Lecture 5
List and data frame

GEOG 489 SPRING 2020

Matrix

2-d looping

The goal is to write a loop that will go through all possible combinations of a matrix in 2d.

```
myMatrix <- matrix(seq(10,60,by=10),nrow=3,ncol=2)
for(current_col in seq(2))
{
    for(current_row in seq(3))
     {
        print(paste("the center cell value is:",
                            myMatrix[current_row,current_col]))
    }
}
myMatrix
```

List

- A list is a collection of objects of different types

Lists have somewhat complex indexing, but are important for dealing with complicated objects (such as GIS data).

- A list is, at its core, another type of vector.

Most of the vectors we have used so far (vectors and matrices) are *atomic* vectors, which cannot be broken down into smaller components.

A list, by comparison, is a *recursive* vector. It can be broken down into smaller components.

List

1) Create a list

A list of different data types (namely a character, numeric, and logical)

```
j <- list(name="Joe", salary=55000, union=T)
```

2) Create a empty list

```
z <- vector(mode="list")
```

```
z[["abc"]] <- 3
```

List

3) General list operations: indexing

lists have a few ways to index them. Going back to our "j" list:

j$salary   # Just use the dollar sign and the component name (or an abbreviated one)

j[["salary"]]

j[[2]]   # This works because the 2nd element of the list is the salary vector.

List

4) Double brackets vs. single brackets

Single bracket: returns an object of class list (a sublist)

```
class(j[2])   # “list”
```

Double bracket: returns an object in its native class

```
class(j[[2]])     # “numeric”
```

List

4) Adding and deleting list elements

```
z <- list(a="abc",b=12)

z$c <- "sailing"     # This adds a new component "c" to z which is a character vector.


# To delete a list component, set it to NULL:

z$b <- NULL
```

List

5) "unlist" a list: coerce everything in a list down to an atomic vector.

```
j <- list(name="Joe", salary=55000, union=T)

ulj <- unlist(j)     # This becomes an atomic vector
```

List

6) lapply() and sapply(): apply functions to lists

lapply() and sapply() are equivalents to apply, except they take list inputs.

```
z <- list(1:3,25:29)

output <- lapply(X=z,FUN=median)    # output is “list”
```

Note: sapply() is very similar to lapply(), but does a check to see if it can "simplify" the output: i.e. reduce it to a matrix or vector:

```
sapply_out <- sapply(X=z,FUN=median) # output is “vector”
```

List

7) Recursive list: a list within a list

```
b <- list(u=5,v=12)

c <- list(w = 13:15)

a <- list(b,c)
```

Data Frame

1) Create data frame

```
random_data <-
data.frame(data1=runif(5),data2=runif(5),data3=runif(5))
```

```
# Note that runif(x) generates x random numbers between 0 and 1.
```

2) Data frame indexing

```
random_data[2:5, ] # Returns observation (rows) 2 through
5, all variables (columns).
```

```
random_data[random_data$data1 >= 0.5, ]
```

```
subset(random_data,data1 >= 0.5)
```

Data Frame

3) Add data frame rows and columns

rbind() adds a row (and must have the same number of elements as the data.frame).

```
random_data <- data.frame(data1=runif(5),data2=runif(5),data3=runif(5))

xnewrow <- c(2,"abc",6)

xnew <- rbind(random_data,xnewrow)
```

cbind() needs to have the same number of rows as the data frame.

```
random_data_new <- cbind(random_data,(random_data$data1-
random_data$data2))
```

Data Frame

4) apply() and lapply: apply() can be used on a data frame IF all the columns are the same type

```
random_data <-
data.frame(data1=runif(5),data2=runif(5),data3=runif(5))

apply(random_data,2,max)
```

lapply() applied to a data frame will apply the function to each *column*, and return a list.

```
d2 <- data.frame(ages=c(10,7,12),names=c("Jill","Jillian","Jack"))

dl <- lapply(d2,sort)
```

Data Frame

5) Merge data frame: Data frames are R's version of a spreadsheet. Like any properly formatted table, we can use relational operators to join two tables together. The basic command is merge

```
d1 <- data.frame(names=c("Jack","Jill","John"),states=c("CA","IL","IL"))
d2 <- data.frame(ages=c(10,7,12),names=c("Jill","Jillian","Jack"))
merge(d1,d2)


# What if we don't have the same variable name?
names(d2) <- c("ages","kids")
merge(x=d1,y=d2,by.x="names",by.y="kids")
```

Assignment 1

Your goal is to write a function that takes two inputs:

x = a vector of numbers

d = a single value

The function should return a vector of numeric indices of all vector locations in which the element of x divided by d has no remainder.

Assignment 1 is due on Tuesday, Feb. 11 at midnight.

Please submit your assignment on Compass 2g

Quiz 2

Please create a matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, and use the 2-d loop to change the elements of the first row to 10, and to change the elements of the second row to 20. So you will get a matrix $\begin{bmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \end{bmatrix}$

The R file needs to be named: LastName_FirstName_Quiz2.R

Please submit the quiz R file on Compass by the end of this class.