3  
## 9 4  
## 10 5

# LENGTH OF LIST  
length(list5)

## [1] 3

names(list5)

## [1] "data.frame" "vector" "list"

list5

## $data.frame  
## First Second Sport  
## 1 10 -4 Hockey  
## 2 9 -3 Football  
## 3 8 -2 Baseball  
## 4 7 -1 Curlin  
## 5 6 0 Rugby  
## 6 5 1 Lacrosse  
## 7 4 2 Basketball  
## 8 3 3 Tennis  
## 9 2 4 Cricket  
## 10 1 5 Soccer  
##   
## $vector  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## $list  
## $list[[1]]  
## [1] 1 2 3  
##   
## $list[[2]]  
## [1] 3 4 5 6 7

## File Operations

# Its time that we load data in R.  
# Most common way to get data is reading comma separated values(CSV)  
  
# Reading CSVs  
theUrl = "http://www.jaredlander.com/data/Tomato%20First.csv"  
# visit https://www.jaredlander.com/data/ for other Datasets  
tomato = read.table(file=theUrl, header=TRUE, sep =",")  
head(tomato)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

#It might be tempting to use read.csv but that is more trouble than it is worth,  
#...and all it does is call read.table with some arguments preset.  
  
# Sometimes CSVs(or tab delimited files) are poorly built,   
# where the cell separator has been used inside a cell.  
# In this case read.csv2(or read.delim2)should be used instead of read.table.  
  
# Reading Excel Data - Not worth the Effort.  
# Unfortunately, it is difficult to read Excel data into R - Requires additional packages to be installed.  
# Convert into CSV and read.  
  
# Reading Text Files  
  
#R Binary Files  
# save the tomato data.frame to Disk  
save(tomato, file="tomato.rdata")  
# remove tomato from memory  
rm(tomato)  
  
# read it from the rdata file  
load("tomato.rdata")  
head(tomato)

## Round Tomato Price Source Sweet Acid Color Texture Overall  
## 1 1 Simpson SM 3.99 Whole Foods 2.8 2.8 3.7 3.4 3.4  
## 2 1 Tuttorosso (blue) 2.99 Pioneer 3.3 2.8 3.4 3.0 2.9  
## 3 1 Tuttorosso (green) 0.99 Pioneer 2.8 2.6 3.3 2.8 2.9  
## 4 1 La Fede SM DOP 3.99 Shop Rite 2.6 2.8 3.0 2.3 2.8  
## 5 2 Cento SM DOP 5.49 D Agostino 3.3 3.1 2.9 2.8 3.1  
## 6 2 Cento Organic 4.99 D Agostino 3.2 2.9 2.9 3.1 2.9  
## Avg.of.Totals Total.of.Avg  
## 1 16.1 16.1  
## 2 15.3 15.3  
## 3 14.3 14.3  
## 4 13.4 13.4  
## 5 14.4 15.2  
## 6 15.5 15.1

# Built-in datasets in R  
data()# List of built-in Datasets in R  
  
# Loading  
data(mtcars)  
# Print the first 6 rows  
head(mtcars, 6)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

## Basic Satistic Functions

# Basic Statistics - Mean, Variances,Correlations and T-tests  
  
# Generate a random sample of 100 numbers between 1 and 100  
x = sample(x=1:100, size = 100, replace = TRUE)  
x # the output of "x" is a vector of data

## [1] 4 15 65 69 50 94 44 73 72 92 17 68 61 67 72 14 75 86 60 42 68 31 94 45 54  
## [26] 23 61 63 99 99 52 78 81 83 43 87 54 26 4 78 79 86 88 68 57 46 41 16 99 49  
## [51] 4 89 49 69 1 98 16 2 73 61 78 49 1 42 14 8 76 69 52 72 88 76 84 65 35  
## [76] 98 3 58 64 62 10 13 28 60 16 54 99 56 68 91 50 36 48 10 29 17 49 37 78 56

# Simple Arithmetic Mean  
mean(x)

## [1] 54.53

# Calculate Mean when Missing Data is found  
y = x # copy x to y  
y[sample(x=1:100, size = 20, replace = FALSE)] = NA  
y

## [1] 4 15 65 NA 50 94 44 NA 72 92 17 68 61 67 72 14 75 86 60 NA NA 31 94 45 NA  
## [26] 23 61 NA 99 99 52 78 81 83 43 NA NA 26 4 78 79 86 88 68 57 46 41 16 99 49  
## [51] 4 89 49 69 NA NA NA 2 73 NA 78 49 NA 42 14 8 76 69 NA 72 88 76 84 65 35  
## [76] 98 3 58 64 62 10 NA NA 60 16 54 99 56 68 NA NA 36 NA NA 29 17 49 37 78 56

mean(y)# Will give NA!

## [1] NA

# Remove missing value(s)and calculate mean  
mean(y, na.rm=TRUE) # Now, it will give the mean value

## [1] 55.925

# Weighted Mean  
Grades = c(95,72,87,66)  
Weights = c(1/2, 1/4, 1/8, 1/8)  
mean(Grades)# Simple Arithmetic mean

## [1] 80

weighted.mean(x = Grades, w = Weights)# Weighted Mean

## [1] 84.625

#Variance  
var(x)

## [1] 797.2819

#Calculating Variance using formula!  
sum((x-mean(x))^2)/ (length(x)-1)

## [1] 797.2819

# Standard Deviation  
sqrt(var(x))

## [1] 28.23618

sd(x)

## [1] 28.23618

sd(y)

## [1] NA

sd(y, na.rm=TRUE)

## [1] 27.96959

# Other Commonly Used Functions  
min(x)

## [1] 1

max(x)

## [1] 99

median(x)

## [1] 59

min(y)

## [1] NA

min(y, na.rm=TRUE)

## [1] 2

# Summary Statistics  
summary(x)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 35.75 59.00 54.53 76.00 99.00

summary(y)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 2.00 36.75 60.50 55.92 78.00 99.00 20

# Quantiles  
quantile(x, probs = c(0.25, 0.75)) # Calculate 25th and 75th Quantile

## 25% 75%   
## 35.75 76.00

#quantile(x, probs = c(0.1,0.25,0.5, 0.75,0.99))  
  
#quantile(y, probs = c(0.25, 0.75)) # Calculate 25th and 75th Quantile  
quantile(y, probs = c(0.25, 0.75), na.rm = TRUE)

## 25% 75%   
## 36.75 78.00

# Correlation and Covariance  
#install.packages("ggplot2")  
library(ggplot2)# require(ggplot2)  
head(economics)# Built-in dataset in ggplot2 package

## # A tibble: 6 x 6  
## date pce pop psavert uempmed unemploy  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1967-07-01 507. 198712 12.6 4.5 2944  
## 2 1967-08-01 510. 198911 12.6 4.7 2945  
## 3 1967-09-01 516. 199113 11.9 4.6 2958  
## 4 1967-10-01 512. 199311 12.9 4.9 3143  
## 5 1967-11-01 517. 199498 12.8 4.7 3066  
## 6 1967-12-01 525. 199657 11.8 4.8 3018

cor(economics$pce, economics$psavert) #pce-Personal Consumption Expenditure;psavert -Personal Savings Rate

## [1] -0.7928546

# To compare correlation for Multiple variables  
cor(economics[, c(2,4:6)])

## pce psavert uempmed unemploy  
## pce 1.0000000 -0.7928546 0.7269616 0.6145176  
## psavert -0.7928546 1.0000000 -0.3251377 -0.3093769  
## uempmed 0.7269616 -0.3251377 1.0000000 0.8693097  
## unemploy 0.6145176 -0.3093769 0.8693097 1.0000000

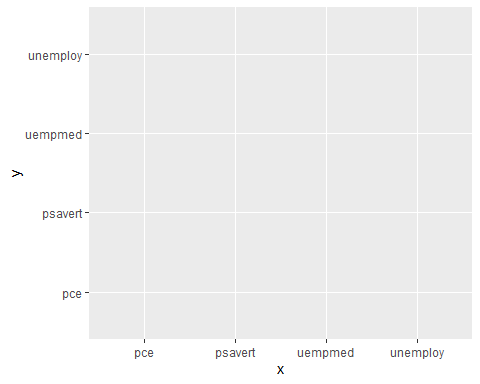
#install.packages("reshape2")  
require(reshape2)

## Loading required package: reshape2

# Also load the Scales package for some extra plotting features  
#install.packages("scales")  
library(scales)  
  
econCor = cor(economics [ , c(2,4:6)])  
# use "melt()" to change into long format  
?melt() # Help on melt function  
econMelt = melt(econCor, varnames = c("x" ,"y"), value.name = "Correlation")  
# Order it according to correlation  
econMelt = econMelt[order(econMelt$Correlation),]  
# Display the melted data  
econMelt

## x y Correlation  
## 2 psavert pce -0.7928546  
## 5 pce psavert -0.7928546  
## 7 uempmed psavert -0.3251377  
## 10 psavert uempmed -0.3251377  
## 8 unemploy psavert -0.3093769  
## 14 psavert unemploy -0.3093769  
## 4 unemploy pce 0.6145176  
## 13 pce unemploy 0.6145176  
## 3 uempmed pce 0.7269616  
## 9 pce uempmed 0.7269616  
## 12 unemploy uempmed 0.8693097  
## 15 uempmed unemploy 0.8693097  
## 1 pce pce 1.0000000  
## 6 psavert psavert 1.0000000  
## 11 uempmed uempmed 1.0000000  
## 16 unemploy unemploy 1.0000000

# Let's Visualize Correlation  
## Plot it with ggplot  
# Initialize the plot with x and y on the respective axes  
ggplot(econMelt,aes (x=x, y=y),geom\_tile(aes(fill = Correlation)),scale\_fill\_gradient2(low = muted("red"), mid = "white", high = "steelblue",guide = guide\_colorbar(ticks=FALSE, barheight=10), limit=c(-1,1), theme\_minimal(), labs(x= NULL, y=NULL)))



## Coorelations

# Correlation  
  
# Prepare the Data  
mydata <- mtcars[, c(1,3,4,5,6,7)]  
head(mydata)

## mpg disp hp drat wt qsec  
## Mazda RX4 21.0 160 110 3.90 2.620 16.46  
## Mazda RX4 Wag 21.0 160 110 3.90 2.875 17.02  
## Datsun 710 22.8 108 93 3.85 2.320 18.61  
## Hornet 4 Drive 21.4 258 110 3.08 3.215 19.44  
## Hornet Sportabout 18.7 360 175 3.15 3.440 17.02  
## Valiant 18.1 225 105 2.76 3.460 20.22

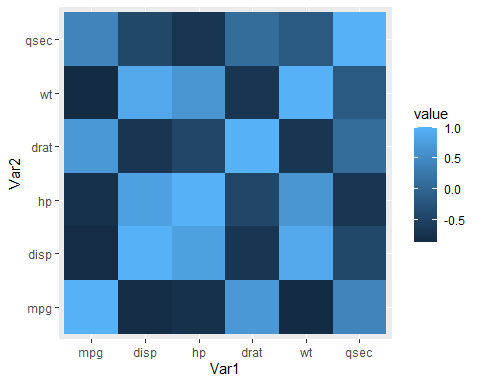
# Compute the correlation matrix - cor()  
cormat <- round(cor(mydata),2)  
head(cormat)

## mpg disp hp drat wt qsec  
## mpg 1.00 -0.85 -0.78 0.68 -0.87 0.42  
## disp -0.85 1.00 0.79 -0.71 0.89 -0.43  
## hp -0.78 0.79 1.00 -0.45 0.66 -0.71  
## drat 0.68 -0.71 -0.45 1.00 -0.71 0.09  
## wt -0.87 0.89 0.66 -0.71 1.00 -0.17  
## qsec 0.42 -0.43 -0.71 0.09 -0.17 1.00

# Create the correlation heatmap with ggplot2  
# The package reshape is required to melt the correlation matrix.  
library(reshape2)  
melted\_cormat <- melt(cormat)  
head(melted\_cormat)

## Var1 Var2 value  
## 1 mpg mpg 1.00  
## 2 disp mpg -0.85  
## 3 hp mpg -0.78  
## 4 drat mpg 0.68  
## 5 wt mpg -0.87  
## 6 qsec mpg 0.42

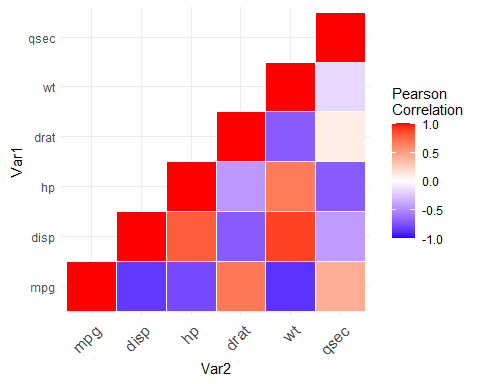
#The function geom\_tile()[ggplot2 package] is used to visualize the correlation matrix :  
library(ggplot2)  
ggplot(data = melted\_cormat, aes(x=Var1, y=Var2, fill=value)) +   
 geom\_tile()



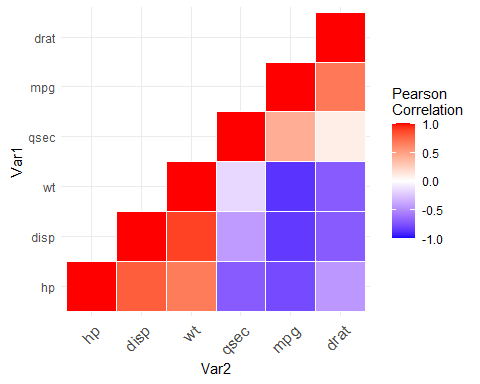
#Doesnot Look Great.. Let's Enhance the viz!  
  
#Get the lower and upper triangles of the correlation matrix  
## a correlation matrix has redundant information. We'll use the functions below to set half of it to NA.  
  
# Get lower triangle of the correlation matrix  
get\_lower\_tri<-function(cormat){  
 cormat[upper.tri(cormat)] <- NA  
 return(cormat)  
}  
# Get upper triangle of the correlation matrix  
get\_upper\_tri <- function(cormat){  
 cormat[lower.tri(cormat)]<- NA  
 return(cormat)  
}  
  
upper\_tri <- get\_upper\_tri(cormat)  
upper\_tri

## mpg disp hp drat wt qsec  
## mpg 1 -0.85 -0.78 0.68 -0.87 0.42  
## disp NA 1.00 0.79 -0.71 0.89 -0.43  
## hp NA NA 1.00 -0.45 0.66 -0.71  
## drat NA NA NA 1.00 -0.71 0.09  
## wt NA NA NA NA 1.00 -0.17  
## qsec NA NA NA NA NA 1.00

# Finished correlation matrix heatmap  
## Melt the correlation data and drop the rows with NA values  
# Melt the correlation matrix  
library(reshape2)  
melted\_cormat <- melt(upper\_tri, na.rm = TRUE)  
# Heatmap  
library(ggplot2)  
ggplot(data = melted\_cormat, aes(Var2, Var1, fill = value))+  
 geom\_tile(color = "white")+  
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "white",   
 midpoint = 0, limit = c(-1,1), space = "Lab",   
 name="Pearson\nCorrelation") +  
 theme\_minimal()+   
 theme(axis.text.x = element\_text(angle = 45, vjust = 1,   
 size = 12, hjust = 1))+  
 coord\_fixed()



reorder\_cormat <- function(cormat){  
 # Use correlation between variables as distance  
 dd <- as.dist((1-cormat)/2)  
 hc <- hclust(dd)  
 cormat <-cormat[hc$order, hc$order]  
}  
  
# Reorder the correlation matrix  
cormat <- reorder\_cormat(cormat)  
upper\_tri <- get\_upper\_tri(cormat)  
# Melt the correlation matrix  
melted\_cormat <- melt(upper\_tri, na.rm = TRUE)  
# Create a ggheatmap  
ggheatmap <- ggplot(melted\_cormat, aes(Var2, Var1, fill = value))+  
 geom\_tile(color = "white")+  
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "white",   
 midpoint = 0, limit = c(-1,1), space = "Lab",   
 name="Pearson\nCorrelation") +  
 theme\_minimal()+ # minimal theme  
 theme(axis.text.x = element\_text(angle = 45, vjust = 1,   
 size = 12, hjust = 1))+  
 coord\_fixed()  
# Print the heatmap  
print(ggheatmap)



#Add correlation coefficients on the heatmap  
  
  
ggheatmap +   
 geom\_text(aes(Var2, Var1, label = value), color = "black", size = 4) +  
 theme(  
 axis.title.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 panel.grid.major = element\_blank(),  
 panel.border = element\_blank(),  
 panel.background = element\_blank(),  
 axis.ticks = element\_blank(),  
 legend.justification = c(1, 0),  
 legend.position = c(0.6, 0.7),  
 legend.direction = "horizontal")+  
 guides(fill = guide\_colorbar(barwidth = 7, barheight = 1,  
 title.position = "top", title.hjust = 0.5))

