# Applied_Stat_2_Lab2

---

title: “EDA and data visualization”

author: “Monica Alexander”

date: today

date-format: “DD/MM/YY”

execute:

warning: false

message: false

format:

pdf:

toc: true

number-sections: true

---

# Overview

This week we will be going through some exploratory data analysis (EDA) and data visualization steps in R. The aim is to get you used to some possible steps and tools that you could take to understand the main characteristics and potential issues in a dataset.

We will be using the [‘opendatatoronto‘](https://sharlagelfand.github.io/opendatatoronto/) R package, which interfaces with the City of Toronto Open Data Portal.

A good resource is part 1 (especially chapters 3 and 7) of ‘R for Data Science’ by Hadley Wickham, available for free here: https://r4ds.had.co.nz/.

## What to hand in via GitHub

**There are exercises at the end of this lab**. Please make a new .Rmd/.qmd file with your answers, call it something sensible (e.g. ‘week_2_lab.qmd‘), commit to your git repo from last week (ideally in a ‘labs‘ folder), and push to GitHub. Due on Monday by 9am.

## A note on packages

You may need to install various packages used (using the ‘install.packages‘ function).

Load in all the packages we need:

“‘{r}

#| message: false

library(opendatatoronto)

library(tidyverse)

library(stringr)

library(skimr) # EDA

library(visdat) # EDA

library(janitor)

library(lubridate)

library(ggrepel)

“‘

# TTC subway delays

This package provides an interface to all data available on the [Open Data Portal](https://open.toronto.ca/) provided by the City of Toronto.

Use the ‘list_packages‘ function to look whats available look at what’s available

“‘{r}

all_data <- list_packages(limit = 500)

head(all_data)

“‘

Let’s download the data on TTC subway delays in 2022.

“‘{r}

res <- list_package_resources(“996cfe8d-fb35-40ce-b569-698d51fc683b”) # obtained code from searching data frame above

res <- res |> mutate(year = str_extract(name, “202.?”))

delay_2022_ids <- res |> filter(year==2022) |> select(id) |> pull()

delay_2022 <- get_resource(delay_2022_ids)

# make the column names nicer to work with

delay_2022 <- clean_names(delay_2022)

```

Let's also download the delay code and readme, as reference.

```{r}

# note: I obtained these codes from the 'id' column in the 'res' object above

delay_codes <- get_resource("3900e649-f31e-4b79-9f20-4731bbfd94f7")

delay_data_codebook <- get_resource("ca43ac3d-3940-4315-889b-a9375e7b8aa4")

```

This dataset has a bunch of interesting variables. You can refer to the readme for descriptions. Our outcome of interest is 'min_delay', which give the delay in mins.

```{r}

head(delay_2022)

```

# EDA and data viz

The following section highlights some tools that might be useful for you when you are getting used to a new dataset. There's no one way of exploration, but it's important to always keep in mind:

- what should your variables look like (type, values, distribution, etc)

- what would be surprising (outliers etc)

- what is your end goal (here, it might be understanding factors associated with delays, e.g. stations, time of year, time of day, etc)

In any data analysis project, if it turns out you have data issues, surprising values, missing data etc, it's important you **document** anything you found and the subsequent steps or **assumptions** you made before moving onto your data analysis / modeling.

## Data checks

### Sanity Checks

We need to check variables should be what they say they are. If they aren't, the natural next question is to what to do with issues (recode? remove?)
```

E.g. check days of week

“‘{r}

unique(delay_2022$day)

“‘

Check lines: oh no. some issues here. Some have obvious recodes, others, not so much.

“‘{r}

unique(delay_2022$line)

“‘

The ‘skimr‘ package might also be useful here

“‘{r}

skim(delay_2022)

“‘

### Missing values

Calculate number of NAs by column

“‘{r}

delay_2022 |>

summarize(across(everything(), ~ sum(is.na(.x))))

“‘

The ‘visdat‘ package is useful here, particularly to see how missing values are distributed. (commented out because couldn’t get pdf to render in quarto)

“‘{r}

vis_dat(delay_2022)

#vis_miss(delay_2022)

“‘

### Duplicates?

The ‘get_dupes‘ function from the ‘janitor‘ package is useful for this.

“‘{r}

get_dupes(delay_2022)

“‘

```{r}

delay_2022 <- delay_2022 |> distinct()

```

## Visualizing distributions

Histograms, barplots, and density plots are your friends here.

Let's look at the outcome of interest: 'min_delay'. First of all just a histogram of all the data:

```{r}

## Removing the observations that have non-standardized lines

delay_2022 <- delay_2022 |> filter(line %in% c("BD", "YU", "SHP", "SRT"))

ggplot(data = delay_2022) +

geom_histogram(aes(x = min_delay))

```

To improve readability, could plot on logged scale:

```{r}

ggplot(data = delay_2022) +

geom_histogram(aes(x = min_delay)) +

scale_x_log10()

```

Our initial EDA hinted at an outlying delay time, let's take a look at the largest delays below. Join the 'delay_codes' dataset to see what the delay is. (Have to do some mangling as SRT has different codes).

```{r}

delay_2022 <- delay_2022 |>

left_join(delay_codes |> rename(code = 'SUB RMENU CODE', code_desc = 'CODE DESCRIPTION...3') |> select(code, code_desc))

delay_2022 <- delay_2022 |>

mutate(code_srt = ifelse(line=="SRT", code, "NA")) |>

left_join(delay_codes |> rename(code_srt = 'SRT RMENU CODE', code_desc_srt = 'CODE DESCRIPTION...7') |> select(code_srt, code_desc_srt)) |>
```

mutate(code = ifelse(code_srt==“NA”, code, code_srt),

code_desc = ifelse(is.na(code_desc_srt), code_desc, code_desc_srt)) |>

select(-code_srt, -code_desc_srt)

“‘

The largest delay is due to Fires.

“‘{r}

delay_2022 |>

left_join(delay_codes |> rename(code = ‘SUB RMENU CODE‘, code_desc = ‘CODE DESCRIPTION…3‘) |> select(code, code_desc)) |>

arrange(-min_delay) |>

select(date, time, station, line, min_delay, code, code_desc)

“‘

#### Grouping and small multiples

A quick and powerful visualization technique is to group the data by a variable of interest, e.g. ‘line‘

“‘{r}

ggplot(data = delay_2022) +

geom_histogram(aes(x = min_delay, y = ..density.., fill = line), position = ‘dodge’, bins = 10) +

scale_x_log10()

“‘

I switched to density above to look at the the distributions more comparably, but we should also be aware of differences in frequency, in particular, SHP and SRT have much smaller counts:

“‘{r}

ggplot(data = delay_2022) +

geom_histogram(aes(x = min_delay, fill = line), position = ‘dodge’, bins = 10) +

scale_x_log10()

“‘

If you want to group by more than one variable, facets are good:

```{r}
ggplot(data = delay_2022) +
geom_density(aes(x = min_delay, color = day), bw = .08) +
scale_x_log10() +
facet_wrap(~line)
```

Side note: the station names are a mess. Try and clean up the station names a bit by taking just the first word (or, the first two if it starts with "ST"):

```{r}
delay_2022 <- delay_2022 |>
mutate(station_clean = ifelse(str_starts(station, "ST"), word(station, 1,2), word(station, 1)))
```

## Visualizing time series

Daily plot is messy (you can check for yourself). Let's look by week to see if there's any seasonality. The 'lubridate' package has lots of helpful functions that deal with date variables. First, mean delay (of those that were delayed more than 0 mins):

```{r}
delay_2022 |>
filter(min_delay>0) |>
mutate(week = week(date)) |>
group_by(week, line) |>
summarise(mean_delay = mean(min_delay)) |>
ggplot(aes(week, mean_delay, color = line)) +
geom_point() +
geom_smooth() +
facet_grid(~line)
```

What about proportion of delays that were greater than 10 mins?

```{r}
```

delay_2022 |>

mutate(week = week(date)) |>

group_by(week, line) |>

summarise(prop_delay = sum(min_delay>10)/n()) |>

ggplot(aes(week, prop_delay, color = line)) +

geom_point() +

geom_smooth() +

facet_grid(~line)

‘‘‘

## Visualizing relationships

Note that **scatter plots** are a good precursor to modeling, to visualize relationships between continuous variables. Nothing obvious to plot here, but easy to do with ‘geom_point‘.

Look at top five reasons for delay by station. Do they differ? Think about how this could be modeled.

‘‘‘{r}

delay_2022 |>

group_by(line, code_desc) |>

summarise(mean_delay = mean(min_delay)) |>

arrange(-mean_delay) |>

slice(1:5) |>

ggplot(aes(x = code_desc,

y = mean_delay)) +

geom_col() +

facet_wrap(vars(line),

scales = “free_y”,

nrow = 4) +

coord_flip()

‘‘‘

## PCA (additional)

Principal components analysis is a really powerful exploratory tool, particularly when you have a lot of variables. It allows you to pick up potential clusters and/or outliers that can help to inform model building.

Let's do a quick (and imperfect) example looking at types of delays by station.

The delay categories are a bit of a mess, and there's hundreds of them. As a simple start, let's just take the first word:

```{r}
delay_2022 <- delay_2022 |>
mutate(code_red = case_when(
str_starts(code_desc, "No") ~ word(code_desc, 1, 2),
str_starts(code_desc, "Operator") ~ word(code_desc, 1,2),
TRUE ~ word(code_desc,1))
)
```

Let's also just restrict the analysis to causes that happen at least 50 times over 2022 To do the PCA, the dataframe also needs to be switched to wide format:

```{r}
dwide <- delay_2022 |>
group_by(line, station_clean) |>
mutate(n_obs = n()) |>
filter(n_obs>1) |>
group_by(code_red) |>
mutate(tot_delay = n()) |>
arrange(tot_delay) |>
filter(tot_delay>50) |>
group_by(line, station_clean, code_red) |>
summarise(n_delay = n()) |>
pivot_wider(names_from = code_red, values_from = n_delay) |>
mutate(
across(everything(), ~ replace_na(.x, 0))
```

)

```

Do the PCA:

```{r}

delay_pca <- prcomp(dwide[,3:ncol(dwide)])

df_out <- as_tibble(delay_pca$x)

df_out <- bind_cols(dwide |> select(line, station_clean), df_out)

head(df_out)

```

Plot the first two PCs, and label some outlying stations:

```{r}

ggplot(df_out,aes(x=PC1,y=PC2,color=line )) + geom_point() + geom_text_repel(data = df_out |> filter(PC2>100|PC1<100*-1), aes(label = station_clean))

```

Plot the factor loadings. Some evidence of public v operator?

```{r}

df_out_r <- as_tibble(delay_pca$rotation)

df_out_r$feature <- colnames(dwide[,3:ncol(dwide)])

df_out_r

ggplot(df_out_r,aes(x=PC1,y=PC2,label=feature )) + geom_text_repel()

```

# Lab Exercises

To be handed in via submission of quarto file (and rendered pdf) to GitHub.

1. Using the 'delay_2022' data, plot the five stations with the highest mean delays. Facet the graph by 'line'

2. Restrict the 'delay_2022' to delays that are greater than 0 and to only have delay reasons that appear in the top 50% of most frequent delay reasons. Perform a regression to study the association between delay minutes, and two covariates: line and delay reason. It's up to you how to specify the model, but make sure it's appropriate to the data types. Comment briefly on the results, including whether results generally agree with the exploratory data analysis above.
```

3. Using the ‘opendatatoronto‘ package, download the data on mayoral campaign contributions for 2014 and clean it up. Hints:

- find the ID code you need for the package you need by searching for ‘campaign’ in the ‘all_data‘ tibble above
- you will then need to ‘list_package_resources‘ to get ID for the data file
- note: the 2014 file you will get from ‘get_resource‘ has a bunch of different campaign contributions, so just keep the data that relates to the Mayor election
- clean up the data format (fixing the parsing issue and standardizing the column names using ‘janitor‘)

4. Summarize the variables in the dataset. Are there missing values, and if so, should we be worried about them? Is every variable in the format it should be? If not, create new variable(s) that are in the right format.

5. Visually explore the distribution of values of the contributions. What contributions are notable outliers? Do they share a similar characteristic(s)? It may be useful to plot the distribution of contributions without these outliers to get a better sense of the majority of the data.

6. List the top five candidates in each of these categories:

- total contributions
- mean contribution
- number of contributions

7. Repeat 6 but without contributions from the candidates themselves.

8. How many contributors gave money to more than one candidate?