STA141_Assignment2

Chad Pickering A03

Wednesday, October 21, 2015

Corresponded with: Janice Luong, Rico Lin, Ricky Safran, Sierra Tevlin, Hannah Kosinovsky

Resources: Office hours, Piazza forums, R help documentation, Stack Overflow

This is a blend of two methods(one less dynamic, one more dynamic), tweaked and molded into a succinct solution that satisfies each question asked. Many lines are commented for convenience and easy reading.

Step 1: This step generates the file names from scratch, and extracts relevant data where var is one of the seven measurement columns (cloudhigh, cloudmid, etc.). Each line has a comment explaining what it does.

```
setwd("C:/Users/cpickering/Syncplicity Folders/ChadSync/STATISTICS/STA141/Assignment 2")

readFile <- function(q, vars) #generates date, long, lat data frame
{
  #generate all file names from scratch in correct order (1-72)
  txt_names <- paste(vars, 1:72, ".txt", sep = "")

  txt <- read.table(txt_names[q], skip = 7, strip.white = TRUE) #reads tabular data set

  lat <- as.character(txt[[1]]) #get 24 unique lats from first column in table
  lat_rep <- rep(lat, times = 24) #repeat set of 24, 24 times for 576 elements

  header6 <- readLines(txt_names[q], n = 6) #get 6th row from each .txt file
  long_levels <- strsplit(header6[6], " +")[[1]] #break up string to individual longitudes
  long <- long_levels[-1] #remove first empty quotation element
  long_rep <- rep(long, each = 24) #repeat each longitude value 24 times

  time <- strsplit(header6[5], " +")[[1]] #extract date column, separate individual elements
  date <- time[4] #take fourth element
  date_rep <- rep(date, each = length(long_rep)) #repeat date in file 576 times

  obs_matrix <- as.matrix(txt[, -(1:3)]) #exclude first 3 columns, give obs as matrix
  obs <- as.numeric(obs_matrix) #convert to numeric

  #nasa df with date, long, lat, var
  nasa_dataframe <- data.frame(date = date_rep, long = long_rep, lat = lat_rep, var = obs)
}
```

In this step, a 4-column data frame is created for each variable. There is a comment for each line.

```
gendfs <- function(h)
{
  df_creation <- lapply(1:72, readFile, var = h) #varname h gets passed to readFile
  var <- do.call(rbind, df_creation) #apply function rbind to specific varname (1 -> 72)
  names(var)[4] <- h #in each separate df, put the measurements/var in the fourth column
  var #output var name
}
```

Here, I pass in “cloudhigh”, etc. to h in gendfs, and store the result in cloudhigh, etc.

```r
cloudhigh <- gendfs("cloudhigh")
cloudmid <- gendfs("cloudmid")
cloudlow <- gendfs("cloudlow")
ozone <- gendfs("ozone")
pressure <- gendfs("pressure")
surftemp <- gendfs("surftemp")
temperature <- gendfs("temperature")
```

Step 2: By the transitive property, all first three columns (date, long, lat) in each four-column data frame are equal element-wise. Therefore, the dates for each set of files are exactly the same (576 observations per variable per date), and each set of longitude/latitude pairs for each .txt file are the same as well.

```r
identical(cloudhigh[, 1:3], cloudmid[, 1:3])
```

```
## [1] TRUE
```

```r
identical(cloudhigh[, 1:3], cloudlow[, 1:3])
```

```
## [1] TRUE
```

```r
identical(cloudhigh[, 1:3], ozone[, 1:3])
```

```
## [1] TRUE
```

```r
identical(cloudhigh[, 1:3], pressure[, 1:3])
```

```
## [1] TRUE
```

```r
identical(cloudhigh[, 1:3], surftemp[, 1:3])
```

```
## [1] TRUE
```

```r
identical(cloudhigh[, 1:3], temperature[, 1:3])
```

```
## [1] TRUE
```

Select all columns in the seven data frames to cbind() into the final data frame for step 2:

```r
datecoords <- subset(cloudhigh, select = -c(cloudhigh))
cloudhigh_data <- subset(cloudhigh, select = cloudhigh)
cloudmid_data <- subset(cloudmid, select = cloudmid)
cloudlow_data <- subset(cloudlow, select = cloudlow)
ozone_data <- subset(ozone, select = ozone)
pressure_data <- subset(pressure, select = pressure)
surftemp_data <- subset(surftemp, select = surftemp)
temperature_data <- subset(temperature, select = temperature)
nasa <- cbind(datecoords, cloudhigh_data, cloudmid_data, cloudlow_data,
              ozone_data, pressure_data, surftemp_data, temperature_data)
```

NA investigation: Here I investigate where NA values occur and which rows have them.

```
table(is.na(nasa)) #110 NAs
```

```
##
##  FALSE   TRUE
## 414610    110
```

```
sapply(nasa, function(x) length(x[is.na(x)])) #all 110 are cloudlow observations
```

```
##        date        long         lat   cloudhigh    cloudmid    cloudlow
##           0           0           0           0           0         110
##       ozone    pressure    surftemp temperature
##           0           0           0           0
```

```
#which(is.na(nasa$cloudlow)) #all rows in which cloudlow is NA - commented out, not needed
```

Time adjustment: The time extracted from the raw data is not in a format that R recognizes as a date. The following formats it:

```
date_revised <- as.Date(nasa$date, format = "%d-%b-%Y")#change current date format
nasa$date <- date_revised #store revised dates in date column in df
```

Longitude adjustment: The longitudes in the intlvtn.dat file are ideal because they are in a form that can be plotted. The following fixes the extracted longitudes to match these:

```
long_tmp <- as.character(nasa$long) #adjust each element of long to character
long_removew <- function(x) substr(x, 1, nchar(x) - 1)
wremoved <- long_removew(long_tmp) #removes W in all longitude elements
num_wremoved <- as.numeric(wremoved) #converts new long elements to numeric
neg_wremoved <- num_wremoved * (-1) #adds a negative to longitude elements
nasa$long <- neg_wremoved #stores new longitudes in long column in df
```

Latitude adjustment: The latitudes in the intlvtn.dat file are ideal because they are in a form that can be plotted. The following fixes the extracted latitudes to match these:

```
lat_char <- as.character(nasa$lat) #adjust each element of lat to character
lat_last <- substring(lat_char, nchar(lat_char)) #return only last character of each element - N/S
lat_southern <- (lat_last == "S") #logical vector - TRUE when S
lat_remover <- function(x) substr(x, 1, nchar(x) - 1)
nsremoved <- lat_remover(lat_char) #removes N, S from all elements
num_nsremoved <- as.numeric(nsremoved) #converts new lat elements to numeric
num_nsremoved[lat_southern] <- num_nsremoved[lat_southern] * (-1) #multiply S lats by -1
nasa$lat <- num_nsremoved #stores new latitudes in lat column in df
```

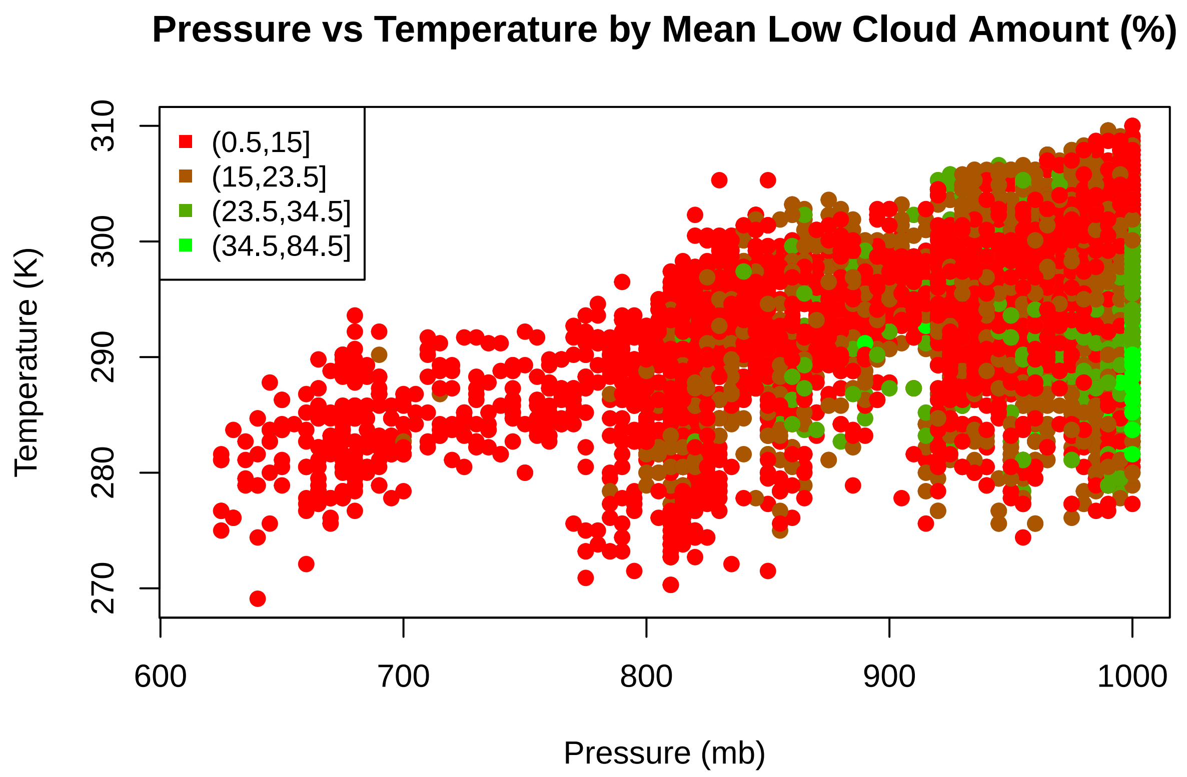
Step 3 - Elevation: This step extracts the elevation data, repeats them, and stores that numeric data into a column in the data frame.

```r
#read in .dat file without first row
elevation_table <- read.table("intlvtn.dat", skip = 1, strip.white = TRUE)
elevobs_only <- as.matrix(elevation_table[, -1]) #give original table (all obs) without 1st column
elev_num <- as.numeric(elevobs_only) #convert matrix to numeric
elevations_rep <- rep(elev_num, times = 72) #repeat elevation data 72 times (once for each month)
nasa$elevation <- elevations_rep
```

Step 4: The following are the specific explorations of the NASA data.

Part 1: Plot temperature versus pressure. Color code the points by the value of cloudlow. I cut the cloudlow percentages by quartiles, and assigned a color to each cut category for easy reading. The plot itself shows that as pressure decreases, temperature also decreases in general. The mean percentage of the sky covered by low clouds also decreases as pressure decreases. It does not look as though temperature affects the mean percentage of low clouds covering the sky. It should be noted that there are three rather distinct areas on the plot, and the frequency of points with a lower pressure is much less than the frequency of points with a pressure near 1000mb. In conclusion, it looks as though there are more low clouds when the pressure increases. The correlation between any of the three variables are not strong.

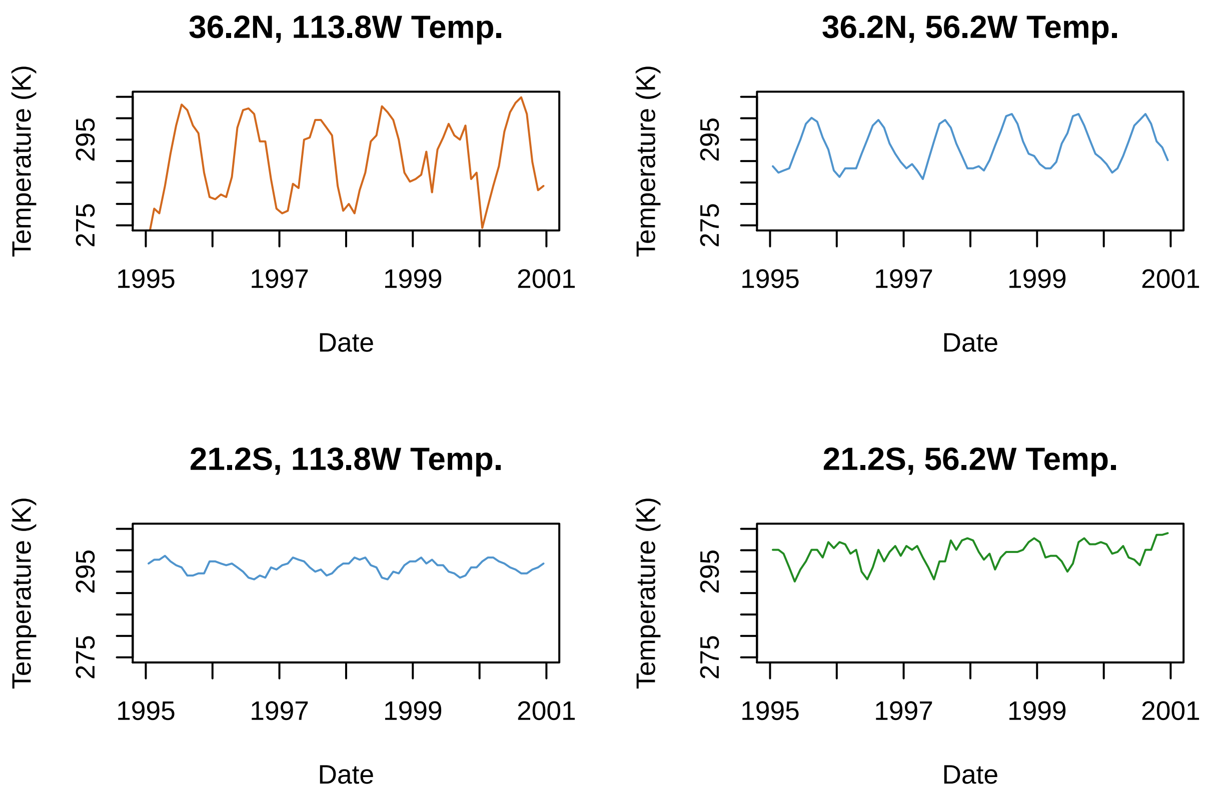
```r
cloudlow_quantile <- quantile(nasa$cloudlow, probs=seq(0,1,0.25), na.rm=TRUE) #define quartiles
cloudlow_cut <- cut(nasa$cloudlow, cloudlow_quantile) #cut by quartile

clow_palette <- colorRampPalette(c('red','green')) #define color boundaries

clow_colassign <- clow_palette(4)[as.numeric(cloudlow_cut)] #call clow_palette on the cut categories

par(mfrow=c(1,1), mar=c(5.1,4.1,4.1,2.1))

plot(nasa$pressure, nasa$temperature, xlab = "Pressure (mb)",
     ylab = "Temperature (K)", main = "Pressure vs Temperature by Mean Low Cloud Amount (%)",
     pch = 19, cex = 1, col = clow_colassign)
legend("topleft", legend = levels(cloudlow_cut), col = unique(clow_colassign), pch = 15, cex = 0.9)
```



Part 2: For the points at the four corners of the spatial grid, display the values of temperature over time. The four corners of the spatial grid have been defined and the temperatures over time were plotted. The volatility of temperature depends directly on the climate of the where the coordinate is. For example, the northwestern point is near Las Vegas, and temperature volatility in desert regions is high, which agrees with the plot. The northeastern and southwestern points are hundreds of miles into the ocean, and therefore, the volatility of temperature is rather low in comparison. The level of volatility of monthly temperature averages in the ocean is predominantly affected by frequency of low-pressure systems, proximity to the equator, and distance from land. The difference in amplitude between the two plots is a direct example of this. The southwestern point is closer to the equator and farther out in the ocean than the northeastern point. The southeastern point has a rather small temperature range and the volatility does not follow a smooth pattern (e.g. there are many fluctuations in averages in the winter months when temperatures are at their maximum for the year), which can possibly be attributed to the more quasi-tropical climate there.

```r
min_long <- min(nasa$long) #assign specific values
max_long <- max(nasa$long)
min_lat <- min(nasa$lat)
max_lat <- max(nasa$lat)

topleft <- nasa[nasa$long == min_long & nasa$lat == max_lat, ] #top left coordinate, etc.
toplefttemp <- topleft$temperature #only temp col

topright <- nasa[nasa$long == max_long & nasa$lat == max_lat, ]
toprighttemp <- topright$temperature

bottomleft <- nasa[nasa$long == min_long & nasa$lat == min_lat, ]
```

```r
bottomlefttemp <- bottomleft$temperature

bottomright <- nasa[nasa$long == max_long & nasa$lat == min_lat, ]
bottomrighttemp <- bottomright$temperature

eachdate <- unique(nasa$date) #take unique dates

par(mfrow=c(2,2), mar=c(5.1,4.1,4.1,2.1)) #2x2 plot

plot(eachdate, toplefttemp, type = "o", pch = NA_integer_, xlab = "Date",
     ylab = "Temperature (K)", main = "36.2N, 113.8W Temp.",
     ylim = c(275, 305), col = "chocolate")

plot(eachdate, toprighttemp, type = "o", pch = NA_integer_, xlab = "Date",
     ylab = "Temperature (K)", main = "36.2N, 56.2W Temp.",
     ylim = c(275, 305), col = "steelblue3")

plot(eachdate, bottomlefttemp, type = "o", pch = NA_integer_, xlab = "Date",
     ylab = "Temperature (K)", main = "21.2S, 113.8W Temp.",
     ylim = c(275, 305), col = "steelblue3")

plot(eachdate, bottomrighttemp, type = "o", pch = NA_integer_, xlab = "Date",
     ylab = "Temperature (K)", main = "21.2S, 56.2W Temp.",
     ylim = c(275, 305), col = "forestgreen")
```



Part 3: For all points on the grid, compute the average and standard deviation for each of the 7 variables across time. For each unique latitude/longitude pair, I am generating the column means and column standard deviations for each of the seven variables. I gave an output of just the first 3 unique pairs as a sample.

```
nasa_sevenvars <- nasa[c(4:10)] #give all cols except first three and elevation
splitcoords <- split(nasa_sevenvars, list(nasa$long, nasa$lat)) #split on chosen cols and long/lat

library(fBasics)
```

```
## Loading required package: timeDate
## Loading required package: timeSeries
##
##
## Rmetrics Package fBasics
## Analysing Markets and calculating Basic Statistics
## Copyright (C) 2005-2014 Rmetrics Association Zurich
## Educational Software for Financial Engineering and Computational Science
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
## https://www.rmetrics.org --- Mail to: info@rmetrics.org
```

```
latlong <- function(a)
{
  coord_means <- colMeans(splitcoords[[a]]) #colMeans of the a'th element of long/lat pair
  coord_sds <- colStdevs(splitcoords[[a]]) #colStdevs of the a'th element of long/lat pair

  cbind(mean = round(coord_means, 3), sd = round(coord_sds, 3)) #make 2 cols, round to 1000th
}
by_latlong <- lapply(1:length(splitcoords), latlong) #iterate through each unique pair
head(by_latlong, 1) #in the interest of space, this is the first 1 of 576
```

```
## [[1]]
##                 mean     sd
## cloudhigh      1.993  2.770
## cloudmid       5.778  3.818
## cloudlow      37.174  5.783
## ozone        268.250 12.260
## pressure    1000.000  0.000
## surftemp     296.242  1.499
## temperature  296.108  1.476
```

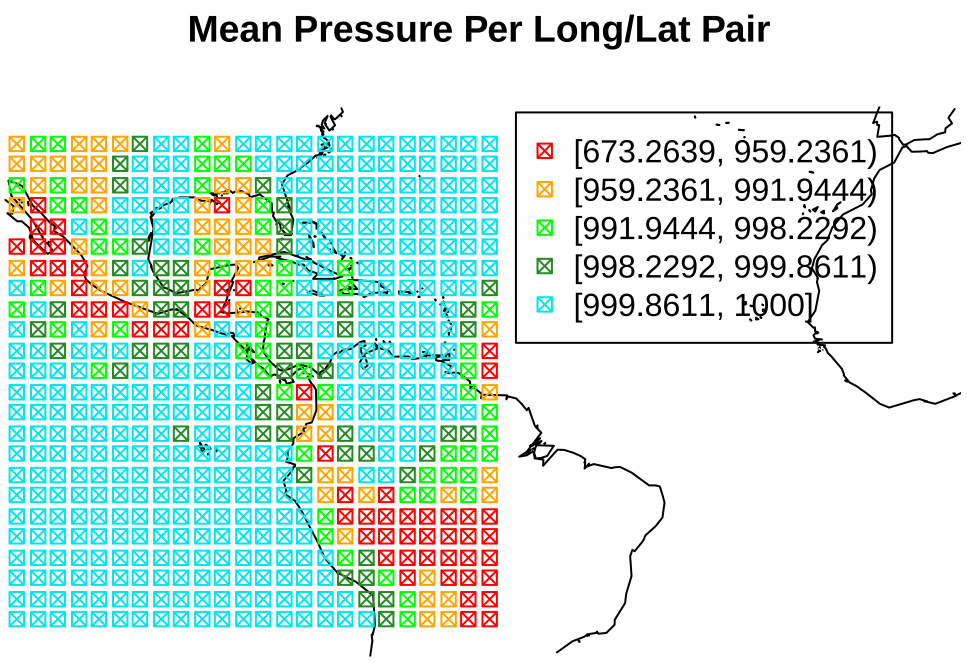
```
#by_latlong is, in reality, run in its entirety
```

Part 4: Display the average value for pressure computed in the previous question on a map. In this part, pressure is extracted from the main data frame, and the mean pressure for each unique lat/long pair is calculated and stored in a new data frame. After formatting the coordinates, I cut by deciles, and as the last 50% are all the same value (1000), all of those values are stored as one large category ([999.8611, 1000]). The formatting of the plot was rather similar to that of Part 1 - every unique lat/long pair was assigned a color based on its average pressure over time. In the plot, it looks as though the highest mean pressures over time occur on the ocean, and the lowest mean pressures over time generally occur at higher elevations.

```r
longlat_4_mean <- function(phi)
{
  ll_mean <- colMeans(splitcoords[[phi]])
  cbind(mean = ll_mean)
}

mean_per_longlat <- lapply(1:length(splitcoords), longlat_4_mean) #mean for each var per long/lat
mean_per_longlat_trans <- t(as.data.frame(mean_per_longlat)) #transpose rows/columns from prev.
trans_df <- as.data.frame(mean_per_longlat_trans) #need pressure as a column in the data frame
pressure_mu <- trans_df$pressure

long_rev <- unique(nasa$long) #give unique longitudes
long_rev_rep <- rep(long_rev, each = 24) #repeat set long_rev 24 times

lat_rev <- rev(unique(nasa$lat)) #reverse the order of the latitudes
lat_rev_rep <- rep(lat_rev, times = 24) #repeat each lat_rev 24 times

pressure_longlat <- cbind(long = long_rev_rep, lat = lat_rev_rep, pressure_mu) #one df w/ long/lat/pr
pressure_longlat_df <- as.data.frame(pressure_longlat)

pressure_longlat_q <- quantile(pressure_longlat_df$pressure, seq(0, 1, 0.1), na.rm=TRUE) #deciles
pressure_longlat_dist <- cut(pressure_longlat_df$pressure,
                             c(673.2639, 959.2361, 991.9444, 998.2292, 999.8611, 1000.0000))
pressure_color <- c("red", "orange", "green", "forestgreen", "turquoise2")[pressure_longlat_dist]

par(mfrow=c(1,1), mar=c(5.1,4.1,4.1,2.1)) #2x2 plot

library(maps)
```

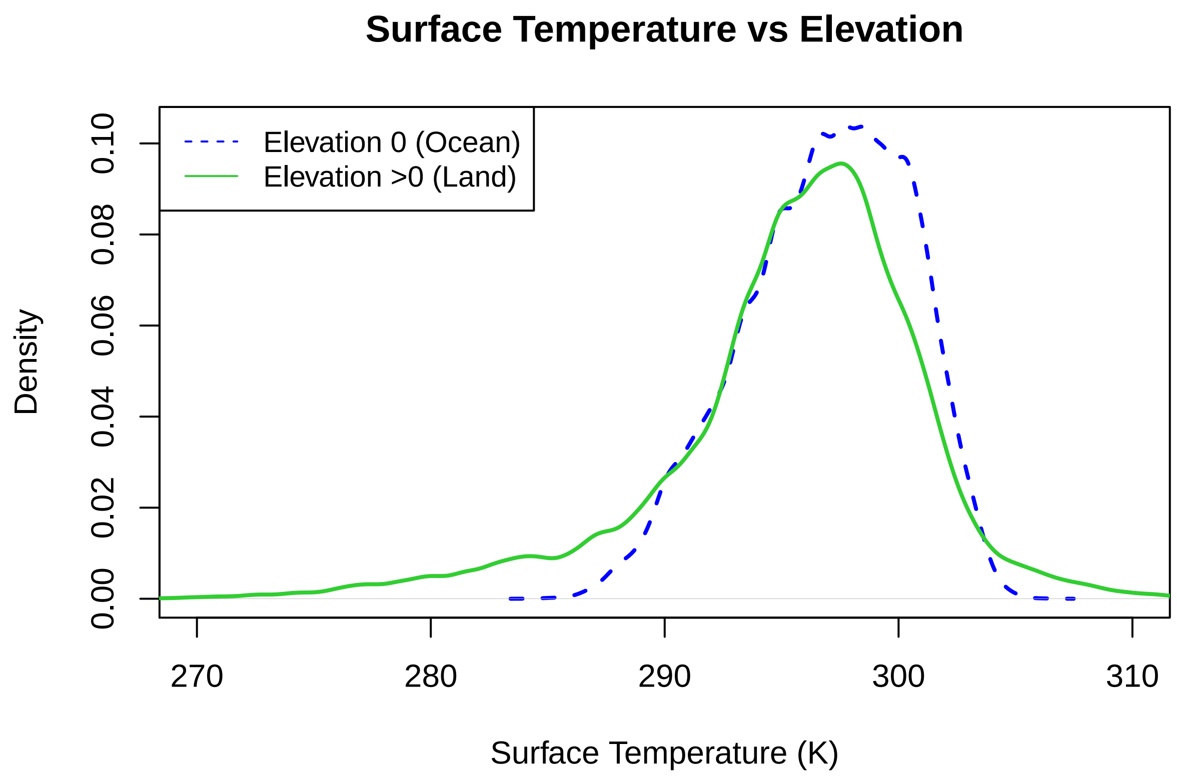
```
##
## Attaching package: 'maps'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     ozone
```

```r
map('world', interior = FALSE, xlim = c(-115, 0), ylim = c(-25, 40)) #would put axes; can't implement
points(pressure_longlat_df$long, pressure_longlat_df$lat,
       col = pressure_color, pch = 7, title("Mean Pressure Per Long/Lat Pair")) #x - long, y - lat
legend(-53, 40, c("[673.2639, 959.2361)", "[959.2361, 991.9444)",
                  "[991.9444, 998.2292)", "[998.2292, 999.8611)", "[999.8611, 1000]"),
                  pch = 7, col = c("red", "orange", "green", "forestgreen", "turquoise2"))
```

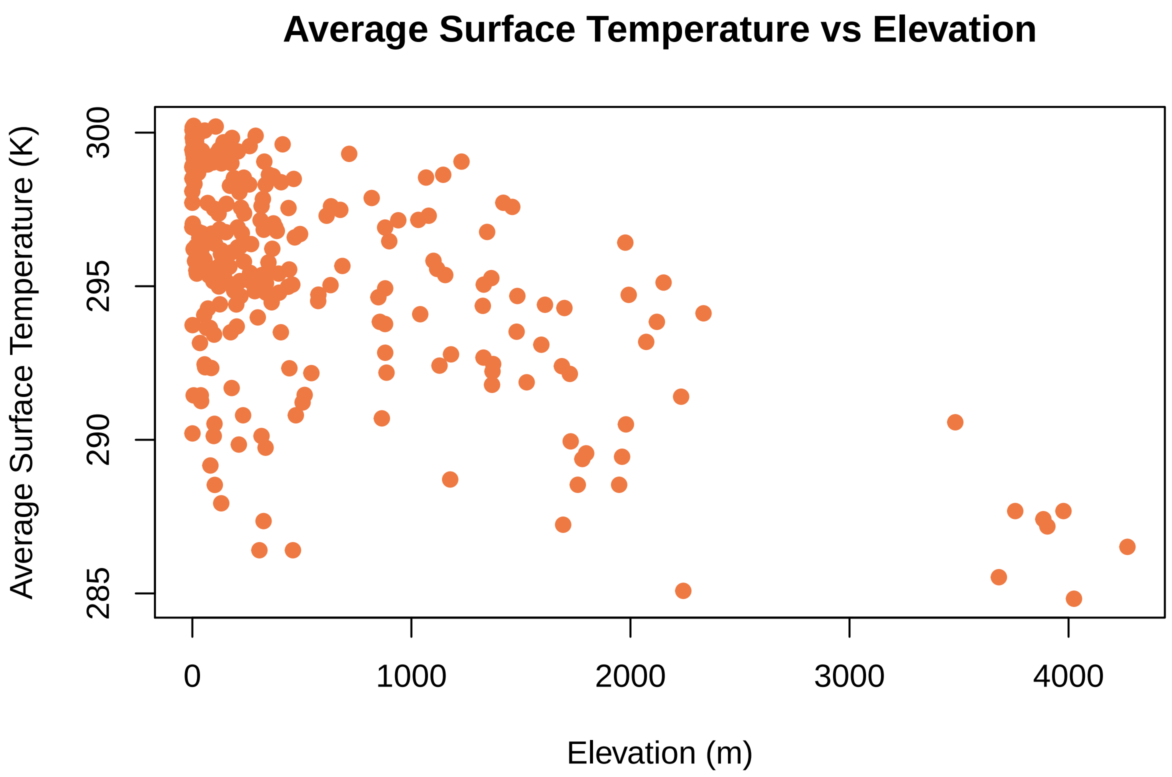


Part 5: Display average surface temperature versus elevation. First, I wanted to explore how ocean and land affected surface temperature. Surface temperature over land is much more varied than surface temperature over ocean, which agrees with the analysis for Part 2. The intended solution for this problem is as follows: after splitting the surface temperature column over elevation and giving the average surface temperature for each unique elevation value, the elevations are sorted and the two variables are plotted against each other. As the elevation increases, average surface temperature generally decreases. There are many more unique elevations near sea level, and the average surface temperatures vary quite a bit - in the region investigated, there are a wide variety of near-sea-level areas with different climates, so this is expected.

```
par(mfrow=c(1,1), mar=c(5.1,4.1,4.1,2.1))

elevation_zero <- nasa$elevation == 0 #plot when elevation is zero (ocean)
plot(density(nasa$surftemp[elevation_zero]), xlab = "Surface Temperature (K)",
     main = "Surface Temperature vs Elevation", xlim = c(270,310), col = "blue", lty = 2, lwd = 2)

elevation_land <- nasa$elevation != 0 #add lines on plot for non-zero elevation (land)
lines(density(nasa$surftemp[elevation_land]), xlab = "Surface Temperature (K)",
      main = "Surface Temperature vs Elevation", xlim = c(270,310), col = "limegreen", lwd = 2)

legend("topleft", legend = c("Elevation 0 (Ocean)", "Elevation >0 (Land)"),
       col = c("blue", "limegreen"), lty = c(2, 1), cex = 0.9)
```



```r
#Intended solution:
par(mfrow=c(1,1), mar=c(5.1,4.1,4.1,2.1))

uq_elev <- unique(nasa$elevation) #get all unique elevation values
surftemp_nasa <- subset(nasa, select = c(surftemp)) #give only surftemp column of nasa df
surftemps_elev <- split(surftemp_nasa, list(nasa$elevation)) #split surftemp on unique elevations
surftemp_averages <- function(index)
{
  surftemp_mu <- colMeans(surftemps_elev[[index]])
  surftemp_mu
}
iteratesurftemp <- sapply(1:length(uq_elev), surftemp_averages) #gives stemp means for uq elevations
avs_extract_num <- as.numeric(iteratesurftemp)
elevation_order <- sort(unique(nasa$elevation)) #gives unique elevations small -> large
plot(elevation_order, avs_extract_num, col = "sienna2", xlab = "Elevation (m)",
     ylab = "Average Surface Temperature (K)",
     main = "Average Surface Temperature vs Elevation", pch = 19, cex = 1)
```


Average Surface Temperature vs Elevation
Average Surface Temperature (K)
285
290
295
300
0
1000
2000
3000
4000
Elevation (m)