PS1

Joanne Wu
3033096627

September 7, 2017

Problem 1

This is the class survey.

Problem 2

The first part of this problem is downloading the data. After the data is downloaded, the `file` command tells me that what I have downloaded is not a CSV file, so I will have to unzip that using the `unzip` command.

```
curl -o 'apricotsdata' ”http://data.un.org/Handlers/DownloadHandler.ashx?
    DataFilter=itemCode:526&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=
    countryName:asc,elementCode:asc,year:desc”
file apricotsdata
unzip apricotsdata
filename=$(echo $(unzip apricotsdata) | cut -d ' ' -f 4)
mv ${filename} apricots.csv
```

2a.

To extract the country-level and region-level data into new files, I utilized the fact that regions in the file were denoted with a '+' while countries were not. The main problem that I had with this problem was that the quotation marks were messing up the sorting, but that was solved using the `sed` command.

```
grep '+' apricots.csv >> RegionLevelData.csv
grep -v '+' apricots.csv  >> CountryLevelData.csv

grep 2005 CountryLevelData.csv | grep 'Area Harvested' | sed 's/\”//g' |
    sort -r -n -t ',' -k 6 | head -n 5
```

This took the lines with 2005 and ”Area Harvested” in them and took out the quotation marks so that I could sort numerically.

```
for y in 1965 1975 1985 1995 2005 ; do echo ”${y}: $(grep ${y}
    CountryLevelData.csv | grep 'Area Harvested' | sed 's/\”//g' | sort -r
     -n -t ',' -k 6 | head -n 5 | cut -d',' -f1)”; done
```

This took each individual year and ran the same process as we did for the 2005 year. In conclusion, most of the top 5 have remained at the top until 2005. In that year, Pakistan, Uzbekistan, and Algeria beat out Ukraine, Tunisia, and Spain, who have been in the top 5 in the previous decades.

2b.

```
function myfun {
    if [ $# != 1 ];
        then
            echo "Error: Please input only one code." ;
    elif [ "$1" == "-h" ];
        then
            echo "Enter the code for the crop whose data you would like to
                view. E.g. enter 526 for apricots , or enter 572 for
                avocados.";
    else
        $(curl -o $1 "http://data.un.org/Handlers/DownloadHandler.ashx?
            DataFilter=itemCode:{$1}&DataMartId=FAO&Format=csv&c
            =2,3,4,5,6,7&s=countryName:asc,elementCode:asc,year:desc" ;
        unzip $1;
        filename=$(echo $(unzip $1) | cut -d ' ' -f 4);
        mv ${filename} $1.csv)
        head $1.csv;
    fi;
}
```

This code first checks that there is only one argument given; otherwise it will show an error message. It then checks if the user entered -h as an argument, which prompts the help dialogue. Then, if neither of the two occur, the function takes the argument and uses curl to download the data, which is then unzipped and renamed. Finally, head is used to show the first couple lines of the CSV file.
There was a slight problem that I encountered while I was creating the function: when I call the function, the shell eventually will prompt me if I want to replace the unzipped file, and then after inputting 'y' will return -bash: Archive:: command not found but still rename the file and show me the first few lines.

Problem 3

This problem asked to automatically download all the .txt files from the website. I started off by using curl to see what kind of files were on the site, grep and [:graph:] to take only the .txt files, and then cut twice to find the names of the files. The use of [:graph:] made sure that I included the hyphens in the file names. I then put them all into a variable named textFiles.

```
textFiles=$(curl https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ | grep [:
    graph:]*.txt | cut -d '>' -f 7 | cut -d '<' -f 1)
```

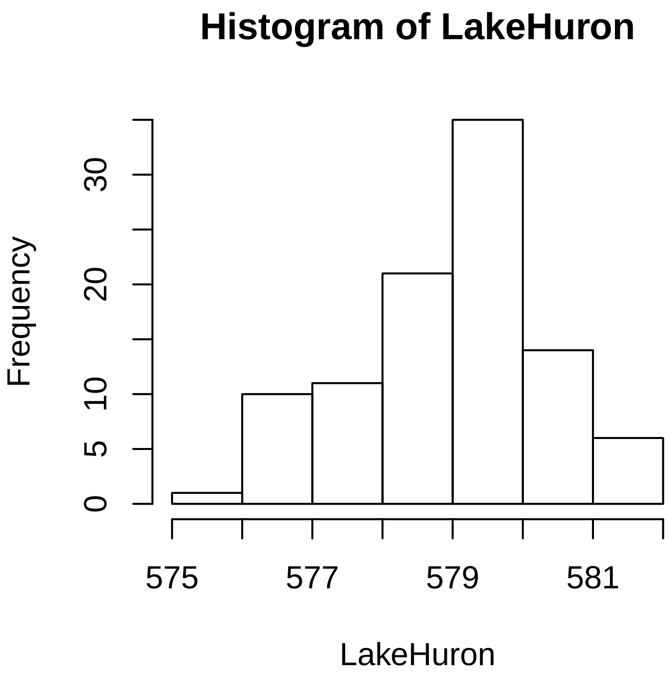
When I have all the file names that I want to download, all I have to do is run a for-loop that echoes what I want to start downloading, curl using the file name, and then echo when my file is done downloading.

```
for t in ${textFiles} ; do
    echo "Download for ${t} is starting...";
    curl -o ${t} https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/${t};
    echo "Download complete. File downloaded: ${t}";
done
```

Problem 4

The height of the water level in Lake Huron fluctuates over time. Here I 'analyze' the variation using R. I show a histogram of the lake levels for the period 1875 to 1972.

```
library(knitr)
hist(LakeHuron)
```



```
lowHi <- c(which.min(LakeHuron), which.max(LakeHuron))
yearExtrema <- attributes(LakeHuron)$tsp[1]-1 + lowHi
```

This output was produced using the following code:
The height of the water level in Lake Huron fluctuates over time. Here I 'analyze' the variation using R. I show a histogram of the lake levels for the period \rinline{start(time(LakeHuron))[1]} to \rinline{end(time(LakeHuron))[1]}.

```
# %% begin.rcode, fig.height = 4, fig.width = 4
# % library(knitr)
# % hist(LakeHuron)
# % lowHi <- c(which.min(LakeHuron), which.max(LakeHuron))
# % yearExtrema <- attributes(LakeHuron)$tsp[1]-1 + lowHi
# %% end.rcode
```