Programming in Data science
Getting started with List in R Language

Asad Waqar Malik

Department of Information Systems
Faculty of Computer Science and Information Technology
University of Malaya
Malaysia.
asad.malik@um.edu.my

Sept 26, 2019

Lecture Outline

List Basics

- Lists are a special type of vector that can contain elements of different classes.
- Very commonly used data type in R.
- Lists can be explicitly created using the `list(...)` function, which takes an arbitrary number of arguments:

Sample

```
> x <- list(1, "a", TRUE, 1 + 4i)
> x
[[1]]
[1] 1
[[2]]
[1] "a"
[[3]]
[1] TRUE
[[4]]
[1] 1+4i
```

List Basics

- The component of the list may also have a name attached to it.

Sample

```
> my.list <- list(stud.id=12456, stud.name="Ali", ↩
    stud.marks=c(56,65,68,78))
> my.list
$stud.id
[1] 12456
$stud.name
[1] "Ali"
$stud.marks
[1] 56 65 68 78
```

- Access individual member of list.

```
> my.list$stud.id
```

Creating list from Vectors

- The following variable x is a list containing copies of three vectors n, s, b, and a numeric value 3.

Sample Code

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc", "dd", "ee")
> b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
> x = list(n, s, b, 3)
```

- Not all the vectors need to be of the same size.

Slicing

- List Slicing – We retrieve a list slice with the single square bracket "[]" operator. The following is a slice containing the second member of x, which is a copy of s.

Sample Code

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc", "dd", "ee")
> b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
> x = list(n, s, b, 3)
```

Slicing

```
> x[2]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
```

Index-based Slicing

- With an index vector, we can retrieve a slice with multiple members. Here a slice containing the second and fourth members of x.

Sample Code

```
> x[c(2, 4)]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
[[2]]
[1] 3
```

- Reference a list member directly using "[[]]" operator.

```
x[[2]][1] = "ta"
```

Reference List Members by Name

- We can assign names to list members, and reference them by names instead of numeric indexes. For example, in the following, v is a list of two members, named "bob" and "john".

Sample Code

```
> v = list(bob=c(2, 3, 5), john=c("aa", "bb"))
> v
$bob
[1] 2 3 5

$john
[1] "aa" "bb"
```

Reference List Members by Name

- We retrieve a list slice with the single square bracket "[]" operator.

```
> v["bob"]
$bob
[1] 2 3 5
```

- With an index vector, we can retrieve a slice with multiple members. Notice how they are reversed from their original positions in v.

```
> v[c("john", "bob")]
$john
[1] "aa" "bb"
$bob
[1] 2 3 5
```

Add new members in a List

- We can add new members to our list.

Update List

```
> my.list <- list(id=101, Name="Ali", Marks = c↩
    (56,58,65))
> new.list <- list(age=17, sex= "Male")
> my.list <- list(c(my.list, new.list))
> my.list   # Updated list with new members
```

- Adding an element at the end of the list. Can we also add new elements at the start of the list?

Modify list attributes

- Use `names(..)` to modify the attribute names.

Modify list attributes

```
> my.list <- list(ID=101, NAME="Ali")
> my.list
$ID
[1] 101

$NAME
[1] "Ali"
> names(my.list)
[1] "ID"    "NAME"

> names(my.list) <- c("Identification", "Nickname")
> names(my.list)
[1] "Identification" "Nickname"
```

Remove List Members

- Remove list members either using -ve index or through assign NULL value.

Remove List Members

```
> length(my.list)       # check the length
> my.list <- my.list[-4]     # remove the 4th item
> my.list[4] <- NULL
```

Operations on List

Predefined lists

```
> c(1,2,3) + 3  # easily operate on all list values at once
> letters #Predefined lists
> LETTERS #Predefined lists
> month.abb # Months abbrevations
> month.name # Complete name
```

List conversion to Vector

- List conversion using `unlist(..)` function.

Code

```
#List to vector conversion
> v <- unlist(my.list)
```

```
n1 <- list(1,2,3)
c1 <- list(4,5,6)
print("Original lists:")
print(n1)
print(c1)
print("Convert the lists to vectors:")
v1 = unlist(n1)
v2 = unlist(c1)
print("Add two vectors:")
v = v1 + v2
print("New vector:")
print(v)
```

Merge List

Merge Operation

```
> num_list <- list(1,2,3,4,5)
> day_list <- list("Mon","Tue","Wed", "Thurs", "Fri↩
   ")
> merge_list <- c(num_list, day_list)
> merge_list
```

Search Path Attachment

- Attach list to the R search path and access its members without explicitly mentioning the list. It should to be detached for cleanup.

Sample Code

```
> attach(v)
> bob
[1] 2 3 5
> detach(v)
```

apply(..) function family

- The apply(..) function is the most basic of all collection.
- It variation includes sapply(..), and lapply(..).
- Purpose of apply(..) is avoid explicit uses of loop constructs.

Function	Arguments	Objective	Input	Output
apply	apply(x, MARGIN, FUN)	Apply a function to the rows or columns or both	Data frame or matrix	vector, list, array
lapply	lapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	list
sapply	sappy(X FUN)	Apply a function to all the elements of the input	List, vector or data frame	vector or matrix

`apply(..)` function

Format

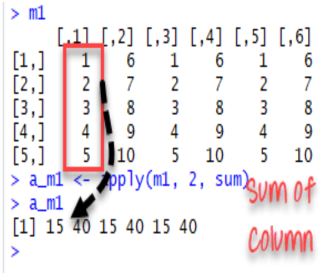
```
apply(x, MARGIN, FUN)
```

Here:
-x: an array or matrix
-MARGIN: take a value or range between 1 and 2 to define where to apply the function:
-MARGIN=1‘: the manipulation is performed on rows
-MARGIN=2‘: the manipulation is performed on columns
-FUN: tells which function to apply. Built functions like `mean, median, sum, min, max` and even user-defined functions can be applied.

apply(..) function

apply(..) Function Code

```
>m1 <- matrix(c<-(1:10),nrow=5, ncol=6)
>a_m1 <- apply(m1, 2, sum)
>a_m1
```



```
> m1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6    1    6    1    6
[2,]    2    7    2    7    2    7
[3,]    3    8    3    8    3    8
[4,]    4    9    4    9    4    9
[5,]    5   10    5   10    5   10
> a_m1 <- apply(m1, 2, sum)
> a_m1
[1] 15 40 15 40 15 40
>
```

`lapply(..)` function

Format

`lapply(X, FUN)`

Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x.

lapply(..) Function

- l in lapply(..) stands for list.
- The difference between lapply(..) and apply(..) lies between the output return. The output of lapply(..) is a list, lapply(..) can be used for other object like data frames.

Sample Code

```
>movies <- c("Fall","BATMAN")
>movies_lower <-lapply(movies, tolower)
>movies_lower
Output:
List of 3
 $ : chr "fall"
 $ : chr "batman"
```

```
>movies_lower <-unlist(lapply(movies,tolower))
```

`sapply(..)` function

Format

`sapply(X, FUN)`

Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x

- `sapply(..)` function does the same jobs as `lapply(..)` function but returns a vector.

sapply(..) Function

Sample Code

```
>x <- list(Z1 = 1, Z2 = 100:200)
>sapply(x, sum)
Z1       Z2
1    15150
```

Practise questions

- Write a R program to count number of objects in a given list?

```
> list_data <- list(c("Red","Green","Black"),
> list("Python", "PHP", "Java"))
> print("List:")
> print(list_data)
> print("Number of objects in the said list:")
> ?????
```

length()

Practise questions

- Write a R program to assign NULL to a given list element?

```
> l = list(1, 2, 3, 4, 5)
> print("Original list:")
> print(l)
> print("Set 2nd and 3rd elements to NULL")
> ????
```

l[c(2,3)] <- NULL

Practise questions

- Write a R program to create a list named s containing sequence of 15 capital letters, starting from 'E'?

```
> l1 <- ????
```

LETTERS[c(5:19)]

Practise questions

- Write a R program to Add 10 to each element of the first vector in a given list?
  Sample list: (g1 = 1:10, g2 = "R Programming", g3 = "HTML").

```
> list1 <- list(g1 = 1:10, g2 = "R Programming", g3↩
      = "HTML")
> print("Original list:")
> print(list1)
> print("New list:")
> ????
```

Practise questions

- Write a R program to extract all elements of a first vector except the third element of it from a given list. Sample list: (g1 = 1:10, g2 = "R Programming", g3 = "HTML").

```
> list1 = list(g1 = 1:10, g2 = "R Programming", g3 ↩
    = "HTML")
> print("Original list:")
> print(list1)
> print("First vector:")
> ????
```

Practise questions

- Write a R program to add a new item g4 = "Python" to a given list. Sample list: (g1 = 1:10, g2 = "R Programming", g3 = "HTML").

```
> list1 = list(g1 = 1:10, g2 = "R Programming", g3 ↩
    = "HTML")
> print("Original list:")
> print(list1)
> print("Add a new vector to the said list:")
> ????
```

Practise questions

- Write a R program to get the length of the first two vectors of a given list. Sample list: (g1 = 1:10, g2 = "R Programming", g3 = "HTML").

```
> list1 = list(g1 = 1:10, g2 = "R Programming", g3 = "HTML")
> print("Original list:")
> print(list1)
> print("Length of the vector g1 and g2 of the said list")
> ????
```

Practise questions

- Write a R program to find all elements of a given list that are not in another given list? Hint, see `setdiff(..)`.

```
> l1 = list("x", "y", "z")
> l2 = list("X", "Y", "Z", "x", "y", "z")
> print("Original lists:")
> print(l1)
> print(l2)
> print("All elements of l2 that are not in l1:")
> ????
```

Online quiz

Online Quiz