EDA and data visualization

Xuanze Li

11/03/24

Table of contents

1 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` 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/.

1.1 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.

1.2 A note on packages

You may need to install various packages used (using the `install.packages` function). Load in all the packages we need:

```
library(opendatatoronto)
library(tidyverse)
library(stringr)
library(skimr) # EDA
library(visdat) # EDA
library(janitor)
library(lubridate)
library(ggrepel)
```

2 TTC subway delays

This package provides an interface to all data available on the Open Data Portal provided by the City of Toronto.

Use the `list_packages` function to look whats available look at what's available

```
all_data <- list_packages(limit = 500) # look at all the available datasets
head(all_data)
```

```
# A tibble: 6 x 11
  title              id    topics civic_issues publisher excerpt dataset_category
  <chr>              <chr> <chr>  <chr>        <chr>     <chr>   <chr>
1 Toronto Island F~ toro~ Trans~ <NA>         Parks, F~ "This ~ Table
2 Committee of Adj~ 260e~ City ~ <NA>         City Pla~ "This ~ Table
3 Dinesafe          ea1d~ Publi~ <NA>         Toronto ~ "Snaps~ Table
4 Residential Fron~ 4a65~ Locat~ Mobility,Cl~ Transpor~ "Legal~ Table
5 Property Boundar~ 1aca~ Locat~ <NA>         Informat~ "This ~ Document
6 Lobbyist Registry 6a87~ City ~ <NA>         Lobbyist~ "The L~ Document
# i 4 more variables: num_resources <int>, formats <chr>, refresh_rate <chr>,
```

```
#   last_refreshed <date>
```

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

```
res <- list_package_resources("996cfe8d-fb35-40ce-b569-698d51fc683b") # obtained code from
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.

```
# 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.

```
head(delay_2022)
```

```
# A tibble: 6 x 10
  date                time  day      station code  min_delay min_gap bound line
  <dttm>              <chr> <chr>    <chr>   <chr>     <dbl>   <dbl> <chr> <chr>
1 2022-01-01 00:00:00 15:59 Saturday LAWREN~ SRDP          0       0 N     SRT
2 2022-01-01 00:00:00 02:23 Saturday SPADIN~ MUIS          0       0 <NA>  BD
3 2022-01-01 00:00:00 22:00 Saturday KENNED~ MRO           0       0 <NA>  SRT
4 2022-01-01 00:00:00 02:28 Saturday VAUGHA~ MUIS          0       0 <NA>  YU
5 2022-01-01 00:00:00 02:34 Saturday EGLINT~ MUATC         0       0 S     YU
6 2022-01-01 00:00:00 05:40 Saturday QUEEN ~ MUNCA         0       0 <NA>  YU
# i 1 more variable: vehicle <dbl>
```

3 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.

3.1 Data checks

3.1.1 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

```
unique(delay_2022$day)
```

```
[1] "Saturday"  "Sunday"    "Monday"    "Tuesday"   "Wednesday" "Thursday"
[7] "Friday"
```

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

```
unique(delay_2022$line)
```

```
 [1] "SRT"              "BD"               "YU"               "YU/BD"
 [5] "SHP"              NA                 "BD/YU"            "YU / BD"
 [9] "YU/ BD"           "B/D"              "Y/BD"             "YU/BD LINES"
[13] "YUS"              "YU & BD"          "YUS AND BD"       "YUS/BD"
[17] "69 WARDEN SOUTH" "YU/BD LINE"       "LINE 2 SHUTTLE"   "57 MIDLAND"
[21] "96 WILSON"        "506 CARLTON"
```

```
delay_2022 |>
  group_by(line) |>
  tally() |>
  arrange(-n)
```

```
# A tibble: 22 x 2
   line             n
   <chr>        <int>
 1 YU           10637
 2 BD            6788
 3 SRT           1196
 4 SHP            852
 5 YU/BD          335
 6 <NA>            39
 7 YU / BD         12
 8 YU & BD          8
 9 BD/YU            7
10 YU/BD LINES      4
# i 12 more rows
```

The `skimr` package might also be useful here

```r
skim(delay_2022)
```

Table 1: Data summary

Name	delay_2022
Number of rows	19895
Number of columns	10
Column type frequency:	
character	6
numeric	3
POSIXct	1
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
time	0	1.00	5	5	0	1406	0
day	0	1.00	6	9	0	7	0
station	0	1.00	5	22	0	296	0
code	0	1.00	3	5	0	179	0
bound	5546	0.72	1	1	0	5	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
line	39	1.00	2	15	0	21	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
min_delay	0	1	3.67	12.00	0	0	0	4	458	
min_gap	0	1	5.33	12.66	0	0	0	8	463	
vehicle	0	1	3571.59	2646.62	0	0	5192	5701	8871	

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	2022-01-01	2022-12-31	2022-06-29	365

3.1.2 Missing values

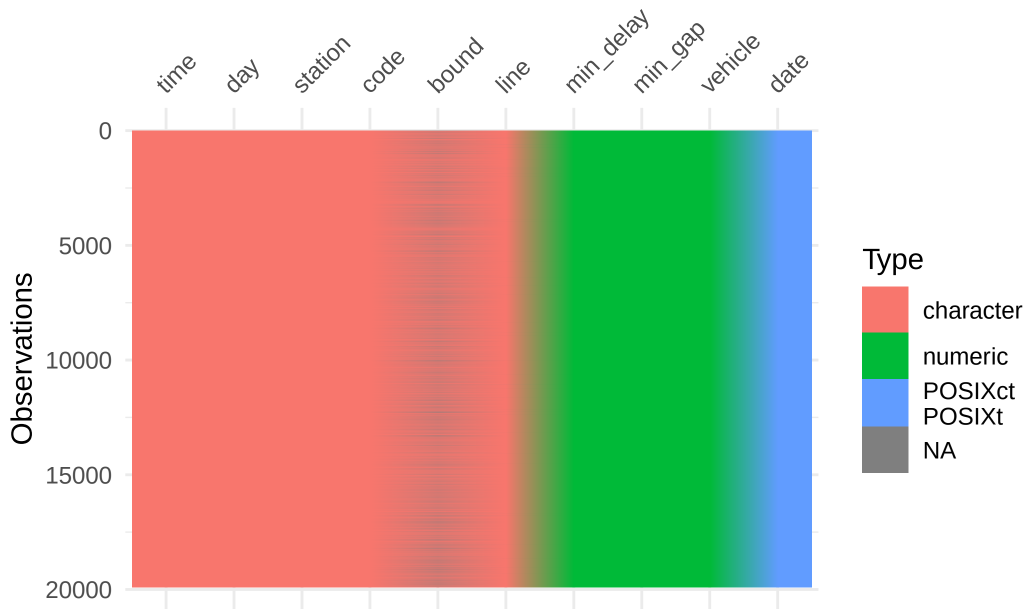
Calculate number of NAs by column

```
delay_2022 |>
  summarize(across(everything(), ~ sum(is.na(.x))))
```

```
# A tibble: 1 x 10
   date  time   day station  code min_delay min_gap bound  line vehicle
  <int> <int> <int>   <int> <int>     <int>   <int> <int> <int>   <int>
1     0     0     0       0     0         0       0  5546    39       0
```

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)

```
vis_dat(delay_2022)
```




```
#vis_miss(delay_2022)
```

3.1.3 Duplicates?

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

```
get_dupes(delay_2022)
```

```
# A tibble: 28 x 11
   date                time  day     station code  min_delay min_gap bound line
   <dttm>              <chr> <chr>   <chr>   <chr>     <dbl>   <dbl> <chr> <chr>
 1 2022-01-12 00:00:00 13:27 Wednes~ FINCH ~ TUNOA         3       6 S     YU
 2 2022-01-12 00:00:00 13:27 Wednes~ FINCH ~ TUNOA         3       6 S     YU
 3 2022-01-12 00:00:00 17:49 Wednes~ FINCH ~ TUNOA         3       6 S     YU
 4 2022-01-12 00:00:00 17:49 Wednes~ FINCH ~ TUNOA         3       6 S     YU
 5 2022-01-17 00:00:00 02:00 Monday  SCARBO~ TRST          0       0 <NA>  SRT
 6 2022-01-17 00:00:00 02:00 Monday  SCARBO~ TRST          0       0 <NA>  SRT
 7 2022-01-20 00:00:00 02:30 Thursd~ YONGE ~ TUST          0       0 <NA>  YU
 8 2022-01-20 00:00:00 02:30 Thursd~ YONGE ~ TUST          0       0 <NA>  YU
 9 2022-01-20 00:00:00 08:51 Thursd~ WILSON~ TUNOA         3       6 S     YU
```

```
10 2022-01-20 00:00:00 08:51 Thursd~ WILSON~ TUNOA         3       6 S     YU
# i 18 more rows
# i 2 more variables: vehicle <dbl>, dupe_count <int>
```

```
delay_2022 <- delay_2022 |> distinct()
```

3.2 Visualizing distributions

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

First, some small cleaning.

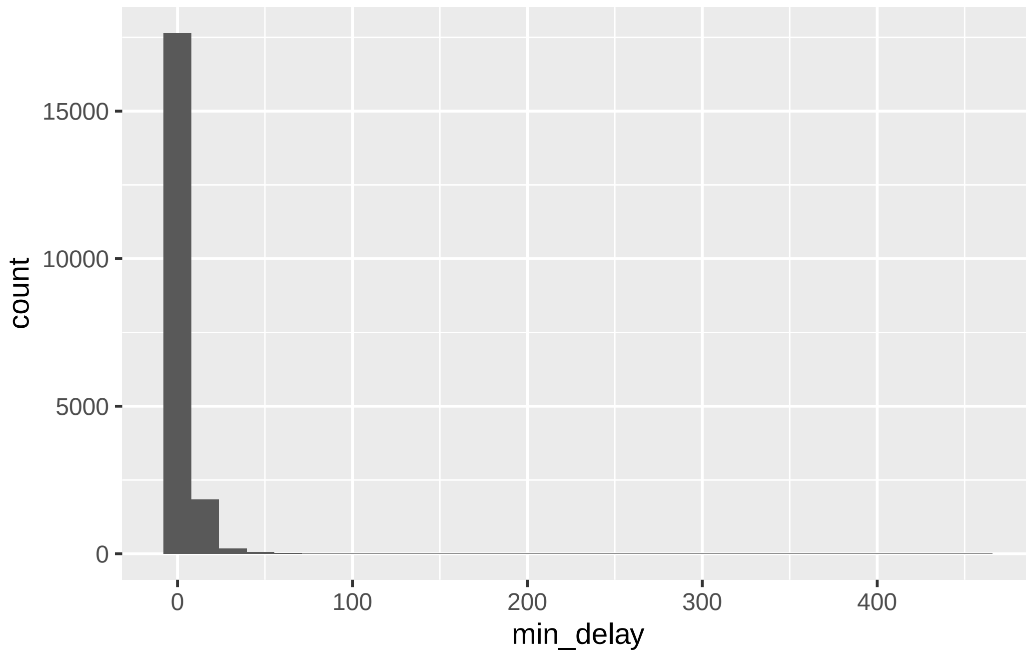
```
delay_2022 |>
  group_by(line) |>
  tally() |>
  arrange(-n)
```

```
# A tibble: 22 x 2
   line            n
   <chr>       <int>
 1 YU          10629
 2 BD           6786
 3 SRT          1194
 4 SHP           851
 5 YU/BD         334
 6 <NA>           39
 7 YU / BD        12
 8 YU & BD         8
 9 BD/YU           7
10 YU/BD LINES     4
# i 12 more rows
```

```
delay_2022 <- delay_2022 |>
  mutate(contains_yu_bd = str_detect(str_to_lower(line), "bd")&str_detect(str_to_lower(lin
  mutate(line = ifelse(contains_yu_bd, ifelse(line=="YU/BD", line, "YU/BD"), line))  |>
  select(-contains_yu_bd)
```

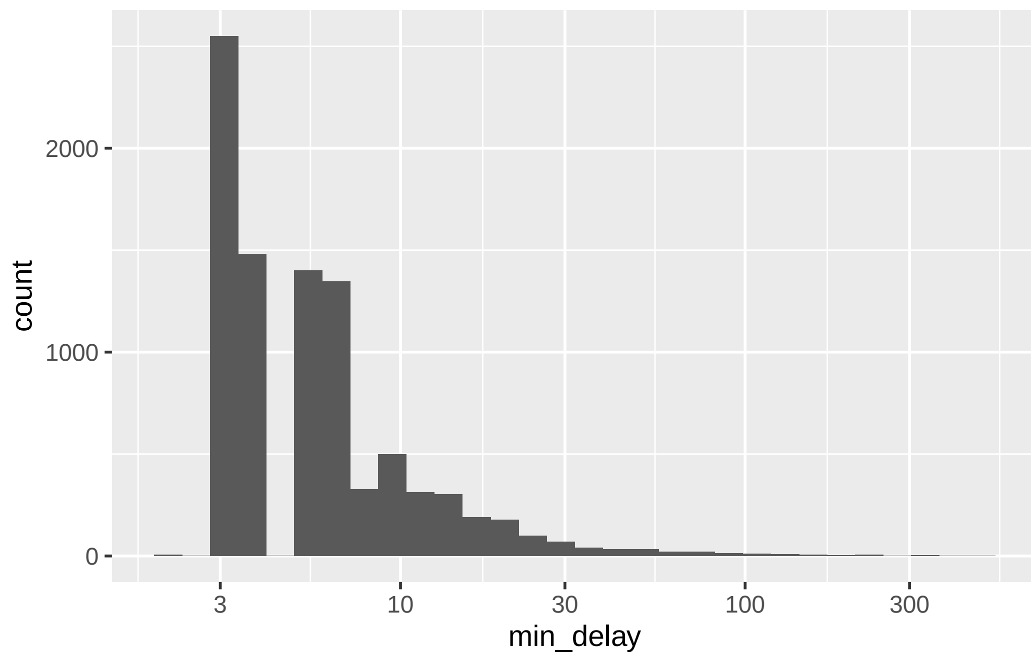
Let’s look at the outcome of interest: `min_delay`. First of all just a histogram of all the data:

```r
## Removing the observations that have non-standardized lines

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

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).

```
delay_2022 <- delay_2022 |>
  left_join(delay_codes |> rename(code = `SUB RMENU CODE`, code_desc = `CODE DESCRIPTION..


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 DESCR
  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.

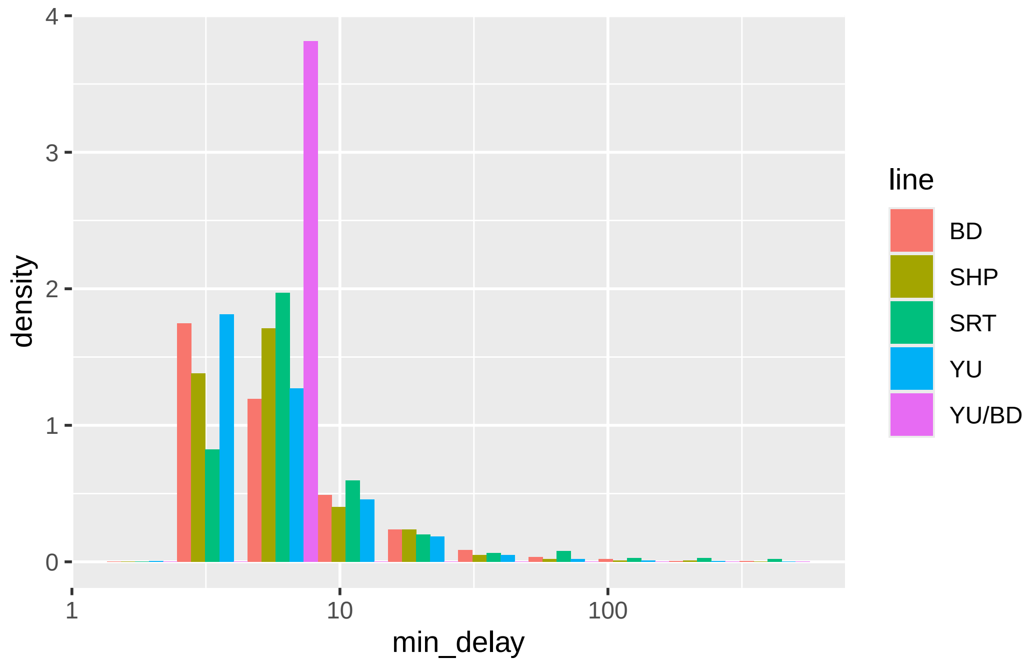
```
delay_2022 |>
  left_join(delay_codes |> rename(code = `SUB RMENU CODE`, code_desc = `CODE DESCRIPTION..
  arrange(-min_delay) |>
  select(date, time, station, line, min_delay, code, code_desc)
```

```
# A tibble: 19,831 x 7
   date                time  station              line  min_delay code  code_desc
   <dttm>              <chr> <chr>                <chr>     <dbl> <chr> <chr>
 1 2022-12-08 00:00:00 17:52 MIDLAND STATION      SRT         458 MRPLB Fire/Smo~
 2 2022-08-22 00:00:00 12:20 SRT LINE             SRT         451 PRSO  Signals ~
 3 2022-04-28 00:00:00 06:02 JANE STATION         BD          388 PUTR  Rail Rel~
 4 2022-07-26 00:00:00 07:06 YONGE BD STATION     BD          382 MUPLB Fire/Smo~
 5 2022-08-15 00:00:00 12:57 DUFFERIN STATION     BD          327 MUPR1 Priority~
 6 2022-01-26 00:00:00 20:15 KENNEDY SRT STATION SRT          315 MRWEA Weather ~
 7 2022-08-02 00:00:00 21:23 HIGHWAY 407 STATION YU           312 MUPR1 Priority~
 8 2022-01-17 00:00:00 21:30 SHEPPARD WEST TO U~ YU           291 MUFM  Force Ma~
 9 2022-01-25 00:00:00 21:03 SCARBOROUGH CTR ST~ SRT          285 PRSL  Loop Rel~
10 2022-06-17 00:00:00 12:25 KIPLING STATION      BD          241 SUUT  Unauthor~
# i 19,821 more rows
```

3.2.0.1 Grouping and small multiples

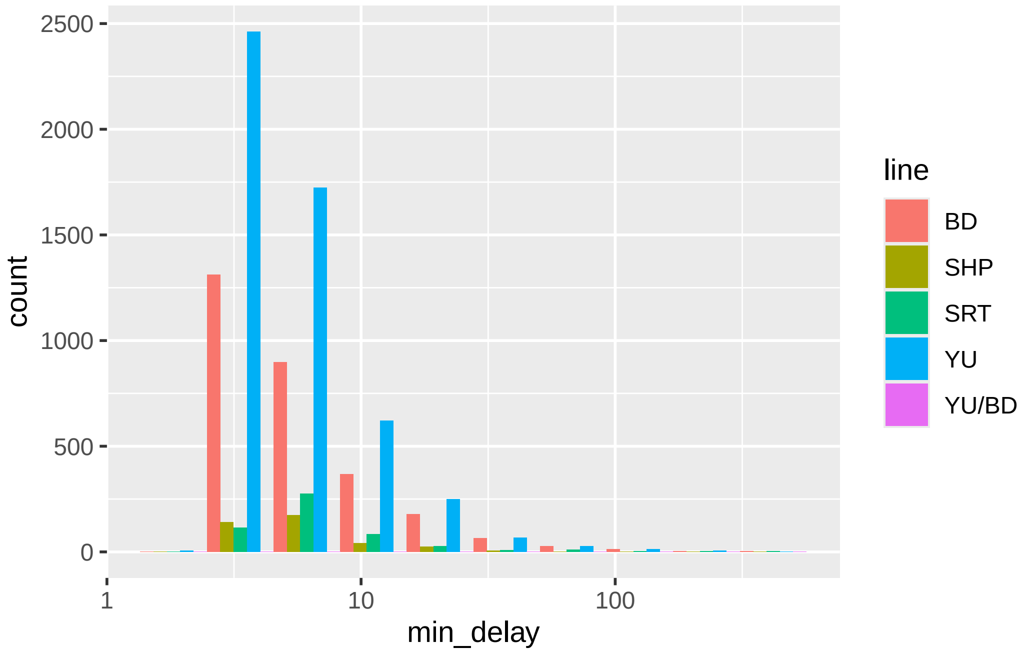
A quick and powerful visualization technique is to group the data by a variable of interest, e.g. `line`

```r
ggplot(data = delay_2022) +
  geom_histogram(aes(x = min_delay, y = ..density.., fill = line), position = 'dodge', bin
  scale_x_log10()
```



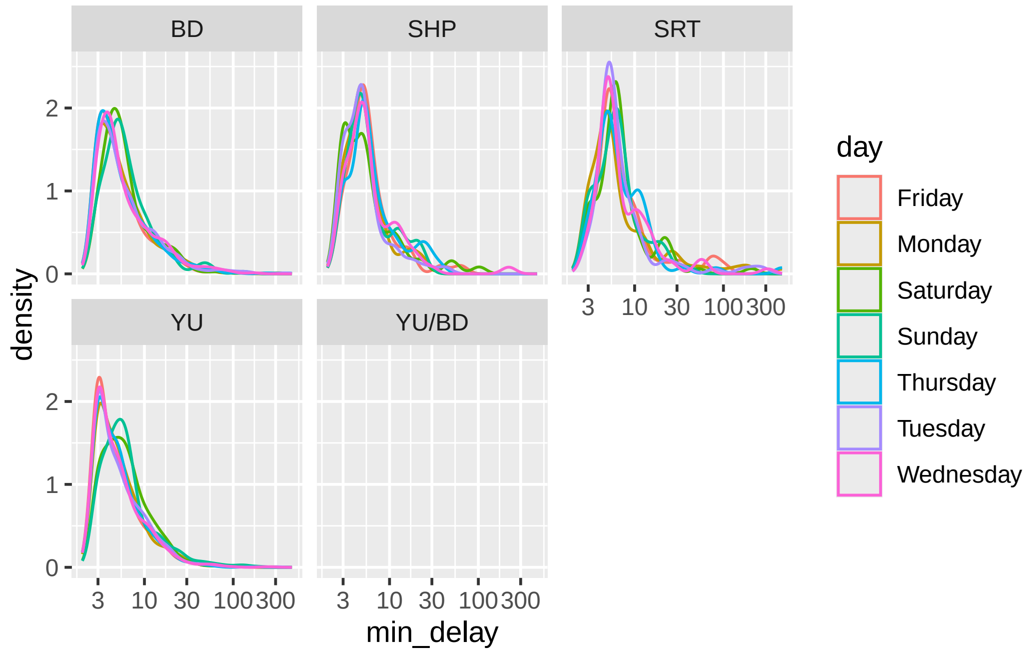
I switched to density above to look at the the distributions more comparably, but we should also be aware of differences in frequency, in particular, SHP and SRT have much smaller counts:

```
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:

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



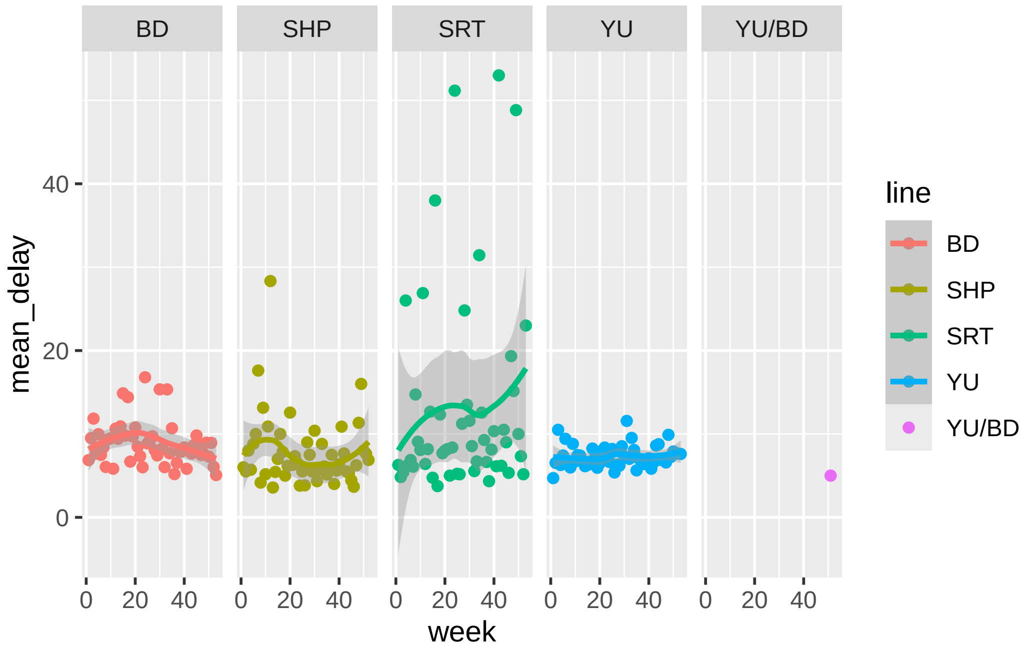
Side note: the station names are a mess. Try and clean up the station names a bit by taking just the first word (or, the first two if it starts with "ST"):

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

3.3 Visualizing time series

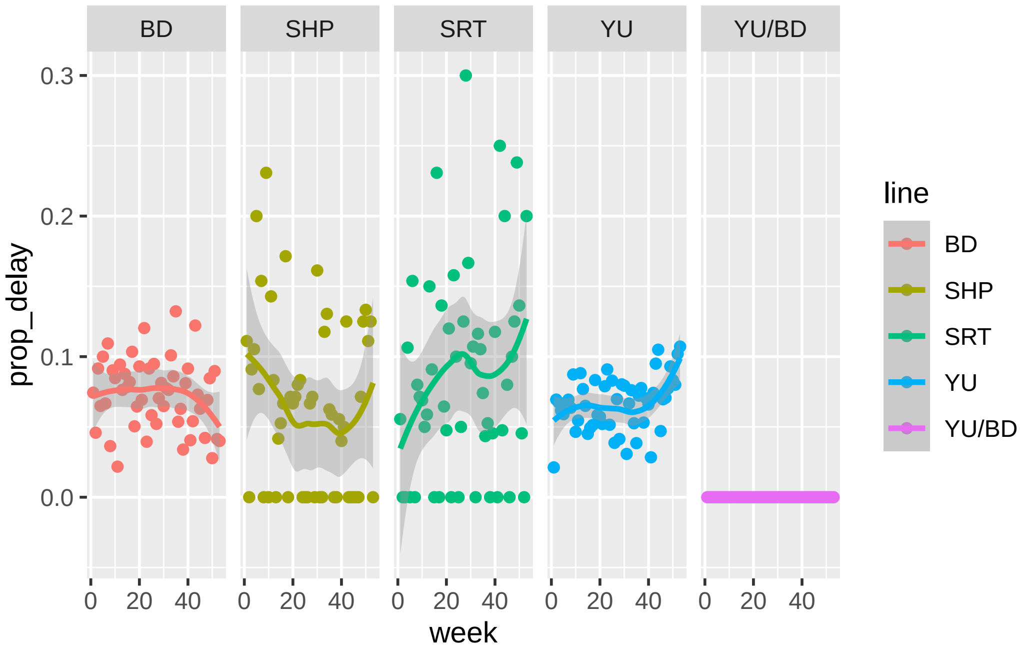
Daily plot is messy (you can check for yourself). Let's look by week to see if there's any seasonality. The `lubridate` package has lots of helpful functions that deal with date variables. First, mean delay (of those that were delayed more than 0 mins):

```
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?

```
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)
```

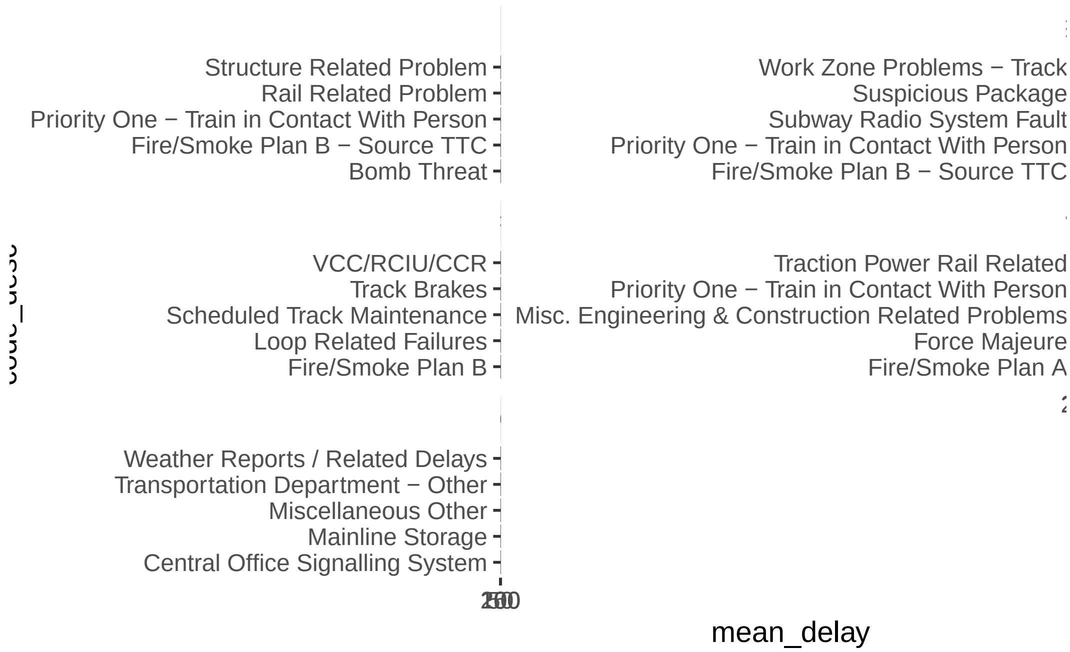


3.4 Visualizing relationships

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.

```
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()
```



3.5 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:

```
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))
  )
```

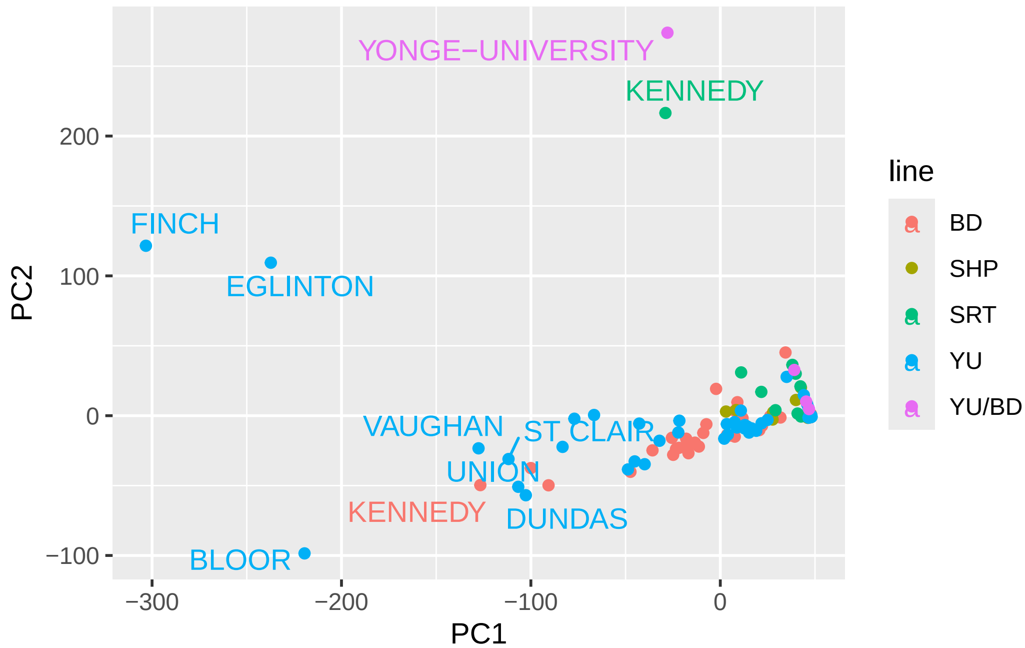
Do the PCA:

```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)
```

```
# A tibble: 6 x 41
# Groups:   line, station_clean [6]
  line  station_clean     PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
  <chr> <chr>           <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 BD    BATHURST        -17.1  -23.2  -12.2  12.9   -1.47  -6.57   1.99   6.28
2 BD    BAY               7.66 -15.0   -4.70  9.77   1.35   1.29  -5.88  -0.726
3 BD    BLOOR            34.4   45.2   -7.33  5.62   0.553 -6.01   1.11  -0.900
4 BD    BLOOR-DANFORTH  47.8    1.09   5.53  0.194 -9.27  -4.13  -0.141 -0.587
5 BD    BROADVIEW       -23.4  -23.8  -13.9  14.3    4.57   4.13  -3.70  -5.64
6 BD    CASTLE           15.1  -10.3   -3.30  7.13  -3.40  -0.301 -0.561  3.19
# i 31 more variables: PC9 <dbl>, PC10 <dbl>, PC11 <dbl>, PC12 <dbl>,
#   PC13 <dbl>, PC14 <dbl>, PC15 <dbl>, PC16 <dbl>, PC17 <dbl>, PC18 <dbl>,
#   PC19 <dbl>, PC20 <dbl>, PC21 <dbl>, PC22 <dbl>, PC23 <dbl>, PC24 <dbl>,
#   PC25 <dbl>, PC26 <dbl>, PC27 <dbl>, PC28 <dbl>, PC29 <dbl>, PC30 <dbl>,
#   PC31 <dbl>, PC32 <dbl>, PC33 <dbl>, PC34 <dbl>, PC35 <dbl>, PC36 <dbl>,
#   PC37 <dbl>, PC38 <dbl>, PC39 <dbl>
```

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
```



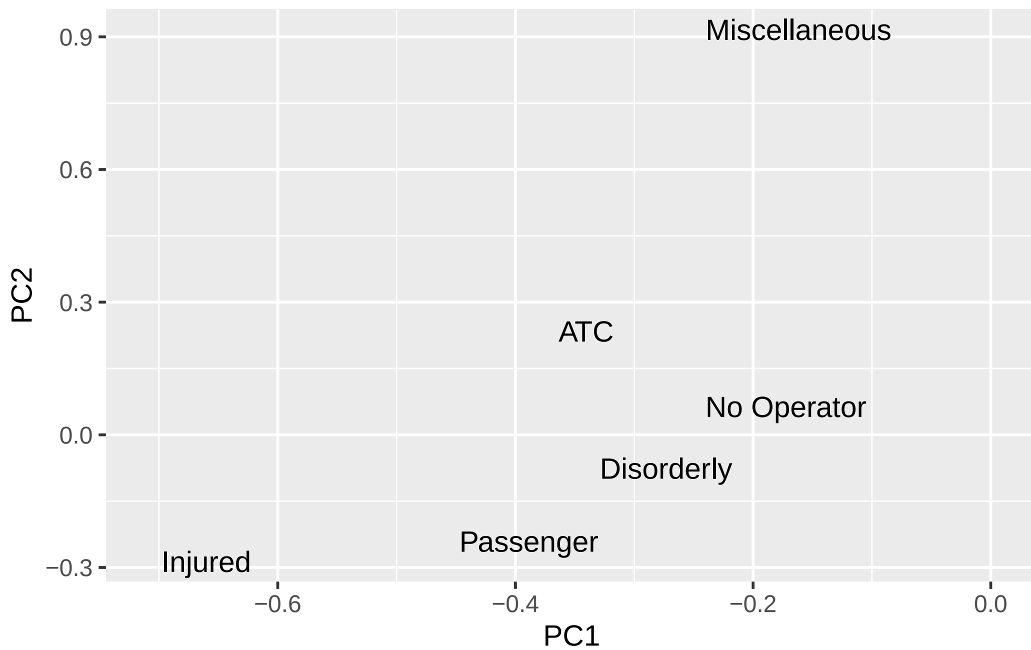
Plot the factor loadings. Some evidence of public v operator?

```r
df_out_r <- as_tibble(delay_pca$rotation)
df_out_r$feature <- colnames(dwide[,3:ncol(dwide)])

df_out_r
```

```
# A tibble: 39 x 40
       PC1      PC2      PC3      PC4      PC5      PC6      PC7      PC8      PC9
     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
 1 -0.127  -0.0386  -1.47e-2  0.0323   0.0283   0.0799 -0.109    0.0527 -0.125
 2 -0.304  -0.126   -4.13e-2  0.0740   0.105    0.221  -0.129    0.517   0.614
 3 -0.0535  0.00111 -3.11e-2  0.0238   0.0136   0.0726  0.0480   0.217  -0.322
 4 -0.0135 -0.0140  -5.85e-3  0.00420  0.0461   0.0407 -0.00357 -0.0283  0.0455
 5 -0.0120 -0.00461 -2.98e-3  0.00724 -0.0177   0.0363  0.0429   0.104  -0.0286
 6 -0.0903 -0.00945 -4.32e-2 -0.0330  -0.0533   0.0944 -0.0580  -0.133  -0.307
 7 -0.0161 -0.00317 -2.44e-4  0.00537  0.00835  0.0420 -0.00534 -0.0360  0.00483
 8 -0.709  -0.273   -2.43e-1  0.108   -0.117   -0.395   0.350   -0.204  -0.0316
 9 -0.251   0.903   -2.53e-1  0.106    0.202   -0.0313  0.0224  -0.0149  0.0384
```

```
10 -0.0404  0.0265  -4.81e-2 -0.0740  -0.126    0.480   0.142   -0.331   0.0595
# i 29 more rows
# i 31 more variables: PC10 <dbl>, PC11 <dbl>, PC12 <dbl>, PC13 <dbl>,
#   PC14 <dbl>, PC15 <dbl>, PC16 <dbl>, PC17 <dbl>, PC18 <dbl>, PC19 <dbl>,
#   PC20 <dbl>, PC21 <dbl>, PC22 <dbl>, PC23 <dbl>, PC24 <dbl>, PC25 <dbl>,
#   PC26 <dbl>, PC27 <dbl>, PC28 <dbl>, PC29 <dbl>, PC30 <dbl>, PC31 <dbl>,
#   PC32 <dbl>, PC33 <dbl>, PC34 <dbl>, PC35 <dbl>, PC36 <dbl>, PC37 <dbl>,
#   PC38 <dbl>, PC39 <dbl>, feature <chr>
```

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



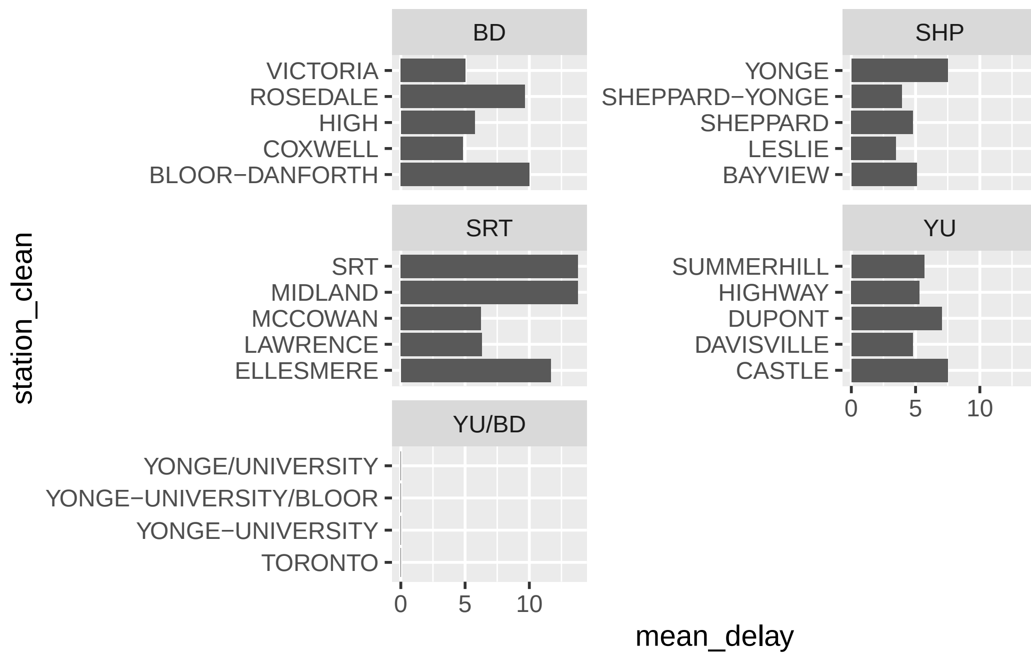
4 Lab Exercises

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`

```
delay_2022 %>%
  group_by(line, station_clean) %>%
```

```
  summarise(mean_delay = mean(min_delay), n_obs = n()) %>%
  filter(n_obs>1) |>
  arrange(-mean_delay) %>%
  slice(1:5) %>%
  ggplot(aes(x = station_clean,
             y = mean_delay)) +
  geom_col() +
  facet_wrap(vars(line),
             scales = "free_y",
             nrow = 4) +
  coord_flip()
```



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.

```
delay_positive <- delay_2022 %>%
  filter(min_delay > 0)
```

```r
grouped_delays <- delay_positive %>%
  group_by(code_desc)

augmented_delays <- grouped_delays %>%
  mutate(n_delays = n())

n_delays_summary <- summary(augmented_delays$n_delays)
print(n_delays_summary)
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.0   108.0   296.0   382.3   703.0   963.0
```

```r
filtered_delays <- augmented_delays %>%
  filter(n_delays > 295)

model <- lm(log(min_delay) ~ line + code_desc, data = filtered_delays)

model_summary <- summary(model)
print(model_summary)
```

```
Call:
lm(formula = log(min_delay) ~ line + code_desc, data = filtered_delays)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4301 -0.3447 -0.0835  0.2602  3.3688

Coefficients:
                                                                  Estimate
(Intercept)                                                       1.568235
lineSHP                                                           0.157335
lineSRT                                                           0.260871
lineYU                                                           -0.004379
lineYU/BD                                                         0.039357
code_descDisorderly Patron                                        0.124720
code_descInjured or ill Customer (On Train) - Medical Aid Refused  0.227137
code_descNo Operator Immediately Available                       -0.214618
code_descOPTO (COMMS) Train Door Monitoring                      -0.120574
code_descPassenger Assistance Alarm Activated - No Trouble Found -0.255706
code_descPassenger Other                                          0.557621
```

```
code_descTransportation Department - Other                          0.001846
code_descUnauthorized at Track Level                                0.699601
                                                                  Std. Error
(Intercept)                                                          0.027765
lineSHP                                                              0.044087
lineSRT                                                              0.054902
lineYU                                                               0.019657
lineYU/BD                                                            0.520996
code_descDisorderly Patron                                           0.027046
code_descInjured or ill Customer (On Train) - Medical Aid Refused   0.035957
code_descNo Operator Immediately Available                           0.031945
code_descOPTO (COMMS) Train Door Monitoring                          0.027709
code_descPassenger Assistance Alarm Activated - No Trouble Found    0.030334
code_descPassenger Other                                             0.035079
code_descTransportation Department - Other                           0.037103
code_descUnauthorized at Track Level                                 0.032830
                                                                  t value
(Intercept)                                                         56.482
lineSHP                                                              3.569
lineSRT                                                              4.752
lineYU                                                              -0.223
lineYU/BD                                                            0.076
code_descDisorderly Patron                                           4.611
code_descInjured or ill Customer (On Train) - Medical Aid Refused   6.317
code_descNo Operator Immediately Available                          -6.718
code_descOPTO (COMMS) Train Door Monitoring                         -4.351
code_descPassenger Assistance Alarm Activated - No Trouble Found   -8.430
code_descPassenger Other                                            15.896
code_descTransportation Department - Other                           0.050
code_descUnauthorized at Track Level                                21.310
                                                                  Pr(>|t|)
(Intercept)                                                        < 2e-16 ***
lineSHP                                                           0.000362 ***
lineSRT                                                           2.08e-06 ***
lineYU                                                            0.823738
lineYU/BD                                                         0.939786
code_descDisorderly Patron                                        4.10e-06 ***
code_descInjured or ill Customer (On Train) - Medical Aid Refused 2.91e-10 ***
code_descNo Operator Immediately Available                        2.05e-11 ***
code_descOPTO (COMMS) Train Door Monitoring                       1.38e-05 ***
code_descPassenger Assistance Alarm Activated - No Trouble Found   < 2e-16 ***
code_descPassenger Other                                           < 2e-16 ***
code_descTransportation Department - Other                        0.960320
```

```
code_descUnauthorized at Track Level                              < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5199 on 4764 degrees of freedom
Multiple R-squared:  0.2401,	Adjusted R-squared:  0.2381 
F-statistic: 125.4 on 12 and 4764 DF,  p-value: < 2.2e-16
```

Well from the residual distributions, we see that there might be a small skew given the range of -1.4699 to 3.2161, but the median is approximately 0, indicating it's actually a good fit. The overall R^2 is about 25%, which means that this given features is able to explain around 25% of the dependent variable, which is not bad, but can certainly be improved if exploring better features for regression. The F-statistic and p-Value definitely confirmed that the reasons selected explains a significant part of delays in the TTC.

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`)

```r
list_package_resources("e869d365-2c15-4893-ad2a-744ca867be3b")
```

```
# A tibble: 4 x 4
  name                                 id                     format last_modified
  <chr>                                <chr>                  <chr>  <date>       
1 Campaign Contributions 2018 Data     5f54ab3d-44d7-4e5c-9c~ ZIP    2023-04-26   
2 Campaign Contributions 2018 Readme   eea9eecd-75ba-4a27-9f~ XLSX   2023-04-26   
3 Campaign Contributions 2014 Data     8b42906f-c894-4e93-a9~ ZIP    2023-04-26   
4 Campaign Contributions 2014 Readme   10158522-4f3b-4957-9f~ XLS    2023-04-26   
```

```r
all_campaigns <- get_resource("8b42906f-c894-4e93-a98e-acac200f34a4")
df <- all_campaigns[[2]]

colnames(df) <- as.character(df[1, ])
df <- df[-1, ]
```

```r
df <- setNames(df, tolower(gsub("[^[:alnum:] ]", "_", colnames(df))))
df <- setNames(df, gsub(" ", "_", colnames(df)))
```

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.

```r
skim(df)
```

Table 5: Data summary

Name	df
Number of rows	10199
Number of columns	13
Column type frequency:	
character	13
Group variables	None

Variable type: character

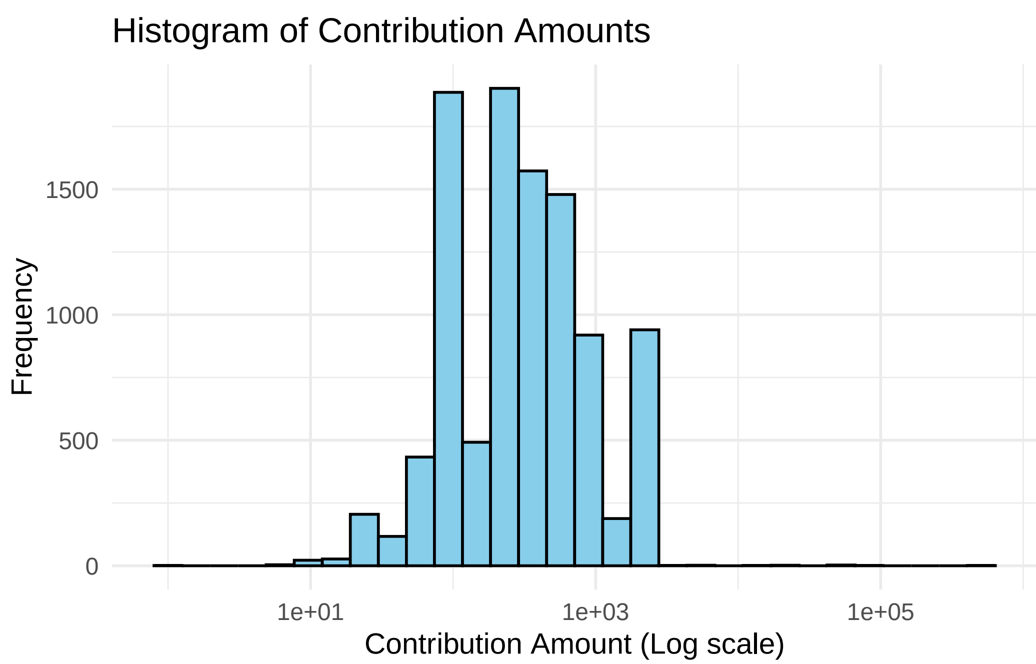
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
contributor_s_name	0	1	4	31	0	7545	0
contributor_s_address	10197	0	24	26	0	2	0
contributor_s_postal_code	0	1	7	7	0	5284	0
contribution_amount	0	1	1	18	0	209	0
contribution_type_desc	0	1	8	14	0	2	0
goods_or_service_desc	10188	0	11	40	0	9	0
contributor_type_desc	0	1	10	11	0	2	0
relationship_to_candidate	10166	0	6	9	0	2	0
president___business_manager	10197	0	13	16	0	2	0
authorized_representative	10197	0	13	16	0	2	0
candidate	0	1	9	18	0	27	0
office	0	1	5	5	0	1	0
ward	10199	0	NA	NA	0	0	0

```r
df <- df |>
  mutate(contribution_amount = as.numeric(contribution_amount))
```

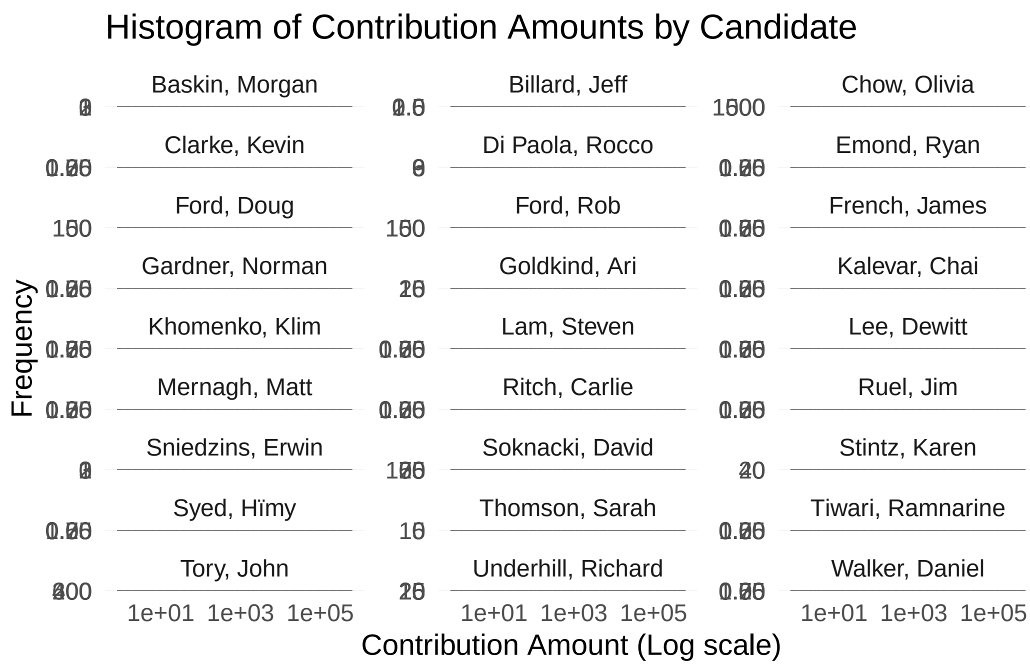
Well yes, there are missing values in the dataset. For example, "contributors_address" have only 2 unique values, while missing 10197 values in total, this suggest that there is a huge portion of the data is not avaliable. So is the case with "good_or_service_desc", "relationship_to_cadidate", "president_business_manager" and "authorized_representive", and "ward". Apperantly the variable type of contribution_amount is not correct so we need to change them to numeric.

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.

```
df %>%
  ggplot(aes(x = contribution_amount)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  scale_x_log10() +
  labs(title = "Histogram of Contribution Amounts",
       x = "Contribution Amount (Log scale)",
       y = "Frequency") +
  theme_minimal()
```



```r
df %>%
  ggplot(aes(x = contribution_amount)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  scale_x_log10() +
  facet_wrap(~candidate, scales = "free_y", ncol = 3) +
  labs(title = "Histogram of Contribution Amounts by Candidate",
       x = "Contribution Amount (Log scale)",
       y = "Frequency") +
  theme_minimal()
```



Outliers: The candidates with notably higher contribution amounts are Rob Ford, Doug Ford, and Olivia Chow. They have contributions that span a wider range on the log scale, indicating larger individual contributions. Distribution: The histogram shows a clear right skewness in the contribution amounts, suggesting that there are a lot of smaller contributions and relatively few large ones.

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

  - total contributions
  - mean contribution
  - number of contributions

```r
total_contributions <- df %>%
  group_by(candidate) %>%
  summarise(total_contr = sum(contribution_amount, na.rm = TRUE)) %>%
  arrange(-total_contr) %>%
  ungroup()

print(total_contributions)
```

```
# A tibble: 27 x 2
   candidate          total_contr
   <chr>                    <dbl>
 1 Tory, John            2767869.
 2 Chow, Olivia          1638266.
 3 Ford, Doug             889897.
 4 Ford, Rob              387648.
 5 Stintz, Karen          242805
 6 Soknacki, David        132431
 7 Goldkind, Ari             41125.
 8 Thomson, Sarah            34628.
 9 Di Paola, Rocco           21126
10 Underhill, Richard        15660
# i 17 more rows
```

```r
mean_contributions <- df %>%
  group_by(candidate) %>%
  summarise(mean_contr = mean(contribution_amount, na.rm = TRUE)) %>%
  arrange(-mean_contr) %>%
  ungroup()

print(mean_contributions)
```

```
# A tibble: 27 x 2
   candidate        mean_contr
   <chr>                 <dbl>
 1 Sniedzins, Erwin      2025
 2 Syed, Himy            2018
 3 Ritch, Carlie         1887.
 4 Ford, Doug            1456.
 5 Clarke, Kevin         1200
 6 Di Paola, Rocco       1174.
```

```
 7 Tory, John            1064.
 8 Gardner, Norman       1000
 9 Stintz, Karen          995.
10 Kalevar, Chai          900
# i 17 more rows
```

```
number_of_contributions <- df %>%
  group_by(candidate) %>%
  tally(name = "num_contributions") %>%
  arrange(-num_contributions) %>%
  ungroup()

print(number_of_contributions)
```

```
# A tibble: 27 x 2
   candidate          num_contributions
   <chr>                          <int>
 1 Chow, Olivia                    5708
 2 Tory, John                      2602
 3 Ford, Doug                       611
 4 Ford, Rob                        538
 5 Soknacki, David                  314
 6 Stintz, Karen                    244
 7 Goldkind, Ari                     47
 8 Underhill, Richard                41
 9 Thomson, Sarah                    40
10 Di Paola, Rocco                   18
# i 17 more rows
```

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

```
colnames(df)
```

```
 [1] "contributor_s_name"          "contributor_s_address"
 [3] "contributor_s_postal_code"   "contribution_amount"
 [5] "contribution_type_desc"      "goods_or_service_desc"
 [7] "contributor_type_desc"       "relationship_to_candidate"
 [9] "president__business_manager" "authorized_representative"
[11] "candidate"                   "office"
[13] "ward"
```

```r
df_not_include_self <- df |>
  filter(contributor_s_name != candidate)

total_external_contributions <- df_not_include_self %>%
  group_by(candidate) %>%
  summarise(total_contr = sum(contribution_amount, na.rm = TRUE)) %>%
  arrange(-total_contr) %>%
  ungroup()

print(total_external_contributions)
```

```
# A tibble: 17 x 2
   candidate          total_contr
   <chr>                    <dbl>
 1 Tory, John            2765369.
 2 Chow, Olivia          1634766.
 3 Ford, Doug             331173.
 4 Stintz, Karen          242805
 5 Ford, Rob              174510.
 6 Soknacki, David        132431
 7 Thomson, Sarah          27702.
 8 Goldkind, Ari           17501
 9 Underhill, Richard      15660
10 Di Paola, Rocco         15126
11 Ritch, Carlie            5660
12 Sniedzins, Erwin         5600
13 Gardner, Norman          3000
14 Baskin, Morgan           1550
15 Billard, Jeff            1486.
16 Tiwari, Ramnarine        1000
17 Lam, Steven               300
```

```r
mean_external_contributions <- df_not_include_self %>%
  group_by(candidate) %>%
  summarise(mean_contr = mean(contribution_amount, na.rm = TRUE)) %>%
  arrange(-mean_contr) %>%
  ungroup()

print(mean_external_contributions)
```

```
# A tibble: 17 x 2
```

```
   candidate          mean_contr
   <chr>                   <dbl>
 1 Ritch, Carlie           1887.
 2 Sniedzins, Erwin        1867.
 3 Tory, John              1063.
 4 Gardner, Norman         1000
 5 Tiwari, Ramnarine       1000
 6 Stintz, Karen            995.
 7 Di Paola, Rocco          890.
 8 Thomson, Sarah           729.
 9 Ford, Doug               545.
10 Billard, Jeff            496.
11 Soknacki, David          422.
12 Underhill, Richard       382.
13 Goldkind, Ari            380.
14 Ford, Rob                329.
15 Lam, Steven              300
16 Chow, Olivia             286.
17 Baskin, Morgan           194.
```

```r
number_of_external_contributions <- df_not_include_self %>%
  group_by(candidate) %>%
  tally(name = "num_external_contributions") %>%
  arrange(-num_external_contributions) %>%
  ungroup()

print(number_of_external_contributions)
```

```
# A tibble: 17 x 2
   candidate          num_external_contributions
   <chr>                                   <int>
 1 Chow, Olivia                             5706
 2 Tory, John                               2601
 3 Ford, Doug                                608
 4 Ford, Rob                                 531
 5 Soknacki, David                           314
 6 Stintz, Karen                             244
 7 Goldkind, Ari                              46
 8 Underhill, Richard                         41
 9 Thomson, Sarah                             38
10 Di Paola, Rocco                            17
```

```
11 Baskin, Morgan                              8
12 Billard, Jeff                               3
13 Gardner, Norman                             3
14 Ritch, Carlie                               3
15 Sniedzins, Erwin                            3
16 Lam, Steven                                 1
17 Tiwari, Ramnarine                           1
```

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

```r
grouped_by_contributor <- df %>%
  group_by(contributor_s_name)

unique_candidates_per_contributor <- grouped_by_contributor %>%
  distinct(candidate)

count_candidates_per_contributor <- unique_candidates_per_contributor %>%
  tally()

multi_candidate_contributors <- count_candidates_per_contributor %>%
  filter(n > 1)

num_multi_candidate_contributors <- nrow(multi_candidate_contributors)

print(num_multi_candidate_contributors)
```

```
[1] 184
```