Lab #2 Activities

Thanh Dat Nguyen - THAND2301

Instructions:

- Fill in your name in line 3.
- Knit to pdf to read the questions in a more readable format.
- Fill in the code chunks below and answer the questions with text responses. It is recommended that you knit to pdf after you fill in each code chunk. Be sure when adding in text responses to never copy-paste symbols from outside of the document.
- Your responses must use code that was covered in class; other methods to solve the problems will not be accepted.
- Submit your knit pdf file to Crowdmark.

A reminder that the R code we have covered in class is available on our STAT 2150 A01 UM Learn page, under Content > Course Material.

Your knit pdf file should show the result answering each question. To do this, after creating an R object, you should also print it in a new line within the code chunk.

Introduction to Questions 1 and 2:

We have seen the `which()` function used to find a subset of a vector that satisfies some criteria. In the same way, `which()` can be used to find a subset of a data frame that satisfies some criteria. Consider, for example, the data frame `df` created below

```
students = 1:6
grades = rep(c("A","B","C"),2)
df = data.frame(students,grades)
df
```

```
##   students grades
## 1        1      A
## 2        2      B
## 3        3      C
## 4        4      A
## 5        5      B
## 6        6      C
```

We can use the `which()` function to extract rows from `df` that satisfy some criteria – for example, all the students with a grade of B. The output will be a new data frame with only the desired rows. In order to do this, we access the `grades` column, find *which* indices of this column (that is, which rows) have a B grade, and then we go back to the whole data frame, keep the identified rows, and store the smaller data frame in a new object:

```
Bstudents = df[which(df$grades == "B"),] # nothing after comma, which means keep all columns
Bstudents
```

```
##   students grades
## 2        2      B
## 5        5      B
```

Question 1:

There is a built-in dataset in R called `airquality`. You can see the documentation on the dataset by typing `?airquality` at the Console. The dataset contains daily readings of air quality values in New York over a 5-month span in 1973.

Create a new data frame with all of the variables of the original dataset, but only for days when the ozone reading was greater than 120. Print the new data frame.

```
# Question 1
# Create a new data frame with ozone readings greater than 120
?airquality
```

```
## starting httpd help server ... done
```

```
high_ozone_df <- airquality[which(airquality$Ozone > 120) ,]
high_ozone_df
```

```
##     Ozone Solar.R Wind Temp Month Day
## 62    135     269  4.1   84     7   1
## 99    122     255  4.0   89     8   7
## 117   168     238  3.4   81     8  25
```

The high ozone readings above 120 occurred in which months?

The high ozone readings above 120 occurred in July and August

Create and print a new data frame that sorts the original data frame with the lowest ozone readings at the top and the highest ozone readings at the bottom.

```
# Create and print a new data frame that sorts the original data frame with the lowest ozone readings a
# and the highest ozone readings at the bottom
sorted_df <- airquality[order(airquality$Ozone), ]
print(sorted_df)
```

```
##     Ozone Solar.R Wind Temp Month Day
## 21      1       8  9.7   59     5  21
## 23      4      25  9.7   61     5  23
## 18      6      78 18.4   57     5  18
## 11      7      NA  6.9   74     5  11
## 76      7      48 14.3   80     7  15
## 147     7      49 10.3   69     9  24
## 9       8      19 20.1   61     5   9
## 94      9      24 13.8   81     8   2
## 114     9      36 14.3   72     8  22
## 137     9      24 10.9   71     9  14
## 73     10     264 14.3   73     7  12
## 13     11     290  9.2   66     5  13
## 20     11      44  9.7   62     5  20
## 22     11     320 16.6   73     5  22
```

```
## 3      12     149 12.6   74     5   3
## 50     12     120 11.5   73     6  19
## 51     13     137 10.3   76     6  20
## 138    13     112 11.5   71     9  15
## 141    13      27 10.3   76     9  18
## 144    13     238 12.6   64     9  21
## 14     14     274 10.9   68     5  14
## 16     14     334 11.5   64     5  16
## 148    14      20 16.6   63     9  25
## 151    14     191 14.3   75     9  28
## 12     16     256  9.7   69     5  12
## 82     16       7  6.9   74     7  21
## 95     16      77  7.4   82     8   3
## 143    16     201  8.0   82     9  20
## 4      18     313 11.5   62     5   4
## 15     18      65 13.2   58     5  15
## 140    18     224 13.8   67     9  17
## 152    18     131  8.0   76     9  29
## 8      19      99 13.8   59     5   8
## 49     20      37  9.2   65     6  18
## 87     20      81  8.6   82     7  26
## 130    20     252 10.9   80     9   7
## 153    20     223 11.5   68     9  30
## 47     21     191 14.9   77     6  16
## 113    21     259 15.5   77     8  21
## 132    21     230 10.9   75     9   9
## 135    21     259 15.5   76     9  12
## 108    22      71 10.3   77     8  16
## 7      23     299  8.6   65     5   7
## 28     23      13 12.0   67     5  28
## 44     23     148  8.0   82     6  13
## 110    23     115  7.4   76     8  18
## 131    23     220 10.3   78     9   8
## 145    23      14  9.2   71     9  22
## 133    24     259  9.7   73     9  10
## 142    24     238 10.3   68     9  19
## 74     27     175 14.9   81     7  13
## 6      28      NA 14.9   66     5   6
## 105    28     273 11.5   82     8  13
## 136    28     238  6.3   77     9  13
## 38     29     127  9.7   82     6   7
## 19     30     322 11.5   68     5  19
## 149    30     193  6.9   70     9  26
## 111    31     244 10.9   78     8  19
## 24     32      92 12.0   61     5  24
## 64     32     236  9.2   81     7   3
## 129    32      92 15.5   84     9   6
## 17     34     307 12.0   66     5  17
## 78     35     274 10.3   82     7  17
## 97     35      NA  7.4   85     8   5
## 2      36     118  8.0   72     5   2
## 146    36     139 10.3   81     9  23
## 31     37     279  7.4   76     5  31
## 48     37     284 20.7   72     6  17
```

```
## 41      39     323 11.5   87     6  10
## 93      39      83  6.9   81     8   1
## 67      40     314 10.9   83     7   6
## 1       41     190  7.4   67     5   1
## 104     44     192 11.5   86     8  12
## 112     44     190 10.3   78     8  20
## 134     44     236 14.9   81     9  11
## 29      45     252 14.9   81     5  29
## 116     45     212  9.7   79     8  24
## 139     46     237  6.9   78     9  16
## 128     47      95  7.4   87     9   5
## 77      48     260  6.9   81     7  16
## 63      49     248  9.2   85     7   2
## 90      50     275  7.4   86     7  29
## 88      52      82 12.0   86     7  27
## 92      59     254  9.2   81     7  31
## 109     59      51  6.3   79     8  17
## 79      61     285  6.3   84     7  18
## 81      63     220 11.5   85     7  20
## 66      64     175  4.6   83     7   5
## 91      64     253  7.4   83     7  30
## 106     65     157  9.7   80     8  14
## 98      66      NA  4.6   87     8   6
## 40      71     291 13.8   90     6   9
## 118     73     215  8.0   86     8  26
## 126     73     183  2.8   93     9   3
## 120     76     203  9.7   97     8  28
## 68      77     276  5.1   88     7   7
## 96      78      NA  6.9   86     8   4
## 125     78     197  5.1   92     9   2
## 80      79     187  5.1   87     7  19
## 85      80     294  8.6   86     7  24
## 89      82     213  7.4   88     7  28
## 122     84     237  6.3   96     8  30
## 71      85     175  7.4   89     7  10
## 123     85     188  6.3   94     8  31
## 100     89     229 10.3   90     8   8
## 127     91     189  4.6   93     9   4
## 124     96     167  6.9   91     9   1
## 69      97     267  6.3   92     7   8
## 70      97     272  5.7   92     7   9
## 86     108     223  8.0   85     7  25
## 101    110     207  8.0   90     8   9
## 30     115     223  5.7   79     5  30
## 121    118     225  2.3   94     8  29
## 99     122     255  4.0   89     8   7
## 62     135     269  4.1   84     7   1
## 117    168     238  3.4   81     8  25
## 5       NA      NA 14.3   56     5   5
## 10      NA     194  8.6   69     5  10
## 25      NA      66 16.6   57     5  25
## 26      NA     266 14.9   58     5  26
## 27      NA      NA  8.0   57     5  27
## 32      NA     286  8.6   78     6   1
```

```
## 33     NA     287  9.7   74     6   2
## 34     NA     242 16.1   67     6   3
## 35     NA     186  9.2   84     6   4
## 36     NA     220  8.6   85     6   5
## 37     NA     264 14.3   79     6   6
## 39     NA     273  6.9   87     6   8
## 42     NA     259 10.9   93     6  11
## 43     NA     250  9.2   92     6  12
## 45     NA     332 13.8   80     6  14
## 46     NA     322 11.5   79     6  15
## 52     NA     150  6.3   77     6  21
## 53     NA      59  1.7   76     6  22
## 54     NA      91  4.6   76     6  23
## 55     NA     250  6.3   76     6  24
## 56     NA     135  8.0   75     6  25
## 57     NA     127  8.0   78     6  26
## 58     NA      47 10.3   73     6  27
## 59     NA      98 11.5   80     6  28
## 60     NA      31 14.9   77     6  29
## 61     NA     138  8.0   83     6  30
## 65     NA     101 10.9   84     7   4
## 72     NA     139  8.6   82     7  11
## 75     NA     291 14.9   91     7  14
## 83     NA     258  9.7   81     7  22
## 84     NA     295 11.5   82     7  23
## 102    NA     222  8.6   92     8  10
## 103    NA     137 11.5   86     8  11
## 107    NA      64 11.5   79     8  15
## 115    NA     255 12.6   75     8  23
## 119    NA     153  5.7   88     8  27
## 150    NA     145 13.2   77     9  27
```

Look at the bottom of the ozone column of the sorted data frame. The "NAs" indicate that ozone readings were not collected on some days. We can create a new data frame where missing values of any variable are removed:

```r
newdf = na.omit(airquality)
```

Write the R code that determines the number of rows remaining in the `newdf` data frame.

```r
nrow(newdf)
```

```
## [1] 111
```

Question 2:

There are five months of data in the `airquality` dataset, for May, June, July, August, and September (coded as 5, 6, 7, 8, and 9). Create five new data frames with all of the data for each of these five months. (Refer to the example in the introduction to this lab.) However, do not split up the full `airquality` dataset. Instead, split up the `newdf` dataset created above which removes missing values. After creating your five data frames, do **not** print them.

```r
# Question 2
# Create five new data frames with all of the data for each of the five months
may_df <- newdf[which(newdf$Month == 5), ]
june_df <- newdf[which(newdf$Month == 6), ]
july_df <- newdf[which(newdf$Month == 7), ]
august_df <- newdf[which(newdf$Month == 8), ]
september_df <- newdf[which(newdf$Month == 9), ]
```

Access each of the five data frames you have created, and calculate the mean solar radiation for each of the five months.

```r
# Access each of the five data frames you have created, and calculate the mean solar radiation for each
mean_solar_radiation <- c(mean(may_df$Solar), mean(june_df$Solar), mean(july_df$Solar), mean(august_df$S
mean_solar_radiation
```

```
## [1] 182.0417 184.2222 216.4231 173.0870 168.2069
```

Which month has the lowest average solar radiation?

September is the lowest average solar radiation

Question 3:

(a) Generate a vector of 1000 random numbers from a normal distribution with mean 6 and standard deviation 2. Store that vector as `x`. (Note that you cannot use commas in R to represent 1,000; instead, simply write 1000.)

```r
# Question 3
# Generate a vector of 1000 random numbers from a normal distribution with mean 6 and standard deviatio
x <- rnorm(1000, mean = 6, sd = 2)
```

(b) Calculate the proportion of numbers in `x` that are greater than 9. (Each time you knit, you will get a different answer because a new set of 1000 values will be generated.)

```r
prop_greater_than_9 <- length(x[x > 9]) / length(x)
prop_greater_than_9
```

```
## [1] 0.074
```

Question 4:

If a student is familiar with rules of logarithms and exponents, they will see that this expression:

$$\ln\left(e^3 \cdot \sqrt{e^5}\right)$$

can be simplified to simply 5.5, because:

$$
\begin{aligned}
\ln\left(e^3 \cdot \sqrt{e^5}\right) &= \ln\left(e^3 \cdot (e^5)^{(0.5)}\right) \\
&\text{since } \sqrt{x} = x^{0.5} \\
&= \ln(e^3 \cdot e^{2.5}) \\
&\text{since } (e^a)^b = e^{a \cdot b} \\
&= \ln(e^{5.5}) \\
&\text{since } e^a \cdot e^b = e^{a+b} \\
&= 5.5 \cdot \ln(e) \\
&\text{since } \ln(e^x) = x \cdot \ln(e) \\
&= 5.5 \\
&\text{since } \ln(e) = 1
\end{aligned}
$$

Suppose you do not remember all of those logarithm and exponent properties and your professor asks you to simplify the given expression. Write the R code that will code the expression, so that when you knit to pdf, you will see the simplified answer of 5.5.

```r
# Question 4
# Simplify the given expression using R
simplified_expression <- log(exp(3) * sqrt(exp(5)))
simplified_expression
```

```
## [1] 5.5
```

You will be expected to remember these logarithm and exponent properties when we get to Unit 4 of this course!