Data Structures in R- Exercise

Pratik Agrawal (804861)

10/12/2019

Data structures: Vectors

1.Create vectors name, age and hours using the function c().

```
name = c("Steve","Sophia","Oliver","Harry","Peter","Olivia","Marc")
print(name)
```

```
## [1] "Steve"  "Sophia" "Oliver" "Harry"  "Peter"  "Olivia" "Marc"
```

```
age = c(15,16,21,18,19,20,21)
print(age)
```

```
## [1] 15 16 21 18 19 20 21
```

```
hours = c(6.4, 5.3, 7.2, 5.2 ,8.5, 7.3, 6.0)
print(hours)
```

```
## [1] 6.4 5.3 7.2 5.2 8.5 7.3 6.0
```

2. Determine the name of the second user.

```
name[2]
```

```
## [1] "Sophia"
```

3. Determine the names of all users except the second one.

```
print(name[-2])
```

```
## [1] "Steve"  "Oliver" "Harry"  "Peter"  "Olivia" "Marc"
```

4. Determine the names of the second, third and fourth users.

```
print(name[2:4])
```

```
## [1] "Sophia" "Oliver" "Harry"
```

5. Determine the names of the second and fourth users.

```
print(name[c(2,4)])
```

```
## [1] "Sophia" "Harry"
```

6. Determine the names of all users except the second and fourth ones.

```
print(name[-c(2,4)])
```

```
## [1] "Steve"  "Oliver" "Peter"  "Olivia" "Marc"
```

7. Determine the number of users in your dataset.

```
length(name)
```

```
## [1] 7
```

8. Determine where users are aged older than 19.

```
as.logical(age > 19)
```

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

9. Determine the indices of users aged older than 19.

```
#match(age[age > 19],age) #not working matches only first occurence
which(age %in% age[age > 19])
```

```
## [1] 3 6 7
```

10. Determine how old those users are.

```
age[age > 19]
```

```
## [1] 21 20 21
```

11. Combine vectors age and hours into vector ageHours using the function c(). Check the result.

```
ageHours = c(age,hours)
print(ageHours)
```

```
##  [1] 15.0 16.0 21.0 18.0 19.0 20.0 21.0  6.4  5.3  7.2  5.2  8.5  7.3  6.0
```

Data structures: Matrices

12. Create matrix ageHours.m from vector ageHours. Check the result. Hint: The number of rows should be equal to the number of users in the data set.

```
ageHours.m = matrix(ageHours,nrow=7,ncol=2)
print(ageHours.m)
```

```
##      [,1] [,2]
## [1,]   15  6.4
## [2,]   16  5.3
## [3,]   21  7.2
## [4,]   18  5.2
## [5,]   19  8.5
## [6,]   20  7.3
## [7,]   21  6.0
```

13. Determine the age of the first user in ageHours.m.

```
ageHours.m[1,1]
```

```
## [1] 15
```

14. Determine the age of all users in ageHours.m.

```
ageHours.m[,1]
```

```
## [1] 15 16 21 18 19 20 21
```

15. Determine the age of all users in ageHours.m except the first one.

```
ageHours.m[2:7,1]
```

```
## [1] 16 21 18 19 20 21
```

16. Determine the age of all users in ageHours.m except the first and third ones.

```r
ageHours.m[-c(1,3),1]
```

```
## [1] 16 18 19 20 21
```

17. Recreate matrix ageHours.m from vectors age and hours using the function cbind(). Check the result.

```r
ageHours.m = cbind(age,hours)
print(ageHours.m)
```

```
##      age hours
## [1,]  15   6.4
## [2,]  16   5.3
## [3,]  21   7.2
## [4,]  18   5.2
## [5,]  19   8.5
## [6,]  20   7.3
## [7,]  21   6.0
```

18. Determine the column names of ageHours.m.

```r
colnames(ageHours.m)
```

```
## [1] "age"   "hours"
```

19. Rename the column names of ageHours.m to Age and AverageHours. Check the result.

```r
colnames(ageHours.m) = c("Age","AverageHours")
print(colnames(ageHours.m))
```

```
## [1] "Age"          "AverageHours"
```

20. Rename the row names of ageHours.m using vector name. Check the result.

```r
rownames(ageHours.m) = name
print(rownames(ageHours.m))
```

```
## [1] "Steve"  "Sophia" "Oliver" "Harry"  "Peter"  "Olivia" "Marc"
```

21. Determine the age of Steve in ageHours.m using the column and row names.

```
ageHours.m["Steve","Age"]
```

```
## [1] 15
```

22. Determine the age of all users in ageHours.m using the column and row names.

```
ageHours.m[name,"Age"]
```

```
##  Steve Sophia Oliver  Harry  Peter Olivia   Marc
##     15     16     21     18     19     20     21
```

23. Determine the age of all users in ageHours.m except Steve's one using the column and row names.

```
ageHours.m[name !="Steve","Age"]
```

```
## Sophia Oliver  Harry  Peter Olivia   Marc
##     16     21     18     19     20     21
```

24. Determine the age of all users in ageHours.m except Steve's and Oliver's ones using the column and row names.

```
ageHours.m[name %in% c("Sophia","Harry","Peter","Olivia","Marc"),"Age"]
```

```
## Sophia  Harry  Peter Olivia   Marc
##     16     18     19     20     21
```

Data structures: Data frames

25. Create the data frame users from vector name, age and hours.

```
users = data.frame(name,age,hours)
print(users)
```

```
##     name age hours
## 1  Steve  15   6.4
## 2 Sophia  16   5.3
## 3 Oliver  21   7.2
## 4  Harry  18   5.2
## 5  Peter  19   8.5
## 6 Olivia  20   7.3
## 7   Marc  21   6.0
```

26. Determine the column names of users.

```
colnames(users)
```

```
## [1] "name"  "age"   "hours"
```

27. Determine the names of users in at least 2 different ways

```
users[["name"]]
```

```
## [1] Steve  Sophia Oliver Harry  Peter  Olivia Marc
## Levels: Harry Marc Oliver Olivia Peter Sophia Steve
```

```
users$name
```

```
## [1] Steve  Sophia Oliver Harry  Peter  Olivia Marc
## Levels: Harry Marc Oliver Olivia Peter Sophia Steve
```

28. Recreate data frame users by embedding data directly in the program. Use name, age and hours as its column names

You are supposed to insert everything as text and then to use read.table function. Alternatively, you can use data.frame function.

```
users = read.table(file="users.txt",sep=",",col.names = c("name","age","hours"))
print(users)
```

```
##     name age hours
## 1  Steve  15   6.4
## 2 Sophia  16   5.3
## 3 Oliver  21   7.2
## 4  Harry  18   5.2
## 5  Peter  19   8.5
## 6 Olivia  20   7.3
## 7   Marc  21   6.0
```

29. Assume, all users are now 1 year older. Update the data frame users. Check the result.

```
users$age =users$age+1
users$age
```

```
## [1] 16 17 22 19 20 21 22
```

30. Determine the average age of users in the data frame users.

```r
mean(users$age)
```

```
## [1] 19.57143
```

[1] 6.557143 ## 31. Determine the columns age and hours in the data frame users.

```r
subset(users,select = c("age","hours"))
```

```
##   age hours
## 1  16   6.4
## 2  17   5.3
## 3  22   7.2
## 4  19   5.2
## 5  20   8.5
## 6  21   7.3
## 7  22   6.0
```

32. Determine the averages for columns age and hours in the data frame users.

```r
colMeans(subset(users,select = c("age","hours")))
```

```
##       age     hours
## 19.571429  6.557143
```

33. Determine the average hours for different ages in the data frame users.

```r
aggregate(users[, 3], list(users$age), mean)
```

```
##   Group.1   x
## 1      16 6.4
## 2      17 5.3
## 3      19 5.2
## 4      20 8.5
## 5      21 7.3
## 6      22 6.6
```

34. Assume, Steve has increased his average social media consumption per day by 10 hours. Update the data frame users. Check the result.

```r
index <- users$name == "Steve"
users$hours[index] <- (users$hours[index] +10)
users$hours[index]
```

```
## [1] 16.4
```

Factors

35. If f = c(1, 2, 3, 3, 5, 3, 2, 4, NA), what are the levels of factor(f)?

```r
f = c(1, 2, 3, 3, 5, 3, 2, 4, NA)
factor(f)
```

```
## [1] 1    2    3    3    5    3    2    4    <NA>
## Levels: 1 2 3 4 5
```

36. Let a <- c(11, 22, 47, 47, 11, 47, 11). If an R expression factor(a, levels=c(11, 22, 47), ordered=TRUE) is executed, what will be the 4th element in the output?

```r
a <- c(11, 22, 47, 47, 11, 47, 11)
factor(a, levels=c(11, 22, 47), ordered=TRUE)
```

```
## [1] 11 22 47 47 11 47 11
## Levels: 11 < 22 < 47
```

There won't be any 4th element in output as there are only 3 levels.

37. If z <- factor(c("p", "q", "p", "r", "q")) and levels of z are "p", "q" ,"r", write an R expression that will change the level "p" to "w" so that z is equal to: "w", "q" , "w", "r" , "q".

```r
z <- factor(c("p", "q", "p", "r", "q"))
levels(z)[levels(z)=="p"] <- "w"
z
```

```
## [1] w q w r q
## Levels: w q r
```

38. If:

s1 <- factor(sample(letters, size=5, replace=TRUE)) and

s2 <- factor(sample(letters, size=5, replace=TRUE)),

write an R expression that will concatenate s1 and s2 in a single factor with 10 elements.

```r
s1 <- factor(sample(letters, size=5, replace=TRUE))
s2 <- factor(sample(letters, size=5, replace=TRUE))
print(factor(c(as.character(s1),as.character(s2))))
```

```
##  [1] q s l w e k r n g e
## Levels: e g k l n q r s w
```

39. Consider the factor responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree")), with the following output:

[1] Agree Agree Strongly Agree Disagree Agree

Levels: Agree Disagree Strongly Agree

Later it was found that new a level "Strongly Disagree" exists. Write an R expression that will include "strongly ## disagree" as new level attribute of the factor and returns the following output:

[1] Agree Agree Strongly Agree Disagree Agree

Levels: Strongly Agree Agree Disagree Strongly Disagree **

```
responses <- factor(c("Agree", "Agree", "Strongly Agree", "Disagree", "Agree"))
 levels(responses) <- c(levels(responses), "Strongly Disagree")
 responses
```

```
## [1] Agree          Agree          Strongly Agree Disagree       Agree
## Levels: Agree Disagree Strongly Agree Strongly Disagree
```

40. Let x <- data.frame(q=c(2, 4, 6), p=c("a", "b", "c")). Write an R statement that will replace levels a, b, c with labels "fertiliser1", "fertliser2", "fertiliser3".

```
x <- data.frame(q=c(2, 4, 6), p=c("a", "b", "c"))
levels(x$p) <- c("fertiliser1", "fertliser2", "fertiliser3")
levels(x$p)
```

```
## [1] "fertiliser1" "fertliser2"  "fertiliser3"
```

41. If x <- factor(c("high", "low", "medium", "high", "high", "low", "medium")), write an R expression that will provide unique numeric values for various levels of x with the following output:

levels value 1 high 1 2 low 2 3 medium 3

```
x <- factor(c("high", "low", "medium", "high", "high", "low", "medium"))
data.frame(levels = unique(x), value = unique(as.numeric(x)), row.names = seq(1:3))
```

```
##   levels value
## 1   high     1
## 2    low     2
## 3 medium     3
```