SESSION 5:
Data Management Using R
Assignment 3

Submitted by: Munmun Ghosal
Login Id: munmun55@gmail.com
(M):+91-8007178659

Table of Contents

1. Problem Statement

A. Test whether two vectors are exactly equal (element by element)

```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))
```

B. Sort the character vector in ascending order and descending order

```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))
```

C. What is the major difference between str() and paste() show an example.

D. Introduce a separator when concatenating the strings

2. Solution

A. Test whether two vectors are exactly equal (element by element)

The R-script for the given problem is as follows:

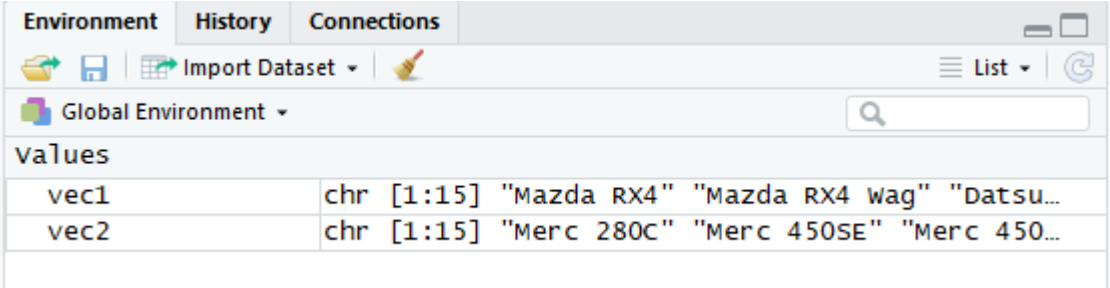
```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))

isTRUE(all.equal(vec1,vec2))        # returns true/false
identical(vec1,vec2)                # returns true/false
all.equal(vec1,vec2)                # returns number of differences
```

Explanation:

- isTRUE(all.equal(vec1,vec2)) returns TRUE if vec1 is equal to vec2;else it returns FALSE.
- identical(vec1,vec2) returns TRUE if vec1 is identical/same to vec2;else it returns FALSE.
- all.equal(vec1,vec2) returns number of differences between vec1 and vec2.

The output of the R-Script (from Console window) is given as follows:

```
Console   Terminal
~/testdir/
> vec1 = c(rownames(mtcars[1:15,]))
> vec2 = c(rownames(mtcars[11:25,]))
>
> isTRUE(all.equal(vec1,vec2)) # returns true/false
[1] FALSE
> identical(vec1,vec2)         # returns true/false
[1] FALSE
> all.equal(vec1,vec2)         # returns number of differences
[1] "15 string mismatches"
>
```



B. Sort the character vector in ascending order and descending order

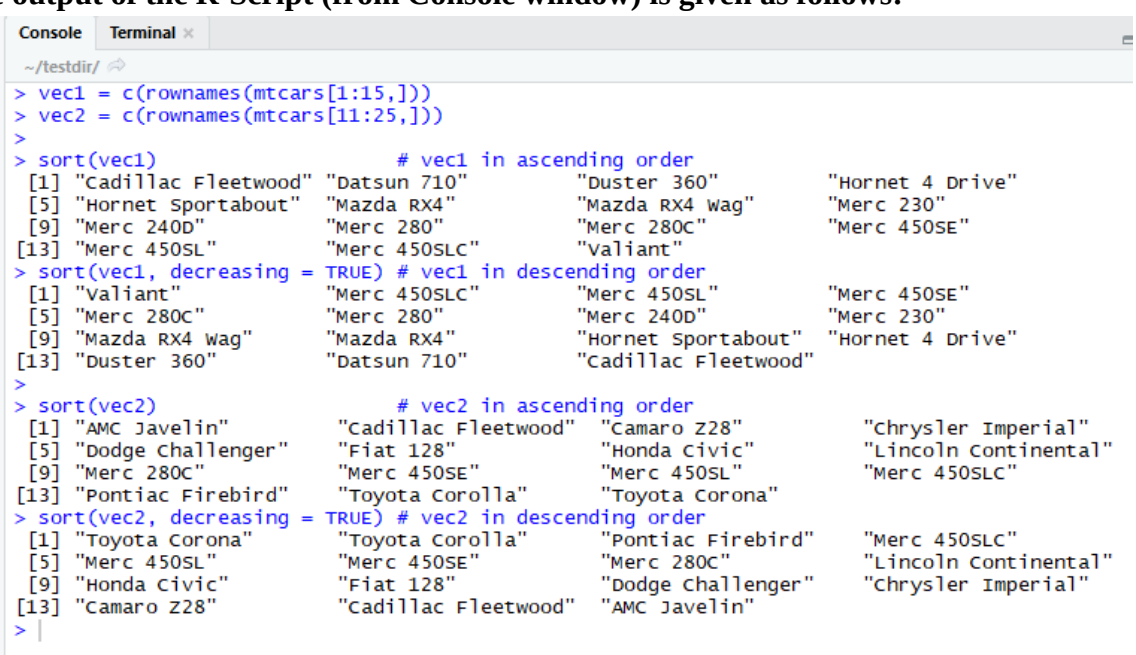
The R-script for the given problem is as follows:

```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))

sort(vec1)                # vec1 in ascending order
sort(vec1, decreasing = TRUE) # vec1 in descending order

sort(vec2)                # vec2 in ascending order
sort(vec2, decreasing = TRUE) # vec2 in descending order
```

Explanation:

- sort(vec1) function arranges the character vector vec1 in ascending order. For descending order "decreasing" parameter is set as "TRUE"
- sort(vec2) function arranges the character vector vec2 in ascending order. For descending order "decreasing" parameter is set as "TRUE"

The output of the R-Script (from Console window) is given as follows:



C. Major difference between str() and paste() show an example.

Explanation:

str() gives the class of variable, number of values and the elements whereas paste() printsor displays the actual elements .

For example:

str(mtcars$mpg) gives the class of mtcars$mpg as num, number of values as 32(1:32) and the elements as 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...

whereas **paste(mtcars$mpg)** prints the actual elements present in mtcars$mpg.

```
> str(mtcars$mpg)
 num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
> paste(mtcars$mpg)
 [1] "21"   "21"   "22.8" "21.4" "18.7" "18.1" "14.3" "24.4" "22.8" "19.2"
"17.8" "16.4"
[13] "17.3" "15.2" "10.4" "10.4" "14.7" "32.4" "30.4" "33.9" "21.5" "15.5"
"15.2" "13.3"
[25] "19.2" "27.3" "26"   "30.4" "15.8" "19.7" "15"   "21.4"
```

The R-script for the given problem is as follows:

```
str(mtcars$mpg)
paste(mtcars$mpg)
```

The output of the R-Script (from Console window) is given as follows:

```
> str(mtcars$mpg)
 num [1:32] 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
> paste(mtcars$mpg)
 [1] "21"   "21"   "22.8" "21.4" "18.7" "18.1" "14.3" "24.4" "22.8" "19.2"
"17.8" "16.4"
[13] "17.3" "15.2" "10.4" "10.4" "14.7" "32.4" "30.4" "33.9" "21.5" "15.5"
"15.2" "13.3"
[25] "19.2" "27.3" "26"   "30.4" "15.8" "19.7" "15"   "21.4"
```

D. Introduce a separator when concatenating the strings

The R-script for the given problem is as follows:

```
paste(rownames(mtcars[1,]), rownames(mtcars[2,]), sep = " ")
paste(rownames(mtcars[1,]), rownames(mtcars[4,]), sep = ",")
paste(rownames(mtcars[2,]), rownames(mtcars[1,]), sep = "--")
paste(rownames(mtcars[3,]), rownames(mtcars[10,]), sep = "$")
paste("hello","world",sep=" @ ")
paste("Assignment","5","3",sep="_")
```

Explanation:

The above R-script shows 6 examples where separators are introduced while concatenating the strings.The separators are introduced by setting the parameter “sep” as “ ” or “,” or "--" or “$” or “@” or “_” or any other separator .

For example:

paste(rownames(mtcars[1,]), rownames(mtcars[2,]), sep = " ") introduces a separator ,a single blank **" "** between the strings rownames(mtcars[1,]) and rownames(mtcars[2,].

paste(rownames(mtcars[1,]), rownames(mtcars[4,]), sep = ",") introduces a separator comma **", "** between the strings rownames(mtcars[1,]) and rownames(mtcars[4,].

paste(rownames(mtcars[2,]), rownames(mtcars[1,]), sep = "--") introduces a separator "-- " between the strings rownames(mtcars[2,]) and rownames(mtcars[1,].

paste(rownames(mtcars[3,]), rownames(mtcars[10,]), sep = "$") introduces a separator dollar "$ " between the strings rownames(mtcars[3,]) and rownames(mtcars[10,].

paste("hello","world",sep="@") introduces a separator "@" between the strings “hello” and “world”

paste("Assignment","5","3",sep="_") introduces a separator underscore "_ " between the strings "Assignment","5" and "3".

.

The output of the R-Script (from Console window) is given as follows:

```
> paste(rownames(mtcars[1,]), rownames(mtcars[2,]), sep = " ")
[1] "Mazda RX4 Mazda RX4 Wag"
> paste(rownames(mtcars[1,]), rownames(mtcars[4,]), sep = ",")
[1] "Mazda RX4,Hornet 4 Drive"
> paste(rownames(mtcars[2,]), rownames(mtcars[1,]), sep = "--")
[1] "Mazda RX4 Wag--Mazda RX4"
> paste(rownames(mtcars[3,]), rownames(mtcars[10,]), sep = "$")
[1] "Datsun 710$Merc 280"
> paste("hello","world",sep=" @ ")
[1] "hello @ world"
> paste("Assignment","5","3",sep="_")
[1] "Assignment_5_3"
```