STAT243 PS1

Zhaochen Gao 3034268813

8/29/2018

Problem 3

A friend of mine is planning to get married in Death Valley National Park in March. She wants to hold it as late in March as possible but without having a high chance of a very hot day. This problem will automate the task of generating information about what day to hold the wedding on using data from https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/.

(a) Download yearly climate data for a set of years of interest into a new (temporary) subdirectory within your working directory. Do not download all the years and feel free to focus on a small number of years to reduce the amount of data you need to download. Note that data for Death Valley is only present in the last few decades. As you are processing the files, report the number of observations in each year by printing the information to the screen, including if there are no observations for that year.

```
#choose to focus on data from Year 2015 to Year 2018
for c in {2015..2018}; do
  #download the zip files from the url given and save them to the working directory
  curl -s https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/${c}.csv.gz > ${c}.csv.gz
  gunzip ${c}.csv.gz #unzip each file
  echo "Year ${c} has $(cut -d"," -f1 ${c}.csv|wc -l) oberservations"
  #count the number of observations and print out the result
done;
```

```
## Year 2015 has  35233244 oberservations
## Year 2016 has  35384539 oberservations
## Year 2017 has  34748555 oberservations
## Year 2018 has  20283714 oberservations
```

(b) Subset to the station corresponding to Death Valley, to TMAX (maximum daily temperature), and to March, and put all the data into a single file. In subsetting to Death Valley, get the information programmatically from the ghcnd-stations.txt file one level up in the website. Do NOT type in the station ID code when you retrieve the Death Valley data from the yearly files.

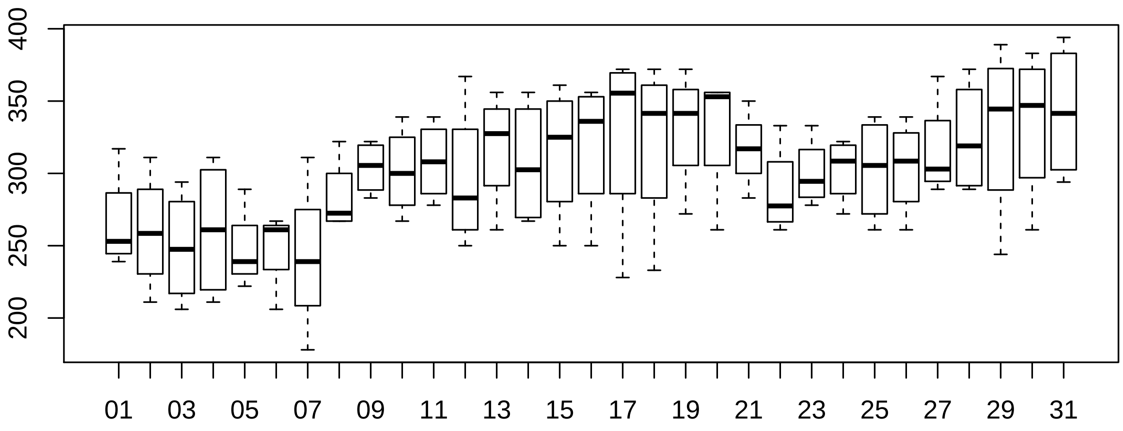
```
#download the txt file which contains station info
curl -s https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt > ghcnd-stations.txt
#find the corresponding stationID for Death Valley
DeathID=$(grep DEATH ghcnd-stations.txt | cut -d' ' -f1)

for c in {2015..2018}; do
  #find the observations in March and save the output as a temporary file
  grep "${c}03" ${c}.csv > tmp.csv
  #find the observations for TMAX(maximum daily temperature) and update the temporary file
  grep "TMAX" tmp.csv > tmp2.csv
  #find the observations for Death Valley and get the final data
  grep ${DeathID} tmp2.csv > ${c}_m.csv
  #remove the temporary files
  rm tmp.csv tmp2.csv
done;

#combine the datasets for different years together and remove the seperate datasets
cat 2015_m.csv 2016_m.csv 2017_m.csv 2018_m.csv > DeathValleyMarchTMAX.csv
rm 2015_m.csv 2016_m.csv 2017_m.csv 2018_m.csv
```

(c) Create an R chunk that makes a single plot of side-by-side boxplots containing the maximum daily temperature on each day in March.

```
#read in the generated dataset from last step
DVMT = read.csv("DeathValleyMarchTMAX.csv",header = FALSE, sep = ",")
#extract day values
DVMT$Day = substr(DVMT$V2,7,8)
#make a boxplot
boxplot(DVMT$V4~DVMT$Day,data=DVMT)
```



(d) Now generalize your code from parts (a) and (b). Write a shell function that takes as arguments a string for identifying the location, the weather variable of interest, and the time period, and returns the results. Your function should detect if the user provides the wrong number of arguments or a string that doesn't allow one to identify a single weather station and return a useful error message. It should also give useful help information if the user invokes the function as: "get_weather -h". Finally the function should remove the raw downloaded data files. Hint: to check for equality in an if statement, you generally need syntax like: if ["var" == "7"]

```
function locweather(){

  curl -s https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt > ghcnd-stations.txt
  ID=$(grep $1 ghcnd-stations.txt | cut -d' ' -f1)

  for c in {2015..2018}; do
    #download the selected years' zip files and unzip them to the working directory
    curl -s https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/${c}.csv.gz > ${c}.csv.gz
    gunzip ${c}.csv.gz
    #print out the number of oberservations to the screen
    echo "Year ${c} has $(cut -d"," -f1 ${c}.csv|wc -l) oberservations"

    #find the obeservations representing specified weather variable
    grep $2 ${c}.csv > tmp.csv

    #find the obeservations representing specified month
    if [$3 == "January"]
      then grep "${c}01" tmp.csv > tmp2.csv
    fi
    if [$3 == "Feburary"]
      then grep "${c}02" tmp.csv > tmp2.csv
    fi
    if [$3 == "March"]
      then grep "${c}03" tmp.csv > tmp2.csv
    fi
    if [$3 == "April"]
      then grep "${c}04" tmp.csv > tmp2.csv
    fi
    if [$3 == "May"]
      then grep "${c}05" tmp.csv > tmp2.csv
    fi
    if [$3 == "June"]
      then grep "${c}06" tmp.csv > tmp2.csv
    fi
    if [$3 == "July"]
      then grep "${c}07" tmp.csv > tmp2.csv
    fi
    if [$3 == "August"]
      then grep "${c}08" tmp.csv > tmp2.csv
    fi
    if [$3 == "September"]
      then grep "${c}09" tmp.csv > tmp2.csv
    fi
    if [$3 == "October"]
```

```
   then grep "${c}10" tmp.csv > tmp2.csv
  fi
  if [$3 == "November"]
   then grep "${c}11" tmp.csv > tmp2.csv
  fi
  if [$3 == "December"]
   then grep "${c}12" tmp.csv > tmp2.csv
  fi

  #find the obeservations representing specified location
  grep $ID tmp2.csv > ${c}_m.csv
  #remove temporary files
  rm tmp.csv tmp2.csv
 done;

 #combine the datasets for different years together and remove the seperate datasets
 cat 2015_m.csv 2016_m.csv 2017_m.csv 2018_m.csv > locweather.csv
 rm 2015_m.csv 2016_m.csv 2017_m.csv 2018_m.csv
}
```

Problem 4

Your task here is to automatically download all the files ending in .txt from this National Climate Data Center website: https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/. Your shell script should provide a status message to the user, telling the name of the file as it downloads each file. You should be able to use UNIX utilities to extract the individual file names from the HTML index file linked to above. Alternatively you may use tools from Unit 3, but any use of R should be done directly from the command line and should only involve a line or two of R code. Do not hard code the names of the .txt files into your code. Note that when we work on webscraping in Unit 3, we'll see more elegant ways to process HTML than we are using here, but doing it "manually" here is good for practicing with the shell commands and allows us to do quick-and-dirty processing.

```
#get the source of url and find out the names of linked txt files
url="https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/"
curl -s $url|sed -n 's/.*href="\([^"]*\).*/\1/p'|grep "txt" >filenames.csv

while IFS= read -r line
do
 echo "We are downloading ${line}." #print out the message about downloading files
  curl -s "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/$line" > ${line} #download the files
done < filenames.csv
```

```
## We are downloading ghcnd-countries.txt.
## We are downloading ghcnd-inventory.txt.
## We are downloading ghcnd-states.txt.
## We are downloading ghcnd-stations.txt.
## We are downloading ghcnd-version.txt.
## We are downloading mingle-list.txt.
## We are downloading readme.txt.
## We are downloading status.txt.
```

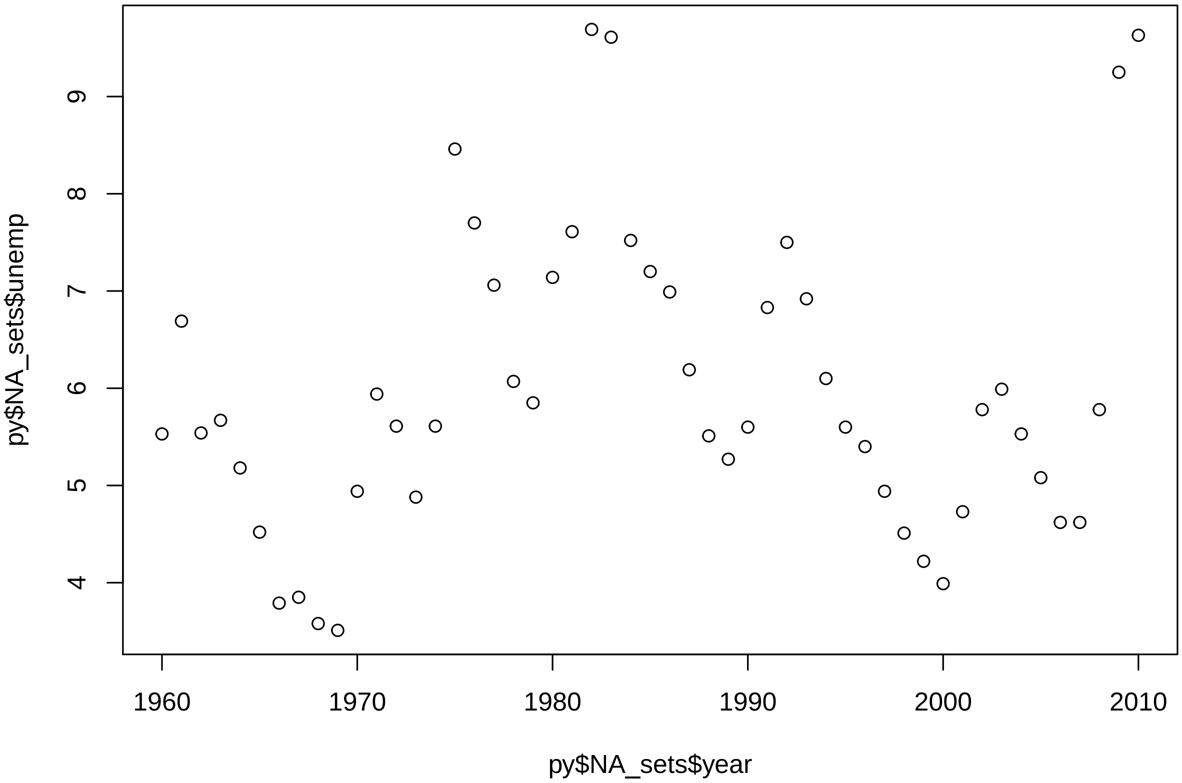
Problem 5

(b) (extra credit) The reticulate package and R Markdown allow you now to have Python and R chunks in a document that interact with each other. Demonstrate the ability to use this functionality, in particular sending data from R to Python and back to R, with some processing done in Python (it doesn't have to be complicated processing). There's a blog post here to get you started: http://feedproxy.google.com/r/RBloggers/3/76l-uQEOoiU/?utm_source=feedburner&utm_medium=email.

```
#read in a csv file
cpds = read.csv("cpds.csv",header = TRUE, sep = ",")
library(reticulate)
#direct the rstudio to the path of python
path_to_python <- "~/Users/Winnie/anaconda3/bin/python"
use_python(path_to_python)
```

```
import pandas as pd
#read in the r dataset
NA_sets = pd.DataFrame(data=r.cpds)
#keep the year and unemployment rates for USA
NA_sets = NA_sets[NA_sets['country']=='USA']
NA_sets = NA_sets[['year', 'unemp']]
#drop the missing values
NA_sets = NA_sets.dropna()
#group the data by year and take the yearly average
NA_sets = NA_sets.groupby('year').mean().reset_index()
```

```
library(ggplot2)
#create a plot of unemployment rates versus year
plot(py$NA_sets$unemp ~ py$NA_sets$year)
```


pyNA_setsunemp
4
5
6
7
8
9
1960
1970
1980
1990
2000
2010
pyNA_setsyear