STA 141B NASA

Suvethika Kandasamy

2024-04-16

When approaching this assignment, I first attempted to just extract the data from one file then used that to create a function to replicate. I continuously validated each step in my process by printing out the variables I created and double checking if my code was doing what I intended.

So starting with cloudhigh1 I extracted the date, latitude, longitude, and data values using the substr() formula with some trail and error on the exact character numbers. I orignally tried strssplit first that didn't work so then using substr function to extract just the date out of that line and then used as.date function to convert it to a date type.

For the date column I just had to convert the value to a date. For the remaining variables, I used strsplit() and unlist() functions to seperate out the values. In this case, I had multiple empty strings after splitting so I removed those using nzchar() which I found when asking ChatGPT about how I can remove those values. After that I just made the values numeric and for the latitiude and longitude I included an if statement to make some of the values negative. Originally when splitting the latitude and longitude the direction values were also still included so I used gsub() to remove them before converting them into a numeric value.

My idea behind creating the dataframe was just to use the rep() function to repeat the date, latitiude, longitude as many times as necessary to match the number of data points in the file.

I have heard of these strsplit() and gsub() but this is my first time using them in an assignment so I refered to the help menu for these functions and additionally tested out how they worked prior to applying them to the data. The other functions for the most part I am familiar with.

```r
# function to extract date, longitude, latitude, and cloud high to make a base dataframe
process_cloudhigh_file <- function(file_path) {
  allline <- readLines(file_path, n = -1)
  # Extract date
  date <- allline[5]
  date_substring <- substr(date, start = 25, stop = 35)
  date_as_date <- as.Date(date_substring, format = "%d-%b-%Y")

  # Extract longitudes
  longitudes <- allline[6]
  longitudes <- unlist(strsplit(longitudes, " ", fixed = TRUE))
  longitudes_cleaned <- longitudes[nzchar(longitudes)]
  longitudes_isol <- as.numeric(gsub("[WE]", "", longitudes_cleaned))
  for (i in seq_along(longitudes_cleaned)) {
    if (grepl("W", longitudes_cleaned[i])) {
      longitudes_isol[i] <- -longitudes_isol[i]
    }
  }

  # Extract latitudes
  latitude_raw <- allline[8:length(allline)]
  latitude_isol <- substr(latitude_raw, start = 2, stop = 5)
  new_latitude_isol <- gsub("[NS]", "", latitude_isol)
  new_latitude_isol <- as.numeric(new_latitude_isol)
  for (i in seq_along(latitude_isol)) {
    if (grepl("S", latitude_isol[i])) {
      new_latitude_isol[i] <- -new_latitude_isol[i]
    }
  }

  # Extract cloud high values
  data <- allline[8:length(allline)]
  valuesraw <- substr(data, start = 13, stop = nchar(data))
  valuesplit <- unlist(strsplit(valuesraw, " ", fixed = TRUE))
  valuesplit_cleaned <- as.numeric(valuesplit[nzchar(valuesplit)])

  # Create data frame
  df <- data.frame(Date = rep(date_as_date, each = length(valuesplit_cleaned)),
                   Latitude = rep(new_latitude_isol, each = length(longitudes_isol)),
                   Longitude = rep(longitudes_isol, times = length(new_latitude_isol)),
                   CloudHigh = valuesplit_cleaned)

  return(df)
}
```

After this I cross validated that the date across cloudhigh1, cloudlow1, cloudmid1, ... were the same and that the combinations of latitude and longitude were the same across all files. Since they were the same for the remaining variables I decided to just extract the data point and cbind the column to the dataframe that I made for cloudhigh to avoid recreating the date, latitude, and longitude columns. I utilized the exact same logic as the original function.

```r
# getting the data for the other variables
process_data_file <- function(file_path, variable_name) {
  allline <- readLines(file_path, n = -1)
  # Extract data values
  data <- allline[8:length(allline)]
  valuesraw <- substr(data, start = 13, stop = nchar(data))
  valuesplit <- unlist(strsplit(valuesraw, " ", fixed = TRUE))
  valuesplit_cleaned <- as.numeric(valuesplit[nzchar(valuesplit)])
  # Create data frame
  df <- data.frame(valuesplit_cleaned)
  names(df) <- variable_name
  return(df)
}
```

To read all 72 files for each variable, I created a for loop to read all the files and put all the 72 dataframes into a list to finally rbind them at the end. I validated that I had the right number of rows by making sure each df had 41,472 rows because each file had 576 data points and there were 72 files.

```r
all_data <- list()
# Loop over all files from cloudhigh1.txt to cloudhigh72.txt
for (i in 1:72) {
  # Construct file path for cloudhigh files
  file_path_high <- paste0("~/Downloads/NASA/cloudhigh", i, ".txt")
  # Process the current file and store the result in the list
  all_data[[i]] <- process_cloudhigh_file(file_path_high)
}
# Combine all data frames into a single data frame
final_data <- do.call(rbind, all_data)
# Print the final data frame
```

I repeated this for the other variables and cbinded the column to a bigger final data frame. I attempted to make this more efficient with a nested for loop but I continuously ran into errors and was unable to efficiently do it. So I just went over each variable individually.

```r
cloudlow_data <- list()
# Loop over all files from cloudlow1.txt to cloudlow72.txt
for (i in 1:72) {
  # Construct file path for cloudlow files
  file_path_low <- paste0("~/Downloads/NASA/cloudlow", i, ".txt")
  # Process the current file and store the result in the list
  cloudlow_data[[i]] <- process_data_file(file_path_low, "CloudLow")
}
# Combine all data frames into a single data frame
low_data_combined <- do.call(rbind, cloudlow_data)
# Print the final data frame
finaldata <- cbind(final_data, low_data_combined)

cloudmid_data <- list()
for (i in 1:72) {
  file_path_mid <- paste0("~/Downloads/NASA/cloudmid", i, ".txt")
  cloudmid_data[[i]] <- process_data_file(file_path_mid, "CloudMid")
}
mid_data_combined <- do.call(rbind, cloudmid_data)
finaldata <- cbind(finaldata, mid_data_combined)

ozone_data <- list()
for (i in 1:72) {
  file_path_ozone <- paste0("~/Downloads/NASA/ozone", i, ".txt")
  ozone_data[[i]] <- process_data_file(file_path_ozone, "Ozone")
}
ozone_data_combined <- do.call(rbind, ozone_data)
finaldata <- cbind(finaldata, ozone_data_combined)


pressure_data <- list()
for (i in 1:72) {
  file_path_pressure <- paste0("~/Downloads/NASA/pressure", i, ".txt")
  pressure_data[[i]] <- process_data_file(file_path_pressure, "Pressure")
}
pressure_data_combined <- do.call(rbind, pressure_data)
finaldata <- cbind(finaldata, pressure_data_combined)

surftemp_data <- list()
for (i in 1:72) {
  file_path_surftemp <- paste0("~/Downloads/NASA/surftemp", i, ".txt")
  surftemp_data[[i]] <- process_data_file(file_path_surftemp, "SurfTemp")
}
surftemp_data_combined <- do.call(rbind, surftemp_data)
finaldata <- cbind(finaldata, surftemp_data_combined)

temperature_data <- list()
for (i in 1:72) {
  file_path_temperature <- paste0("~/Downloads/NASA/temperature", i, ".txt")
  temperature_data[[i]] <- process_data_file(file_path_temperature, "Temperature")
}
temperature_data_combined <- do.call(rbind, temperature_data)
finaldata <- cbind(finaldata, temperature_data_combined)
```

While running this sometimes I would get an error about NA values in which I looked into a bit more using warnings() and printing out each variable before and after I manipualte something to debug the code. I was running into issues bc the cloudlow data had NA value but I just left them as NA on the table as told to after looking at the Piazza. I used chatgpt to help replicate the process and that is how I originally came across the paste0() function. I wasn't familiar to it prior to this assignment.

I was able to create a full dataframe with all the variables with 41,472 rows. After that I looked into adding the elevation column to the data frame by checking if read.table() worked first and it did so I cbinded the values after creating a vector.

```r
#adding elevation values to the dataframe
elevationraw = read.table("~/Downloads/NASA/intlvtn.dat", header = TRUE)
e_vals = as.vector(t(elevationraw))
finaldata = cbind(finaldata, Elevation = e_vals)
head(finaldata)
```

```
##         Date Latitude Longitude CloudHigh CloudLow CloudMid Ozone Pressure
## 1 1995-01-16     36.2    -113.8      26.0      7.5     34.5   304      835
## 2 1995-01-16     36.2    -111.2      23.0      7.0     32.0   306      810
## 3 1995-01-16     36.2    -108.8      23.0      7.0     32.0   306      810
## 4 1995-01-16     36.2    -106.2      17.0      7.0     29.5   294      775
## 5 1995-01-16     36.2    -103.8      19.5     11.0     33.0   308      795
## 6 1995-01-16     36.2    -101.2      17.0     14.5     34.0   310      915
##   SurfTemp Temperature Elevation
## 1    272.7       272.1   1526.25
## 2    270.9       270.3   1759.56
## 3    270.9       270.3   1948.38
## 4    269.7       270.9   2241.31
## 5    273.2       271.5   1692.75
## 6    275.6       275.6    865.19
```

Other general methods of debugging I used was checking the dimensions of a dataframe I made and checking the class of variables as I went along.

Compute the median for surftemp for each date across all locations (longitude and latitude pairs).

```r
#install.packages("dplyr")
library(dplyr)
```

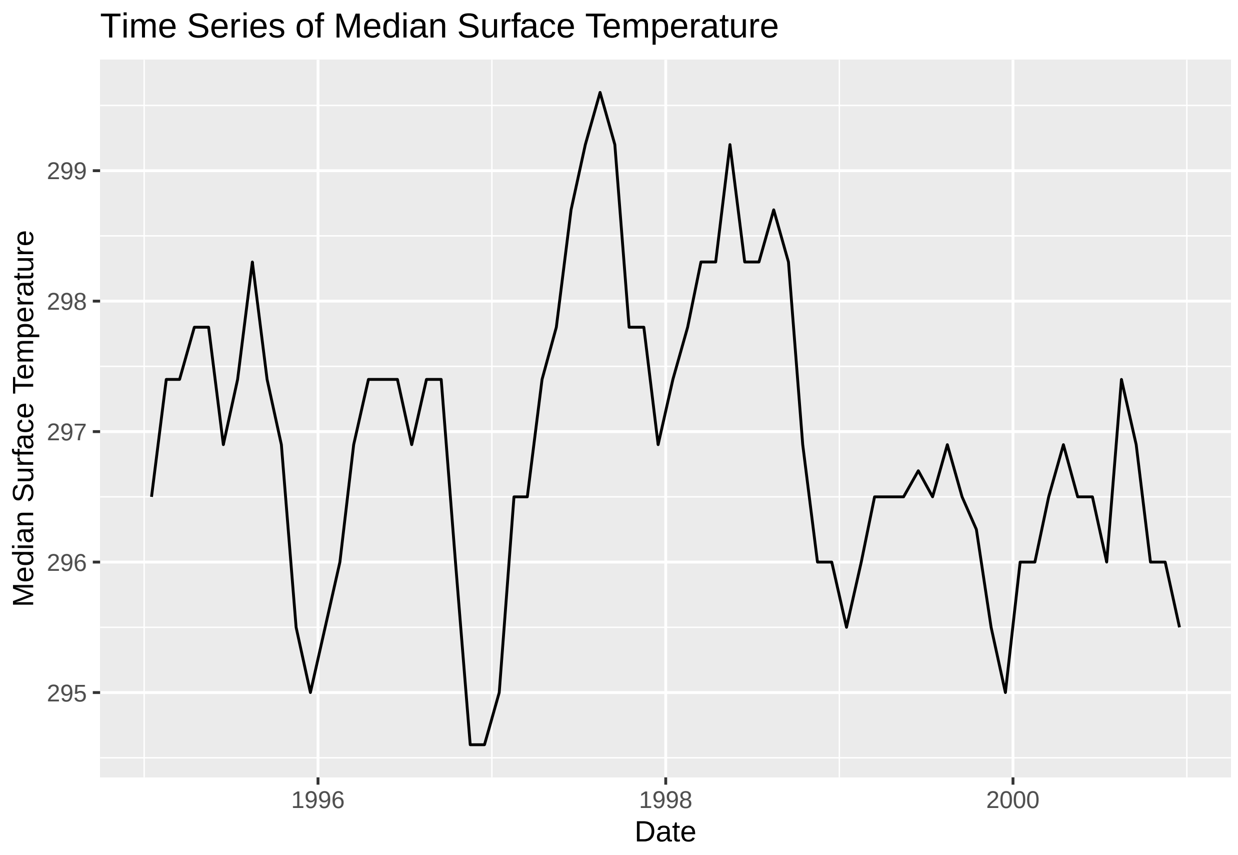
```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
median_surf_temp <- finaldata %>%
  group_by(Date) %>%
  summarise(Median_SurfTemp = median(SurfTemp, na.rm = TRUE))
print(median_surf_temp)
```

```
## # A tibble: 72 x 2
##    Date       Median_SurfTemp
##    <date>               <dbl>
##  1 1995-01-16            296.
##  2 1995-02-16            297.
##  3 1995-03-16            297.
##  4 1995-04-16            298.
##  5 1995-05-16            298.
##  6 1995-06-16            297.
##  7 1995-07-16            297.
##  8 1995-08-16            298.
##  9 1995-09-16            297.
## 10 1995-10-16            297.
## # i 62 more rows
```

I utilized dplyr because it is the simplest way to run a query in R and I have experience using it for other projects in R. The median across the dates are all around the same range and nothing stood out in pariticular.

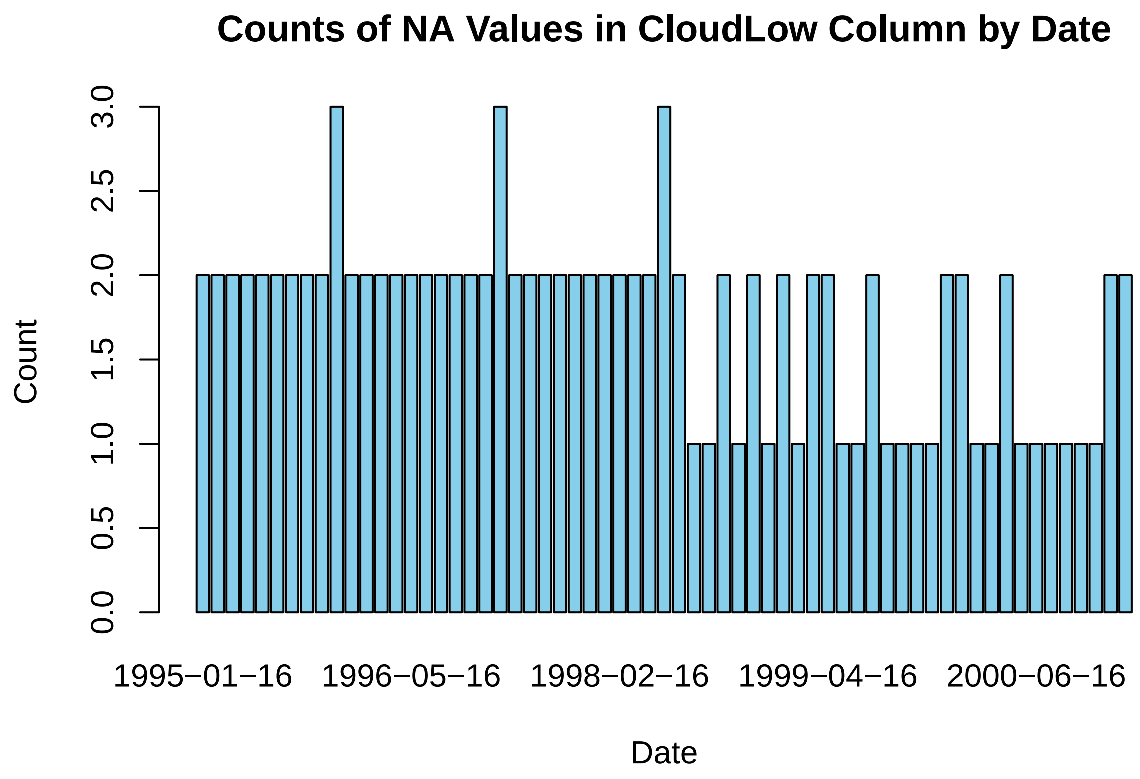
```r
library(ggplot2)

ggplot(median_surf_temp, aes(x = Date, y = Median_SurfTemp)) +
  geom_line() +
  labs(x = "Date", y = "Median Surface Temperature") +
  ggtitle("Time Series of Median Surface Temperature")
```



I used ggplot to plot the time series because that I have experience using this package in multiple statistics classes and it is the most visually clean rather than using the base R to visualize it.

How many missing values are there? In which variables? Are they at the same locations? How are they distributed over time?

```r
cloudlow_na <- is.na(finaldata$CloudLow)

dates_with_na <- finaldata$Date[cloudlow_na]

date_counts <- table(dates_with_na)

barplot(date_counts,
        main = "Counts of NA Values in CloudLow Column by Date",
        xlab = "Date",
        ylab = "Count",
        col = "skyblue")
```



all of the counts added up back to 110 which is the original number of NA values from cloudlow. All the other variables didn't have NA values.

```
# get the locations of the missing values
missing_cloudlow <- finaldata[is.na(finaldata$CloudLow), ]
head(missing_cloudlow)
```

```
##            Date Latitude Longitude CloudHigh CloudLow CloudMid Ozone Pressure
## 547  1995-01-16     18.8     -68.8        31       NA     29.5   254      630
## 548  1995-01-16     18.8     -66.2        31       NA     29.5   254      630
## 1123 1995-02-16     18.8     -68.8        23       NA     26.0   248      630
## 1124 1995-02-16     18.8     -66.2        23       NA     26.0   248      630
## 1699 1995-03-16     18.8     -68.8        19       NA     31.0   254      640
## 1700 1995-03-16     18.8     -66.2        19       NA     31.0   254      640
##      SurfTemp Temperature Elevation
## 547     288.8       283.2   3976.75
## 548     288.8       283.2   3756.50
## 1123    289.3       283.7   3976.75
## 1124    289.3       283.7   3756.50
## 1699    288.3       282.7   3976.75
## 1700    288.3       282.7   3756.50
```

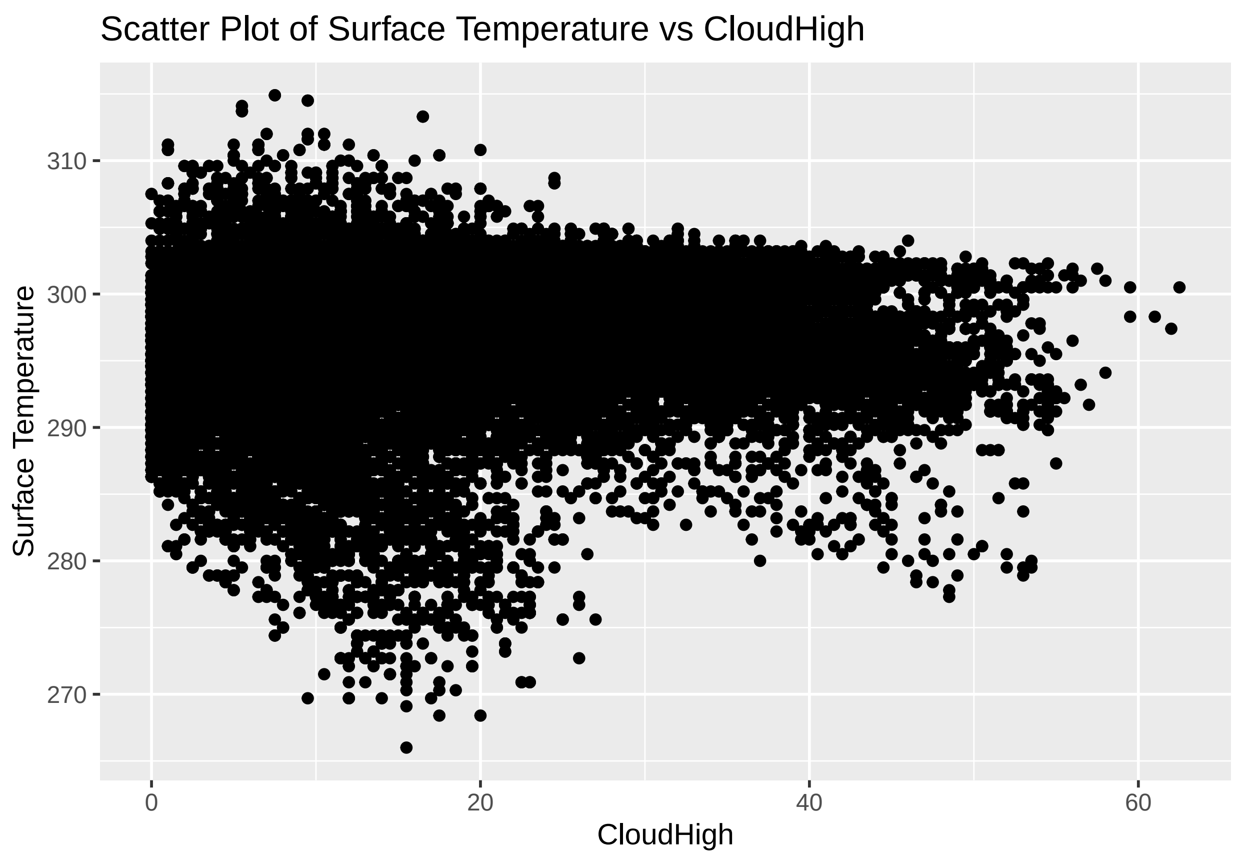
```
table(missing_cloudlow$Longitude, missing_cloudlow$Latitude)
```

```
##
```

```
##          16.2 18.8 21.2
##    -68.8   15   32    0
##    -66.2    0   32   31
```

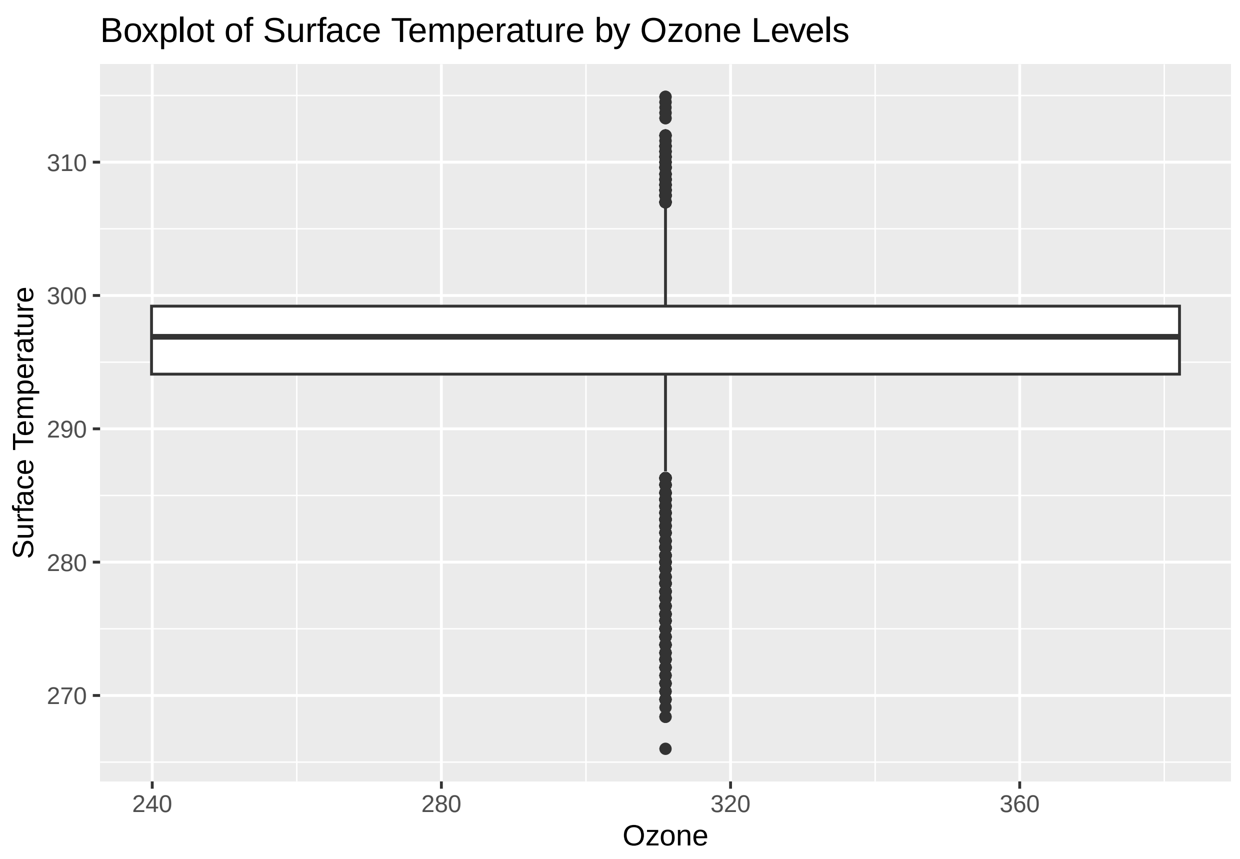
Are there any relationships between surface temperature (SurfTemp) and other variables, such as cloud cover (CloudHigh, CloudLow, CloudMid), ozone levels (Ozone), or pressure (Pressure)?

```
library(ggplot2)
library(dplyr)

# summary statistics
summary_stats <- finaldata %>%
  summarise(across(c(SurfTemp, CloudHigh, CloudLow, CloudMid, Ozone, Pressure), list(mean = mean, mediar

# Scatter plot of SurfTemp vs CloudHigh
ggplot(finaldata, aes(x = CloudHigh, y = SurfTemp)) +
  geom_point() +
  labs(x = "CloudHigh", y = "Surface Temperature", title = "Scatter Plot of Surface Temperature vs Clouc
```



```
# Boxplot of SurfTemp by Ozone levels
ggplot(finaldata, aes(x = Ozone, y = SurfTemp)) +
  geom_boxplot() +
  labs(x = "Ozone", y = "Surface Temperature", title = "Boxplot of Surface Temperature by Ozone Levels")
```

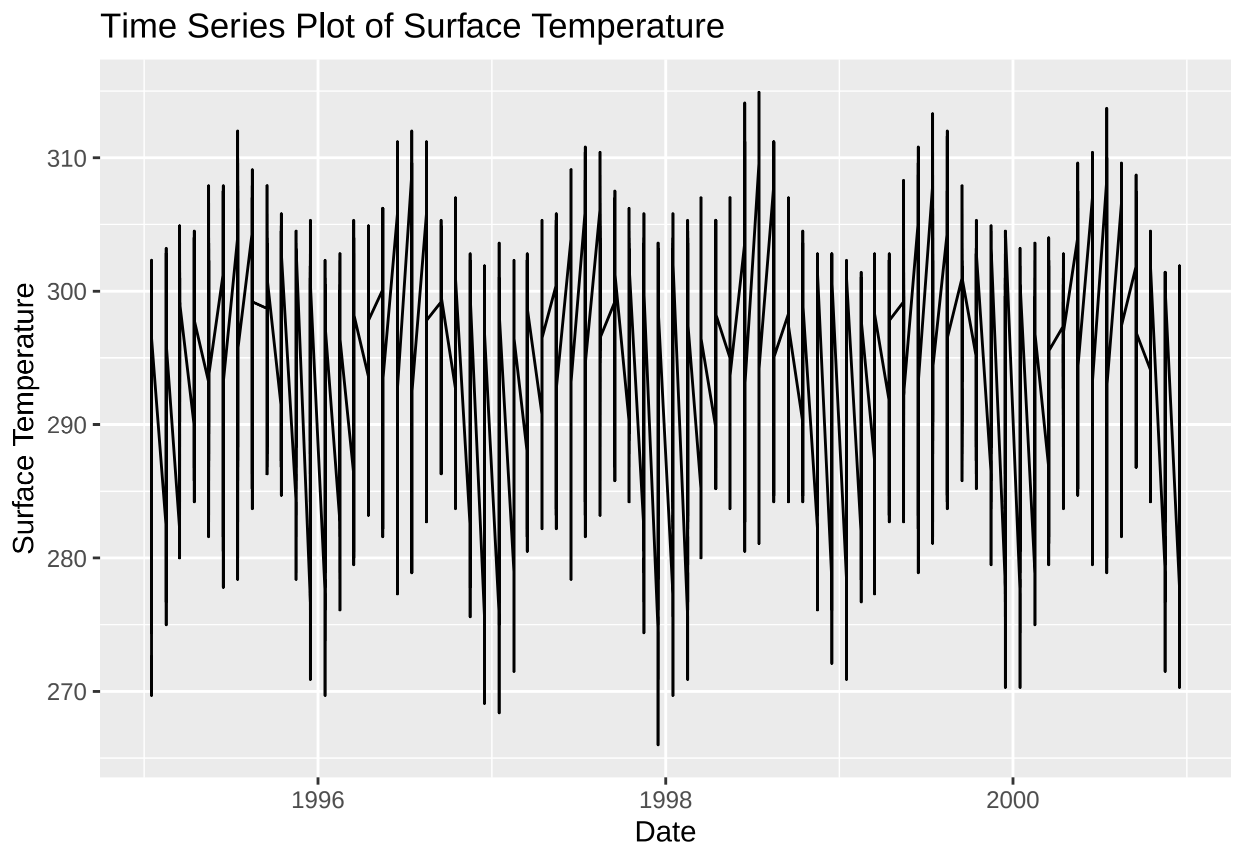
```
## Warning: Continuous x aesthetic
## i did you forget ‘aes(group = ...)‘?
```



```r
# Pearson correlation between SurfTemp and other variables
correlation_matrix <- cor(finaldata[, c("SurfTemp", "CloudHigh", "CloudLow", "CloudMid", "Ozone", "Pres
correlation_matrix
```

```
##              SurfTemp   CloudHigh    CloudLow    CloudMid       Ozone
## SurfTemp   1.00000000  0.03723012 -0.07084080 -0.31038715 -0.21335357
## CloudHigh  0.03723012  1.00000000 -0.56476722  0.62437510 -0.08004675
## CloudLow  -0.07084080 -0.56476722  1.00000000 -0.43694825  0.01580934
## CloudMid  -0.31038715  0.62437510 -0.43694825  1.00000000 -0.02450518
## Ozone     -0.21335357 -0.08004675  0.01580934 -0.02450518  1.00000000
## Pressure   0.33723962 -0.09388569  0.31380041 -0.21528966 -0.07777430
##              Pressure
## SurfTemp   0.33723962
## CloudHigh -0.09388569
## CloudLow   0.31380041
## CloudMid  -0.21528966
## Ozone     -0.07777430
## Pressure   1.00000000
```

```r
# Time series plot of SurfTemp
ggplot(finaldata, aes(x = Date, y = SurfTemp)) +
  geom_line() +
  labs(x = "Date", y = "Surface Temperature", title = "Time Series Plot of Surface Temperature")
```



I looked at some other interesting realtionships between different variables within the dataframe and came across many intereseting observations. Like the Surface Temp going to a cycle every year and shows that the temperature rises and lowers with the time of year. I also tried to see if any of the variables were highly correlated with one another which is only the case between the cloud cover variables.

Through the exploration of the data, several key insights have been learned. Seasonal trends in surface temperature reveal cyclical fluctuations, suggesting a strong correlation with seasonal changes. Analysis of cloud cover variables indicates potential impacts on temperature regulation, with higher cloud cover possibly leading to cooler temperatures. However, no significant relationship was found between ozone levels and surface temperature. While missing data in the cloud cover variable raises concerns about data quality, it did not significantly affect the overall analysis with there being plenty of other data to rely on. I think regardless there should be more information on why there are multiple missing values in specifically those four locations and only in cloudlow but not the other cloud cover variables.

The correlation analysis highlights the interdependence between cloud cover variables, emphasizing the need for comprehensive assessments of multiple factors when studying temperature dynamics. Overall, this exploration underscores the complexity of climate interactions and the importance of continued research to deepen our understanding.