TP-2

Exercise 1

Create two *data.frames* A and B. From these two *data.frames* get the C data frame. What function we will use in this case?

A =

Ident	sexe	Poids
1	H	75
2	F	68
3	F	48
4	H	72
5	H	83

B =

Ident	sexe	Taille
1	H	182
2	F	165
3	F	160
4	H	178
5	H	183

C =

Ident	sexe	Poids	Taille
1	H	75	182
2	F	68	165
3	F	48	160
4	H	72	178
5	H	83	183

First， we create the data frame A and data frame B, in this case, the function is to add different columns of each data frame(Poids of A and Taille of B) together and keep the same columns(Idents don't change)

so the function is merge()

Data Frame A:

```
> A<-data.frame(
+ Ident=c(1,2,3,4,5),
+ sexe=c("H","F","F","H","H"),
+ Poids=c(75,68,48,72,83))
> print(A)
  Ident sexe Poids
1     1    H    75
2     2    F    68
3     3    F    48
4     4    H    72
5     5    H    83
```

Data Frame B:

```
> B<-data.frame(
+ Ident=c(1,2,3,4,5),
+ sexe=c("H","F","F","H","H"),
+ Taille=c(182,165,160,178,183))
> print(B)
  Ident sexe Taille
1     1    H    182
2     2    F    165
3     3    F    160
4     4    H    178
5     5    H    183
```

Function merge(A,B,all=TRUE):

```
> C<-merge(A,B,all=TRUE)
> print(C)
  Ident sexe Poids Taille
1     1    H    75    182
2     2    F    68    165
3     3    F    48    160
4     4    H    72    178
5     5    H    83    183
> |
```

Exercise 2

Extract all numbers between 2 and 3 of the following vector:

> x=c(0.2, 0.6, 2.1, 3.7, 2.8, 2.7, 1.9, 2.3, 5.9)

We select the number between 2 and 3 in x by using &

```
> x=c(0.2, 0.6, 2.1, 3.7, 2.8, 2.7, 1.9, 2.3, 5.9)
> print(x[2<x&x<3])
[1] 2.1 2.8 2.7 2.3
> |
```

Exercise 3

Create the following matrix Y (respecting the row names and column names):

	column 1	column 2	column 3	column 4
row-1	1	6	5	0
row-2	0	6	6	1
row-3	3	0	2	2
row-4	4	4	3	4

We create the matrix by using function matrix

```
> rownames=c("row-1","row-2","row-3","row-4")
> colnames=c("col-1","col-2","col-3","col-4")
>  number=c(1,6,5,0,0,6,6,1,3,0,2,2,4,4,3,4)
> m<-matrix(number,nrow=4,ncol=4,byrow=TRUE,dimnames=list(rownames,colnames))
> print(m)
      col-1 col-2 col-3 col-4
row-1     1     6     5     0
row-2     0     6     6     1
row-3     3     0     2     2
row-4     4     4     3     4
>
```

Calculate the determinant and invert the matrix using the necessary functions.

Determinant：

```
> print(det(m))
[1] 130
>
```

Invert the matrix:

```
> print(solve(m))
            row-1      row-2      row-3         row-4
col-1  0.40000000 -0.4000000  0.2000000  2.775558e-17
col-2  0.17692308 -0.1384615 -0.3153846  1.923077e-01
col-3 -0.09230769  0.2461538  0.3384615 -2.307692e-01
col-4 -0.50769231  0.3538462 -0.1384615  2.307692e-01
>
```

Exercise 4

a) Load the “Orange” data (available in R). Calculate the basic statistics (mean, standard deviation, min, etc..) of the last two variables of this data set.

Load the Orange Data and here we choose 2 variables “age” and “circumference”：

```
> Orange
   Tree  age circumference
1     1  118            30
2     1  484            58
3     1  664            87
4     1 1004           115
```

Mean, standard deviation, min, etc..) of variables age and circumference

```
> mean(Orange$age)
[1] 922.1429
> median(Orange$age)
[1] 1004
> var(Orange$age)
[1] 241930.7
> sd(Orange$age)
[1] 491.8645
> max(Orange$age)
[1] 1582
> min(Orange$age)
[1] 118
>
```

```
> mean(Orange$circumference)
[1] 115.8571
> median(Orange$circumference)
[1] 115
> var(Orange$circumference)
[1] 3304.891
> sd(Orange$circumference)
[1] 57.48818
> max(Orange$circumference)
[1] 214
> min(Orange$circumference)
[1] 30
>
```

b) Calculate the quartiles of both variables.

We can use the function summary() or quantile() to calcuate

```
> apply(Orange[,2:3],2,FUN=quantile)
      age circumference
0%    118          30.0
25%   484          65.5
50%  1004         115.0
75%  1372         161.5
100% 1582         214.0
>
```

c) Using the ***"apply"*** function, calculate all deciles of both variables using the ***"probs"*** argument of the ***"quantile "*** function.

```
> apply(Orange[,2:3],2,FUN=quantile,probs=c(1:10/10))
      age circumference
10%   118          32.4
20%   484          56.6
30%   664          76.2
40%   664         109.8
50%  1004         115.0
60%  1231         139.4
70%  1231         144.4
80%  1372         172.4
90%  1582         193.4
100% 1582         214.0
>
```

Exercise 5

a) Create the vector **k** formed of three times the sequence of numbers (8; 2; 6).

```
> u<-c(8,2,6)
> k<-u^3
> k
[1] 512   8 216
> 
```

b) Create the vector **w** composed of seven times the number 4, 5 times the number 9 and 3 times the number 2 (by two different methods).

First method

```
> w=c(4^7,9^5,2^3)
> w
[1] 16384 59049     8
> 
```

Second method

```
> a<-c(4,9,2)
> b<-c(7,5,3)
> w<-a^b
> w
[1] 16384 59049     8
```

Exercise 6

a) Enter the variable **"size"** which contain the following 9 values:
178, 175, 160, 191, 176, 155, 163, 174, 182.

```
> size=c(178, 175, 160, 191, 176, 155, 163, 174, 182)
> size
[1] 178 175 160 191 176 155 163 174 182
> 
```

b) Enter the variable **"size_1"** containing the following 5 values: 164, 172, 156, 195, 166.

```
> size_1=c(164, 172, 156, 195, 166)
> size_1
[1] 164 172 156 195 166
> 
```

c) From the variable **"size"** and **"size_1"**, create the variable **"new.size"** containing: the five values of **"size1"** repeated twice and the last seven values of **"size"** .

```
> new.size<-c(rep(size_1,times=2),tail(size,7))
> new.size
 [1] 164 172 156 195 166 164 172 156 195 166 160 191 176 155 163 174 182
> 
```

Using the method rep() to realize the repeat of the vector and tail() to select numbers from the end of the vector

d) Save in your working directory, the variable **“new.size”** in a .csv format file.

```
> getwd()
[1] "C:/Users/E560/Documents"
> write.csv(new.size,file="new.size.csv",row.names=FALSE,quote=FALSE)
> read.csv("new.size.csv")
     x
1  164
2  172
3  156
4  195
5  166
6  164
7  172
8  156
9  195
10 166
11 160
12 191
13 176
14 155
15 163
16 174
17 182
> 
```

We use the method write() to store csv file and read() to load csv fil

Exercise 7

Load the ***"iris"*** data set, and then view the first 7 lines. Create a subset of data containing only the data from the modality ***"versicolor"*** of the variable **"*Species*"** (call this new data set ***"new.iris"***).

```
> new.iris=iris[(iris$Species %in% c("versicolor")), ]
> new.iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
51           7.0         3.2          4.7         1.4 versicolor
52           6.4         3.2          4.5         1.5 versicolor
53           6.9         3.1          4.9         1.5 versicolor
54           5.5         2.3          4.0         1.3 versicolor
55           6.5         2.8          4.6         1.5 versicolor
56           5.7         2.8          4.5         1.3 versicolor
57           6.3         3.3          4.7         1.6 versicolor
58           4.9         2.4          3.3         1.0 versicolor
59           6.6         2.9          4.6         1.3 versicolor
60           5.2         2.7          3.9         1.4 versicolor
61           5.0         2.0          3.5         1.0 versicolor
62           5.9         3.0          4.2         1.5 versicolor
63           6.0         2.2          4.0         1.0 versicolor
64           6.1         2.9          4.7         1.4 versicolor
65           5.6         2.9          3.6         1.3 versicolor
66           6.7         3.1          4.4         1.4 versicolor
67           5.6         3.0          4.5         1.5 versicolor
68           5.8         2.7          4.1         1.0 versicolor
69           6.2         2.2          4.5         1.5 versicolor
70           5.6         2.5          3.9         1.1 versicolor
71           5.9         3.2          4.8         1.8 versicolor
72           6.1         2.8          4.0         1.3 versicolor
73           6.3         2.5          4.9         1.5 versicolor
74           6.1         2.8          4.7         1.2 versicolor
75           6.4         2.9          4.3         1.3 versicolor
76           6.6         3.0          4.4         1.4 versicolor
77           6.8         2.8          4.8         1.4 versicolor
78           6.7         3.0          5.0         1.7 versicolor
79           6.0         2.9          4.5         1.5 versicolor
80           5.7         2.6          3.5         1.0 versicolor
81           5.5         2.4          3.8         1.1 versicolor
82           5.5         2.4          3.7         1.0 versicolor
83           5.8         2.7          3.9         1.2 versicolor
84           6.0         2.7          5.1         1.6 versicolor
85           5.4         3.0          4.5         1.5 versicolor
86           6.0         3.4          4.5         1.6 versicolor
87           6.7         3.1          4.7         1.5 versicolor
88           6.3         2.3          4.4         1.3 versicolor
89           5.6         3.0          4.1         1.3 versicolor
90           5.5         2.5          4.0         1.3 versicolor
91           5.5         2.6          4.4         1.2 versicolor
92           6.1         3.0          4.6         1.4 versicolor
93           5.8         2.6          4.0         1.2 versicolor
94           5.0         2.3          3.3         1.0 versicolor
95           5.6         2.7          4.2         1.3 versicolor
96           5.7         3.0          4.2         1.2 versicolor
97           5.7         2.9          4.2         1.3 versicolor
98           6.2         2.9          4.3         1.3 versicolor
99           5.1         2.5          3.0         1.1 versicolor
100          5.7         2.8          4.1         1.3 versicolor
> 
```

b) Sort by descending order the ***"new.iris"*** data according to the variable Sepal.Length.

```
> new.iris[order(new.iris$Sepal.Length,decreasing = TRUE), ]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
51           7.0         3.2          4.7         1.4 versicolor
53           6.9         3.1          4.9         1.5 versicolor
77           6.8         2.8          4.8         1.4 versicolor
66           6.7         3.1          4.4         1.4 versicolor
78           6.7         3.0          5.0         1.7 versicolor
87           6.7         3.1          4.7         1.5 versicolor
59           6.6         2.9          4.6         1.3 versicolor
76           6.6         3.0          4.4         1.4 versicolor
55           6.5         2.8          4.6         1.5 versicolor
52           6.4         3.2          4.5         1.5 versicolor
75           6.4         2.9          4.3         1.3 versicolor
57           6.3         3.3          4.7         1.6 versicolor
73           6.3         2.5          4.9         1.5 versicolor
88           6.3         2.3          4.4         1.3 versicolor
69           6.2         2.2          4.5         1.5 versicolor
98           6.2         2.9          4.3         1.3 versicolor
64           6.1         2.9          4.7         1.4 versicolor
72           6.1         2.8          4.0         1.3 versicolor
74           6.1         2.8          4.7         1.2 versicolor
92           6.1         3.0          4.6         1.4 versicolor
63           6.0         2.2          4.0         1.0 versicolor
79           6.0         2.9          4.5         1.5 versicolor
84           6.0         2.7          5.1         1.6 versicolor
86           6.0         3.4          4.5         1.6 versicolor
62           5.9         3.0          4.2         1.5 versicolor
71           5.9         3.2          4.8         1.8 versicolor
68           5.8         2.7          4.1         1.0 versicolor
83           5.8         2.7          3.9         1.2 versicolor
93           5.8         2.6          4.0         1.2 versicolor
56           5.7         2.8          4.5         1.3 versicolor
80           5.7         2.6          3.5         1.0 versicolor
96           5.7         3.0          4.2         1.2 versicolor
97           5.7         2.9          4.2         1.3 versicolor
100          5.7         2.8          4.1         1.3 versicolor
65           5.6         2.9          3.6         1.3 versicolor
67           5.6         3.0          4.5         1.5 versicolor
70           5.6         2.5          3.9         1.1 versicolor
89           5.6         3.0          4.1         1.3 versicolor
95           5.6         2.7          4.2         1.3 versicolor
54           5.5         2.3          4.0         1.3 versicolor
81           5.5         2.4          3.8         1.1 versicolor
82           5.5         2.4          3.7         1.0 versicolor
90           5.5         2.5          4.0         1.3 versicolor
91           5.5         2.6          4.4         1.2 versicolor
85           5.4         3.0          4.5         1.5 versicolor
60           5.2         2.7          3.9         1.4 versicolor
99           5.1         2.5          3.0         1.1 versicolor
61           5.0         2.0          3.5         1.0 versicolor
94           5.0         2.3          3.3         1.0 versicolor
58           4.9         2.4          3.3         1.0 versicolor
> 
```

Exercise 8
Convert the matrix A of type *character* in a *digital* matrix.

> A
     [,1] [,2]
[1,] "8" "16"
[2,] "9" "2"

First we create a matrix A, then we use the method as.obj to convert

```
> A<-matrix(c(8,16,9,2),nrow=2,ncol=2,byrow=TRUE)
> A
     [,1] [,2]
[1,]    8   16
[2,]    9    2
> is.numeric(A)
[1] TRUE
> as.character(A)
[1] "8"  "9"  "16" "2"
> A<-as.character(A)
> is.character(A)
[1] TRUE
>
```

Exercise 9
Create the following data frame:

> person
  height weight age c.eyes
1   160   52 18   green
2   180   96 43   blue
3   175   60 29   blue

First step is to create the data frame

```
> person<-data.frame(
+ height=c(160,180,175),
+ weight=c(52,96,60),
+ age=c(18,43,29),
+ c.eyes=c("green","blue","blue"))
> person
  height weight age c.eyes
1    160     52  18   green
2    180     96  43    blue
3    175     60  29    blue
>
```

1) Change the name of the column 3 by "new.age"

Use the colnames() function

```
> colnames(person)[3]<-"new.age"
> person
  height weight new.age c.eyes
1    160     52      18   green
2    180     96      43    blue
3    175     60      29    blue
>
```

2) Change the name of the line 2 by "Mary"

Use the rownames() function

```
> person
     height weight new.age c.eyes
1       160     52      18  green
Mary    180     96      43   blue
3       175     60      29   blue
> 
```

3) Delete the row names

```
> row.names(person)<-NULL
> row.names(person)<-c()
> 
```

Setting the row.names equals NULL or just a vector with no content

4) Change all column names by a, b, c, d

```
> colnames(person)<-c("a","b","c","d")
> person
    a  b  c     d
1 160 52 18 green
2 180 96 43  blue
3 175 60 29  blue
> 
```

5) Extract the element of row 1 and column 3

```
> person
    a  b  c     d
1 160 52 18 green
2 180 96 43  blue
3 175 60 29  blue
> person[1,3]
[1] 18
> 
```

6) Extract the variable 2 (result in data.frame , result in vector)

Result in Dataframe

```
> person["weight"]
  weight
1     52
2     96
3     60
> t<-(person["weight"])
> is.data.frame(t)
[1] TRUE
> 
```

Result in Dataframe

```
> result_vector<-as.vector(person$weight)
> result_vector
[1] 52 96 60
> is.vector(result_vector)
[1] TRUE
>
```

7) Extract the element 1 and 3 of the variable 3

```
> person$age[c(1,3)]
[1] 18 29
>
```

8) Extract the upper 160 and lower 180 values of the 'height' variable

```
> h<-person$height
> h[h>160&h<180]
[1] 175
>
```

9) Extract the weight values of the people whose height values are greater than 170

```
> person$weight[person$height>170]
[1] 96 60
> person
  height weight age c.eyes
1    160     52  18  green
2    180     96  43   blue
3    175     60  29   blue
>
```

10) Extract all people who have a weight greater than 52 kg

```
> person[which(person$weight>52),]
  height weight age c.eyes
2    180     96  43   blue
3    175     60  29   blue
>
```

11) Change the height of the first 2 people per 190 and 158

```
> person[1:2,1:2]<-c(190,158)
> person
  height weight age c.eyes
1    190    190  18  green
2    158    158  43   blue
3    175     60  29   blue
>
```

Exercise 10

Create the following list:

> my_list

[[1]]

[1] 5

[[2]]

[1] 160 180 175

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

[[4]]
  height weight age c.eyes
1    160     52  18   green
2    180     96  43    blue
3    175     60  29    blue
```

1) Give names to the elements of the list

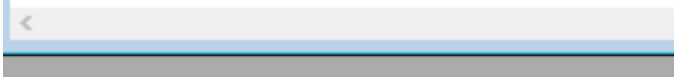
First we create the my_list

```
> my_list<-list(5,c(160,180,175),matrix(c(1,5,9,2,6,10),nrow=2,byrow=TRUE),data.frame(
+ height=c(160,180,175),
+  weight=c(52,96,60),
+ age=c(18,43,29),
+ c.eyes=c("green","blue","blue")))
> my_list
[[1]]
[1] 5

[[2]]
[1] 160 180 175

[[3]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10

[[4]]
  height weight age c.eyes
1    160     52  18  green
2    180     96  43   blue
3    175     60  29   blue

>
```

2) Extract the second element of the list (result in vector, result in list)

Result in vector: using unlist function

```
> ?unlist
> result_vector<-unlist(my_list[2])
> result_vector
[1] 160 180 175
> is.vector(result_vector)
[1] TRUE
>
```



Result in list:

```
> my_list[2]
[[1]]
[1] 160 180 175

> is.list(my_list[2])
[1] TRUE
> 
```

3) Extract the first and third elements of the list

```
> my_list[c(1,3)]
[[1]]
[1] 5

[[2]]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
```

4) Extract the third element of the second column of the fourth compartment

```
> my_list[[4]][[3,2]]
[1] 60
> 
```