Lab 1 Activity

Lab Activity

Some R Practice

1. The `rep()` function can be used to repeat objects:

```r
# we define vectors with the `c()` function

vec <- c(2, 3)
vec
```

```
## [1] 2 3
```

```r
# can you spot the difference between the `times =` and `each =` argument?

rep(vec, times = 3)
```

```
## [1] 2 3 2 3 2 3
```

```r
rep(vec, each = 3)
```

```
## [1] 2 2 2 3 3 3
```

Task

- Repeat the vector `c(-1,3,-5,7,-9)` twice, with each element repeated 3 times, and store the result. Display the result sorted from largest to smallest (use the `sort()` function; run `help(sort)` for the help page of the `sort()` function).

2.

Task

- If I have a vector with 5 elements, I use the `rep()` function to repeat each element 4 times, and then I repeat the resulting vector 3 times, how long is the final vector?

3. Let's look at some common R objects and how to subset them:

vectors (1-Dimensional):

```r
x <- c(2,1,5,6, 17)

# element 3 is extracted like so

x[3]
```

```
## [1] 5
```

```
# elements 1 and 3 can be extracted like so

x[c(1,3)]
```

```
## [1] 2 5
```

```
# elements can be dropped by using the "-" sign
# to drop the elements 2 and 4

x[-c(2,4)]
```

```
## [1]  2  5 17
```

matrices (2-Dimensional):

```
# many ways to define matrices
# a quick way is to use the `cbind()` of `rbind()` functions
# `cbind()` glues together 1D vectors as columns
# `rbind()` glues together 1D vectors as rows

# for example we create a matrix we 3 rows and 4 columns

mat <- rbind(c(4,2,5, 8),
             c(14,78,6, 38),
             c(33,7,10, 326))

# to extract the element in the second row and the fourth column, we run

mat[2, 4]
```

```
## [1] 38
```

```
# so for 2D objects, [Row, Column]

# to get the second column

mat[,2]
```

```
## [1]  2 78  7
```

lists (1-Dimensional):

```
# lists can store any type of object
# to create a list

list_1 <- list(x, mat)

# to extract elements from a list we use [[]]
# so to extract the second element

list_1[[2]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    2    5    8
## [2,]   14   78    6   38
## [3,]   33    7   10  326
```

```
# we can also name list elements

list_names <- list("vector" = x,
                   "matrix" = mat)

# now we can also use the `$` operator to access named elements

list_names$matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    2    5    8
## [2,]   14   78    6   38
## [3,]   33    7   10  326
```

```
# the [[]] method still works just fine

list_names[[2]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    2    5    8
## [2,]   14   78    6   38
## [3,]   33    7   10  326
```

Tasks

- If you create a new vector `y` that contains the first to the fourth element of `x` and the third row of the `mat` object, what will be the length of `y`?

- Create `y` as specified above. Print `y`.
- Extract the second and fifth elements of the “vector” element inside the `list_names` object in a single line of code
- assign `NA` to the element in the third row and second column of the “matrix” element inside the `list_names` object in a single line of code. (hint: you will need to use the `<-` operator). Print the “matrix” element inside the `list_names` object to confirm.