Data Extraction with Bash

Matthew Seguin

1.

a.

We can see from the website that the file name of the data for each year is given by "year.csv.gz" where year is replaced for example by 2024. Similarly the link to download the file is just the website link followed by the /filename, that is https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/year.csv.gz

Here is some bash shell code for a function that takes in years of interest and downloads the gzip file for each of those years, unzips it, and prints the number of entries (number of lines) for each of the unzipped files. $@ gives all of the inputs to our function separated by spaces so we can iterate over it, then we create the filename and link, use curl with the o flag to specify where to download to, use gunzip to unzip the file, and finally use wc with the l flag to show the number of rows.

```
mkdir tmp

function GetWeatherData(){
  for i in $@; do
    # For each argument passed make the file name year.csv.gz
    filename="$i.csv.gz"
    # Construct the link by adding the filename to the end
    link="https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/$filename"
    # Download the file
    curl -s -o "tmp/$filename" $link
    # Unzip the file
    gunzip "tmp/$filename"
    # Print out the number of entries in the file
    echo "Entries: $(wc -l "tmp/$i.csv")"
  done
}
GetWeatherData 2024 2022 2020 2003 1999
```

```
Entries: 23559058 tmp/2024.csv
Entries: 37811001 tmp/2022.csv
Entries: 36866367 tmp/2020.csv
Entries: 36599201 tmp/2003.csv
Entries: 34576115 tmp/1999.csv
```

b.

First we want to download the ghcnd-stations.txt file (we are given it is one level up) and see its contents to see how to get the data for only Death Valley from our csv files. Here I use curl to download it then head to examine its structure relative to one of our csv files.

```
# Download stations information file
curl -s -o "ghcnd-stations.txt" "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt"
# Preview file
head -n 5 ghcnd-stations.txt
# Download csv information file
curl -s -o "readme-by-year.txt" "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/readme-by_ye
# Show contents
echo # Just adding space
cat "readme-by-year.txt"
# Check the format of a csv
echo # Just adding space
head -n 5 tmp/2024.csv
```

```
ACW00011604  17.1167  -61.7833   10.1    ST JOHNS COOLIDGE FLD
ACW00011647  17.1333  -61.7833   19.2    ST JOHNS
AE000041196  25.3330   55.5170   34.0    SHARJAH INTER. AIRP           GSN     41196
AEM00041194  25.2550   55.3640   10.4    DUBAI INTL                            41194
AEM00041217  24.4330   54.6510   26.8    ABU DHABI INTL                        41217

The "year".csv files contain all daily and station elements found in GHCN daily for the given year.
files are updated daily for the entire period of record.

The following information serves as a definition of each field in one line of data covering one stat
Each field described below is separated by a comma ( , ) and follows the order below:

ID = 11 character station identification code
YEAR/MONTH/DAY = 8 character date in YYYYMMDD format (e.g. 19860529 = May 29, 1986)
ELEMENT = 4 character indicator of element type
DATA VALUE = 5 character data value for ELEMENT
M-FLAG = 1 character Measurement Flag
Q-FLAG = 1 character Quality Flag
S-FLAG = 1 character Source Flag
OBS-TIME = 4-character time of observation in hour-minute format (i.e. 0700 =7:00 am)

See section III of the GHCN-Daily readme.txt file (ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/readm
for an explanation of ELEMENT codes and their units as well as the M-FLAG, Q-FLAG and S-FLAG.

The OBS-TIME field is populated with the observation times contained in NOAA/NCEI's HOMR station his


ASN00014508,20240101,TMAX,358,,,S,
ASN00014508,20240101,TMIN,273,,,S,
ASN00014508,20240101,PRCP,0,,,S,
ASN00014508,20240101,TAVG,305,H,,S,
ASN00032040,20240101,TMIN,228,,,S,
```

We can see that the Station IDs are the first 11 characters of every line in the ghcnd-stations.txt file, so we can use grep to find the line for Death Valley then subset the characters of the result for the Station ID.

Then use grep to filter our csv files for only the lines with the correct Station ID for Death Valley and for only dates in March (which will have form YEAR03DAY).

Then for each day we can see there is a row for TMAX, TMIN, PRCP, and TAVG with the value being in the 4th column. So we can use grep to find TMAX after we already filtered for everything else then take the 1st to 4th columns with cut using the deliminator flag. First I create the combined csv file (making sure its contents are empty so I don't add the same data more than once).

```
# Find the line for Death Valley
DVLine="$(grep -i "Death Valley" ghcnd-stations.txt)"
# Print result
echo $DVLine
# Get the Station ID for Death Valley
DVStationID="${DVLine::11}"
# Print result
echo $DVStationID
# Make sure our combined csv is empty
> DVData.csv

function DVDataToFile(){
  for i in $@; do
    # For each argument passed make the file name year.csv
    filename="$i.csv"
    # Search the file for the Death Valley Station ID
    grep $DVStationID "tmp/$filename" |
         # Search the file for days in March
         grep -E "$i\03[0-9][0-9]" |
              # Search the file for TMAX
              grep "TMAX" |
                   # Get the first 4 columns and append to combined file
                   cut -d "," -f1,2,3,4 >> DVData.csv
  done
}
# Call function
DVDataToFile 2024 2022 2020 2003 1999
echo # Just adding space
# Preview csv file
head DVData.csv
```

```
USC00042319 36.4625 -116.8672 -59.1 CA DEATH VALLEY NP HCN
USC00042319

USC00042319,20240301,TMAX,256
USC00042319,20240302,TMAX,244
USC00042319,20240303,TMAX,244
USC00042319,20240304,TMAX,233
USC00042319,20240305,TMAX,233
USC00042319,20240306,TMAX,250
USC00042319,20240307,TMAX,261
USC00042319,20240308,TMAX,267
```

```
USC00042319,20240309,TMAX,244
USC00042319,20240310,TMAX,228
```

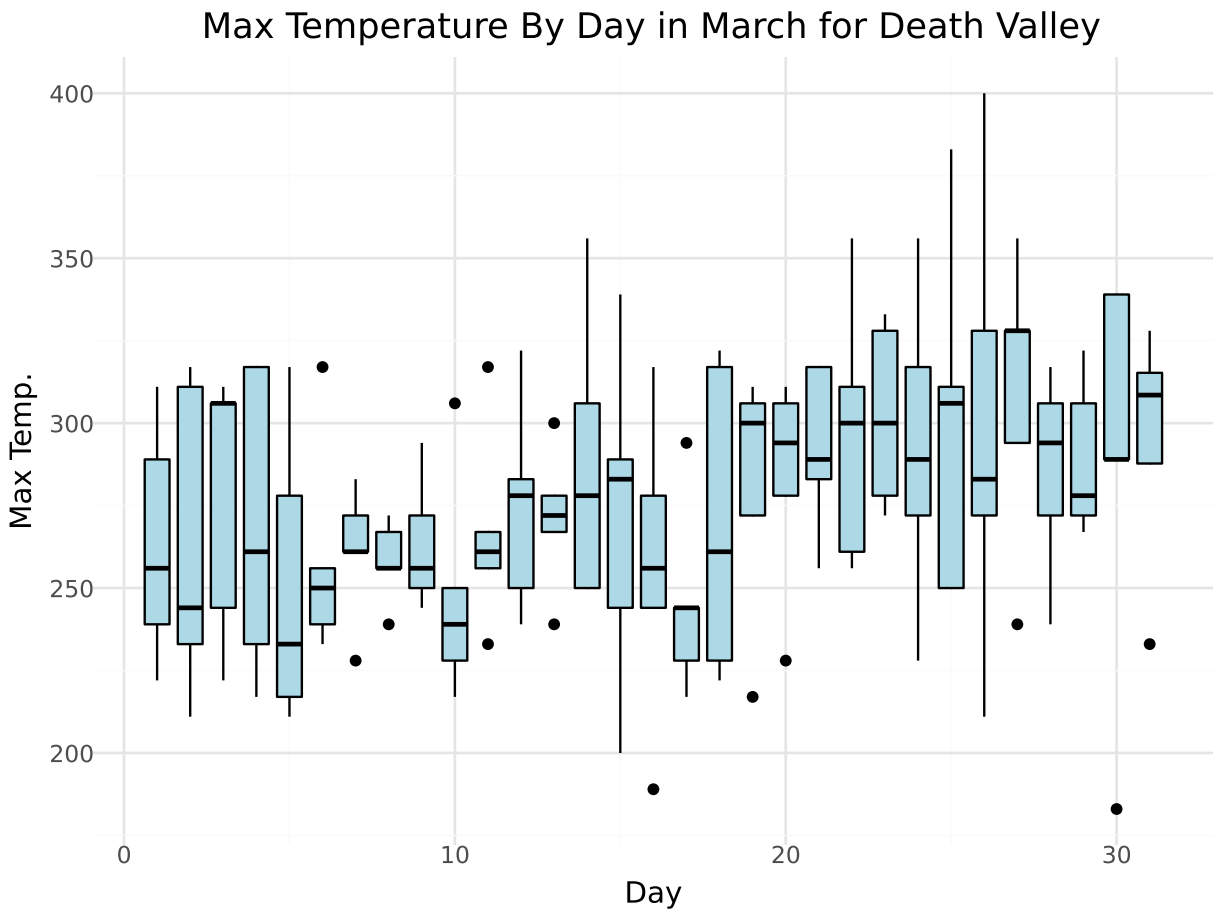
c.

I have much more experience plotting in R so I will use a package called plotnine for this which imitates ggplot from R.

```
from plotnine import *
import pandas as pd
# Read in our csv, making sure we don't use any data as column names
DVData = pd.read_csv("DVData.csv", header = None)
# Rename columns to informative names
DVData = DVData.rename(columns = {0: "Station ID",
                                  1: "Date",
                                  2: "Data Type",
                                  3: "Value"})
# Add a column for the day of the month
DVData = DVData.assign(Day = [int(str(date)[6:8]) for date in DVData["Date"]])

# Plot data in graph

(
ggplot(DVData, # Use Day as the x-axis and grouping
       aes(x = "Day",
           group = "Day",
           # Use value (temp) as y-axis
           y = "Value"
           )
       ) +
       # Create boxplot
       geom_boxplot(color = "black",
                    fill = "lightblue"
                    ) +
       # Relabel graph
       labs(title = "Max Temperature By Day in March for Death Valley",
            y = "Max Temp.") +
       # Use a simplistic theme
       theme_minimal()
).show()
```


Max Temperature By Day in March for Death Valley
Max Temp.
400
350
300
250
200
0
10
20
30
Day

d.

We want to automate parts a and b if we are given a location, weather element, years of interest, and month of interest. I actually already wrote a bash function for if we are given the years of interest for downloading and unzipping the csv files. Then I made a function for getting the combined csv as well but all I have to do is generalize it instead of for only March and Death Valley. Just to have it all in one function I will rewrite everything.

I am assuming that the order of arguments are given as: location, weather element, month of interest, year(s) of interest.

```
function get_weather(){
  # Get documentation
  curl -s -o "ghcnd-stations.txt" "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt
  # Check if user asked for help
  if [ "$1" == "-h" ]; then
  echo "
  Get weather takes in a location, weather element of interest,
  month of interest, and year(s) of interest and generates a csv file with the
  weather data for the location and month from all of those years.

  Possible locations can be found with: cat ghcnd-stations.txt
  Possible weather elements are TMIN, TMAX, TAVG, PRCP, SNOW, SNWD, TOBS
  Month should be entered as a 2 digit number, if less than 10 add a preceding 0
  Year should be 4 digits

  There should be at least 4 arguments provided
  "
  exit
  fi
  # Check if number of arguments is too small
  j=0
  for arg in $@; do
  (( j=$j+1 ))
  done

  if (( $j < 4 )); then
    echo "
    Not enough arguments provided, need at least 4. Try: get_weather -h
    "
    exit
  fi
  # Find the station line
  StationLine="$(grep -i "$1" ghcnd-stations.txt)"
  # Get the Station ID
  StationID="${StationLine::11}"
  if [ -z $StationID ]; then
    echo "
    Station not found, check that your location is spelled correctly. Try: get_weather -h
    "
    exit
  fi
```

```
  # Make sure our combined csv is empty
  combfilename="$(echo "${1}${3}${2}.csv" | sed "s/[[:blank:]]//g")"
  touch "$combfilename"
  > "$combfilename"
  for i in ${@:4}; do
    # For each argument passed make the file name year.csv.gz
    filename="$i.csv.gz"
    # Construct the link by adding the filename to the end
    link="https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/$filename"
    # Download the file
    curl -s -o "tmp/$filename" $link
    # Unzip the file
    gunzip "tmp/$filename"
    # Print out the number of entries in the file
    echo "Entries: $(wc -l "tmp/$i.csv")"
    # For each argument passed make the file name year.csv
    filename="$i.csv"
    # Search the file for the Station ID
    grep $StationID "tmp/$filename" |
         # Search the file for days in month given
         grep -E "$i$3[0-9][0-9]" |
              # Search the file for weather element
              grep -i "$2" |
                   # Get the first 4 columns and append to combined file
                   cut -d "," -f 1,2,3,4 >> "$combfilename"
    rm "tmp/$filename"
  done
}

get_weather "Arches" "TMIN" "02" 2023 2021
echo # Just adding space
head "Arches02TMIN.csv"
```

```
Entries: 37728211 tmp/2023.csv
Entries: 37809415 tmp/2021.csv

USC00420336,20230201,TMIN,-89
USC00420336,20230202,TMIN,-89
USC00420336,20230203,TMIN,-72
USC00420336,20230204,TMIN,-61
USC00420336,20230205,TMIN,-61
USC00420336,20230206,TMIN,-50
USC00420336,20230207,TMIN,-17
USC00420336,20230208,TMIN,-39
USC00420336,20230209,TMIN,-67
USC00420336,20230210,TMIN,-61
```