EDA and data visualization

Monica Alexander

20/01/23

Table of contents


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

1 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)
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 Traffic Cameras   a330~ Trans~ <NA>         Transpor~ "This ~ Map
2 Police Facility ~ 9aee~ Locat~ <NA>         Toronto ~ "A geo~ Map
3 City Council and~ 3bfa~ City ~ <NA>         City Cle~ "This ~ Table
4 EarlyON Child an~ earl~ Commu~ Poverty red~ Children~ "Early~ Map
5 COVID-19 Immuniz~ d3f2~ Health <NA>         Toronto ~ "This ~ Map
6 Short Term Renta~ 2ab2~ Permi~ Affordable ~ Municipa~ "This ~ Table
# ... with 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)
```

```
# 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`
```

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

```
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
# ... with 1 more variable: vehicle <dbl>
```

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

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

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

```
Joining, by = "code"
```

```
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)
```

```
Joining, by = "code_srt"
```

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

2 Lab Exercises

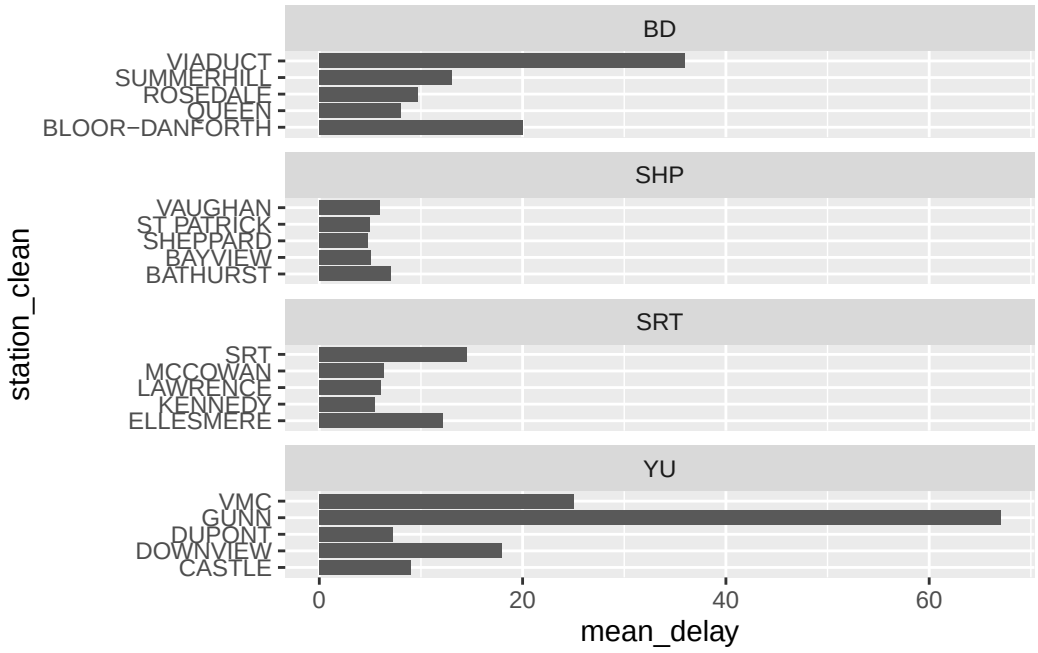
To be handed in via submission of quarto file (and rendered pdf) to GitHub.

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)) |>
```

```
    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()
```

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



2. Using the `opendatatoronto` package, download the data on mayoral campaign contributions for 2014. 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

```
res <- list_package_resources("f6651a40-2f52-46fc-9e04-b760c16edd5c") # obtained code
res_data <-get_resource("5b230e92-0a22-4a15-9572-0b19cc222985")
```

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

```
res_readme <-get_resource("aaf736f4-7468-4bda-9a66-4bb592e9c63c")
```

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

```
data<-res_data[["2_Mayor_Contributions_2014_election.xls"]]
```

3. Clean up the data format (fixing the parsing issue and standardizing the column names using `janitor`)

```
# use janitor to make first row as column names
data<-data |> row_to_names(row_number = 1)

# make the column names nicer to work with
data <- clean_names(data)
```

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.

```
skim(data)
```

Table 1: Data summary

Name	data
Number of rows	10199
Number of columns	13

Table 1: Data summary

Column type frequency:	
character	13
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
contributors_name	0	1	4	31	0	7545	0
contributors_address	10197	0	24	26	0	2	0
contributors_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

There are missing values in columns 'contributors_address', 'goods_or_service_desc', 'relationship_to_candidate', 'president_business_manager', 'authorized_representative', and 'ward'. We should worry about the missing values because the missing% is huge, nearly the entire columns are missing. In addition, there are total 13 columns, but 6 columns are missing, hence a problem.

```
summary(data)
```

```
 contributors_name  contributors_address contributors_postal_code
 Length:10199       Length:10199         Length:10199
 Class :character   Class :character     Class :character
 Mode  :character   Mode  :character     Mode  :character
 contribution_amount contribution_type_desc goods_or_service_desc
 Length:10199        Length:10199           Length:10199
 Class :character    Class :character       Class :character
```

```
 Mode  :character    Mode  :character       Mode  :character
 contributor_type_desc relationship_to_candidate president_business_manager
 Length:10199          Length:10199              Length:10199
 Class :character      Class :character          Class :character
 Mode  :character      Mode  :character          Mode  :character
 authorized_representative  candidate            office
 Length:10199              Length:10199       Length:10199
 Class :character          Class :character   Class :character
 Mode  :character          Mode  :character   Mode  :character
     ward
 Length:10199
 Class :character
 Mode  :character
```

'contribution_amount' should be in numeric format.

```
# verify there is no char value and all values can be converted to numeric
# unique(data$contribution_amount)

data<- data|>
  mutate(contribution_amount_num=as.numeric(contribution_amount))
```

```
# list unique values for each column in data
# sapply(data, unique)

unique(data$goods_or_service_desc)
```

```
 [1] NA
 [2] "musical services at Chowstock fundraiser"
 [3] "Accounting/bookkeeping"
 [4] "Accounting services"
 [5] "web hosting and design"
 [6] "photography"
 [7] "advertising"
 [8] "musical services Chowstock fundraiser"
 [9] "TV and bracket"
[10] "pizza for volunteers"
```

Two values in 'goods_or_service_desc' are the same thing (musical services at Chowstock fundraiser, musical services Chowstock fundraiser). May need to convert to the same value later if using this column.

Some contributor names are in uppercase letters, hence converting all names related columns into lowercase letters for convenience.

```
data$contributors_name<-tolower(data$contributors_name)
data$candidate<-tolower(data$candidate)
```

```
# there are duplicates in the data, but these may bot be actual duplicates since many of t
get_dupes(data)
```

```
No variable names specified - using all columns.
```

```
# A tibble: 1,716 x 15
   contributors_name contributors_address contributors_postal_~ contribution_am~
   <chr>             <chr>                <chr>                 <chr>
 1 a'court, k susan  <NA>                 M4M 2J8               100
 2 a'court, k susan  <NA>                 M4M 2J8               100
 3 adain, jacqueline <NA>                 M4C 5N8               100
 4 adain, jacqueline <NA>                 M4C 5N8               100
 5 adams, don        <NA>                 M4L 3A5               25
 6 adams, don        <NA>                 M4L 3A5               25
 7 adams, don        <NA>                 M4L 3A5               25
 8 adams, marion     <NA>                 K0C 2K0               300
 9 adams, marion     <NA>                 K0C 2K0               300
10 agnew, arel       <NA>                 M6G 1V2               100
# ... with 1,706 more rows, and 11 more variables:
#   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>, contribution_amount_num <dbl>,
#   dupe_count <int>
```

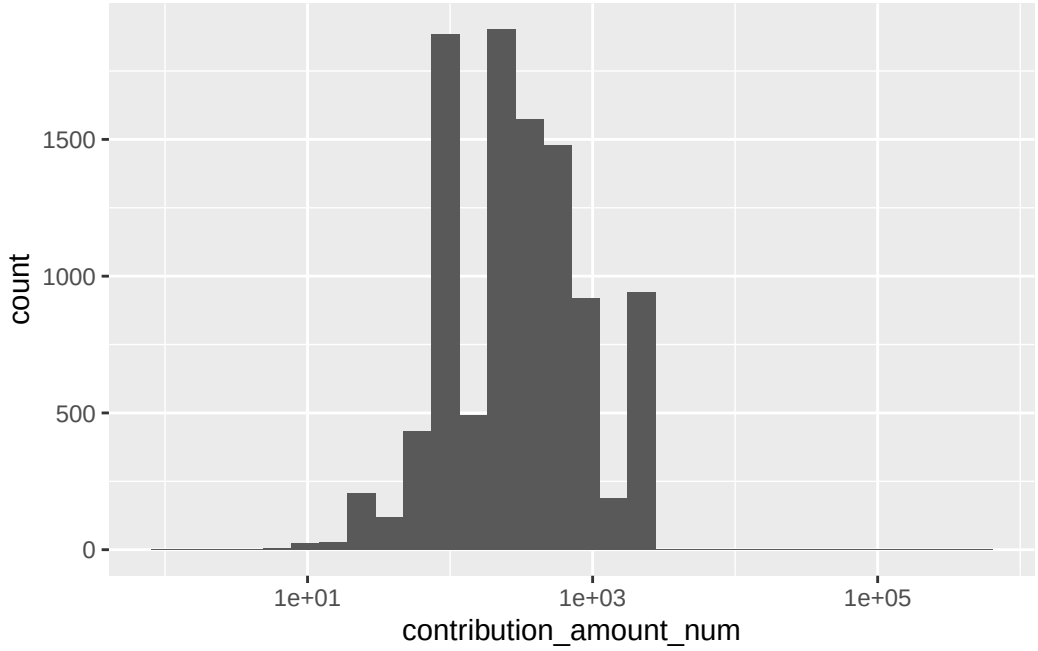
5. Visually explore the distribution of values of the contributions. What contributions are notable outliers? Do they share a similar characteristic(s)? It may be useful to plot the distribution of contributions without these outliers to get a better sense of the majority of the data.

```
summary(data$contribution_amount_num)
```

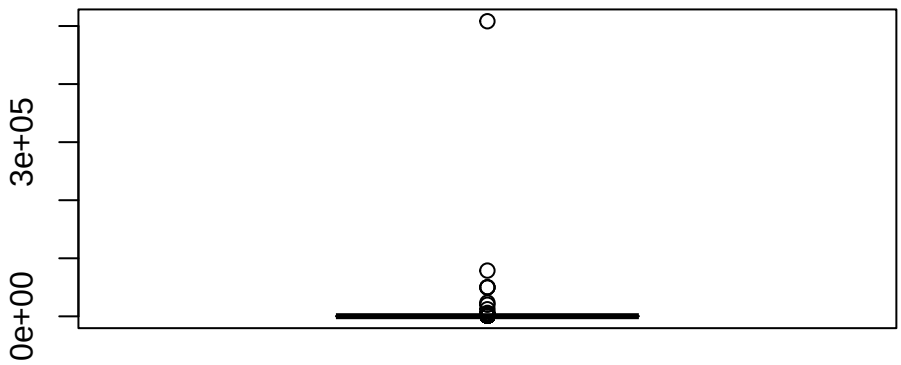
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1     100     300     608     500  508225
```

```r
# because of the large outlier, the small numbers squeezed too closely,
# so plot in log scale to see all the numbers easily
ggplot(data = data) +
  geom_histogram(aes(x = contribution_amount_num)) +
  scale_x_log10()
```

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




```r
# find outliers from boxplot
outliers<-boxplot(data$contribution_amount_num)
```




```
outliers$stats
```

```
     [,1]
[1,]    1
[2,]  100
[3,]  300
[4,]  500
[5,] 1100
```

There is an extremely large amount 508225, and the outliers are outside the extreme whiskers of the boxplot (<1 or >1100).

```
# find common characteristics of outliers
out<-outliers$out
summary(out)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1150    2059    2500    2966    2500  508225
```

```r
data1<- data |>
  filter(contribution_amount_num %in% out)

# extract the first part of postal code to find a pattern
data1 <- data1 |>
  mutate(postal_code_area = word(contributors_postal_code, 1))
```

```r
# list the 5 most common values in contribution_type_desc, contributor_type_desc,
# candidate, and postal_code_area within the outliers
data1 |>
  group_by(contribution_type_desc) |>
  summarise(n = n()) |>
  arrange(-n) |>
  slice(1:5)
```

```
# A tibble: 2 x 2
  contribution_type_desc     n
  <chr>                  <int>
1 Monetary                1134
2 Goods/Services             5
```

```r
data1 |>
  group_by(contributor_type_desc) |>
  summarise(n = n()) |>
  arrange(-n) |>
  slice(1:5)
```

```
# A tibble: 2 x 2
  contributor_type_desc     n
  <chr>                 <int>
1 Individual             1138
2 Corporation               1
```

```r
data1 |>
  group_by(candidate) |>
  summarise(n = n()) |>
  arrange(-n) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate         n
  <chr>         <int>
1 tory, john      770
2 chow, olivia    135
3 stintz, karen    82
4 ford, doug       67
5 ford, rob        33
```

```
data1 |>
  group_by(postal_code_area) |>
  summarise(n = n()) |>
  arrange(-n) |>
  slice(1:5)
```

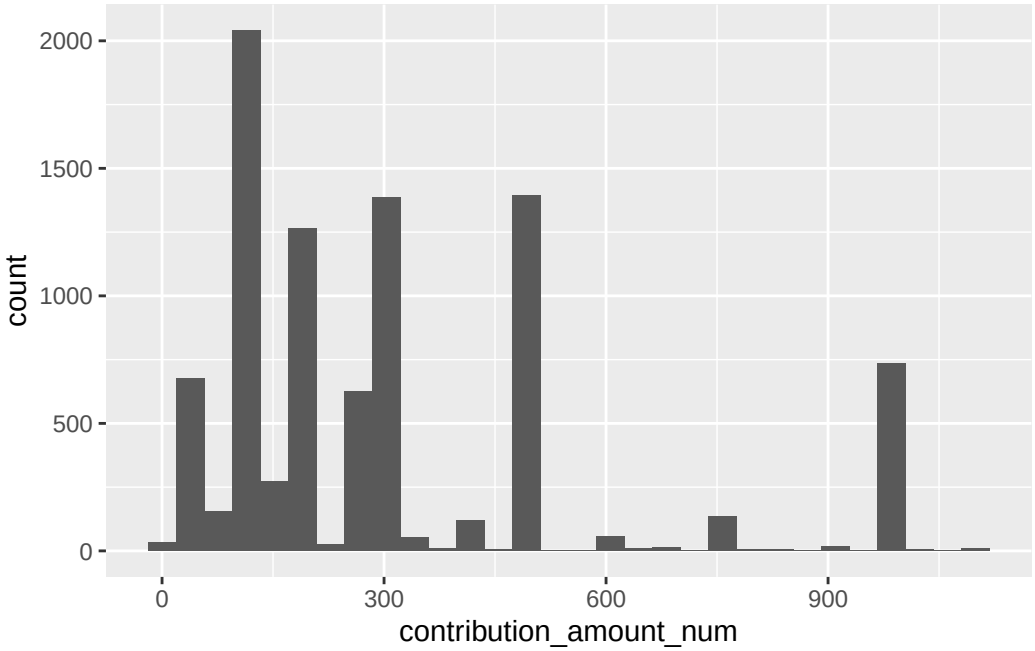
```
# A tibble: 5 x 2
  postal_code_area     n
  <chr>            <int>
1 M4W                137
2 M4V                 89
3 M5R                 84
4 M4N                 74
5 M9A                 49
```

Most outliers make monetary contribution and are individual contributor. Tory, John is the most common candidate within the outliers. Most outliers are in the postal area M4W.

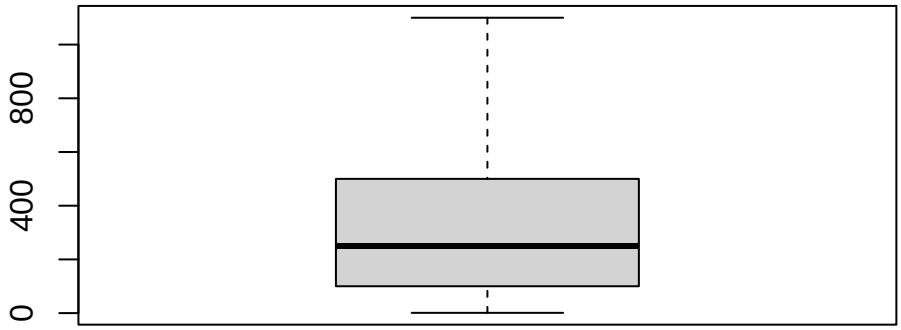
```
# plot histogram after removing outliers
data2<- data |>
  filter(!(contribution_amount_num %in% out))

ggplot(data = data2) +
  geom_histogram(aes(x = contribution_amount_num))
```

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




```
boxplot(data2$contribution_amount_num)
```



```
summary(data2$contribution_amount_num)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0   100.0   250.0   311.6   500.0  1100.0
```

Majority of contribution amounts are between 100 and 500.

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

   - total contributions
   - mean contribution
   - number of contributions

```
data |>
  group_by(candidate) |>
  summarise(total_contributions = sum(contribution_amount_num)) |>
  arrange(-total_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate     total_contributions
  <chr>                       <dbl>
1 tory, john                2767869.
2 chow, olivia              1638266.
3 ford, doug                 889897.
4 ford, rob                  387648.
5 stintz, karen              242805
```

```
data |>
  group_by(candidate) |>
  summarise(mean_contributions = mean(contribution_amount_num)) |>
  arrange(-mean_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate        mean_contributions
  <chr>                         <dbl>
1 sniedzins, erwin              2025
2 syed, hïmy                    2018
3 ritch, carlie                 1887.
4 ford, doug                    1456.
5 clarke, kevin                 1200
```

```
data |>
  group_by(candidate) |>
  summarise(number_of_contributions = n()) |>
  arrange(-number_of_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate       number_of_contributions
  <chr>                             <int>
1 chow, olivia                       5708
2 tory, john                         2602
3 ford, doug                          611
4 ford, rob                           538
5 soknacki, david                     314
```

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

```
data |>
  filter(contributors_name != candidate) |>
  group_by(candidate) |>
  summarise(total_contributions = sum(contribution_amount_num)) |>
  arrange(-total_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate     total_contributions
  <chr>                       <dbl>
1 tory, john                2765369.
2 chow, olivia              1634766.
3 ford, doug                 331173.
4 stintz, karen              242805
5 ford, rob                  174510.
```

```
data |>
  filter(contributors_name != candidate) |>
  group_by(candidate) |>
  summarise(mean_contributions = mean(contribution_amount_num)) |>
  arrange(-mean_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate         mean_contributions
  <chr>                          <dbl>
1 ritch, carlie                  1887.
2 sniedzins, erwin               1867.
3 tory, john                     1063.
4 gardner, norman                1000
5 tiwari, ramnarine              1000
```

```
data |>
  filter(contributors_name != candidate) |>
  group_by(candidate) |>
  summarise(number_of_contributions = n()) |>
  arrange(-number_of_contributions) |>
  slice(1:5)
```

```
# A tibble: 5 x 2
  candidate       number_of_contributions
  <chr>                             <int>
1 chow, olivia                       5706
2 tory, john                         2601
3 ford, doug                          608
4 ford, rob                           531
5 soknacki, david                     314
```

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

```
data |>
  group_by(contributors_name) |>
  summarise(number_of_candidate=n_distinct(candidate)) |>
  filter(number_of_candidate>1) |>
  count()
```

```
# A tibble: 1 x 1
      n
  <int>
1   200
```

200 contributors gave money to more than one candidate.