Problem Set 1

Shisham Adhikari

Math 345

Due: Before class on Thursday, 02/04/2021

Collaborators: Grayson White

Goals of this lab

1. Review R!
2. Ease our way back into the semester.
3. Hopefully learn at least one new thing.

Problem 1: Something New

Let's start by (hopefully) learning something new. Go online and look up the pacman package. Why is it called pacman? Why might this package be helpful?

As pacman in the videogame eats the dots inside an enclosed maze to get stronger, pacman package combines/eats the functionality of base library related functions to give more intuitively named functions, so it is called pacman.

This package might be helpful to:

- provide intuitively named functions for the base functions
- integrate the functionality of the multiple functions to perform multiple tasks in single command
- increase workflow by reducing code and time recalling obscure functions and
- substitute for the functions library() and require()

a) Load a few of your favorite packages using p_load().

```r
library(pacman)
p_load(tidyverse)
p_load(lubridate)
p_load(wordcloud)
p_load(ggmap)
```

Problem 2: List of my Favorite Things

Here's a list:

```r
# here's a list
l <- list(1:4, "a", 47, "zucchini the dog", NA)
```

a) Return a boolean vector that is true when our object is a character class.

```r
is.character(l)
```

```
## [1] FALSE
```

b) Write code that returns the locations in our list of the character class data.

```
loc <- rep(NA, length(l))
for(i in 1:length(l)){
  loc[i] <- is.character(l[[i]])
}
which(loc %in% TRUE)
```

```
## [1] 2 4
```

c) Find the sum of all the numbers (including integers) in the list l.

```
sum(as.numeric(unlist(l)), na.rm=TRUE)
```

```
## [1] 57
```

Problem 3: Fun(ctions) with Matrices

```
# define the matrix
m <- matrix(data = 1:20, nrow = 4)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
```

a) Switch columns 1 and 3. Then switch rows 1 and 2. Print the results.

```
m[, c(1,3)]  <- m[ , c(3,1)]
m[c(1,2), ]  <- m[c(2,1) , ]
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   10    6    2   14   18
## [2,]    9    5    1   13   17
## [3,]   11    7    3   15   19
## [4,]   12    8    4   16   20
```

b) Write a nice function that takes 4 inputs: a matrix, two integers, and the text "row" or "column". Write a function that switches the rows or columns, depending on the input, corresponding to the two integers. For example, if the input was `matrix_swap(m, 1, 3, "column")`, the function would return the matrix m with columns 1 and 3 switched, as we did in part (a). Make sure your function responds appropriately when the inputs are nonsensical; for example, if a user asks to swap a column that does not exist in the supplied matrix. Test it and print the results for a at least 3 new matrices that you create (one row swap, one column swap, and one nonsensical case).

```
# Write your function here
nice_fun <- function(mat, x, y, z="row"|"column") {
  if(x>nrow(mat)|y>ncol(mat)){
      print("Nonsensical inputs, check the values of x and y")
  } else {
  if (z=="row") {
    mat[c(x,y), ]  <- mat[c(y,x) , ]
    print(mat)
  } else {
  mat[, c(x,y)]  <- mat[ , c(y,x)]
  print(mat)
  }
  }
}
```

```
# Test your function here
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   10    6    2   14   18
## [2,]    9    5    1   13   17
## [3,]   11    7    3   15   19
## [4,]   12    8    4   16   20
```

```
nice_fun(m, 1,2,"row")
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    5    1   13   17
## [2,]   10    6    2   14   18
## [3,]   11    7    3   15   19
## [4,]   12    8    4   16   20
```

```
nice_fun(m, 1,2,"column")
```

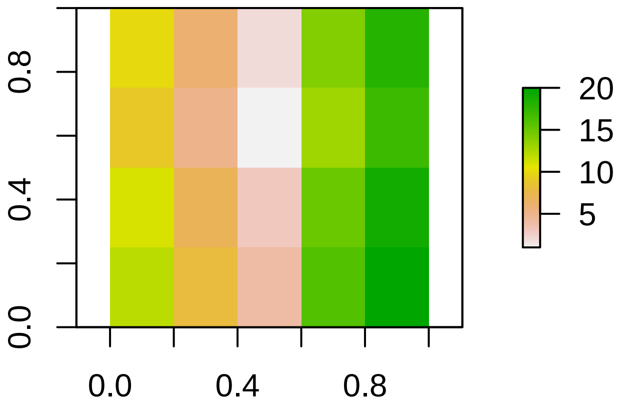
```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    6   10    2   14   18
## [2,]    5    9    1   13   17
## [3,]    7   11    3   15   19
## [4,]    8   12    4   16   20
```

```
nice_fun(m, 10,2,"row")
```

```
## [1] "Nonsensical inputs, check the values of x and y"
```

c) Plot one of the matrices as a raster using your preferred method.

```
# Plot code here
library(raster)
r=raster(m)
plot(r)
```


0.0
0.4
0.8
0.0
0.4
0.8
20
15
10
5