Lists in R

Introduction to data science (DSC 105) Fall 2024

Marcus Birkenkrahe

November 20, 2024

Practice creating, naming, slicing, changing lists

1. Create a *named* `list` called "List" of the following members:
   - A `numeric` vector consisting of the whole numbers 1 to 10.
   - A `logical` 3x3 identity matrix - use `diag`, `as.logical`, and `matrix` to build it (`diag(3)` is a 3x3 identity matrix).
   - A `character` vector, whose elements are the names Matthew, Mark, Luke, and John.

```
list(
  1:10,
  matrix(as.logical(diag(3)), ncol=3),
  c("Matthew","Mark","Luke","John")
) -> List
```

2. Print `List` and the structure of `List`.

```
List
str(List)
```

```
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
      [,1]  [,2]  [,3]
[1,]  TRUE FALSE FALSE
```

```
[2,] FALSE  TRUE FALSE
[3,] FALSE FALSE  TRUE

[[3]]
[1] "Matthew" "Mark"    "Luke"    "John"
List of 3
 $ : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ : logi [1:3, 1:3] TRUE FALSE FALSE FALSE TRUE FALSE ...
 $ : chr [1:4] "Matthew" "Mark" "Luke" "John"
```

3. Name the list members `"one to ten"`, `"identity"`, and `"evangelists"` using `names`. Print the `names` of `List`.

```
names(List) <- c("one to ten",
                 "identity",
                 "evangelists")
names(List)
```

```
[1] "one to ten"  "identity"    "evangelists"
```

4. Print the list structure again.

```
str(List)
```

```
List of 3
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:3, 1:3] TRUE FALSE FALSE FALSE TRUE FALSE ...
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
```

5. Remove last row of the matrix member and store the resulting list in `List2`. Print the structure of `List2` to confirm. In the correct result, the `identity` member should have the dimension 2 x 3 (`logi[1:2,1:3]`)..

   - Tip: You can simply overwrite any list member with a new value. Before doing that, test your code after making a copy of the list!
   - Test with copy:

```
L <- List
str(L)
L$identity[-3,]
L$identity <- L$identity[-3,]
str(L)
```

```
List of 3
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:3, 1:3] TRUE FALSE FALSE FALSE TRUE FALSE ...
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
      [,1]  [,2]  [,3]
[1,]  TRUE FALSE FALSE
[2,] FALSE  TRUE FALSE
List of 3
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
```

- Looks good, so do it with the original:

```
List$identity <- List$identity[-3,]
str(List)
List of 3
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
```

- Alternative: Build `List2` from the named members of `List` using the `list` function. Name each `list` member. Do not save until you're sure. You can use `rm` to remove objects that you've created.

```
list(
  "one to ten" = List[[1]],
  "identity" = List$identity[-3,],
  "evangelists" = List$evangelists) -> L
str(L)
List of 3
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
```

6. Create an ordered `factor` called `price` with the elements: cheap, pricy, cheap, cheap, pricy, where "cheap" is below "pricy".

Tip: The parameter to order a factor is called `ordered`, and the parameter to order its levels is called `levels`.

```
## build the factor
factor(c("cheap", "pricy", "cheap", "cheap", "pricy"))
## order the factor and save it
factor(c("cheap", "pricy", "cheap", "cheap", "pricy"),
       ordered=TRUE,
       levels=c("cheap", "pricy")) -> price
price
```

```
[1] cheap pricy cheap cheap pricy
Levels: cheap pricy
[1] cheap pricy cheap cheap pricy
Levels: cheap < pricy
```

7. Add the factor `price` to the end of `List2`. Remember to make a copy of list before altering it for good!

   - Adding the list member:

     ```
     L <- List2
     L[[4]] <- price
     str(L)
     ```

     ```
     Error: object 'List2' not found
     List of 4
      $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
      $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
      $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
      $            : Ord.factor w/ 2 levels "cheap"<"pricy": 1 2 1 1 2
     ```

   ```
   List of 4
    $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
    $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
    $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
    $ price      : Ord.factor w/ 2 levels "cheap"<"pricy": 1 2 1 1 2
   ```

   - Now for the real list, `List2`:

     ```
     List2[[4]] <- price
     str(List2)
     ```

```
    Error in List2[[4]] <- price : object 'List2' not found
    Error in str(List2) : object 'List2' not found

List of 4
 $ one to ten : int [1:10] 1 2 3 4 5 6 7 8 9 10
 $ identity   : logi [1:2, 1:3] TRUE FALSE FALSE TRUE FALSE FALSE
 $ evangelists: chr [1:4] "Matthew" "Mark" "Luke" "John"
 $ price      : Ord.factor w/ 2 levels "cheap"<"pricy": 1 2 1 1 2
```

8. Remove the last element of the `factor` member in `List2` using the `length` function, and save the result to `price2`.

```
## store last member index
length(List2) -> last_member
## show last member
List2[[last_member]]
## remove last element from last member and save it
List2[[last_member]][-length(last_member)] -> price2
price2

Error: object 'List2' not found
Error: object 'List2' not found
Error: object 'List2' not found
Error: object 'price2' not found
```

9. Replace the `price` factor by the new one and update `List2` to `List3`.

   - Tip: To remove member N from a list L, use `L[-N]`.
   - Solution 1: Replacing the element directly

```
List3 <- List2
str(List3)
List3$price <- price2
str(List3)

Error: object 'List2' not found
Error in str(List3) : object 'List3' not found
Error: object 'price2' not found
Error in str(List3) : object 'List3' not found
```

10. Transpose the matrix `identity` using the `t` function and save the resulting matrix in a new list `List4` in the second position as before.

    Transposition:

    ```
    ## transpose matrix and save it
    t(List3[[2]]) -> mat
    mat
    ```

    ```
    Error in t(List3[[2]]) : object 'List3' not found
    Error: object 'mat' not found
    ```

    Make copy of list and replace matrix member:

    ```
    ## Copy list into new list
    List4 <- List3
    str(List4)
    ## Replace 2nd element
    List4[[2]] <- mat
    str(List4)
    ```

    ```
    Error: object 'List3' not found
    Error in str(List4) : object 'List4' not found
    Error: object 'mat' not found
    Error in str(List4) : object 'List4' not found
    ```

Your final result should look like this:

```
List4
```

```
Error: object 'List4' not found
```