Homework 2

Matthew Runyon

9/4/2020

Problem #1: Given the following system of equations, solve for x and y:

`2x+5y=33`

`x+3y=19`

This system of equations can be solved using the `solve()` function. However, we first need to set up these equations as matrices to be used in this function, one matrix for the variable coefficients (`A`), and one matrix for the numeric value on the right-hand side of the equation (`B`). To do so, we will use the `matrix()` function.

Regarding matrix `A`, because these equations have only an `x` and `y` variable to solve for, and there are two equations, we will set up a 2x2 matrix by setting the `ncol` and `nrow` arguments equal to 2. The `data` argument is satisfied by compiling the variable coefficients as a vector, starting with everything in the first equation before moving to the second equation. Since we have multiple coefficients for the variables in the same equation, we also want to set the `byrow` argument equal to `TRUE` so that the coefficients are read in the correct order.

```
A<-matrix(data=c(2,5,1,3), nrow=2, ncol=2, byrow=T)

A
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    1    3
```

Regarding matrix `B`, this will be a 2x1 matrix, so we will set the `nrow` and `ncol` argurments equal to 2 and 1, respectively. Since there is only one column, we do not have to specify a `byrow` argument.

```
B<-matrix(data=c(33,19), nrow=2, ncol=1)

B
```

```
##      [,1]
## [1,]   33
## [2,]   19
```

Now that the matrices have been constructed, we can use the `solve()` function to obtain the values of the `x` and `y` variables, respectively. These will be computed such that the first value returned is the `x` variable value, and the second is the `y` variable value.

```
solve(A,B)
```

```
##      [,1]
## [1,]    4
## [2,]    5
```

Therefore, `x=4` and `y=5`.

Problem #2: Create a vector that goes from 1 to 100, then calculate the average without using the function `mean()`:

First, we will create a vector that spans 1 to 100 using the `c()` function.

```
Vector1<-c(1:100)
```

Next, we can use the `sum()` and `length()` functions to calculate the average of this vector in the same way that the `mean()` function would do. `sum()` will give the summation value of the vector, and `length()` will give the number of values in the vector.

```
Sum<-sum(Vector1)

Sum
```

```
## [1] 5050
```

```
Length<-length(Vector1)

Length
```

```
## [1] 100
```

The average is found by dividing the `sum()` value by the `length()` value.

```
Sum/Length
```

```
## [1] 50.5
```

Problem #3: Install the package `dplyr`:

Installing the package `dplyr` can be done by running the `install.packages()` function. **NOTE** I already have `dplyr` installed, so running the following code returns several aberrant warning messages.

```
install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/mrunyon2/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'dplyr' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'dplyr'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying C:
## \Users\mrunyon2\Documents\R\win-library\3.6\00LOCK\dplyr\libs\x64\dplyr.dll
## to C:\Users\mrunyon2\Documents\R\win-library\3.6\dplyr\libs\x64\dplyr.dll:
## Permission denied
```

```
## Warning: restored 'dplyr'
```

```
##
## The downloaded binary packages are in
##  C:\Users\mrunyon2\AppData\Local\Temp\Rtmp4e7G1Z\downloaded_packages
```

After successfully installing this package, it can be enabled by using the `library()` function.

```
library("dplyr")
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Problem #4: Create a data frame of two rows with A, B, C, and D in the top row and D, E, F, and G in the bottom row:

To do set this up, I am first formatting my data as a matrix before making it a data frame. To do this, I set up a matrix with a vector containing `A, B, C, D, D, E, F, G`, had it formatted to two rows by setting the `nrow` argument equal to 2, and had it read in the proper order by setting the `byrow` argument equal to `TRUE`. To better format the data frame, I went ahead and renamed the rows to `Row_1` and `Row_2` using the `row.names()` function.

```
Q4<-data.frame(
  matrix(c("A", "B", "C", "D", "D", "E", "F", "G"), nrow=2, ncol=4, byrow=TRUE),
  row.names=c("Row_1", "Row_2"))
```

Furthermore, I renamed the individual columns as follows with the `colnames()` function.

```
colnames(Q4)<-c("Col_1", "Col_2", "Col_3", "Col_4")
Q4
```

```
##       Col_1 Col_2 Col_3 Col_4
## Row_1     A     B     C     D
## Row_2     D     E     F     G
```