Jared Yu

HW 1

STA 141C

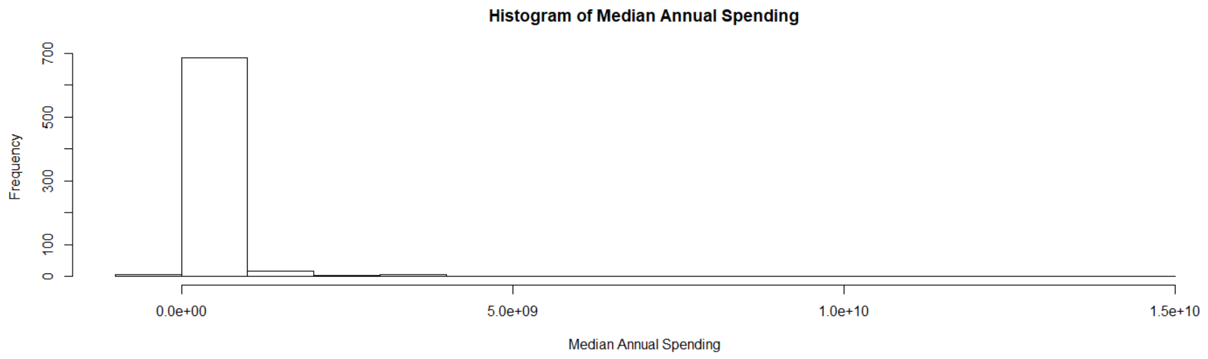
Computation

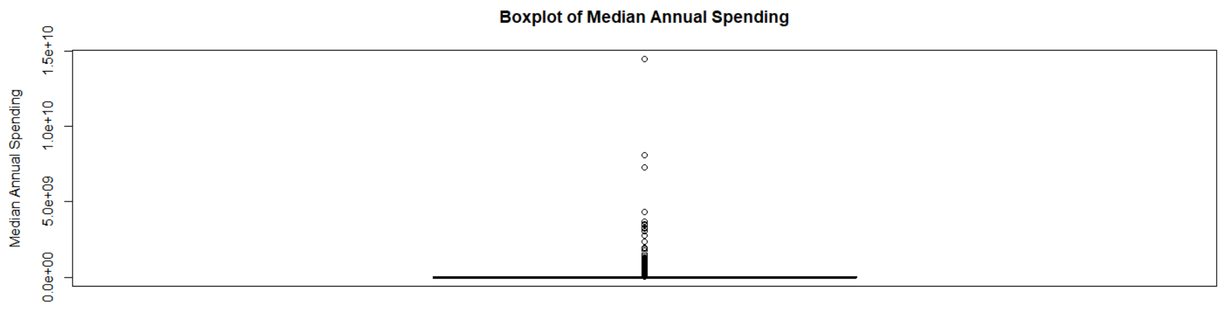
1. After using the unz() function to get the relevant column from each of the files, the median function and sum function were used to determine the annual median data. The top 5 by median annual spending are:

Funding Agency Name	Median Annual Spending	Agency ID
Defense Security Cooperation Agency	14,495,443,353	1219
Headquarters, NASA	8,107,436,839	862
Office of the Secretary	7,256,608,663	930
Department of State	4,326,045,181	315
Defense Human Resources Activity	3,668,429,160	1205

There are many NA's however, so the calculations are not exact. In total, there are 488,241 rows that are NA for either the spending, date, or both. Additionally, there are 9,141 values for which the date is beyond 2018, making their truthfulness questionable. It is possible that they are future expenses which are already accounted for, since over 8,000 are in the year 2019. Since they have either not occurred yet or have errors, their spending values have been set to NA.

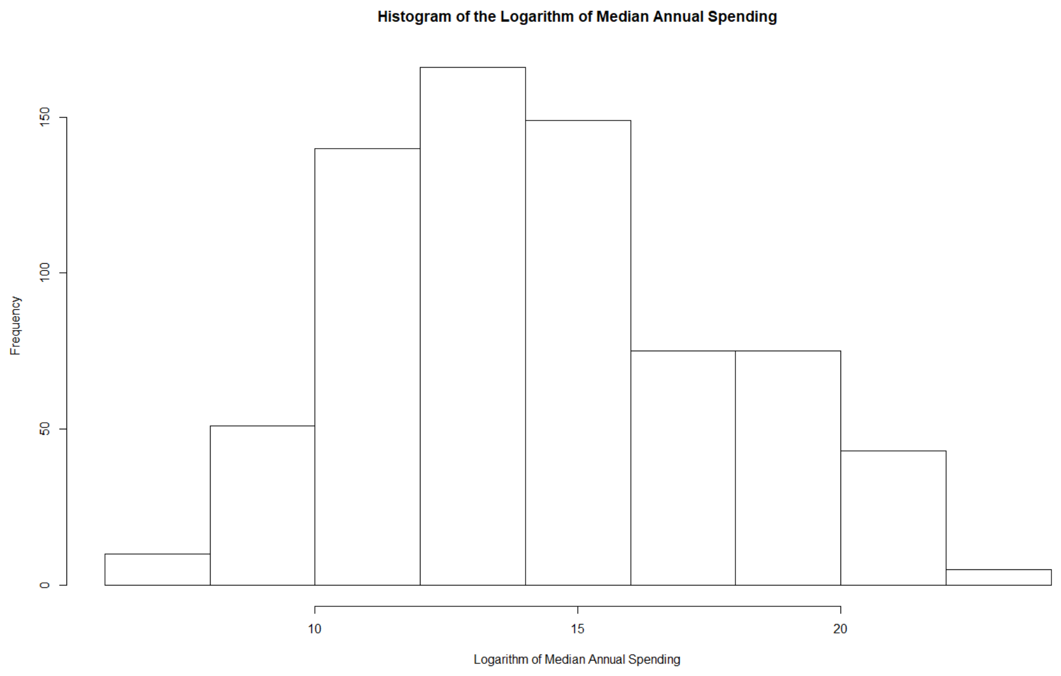
2. Below is a histogram and boxplot of the median annual spending:





There is a strong skew to the right, where there seems to be only a few outliers. The majority of all the observations exist within a certain range. To better understand the distribution, it would make sense to take the logarithm to get a better idea of what the distribution appears like, to account for the extreme outliers. This will be done below. A boxplot also shows that there is a concentration of points at the lower end, while there are fewer and fewer points as spending increases, with several extreme outliers. It is difficult to explain clearly the spending until the log transformation is done below.

3. Below is a histogram of the logarithm of the median annual spending:

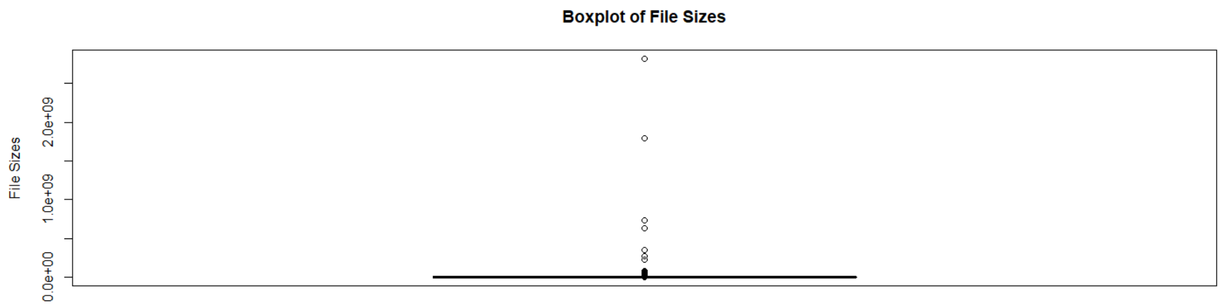


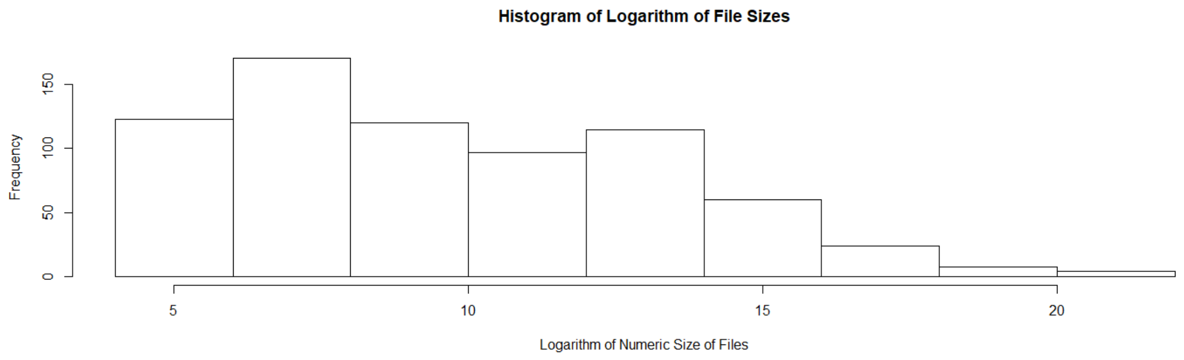
After using the log function on the data, the data now much more closely follows a normal distribution with the typical bell curve. There is still somewhat of a skew towards the right, while the data remains quite unimodal. This seems to imply that most of the spending is towards the lower end, while the frequency tends to decrease as spending increases.

4. After taking the logarithm of the median annual spending, it is quite clear that there is only a unimodal distribution, with no obvious difference in the spending whether it is considered high or low. If there were a significant difference between high or low spending agencies, it would make sense that there would be a bimodal distribution. In that case lower spending would be balanced around the left, and high spending would balance towards the right. However, it is also possible to note that there are some considerable outliers, and it is possible to say that the gap between them and the rest of the data exists at some arbitrary limit.

Reflecting

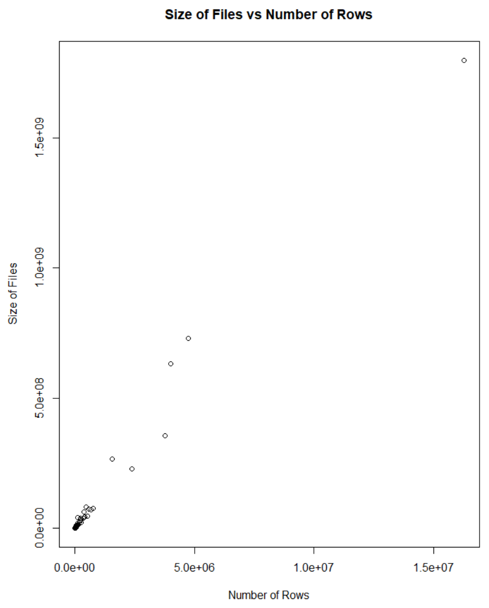
1. Below are a boxplot and a log histogram of the file sizes:

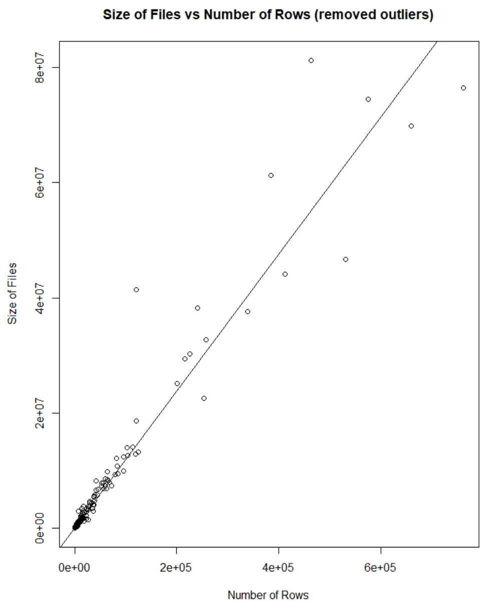




The first plot is a boxplot showing how there are a few outliers that create an extreme skew to the appearance of the distribution of the data. To get a better understanding of how the file sizes are distributed, the log of the data is taken and plotted in a histogram. Using the logarithm of the data, the right skew still is present, but it is much less extreme. It seems that most of the file sizes are on the smaller end, with fewer as the file sizes get progressively larger.

2. Below are two scatter plots of the size of the files against the number of rows:





The plot on the left shows the original scatter plot of file size against number of rows. The plot on the left removes 6 of the largest values which are considered outliers for the purpose of visualization. A regression line is drawn through the second plot to show the linear relationship between the two values. It is apparent that as the number of rows increases, the size of the files will also increase accordingly. However, there seems to be a larger variance in the file size as the number of rows increase. It is possible that within a certain range, files are kept orderly and complete without any empty rows. When the number of rows become large, certain values may possibly be left out, or extra values without much meaning are placed inside. This could possibly explain the variation.

3. The portion of the code that spent the most time processing data in the first half was the lapply statement which was used to read all the files from the zip to a list. This portion took 236.46 seconds, or roughly 4 minutes according to the system.time() function. A trick was to read only certain columns such as spending, name, date, and ID. This reduced the overall size of the files.

   The second part which calculates the number of rows from each file took the longest overall. It took 363.09 seconds, or roughly 6 minutes.

   The other sections of code took considerably less time, with none of them exceeding a minute in duration. The code that turned the list into a large dataframe took less than 15 seconds and he code that performs the query to find the median annual spending took less than 3 seconds.

4. According to a post on StackOverflow, the max number of rows in a dataframe is 2^31. The total number of rows in the dataset is 41,890,074. If this was multiplied by 10, it would still be under 2^31, or 2,147,483,648. So, in terms of working with the data structures within RStudio, there shouldn't be an issue even if there were 10 times as many files.

   The problem would come if the size of the files increased by 10 times. The largest file size multiplied by 10 would come out to 17.98441 GB, this is larger than the 16 GB of RAM available on the laptop. When the laptop is processing files which exceed the threshold, the memory leak would occur, regardless of whether or not the program utilizes unz() or unzip().

5. To significantly increase the speed at which so many files are processed, it may be necessary to use techniques such as parallel computing. Using this type of model, it would be possible to break up the task of computation into parts for different processors or servers to handle. This can be done using multiple graphics cards with their enhanced GPU speed to accomplish this task. This could be like using MapReduce with parallel computing in large clusters.

   Another possibility is to query the data more directly using SQL. Through this method it would be possible to ignore the NA's and other unnecessary columns directly. Then the total downloaded file would be much smaller than the awards.zip that we worked with. However, if there was such a connection to the database, it would have been possible to determine the median annual spending directly.

References:

https://stackoverflow.com/questions/5788117/only-read-limited-number-of-columns

https://stackoverflow.com/questions/1740524/count-number-of-objects-in-list

https://www.statmethods.net/management/sorting.html

https://github.com/clarkfitzg/sta141c-winter19/blob/master/lecture/01-10-group_by.md

https://www.techopedia.com/definition/8777/parallel-computing

https://stackoverflow.com/questions/5233769/practical-limits-of-r-data-frame

https://stackoverflow.com/questions/23389547/include-code-that-does-not-run-in-rpresentation-markdown

http://uc-r.github.io/lists_subsetting

https://piazza.com/class/jqmje0ujwrm2wx?cid=24

https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

https://www.medcalc.org/manual/log_transformation.php