Lab #4 Activities

Adejare Taiwo

A reminder that the R code we have covered in class is available on the lecture section UM Learn page, under **Content** > **Course Material**.

Knit this file to pdf to see the questions in a more readable format.

Question 1:

The `mtcars` dataset is built-in to R.

(a) Create a new data frame, and then **print it**, consisting of all columns of `mtcars`, plus an additional column called `mpg_cat` consisting of "low", "medium low", "medium high", or "high" for each car, depending on if the miles per gallon is:

- less than 15: "low"
- in the range [15,20): "med low"
- in the range [20,30): "med hi"
- at least 30: "high"

```
mtcars$mpg_cat = ifelse(mtcars$mpg < 15, "low", ifelse(mtcars$mpg < 20, "med low", ifelse(mtcars$mpg < 3
print(mtcars)
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb mpg_cat
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4  med hi
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4  med hi
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1  med hi
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1  med hi
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 med low
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 med low
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4     low
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2  med hi
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2  med hi
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 med low
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 med low
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 med low
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 med low
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 med low
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4     low
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4     low
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4     low
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1    high
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2    high
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1    high
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1  med hi
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2 med low
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2 med low
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4     low
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 med low
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1  med hi
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2  med hi
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2    high
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4 med low
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 med low
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 med low
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2  med hi
```

(b) Create an R list with three components:

- the miles per gallon measurements for all cars with weights below 3000 lbs (only the mpg measurements, no other columns)
- the miles per gallon measurements for all cars with weights between between 3000 and 5000 lbs
- the miles per gallon measurements for all cars with weights above 5000 lbs

and **print the list**. Note the `wt` column gives weights in 1000 lbs.

(There are no cars exactly on the boundaries between categories.)

Then use the `sapply()` function to obtain the mean miles per gallon for cars in each of these three weight categories.

```
lightCarsMpg = mtcars$mpg[mtcars$wt < 3];
mediumCarsMpg = mtcars$mpg[mtcars$wt >= 3 & mtcars$wt < 5]
heavyCarsMpg = mtcars$mpg[mtcars$wt >= 5]

mpgList = list(
  lightCarsMpg = lightCarsMpg,
  mediumCarsMpg = mediumCarsMpg,
  heavyCarsMpg = heavyCarsMpg
)

print(mpgList)
```

```
## $lightCarsMpg
##  [1] 21.0 21.0 22.8 32.4 30.4 33.9 21.5 27.3 26.0 30.4 19.7 21.4
## 
## $mediumCarsMpg
##  [1] 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 15.5 15.2 13.3 19.2
## [16] 15.8 15.0
## 
## $heavyCarsMpg
## [1] 10.4 10.4 14.7
```

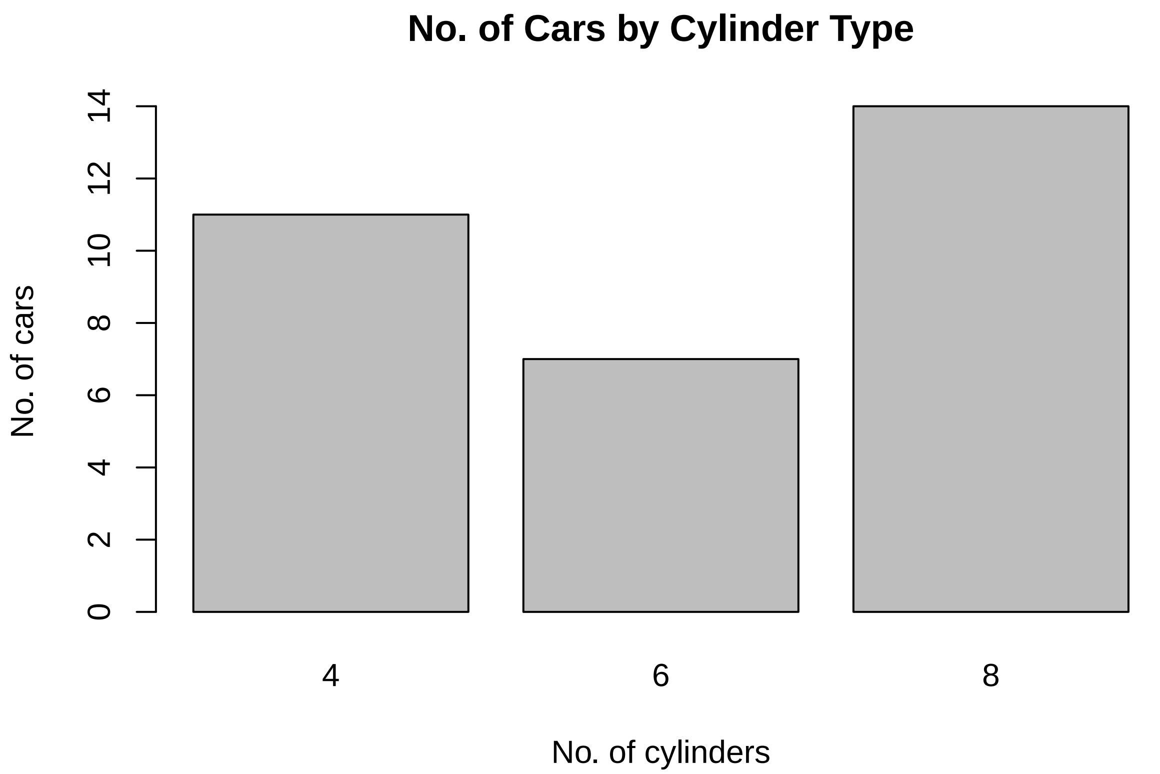
```
meanMpg = sapply(mpgList, mean)
print(meanMpg)
```

```
##  lightCarsMpg mediumCarsMpg  heavyCarsMpg 
##      25.65000      17.62353      11.83333
```

(c) Make a bar chart showing the number of cars in the dataset that are 4-cylinder, 6-cylinder, or 8-cylinder.

```r
cylinderCount = table(mtcars$cyl);

barplot(cylinderCount,
  main = "No. of Cars by Cylinder Type",
  xlab = "No. of cylinders",
  ylab = "No. of cars"
)
```



Question 2:

Consider the below vector `rolls`, where we simulate 100 rolls of two six-sided dice, where each die has the numbers (1,2,3,4,5,6). The values in the vector are the sum of the two dice (the possible sums are 2, 3, . . . , 11, 12) for these 100 rolls.

```r
myseed = 2
set.seed(myseed)
rolls = sample(2:12,100,replace=TRUE,prob=c(1,2,3,4,5,6,5,4,3,2,1)/36)
rolls
```

```
##   [1]  8  4  9  8  3  3  7 11  5  5  5  8 10  8  6 11  2  8  5  7  9  6 11  7  6
##  [26]  5  7  6 12  7  7  7 10 11  5  9 11  8  4  7  2  8  7  7  3 10  2  6  5 10
##  [51]  7  7  4  3  8 10 10  2  9  4 10 11  9  8 11  6  6  5  8  7  8  6  7  8  8
##  [76] 10  8 11  6  9  6  4  7  6  8 11 12  6  4  6  2  6  6  9  5  8  6  7  7  6
```

We can see that a 12 (obtained from two sixes) occurs for the first time on roll #29. Write the R code that takes in the `rolls` vector and uses a `while()` loop to determine when the first 12 has occurred. When the `while()` loop stops running, a variable should contain the value 29 after iteratively going through the `rolls` vector. Print the variable to show that it contains 29.

```
count = 1;
while(rolls[count] != 12) {
  count = count + 1
}

print(count)
```

```
## [1] 29
```