

Session 5 – Data Management using R

Assignment - 3

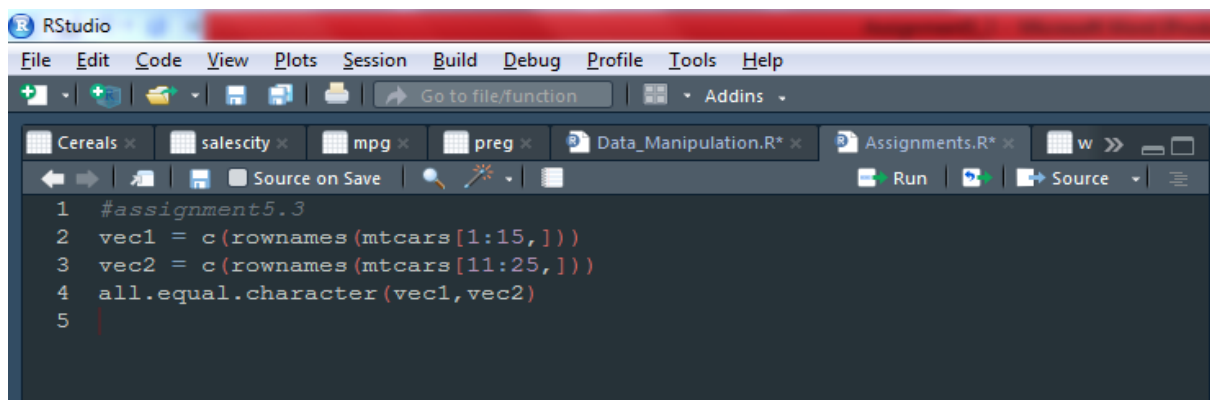
Problem Statement

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

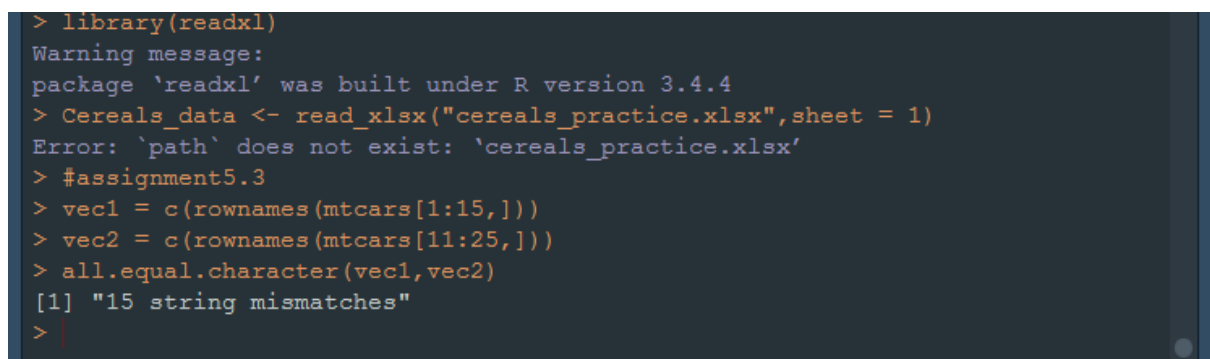
```
vec1 = c(rownames(mtcars[1:15,]))
```

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

Ans.

A screenshot of the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The file explorer shows several open files: Cereals, salescity, mpg, preg, Data_Manipulation.R, and Assignments.R*. The main editor window displays the following R code:

```
1 #assignment5.3
2 vec1 = c(rownames(mtcars[1:15,]))
3 vec2 = c(rownames(mtcars[11:25,]))
4 all.equal.character(vec1,vec2)
5 |
```

A screenshot of the R console output. It shows the execution of the code from the previous block. The output includes a warning message about the 'readxl' package and an error message about a non-existent file path. The final output of the comparison function is shown.

```
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.4.4
> Cereals_data <- read_xlsx("cereals_practice.xlsx",sheet = 1)
Error: `path` does not exist: 'cereals_practice.xlsx'
> #assignment5.3
> vec1 = c(rownames(mtcars[1:15,]))
> vec2 = c(rownames(mtcars[11:25,]))
> all.equal.character(vec1,vec2)
[1] "15 string mismatches"
> |
```

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

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

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

Ans.

```
5 vec1_asc <- sort(vec1,decreasing = FALSE)
6 vec1_asc
7 vec2_asc <- sort(vec2,decreasing = FALSE)
8 vec2_asc
9 vec2_desc <- sort(vec2,decreasing = TRUE)
10 vec2_desc
11 vec1_desc <- sort(vec1,decreasing = TRUE)
12 vec1_desc
13 |
```

```
> vec1_asc <- sort(vec1,decreasing = FALSE)
> vec1_asc
[1] "Cadillac Fleetwood" "Datsun 710"      "Duster 360"
[4] "Hornet 4 Drive"     "Hornet Sportabout" "Mazda RX4"
[7] "Mazda RX4 Wag"      "Merc 230"        "Merc 240D"
[10] "Merc 280"           "Merc 280C"       "Merc 450SE"
[13] "Merc 450SL"         "Merc 450SLC"     "Valiant"
> vec2_asc <- sort(vec2,decreasing = FALSE)
> vec2_asc
[1] "AMC Javelin"      "Cadillac Fleetwood" "Camaro Z28"
[4] "Chrysler Imperial" "Dodge Challenger"   "Fiat 128"
[7] "Honda Civic"       "Lincoln Continental" "Merc 280C"
[10] "Merc 450SE"        "Merc 450SL"        "Merc 450SLC"
[13] "Pontiac Firebird"  "Toyota Corolla"     "Toyota Corona"
> vec2_desc <- sort(vec2,decreasing = TRUE)
> vec2_desc
[1] "Toyota Corona"      "Toyota Corolla"     "Pontiac Firebird"
[4] "Merc 450SLC"        "Merc 450SL"        "Merc 450SE"
[7] "Merc 280C"         "Lincoln Continental" "Honda Civic"
[10] "Fiat 128"          "Dodge Challenger"   "Chrysler Imperial"
[13] "Camaro Z28"        "Cadillac Fleetwood" "AMC Javelin"
> vec1_desc <- sort(vec1,decreasing = TRUE)
> vec1_desc
[1] "Valiant"           "Merc 450SLC"        "Merc 450SL"
[4] "Merc 450SE"        "Merc 280C"         "Merc 280"
[7] "Merc 240D"        "Merc 230"          "Mazda RX4 Wag"
[10] "Mazda RX4"        "Hornet Sportabout"  "Hornet 4 Drive"
[13] "Duster 360"       "Datsun 710"        "Cadillac Fleetwood"
> |
```

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

Ans.

Paste()

`paste` converts its arguments (*via* `as.character`) to character strings, and concatenates them (separating them by the string given by `sep`). If the arguments are vectors, they are concatenated term-by-term to give a character vector result. Vector arguments are recycled as needed, with zero-length arguments being recycled to `""`.

Usage

```
paste(..., sep = " ", collapse = NULL)
paste0(..., collapse = NULL)
```

```
13 paste("Hello","World",sep = " ",collapse = "-")
14 |
```

```
> paste("Hello","World",sep = " ",collapse = "-")
[1] "Hello World"
> |
```

Str()

Compactly Display The Structure Of An Arbitrary R Object

Compactly display the internal structure of an R object, a diagnostic function and an alternative to `summary` (and to some extent, `dput`). Ideally, only one line for each 'basic' structure is displayed. It is especially well suited to compactly display the (abbreviated) contents of (possibly nested) lists. The idea is to give reasonable output for any R object. It calls `args` for (non-primitive) function objects.

```
14 str(vec1)
15 str(vec2_asc)
16 |
```

16:1 (Top Level) R Script

Console Terminal x

~/

```
[13] "Duster 360"          "Datsun 710"          "Cadillac Fleetwood"
> paste("Hello","World",sep = " ",collapse = -)
Error: unexpected ')' in "paste("Hello","World",sep = " ",collapse = -)"
> paste("Hello","World",sep = " ",collapse = -)
Error: unexpected ')' in "paste("Hello","World",sep = " ",collapse = -)"
> paste("Hello","World",sep = " ",collapse = "-")
[1] "Hello World"
> str(vec1)
chr [1:15] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
> str(vec2_asc)
chr [1:15] "AMC Javelin" "Cadillac Fleetwood" "Camaro Z28" ...
> |
```

4. Introduce a separator when concatenating the strings.

Ans.

```
16 paste("Hi","This","Is","R","Language",sep = "--")
17 |
```

17:1 (Top Level) R Script

Console Terminal x

~/

```
Error: unexpected ')' in "paste("Hello","World",sep = " ",collapse = -)"
> paste("Hello","World",sep = " ",collapse = -)
Error: unexpected ')' in "paste("Hello","World",sep = " ",collapse = -)"
> paste("Hello","World",sep = " ",collapse = "-")
[1] "Hello World"
> str(vec1)
chr [1:15] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
> str(vec2_asc)
chr [1:15] "AMC Javelin" "Cadillac Fleetwood" "Camaro Z28" ...
> paste("Hi","This","Is","R","Language",sep = "--")
[1] "Hi--This--Is--R--Language"
> |
```