R Lab 2

Bibek Sapkota

Variables

task 1: Created variables x and y . Assign the product of x and y to z and Print z.

```
x <- 2+5
y <- 6
z <- x*y
z
```

```
## [1] 42
```

task 2: Created variables a and b and divide a/b then assign the result to c and Print c.

```
a <- 10-7
b <- 4
c <- a/b
c
```

```
## [1] 0.75
```

Functions

task 1: Created the variable x and Assign the function (c),“c is a one dimensional array” to x, and print variable x.

```
x = c(100,200,300)
x
```

```
## [1] 100 200 300
```

Data Types in R

task 1: Created the variable x and checked the class of x.

```
x =5
class(x)
```

```
## [1] "numeric"
```

task 2:Checked the variable x is integer.

```
is.integer(x)
```

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

Vectors, Matrices, Arrays, Lists and DataFrames

task 1: Created the variable xx and Assign matrix with dimensions with 3rows and 2cols.

```
xx = matrix(1:6, nrow=3, ncol =2)
xx
```

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

task 2: Print the class of variable xx.

```
class(xx)
```

```
## [1] "matrix" "array"
```

task 3:Checked the variable xx is vector or not.

```
is.vector(xx)
```

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

task 4:checked the variable xx is matrix or not.

```
is.matrix(xx)
```

```
## [1] TRUE
```

task 5: Checked the length of variable xx.

```
length(xx)
```

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

task 5:Checked the dimension of the variable xx.

```
dim(xx)
```

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

Datasets for apply family

task 1:Imported the dataset of mtcars. Print the head of mtcars which will print 1st 6rows.

```
data("mtcars")
head(mtcars)
```

```
##                   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
```

Apply function

task 1: The function max is applied to calculated row wise maximum values.

```
apply (t(beaver1),1,max)
```

```
##     day    time    temp   activ
##  347.00 2350.00   37.53    1.00
```

task 2: The function mean is applied for each column and mean is calculated.

```
apply(mtcars,2,mean)
```

```
##        mpg        cyl       disp         hp       drat         wt       qsec
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250  17.848750
##         vs         am       gear       carb
##   0.437500   0.406250   3.687500   2.812500
```

task 3: Calculate the remainder when each element of the columns in the mtcars dataset is divided by 10 and display the first 6 rows

```
head(apply(mtcars,2,function(x) x%%10))
```

```
##                   mpg cyl disp hp drat    wt qsec vs am gear carb
## Mazda RX4         1.0   6    0  0 3.90 2.620 6.46  0  1    4    4
## Mazda RX4 Wag     1.0   6    0  0 3.90 2.875 7.02  0  1    4    4
## Datsun 710        2.8   4    8  3 3.85 2.320 8.61  1  1    4    1
## Hornet 4 Drive    1.4   6    8  0 3.08 3.215 9.44  1  0    3    1
## Hornet Sportabout 8.7   8    0  5 3.15 3.440 7.02  0  0    3    2
## Valiant           8.1   6    5  5 2.76 3.460 0.22  1  0    3    1
```

lapply function

task 1: Created the one-dimensional vector and assign to movies, then apply movies to lower function using lapply and checked the structure of each elements and print it.

```
movies <- c("SPYDERMAN","BATMAN","VERTIGO","CHINATOWN")
movies_lower <-lapply(movies, tolower)
str(movies_lower)
```

```
## List of 4
##  $ : chr "spyderman"
##  $ : chr "batman"
##  $ : chr "vertigo"
##  $ : chr "chinatown"
```

task 2: Converted list into vector using unlist and then Display it.

```
movies_lower <-unlist(lapply(movies,tolower))
str(movies_lower)
```

```
##  chr [1:4] "spyderman" "batman" "vertigo" "chinatown"
```

sapply function

task 1: Created a dataset cars that is assign to dt then calculated the minimum value of each column in the cars dataset using both lapply (resulting in a list) and sapply (resulting in a simplified vector), and then displays the list of minimum values.

```
dt <- cars
lmn_cars <- lapply(dt, min)
smn_cars <- sapply(dt, min)
lmn_cars
```

```
## $speed
## [1] 4
##
## $dist
## [1] 2
```

task 2: Displays the list of minimum values that has used sapply .

```
smn_cars
```

```
## speed  dist
##     4     2
```

task 3: The code calculates the maximum value of each column in the cars dataset using both 'lapply' (resulting in a list) and 'sapply' (resulting in a simplified vector), and then displays the list of maximum values.

```
lmxcars <- lapply(dt, max)
smxcars <- sapply(dt, max)
lmxcars
```

```
## $speed
## [1] 25
##
## $dist
## [1] 120
```

```
smxcars
```

```
## speed  dist
##    25   120
```

task 4:The 'avg' that calculates the average of the minimum and maximum values in a vector(x) and applies this function to each column of the dataset using 'sapply', and then displays the resulting vector of averages.

```
avg <- function(x) {
( min(x) + max(x) ) / 2}
fcars <- sapply(dt, avg)
fcars
```

```
## speed  dist
##  14.5  61.0
```

Saving and Loading Data

task 1:Creates a data frame named 'EU' containing columns for European country names and their corresponding populations and Displaying it.

```
EUCountryNames = c("United Kingdom", "Germany","France","Italy")
EUPopulation = c(63843856, 82562004,64982894,61142221)
EU = data.frame(EUCountryNames,EUPopulation)
EU
```

```
##   EUCountryNames EUPopulation
## 1 United Kingdom     63843856
## 2        Germany     82562004
## 3         France     64982894
## 4          Italy     61142221
```

task 2: Store the data on disk in the native R format RData.

```
save(EU,file = "EUInfo.RData")
```

task 3:It shows absolute filepath representing the current working directory of the R process

```r
getwd()
```

```
## [1] "C:/Users/sapko/OneDrive/Documents/(3rd sem) Data_Science/Introduction to data science/Week -2"
```

task 4:Removed the created data frame and then load the one we have from the disk.

```r
rm(EU)
load("EUInfo.RData")
EU
```

```
##   EUCountryNames EUPopulation
## 1 United Kingdom     63843856
## 2        Germany     82562004
## 3         France     64982894
## 4          Italy     61142221
```

task 5:It code saves the dataframe EU to a CSV file named "EUInfo.csv".

```r
write.csv(EU,"EUInfo.csv")
```

task 6:It reads the CSV file "EUInfo.csv" into a new data frame named 'EU2' and then displays its contents.

```r
EU2 = read.csv("EUInfo.csv")
EU2
```

```
##   X EUCountryNames EUPopulation
## 1 1 United Kingdom     63843856
## 2 2        Germany     82562004
## 3 3         France     64982894
## 4 4          Italy     61142221
```

task 7:Installing the packages "xlsx"

```r
# install.packages("xlsx")
```

task 8:Load the package "xlsx"

```r
library(xlsx)
```

task 9:It saves the dataframe EU to an Excel file named "EUInfoNew.xlsx" with the sheet name "Sheet1" using the write.xlsx() function from the "xlsx" package.

```r
write.xlsx(EU,"EUInfoNew.xlsx",sheetName="Sheet1")
```

task 10:It reads the Excel file "EUInfoNew.xlsx" from the sheet named "Sheet1" into a new data frame named EU3 and then displays its contents.

```
EU3 = read.xlsx("EUInfoNew.xlsx",sheetName="Sheet1")
EU3
```

```
##   NA. EUCountryNames EUPopulation
## 1   1 United Kingdom     63843856
## 2   2        Germany     82562004
## 3   3         France     64982894
## 4   4          Italy     61142221
```

Tutorial - 2

task 1: Get random 10 values from normal distribution

```
x <- rnorm(10)
x
```

```
##  [1] -0.9375672911  1.2323099533  0.7563334662 -0.1849756888  0.0263467944
##  [6]  0.4945899405 -0.4992627867 -0.1465388169  0.5910533193  0.0004595711
```

task 2: Display value x

```
print(x)
```

```
##  [1] -0.9375672911  1.2323099533  0.7563334662 -0.1849756888  0.0263467944
##  [6]  0.4945899405 -0.4992627867 -0.1465388169  0.5910533193  0.0004595711
```

task 3:list objects from the session

```
ls()
```

```
##  [1] "a"              "avg"            "b"              "c"
##  [5] "dt"             "EU"             "EU2"            "EU3"
##  [9] "EUCountryNames" "EUPopulation"   "fcars"          "lmn_cars"
## [13] "lmxcars"        "movies"         "movies_lower"   "mtcars"
## [17] "smn_cars"       "smxcars"        "x"              "xx"
## [21] "y"              "z"
```

task 4:Remove Object from the session

```
rm(x)
```

task 5:Get 10 values from 1 to 10 and Display it.

```
a <- c(1:10)
print(a)
```

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

task 6:Get 10 values from 1 to 10 and Display it.

```r
b <-1:10
print(b)
```

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

task 7: Create vector using : operator

```r
x <- c(1, 5, 4, 9, 0)
typeof(x)
```

```
## [1] "double"
```

```r
length(x)
```

```
## [1] 5
```

```r
x <- c(1, 5.4, TRUE, "hello")
x
```

```
## [1] "1"     "5.4"   "TRUE"  "hello"
```

```r
typeof(x)
```

```
## [1] "character"
```

task 8:Create vector using seq() function. specify step size and length of the vector

```r
seq(1, 3, by=0.2)
```

```
##  [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
```

```r
seq(1, 5, length.out=4)
```

```
## [1] 1.000000 2.333333 3.666667 5.000000
```

Accessing an element of vector

task 1:Accessing 3rd element then accessing 2nd and 4th element and accessing all but not 1st element

```r
x <- 1:7

x[3]
```

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

```r
x[c(2, 4)]
```

```
## [1] 2 4
```

```r
x[-1]
```

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

```r
# cannot mix positive and negative integers
# x[c(2, -4)]
```

Using logical vector as index

task 1: filtering vectors based on conditions. condition will repeat if there is less conditions and more variable

```r
x <- 1:5
x[c(TRUE, FALSE, FALSE, TRUE)]
```

```
## [1] 1 4 5
```

```r
x[x < 0]
```

```
## integer(0)
```

```r
x[x > 0]
```

```
## [1] 1 2 3 4 5
```

Using Character vector as index

```r
x <- c("first"=3, "second"=0, "third"=9)
names(x)
```

```
## [1] "first"  "second" "third"
```

```r
x["second"]
```

```
## second
##      0
```

Create

task 1:Create a list of strings

```r
list_size <- list("small", "medium", "big")
print(list_size)
```

```
## [[1]]
## [1] "small"
##
## [[2]]
## [1] "medium"
##
## [[3]]
## [1] "big"
```

task 2:Create x and y vector

```r
x <- as.numeric(1:30)
y <- x ^ 3
```

task 3:Create a dataframe using x and y vector and then viewing a structure of dataframe.

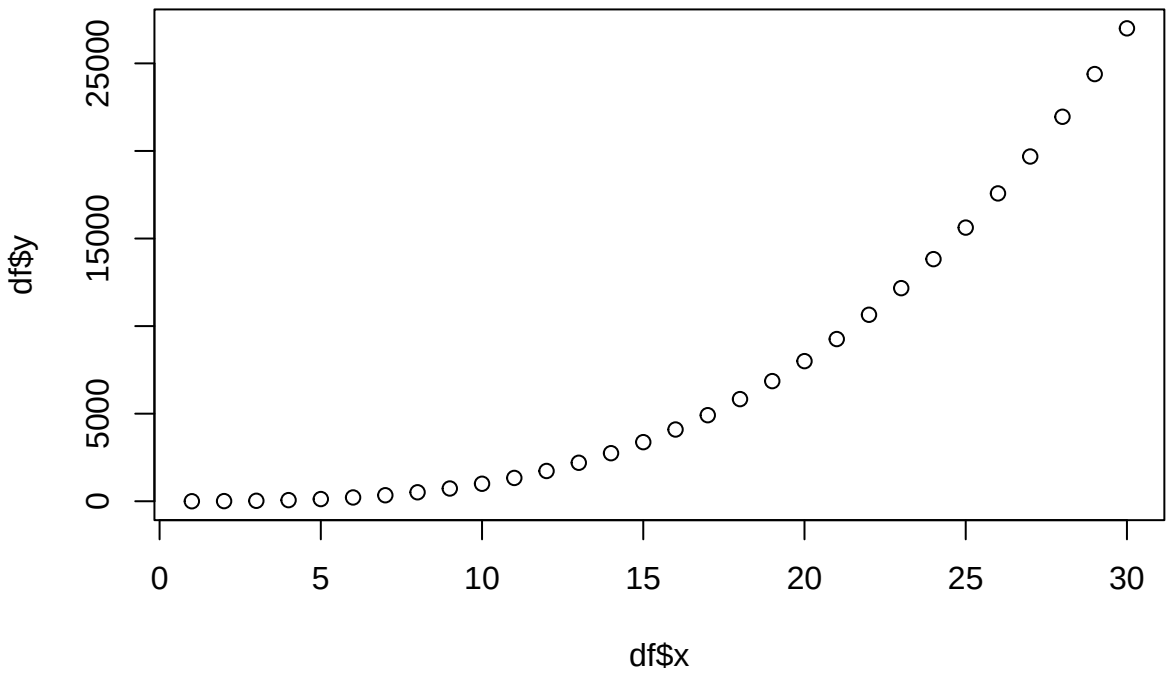
```r
df <- data.frame(x, y)

str(df)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ x: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ y: num  1 8 27 64 125 216 343 512 729 1000 ...
```

Create plot of x and y variables in R Studio and interpret it carefully

task 1:create a plot using R base package

```r
plot(df$x, df$y)
```

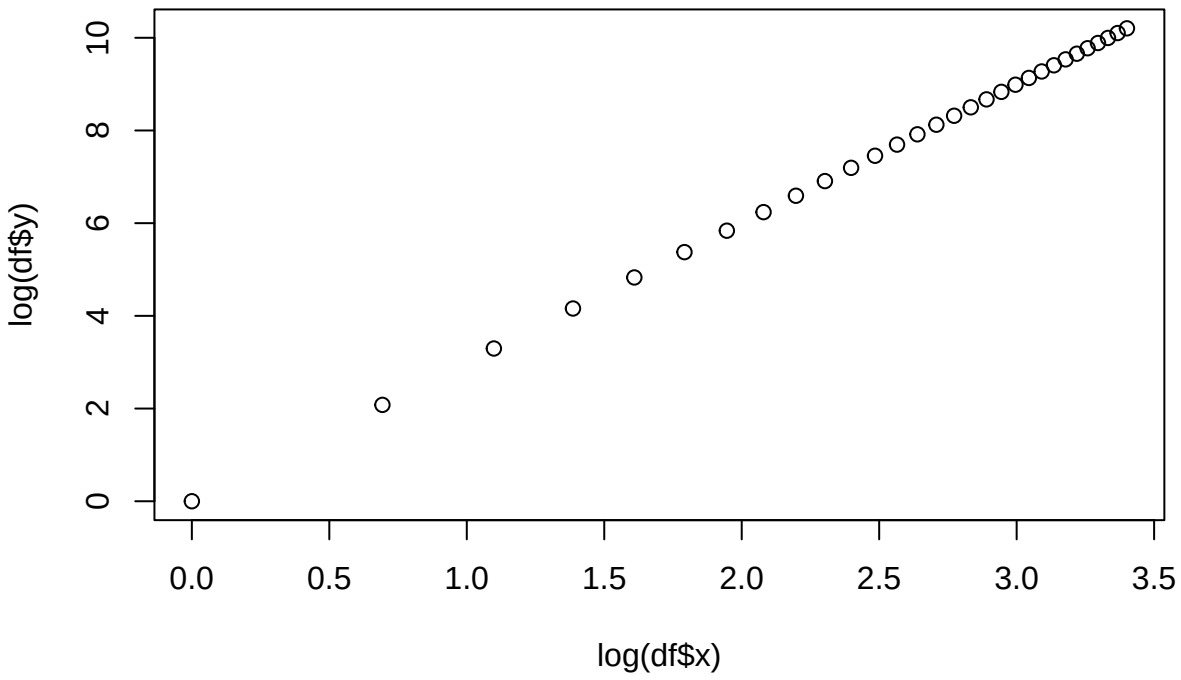



```
corr <- cor.test(x = df$x, y = df$y, method="spearman")
corr
```

```
##
##  Spearman's rank correlation rho
##
## data:  df$x and df$y
## S = 0, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
##   1
```

task 2:Use log transformation to the x and y variables.

```
plot(log(df$x), log(df$y))
```




```
corr <- cor.test(x = log(df$x), y = log(df$y), method="pearson")
corr
```

```
##
##  Pearson's product-moment correlation
##
## data:  log(df$x) and log(df$y)
## t = Inf, df = 28, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  1 1
## sample estimates:
## cor
##   1
```

R Matrix

task 1: creating the matrix with 3rows and 3 columns

```
matrix(1:9, nrow = 3, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
```

```
## [2,]    2    5    8
## [3,]    3    6    9
```

```
matrix(1:9, nrow = 3)
```

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

task 2:fill matrix row-wise

```
matrix(1:9, nrow=3, byrow=TRUE)
```

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

task 3:Create matrix and then renaming its columns in X,Y,Z and Rows in A,B,C

```
x <- matrix(1:9, nrow = 3, dimnames = list(c("X","Y","Z"), c("A","B","C")))
x
```

```
##   A B C
## X 1 4 7
## Y 2 5 8
## Z 3 6 9
```

task 4:access column names and rownames

```
colnames(x)
```

```
## [1] "A" "B" "C"
```

```
rownames(x)
```

```
## [1] "X" "Y" "Z"
```

task 5:change column and row names

```
colnames(x) <- c("C1","C2","C3")
rownames(x) <- c("R1","R2","R3")
x
```

```
##    C1 C2 C3
## R1  1  4  7
## R2  2  5  8
## R3  3  6  9
```

task 6:‘cbind’ 1st plot column wise and ‘rbind’ plot row wies

```r
cbind(c(1,2,3),c(4,5,6))
```

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

```r
rbind(c(1,2,3),c(4,5,6))
```

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

task 7:Use dim to create matrix

```r
x <- c(1,2,3,4,5,6)
dim(x) <- c(2,3)
x
```

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

```r
x <- c(1,2,3,4,5,6)
class(x)
```

```
## [1] "numeric"
```

```r
dim(x) <- c(2,3)
x
```

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

```r
class(x)
```

```
## [1] "matrix" "array"
```

Access Elements of Matrix

task 1:Create a matrix with dimension 3x3

```r
x <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)
x
```

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

task 2:select rows 1 & 2 and columns 2 & 3

```
x[c(1,2),c(2,3)]
```

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

task 3:leaving column field blank will select entire columns

```
x[c(3,2),]
```

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

task 4:leaving row as well as column field blank will select entire matrix

```
x[,]
```

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

task 5:select all rows except first

```
x[-1,]
```

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

Lists

```
x <- list("a" = 2.5, "b" = TRUE, "c" = 1:3)
x
```

```
## $a
## [1] 2.5
##
## $b
## [1] TRUE
##
## $c
## [1] 1 2 3
```

```r
str(x)
```

```
## List of 3
##  $ a: num 2.5
##  $ b: logi TRUE
##  $ c: int [1:3] 1 2 3
```

Accessing the list components

task 1:Create a list and assign to x and Display it

```r
x <- list(name = "John", age = 19, speaks = c("English", "French"))
x
```

```
## $name
## [1] "John"
##
## $age
## [1] 19
##
## $speaks
## [1] "English" "French"
```

task 2:access elements by name

```r
x$name
```

```
## [1] "John"
```

```r
x$age
```

```
## [1] 19
```

```r
x$speaks
```

```
## [1] "English" "French"
```

task 3:access elements by integer index

```r
x[c(1, 2)]
```

```
## $name
## [1] "John"
##
## $age
## [1] 19
```

```r
x[-2]
```

```
## $name
## [1] "John"
##
## $speaks
## [1] "English" "French"
```

task 4:access elements by logical index

```r
x[c(TRUE, FALSE, FALSE)]
```

```
## $name
## [1] "John"
```

task 5:access elements by character index

```r
x[c("age", "speaks")]
```

```
## $age
## [1] 19
##
## $speaks
## [1] "English" "French"
```

task 7:Create a list

```r
x <- list(name = "John", age = 19, speaks = c("English", "French"))
x
```

```
## $name
## [1] "John"
##
## $age
## [1] 19
##
## $speaks
## [1] "English" "French"
```

task 8:access element by name using single bracket []

```r
x["age"]
```

```
## $age
## [1] 19
```

task 9:check the type of the result (single bracket returns a list)

```r
typeof(x["age"])
```

```
## [1] "list"
```

task 10:access element by name using double bracket [[]]

```r
x[["age"]]
```

```
## [1] 19
```

task 11:check the type of the result (double bracket returns the content)

```r
typeof(x[["age"]])
```

```
## [1] "double"
```

Add a component in list

task 1:Create a list and display it

```r
x <- list(name = "Clair", age = 19, speaks = c("English", "French"))
x
```

```
## $name
## [1] "Clair"
##
## $age
## [1] 19
##
## $speaks
## [1] "English" "French"
```

task 2:assign a new element to the list using double brackets [[]] and print the updated list

```r
x[["married"]] <- FALSE
x
```

```
## $name
## [1] "Clair"
##
## $age
## [1] 19
##
## $speaks
## [1] "English" "French"
##
## $married
## [1] FALSE
```

Delete a component in list

task 1:Create a list and display it

```
x <- list(name = "Clair", age = 19, speaks = c("English", "French"))
```

task 2:remove an element from the list using double brackets [[]]

```
x[["age"]] <- NULL
x
```

```
## $name
## [1] "Clair"
##
## $speaks
## [1] "English" "French"
```

task 3:print the structure of the updated list

```
str(x)
```

```
## List of 2
##  $ name  : chr "Clair"
##  $ speaks: chr [1:2] "English" "French"
```

task 4:remove an element from the list using $ notation

```
x$married <- NULL
x
```

```
## $name
## [1] "Clair"
##
## $speaks
## [1] "English" "French"
```

task 5:print the structure of the updated list

```
str(x)
```

```
## List of 2
##  $ name  : chr "Clair"
##  $ speaks: chr [1:2] "English" "French"
```