MTH3240 R Notes 5

1 Lists

1.1 Creating and Examining Lists

- Recall that *vectors* must be *homogeneous* (their **values** have to all be the **same type**).
- ***Lists*** are like vectors, but their elements can be *heterogeneous* (i.e. **not** all the same type).

  In fact, their elements can be *any* R objects. *Lists* are used to bundle objects together under one name.
- Lists are created and examined using:

```
list()            # Create a list
length()          # Returns the number of elements in a list
is.list()         # Indicates whether or not an object is a list
str()             # Describes the structure of a list
```

- Here's a simple example of a list containing three types of elements: a numeric value, a `"character"` value, and a `"logical"` value:

```
# y stores numeric, "character", and "logical" values
y <- list(2, "a", TRUE)
y

## [[1]]
## [1] 2
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE

is.list(y)

## [1] TRUE
```

  Notice R use **double square brackets** `[[ ]]` to display the indices of the list elements .

- Here's a list whose three elements consist of two vectors and a matrix:

```
x <- list(x1 = c(1, 2, 3),
          x2 = c("a", "b", "c"),
          x3 = matrix(1:4, nrow = 2, ncol = 2))
x

## $x1
## [1] 1 2 3
##
## $x2
## [1] "a" "b" "c"
##
## $x3
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

  Here, we assigned names to the list elements (`x1`, `x2`, and `x3` above). When R prints out a list whose elements have names, the names are preceded by a **dollar sign** `$`.

- `length()` tells how many elements are contained in a list:

```
length(x)

## [1] 3
```

- `str()` tells us the "structure" of the list:

```
str(x)

## List of 3
##  $ x1: num [1:3] 1 2 3
##  $ x2: chr [1:3] "a" "b" "c"
##  $ x3: int [1:2, 1:2] 1 2 3 4
```

  The output above says that `x` is a list with three elements: a numeric vector `x1`, a `"character"` vector `x2`, and an `"integer"` matrix `x3`.

Section 1.1 Pre-Lab Exercises

Exercise 1.1

a) Write commands that create a list named `Employees` containing three elements, each being one of the following vectors. Report your R command(s):

   - `Name`, a `"character"` vector containing the names of four employees:

```
c("Joe", "Kim", "Ann", "Bob")
```

- `Salary`, a numeric vector containing their salaries:

```
c(56000, 67000, 60000, 55000)
```

- `Union`, a `"logical"` vector indicating whether or not they belong to a union:

```
c(TRUE, TRUE, FALSE, FALSE)
```

b) Use `str()` to look at the structure of the list. Report the results.

c) Use `length()` to find the number of elements in the list. Report the result.

1.2 General List Operations

1.2.1 Accessing List Elements

- We access elements of a list using **double square brackets** `[[ ]]` or the **dollar sign** `$`:

```
[[ ]]          # Access a list element via its index or name
$              # Access a list element via its name
```

- The main distinctions between using `[[ ]]` and `$` are:
  - `[[ ]]` can be used with numerical indices or names of list elements (in quotation marks).
  - `$` can only be used with names of list elements (without quotation marks).
- Here's an example using `[[ ]]` (and the list `x` from above):

```
x[[2]]                          # We could also use x[["x2"]]

## [1] "a" "b" "c"
```

Note that a `"character"` vector is returned. We could also have used `x[["x2"]]`.

- Here's an example using `$`:

```
x$x2

## [1] "a" "b" "c"
```

A `"character"` vector is returned here too.

1.2.2 Adding and Deleting List Elements

- The operators `[[ ]]` and `$` can also be used to add or delete a list element.

  Below, we add the value 16 as the fourth element of `x` and give it the name `x4`:

```
x$x4 <- 16                      # We could also use x[["x4"]] <- 16
x

## $x1
## [1] 1 2 3
##
## $x2
## [1] "a" "b" "c"
##
## $x3
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## $x4
## [1] 16
```

  Above, we could also have used `x[["x4"]] <- 16`.

- To delete a list element, we set its value to `NULL`:

```
x$x4 <- NULL
x

## $x1
## [1] 1 2 3
##
## $x2
## [1] "a" "b" "c"
##
## $x3
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Section 1.2 Pre-Lab Exercises

Exercise 1.2 Here's the `Employees` list from Exercise 1:

```r
Employees <- list(Name = c("Joe", "Kim", "Ann", "Bob"),
                  Salary = c(56000, 67000, 60000, 55000),
                  Union = c(TRUE, TRUE, FALSE, FALSE))
```

a) Guess what will returned by the following commands, then check your answers:

```r
Employees[[2]]
```

```r
Employees$Salary
```

b) Write a command involving `[[ ]]` that returns the `"logical"` union membership vector.

c) Write a command involving `$` that returns the `"logical"` union membership vector.

d) Here's a numeric vector containing the employee's years of experience:

```r
c(14, 6, 9, 12)
```

The following commands both do the same thing. What do they do?

```r
Employees[["Experience"]] <- c(14, 6, 9, 12)
```

```r
Employees$Experience <- c(14, 6, 9, 12)
```

1.3 Named List Elements

- We can get or add list element names using:

```
names()        # Examine or change the names of list elements
```

- As an example, consider again the list `x`:

```r
x

## $x1
## [1] 1 2 3
##
## $x2
## [1] "a" "b" "c"
```

```
##
## $x3
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

To get the list element names, we type:

```
names(x)

## [1] "x1" "x2" "x3"
```

and to change them to, say, `y1`, `y2`, and `y3`, we type:

```
names(x) <- c("y1", "y2", "y3")
names(x)

## [1] "y1" "y2" "y3"
```

- To remove the names, we'd type:

```
names(x) <- NULL
```

Section 1.3 Pre-Lab Exercises

Exercise 1.3 Here's a simple list `x`:

```
x <- list(x1 = c(1, 2), x2 = c("a", "b"), x3 = 12)
```

a) What will the following command return? Check your answer.

```
names(x)
```

b) Write a command that will change the names of the elements of `x` to `y1`, `y2`, and `y3`. Make sure to check your answer.