Exploratory Data Analysis

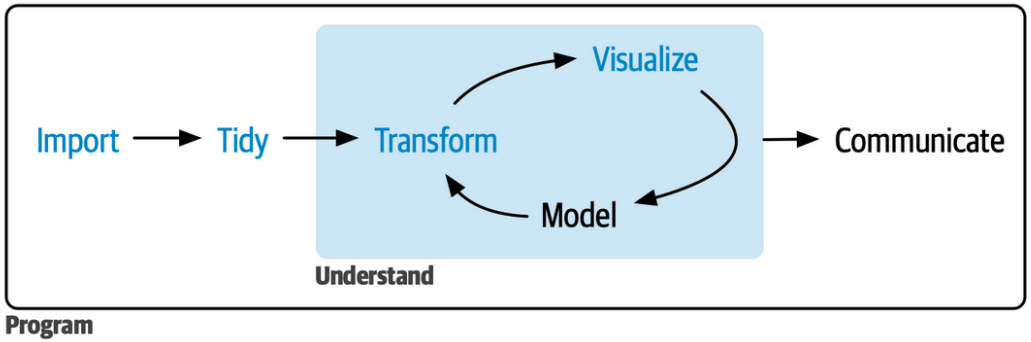
JSC 370: Data Science II

January 22, 2024

Exploratory Data Analysis

- Exploratory data analysis is the process of summarizing data
- It should be the first step in your analysis pipeline
- It involves:
    - 1) checking data (import issues, outliers, missing values, data errors)
    - 2) cleaning data
    - 3) summary statistics of key variables (univariate and bivariate)
    - 4) basic plots and graphs

Pipeline



From: R for Data Science (2e) by Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund

EDA involves Import -> Tidy -> Transform -> Visualize. Basically it is

everything before we do modeling, prediction or inference.

EDA Checklist

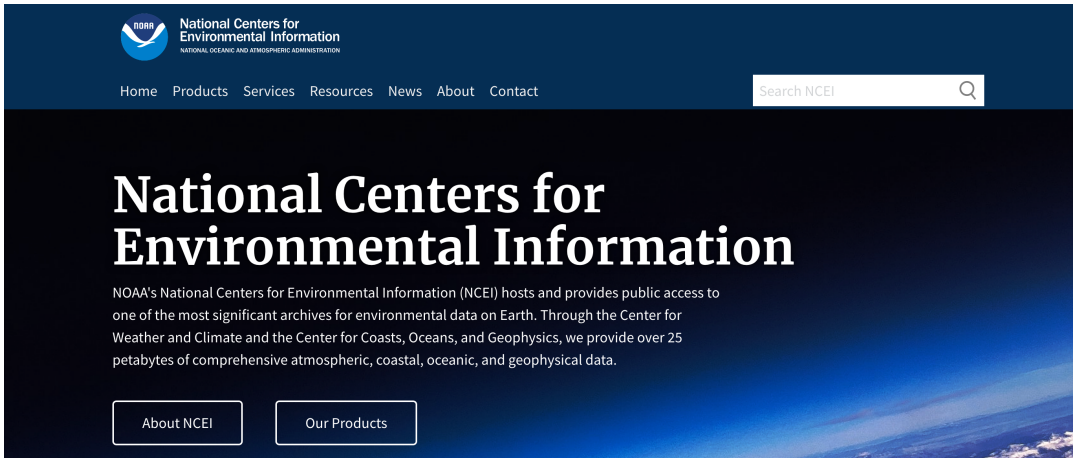
The goal of EDA is to better understand your data. Let's use the checklist:

- Formulate a question
- Read in the data
- Check the dimensions and headers and footers of the data
- Check the variable types in the data
- Take a closer look at some/all of the variables
- Validate with an external source

- Conduct some summary statistics to answer the initial question
- Make exploratory graphs

Case study

We are going to use a dataset created from the National Center for Environmental Information (https://www.ncei.noaa.gov/). The data are 2019 hourly measurements from weather stations across the continental U.S.



Formulate a Question

It is a good idea to first have a question such as:

- what weather stations reported the hottest and coldest daily temperatures?

- what day of the month was on average the hottest?

- is there covariation between temperature and humidity in my dataset?

Read in the Data

There are several ways to read in data (some depend on the type of data you have):

- `read.table` or `read.csv` in base R for delimited files
- `readRDS` if you have a .rds dataset
- `read_csv`, `read_csv2`, `read_delim`, `read_fwf` from `library(readr)` that is part of the tidyverse
- `readxl()` from `library(readxl)` for .xls and .xlsx files
- `read_sas`, `read_spss`, `read_stata` from `library(haven)`

- `fread` from `library(data.table)` for efficiently importing large datasets that are regular delimited files

Read in the Data

We will focus on base R, `data.table` and the `tidyverse`. Let's load the libraries we need to read in the data:

```
library(data.table)
library(tidyverse)
```

Let's load in the data with `data.table`. I have it stored locally, but we will see how to load it straight from github in the lab.

```
met <- data.table::fread("met_all.gz")
```

Read in the Data

`data.table` is a more efficient version of base R's `data.frame`. We create a `data.table` from an external data source by reading it in using `fread()`

We can convert existing `data.frame` or `list` objects to `data.table` by using `setDT()`

The nice thing about `data.table` is that it handles large datasets efficiently and it never reads/converts character type variables to factors, which can be a pain with `data.frame`

The other nice thing about `data.table` is that many of the base R functions work just as if it was a `data.frame`

Check the data

We should check the dimensions of the data set. This can be done several ways:

```
 dim(met)
```

```
## [1] 2377343      30
```

```
 nrow(met)
```

```
## [1] 2377343
```

```
 ncol(met)
```

```
## [1] 30
```

Check the data

- We see that there are 2,377,343 records of hourly temperature in August 2023 from all of the weather stations in the US. The data set has 30 variables. * We should also check the top and bottom of the dataset to make sure that it imported correctly. Use `head(met)` and `tail(met)` for this.

- Next we can take a deeper dive into the contents of the data with `str()`

Check variables

```
## Classes 'data.table' and 'data.frame':    2377343 obs. of  30 variables:
##  $ USAFID           : int  690150 690150 690150 690150 690150 690150 690150 690150 690150 690150 ..
##  $ WBAN             : int  93121 93121 93121 93121 93121 93121 93121 93121 93121 93121 ...
##  $ year             : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ month            : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ day              : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hour             : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ min              : int  56 56 56 56 56 56 56 56 56 56 ...
##  $ lat              : num  34.3 34.3 34.3 34.3 34.3 34.3 34.3 34.3 34.3 34.3 ...
##  $ lon              : num  -116 -116 -116 -116 -116 ...
##  $ elev             : int  696 696 696 696 696 696 696 696 696 696 ...
##  $ wind.dir         : int  220 230 230 210 120 NA 320 10 320 350 ...
##  $ wind.dir.qc      : chr  "5" "5" "5" "5" ...
##  $ wind.type.code   : chr  "N" "N" "N" "N" ...
##  $ wind.sp          : num  5.7 8.2 6.7 5.1 2.1 0 1.5 2.1 2.6 1.5 ...
##  $ wind.sp.qc       : chr  "5" "5" "5" "5" ...
```

```
##  $ ceiling.ht       : int  22000 22000 22000 22000 22000 22000 22000 22000 22000 22000 ...
##  $ ceiling.ht.qc    : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ ceiling.ht.method: chr  "9" "9" "9" "9" ...
```

Check variables

- First, we see that `str()` gives us the class of the data, which in this case is both `data.table` and `data.frame`, as well as the dimensions of the data

- We also see the variable names and their type (integer, numeric, character)

- We can identify major problems with the data at this stage (e.g. a variable that has all missing values)

We can get summary statistics on our `data.table` using `summary()` as we would with a regular `data.frame`

Check variables

```
##       lat             lon        
##  Min.   :24.55   Min.   :-124.29  
##  1st Qu.:33.97   1st Qu.: -98.02  
##  Median :38.35   Median : -91.71  
##  Mean   :37.94   Mean   : -92.15  
##  3rd Qu.:41.94   3rd Qu.: -82.99  
##  Max.   :48.94   Max.   : -68.31  
##                                   
##       elev          wind.dir    
##  Min.   : -13.0   Min.   :  3   
##  1st Qu.: 101.0   1st Qu.:120   
##  Median : 252.0   Median :180   
##  Mean   : 415.8   Mean   :185   
##  3rd Qu.: 400.0   3rd Qu.:260   
##  Max.   :9999.0   Max.   :360   
##                   NA's   :785290
```

```
##  wind.dir.qc        wind.type.code
##  Length:2377343     Length:2377343
##  Class :character   Class :character
```

Check variables more closely

We know that we are supposed to have hourly measurements of weather data for the month of August 2019 for the entire US. We should check that we have all of these components. Let us check:

- the year
- the month
- the hours
- the range of locations (latitude and longitude)

Check variables more closely

We can generate tables for hour (top) and month (bottom) for integer values using `table()`

```
##
##      0      1      2      3      4      5      6
##  99434  93482  93770  96703 110504 112128 106235
##      7      8      9     10     11     12     13
## 101985 100310 102915 101880 100470 103605  97004
##     14     15     16     17     18     19     20
##  96507  97635  94942  94184 100179  94604  94928
##     21     22     23
##  96070  94046  93823

##
##       8
```

```
## 2377343
```

Check variables more closely

For numeric values we should do a summary to see the quantiles, max, min

```
 table(met$year)
```

```
##
##    2019
## 2377343
```

```
 summary(met$lat)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   24.55   33.97   38.35   37.94   41.94   48.94
```

```
 summary(met$lon)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -124.29  -98.02  -91.71  -92.15  -82.99  -68.31
```

Check variables more closely

If we return to our initial question, what weather stations reported the hottest and coldest temperatures, we should take a closer look at our key variable, temperature (temp)

```
summary(met$temp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -40.00   19.60   23.50   23.59   27.80   56.00
##    NA's
##   60089
```

It looks like the temperatures are in Celsius. A temperature of -40 in August is really cold, we should see if this is an implausible value.

Check variables more closely

It also looks like there are a lot of missing data. Let us check the proportion of missings

```
mean(is.na(met$temp))
```

```
## [1] 0.0252757
```

2.5% of the data are missing, which is not a huge amount.

Check variables more closely

In `data.table` we can easily subset the data and select certain columns

```
met_ss <- met[temp == -40.00, .(hour, lat, lon, elev, wind.sp)]
```

```
dim(met_ss)
```

```
## [1] 36  5
```

```
summary(met_ss)
```

```
##       hour            lat             lon        
##  Min.   : 0.00   Min.   :29.12   Min.   :-89.55  
##  1st Qu.: 2.75   1st Qu.:29.12   1st Qu.:-89.55  
##  Median : 5.50   Median :29.12   Median :-89.55  
##  Mean   : 5.50   Mean   :29.12   Mean   :-89.55
```

```
##  3rd Qu.: 8.25   3rd Qu.:29.12   3rd Qu.:-89.55
##  Max.   :11.00   Max.   :29.12   Max.   :-89.55
##
```

Check variables more closely

In `dplyr` we can do the same thing using `filter` and `select`

```
met_ss <- filter(met, temp == -40.00) |>
  select(USAFID, day, hour, lat, lon, elev, wind.sp)
dim(met_ss)
```

```
## [1] 36  7
```

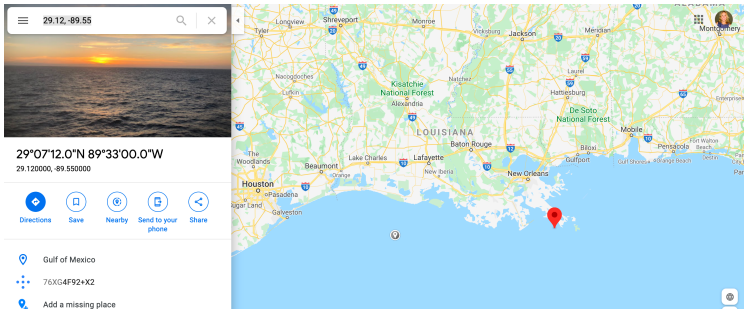
```
names(met_ss)
```

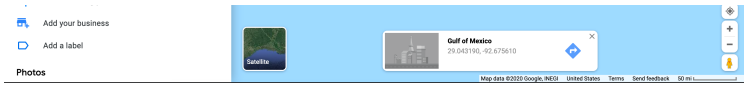
```
## [1] "USAFID"  "day"     "hour"    "lat"
## [5] "lon"     "elev"    "wind.sp"
```

Validate against an external source

We should check outside sources to make sure that our data make sense. For example the observation with -40C is suspicious, so we should look up the location of the weather station.

Go to Google maps and enter the coordinates for the site with -40C (29.12, -89.55)





Summary statistics

If we return to our initial question, we need to generate a list of weather stations that are ordered from highest to lowest. We can then examine the top and bottom of this new dataset. First let us remove the -40C observations and then sort.

In `data.table` we can use `order()` to sort. With over 2 million observations, `data.table` is the best solution since it is more computationally efficient.

```
met <- met[temp > -40]
setorder(met, temp)
```

This is equivalent to

```
met <- met[temp > -10][order(temp)]
```

Summary statistics

```
head(met)[,c(1,8:10,24)]
```

```
##    USAFID    lat      lon elev temp
## 1: 726764 44.683 -111.116 2025 -3.0
## 2: 726764 44.683 -111.116 2025 -3.0
## 3: 726764 44.683 -111.116 2025 -3.0
## 4: 726764 44.683 -111.116 2025 -3.0
## 5: 720411 36.422 -105.290 2554 -2.4
## 6: 726764 44.683 -111.116 2025 -2.0
```

```
tail(met)[,c(1,8:10,24)]
```

```
##    USAFID    lat      lon elev temp
## 1: 720267 38.955 -121.081  467 52.0
## 2: 690150 34.300 -116.166  696 52.8
```

```
## 3: 690150 34.296 -116.162  625 52.8
## 4: 690150 34.300 -116.166  696 53.9
## 5: 690150 34.300 -116.166  696 54.4
```

Summary statistics

The maximum hourly temperature is 56C at site 720267, and the minimum hourly temperature is -3C at site 726764.

Summary statistics

We need to transform our data to answer our initial question. Let's find the daily mean max and min temperatures in `data.table`

```
met_daily <- met[, .(
  temp = mean(temp),
  lat  = mean(lat),
  lon  = mean(lon),
  elev = mean(elev)
  ), by = c("USAFID", "day")][order(temp)]
```

```
head(met_daily)
```

```
##    USAFID day     temp   lat   lon elev
## 1: 726130  11 4.278261 44.27 -71.3 1909
## 2: 726130  31 4.304348 44.27 -71.3 1909
```

```
## 3: 726130  10 4.583333 44.27 -71.3 1909
## 4: 726130  25 4.705882 44.27 -71.3 1909
## 5: 726130  24 5.050000 44.27 -71.3 1909
```

Summary statistics

Let's find the daily mean max and min temperatures in `dplyr`

```
met_daily_dplyr <- met |>
  group_by(USAFID, day) |>
  summarize(temp = mean(temp)) |>
  arrange(desc(temp))
```

(try it yourself)

Summary statistics

The maximum daily temperature is 41.7166667 C at site 690150 and the minimum daily temperature is 4.2782609C at site 726130.

Exploratory graphs

With exploratory graphs we aim to:

- debug any issues remaining in the data
- understand properties of the data
- look for patterns in the data
- inform modeling strategies

Exploratory graphs do not need to be perfect, we will look at presentation ready plots next week.

Exploratory graphs

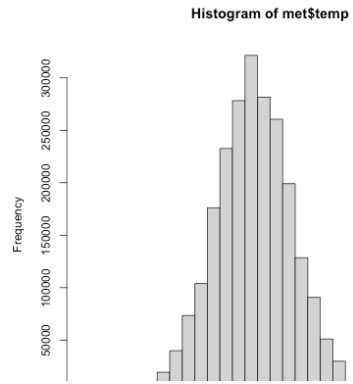
Examples of exploratory graphs include:

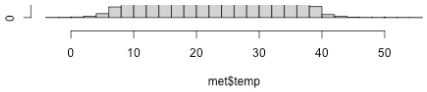
- histograms (for continuous data)
- boxplots (for continuous data)
- scatterplots (for continuous data)
- barplots (for categorical data)
- simple maps (for spatial data)

Exploratory Graphs

Focusing on the variable of interest, temperature, let's look at the distribution (after removing -40C) using a histogram `hist()`

```
hist(met$temp)
```

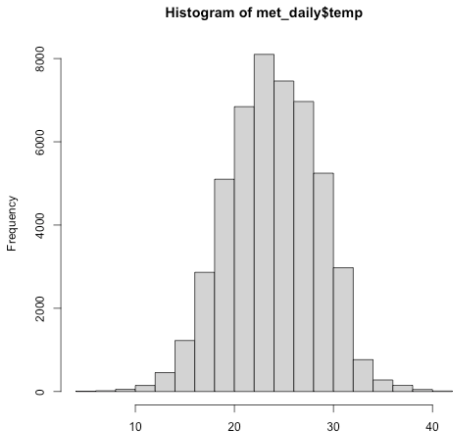




Exploratory Graphs

Let's look at the daily data

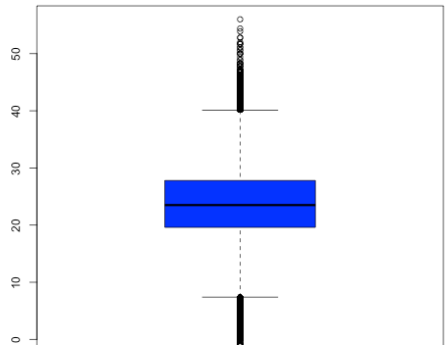
```
hist(met_daily$temp)
```



Exploratory Graphs

A boxplot gives us an idea of the quantiles of the distribution and any outliers

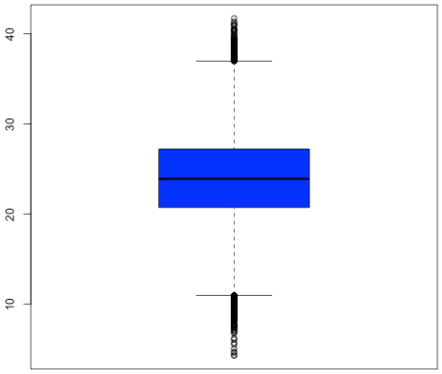
```
boxplot(met$temp, col = "blue")
```



Exploratory Graphs

Let's look at the daily data

```
boxplot(met_daily$temp, col = "blue")
```



Exploratory Graphs

A map will show us where the weather stations are located. First let's get the unique latitutdes and longitudes and see how many meteorological sites there are

```
met_stations <- (unique(met[,c("lat","lon")]))
dim(met_stations)
```

```
## [1] 2827    2
```

Exploratory Graphs

A simple map can be done with the maps library

```
library(maps)
map("world", "usa")
points(met_stations$lon,met_stations$lat, pch=19, col='red')
```

Exploratory Graphs

An interactive map is helpful because it provides a basemap (background)

```
library(leaflet)
l<- leaflet(met_stations) %>%
  addProviderTiles('CartoDB.Positron') %>%
  addCircles(lat = ~lat, lng = ~lon, opacity = 1, fillOpacity = 1, radius = 100)
widgetframe::frameWidget(l)
```

File not found

Firefox can’t find the file at /Users/meredith/Library/CloudStorage/Dropbox/Courses/JSC370/JSC370_2024/slides/03-eda/JSC370-slides-03_files/figure-html//widgets/widget_unnamed-

Exploratory Graphs - Mapping

Let's map the locations of the max and min daily temperatures.

```
min <- met_daily[1]  # First observation.
max <- met_daily[.N] # Last obs, .N is a special symbol in data.table

leaflet() %>%
  addProviderTiles('CartoDB.Positron') %>%
  addCircles(
    data = min,
    lat = ~lat, lng = ~lon, popup = "Min temp.",
    opacity = 1, fillOpacity = 1, radius = 400, color = "blue"
    ) %>%
  addCircles(
    data = max,
    lat = ~lat, lng = ~lon, popup = "Max temp.",
    opacity=1, fillOpacity=1, radius = 1400, color = "red"
```

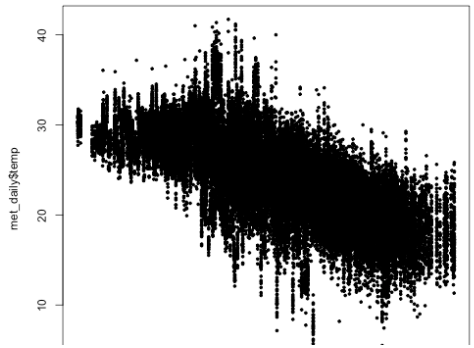
)

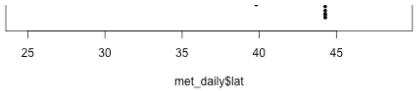
Mapping

Exploratory Graphs

Scatterplots help us look at pairwise relationships. Let's see if there is any trend in temperature with latitude

```
plot(met_daily$lat, met_daily$temp, pch=19, cex=0.5)
```

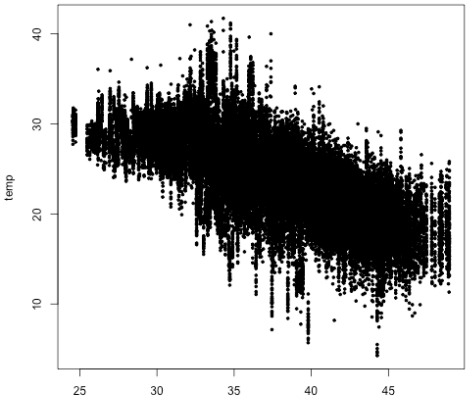




Exploratory Graphs

This is equivalent to

```
met_daily[, plot(lat, temp, pch = 19, cex =0.5)]
```



lat

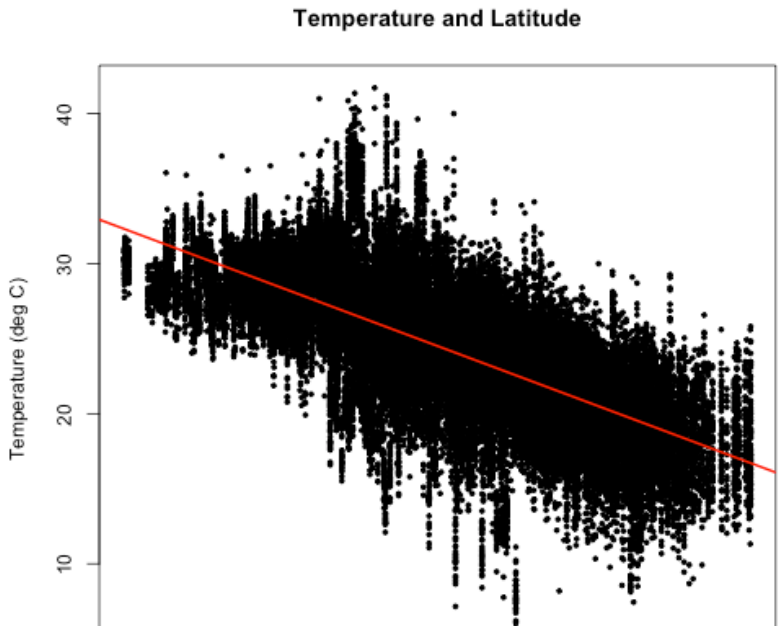
There is a clear decrease in temperatures as you increase in latitude (i.e as you go north)

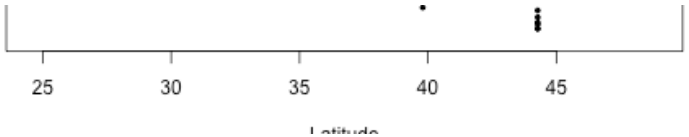
Exploratory Graphs

We can add a simple linear regression line to this plot using `lm()` and `abline()`. We can also add a title and change the axis labels.

```
mod <- lm(temp ~ lat, data = met_daily)
met_daily[, plot(
  lat, temp, pch=19, cex=0.5,
  main = "Temperature and Latitude",
  xlab = "Latitude", ylab = "Temperature (deg C)")
  abline(mod, lwd=2, col="red")
```

Exploratory Graphs

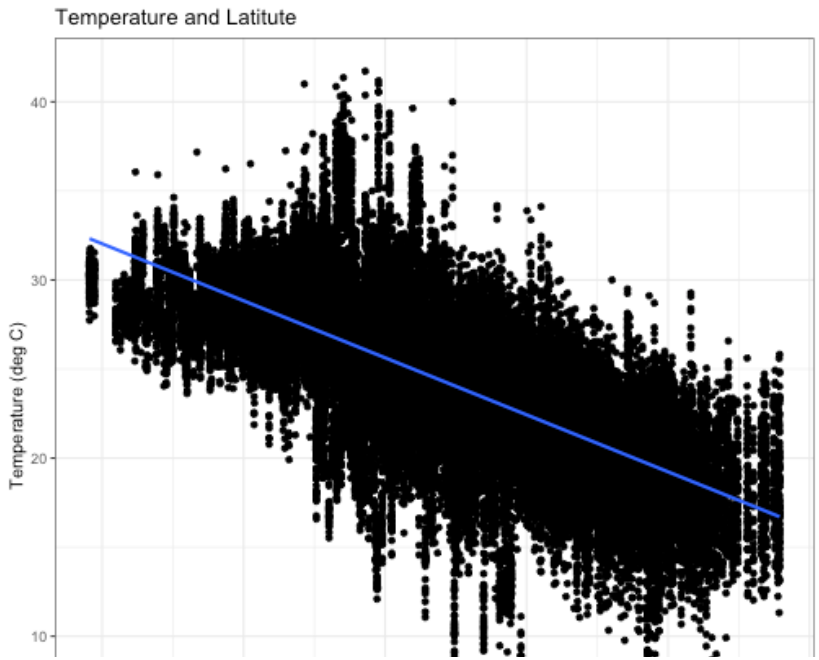


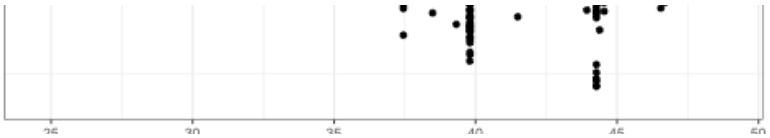


Using **ggplot2** (next class)

```
library(ggplot2)
ggplot(data = met_daily, mapping = aes(x = lat, y = temp)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Temperature and Latitute", xlab  = "Latitute", y = "Temperature (deg C)")+
  theme_bw()
```

Using **`ggplot2`** (next class)





Summary

In EDA we:

- have an initial question that we aim to answer
- import, check, clean the data
- perform any data transformations to answer the initial question
- make some basic graphs to examine the data and visualize the initial question