Katherine Olson

STA 141 Assignment Two

Step 1: For this step I combined the data from all of the files into a list of seven data frames in the format requested. I started by looping through each a list of seven containing all of the file names and sending them to a function. This function used readLines() to read each file and used strsplit() to separate the values. The trickiest part was removing the direction from lat and long and adding a negative to some with a certain direction. I found visualizing how the seven data frames would look and getting there tricky. But once I was hinted towards using rep(), it made much more sense.

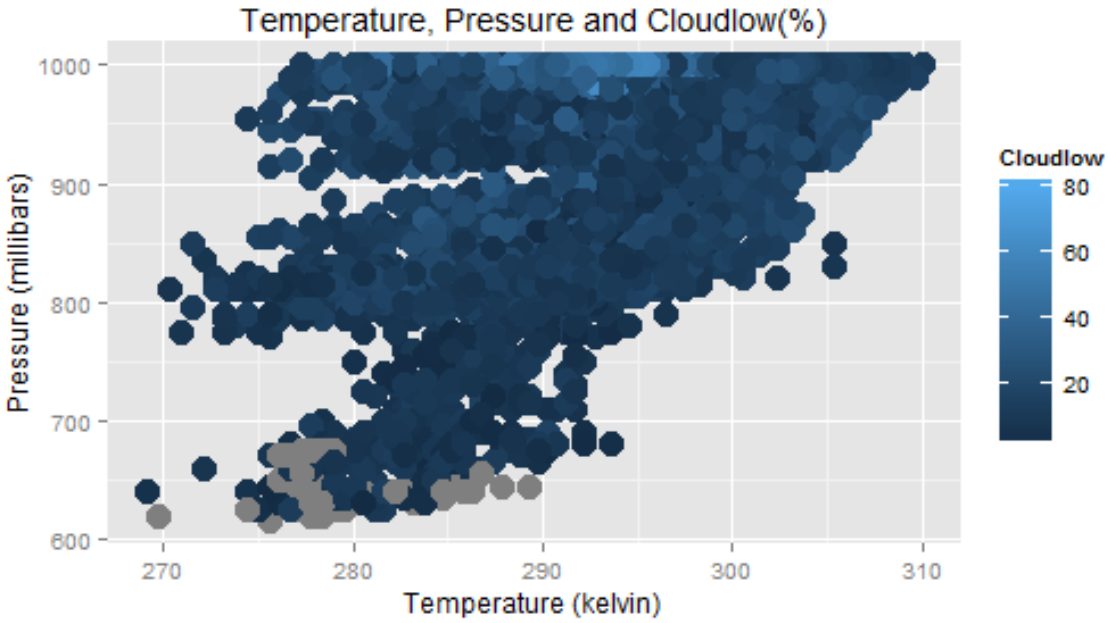
Step 2: To check that location and date were the same, I used the identical function. The results came back true for lat, long, and date, so I continued on to combine the results into one data frame.

Step 3: I struggled a lot with this step. I looped through each value of lat and long and sent this pair along with the read.table() results of the file to a function. This is where I had a hard time. I tried many different approaches that failed and ended up doing the step a strange way. I created two types of latitude, latitude + 0.5 and latitude - 0.5. I did the same for longitude. I created a function that searched through table for the four combinations of these until one was found that was not null. I noticed that this gave an output with half of the values equal to NA, alternating every 24. I also noticed that it I changed the order the latitudes were searched for, I got the exact opposite results, with NAs in the sections of 24 that had values before. So, I repeated the function, once each way, leaving me with two 41,472 each half NA in alternating positions. I then looped through a sequence of 1 to 41,472 and applied a formula that took 24 from the first, 24 from the second, and so on, ignoring all of the NAs, merging the two vectors. Then I added the vector to the data frame.

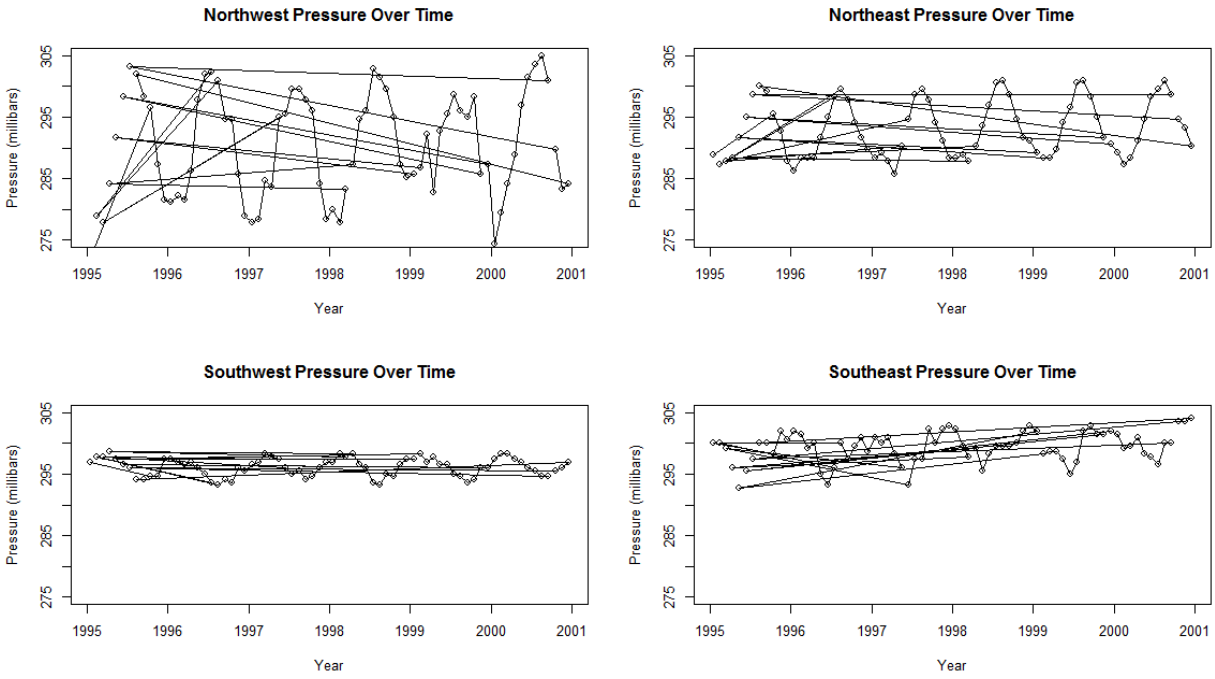
Step 4:

* = The question has a colored graph. All colored graphs will be printed in color on the last page of the report.

*Question 1: The plot can be seen below. It shows a positive relationship between temperature and pressure. Most of the points are a dark blue, indicating a low percent for cloud low. The few points with a higher percent are in the middle of the temperature range and in the higher half of the pressure range. This shows a positive relationship between cloud low and pressure.



Question 2: The plot can be seen below. I could not figure out how to fix the lines, but the shapes can still be seen. The graphs are just messier than I would like. Looking at the plots, the plots for Northeast and Northwest are very similar. The pressure goes up each year, peaking at about the middle of the year, and going back down by the end of the year. Northwest has the largest range of pressure, followed by Northeast, then Southeast, then Southwest. The plots for Southeast and Southwest are very similar. They have a much smaller range than the North graphs. The graphs have a similar up and down pattern, but going down in the middle of the year, and peaking right as the year starts/ends.



Question 3: I sadly could not get this to work. I tried many combinations of looping.

Question 4: Need Q3.

Question 5: Need Q3.

Code Appendix

```
# Step One
# =======
# function that takes in file and creates data frame:
MakeData = function(fileName){
  filePath = paste0("~/UC Davis/STA 141/HW 2/NASA/", fileName)

  # Getting the date: (the same for every row inside this function)
  file = readLines(filePath)
  date = strsplit(file[5], "    :")
  date = sapply(date, `[`, 2)
  date = strsplit(date, " ")
  date = sapply(date, `[`, 2)
  date = as.Date(date, format="%d-%b-%Y")

  # Repeating the date based on the size in row 4 of the file
  size = strsplit(file[4], ":")
  size = sapply(size, `[`, 2)
  size = strsplit(size, " ")
  size1 = as.numeric(sapply(size, `[`, 2))
  size2 = as.numeric(sapply(size, `[`, 4))
  size = size1 * size2
  dates = rep(date, size) # To be added to data frame - 4th col

  # Similar idea for longitude:
  # All in row 6 of file, each one needs to be repeated size1 times
  # want: long1, long2, long3, ... long24 (size1 times)
  longs = unlist(strsplit(file[6], " " ))
  longs = longs[longs != ''] # Gets rid of empty entries
  # Long: West = negative, East = positive
  # Need to make numeric and get rid of letters:
  longNumeric = lapply(longs, function(lon) {
    dir = substr(lon, nchar(lon), nchar(lon)) # W or E
    lon = as.numeric(substr(lon, 1, nchar(lon) - 1) ) # Change to numeric and remove letter
    if (dir == "W") {
```

```
    lon = -lon
  } # If West, make negative
  return(lon)
})

longNumeric = unlist(longNumeric)
# Repeating the necesary amount of times:
longsRep = rep(longNumeric, size1) # Make col 2 of data frame

# Similar idea for lat:
# Want: lat1, lat1, lat1, ...(times size1), ... lat24, ..., lat23, lat24 (times 24)
# Lat: South = negative North = positive
latitudes = sapply(8:length(file), function(i) {
  lats = strsplit(file[i], " /")
  lats = sapply(lats, `[`, 1)
  lats = strsplit(lats, " ")
  lats = sapply(lats, `[`, 2)
})

# Loop through each latitude and remove letter, change to numeric, and change to +/-
latNumeric = lapply(latitudes, function(lat){
  dir = substr(lat, nchar(lat), nchar(lat)) # N or S
  lat = as.numeric(substr(lat, 1, nchar(lat) - 1) ) # Change to numeric and remove letter
  if (dir == "S") {
    lat = -lat
  } # If South, make negative
  return(lat)
}) # sapply returns a strange vector with labels, so have to unlist the list

latNumeric = unlist(latNumeric)
# Taking that vector and repeating it:
latsRep = rep(latNumeric, each = size1) # To be added to data.frame - col 3

# Similar idea for values:
# Takes the values in rows 8 and on and puts them into lector (length 576 each)
values = lapply(8:length(file), function(j) {
  val = strsplit(file[j], " /")
  val = sapply(val, `[`, 2)
  val = unlist(strsplit(val, " "))
  val = val[val != ''] # Got this from piazza
```

```
  val = val[-1] # Removing first element, not a value
  # in cloudlow, there are some "...." instead of values, so fixing that:
  val = lapply(val, function(i){
   if(i == "....")
    i = NA
   return(i)
  })

  val = as.numeric(val) # Changing to number
 })
 values = unlist(values)   # values done - enter as 1st col in data frame

 # Put into data frame:
 return(df = data.frame(Value = values, Latitude = latsRep, Longitude = longsRep, Date =
        dates))
}

fileNames = c("cloudhigh", "cloudlow", "cloudmid", "ozone", "pressure", "surftemp",
        "temperature")
fileNameNums = lapply( fileNames[1:7], function(x) list.files("~/UC Davis/STA 141/HW
        2/NASA", pattern = x)) # list.files idea from piazza
# fileNameNums[[1]] is all of clouthigh.txt[1, 2, ...] and so on through [[7]]

# Creating a list of seven data frames:
sevenFrames = lapply(fileNameNums, function(allFiles){
 # The above loops through each file (seven times for each type)

 manyFrames = (lapply(allFiles, function(files) MakeData(files))) # Sending each file to the
        # function MakeData. The above is a list of many data frames, and within this loop is 1/7

 # Now need to stack all of manyFrames:
 oneFrame = do.call(rbind, manyFrames) # Got from piazza
})
```

 # Step Two

```
# =======
# Checking date for all seven:
identical(sevenFrames[[1]]$Date, sevenFrames[[2]]$Date, sevenFrames[[3]]$Date,
        sevenFrames[[4]]$Date, sevenFrames[[5]]$Date, sevenFrames[[6]]$Date,
        sevenFrames[[7]]$Date) # Probably someway to loop, but fast enough
```

```
# Checking lat:
identical(sevenFrames[[1]]$Latitude, sevenFrames[[2]]$Latitude, sevenFrames[[3]]$Latitude,
        sevenFrames[[4]]$Latitude, sevenFrames[[5]]$Latitude, sevenFrames[[6]]$Latitude,
        sevenFrames[[7]]$Latitude)

# Checking long:
identical(sevenFrames[[1]]$Longitude, sevenFrames[[2]]$Longitude,
sevenFrames[[3]]$Longitude,
        sevenFrames[[4]]$Longitude, sevenFrames[[5]]$Longitude, sevenFrames[[6]]$Longitude,
        sevenFrames[[7]]$Longitude)

# All three return true, so now combining the seven into one data frame:
oneFrame = data.frame(Date = sevenFrames[[1]]$Date, Longitude =
sevenFrames[[1]]$Longitude,
                Latitude = sevenFrames[[1]]$Latitude, Cloudhigh = sevenFrames[[1]]$Value,
                Cloudlow = sevenFrames[[2]]$Value, CloudMid = sevenFrames[[3]]$Value,
                Ozone = sevenFrames[[4]]$Value, Pressure = sevenFrames[[5]]$Value,
                Surftemp = sevenFrames[[6]]$Value, Temperature = sevenFrames[[7]]$Value)


# Step Three
# ========
# Function returns vector with half NAs, half elevation values, alternaing every 24
FindMatch = function(lat, long, eleTable){

  # All possible values for lat and long, kind of reverse rounding:
  lat1 = lat + 0.05
  lat2 = lat - 0.05
  long1 = long + 0.05
  long2 = long - 0.05

  # Searching for all combinations:
    if (!is.null(eleTable[as.character(lat1), as.character(long1)])) {
      return(eleTable[as.character(lat1), as.character(long1)])
      break
  }
    if (!is.null(eleTable[as.character(lat1), as.character(long2)])) {
      return(eleTable[as.character(lat1), as.character(long2)])
      break
  }
```

```
   if (!is.null(eleTable[as.character(lat2), as.character(long1)])) {
     return(eleTable[as.character(lat2), as.character(long1)])
     break
  }
   if(!is.null(eleTable[as.character(lat2), as.character(long2)])) {
     return(eleTable[as.character(lat2), as.character(long2)])
     break
  }
}

# Function returns the same as FindMatch, but opposite NA and elevation:
FindMatch2 = function(lat, long, eleTable){

  # All possible values for lat and long, kind of reverse rounding:
  lat1 = lat - 0.05
  lat2 = lat + 0.05
  long1 = long + 0.05
  long2 = long - 0.05

  # Searching for all combinations:
  if (!is.null(eleTable[as.character(lat1), as.character(long1)])) {
    return(eleTable[as.character(lat1), as.character(long1)])
    break
  }
  if (!is.null(eleTable[as.character(lat1), as.character(long2)])) {
    return(eleTable[as.character(lat1), as.character(long2)])
    break
  }
  if (!is.null(eleTable[as.character(lat2), as.character(long1)])) {
    return(eleTable[as.character(lat2), as.character(long1)])
    break
  }
  if(!is.null(eleTable[as.character(lat2), as.character(long2)])) {
    return(eleTable[as.character(lat2), as.character(long2)])
    break
  }
}
```

```
eleTable = read.table("~/UC Davis/STA 141/HW 2/NASA/intlvtn.dat", check.names = FALSE)
#data.frame

# Loops through lat and long for each row and sends those values to FindMatch() which
# returns the elevation:
# Has 1:24 elevation, 25:48 NA, 49:72 elevation, and so on:
elevation1 = mapply(function(lat, long){
  FindMatch(lat, long, eleTable)
}, oneFrame$Latitude, oneFrame$Longitude)

# Has 1:24 NA, 25:48 elevation, 49:72 NA, and so on:
elevation2 = mapply(function(lat, long){
  FindMatch2(lat, long, eleTable)
}, oneFrame$Latitude, oneFrame$Longitude)

nums= seq(1, 41472, by = 24)

# Definately doing this a very strange way, it is the only thing I can get to work:

# Loops through elevation1 and elevation2 taking the values for elevation we want:
trying = lapply(nums, function(n){
  if((((n + 23) / 24) %%  2) == 0) {# Even
    yo = elevation2[n:(n + 23)]
  }
  else{ # Odd
    yo = elevation1[n:(n + 23)]
  }

}) # It worked!!!!!

elevationFinal = unlist(trying)
# Now need to add elevationFinal as 11th col in oneFrame:
oneFrame$Elevation = elevationFinal
```

 # Step Four

 # ========

Question 1:

```
library(ggplot2)
ggplot(oneFrame, aes(x = Temperature, y = Pressure)) + geom_point(size = 5, aes(color =
        Cloudlow)) + labs(x="Temperature (kelvin)", y = "Pressure (millibars)", title =
        "Temperature, Pressure and Cloudlow(%)")

# Question 2:
# NW:
par(mfrow = c(2,2))
northWest = subset(oneFrame, (oneFrame$Longitude == -113.8) & (oneFrame$Latitude ==
        36.2))
plot(northWest$Date, northWest$Temperature, ylim = c(275, 305), type = "o",xlab = "Year",
        ylab = "Pressure (millibars)", main = "Northwest Pressure Over Time")
#NE:
northEast = subset(oneFrame, (oneFrame$Longitude == -56.2) & (oneFrame$Latitude == 36.2))
plot(northEast$Date, northEast$Temperature, ylim = c(275, 305), type = "o", xlab = "Year", ylab
        = "Pressure (millibars)", main = "Northeast Pressure Over Time")
#SW:
southWest = subset(oneFrame, (oneFrame$Longitude == -113.8) & (oneFrame$Latitude == -
        21.2))
plot(southWest$Date, southWest$Temperature, ylim = c(275, 305), type = "o", xlab = "Year",
        ylab = "Pressure (millibars)", main = "Southwest Pressure Over Time")
#SE:
southEast = subset(oneFrame, (oneFrame$Longitude == -56.2) & (oneFrame$Latitude == -
        21.2))
plot(southEast$Date, southEast$Temperature, ylim = c(275, 305), type = "o", xlab = "Year",
        ylab = "Pressure (millibars)", main = "Southeast Pressure Over Time")

# Question 3:
# Tried many things, could not get it to work:
# Try 1:
meanFrame = data.frame(Latitude = rep(unique(oneFrame$Latitude), 24), Longitude =
rep(unique(oneFrame$Longitude), each = 24))

# Just trying for Cloudlow and if worked would repeat for others
means = lapply(1:length(meanFrame), function(i){
  values = lapply(1:length(oneFrame), function(j){
    if ((meanFrame[i]$Latitude == oneFrame[j]$Latitude) & (meanFrame[i]$Longitude ==
oneFrame[j]$Longitude))
      val = oneFrame[i]$Cloudlow
    #if((mf[1] == of[3]) & (mf[2] == of[2]))
```

```
    # val = of[4]
  })
  mean(values)
})

# Try #2:
# Just trying for one variable, would repeat if worked:
means = mapply(function(lat, long) {
  values = lapply(1:length(oneFrame), function(i){
    if((oneFrame[i]$Latitude == lat) & (oneFrame[i]$Longitude == long)){
      val = oneFrame[i]$Cloudhigh
    }
  })
  mean = mean(values, na.rm = TRUE)
}, unique(oneFrame$Latitude), unique(oneFrame$Longitude))

# Question 4:
# Can't do without Q3 :(

# Question 5:
# Can't do without Q3 :(
```

References:

1. Links:

- http://r.789695.n4.nabble.com/Selecting-non-empty-elements-after-strsplit-of-string-td3087349.html
- https://msdn.microsoft.com/en-us/library/aa578799.aspx
- http://r.789695.n4.nabble.com/remove-last-char-of-a-text-string-td2254377.html
- http://r.789695.n4.nabble.com/how-to-convert-list-into-a-vector-td895952.html
- http://stackoverflow.com/questions/7721262/colouring-plot-by-factor-in-r
- http://www.r-bloggers.com/ggplot2-cheatsheet-for-scatterplots/
- http://rfunction.com/archives/1912
- http://www.inside-r.org/r-doc/base/strftime

2. Piazza

3. Code from lecture

Colored Graphs