



SESSION 5: Data Management Using R

Assignment 3

Table of Contents

1.Introduction	3
2.Objective	3
3.Prerequisites	3
4.Associated Data Files	3
5.Problem Statement	3

Data Analytics

6.Expected Output
3

7.Approximate Time to Complete Task
3

1. Introduction

This assignment will help you understand the concepts learnt in the session.

2. Objective

This assignment will test your skills on Performing SET operations in R.

3. Prerequisites

Not applicable.

4. Associated Data Files

Not applicable.

5. Problem Statement

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

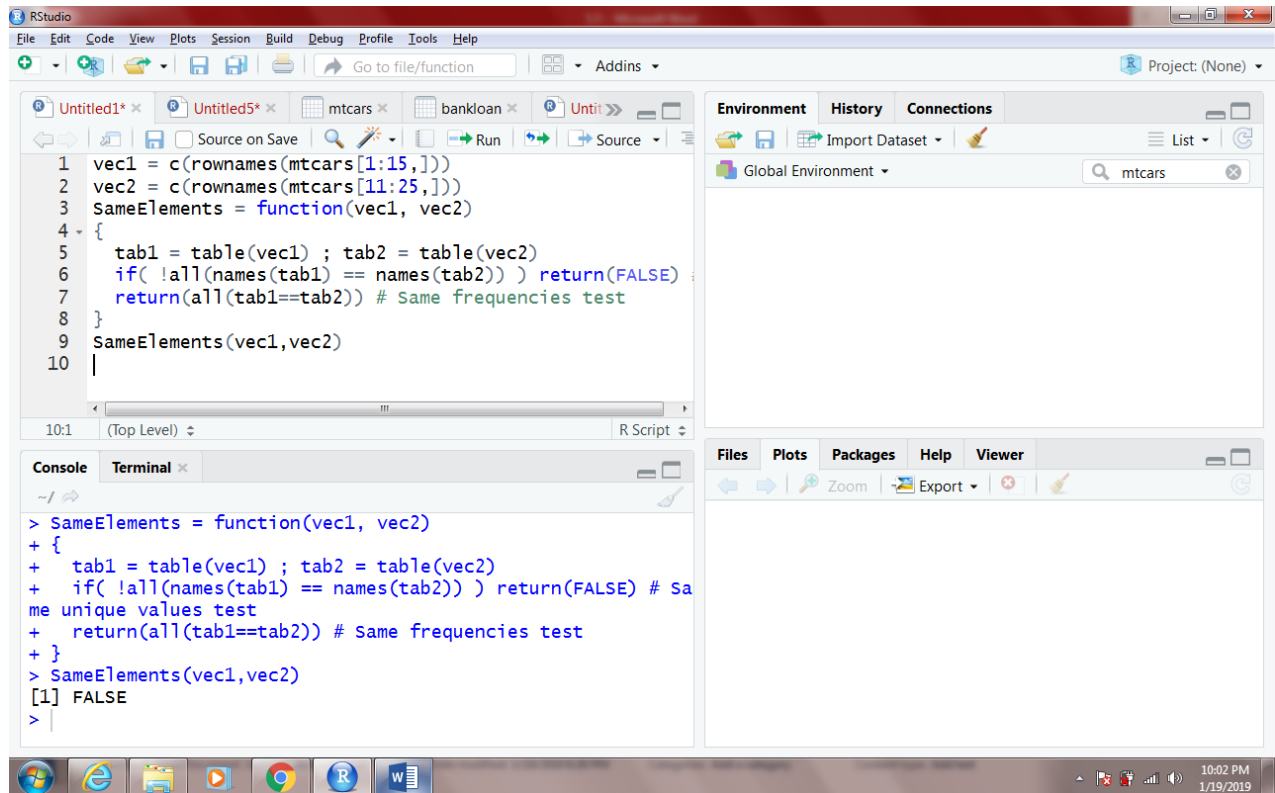
```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))
```

Ans –

```
vec1 = c(rownames(mtcars[1:15,]))
vec2 = c(rownames(mtcars[11:25,]))
SameElements = function(vec1, vec2)
{
  tab1 = table(vec1) ; tab2 = table(vec2)
  if( !all(names(tab1) == names(tab2)) ) return(FALSE) # Same
  unique values test
  return(all(tab1==tab2)) # Same frequencies test
}
```

}

Same Elements (vec1,vec2)



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for a function `SameElements` and its execution.

```
1 vec1 = c(rownames(mtcars[1:15,]))
2 vec2 = c(rownames(mtcars[11:25,]))
3 SameElements = function(vec1, vec2)
4 {
5   tab1 = table(vec1) ; tab2 = table(vec2)
6   if( !all(names(tab1) == names(tab2)) ) return(FALSE)
7   return(all(tab1==tab2)) # Same frequencies test
8 }
9 SameElements(vec1,vec2)
10
```
- Environment:** Shows the `Global Environment` with the `mtcars` dataset loaded.
- Console:** Displays the execution of the function, showing the function definition and the result `[1] FALSE`.

```
> SameElements = function(vec1, vec2)
+ {
+   tab1 = table(vec1) ; tab2 = table(vec2)
+   if( !all(names(tab1) == names(tab2)) ) return(FALSE) # Same
+   unique values test
+   return(all(tab1==tab2)) # Same frequencies test
+ }
> SameElements(vec1,vec2)
[1] FALSE
>
```
- Files, Plots, Packages, Help, Viewer:** These panels are visible at the bottom right of the RStudio window.

2. Sort the character vector in ascending order and descending

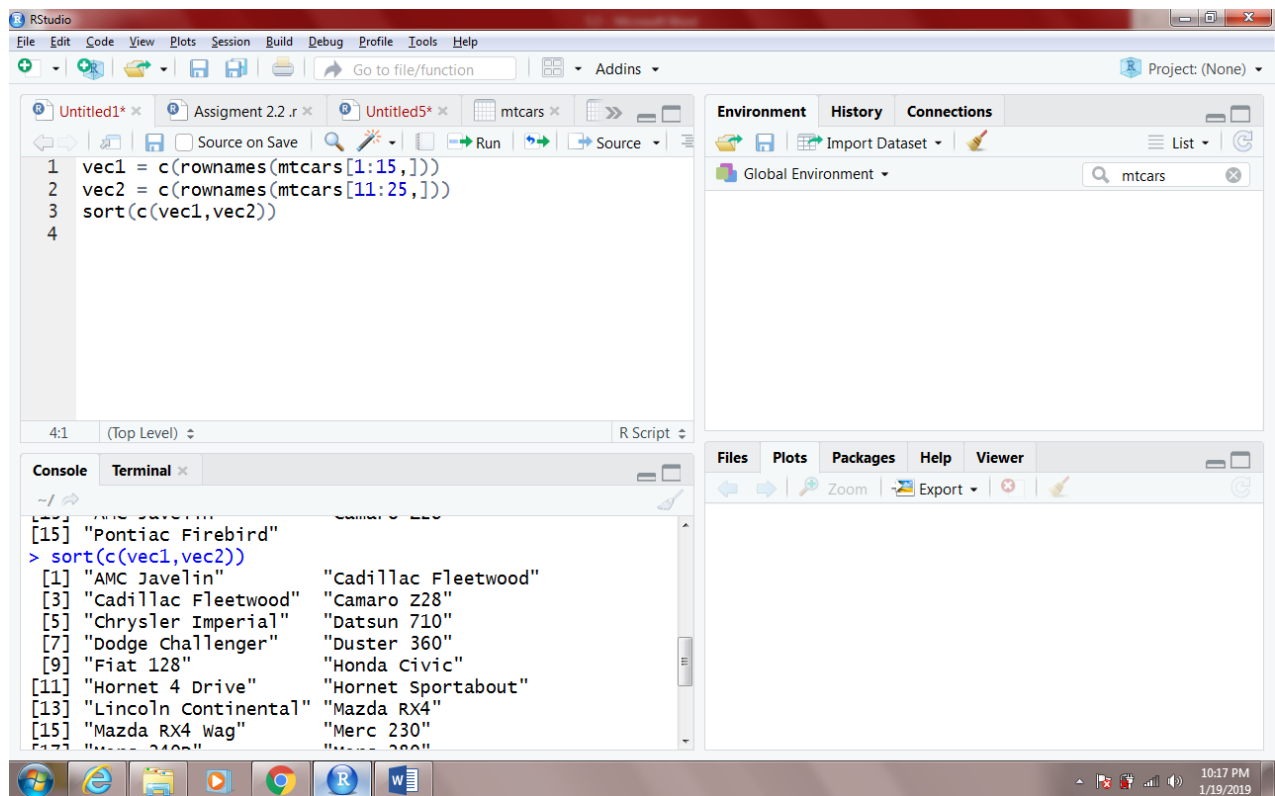
```
order vec1 = c(rownames(mtcars[1:15,]))
```

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

```
Ans. vec1 = c(rownames(mtcars[1:15,]))
```

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

```
sort(c(vec1,vec2))
```



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

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

```
sort(c(vec1,vec2))
```

```
sort(c(vec1,vec2),decreasing = TRUE)
```

Data Analytics

The screenshot shows the RStudio interface. The script editor contains the following R code:

```
1 vec1 = c(rownames(mtcars[1:15,]))
2 vec2 = c(rownames(mtcars[11:25,]))
3 sort(c(vec1,vec2))
4 sort(c(vec1,vec2),decreasing = TRUE)
5
6
```

The console shows the output of the command `sort(c(vec1,vec2),decreasing = TRUE)`:

```
> sort(c(vec1,vec2),decreasing = TRUE)
[1] "Valiant"           "Toyota Corona"
[3] "Toyota Corolla"    "Pontiac Firebird"
[5] "Merc 450SLC"       "Merc 450SLC"
[7] "Merc 450SL"        "Merc 450SL"
[9] "Merc 450SE"        "Merc 450SE"
[11] "Merc 280C"         "Merc 280C"
[13] "Merc 280"          "Merc 240D"
[15] "Merc 230"          "Mazda RX4 Wag"
[17] "Mazda RX4"         "Lincoln Continental"
[19] "Hornet Sportabout" "Hornet 4 Drive"
```

The Environment pane on the right shows the Global Environment with the `mtcars` object loaded. The bottom status bar indicates the time is 10:48 PM on 1/19/2019.

3. What is the major difference between `str_c()` and `paste()` show an example.

Ans - The difference is that `str_c` treats blanks as blanks (not as NAs) and recycles more appropriately.

1. `paste0(..., collapse = NULL)` is a wrapper for `paste(..., sep = "", collapse = NULL)`, which means there is no separator. In other words, with `paste0()` you can not apply some sort of separator, while you do have that option with `paste()`, whereas a single space is the default. `str_c(..., sep = "", collapse = NULL)` is equivalent to `paste()`, which means you do have the option to customize your desired separator. The difference is for `str_c()` the default is no separator, so it acts just like `paste0()` as a default.
2. `Paste()` and `paste0()` are both functions from the base package, whereas `str_c()` comes from the `stringr` package.

`str_c()` treats missing values properly

Example –

```
x <- LETTERS  
x[x %in% c("A", "E", "I", "O", "U")] <- NA  
y <- letters  
y[c(TRUE, FALSE, FALSE)] <- NA  
stringr::str_c(x, y)  
paste0(x, y)
```

`str_c()` warns on inexact recycling

```
paste0(month.abb, letters)  
stringr::str_c(month.abb, letters)
```

Data Analytics

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for creating a vector `x` of letters, filtering it, and creating a matrix `y` of random letters. The code is as follows:

```
1 x <- LETTERS
2 x[x %in% c("A", "E", "I", "O", "U")] <- NA
3 y <- letters
4 y[c(TRUE, FALSE, FALSE)] <- NA
5 stringr::str_c(x, y)
6 paste0(x, y)
7
```
- Console:** Displays the execution output:

```
> x[x %in% c("A", "E", "I", "O", "U")] <- NA
> y <- letters
> y[c(TRUE, FALSE, FALSE)] <- NA
> stringr::str_c(x, y)
[1] NA "Bb" "Cc" NA NA "Ff" NA "Hh" NA NA
[11] "Kk" "Ll" NA "Nn" NA NA "Qq" "Rr" NA "Tt"
[21] NA NA "Ww" "Xx" NA "Zz"
> paste0(x, y)
[1] "NANA" "Bb" "Cc" "DNA" "NAe" "Ff" "GNA"
[8] "Hh" "NAi" "JNA" "Kk" "Ll" "MNA" "Nn"
[15] "NAo" "PNA" "Qq" "Rr" "SNA" "Tt" "NAu"
```
- Environment:** Shows the 'Global Environment' with the variable `mtcars` loaded.
- Files, Plots, Packages, Help, Viewer:** These panels are currently empty.

4. Introduce a separator when concatenating the strings

```
str_c("Letter: ", letters)
str_c("Letter", letters, sep = ": ")
str_c(letters, " is for", "...")
str_c(letters[-26], " comes before ", letters[-1])
```

```
str_c(letters, collapse = "")
str_c(letters, collapse = ", ")
```

```
# Missing inputs give missing outputs
str_c(c("a", NA, "b"), "-d")
# Use str_replace_na to display literal NAs:
str_c(str_replace_na(c("a", NA, "b")), "-d")
# }
```

```
str1 = 'Hello'
str2 = 'World!'
str3 = 'Join'
str4 = 'Me!'
# concatenate two strings using paste function
result = paste (str1, str2, str3, str4, sep="-")
```

Data Analytics

The screenshot shows the RStudio interface. The source editor on the left contains an R script with the following code:

```
1 str1 = 'Hello'
2 str2 = 'World!'
3 str3 = 'Join'
4 str4 = 'Me!'
5 # concatenate two strings using paste function
6 result = paste(str1, str2, str3, str4, sep="-")
7
8 print (result)
9 result
10
```

The console on the bottom left shows the execution of the script:

```
> str2 = 'world'
> str2 = 'World!'
> str3 = 'Join'
> str4 = 'Me!'
> # concatenate two strings using paste function
> result = paste(str1, str2, str3, str4, sep="-")
> print (result)
[1] "Hello-World!-Join-Me!"
>
```

The Environment pane on the right shows the current state of the workspace:

```
# concatenate two strings using paste function
result = paste(str1, str2, str3, str4, sep="-")
str1 = 'Hello'
str2 = 'World!'
str2 = 'World!'
str3 = 'Join'
str4 = 'Me!'
# concatenate two strings using paste function
result = paste(str1, str2, str3, str4, sep="-")
print (result)
```

The screenshot shows the RStudio interface. The source editor on the left contains an R script with the following code:

```
8 str_c(letters, collapse = ", ")
9
10 # Missing inputs give missing outputs
11 str_c(c("a", NA, "b"), "-d")
12 # Use str_replace_NA to display literal NAs:
13 str_c(str_replace_na(c("a", NA, "b")), "-d")
14 # }
15
16 sdata = c('a', 'b', 'c')
17 s<- paste(sdata[1], sdata[2], sdata[3], sep = ' ')
18 s
19
```

The console on the bottom left shows the execution of the script:

```
> sdata = c('a', 'b', 'c')
> s<- paste(sdata[1], sdata[2], sdata[3], sep = ' ')
> s
[1] "a b c"
>
```

The Environment pane on the right shows the current state of the workspace:

```
s<- paste(sdata[1], sdata[2], sdata[3], sep = ' ')
sdata = c('a', 'b', 'c')
s<- paste(sdata[1], sdata[2], sdata[3], sep = ' ')
s
sdata = c('a', 'b', 'c')
s<- paste(sdata[1], sdata[2], sdata[3], sep = ' ')
s
```

6. Expected Format

1. R file should be submitted where applicable.
2. R file should be in PDF or in .r format
3. Proper screenshots of the outputs should be submitted as well
4. The r codes, if submitted in any other format, will be subjected to deduction in marks

Note: Your solution will not be entertained if it is any other format, e.g., .zip, .doc, .rtf etc.

7. Approximate Time to Complete Task

30 mins.