# #2. Data Structures

Four common object types that store data are:

1. **Scalars**: store a single numeric value.
2. **Strings**: store a set of one or more characters.
3. **Vectors**: store several scalar or string elements.
4. **Data Frames**. Store several vectors (meaning that they contain several rows and columns).

Exercises

Arrays

**Exercise 1**  
Create an array (3 dimensional) of 24 elements using the  dim()  function.

vec <- sample(1:5,24,replace = TRUE)

dim(vec) <- c(3,2,4)

dim(vec)

**Exercise 2**  
Create an array (3 dimensional) of 24 elements using the  array()  function.

ar <- array(1:5,c(2,3,4))

**Exercise 3**  
Assign some dimnames of your choice to the array using the  dimnames()  function.

ar <- array(data = 1:24,

dim = c(2, 3, 4))

dimnames(ar)[[1]] <- c("a","b")

dimnames(ar)[[2]] <- c("x","y","z")

dimnames(ar)[[3]] <- c("o","n","u","r")

ar

**Exercise 4**  
Assign some dimnames of your choice to the array using the arguments of the  array()  function.

ar <- array(data = 1:24,

dim = c(2, 3, 4),

dimnames = list(c("a","b"),c("x","y","z"),c("o","n","u","r")))

ar

**Exercise 5**  
Instead of column-major array, make a row-major array (transpose).

aperm(ar, perm = NULL, resize = TRUE)

**Exercise 6**  
For this exercise, and all that follow, download [this file](http://www.r-exercises.com/wp-content/uploads/2015/10/ex.csv), and read it into R using the  read.csv()  function, e.g.:  
temp  
Copy the column named  N  into a new variable  arr.

setwd("/Users/Onurhan/Documents/R Work/R-exercises Files/")

dataset <- read.csv("exercise426.csv")*.*

arr <- dataset$N

**Exercise 7**  
Set dimensions of this variable and convert it into a 3 \* 2 \* 4 array. Add dimnames.

arr <- array(arr,

dim = c(3,2,4),

dimnames = list(c("a","b","c"),c("x","y"),c("o","n","u","r")))

arr

**Exercise 8**  
Print the whole array on the screen.

print(arr)

**Exercise 9**  
Print only elements of height 2, assuming the first dimension represents rows, the second columns and the third heigth.

arr[,,2]

**Exercise 10**  
Print elements of height 1 and columns 3 and 1.

arr[,-2,1]

**Exercise 11**  
Print element of height 2, column 4 and row 2.

arr[2,4,2]

**Exercise 12**  
Repeat the exercises 9-11, but instead of using numbers to reference row, column and height, use  dimnames.

arr[,,"n"]

arr[,"x","o"]

arr["b",,"n"]

Solutions to these exercises

# Matrix

**Exercise 1**  
Create three vectors  x,y,z  with integers and each vector has 3 elements. Combine the three vectors to become a 3×3 matrix  A  where each column represents a vector. Change the row names to  a,b,c.  
Think: How about each row represents a vector, can you modify your code to implement it?

x <- sample(1:10,3,replace = TRUE)

y <- sample(1:10,3,replace = TRUE)

z <- sample(1:10,3,replace = TRUE)

A <- cbind(x,y,z)

A

rownames(A) <- c("a","b","c")

A

**Exercise 2**  
Please check your result from Exercise 1, using  is.matrix(A). It should return  TRUE, if your answer is correct. Otherwise, please correct your answer. Hint: Note that  is.matrix()  will return  FALSE  on a non-matrix type of input. Eg: a vector and so on.

is.matrix(A)

**Exercise 3**  
Create a vector with 12 integers. Convert the vector to a 4\*3 matrix  B  using  matrix(). Please change the column names to  x, y, z  and row names to  a, b, c, d.  
The argument  byrow  in  matrix()  is set to be  FALSE  by default. Please change it to  TRUE  and print  B  to see the differences.

b <- sample(1:50,12,replace = FALSE)

B <- matrix(b, nrow = 4, ncol = 3)

rownames(B) <- c("a","b","c","d")

colnames(B) <- c("x","y","z")

B

B <- matrix(b, nrow = 4, ncol = 3, byrow = TRUE)

B

**Exercise 4**  
Please obtain the transpose matrix of  B  named  tB .

tB <- t(B)

tB

**Exercise 5**  
Now  tB  is a 3×4 matrix. By the rule of matrix multiplication in algebra, can we perform  tB\*tB  in R language? (Is a 3×4 matrix multiplied by a 3×4 allowed?) What result would we get?

tB\*tB

**Exercise 6**  
As we can see from Exercise 5, we were expecting that  tB\*tB  would not be allowed because it disobeys the algebra rules. But it actually went through the computation in R. However, as we check the output result , we notice the multiplication with a single  \*  operator is performing the componentwise multiplication. It is not the conventional matrix multiplication. How to perform the conventional matrix multiplication in R? Can you compute matrix  A  multiplies  tB ?

A%\*%tB

**Exercise 7**  
If we convert  A  to a  data.frame  type instead of a  matrix , can we still compute a conventional matrix multiplication for matrix  A  multiplies matrix  A ? Is there any way we could still perform the matrix multiplication for two  data.frame  type variables? (Assuming proper dimension)

A <- data.frame(A)

as.matrix(A) %\*% as.matrix(A)

**Exercise 8**  
Extract a sub-matrix from  B  named  subB . It should be a 3×3 matrix which includes the last three rows of matrix  B  and their corresponding columns.

subB<-B[2:dim(B)[1],1:3]

subB

**Exercise 9**  
Compute  3\*A ,  A+subB ,  A-subB . Can we compute  A+B? Why?

3\*A

A + subB

A – subB

A+B

**Exercise 10**  
Generate a n \* n matrix (square matrix)  A1  with proper number of random numbers, then generate another n \* m matrix  A2.  
If we have  A1\*M=A2  (Here \* represents the conventional multiplication), please solve for  M.  
Hint: use the  runif()  and  solve()  functions. E.g.,  runif(9)  should give you 9 random numbers.

A1 <- matrix(runif(9),3,3)

A2 <- matrix(runif(12),3,4)

M <- solve(A1,A2)

M

**Dataframes**

. **Exercise 1**  
Create the following data frame, afterwards invert Sex for all individuals.

Name <- c("Alex", "Lilly", "Mark", "Oliver", "Martha", "Lucas", "Caroline")

Age <- c(25, 31, 23, 52, 76, 49, 26)

Height <- c(177, 163, 190, 179, 163, 183, 164)

Weight <- c(57, 69, 83, 75, 70, 83, 53)

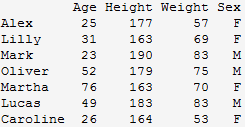
Sex <- as.factor(c("F", "F", "M", "M", "F", "M", "F"))

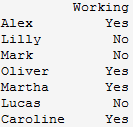
df <- data.frame (row.names = Name, Age, Height, Weight, Sex)

levels(df$Sex)

levels(df$Sex) <- c("M","F")

df

[](http://www.r-exercises.com/wp-content/uploads/2016/01/table1.png)

**Exercise 2**  
Create this data frame (make sure you import the variable Working as character and not factor).  
[](http://www.r-exercises.com/wp-content/uploads/2016/01/table2.png)

Working <- c("Yes","No","No","Yes","Yes","No","Yes")

df2 <- data.frame(row.names = Name,Working) *#Name has been already defined in exercise 1.*

df2

Add this data frame column-wise to the previous one.  
a) How many rows and columns does the new data frame have?

df <- cbind(df,df2)

dim(df)

b) What class of data is in each column?

str(df)

**Exercise 3**  
Check what class of data is the (built-in data set) state.center and convert it to data frame.

class(state.center)

data.state <- as.data.frame(state.center)

**Exercise 4**  
Create a simple data frame from 3 vectors. Order the entire data frame by the first column.

a <- (rnorm(10))

b <- letters[4:13]

c <- c("yes","no","no","no","no","yes","no","yes","yes","no")

df3 <- data.frame(a,b,c)

df3[with (df3, order(a)),]

**Exercise 5**  
Create a data frame from a matrix of your choice, change the row names so every row says id\_i (where i is the row number) and change the column names to variable\_i (where i is the column number). I.e., for column 1 it will say variable\_1, and for row 2 will say id\_2 and so on.

matrix.data <- matrix(1:40, nrow = 10, ncol = 4)

df <- as.data.frame(matrix.data)

colnames(df) <- paste("variable\_", 1:ncol(df))

rownames(df) <- paste("id\_", 1:nrow(df))

df

**Exercise 6**  
For this exercise, we’ll use the (built-in) dataset VADeaths.

1. Make sure the object is a data frame, if not change it to a data frame.

class(VADeaths)

b) Create a new variable, named Total, which is the sum of each row.

df <- data.frame(VADeaths)

df$Total <- rowSums(df)

df <- df[,c(5,1,2,3,4)]

df

c) Change the order of the columns so total is the first variable.

**Exercise 7**  
For this exercise we’ll use the (built-in) dataset state.x77.

1. Make sure the object is a data frame, if not change it to a data frame.

class(state.x77)

df2 <- data.frame(state.x77)

b) Find out how many states have an income of less than 4300.

nrow(df2[df2$Income < 4300,])

c) Find out which is the state with the highest income.

row.names(df2[which.max(df2$Income),])

**Exercise 8**  
With the dataset swiss, create a data frame of only the rows 1, 2, 3, 10, 11, 12 and 13, and only the variables Examination, Education and Infant.Mortality.

df3 <- data.frame(swiss[c(1,2,3,10,11,12,13),c("Examination", "Education", "Infant.Mortality")])

a) The infant mortality of Sarine is wrong, it should be a NA, change it.

df3$Infant.Mortality[4] <- NA

b) Create a row that will be the total sum of the column, name it Total.

Total <- colSums(df3, na.rm = TRUE)

df4 <- rbind(df3,Total)

rownames(df4) = c(rownames(df3),"Total")

df4

c) Create a new variable that will be the proportion of Examination (Examination / Total)

Prop <- df4$Examination / df4["Total","Examination"]

Prop

df4 <- cbind(df4,Prop)

df4

**Exercise 9**  
Create a data frame with the datasets state.abb, state.area, state.division, state.name, state.region. The row names should be the names of the states.

df <- data.frame(state.abb, state.area, state.division, state.region, row.names = state.name)

1. Rename the column names so only the first 3 letters after the full stop appear (e.g. States.abb will be abb).

colnames(df) <- substr(colnames(df), 7, 9)

**Exercise 10**  
Add the previous data frame column-wise to state.x77

new.df <- cbind(state.x77,df)  
a) Remove the variable div.

new.df$div <- NULL

b) Also remove the variables Life Exp, HS Grad, Frost, abb, and are.

new.df <- subset(new.df, ,-c(4, 6, 7, 9, 10))

c) Add a variable to the data frame which should categorize the level of illiteracy:  
[0,1) is low, [1,2) is some, [2, inf) is high.

new.df$Illiteracy.Levels <- ifelse(new.df$Illiteracy >= 0 & new.df$Illiteracy < 1, "Low",

ifelse(new.df$Illiteracy >= 1 & new.df$Illiteracy < 2, "Some",

"High"))

d) Find out which state from the west, with low illiteracy, has the highest income, and what that income is.

x <- subset(new.df, reg == "West" & Illiteracy.Levels == "Low")

row.names(x[which.max(x$Income),])

**Student Self Work Exercises**

1. Create four scalar objects. Each should contain the age of a different family member. You can name these objects whatever you want. Then, using R:
   1. Find the age difference between the oldest and youngest family member.
   2. Find the total age of your family members.
   3. Find the mean age.
2. Create a vector that contains the four objects you created in Exercise 1.
   1. Find the mean of the vector. (Of course, you should get the same answer as in 1.c.)
   2. How old will each of your family members be in 10 years? Hint: If your vector is called v, then you can add a number **c** to *each element* of that vector using v + c.
3. Create a data frame that contains three variables: the name of each of your family members, their age, and their gender.
   1. Use R to find the class of each of these variables.
   2. (Conceptual) What type of variable (nominal, ordinal, continuous) is each of these variables? (No need to use R to answer this.)
   3. Add a variable to the data frame that indicates what year each of your family members will turn 100 years old. What is the mean of this variable?
   4. What is the mean age of your male family members? Of your female family members? Hint: use brackets to get particular elements of the data frame, then find the mean.

References

1. Richard Cotton. Learning R. O’Reilly. 2013.

2. ROBERT I. KABACOFF. R in Action. Data analysis and graphics with R. MANNING. 2011.