In Class Exercises: Lecture 13

1. For this in class exercise we will be using the boulder_ammonia data. Show the R commands to load this data. You may use the `.rda` version from a previous lecture if needed.

```
load("dat/boulder_ammonia.rda")
library(mowateR)
```

2. Was the ammonia level at the end of the day on January 1, 2019 higher than the beginning or vice versa? Write an `if()` statement that prints "Ammonia was higher at the beginning of the day" if it is, and otherwise prints "Ammonia was higher at the end of the day." if it isn't.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
all_days <- day(boulder_ammonia$Date.Time)
all_months <- month(boulder_ammonia$Date.Time)

first_day <- which(all_days == 1 & all_months == 1)
head(first_day)
```

```
## [1] 1 2 3 4 5 6
```

```
tail(first_day)
```

```
## [1] 282 283 284 285 286 287
```

```
boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[1]
```

```
## [1] 0.7907111
```

```
boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[278]
```

```
## [1] 1.40455
```

```
if(boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[1] > boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[278]) {
  print("Ammonia was higher at the beginning of the day.")
} else {
  print("Ammonia was higher at the end of the day.")
}
```

```
## [1] "Ammonia was higher at the end of the day."
```

3. Create an if statement that will check what quantile (first, second, third, or fourth) the last ammonia measurement for January 1st falls into. Then, have it print to the screen which quantile the measurement is in, such as "Ammonia level is within the first quantile."

```
last_ammonia_Jan1 <- boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[278]

ammonia_quantiles <- quantile(boulder_ammonia$AB3.Z7.Ammonia.mg.N.L, probs = c(0.25, 0.5, 0.75))

if (last_ammonia_Jan1 < ammonia_quantiles[1]) {
  print("Ammonia level is within the first quantile.")
} else if (last_ammonia_Jan1 < ammonia_quantiles[2]) {
  print("Ammonia level is within the second quantile.")
} else if (last_ammonia_Jan1 < ammonia_quantiles[3]) {
  print("Ammonia level is within the third quantile.")
} else {
  print("Ammonia level is within the fourth quantile.")
}
```

```
## [1] "Ammonia level is within the first quantile."
```

4. As we have discussed in class, filtering of the data is one of the many activities we will be called upon to perform when working with data. One way that we can accomplish this would be to use loops and conditionals to pull out the data that we find relevant and store it in a new data frame. In order to accomplish this, we would use the 'rbind()' command to add rows that fit our parameters to a new data frame.

```
rbind(<old existing object>, <new object to be added>)
```

To accomplish this we must create a new data frame with the same number of variables/columns and use the rbind() command to add a new observation. For this question, you are to create a new data frame that contains all of the observations starting at the beginning and ending when the ammonia level exceeds 12. Fill in the following code chunk to perform this operation. How many observations are in this dataset?

```
low_ammonia <- boulder_ammonia[1,]

i <- 2
while(boulder_ammonia$AB3.Z7.Ammonia.mg.N.L[i] < 12) {
  low_ammonia <- rbind(low_ammonia, boulder_ammonia[i,])
  i <- i + 1
}

nrow(low_ammonia)
```

5. Loops are useful in and of themselves, and they can be combined to work with intricate data. As a thought exercise, create a 6x4 matrix of integer values, as shown below.

```
# Create a matrix
mat <- matrix(1:24, nrow = 6, ncol = 4)
```

Then create a *for loop* that will cycle through the values and print out the value of each cell in the following format:

```
## [1] "Row 1 and column 1 has a value of 10"
```

To print to the console a combination of variables and string literals, use the 'paste()' command in conjunction with 'print()'. Here is an example of the syntax.

```
print(paste("Hello, my name is", name)
```

```
for(i in 1:6) {
  for(j in 1:4) {
    print(paste("Row", i, "and column", j, "has a value of", mat[i,j]))
  }
}
```

```
## [1] "Row 1 and column 1 has a value of 1"
## [1] "Row 1 and column 2 has a value of 7"
## [1] "Row 1 and column 3 has a value of 13"
## [1] "Row 1 and column 4 has a value of 19"
## [1] "Row 2 and column 1 has a value of 2"
## [1] "Row 2 and column 2 has a value of 8"
## [1] "Row 2 and column 3 has a value of 14"
## [1] "Row 2 and column 4 has a value of 20"
## [1] "Row 3 and column 1 has a value of 3"
## [1] "Row 3 and column 2 has a value of 9"
## [1] "Row 3 and column 3 has a value of 15"
## [1] "Row 3 and column 4 has a value of 21"
## [1] "Row 4 and column 1 has a value of 4"
## [1] "Row 4 and column 2 has a value of 10"
## [1] "Row 4 and column 3 has a value of 16"
## [1] "Row 4 and column 4 has a value of 22"
## [1] "Row 5 and column 1 has a value of 5"
## [1] "Row 5 and column 2 has a value of 11"
## [1] "Row 5 and column 3 has a value of 17"
## [1] "Row 5 and column 4 has a value of 23"
## [1] "Row 6 and column 1 has a value of 6"
## [1] "Row 6 and column 2 has a value of 12"
## [1] "Row 6 and column 3 has a value of 18"
## [1] "Row 6 and column 4 has a value of 24"
```

6. Now write a `for` loop that sums up the values for each of the rows, and prints the total for each row, as follows:

```
## [1] "The total for row 1 is 40"
## [1] "The total for row 2 is 44"
```

```
## [1] "The total for row 3 is 48"
## [1] "The total for row 4 is 52"
## [1] "The total for row 5 is 56"
## [1] "The total for row 6 is 60"

## [1] "The number of even sums is 6"

## [1] "The number of odd sums is 0"
```

Additionally, count the number of even and odd row totals, and output the results after the loop.