Applied_Stat_2_Lab_2

Quarto

```r
library(opendatatoronto)
```

```
Warning: package 'opendatatoronto' was built under R version 4.3.2
```

```r
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.3.2
```

```
Warning: package 'readr' was built under R version 4.3.2
```

```
Warning: package 'forcats' was built under R version 4.3.2
```

```r
library(stringr)
library(skimr) # EDA
library(visdat) # EDA
```

```
Warning: package 'visdat' was built under R version 4.3.2
```

```r
library(janitor)
```

```
Warning: package 'janitor' was built under R version 4.3.2
```

```r
library(lubridate)
library(ggrepel)
```

```
Warning: package 'ggrepel' was built under R version 4.3.2
```

```r
all_data <- list_packages(limit = 500)
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 Committee of Adj~ 260e~ City ~ <NA>         City Pla~ This d~ Table
2 Residential Fron~ 4a65~ Locat~ Mobility,Cl~ Transpor~ Legall~ Table
3 Dinesafe          ea1d~ Publi~ <NA>         Toronto ~ Snapsh~ Table
4 Property Boundar~ 1aca~ Locat~ <NA>         Informat~ This d~ Document
5 Lobbyist Registry 6a87~ City ~ <NA>         Lobbyist~ The Lo~ Document
6 Municipal Licens~ 5da2~ City ~ Affordable ~ Municipa~ This d~ Document
# i 4 more variables: num_resources <int>, formats <chr>, refresh_rate <chr>,
#   last_refreshed <date>
```

```r
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)
```

```r
# note: I obtained these codes from the 'id' column in the `res` object above
delay_codes <- get_resource("3900e649-f31e-4b79-9f20-4731bbfd94f7")
```

```
New names:
* `` -> `...1`
* `CODE DESCRIPTION` -> `CODE DESCRIPTION...3`
* `` -> `...4`
* `` -> `...5`
* `CODE DESCRIPTION` -> `CODE DESCRIPTION...7`
```

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

```r
head(delay_2022)
```

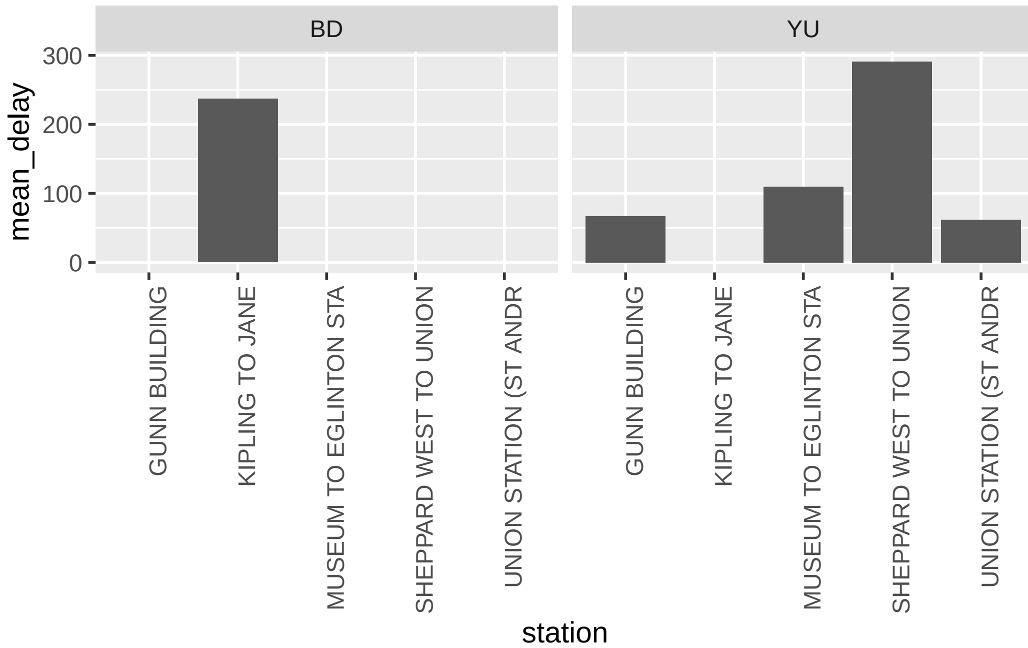
```
# A tibble: 6 x 10
  date                time  day      station code  min_delay min_gap bound line
  <dttm>              <chr> <chr>    <chr>   <chr>     <dbl>   <dbl> <chr> <chr>
1 2022-01-01 00:00:00 15:59 Saturday LAWREN~ SRDP          0       0 N     SRT
2 2022-01-01 00:00:00 02:23 Saturday SPADIN~ 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>
```

Answer 1)

We can do this task through the following code

```r
delay_2022 |>
  group_by(station,line) |>
  summarise(mean_delay=mean(min_delay)) |>
  arrange(-mean_delay)|>
  head(5) |>
  ggplot(aes(x = station,
             y = mean_delay))+
  geom_col()+
  facet_wrap(.~line)+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
`summarise()` has grouped output by 'station'. You can override using the
`.groups` argument.
```



Answer 2)

We will do this by first filtering the codes responsible for 50% of the delays, and then extract them, filter the table on the basis of the codes obtained and finally model the delay.

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

```
Joining with `by = join_by(code)`
```

```
delay_2022_top_0.5 <- delay_2022 |>
                      filter(min_delay>0)|>
                      group_by(code)|>
                      summarise(no_rows = length(code))|>
                      arrange(-no_rows)|>
                      mutate(cumulative_sum = cumsum(no_rows))|>
                      mutate(half_sum = sum(no_rows)/2)|>
                      filter(cumulative_sum<=half_sum)
delay_2022_top_0.5
```

```
# A tibble: 8 x 4
  code  no_rows cumulative_sum half_sum
  <chr>   <int>          <int>    <dbl>
1 SUDP      943            943     4488
2 PUOPO     730           1673     4488
3 MUATC     703           2376     4488
4 MUPAA     523           2899     4488
5 SUUT      427           3326     4488
6 TUNOA     422           3748     4488
7 SUO       340           4088     4488
8 MUIR      319           4407     4488
```

```
frequent_delay_codes <- delay_2022_top_0.5$code
frequent_delay_codes
```

```
[1] "SUDP"  "PUOPO" "MUATC" "MUPAA" "SUUT"  "TUNOA" "SUO"   "MUIR"
```

```
lm_table_delay_code <- delay_2022|>
                      filter(min_delay>0 & (code %in% frequent_delay_codes))

delay_model <- lm(min_delay ~ line + code, data = lm_table_delay_code)
summary(delay_model)
```

```
Call:
lm(formula = min_delay ~ line + code, data = lm_table_delay_code)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.475  -2.450  -1.072   0.890 227.525 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   5.7698     0.3485  16.554  < 2e-16 ***
lineSHP       1.3899     0.5828   2.385 0.017132 *  
lineYU       -0.3203     0.2521  -1.270 0.204022    
codeMUIR      1.5470     0.4432   3.491 0.000486 ***
codeMUPAA    -1.6602     0.3741  -4.438  9.3e-06 ***
codePUOPO    -0.9396     0.3405  -2.759 0.005814 ** 
codeSUDP      0.9928     0.3344   2.969 0.003003 ** 
```

```
codeSUO       5.1117     0.4381  11.667  < 2e-16 ***
codeSUUT      7.7057     0.4069  18.938  < 2e-16 ***
codeTUNOA    -1.3775     0.3954  -3.484 0.000499 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.38 on 4396 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.1668,    Adjusted R-squared:  0.1651
F-statistic:  97.8 on 9 and 4396 DF,  p-value: < 2.2e-16
```

```
lm_table_delay_code
```

```
# A tibble: 4,407 x 11
   date                time  day     station code  min_delay min_gap bound line
   <dttm>              <chr> <chr>   <chr>   <chr>     <dbl>   <dbl> <chr> <chr>
 1 2022-01-01 00:00:00 08:12 Saturd~ FINCH ~ TUNOA         6      12 S     YU
 2 2022-01-01 00:00:00 09:51 Saturd~ FINCH ~ TUNOA         6      12 S     YU
 3 2022-01-01 00:00:00 12:01 Saturd~ DAVISV~ SUDP          3       8 S     YU
 4 2022-01-01 00:00:00 12:14 Saturd~ RUNNYM~ SUUT         20      25 W     BD
 5 2022-01-01 00:00:00 18:20 Saturd~ EGLINT~ MUATC         3      10 S     YU
 6 2022-01-01 00:00:00 18:59 Saturd~ EGLINT~ MUATC         3      10 S     YU
 7 2022-01-01 00:00:00 19:13 Saturd~ HIGHWA~ PUOPO         5      12 S     YU
 8 2022-01-01 00:00:00 23:37 Saturd~ KENNED~ SUDP          7      14 W     BD
 9 2022-01-02 00:00:00 08:14 Sunday  SHEPPA~ PUOPO         6      12 N     YU
10 2022-01-02 00:00:00 08:59 Sunday  EGLINT~ TUNOA         6      12 N     YU
# i 4,397 more rows
# i 2 more variables: vehicle <dbl>, code_desc <chr>
```

It would first seem that our results for the most frequent causes of delays dont seem to agree with our EDA results, where in Delays were caused due to major incidents like accidents, maintenance, power/track failures, fires, bomb threats and so on. However, that is because the parameter for interest in that scenario was mean_delay, which gets skewed by these delay reasons as they cause the highest amount of delay, hence skewing the mean. In terms of the factors which most frequently cause delays, our obtained delay codes make sense, and in a way also agree with the data. Though, it has to be said that in this case, a linear model with an interaction term was not able to get a good fit over the data, with an R^2 of only 0.17.

Answer 3)

We proceed to get the data and perform pre-processing in the following code block

```r
library(opendatatoronto)
library(janitor)
all_data <- search_packages("campaign")
campaign_id <- all_data$id
resource <- list_package_resources(campaign_id[1])
resource
```

```
# 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
campaign_data <- get_resource('8b42906f-c894-4e93-a98e-acac200f34a4')
```

```
New names:
New names:
New names:
New names:
New names:
New names:
New names:
* `` -> `...2`
* `` -> `...3`
```

```r
campaign_data_mayoral <- campaign_data[[2]]
colnames(campaign_data_mayoral) <- as.character(campaign_data_mayoral[1,])
campaign_data_mayoral <- campaign_data_mayoral[-1,]
rownames(campaign_data_mayoral) <- NULL
campaign_data_mayoral <- clean_names(campaign_data_mayoral)
campaign_data_mayoral
```

```
# A tibble: 10,199 x 13
   contributors_name  contributors_address contributors_postal_code
   <chr>              <chr>                <chr>
 1 A D'Angelo, Tullio <NA>                 M6A 1P5
 2 A Strazar, Martin  <NA>                 M2M 3B8
```

```
 3 A'Court, K Susan   <NA>                    M4M 2J8
 4 A'Court, K Susan   <NA>                    M4M 2J8
 5 A'Court, K Susan   <NA>                    M4M 2J8
 6 Aaron, Robert B    <NA>                    M6B 1H7
 7 Abadi, Babak       <NA>                    M5S 2W7
 8 Abadi, Babak       <NA>                    M5S 2W7
 9 Abadi, David       <NA>                    M5S 2W7
10 Abate, Frank       <NA>                    L4H 2K7
# i 10,189 more rows
# i 10 more variables: contribution_amount <chr>, contribution_type_desc <chr>,
#   goods_or_service_desc <chr>, contributor_type_desc <chr>,
#   relationship_to_candidate <chr>, president_business_manager <chr>,
#   authorized_representative <chr>, candidate <chr>, office <chr>, ward <chr>
```

Answer 4)

We have multiple variables in the dataset, detailing the information of both donors and the candidates they donated to. However, there is an issue where in many columns/Data fields are outright blank or have missing values, so upon exploration we see that columns like 'contributors_address','authorized_representative','president_business_manager','relationship_to_candidate' are mostly blank, presumably due to lack of information or privacy concerns. While, columns like 'goods_or_services_desc' are mostly blank due to very few donations in the name of goods or services. At the same time, 'contribution_amount' is recorded as a character data type while 'Contribution_type_desc' should be recorded as a factor. So in the next code block we implement all these changes

```
not_all_na <- function(x) all(!is.na(x))
campaign_data_mayoral <- campaign_data_mayoral|>
                          select(where(not_all_na))
campaign_data_mayoral
```

```
# A tibble: 10,199 x 7
   contributors_name  contributors_postal_code contribution_amount
   <chr>              <chr>                    <chr>
 1 A D'Angelo, Tullio M6A 1P5                  300
 2 A Strazar, Martin  M2M 3B8                  300
 3 A'Court, K Susan   M4M 2J8                  36
 4 A'Court, K Susan   M4M 2J8                  100
 5 A'Court, K Susan   M4M 2J8                  100
 6 Aaron, Robert B    M6B 1H7                  250
 7 Abadi, Babak       M5S 2W7                  500
```

```
 8 Abadi, Babak       M5S 2W7                      500
 9 Abadi, David       M5S 2W7                      300
10 Abate, Frank       L4H 2K7                      150
# i 10,189 more rows
# i 4 more variables: contribution_type_desc <chr>,
#   contributor_type_desc <chr>, candidate <chr>, office <chr>
```

```
campaign_data_mayoral$contributor_type_desc <- as.factor(campaign_data_mayoral$contributor
campaign_data_mayoral$contribution_type_desc <- as.factor(campaign_data_mayoral$contributi
campaign_data_mayoral$contribution_amount <- as.numeric(campaign_data_mayoral$contribution
campaign_data_mayoral
```

```
# A tibble: 10,199 x 7
   contributors_name  contributors_postal_code contribution_amount
   <chr>              <chr>                                  <dbl>
 1 A D'Angelo, Tullio M6A 1P5                                  300
 2 A Strazar, Martin  M2M 3B8                                  300
 3 A'Court, K Susan   M4M 2J8                                   36
 4 A'Court, K Susan   M4M 2J8                                  100
 5 A'Court, K Susan   M4M 2J8                                  100
 6 Aaron, Robert B    M6B 1H7                                  250
 7 Abadi, Babak       M5S 2W7                                  500
 8 Abadi, Babak       M5S 2W7                                  500
 9 Abadi, David       M5S 2W7                                  300
10 Abate, Frank       L4H 2K7                                  150
# i 10,189 more rows
# i 4 more variables: contribution_type_desc <fct>,
#   contributor_type_desc <fct>, candidate <chr>, office <chr>
```

Answer 5)

We use the following plots to see the distribution of contributions in the data

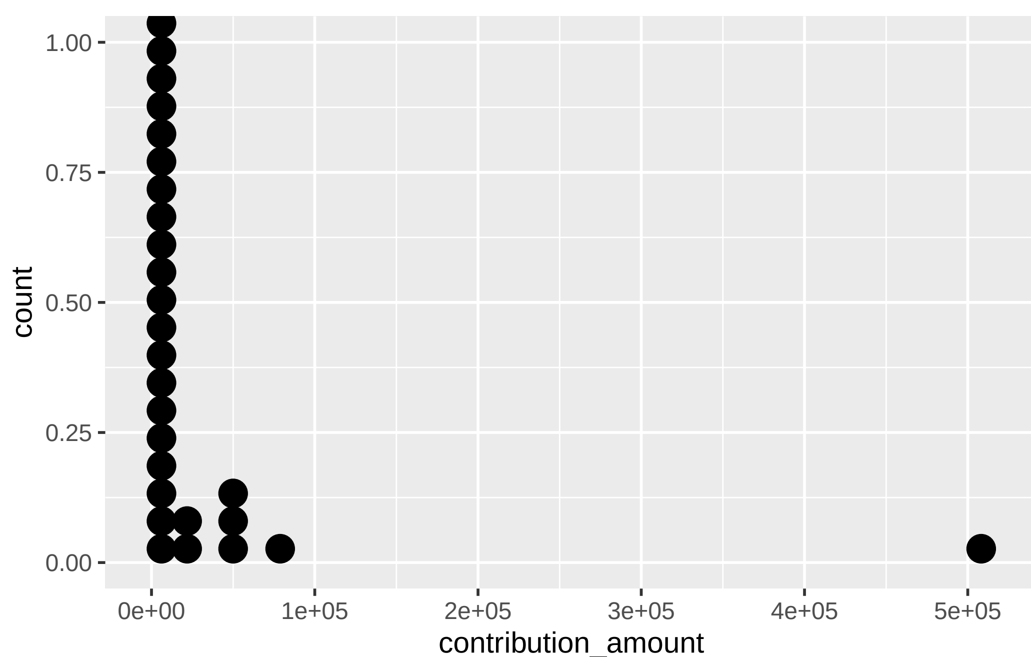
```
campaign_data_mayoral |> arrange(-contribution_amount)
```

```
# A tibble: 10,199 x 7
   contributors_name contributors_postal_code contribution_amount
   <chr>             <chr>                                  <dbl>
 1 Ford, Doug        M9A 2C3                              508225.
 2 Ford, Rob         M9A 3G9                               78805.
```

```
 3 Ford, Doug        M9A 2C3                               50000
 4 Ford, Rob         M9A 3G9                               50000
 5 Ford, Rob         M9A 3G9                               50000
 6 Goldkind, Ari     M5P 1P5                               23624.
 7 Ford, Rob         M9A 3G9                               20000
 8 Ford, Rob         M9A 3G9                               12210
 9 Di Paola, Rocco   M3H 2T1                                6000
10 Thomson, Sarah    M4W 2X6                                4426.
# i 10,189 more rows
# i 4 more variables: contribution_type_desc <fct>,
#   contributor_type_desc <fct>, candidate <chr>, office <chr>
```

```
ggplot(data = campaign_data_mayoral,aes(x=contribution_amount))+
  geom_dotplot()
```

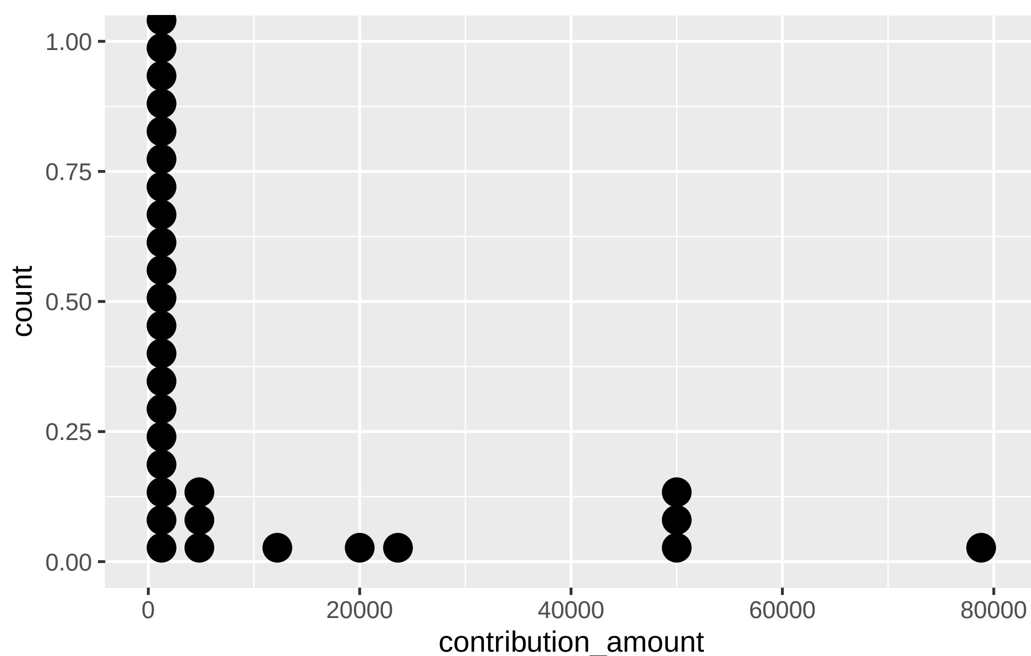
```
Bin width defaults to 1/30 of the range of the data. Pick better value with
`binwidth`.
```




```
campaign_data_mayoral_contribution_distribution <-
  campaign_data_mayoral |> filter(contribution_amount < 500000)
```

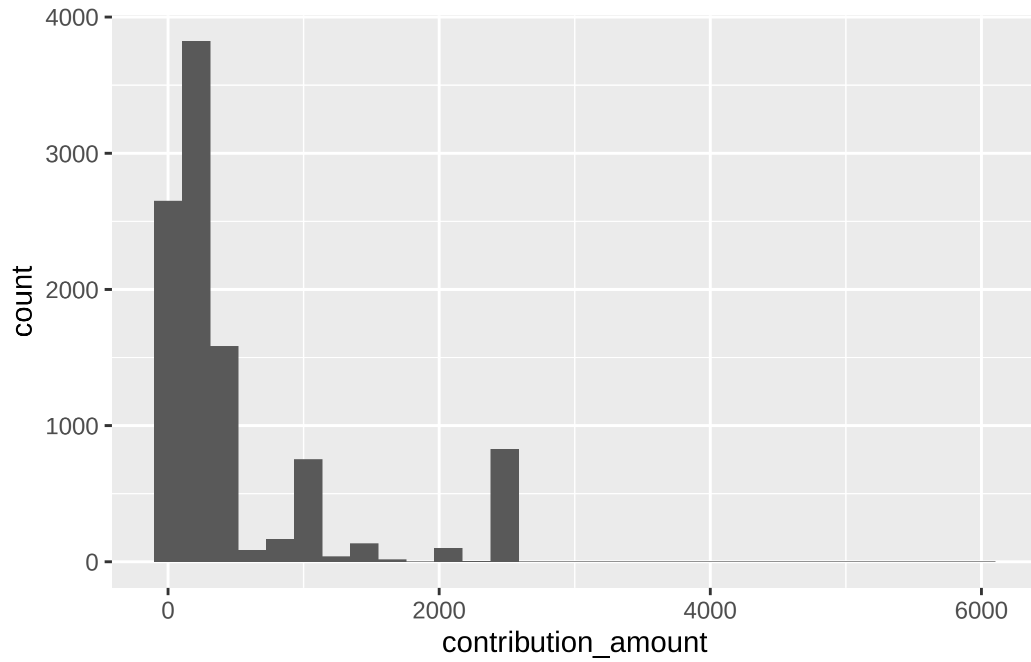
```r
ggplot(data=campaign_data_mayoral_contribution_distribution, aes(x=contribution_amount))+
  geom_dotplot()
```

```
Bin width defaults to 1/30 of the range of the data. Pick better value with
`binwidth`.
```



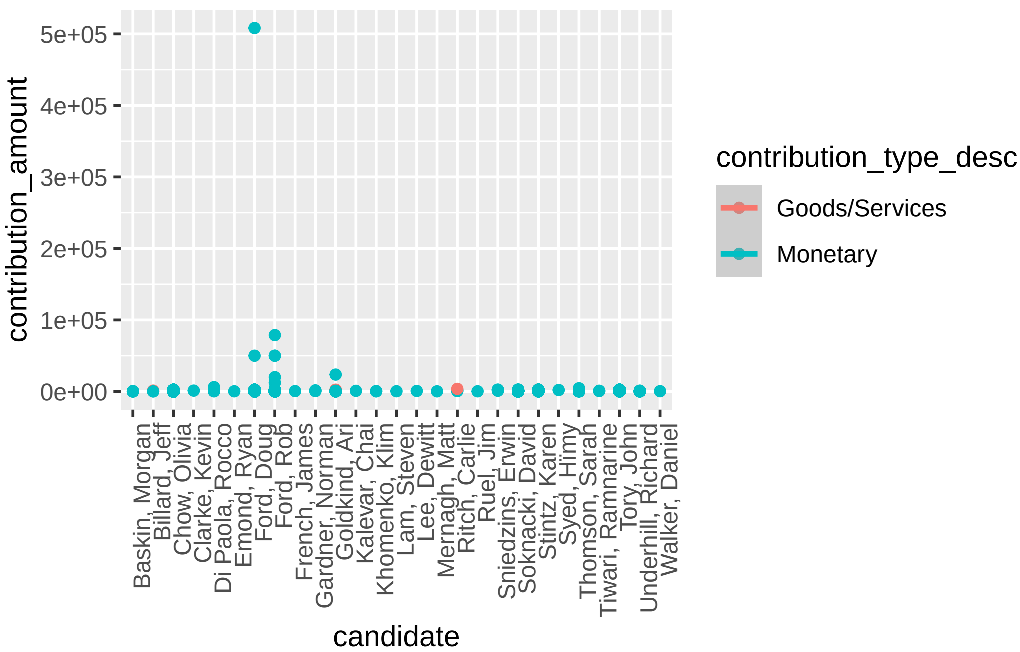

```r
campaign_data_mayoral_contribution_distribution_2 <-
  campaign_data_mayoral |> filter(contribution_amount < 10000)

ggplot(data=campaign_data_mayoral_contribution_distribution_2, aes(x=contribution_amount))
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = campaign_data_mayoral, aes(x = candidate,
                                          y = contribution_amount,
                                          color = contribution_type_desc ))+
  geom_point()+
  geom_smooth()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



In the multiple plots we obtained above we see the following. A main outlier is the self donation of $500,000 done by Doug Ford for himself there are also a few other big donations close to $100,000 done by candidates Ryan Emond and Rob Ford to themselves, these outliers are skewing the graphs of contribution amounts. Accounting for that and filtering for less than $100,000 contributions, we see that again some values above $20,000 are skewing the contribution amount count. Finally, adjusting for donations less than $10,000 we see a semblance of distribution of contributions, with a good majority being less than $2000.

Answer 6)

We can do the following task with the following code block

```
candidate_contri <- campaign_data_mayoral|>
                    group_by(candidate) |>
                    summarise(
                    total_contri = sum(contribution_amount, na.rm = TRUE),
                    mean_contri = mean(contribution_amount, na.rm = TRUE),
                    contri_count = n()
                    )

top_total_contri <- candidate_contri |>
```

```
                    arrange(-total_contri)|>
                    select(candidate,total_contri)|>
                    head(5)
top_mean_contri <- candidate_contri |>
                   arrange(-mean_contri)|>
                   select(candidate,mean_contri)|>
                   head(5)
top_contri_count <- candidate_contri |>
                    arrange(-contri_count)|>
                    select(candidate,contri_count)|>
                    head(5)

top_total_contri
```

```
# A tibble: 5 x 2
  candidate     total_contri
  <chr>                <dbl>
1 Tory, John        2767869.
2 Chow, Olivia      1638266.
3 Ford, Doug         889897.
4 Ford, Rob          387648.
5 Stintz, Karen      242805
```

```
top_mean_contri
```

```
# A tibble: 5 x 2
  candidate        mean_contri
  <chr>                  <dbl>
1 Sniedzins, Erwin       2025
2 Syed, Hïmy             2018
3 Ritch, Carlie          1887.
4 Ford, Doug             1456.
5 Clarke, Kevin          1200
```

```
top_contri_count
```

```
# A tibble: 5 x 2
  candidate       contri_count
  <chr>                  <int>
```

```
1 Chow, Olivia           5708
2 Tory, John             2602
3 Ford, Doug              611
4 Ford, Rob               538
5 Soknacki, David         314
```

Answer 7)

We can do this task with just a few revisions

```
non_candidate_contri <- campaign_data_mayoral |>
                        filter(contributors_name != candidate)

non_candidate_contri <- non_candidate_contri|>
                        group_by(candidate) |>
                        summarise(
                        total_contri_popular = sum(contribution_amount, na.rm = TRUE),
                        mean_contri_popular = mean(contribution_amount, na.rm = TRUE),
                        contri_count_popular = n()
                    )

top_total_contri_popular <- non_candidate_contri |>
                            arrange(-total_contri_popular)|>
                            select(candidate,total_contri_popular)|>
                            head(5)
top_mean_contri_popular <- non_candidate_contri |>
                           arrange(-mean_contri_popular)|>
                           select(candidate,mean_contri_popular)|>
                           head(5)
top_contri_count_popular <- non_candidate_contri |>
                            arrange(-contri_count_popular)|>
                            select(candidate,contri_count_popular)|>
                            head(5)
top_total_contri_popular
```

```
# A tibble: 5 x 2
  candidate     total_contri_popular
  <chr>                        <dbl>
1 Tory, John                2765369.
2 Chow, Olivia              1634766.
3 Ford, Doug                 331173.
```

```
4 Stintz, Karen             242805
5 Ford, Rob                 174510.
```

```
top_mean_contri_popular
```

```
# A tibble: 5 x 2
  candidate         mean_contri_popular
  <chr>                           <dbl>
1 Ritch, Carlie                   1887.
2 Sniedzins, Erwin                1867.
3 Tory, John                      1063.
4 Gardner, Norman                 1000
5 Tiwari, Ramnarine               1000
```

```
top_contri_count_popular
```

```
# A tibble: 5 x 2
  candidate       contri_count_popular
  <chr>                          <int>
1 Chow, Olivia                    5706
2 Tory, John                      2601
3 Ford, Doug                       608
4 Ford, Rob                        531
5 Soknacki, David                  314
```

Answer 8)

We can count the number of people who donated to multiple candidates with the following code:

```
multiple_contri <- campaign_data_mayoral |>
  group_by(contributors_name) |>
  summarise(unique_candidates = n_distinct(candidate))

multiple_contri_count <- sum(multiple_contri$unique_candidates > 1)

multiple_contri_count
```

```
[1] 184
```

We see that 184 people donated to multiple candidates.