STAT243 Problem Set1

Name: Chih Hui Wang SID: 26955255

September 5, 2015

1. (a) To solve this problem, I first downloaded the file and unziped it. Before dividing the data into country and region, I have to observe whether there is any pattern I can use to extract data. Then, I subseted the data by two term "2005" and "Area Harvested" and sort by the value column to get the top five countries. Finally, I generalized the above process and wrote a function that can automatic pritn out the top five countries, once providing a year.

Use **curl** to download the file and modifier **-o** to rename the file. Then, rename it as apricots.zip, otherwise the name of file will be to long to see. Then, **unzip** the file (use the **-c** to output the file and put it into **apricots.csv**.)

```
#Download file
curl -s "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" -osS apricots.zip

#Unzip
unzip -c apricots.zip > apricots.csv

unzip:  cannot find or open apricots.zip, apricots.zip.zip or apricots.zip.ZIP.
```

I noticed that there is a "+" in the end of first column for the region. Therefore, I can use **grep** to pull out those line with "+" as well as without "+" with **-v** in the command. The country-level data will have some unmeaningful row for this problem. I used **head** and remove the line.

```
#To observe data
cat apricots.csv | head -n5

#For region, there will be a "+" in the end of the firt column.
grep "+" apricots.csv > regions.csv

#The last 7 line is unrelated, so I remove them.
grep -v "+" apricots.csv | head -n -7 > countries.csv
```

I subsetted the data by "2005" and "Area Harvested". When subsetting by "2005", there are also some 2005 in other columns (in a format like"2005.000"), so I use **"2005****"** to ensure that I pull out the correct line. After **grep** by **"Area Harvested**", I first observed the data to check whether some countries have different formats. I found that some countries will have , in their name, so we better use **"** to delimit data. Then, I have to sort the data by column 12 which represents value. I use the **"** to delimit the data and then sort the data from big to small **(-r)** by column 12 by the command **-k12** with **-t""** indicating that I want to use **"** to seperate each column. Finally, use **head** to get the top 5 countries and **cut** to get the first column.

```
cut -d'"' -f2 countries.csv | uniq
```

```
#2005 Area Harvested
grep \"2005\" countries.csv | grep "Area Harvested" | sort -nr -t'"' -k12 | head -n5 |
cut -d'"' -f2
```

```
#function
function rankAH() {
grep \"$1\" countries.csv | grep "Area Harvested" | sort -nr -t'"' -k12 | head -n5 |
cut -d'"' -f2
}
#1965, 1975, 1985, 1995
echo 1965
rankAH 1965
echo 1975
rankAH 1975
echo 1985
rankAH 1985
echo 1995
rankAH 1995

1965
1975
1985
1995
```

(b) The parameter of the function is the itemcode. I used **curl** the first step in (a) to download data and rename it. Then, when **unzip** the file, I set a modifier **-p** which can directly print out data and the result can be use by other operation such as **head**, **sort**.

```
function dldata() {
curl -s "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" -o $1.zip
unzip -p $1.zip
}

#Test
#first 5 rows
dldata 572 | head -n5
#sort by value
dldata 572 | head -n5 | sed 's/"//g' | sort -nr -t',' -k6
```

(c) To deal with the problem, first I have to find the table of item and its name, so I go to the website (http://faostat.fao.org/site/384/default.aspx). I used wget to get the html and then see whether any pattern I can utilize. I found that "</td><td>" can help me to pull out the item name and code. I used sed to substite the < to > and then I can easily delimit the data by >. The item code is located at 7 column and the name is located at 15 if using the > as a delimiter. Finally, I wrote a function call Itemname. Once the user input the name, then the function will use the **grep** and **cut** to get the correct itemcode and then output the data by the function **dldata** in (b).

```
#Match file preparation
curl -s "http://faostat.fao.org/site/384/default.aspx" > table.html
grep "</td><td>" table.html | sed 's/</>/g' | cut -d'>' -f7,15 > code.txt
#function
```

```
function dldataName() {
itemcode=$(grep $1 code.txt | cut -d'>' -f1)
curl -s "http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:$itemcode
&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=countryName:asc,elementCode:asc,
year:desc" -o $1.zip
unzip -p $1.zip
}
#Example
dldataName Apricots | head -n5
```

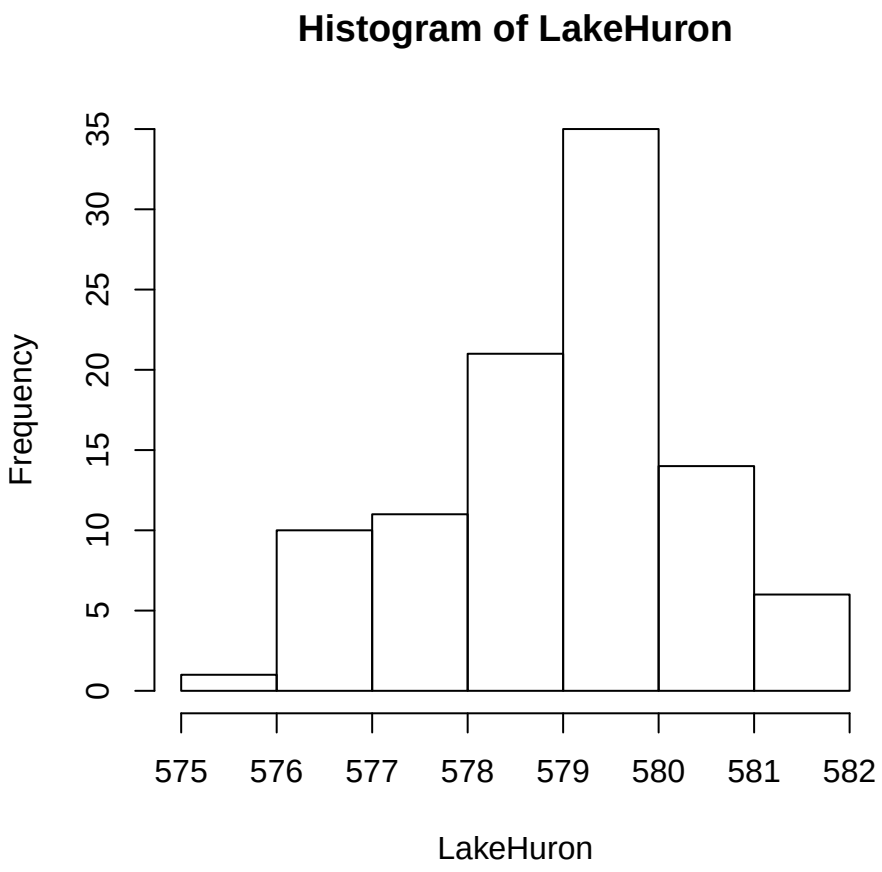
2. To address the problem, I think that I have to examine the source code of the website, find out those line with **".txt"** and then detect any pattern I can utilize to get the whole name of txt file. After that I can use a for loop to download those files. In the for loop, I can write a command to print out which file is downloading.

I use **curl** to get the html and store in a txt file, called **allhtml.txt**. When I saw the output of **grep ".txt" allhtml.txt**, I found that the file name is between two **"**. Therefore, I can use **"** to delimit the data and get out the name by command **cut**. I stored the output into a variable called **dowload**. and then write a for loop to download files as well as print out which file I am downloading now.

```
#Get html
curl -s "http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/" > allhtml.txt

#See pattern
grep ".txt" allhtml.txt | head -n1

#Make a variable
download=$(grep ".txt" allhtml.txt | cut -d'"' -f8)

#For loop
for file in $download
do  echo "Download $file now"
curl -s "http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/$file" -o $file
done

<tr><td valign="top"><img src="/icons/text.gif" alt="[TXT]"></td><td><a href="ghcnd-countries.txt">gh
Download ghcnd-countries.txt now
Download ghcnd-inventory.txt now
Download ghcnd-states.txt now
Download ghcnd-stations.txt now
Download ghcnd-version.txt now
Download readme.txt now
Download status.txt now
```

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.

```
hist(LakeHuron)
```




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

[1] 1964 1876
```