Exploratory Data Analysis

PM 566: Introduction to Health Data Science

Kelly Street

Acknowledgment

These slides were originally developed by Meredith Franklin. They have been modified by George G. Vega Yon and Kelly Street.

Exploratory Data Analysis

- Exploratory data analysis is the process of becoming familiar with a dataset
- It should be the first step in your analysis pipeline
- It involves:
  - checking data (import issues, outliers, missing values, data errors)
  - cleaning data
  - summary statistics of key variables (univariate and bivariate)
  - basic plots and graphs

Exploratory Data Analysis

> Since our eyes and brains are not wired to detect patterns in large data tables filled with text and numbers, communication about data […] rarely comes in the form of raw data or code output. Instead, data and data-driven results are usually either summarized (e.g., using an average/mean) and presented in small summary tables or they are presented visually in the form of graphs, in which shape, distance, color, and size can be used to represent the magnitudes of (and relationships between) the values within our data.

- *Viridical Data Science*, Yu and Barter

Key Questions

- What question(s) am I trying to answer?
- What question(s) *could* this dataset answer?

The Tidyverse Model

Loosely, EDA encompasses the Import -> Tidy -> Transform -> Visualize steps. Basically it is everything before we do modeling, prediction or inference.
EDA may involve some statistical summaries, but it does *not* include formal statistical analysis.

EDA Checklist

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

1. Read in the data
2. Check the size of the data
3. Examine the variables and their types
4. Look at the top and bottom of the data
5. Visualize the distributions of key variables
6. Check your expectations
7. Validate with an external source
8. Formulate a (simple) question
9. Try the easy solution first
10. Challenge your solution

(adapted from **Exploratory Data Analysis with R** by Roger D. Peng)

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:

- Which weather stations reported the hottest and coldest daily temperatures?
- What day of the month was on average the hottest?
- Is there correlation 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 (this is a handy, compressed way of saving R objects)
- `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

There are plenty of ways to do these tasks, but we will focus on base R.

Since our data is stored as a (gzipped) CSV file, we could load it into R with `read.csv`, but we will use the more flexible `read.table`. I have it stored locally, but we will see how to load it straight from GitHub in the lab.

```
met <- read.table('../data/met_all.gz',
                  header = TRUE, sep = ',')
```

We specify that the first line contains column names by setting `header = TRUE` and we indicate that commas are used to separate the different values (rather than tabs, spaces, etc.) by setting `sep = ','`.

Working with `data.frame`s

This gave as a `data.frame` object, which is a standard R format for cleaned, rectangular data. Each row represents an observation and each column represents a variable.

As we have seen, you can access particular parts of the `data.frame` by subsetting with the square brackets, `[,]`. For example, you can pull out the 2nd, 3rd, and 4th elements of the 1st column of our `met` dataset with `met[2:4, 1]`.

You can also pull out specific columns by name, using the `$` operator. Since the first column is called `USAFID`, we could access the same subset as above with `met$USAFID[2:4]` (notice that there is no comma here, because we have already subset down to a single variable).

To see the list of names for the dataset, you can use `names(met)` or `colnames(met)`. To see the top few rows of the dataset, use `head(met)`.

Check the data

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

```r
dim(met)
```

```
[1] 2377343      30
```

```r
nrow(met)
```

```
[1] 2377343
```

```r
ncol(met)
```

```
[1] 30
```

Check the data

- We see that there are 2,377,343 records of hourly temperature in August 2019 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 check for any irregularities. 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

```
str(met)
```

```
'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 ...
```

Check variables

- First, we see that `str()` gives us the class of the data, which in this case is a `data.frame`, as well as the dimensions of the data
- We also see the variable names and their type (integer, numeric, character, etc.)
- We can identify major problems with the data at this stage (e.g. a variable that has all missing values)

Check variables

We can get summary statistics on our `data.frame` using `summary()`.

```
summary(met[,8:13])
```

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

 NA's   :785290
```

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’s check:

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

Check variables more closely

We can generate tables and/or barplots for integer variables:

```
table(met$hour)
```

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

```
table(met$month)
```

```
      8 
2377343 
```

Check variables more closely

We can generate tables and/or barplots for integer variables:

```
barplot(table(met$hour))
```

Check variables more closely

For numeric variables we should do a summary to see the quantiles, min, max, and mean.

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

```
   2019
2377343
```

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

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

```r
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

We can visualize these distributions with a histogram.

```
layout(matrix(1:2, nrow=1))
hist(met$lat)
hist(met$lon)
```

```
layout(1)
```

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`)

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

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

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

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 is a lot of missing data, encoded by `NA` values. Let's check the proportion of missingness by tallying up whether or not every temperature reading is an `NA`. This will give us a vector of `TRUE`/`FALSE` values and then we can take the `mean` (average), because R automatically interprets `TRUE` as `1` and `FALSE` as `0` for mathematical functions.

```r
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 our `data.frame` we can easily subset the data and select certain columns. Here, we select all observations with a temperature of -40C and a specific subset of the variables:

```
met_ss <- met[met$temp == -40.00, c('hour','lat','lon

dim(met_ss)
```

```
[1] 60125     5
```

```
summary(met_ss)
```

```
      hour            lat             lon             elev       
 Min.   : 0.00   Min.   :29.12   Min.   :-89.55   Min.   :36     
 1st Qu.: 2.75   1st Qu.:29.12   1st Qu.:-89.55   1st Qu.:36     
 Median : 5.50   Median :29.12   Median :-89.55   Median :36     
 Mean   : 5.50   Mean   :29.12   Mean   :-89.55   Mean   :36     
 3rd Qu.: 8.25   3rd Qu.:29.12   3rd Qu.:-89.55   3rd Qu.:36     
 Max.   :11.00   Max.   :29.12   Max.   :-89.55   Max.   :36     
 NA's   :60089   NA's   :60089   NA's   :60089    NA's   :60089  
    wind.sp     
 Min.   : NA    
 1st Qu.: NA    
 Median : NA    
 Mean   :NaN    
 3rd Qu.: NA    
 Max.   : NA    
 NA's   :60125  
```

Check variables more closely

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

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

dim(met_ss)
```

```
[1] 36  7
```

```
summary(met_ss)
```

```
     USAFID             day         hour              lat       
lon
 Min.   :720717   Min.   :1   Min.   : 0.00   Min.   :29.12  
Min.   :-89.55  
 1st Qu.:720717   1st Qu.:1   1st Qu.: 2.75   1st Qu.:29.12  
1st Qu.:-89.55  
 Median :720717   Median :1   Median : 5.50   Median :29.12  
Median :-89.55  
 Mean   :720717   Mean   :1   Mean   : 5.50   Mean   :29.12  
Mean   :-89.55  
 3rd Qu.:720717   3rd Qu.:1   3rd Qu.: 8.25   3rd Qu.:29.12  
3rd Qu.:-89.55  
 Max.   :720717   Max.   :1   Max.   :11.00   Max.   :29.12  
Max.   :-89.55  

      elev       wind.sp   
 Min.   :36   Min.   : NA  
 1st Qu.:36   1st Qu.: NA  
 Median :36   Median : NA  
 Mean   :36   Mean   :NaN  
 3rd Qu.:36   3rd Qu.: NA  
 Max.   :36   Max.   : NA  
              NA's   :36  
```

Validate against an external source

We should check outside sources to make sure that our data makes 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)

It doesn’t make much sense to have a -40C reading in the Gulf of Mexico off the coast of Louisiana!

Data cleaning

If we return to our initial question ("Which weather stations reported the hottest and coldest daily temperatures?"), 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 aberrant observations and then we'll sort by temperature.

```
met <- met[met$temp > -40, ]
```

Notice that we do not create a new object, we just overwrite the `met` object. Once you're sure that you want to remove certain observations, this is a good way to avoid confusion (otherwise, it is easy to end up with multiple subsets of the data in your R environment with similar names like `met`, `met_ss`, `met_ss2`, `met_final`, `met_FINAL`, `met_FINAL_REAL`, etc.)

Data cleaning

We will also remove any observations with missing temperature values (`NA`).
The `is.na()` function tells you whether or not a particular value is missing and the `!` operator takes the opposite of a `TRUE`/`FALSE` value, so in combination, they tell you which observations are not missing.

```
met <- met[!is.na(met$temp), ]
```

Sorting

Now, we can use the `order()` function to sort our dataset.

```
met <- met[order(met$temp), ]
```

Again, we just replace the `met` object with this updated version, since we aren't actually losing any data, just changing the order.

Highest and Lowest

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

```
         USAFID    lat    lon elev  temp
1203053 722817 38.767 -104.3 1838 -17.2
1203055 722817 38.767 -104.3 1838 -17.2
1203128 722817 38.767 -104.3 1838 -17.2
1203129 722817 38.767 -104.3 1838 -17.2
1203222 722817 38.767 -104.3 1838 -17.2
1203225 722817 38.767 -104.3 1838 -17.2
```

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

```
       USAFID    lat      lon elev temp
42783 720267 38.955 -121.081  467 52.0
724   690150 34.300 -116.166  696 52.8
749   690150 34.296 -116.162  625 52.8
748   690150 34.300 -116.166  696 53.9
701   690150 34.300 -116.166  696 54.4
42403 720267 38.955 -121.081  467 56.0
```

Summary statistics

The maximum hourly temperature is 56C at site 720267, and the minimum hourly temperature is -17.2C at site 722817.

Summary statistics

We need to transform our data to answer our initial question. Let's find the **daily** mean, max, and min temperatures for each weather station in our `data.frame`. We can do this with the `summarize` function from the `dplyr` package. This package is part of the `tidyverse`, so the syntax is a bit different from what we've seen before.

```
library(dplyr)
met_daily <- summarize(met,
                       temp = mean(temp),
                       lat = mean(lat),
                       lon = mean(lon),
                       elev = mean(elev),
                       .by = c(USAFID, day))
```

What we've done here is told R to summarize the `met` dataset by the variables `USAFID` and `day`, splitting the data into subsets based on those two indexing variables. For each subset (representing a specific station of a specific day), we want the daily average temperature, as well as latitude, longitude, and elevation (though hopefully those don't change too much over the course of a day!)

Summary statistics

Before we continue, check the relative sizes of the `met` and `met_daily` objects. Which one is bigger?

Summary statistics

Now we will order our new dataset by the average daily temperature, just as we ordered the old one by observed temperature.

```
met_daily <- met_daily[order(met_daily$temp), ]

head(met_daily)
```

```
    USAFID day       temp    lat    lon elev
2   722817   3 -17.200000 38.767 -104.3 1838
1   722817   1 -17.133333 38.767 -104.3 1838
3   722817   6 -17.066667 38.767 -104.3 1838
164 726130  11   4.278261 44.270  -71.3 1909
166 726130  31   4.304348 44.270  -71.3 1909
163 726130  10   4.583333 44.270  -71.3 1909
```

```
tail(met_daily)
```

```
      USAFID day     temp      lat       lon     elev
48708 722749   5 40.85714 33.26900 -111.8120 379.0000
48695 723805   5 40.97500 34.76800 -114.6180 279.0000
48721 720339  14 41.00000 32.14600 -111.1710 737.0000
48710 723805   4 41.18333 34.76800 -114.6180 279.0000
48688 722787   5 41.35714 33.52700 -112.2950 325.0000
48438 690150  31 41.71667 34.29967 -116.1657 690.0833
```

Summary statistics

The maximum **daily average** temperature is 41.7166667 C at site 690150 and the minimum daily average temperature is -17.2C at site 722817.

Summary statistics

The code below is similar to our previous example, but doesn't include the latitude, longitude, and elevation. How would you alter this code to find the daily **median**, **max**, or **min** temperatures for each station?

```
summarize(met,
          temp = mean(temp),
          .by = c(USAFID, day))
```

(try it yourself)

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
- boxplots
- scatterplots
- simple maps

Exploratory Graphs

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

```
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 = "blu
```

Exploratory Graphs

We know that these data come from US weather stations, so we might have some idea what to expect just from plotting the latitude and longitude (note that we fix the aspect ratio at 1:1 with `asp = 1`; this prevents the plot from stretching or shrinking to fit the available plotting area):

```
plot(met_daily$lon, met_daily$lat, asp=1)
```

Exploratory Graphs

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

```
met_stations <- (unique(met[,c("la
dim(met_stations)
```

```
[1] 2827    2
```

Exploratory Graphs

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

```
library(leaflet)
leaflet(met_stations) |>
  addProviderTiles('CartoDB.Positr
  addCircles(lat = ~lat, lng = ~lo
             opacity = 1, fillOpac
```

Exploratory Graphs

+

−

Exploratory Graphs

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

```
min <- met_daily[1, ]
max <- met_daily[nrow(met_daily),

leaflet() |>
  addProviderTiles('CartoDB.Positr
  addCircles(
    data = min,
    lat = ~lat, lng = ~lon, popup
    opacity = 1, fillOpacity = 1,
    ) |>
  addCircles(
    data = max,
    lat = ~lat, lng = ~lon, popup
    opacity=1, fillOpacity=1, radi
    )
```

(next slide)

+

−

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=16, cex=0.5)
```

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_c
met_daily[, plot(
  lat, temp, pch=19, cex=0.5,
  main = "Temperature and Latitude
  xlab = "Latitude", ylab = "Tempe
  ]
abline(mod, lwd=2, col="red")
```

(next slide)

Using `ggplot2` (next class)

```
library(ggplot2)
ggplot(data = met_daily, mapping =
  geom_point() + geom_smooth(metho
  labs(title = "Temperature and La
```

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