################################ Vectors #####################

#Creating vectors of different types

intvec <- c(1,2,3)

numvec <- c(1,2,3.3)

logvec <- c(T,F,T)

charvec <- c('VISA',"MASTER",'AMEX')

mixvec <- c(1,2.3,T,'VISA')

# Initializing a vector of specific type

vec1 <- vector(mode = "character",length = 2)

vec2 <- vector(mode = "integer",length = 3)

vec3 <- vector(mode = "numeric",length=4)

vec4 <- vector(mode = "logical")

vec4 <- vector(mode = "logical",length=2)

vec5 <- vector(mode = "complex",length=2)

vec6 <- vector()

vec7 <- c()

class(vec6)

class(vec7)

vec1 <- c(1,2,3,4,5)

vec6 <- c(1,2,3)

vec6 <- c(1.2,3.4,2.7)

vec4 <- c("This is","an R class")

vec4

# You can override the defined type by storing a different type of value

vec1 <- c(T,T)

vec5 <- c(1,2)

# class vs typeof vs mode

# class -> object oriented perspective (user can change)

# typeof -> R internal perspective (user cannot change)

# mode -> S language perspective ( user can change)

class(vec5)

class(vec5) <- "integer"

class(vec5)

typeof(vec5)

typeof(vec5) <- "numeric"

mode(vec5)

mode(vec5) <- "logical"

#checking class types

ls()

get("intvec") #convert the string to object name

for( i in ls()){

print(class(get(i)))

}

sapply(ls(),get)

sapply(ls(),function(x){class(get(x))}) # More on this later

#Basic data manipulation with vectors

intvec

numvec[2]

numvec[]

logvec[2]

intvec[1,2]

intvec[c(1,2)]

intvec[1:2]

intvec[c(1,3)]

intvec[-2]

intvec

intvec\*numvec

intvec\*.5

#reverse a vector

intvec

rev(intvec)

# append a vector to another vector

append(intvec,c(10,20,9))

intvec

append(intvec,numvec)

#length and nchar functions

length(charvec) # no of items in vector

nchar(charvec) # length of each item in a vector

length(mixvec)

nchar(mixvec)

#working with named vectors

charvec

names(charvec)

names(charvec)<-c("Card1","Card2","Card3")

charvec

class(charvec)

charvec["Card2"]

charvec[2]

# accessing a non-existing element

length(numvec)

numvec[10]

numvec

numvec[10]<-NA

class(numvec)

numvec

length(numvec)

numvec[20] <- "character"

numvec

class(numvec[10])

numvec <- c(1,2,3.3)

#negative index

intvec

intvec[-1]

intvec

intvec[c(-1,-2)]

# sort vs order vs rank

intvec<-c(5,3,6,8,10,2)

sort(intvec)

intvec

sort(intvec,decreasing = TRUE)

order(intvec)

intvec[order(intvec)]

order(intvec,decreasing = T)

intvec[order(intvec,decreasing = T)]

intvec

sort(intvec)

order(intvec) #gives the original index of items in the sorted vector

rank(intvec) #gives the rank for each item in the actual vector

intvec <- c(intvec,2,3,4)

intvec

rank(intvec)

intvec2

intvec2[c(T,F,T,F,T,F,T,F,T,F)]

intvec[T]

intvec2[c(T,F)]

#finding the mode of a vector

intvec2<-c(25,22,18,30,14,30,22,10,22,30)

max(intvec2)

min(intvec2)

which.max(intvec2)

which.min(intvec2)

#method 1

tab <- table(intvec2)

tab

class(tab)

typeof(tab)

names(tab)

tab

tab==max(tab)

tab[tab==max(tab)]

names(tab[tab==max(tab)])

as.integer(names(tab[tab==max(tab)]))

#method 2

intvec2

u <- unique(intvec2)

u

match(intvec2,u)

t <- table(match(intvec2,u))==max(table(match(intvec2,u)))

t

u[t==max(t)]

#a few useful functions

intvec <- rnorm(1000,mean = 100,sd = 1)

summary(intvec)

mean(intvec)

median(intvec)

quantile(intvec)

quantile(intvec,.05)

quantile(intvec,seq(0,1,length.out = 11))

quantile(intvec,seq(0,1,length.out = 21))

?quantile

seq(1,10,2)

################################ matrix #####################

# Division, Quotient and Remainder

9/2

9%%5

-9%%5

20%/%3

20%%3

#creating a matrix

matrix(1,3) # column matrix of 1

matrix(2,4) # column matrix of 2

t(matrix(4,2))# row matrix of 4

matrix(0,3,4) # rectangular matrix of dim r,c

intvec2 <- c(1:13)

mat1 <- matrix(intvec2,nrow= 3)

mat1

mat2 <- matrix(intvec2,nrow=4,ncol=3)

mat2

mat3 <- matrix(intvec2,nrow=4,byrow=T) # default is byrow = F

mat3

# Matrix from a vector

a <- c(1:10)

a

dim(a)

dim(a) <- c(2,5)

a

dim(a) <- c(5,2)

a

# product of items in dim should equal to the length of the vector

dim(a) <- c(3,4)

dim(a) <- c(4,3)

dim(a) <- c(3,3)

# Note that the cyclicity property doesnt work in this context

class(a)

typeof(a)

mode(a)

a <- c(1:10)

b <-as.character(c(1:10))

a

b

cbind(a,b)

rownames(cbind(a,b))

colnames(cbind(a,b))

rbind(a,b)

rownames(rbind(a,b))

colnames(rbind(a,b))

#accessing the elements of a matrix

mat1

mat1[2]

mat1[10]

mat1[3,3]

#Recall that mat3 was created using byrow=T

mat3

mat3[3]

as.vector(mat3) # flatteting the matrix

mat3[3]

mat3[2,3]

mat3[3,5] # accessing a non-existing element

mat3[1,]

mat3[,1]

mat3[20]

mat3[7]

#rownames and colnames in matrix

mat4 <- matrix(c(1:6),nrow=2)

mat4

dim(mat4)

nrow(mat4)

ncol(mat4)

rownames(mat4)

rownames(mat4) <-c("R1","R2")

colnames(mat4) <- c("C1","C2","C3")

mat4

mat4["R1",]

mat4[,"C2"]

mat4["R1","C3"]

rm(mat4)

mat4 <- matrix(c(1:6),nrow=2)

mat4

dimnames(mat4) <- list(c("R1","R2"),c("C1","C2","C3"))

mat4

rm(mat4)

mat4 <- matrix(c(1:6),nrow=2,

dimnames = list(c("R1","R2"),c("C1","C2","C3")))

mat4

#Basic matrix operations

is.matrix(intvec2)

is.matrix(mat4)

is.vector(mat4)

mat1

mat1 +2

mat1 \*2

mat1 + mat1 #dim should match to add

mat1 + mat3

mat1

colSums(mat1)

rowSums(mat1)

colMeans(mat2)

rowMeans(mat2)

dim(mat2) # 4 \* 3 matrix

dim(mat1) # 3 \* 4 matrix

mat2 %\*% mat1 # matrix multiplication

mat1\*mat1 # element by element multiplication

mat1

mat1\*mat3

mattrans<-t(mat1)

mattrans

mat1 %\*% mattrans

sum(mat1[1,]\*mattrans[,1])

cbind(mat1,mat1)

cbind(mat1,mat1,mat1)

rbind(mat1,mat1)

rbind(mat1,mat1,mat1)

#Matrix algebra based operatons

matrix(1,3) # column matrix of 1

matrix(2,4) # column matrix of 2

t(matrix(4,2))# row matrix of 4

matrix(0,3,4) # data,nrow,ncol are the arguments

mat1

diag(mat1)

diag(matrix(c(1:9),3,3)) # get the diagonal elements of a matrix

intvec <- c(1,2,3)

diag(intvec) # convert a vector into a diagonal matrix

id3 <- diag(rep(1,3)) # identity matrix 3\*3

id3

identical(id3,t(id3)) # symmetric matrix

identical(mat1,t(mat1)) # non-symmetric matrix

solve(mat1) # only square matrices have an inverse

mat5<-matrix(c(1:4),2)

mat5

mat6 <- solve(mat5)

mat6

mat5 %\*% mat6 # matrix multiplied by inverse will give I

det(mat5) # determinant exists only for square matrices

class(mat1)

################################ array #####################

intvec2

array1 <- array(intvec2,c(2,2,3))

array1

array2 <- array(c(1:10),c(2,3,2))

array2

array2[]

array2[1,2,2]

array2[2,1,1]

array2[1,2,]

array2

array2[1,,]

class(array2)

class(array2[1,,])

class(array2[1,2,])

as.matrix(array2)

array2

class(as.matrix(array2))

as.vector(array2) # always a column vector though displayed in a row

t(as.vector(array2)) # evidence of above

array2[11]

array2[1,2]

array2[1,2,1]

################################ lists #####################

eq1=1

eq1 <- 10

# creating list through vector function

list1 <- vector(mode="list",length=3)

list1

list2 <- list()

list2

class(list2)

class(list1)

# Observe the following tasks. We'll learn it in due course

names(list1) <- c("one","two")

list1

list1$one <- data.frame(title=c("BE","MBA"),grade =c("A+","A+"),

stringsAsFactors = F)

list1$one

list1[3]=c(1,2,3)

list1

list1[[3]]=c(1,2)

list1

a

a <- 5

rm(a)

lst<-list(a <- c(1, 2), b <- TRUE, c<-c("a", "b", "c"))

lst1<-list(a = c(1, 2), b = TRUE, c=c("a", "b", "c"))

str(lst)

str(lst1)

lst

length(lst)

class(lst)

lst[1] # list

class(lst[1])

lst[[1]] # numeric vector

class(lst[[1]])

lst$a # we have assigned a but unable to call a . Why ?

a<-10

lst<-list(a = c(1, 2), b = TRUE, c=c("a", "b", "c"))

lst[1]

lst[[1]]

lst$a

a <- a\*2

a

lst$a

str(lst)

lst$b

lst[[2]]

lst

lst[1,2]

lst[c(1,2)] #This will extract the first two elements of list.

lst[1] #This will extract the first element of list.

lst[1:3] #This will extract the first three elements of the list.

unlist(lst)

class(lst)

class(lst[1])

class(lst[[1]])

lst

lst[[1]][2]

lst[[1]] #This will extract the values of first element.

lst[[3]] #similarly all vlaues of the third element in list.

lst[[1]][1] #this will extract the first value from the first element

lst[[1]][2] #this will extract the 2nd value from the first element

lst[[2]] # this will extract the second element of the list fully

lst[[3]][1] # this will extract the 1st value from the third element

lst[[3]][2] # this will extract the 2nd value from the third element

lst[[3]][3] # this will extract the 3rd value from the third element

class(lst[3])

lst

lst[1]

lst[3]

lst[3][1]

lst[3][2]

lst[3][10]

class(lst[3][10])

lst[3]

lst[[3]][3]

class(lst[[3]])

lst[1][1][1][1][1][1][[1]]

#GOLDEN rule "list will alway contain list "

#here inside lst 1st level of the list is the list

class(lst[1])

class(lst[2])

class(lst[3])

#now we move to the 2 layer of the list

class(lst[[1]])

class(lst[1][1])

class(lst[1][2])

lst[1][2]

lst

lst[[1]][1]

lst[1][1][1]

lst[2][1][1]

lst[3][1][1]

lst[3][1]

lst[3][1][1][1]

#list will alway contain list

class(lst[1][1][1])

class(lst[2][1][1])

class(lst[3][1][1])

lst

lst[1][2]

lst[2][2]

lst[3][2]

class(lst[[1]])

class(lst[[1]][1])

class(lst[[2]])

class(lst[[3]])

lst[[1]][3]

moh\_profile<-list(name=c("Albert","Roger",1),

marital=TRUE,

interest =c("R","Python","Scala"),

Target <- c("google","yahoo","amazon","Citi"),

acadprof = data.frame(title=c("BE","MBA"),grade =c("A+","A+"),

stringsAsFactors = F))

moh\_profile

moh\_profile$name

moh\_profile$interest

moh\_profile$Target # see the difference between = and <-

moh\_profile[[4]]

unlist(moh\_profile)

class(unlist(moh\_profile))

moh\_profile<-list(name=c("Albert","Roger"),

marital=TRUE,

interest =c("R","Python","Scala"),

Target <- c("google","yahoo","amazon","Citi"),

acadprof = list(df2=data.frame(title=c("BE","MBA"),grade =c("A+","A+"),

stringsAsFactors = F),

university=c("NTU","NUS")))

moh\_profile

moh\_profile$acadprof

moh\_profile$acadprof$df2

moh\_profile$acadprof$df2$title

class(moh\_profile$acadprof$df2)

class(moh\_profile[[3]])

class(moh\_profile[[5]])

moh\_profile[[5]][1]

class(moh\_profile[[5]][1])

moh\_profile[[5]][[1]]

class(moh\_profile[[5]][[1]])

moh\_profile[[5]][[1]][1]

class(moh\_profile[[5]][[1]][1])

moh\_profile[[4]][1]

moh\_profile[[5]][[2]]

moh\_profile[[5]][[2]][2]

moh\_profile$acadprof$university

################################ data frames #####################

#creating a data frame

name1 <- c("Antony","Ben","Cathy","Diana")

age <- c(12,18,17,19)

gender <- c("Male","Male","Female","Female")

iqF <- factor(c("High","Low","Medium","Very High"),

ordered = TRUE,

levels =c("Low","Medium","High","Very High"))

name1

age

gender

iqF

df <- data.frame(name1,age,gender,iqF)

df

df$name1

str(df)

gender <- factor(c("Male","Male","Female","Female"))

df <- data.frame(name1,age,gender,iqF,

stringsAsFactors = F)

df

df$name1

df$iqF

str(df)

unclass(df)

class(unclass(df))

names(df)

colnames(df)

rownames(df)

row.names(df) # Use rownames and colnames function for both matrices and dataframes (recent change)

attributes(df)

attributes(mat4)

class(attributes(mat4))

#working with dataframes

df

df[]

df[1] # data frame

df["name1"] # data frame

df[1,] # data frame

df[,1] # character

df[[1]] # character

df$name1 # vector

df[1,2] # vector

df[,2] # vector

df

lst<-list(a = c(1, 2), b <- TRUE, c=c("a", "b", "c"))

lst$c[3]

df[c(1,3),c("age","name1")]

df[1,1]

df$name1[1]

df[names(df)]

df[rev(names(df))]