R Data structures and Functions - 3

Relationship b/w Dataframe, matrix and list

- Dataframes are lists but lists are not data frames
- Matrix can be converted to data frames
- Dataframes have row names and column names which can be accessed and modified
- Length of dataframe is the number of columns and NOT number of rows (for which use now function)

```
df2 <- data.frame(c1=10:12,c2=c('a','b','c')); df2; str(df2)
is.data.frame(df2)
list1 <- list(c(1,2,3,4),c('a','b','c','d'))
is.list(list1)
is.data.frame(df2)
is.list(df2)
str(list1)
dim(df2)    #of matrix or dataframe
length(df2)
nrow(df2)
colnames(df2)
colnames(df2) <- c('b1','b2')
m1 <- matrix(1:12,nrow = 7,ncol = 2,byrow = T); m1
df1 <- as.data.frame(m1); df1

df3 <- cbind(df2,df2); df3 #similarly rbind
as.matrix(df3)
```

Applying a function over a collection

- lapply - over list / vector; returns a list
- sapply - over list (almost same as lapply); returns a vector or matrix
- mapply - over multiple list or vectors
- tapply - over each group (grouped by a factor variable levels)
- apply - over array / matrix margins (1 - by row, 2 - by column usually for 2 dimensional structures such as matrix or data frames)

  IMP: The result returned is of the same **length** as the argument(s)

```
lapply(df1, min)    #explore the type of the returned result

sapply(df1,min)[1]#explore the type of the returned result


vector7 <- c(F, FALSE, T, TRUE,F,T,F)

tapply(df1$V1, vector7, sum)

tapply(df1$V1, vector7, min)

tapply(df1, vector7,min)      #will give error


mapply(max,df1$V1,df1$V2)


apply(m1,1,min) #margin values: 1 - by row, result length same as no. of rows

apply(m1,2,min) #margin values: 2 - by col, result length same as no. of cols

apply(df1,1,min) #the dataframe is coerced to a matrix form using as.matrix and then applied

apply(df1,2,min)
```

```
tab1 <- read.table(file='house_copy.txt',header = T)

lapply(tab1,is.numeric)

apply(tab1,2,table)
```

Using custom function

```
f1 <- function(x){
  paste0(x[1], x[2])
}
sapply(mtcars, f1)
f2 <- function(x){
  max(x)-min(x)
}
sapply(mtcars, f2)
apply(mtcars, 2, f2)
apply(mtcars, 1, f2)
apply(mtcars, 2, f1)
```

Using custom function

```
f3 <- function(x){

  paste(x,x,sep=":",collapse=" ")

}

df1

lapply(df1,f3)

sapply(df1,f3)
```

Using custom function

```
f4 <- function(x){

  if (is.numeric(x)){

    paste0(x[1], x[2])

  }else

    "not applicable"

}

lapply(tab1, f4)
```

Using custom function

```
mapply(max,df1$V1,df1$V2)

f1 <- function(x,y){

  abs(x^2-y)

}

mapply(f1,df1$V1,df1$V2)
```

Exercises

- Given the following vectors use mapply to find the sum of all the first elements of the numerical vectors, sum of all the second elements and so on
- v1 <- c(1,4,9,1)
- v2 <- c('a','b','b','a')
- v3 <- c(2,3,4,1)
- v4 <- c(3,1,3,1)
- Create 3 vectors of the same length containing information about the age, marks, number of books read in a year and the gender for 5 students in a class. Combine these vectors into a dataframe. Use tapply function to find the average number of marks obtained for each gender group and the min number of books read for each gender group.
- With the above dataframe created use the apply function to find the average marks obtained by the students, average number of books read and the average age of the students.
- Create a matrix using the 3 numerical vectors and use the apply function to find the average marks obtained by the students, average number of books read and the average age of the students.
- With the above dataframe created use the lapply function to find the average marks obtained by the students, average number of books read and the average age of the students.
- Create 2 vectors of the same length containing the before training and after training measurement of competency on a scale of 1-10 for 5 students. For example, let vector 1 contains the pre-training competency score of the 5 students as 2,7,4,9,6 and the post training competency score as 5,6,7,6,6 respectively. Use mapply on these 2 vectors to find the difference between post and pre scores. Hint: use minus operator with quotes as a supplied function, i.e. mapply('-',…)

Using with, within and split during data transformation

```
with(mtcars, mpg[cyl == 8  &  disp > 350])
    # is the same as, but nicer than
mtcars$mpg[mtcars$cyl == 8  &  mtcars$disp > 350]

install.packages("ISwR")
library(ISwR)
head(thuesen)
with(thuesen,plot(blood.glucose,short.velocity))
plot(thuesen$blood.glucose,thuesen$short.velocity)

head(ToothGrowth)
with(ToothGrowth,tapply(len, supp, mean))
with(ToothGrowth,tapply(len, list(supp,dose), mean))
with(ToothGrowth,split(len,supp))
```

Using with, within and split during data transformation - 2

```
thu4 <- within(thuesen,{

  log.g = log(blood.glucose)

  m <- mean(log.g)

  std.bg <- log.g - m

  rm(m)

})
```

Important functions

```
subset(mtcars,hp>200,select = c(hp,mpg,cyl))

aggregate(len~supp,data=ToothGrowth,mean)



head(warpbreaks)

table(warpbreaks$wool)

table(warpbreaks$tension)

aggregate(breaks ~ wool + tension, data = warpbreaks, mean)
```

Introduction to dplyr

- install.packages("tidyverse")
- library(tidyverse)
- https://r4ds.had.co.nz/transform.html
- https://r4ds.had.co.nz/data-visualisation.html
- https://r4ds.had.co.nz/exploratory-data-analysis.html
- https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html