Exercises Session 06-11-2019

Exercise Session 06-11-2019

#Group: Santiago Blanco Sanchez, Lais Silva Almeida Zanchetta, Daniel Ferreira Zanchetta

Exercises Block I

1)

```
vector<-c(1:20)
vectorWithPos2 <- vector[2]
vector[2]<-vector[20]
vector[20]<-vectorWithPos2
vector

##  [1]  1 20  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  2
```

2)

Step 1: Create the vector

```
a <- rep(c(1,2,3,NA),5)
```

Step 2: Find out the mean, without considering the NA value

```
mean <- mean(a,na.rm=TRUE)
```

Step 3: Create one recursive function that will be used later onto sapply

```
fun <- function(x){ if(is.na(x)){x<-mean} else{x<-x}}
```

Step 4: Assign the sapply function to the vector at step 1

```
a<-sapply(a, fun)
a

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

3)

First, create 3 vectors and, then, apply the data.frame function.

```
Age <- c(22,25,18,20)
Name <- c("James","Mathew","Olivia","Stella")
Gender <- c("M","M","F","F")
df <- data.frame(Age,Name,Gender)
df
```

```
##   Age   Name Gender
## 1  22  James      M
## 2  25 Mathew      M
## 3  18 Olivia      F
## 4  20 Stella      F
```

4)

```
df[Age>21,]

##   Age   Name Gender
## 1  22  James      M
## 2  25 Mathew      M
```

5)

```
df["adult"]<-c(df$Age>21)
df

##   Age   Name Gender adult
## 1  22  James      M  TRUE
## 2  25 Mathew      M  TRUE
## 3  18 Olivia      F FALSE
## 4  20 Stella      F FALSE
```

6)

Instructions to write the iris.csv file and read it:

```
data("iris")
write.table(iris,file="iris.csv",row.names=FALSE, col.names=TRUE,sep=";")
irisdf <- read.table("iris.csv",header=TRUE,sep=";")
```

Second set of instruction to calculate by rows and columns

```
iris_subset <- unlist(lapply(irisdf, is.numeric))

#Using apply
apply(irisdf[,iris_subset], 1, mean)

##   [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400
2.700
##  [12] 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675
2.675
##  [23] 2.350 2.650 2.575 2.450 2.600 2.600 2.550 2.425 2.425 2.675
2.725
##  [34] 2.825 2.425 2.400 2.625 2.500 2.225 2.550 2.525 2.100 2.275
2.675
##  [45] 2.800 2.375 2.675 2.350 2.675 2.475 4.075 3.900 4.100 3.275
3.850
##  [56] 3.575 3.975 2.900 3.850 3.300 2.875 3.650 3.300 3.775 3.350
3.900
##  [67] 3.650 3.400 3.600 3.275 3.925 3.550 3.800 3.700 3.725 3.850
3.950
```

```
##  [78] 4.100 3.725 3.200 3.200 3.150 3.400 3.850 3.600 3.875 4.000
3.575
##  [89] 3.500 3.325 3.425 3.775 3.400 2.900 3.450 3.525 3.525 3.675
2.925
## [100] 3.475 4.525 3.875 4.525 4.150 4.375 4.825 3.400 4.575 4.200
4.850
## [111] 4.200 4.075 4.350 3.800 4.025 4.300 4.200 5.100 4.875 3.675
4.525
## [122] 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400 4.550
5.025
## [133] 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375 4.450 4.350
3.875
## [144] 4.550 4.550 4.300 3.925 4.175 4.325 3.950
```

```r
apply(irisdf[,iris_subset], 2, mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
#Using colMeans and rowMeans
colMeans(irisdf[ , iris_subset])
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
rowMeans(irisdf[ ,iris_subset])
```

```
##   [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400
2.700
##  [12] 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675
2.675
##  [23] 2.350 2.650 2.575 2.450 2.600 2.600 2.550 2.425 2.425 2.675
2.725
##  [34] 2.825 2.425 2.400 2.625 2.500 2.225 2.550 2.525 2.100 2.275
2.675
##  [45] 2.800 2.375 2.675 2.350 2.675 2.475 4.075 3.900 4.100 3.275
3.850
##  [56] 3.575 3.975 2.900 3.850 3.300 2.875 3.650 3.300 3.775 3.350
3.900
##  [67] 3.650 3.400 3.600 3.275 3.925 3.550 3.800 3.700 3.725 3.850
3.950
##  [78] 4.100 3.725 3.200 3.200 3.150 3.400 3.850 3.600 3.875 4.000
3.575
##  [89] 3.500 3.325 3.425 3.775 3.400 2.900 3.450 3.525 3.525 3.675
2.925
## [100] 3.475 4.525 3.875 4.525 4.150 4.375 4.825 3.400 4.575 4.200
4.850
## [111] 4.200 4.075 4.350 3.800 4.025 4.300 4.200 5.100 4.875 3.675
4.525
## [122] 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400 4.550
5.025
```

```
## [133] 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375 4.450 4.350
3.875
## [144] 4.550 4.550 4.300 3.925 4.175 4.325 3.950
```

7)

This function should consist of the following steps: 1)Define a subset of the iris dataframe, to contain only numeric values. Otherwise, it wont be possible to apply the rowMeans and colMeans functions (both expect numeric dataframe/matrix/vector/list) 2)Calculate the Row Means 3)Calculate the Column Means 4) Return both values

```
funMean <- function(dataset){
  subsetDataSetNumeric <- unlist(sapply(dataset,is.numeric))
  print(rowMeans(dataset[,subsetDataSetNumeric]))
  print(colMeans(dataset[,subsetDataSetNumeric]))
}
funMean(irisdf)
```

```
##   [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400
2.700
##  [12] 2.500 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675
2.675
##  [23] 2.350 2.650 2.575 2.450 2.600 2.600 2.550 2.425 2.425 2.675
2.725
##  [34] 2.825 2.425 2.400 2.625 2.500 2.225 2.550 2.525 2.100 2.275
2.675
##  [45] 2.800 2.375 2.675 2.350 2.675 2.475 4.075 3.900 4.100 3.275
3.850
##  [56] 3.575 3.975 2.900 3.850 3.300 2.875 3.650 3.300 3.775 3.350
3.900
##  [67] 3.650 3.400 3.600 3.275 3.925 3.550 3.800 3.700 3.725 3.850
3.950
##  [78] 4.100 3.725 3.200 3.200 3.150 3.400 3.850 3.600 3.875 4.000
3.575
##  [89] 3.500 3.325 3.425 3.775 3.400 2.900 3.450 3.525 3.525 3.675
2.925
## [100] 3.475 4.525 3.875 4.525 4.150 4.375 4.825 3.400 4.575 4.200
4.850
## [111] 4.200 4.075 4.350 3.800 4.025 4.300 4.200 5.100 4.875 3.675
4.525
## [122] 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400 4.550
5.025
## [133] 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375 4.450 4.350
3.875
## [144] 4.550 4.550 4.300 3.925 4.175 4.325 3.950
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

8)

```r
funFibonacci <- function(n){
  x <- numeric(n)
  n1 <- -1
  n2 <- 1
  for(i in 1:length(x)){
   x[i] <- n1 + n2
   n1 <- n2
   n2 <- x[i]
  }
  x
}
funFibonacci(10)

##  [1]  0  1  1  2  3  5  8 13 21 34
```

9)

Firstly reading the dataset

```r
precipitaciones <-
read.table("C:/Users/Daniel/Documents/precipitacionsbarcelonadesde1786.cs
v", header=TRUE, sep=",")
View(precipitaciones)
```

Secondly, some of the functions seen have been applied to the dataset

```r
colSums(precipitaciones,na.rm=TRUE)

##                  Any     Precip_Acum_Gener    Precip_Acum_Febrer
##             443166.0                8894.2                8180.0
##      Precip_Acum_Marc     Precip_Acum_Abril      Precip_Acum_Maig
##              11440.4               12354.3               12558.2
##      Precip_Acum_Juny    Precip_Acum_Juliol     Precip_Acum_Agost
##               8714.0                6087.6                9205.5
## Precip_Acum_Setembre  Precip_Acum_Octubre Precip_Acum_Novembre
##              18436.1               18500.7               13623.2
## Precip_Acum_Desembre
##              10031.4

rowSums(precipitaciones[,2:13],na.rm=TRUE)

##   [1]  722.2  625.1  728.8  342.5  855.7  591.3  291.8  736.5  653.6
556.1
##  [11]  623.7  555.5  496.2  506.5  602.4  736.6  524.7  557.9  640.0
597.8
##  [21]  541.8  505.6  668.5  402.5  667.4  729.0  317.7  285.3  584.0
421.5
##  [31]  325.0  215.6  506.2  630.0  719.2  501.2  280.5  239.4  292.8
510.7
##  [41]  711.6  454.1  342.3  500.1  397.8  555.6  496.7  546.6  439.6
```

```
294.6
## [51]  392.4  421.9  597.0  472.6  555.6  561.5  644.9  723.0  768.0
649.0
## [61]  651.0  556.1  659.2  679.3  441.3  544.1  584.1  858.6  435.9
620.0
## [71]  836.5  838.6  568.4  559.8  573.0  625.9  774.9  475.8  566.1
541.3
## [81]  580.2  352.6  482.3  386.2  593.2  689.4  980.0  452.7  612.5
927.4
## [91]  733.4  361.0  329.2  717.4  513.4  571.7  687.7  542.8  696.4
536.3
## [101]  373.7  546.7  599.3  605.3  699.4  757.8  562.9  497.6  625.2
546.0
## [111]  622.3  735.1  683.7  598.8  604.4 1030.6  661.7  468.4  448.0
525.9
## [121]  674.1  818.5  626.0  547.9  552.6  506.3  447.3  610.3  740.0
640.9
## [131]  549.4  647.8  694.8  489.3  680.2  886.3  550.7  613.2  443.8
637.3
## [141]  662.9  555.7  748.7  446.1  541.4  636.9  801.9  711.5  517.0
518.2
## [151]  690.1  403.2  529.8  674.4  483.8  585.1  621.5  867.8  745.9
445.9
## [161]  618.7  443.6  629.2  449.3  479.7  964.7  421.8  593.1  579.9
546.9
## [171]  513.9  513.9  465.9  963.8  582.7  417.1  839.0  790.8  481.9
586.0
## [181]  563.1  570.3  578.1  763.5  488.1 1122.7  795.5  456.6  559.2
549.4
## [191]  708.8  815.9  504.9  763.9  538.9  508.3  713.9  678.7  669.7
463.9
## [201]  585.5  982.3  661.4  499.2  598.1  819.3  595.3  676.6  745.7
504.1
## [211]  982.4  572.0  485.1  521.4  469.1  472.3  963.5  601.3  588.4
558.4
## [221]  474.6  493.6  599.1  524.3  720.6  865.5  479.7  580.0  692.4
345.8
## [231]  480.2  518.4  988.0
```

```r
colMeans(precipitaciones, na.rm=TRUE)
```

```
##                  Any     Precip_Acum_Gener   Precip_Acum_Febrer
##           1902.00000              38.17253             35.10730
##       Precip_Acum_Marc     Precip_Acum_Abril     Precip_Acum_Maig
##             49.10043              53.02275             53.89785
##       Precip_Acum_Juny    Precip_Acum_Juliol    Precip_Acum_Agost
##             37.39914              26.12704             39.50858
## Precip_Acum_Setembre   Precip_Acum_Octubre Precip_Acum_Novembre
##             79.12489              79.40215             58.46867
```

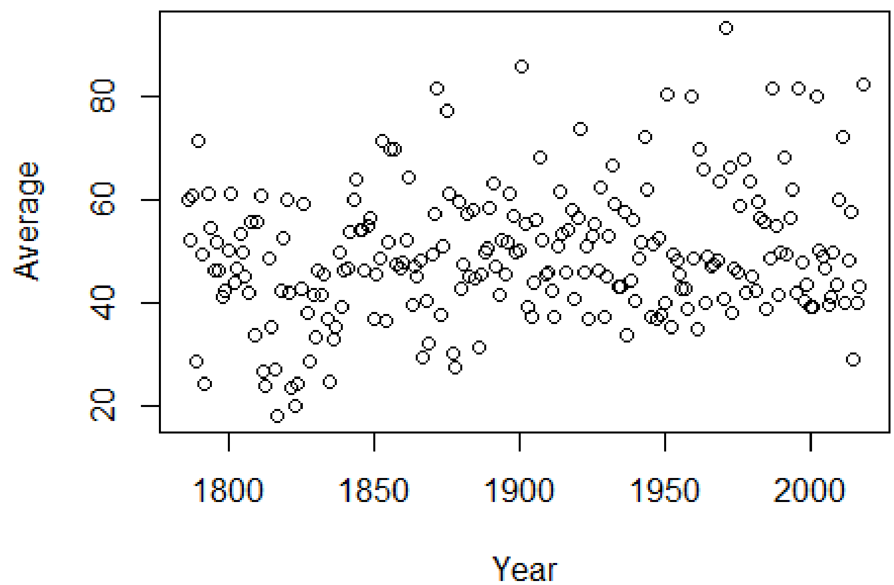
```
## Precip_Acum_Desembre
##              43.05322
```

```r
rowMeans(precipitaciones[,2:13],na.rm=TRUE)
```

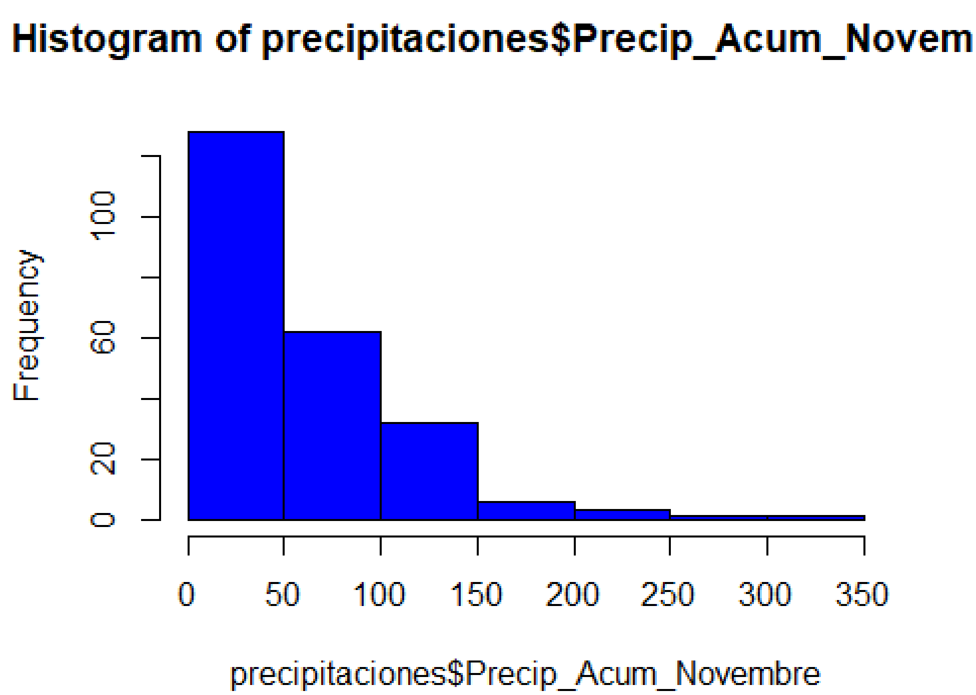
```
##   [1] 60.18333 52.09167 60.73333 28.54167 71.30833 49.27500 24.31667
##   [8] 61.37500 54.46667 46.34167 51.97500 46.29167 41.35000 42.20833
##  [15] 50.20000 61.38333 43.72500 46.49167 53.33333 49.81667 45.15000
##  [22] 42.13333 55.70833 33.54167 55.61667 60.75000 26.47500 23.77500
##  [29] 48.66667 35.12500 27.08333 17.96667 42.18333 52.50000 59.93333
##  [36] 41.76667 23.37500 19.95000 24.40000 42.55833 59.30000 37.84167
##  [43] 28.52500 41.67500 33.15000 46.30000 41.39167 45.55000 36.63333
##  [50] 24.55000 32.70000 35.15833 49.75000 39.38333 46.30000 46.79167
##  [57] 53.74167 60.25000 64.00000 54.08333 54.25000 46.34167 54.93333
##  [64] 56.60833 36.77500 45.34167 48.67500 71.55000 36.32500 51.66667
##  [71] 69.70833 69.88333 47.36667 46.65000 47.75000 52.15833 64.57500
##  [78] 39.65000 47.17500 45.10833 48.35000 29.38333 40.19167 32.18333
##  [85] 49.43333 57.45000 81.66667 37.72500 51.04167 77.28333 61.11667
##  [92] 30.08333 27.43333 59.78333 42.78333 47.64167 57.30833 45.23333
##  [99] 58.03333 44.69167 31.14167 45.55833 49.94167 50.44167 58.28333
## [106] 63.15000 46.90833 41.46667 52.10000 45.50000 51.85833 61.25833
## [113] 56.97500 49.90000 50.36667 85.88333 55.14167 39.03333 37.33333
## [120] 43.82500 56.17500 68.20833 52.16667 45.65833 46.05000 42.19167
## [127] 37.27500 50.85833 61.66667 53.40833 45.78333 53.98333 57.90000
## [134] 40.77500 56.68333 73.85833 45.89167 51.10000 36.98333 53.10833
## [141] 55.24167 46.30833 62.39167 37.17500 45.11667 53.07500 66.82500
## [148] 59.29167 43.08333 43.18333 57.50833 33.60000 44.15000 56.20000
## [155] 40.31667 48.75833 51.79167 72.31667 62.15833 37.15833 51.55833
## [162] 36.96667 52.43333 37.44167 39.97500 80.39167 35.15000 49.42500
## [169] 48.32500 45.57500 42.82500 42.82500 38.82500 80.31667 48.55833
## [176] 34.75833 69.91667 65.90000 40.15833 48.83333 46.92500 47.52500
## [183] 48.17500 63.62500 40.67500 93.55833 66.29167 38.05000 46.60000
## [190] 45.78333 59.06667 67.99167 42.07500 63.65833 44.90833 42.35833
## [197] 59.49167 56.55833 55.80833 38.65833 48.79167 81.85833 55.11667
## [204] 41.60000 49.84167 68.27500 49.60833 56.38333 62.14167 42.00833
## [211] 81.86667 47.66667 40.42500 43.45000 39.09167 39.35833 80.29167
## [218] 50.10833 49.03333 46.53333 39.55000 41.13333 49.92500 43.69167
## [225] 60.05000 72.12500 39.97500 48.33333 57.70000 28.81667 40.01667
## [232] 43.20000 82.33333
```

Finally, exploring the plots

```r
y <- rowMeans(precipitaciones[,2:13],na.rm=TRUE)
x <- precipitaciones[1]
dataToPlot <- data.frame(x,y)
plot(dataToPlot,type="p",xlab="Year",ylab="Average")
```




```
hist(precipitaciones$Precip_Acum_Novembre, col="blue")
```



Exercises Block II

###1)

```
choices<-read.table("C:\\Users\\Daniel\\Documents\\Certificados &
Faculdade\\UPC Master Big Data\\Data Analytics\\Choices.csv",
header=TRUE, sep =";")
View(choices)
str(choices) #Code to show the class of all variables

## 'data.frame':    36000 obs. of  5 variables:
##  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ COUNTRY  : int  6 2 2 2 2 2 2 2 2 2 ...
##  $ CHOICE_ID: int  1 1 1 1 1 2 2 1 1 1 ...
##  $ INFO     : Factor w/ 4 levels "A","B","C","D": 1 1 1 1 1 2 2 1 1 1
...
##  $ MEASURE  : Factor w/ 5 levels "I","II","III",..: 4 4 4 4 4 3 3 4 4
4 ...
```

###2)

```
choices_conting<-table(choices$INFO,choices$MEASURE) #creation of the
contingency table
choices_conting

##
##        I   II  III   IV    V
##   A 1892 2625 1708 2115 1414
##   B 1360 2669 1485 2240 1923
##   C 1677 2186 2093 2745 1627
##   D 1507 1973  855  975  931
```

###3)

```
conTableWithSums<-data.frame(addmargins(choices_conting))
conTableWithSums

##    Var1 Var2  Freq
## 1     A    I  1892
## 2     B    I  1360
## 3     C    I  1677
## 4     D    I  1507
## 5   Sum    I  6436
## 6     A   II  2625
## 7     B   II  2669
## 8     C   II  2186
## 9     D   II  1973
## 10  Sum   II  9453
## 11    A  III  1708
## 12    B  III  1485
## 13    C  III  2093
## 14    D  III   855
```

```
## 15  Sum  III  6141
## 16    A   IV  2115
## 17    B   IV  2240
## 18    C   IV  2745
## 19    D   IV   975
## 20  Sum   IV  8075
## 21    A    V  1414
## 22    B    V  1923
## 23    C    V  1627
## 24    D    V   931
## 25  Sum    V  5895
## 26    A  Sum  9754
## 27    B  Sum  9677
## 28    C  Sum 10328
## 29    D  Sum  6241
## 30  Sum  Sum 36000
```

###4)

```
chisq.test(choices_conting)
```

```
## 
##  Pearson's Chi-squared test
## 
## data:  choices_conting
## X-squared = 860.66, df = 12, p-value < 2.2e-16
```

Reply: In this scenario, given the p-value less than the 0.05 level, we can undertand that Measure and Info are dependent variables.