# STA2201 R code

Quynh Vu

2023-01-20

## Week 1

```
library(tidyverse)
```

```
dm <- read_table("https://www.prdh.umontreal.ca/BDLC/data/ont/Mx_1x1.txt",
                 skip = 2, col_types = "dcddd")
head(dm)
```

```
## # A tibble: 6 x 5
##    Year Age    Female    Male   Total
##   <dbl> <chr>   <dbl>   <dbl>   <dbl>
## 1  1921 0     0.0978  0.129   0.114
## 2  1921 1     0.0129  0.0144  0.0137
## 3  1921 2     0.00521 0.00737 0.00631
## 4  1921 3     0.00471 0.00457 0.00464
## 5  1921 4     0.00461 0.00433 0.00447
## 6  1921 5     0.00372 0.00361 0.00367
```

```
# skip: Number of lines to skip before reading data.
# col_types: d = double; c = character
```

### 1. Tidyverse functions

The pipe `|>` or `%>%` in `magrittr` package (read as "and then") is used to manipulate the tibbles

### a) Piping, filtering, selecting, arranging

e.g. Year 1935

```
filter(dm, Year==1935) # or
```

```
## # A tibble: 111 x 5
##     Year Age     Female    Male   Total
##    <dbl> <chr>    <dbl>   <dbl>   <dbl>
##  1  1935 0     0.0513   0.0652  0.0584
##  2  1935 1     0.00607  0.00742 0.00676
##  3  1935 2     0.00350  0.00321 0.00336
##  4  1935 3     0.00187  0.00321 0.00255
##  5  1935 4     0.0013   0.00238 0.00185
##  6  1935 5     0.00152  0.00186 0.00169
##  7  1935 6     0.00136  0.00174 0.00155
##  8  1935 7     0.00120  0.00154 0.00137
##  9  1935 8     0.000984 0.00130 0.00114
## 10  1935 9     0.000996 0.00140 0.00120
```

```
## # ... with 101 more rows
```

```
dm |> filter(Year==1935)
```

```
## # A tibble: 111 x 5
##     Year Age     Female    Male   Total
##    <dbl> <chr>    <dbl>   <dbl>   <dbl>
##  1  1935 0     0.0513   0.0652  0.0584
##  2  1935 1     0.00607  0.00742 0.00676
##  3  1935 2     0.00350  0.00321 0.00336
##  4  1935 3     0.00187  0.00321 0.00255
##  5  1935 4     0.0013   0.00238 0.00185
##  6  1935 5     0.00152  0.00186 0.00169
##  7  1935 6     0.00136  0.00174 0.00155
##  8  1935 7     0.00120  0.00154 0.00137
##  9  1935 8     0.000984 0.00130 0.00114
## 10  1935 9     0.000996 0.00140 0.00120
## # ... with 101 more rows
```

e.g. 10 year olds in Year 1935

```
dm |> filter(Year==1935, Age==10)
```

```
## # A tibble: 1 x 5
##    Year Age     Female    Male   Total
##   <dbl> <chr>    <dbl>   <dbl>   <dbl>
## 1  1935 10    0.000884 0.00143 0.00116
```

e.g. 10 year olds in 1935 who were female

```
dm |> filter(Year==1935, Age==10) |> select(Female)
```

```
## # A tibble: 1 x 1
##     Female
##      <dbl>
## 1 0.000884
```

e.g. Remove column

```
colnames(dm)
```

```
## [1] "Year"   "Age"    "Female" "Male"   "Total"
```

```
dm |> select(-Total)
```

```
## # A tibble: 10,989 x 4
##     Year Age    Female    Male
##    <dbl> <chr>   <dbl>   <dbl>
##  1  1921 0     0.0978  0.129
##  2  1921 1     0.0129  0.0144
##  3  1921 2     0.00521 0.00737
##  4  1921 3     0.00471 0.00457
##  5  1921 4     0.00461 0.00433
##  6  1921 5     0.00372 0.00361
##  7  1921 6     0.00265 0.00393
##  8  1921 7     0.00295 0.00351
##  9  1921 8     0.00237 0.00285
## 10  1921 9     0.00198 0.00255
## # ... with 10,979 more rows
```

e.g. sort Year in descending order

```
dm |> arrange(-Year)
```

```
## # A tibble: 10,989 x 5
##     Year Age     Female     Male    Total
##    <dbl> <chr>    <dbl>    <dbl>    <dbl>
##  1  2019 0     0.00423  0.00481  0.00453
##  2  2019 1     0.000216 0.000177 0.000196
##  3  2019 2     0.000157 0.000162 0.00016
##  4  2019 3     0.00007  0.00016  0.000117
##  5  2019 4     0.000111 0.000132 0.000122
##  6  2019 5     0.000096 0.000052 0.000074
##  7  2019 6     0.000081 0.000039 0.000059
##  8  2019 7     0.000107 0.000128 0.000118
##  9  2019 8     0.000066 0.000026 0.000046
## 10  2019 9     0.000052 0.000177 0.000116
## # ... with 10,979 more rows
```

### b) Grouping, summarizing, mutating

e.g. ratio of male to female mortality at each age and year

```
dm <- dm |> mutate(mf_ratio = Male/Female) # create new variables
```

e.g. mean female mortality rate by age over all the years

```
summary_mean <- dm |> group_by(Age) |> summarize(mean_mortality = mean(Female, na.rm = TRUE))
dim(summary_mean)
```

```
## [1] 111   2
```

```
dim(dm)
```

```
## [1] 10989     6
```

e.g. apply `mean` function across Male and Female columns by `across`

```
dm |> group_by(Age) |> summarize(mean_mortality_f = mean(Female, na.rm = TRUE),
                                 mean_mortality_m = mean(Male, na.rm = TRUE))
```

```
## # A tibble: 111 x 3
##    Age   mean_mortality_f mean_mortality_m
##    <chr>            <dbl>            <dbl>
##  1 0             0.0254           0.0322
##  2 1             0.00262          0.00297
##  3 10            0.000426         0.000590
##  4 100           0.426            0.462
##  5 101           0.448            0.493
##  6 102           0.493            0.566
##  7 103           0.533            0.647
##  8 104           0.660            0.780
##  9 105           0.805            0.904
## 10 106           0.796            0.720
## # ... with 101 more rows
```

```
dm |> group_by(Age) |> summarize(across(Male:Female, mean))
```

```
## # A tibble: 111 x 3
```

```
##    Age        Male    Female
##    <chr>     <dbl>     <dbl>
##  1 0      0.0322    0.0254
##  2 1      0.00297   0.00262
##  3 10     0.000590  0.000426
##  4 100    0.462     0.426
##  5 101    0.493     0.448
##  6 102    0.566     0.493
##  7 103    0.647     0.533
##  8 104    NA        0.660
##  9 105    NA        NA
## 10 106    NA        NA
## # ... with 101 more rows
```

The `summarize` function produces summary statistics

### c) Pivoting

e.g. wide to long

```
dm_long <- dm |> select(-mf_ratio) |>
                          pivot_longer(Female:Total, names_to = "sex", values_to = "mortality")
head(dm_long)
```

```
## # A tibble: 6 x 4
##    Year Age   sex    mortality
##   <dbl> <chr> <chr>      <dbl>
## 1  1921 0     Female    0.0978
## 2  1921 0     Male      0.129
## 3  1921 0     Total     0.114
## 4  1921 1     Female    0.0129
## 5  1921 1     Male      0.0144
## 6  1921 1     Total     0.0137
```

e.g. long to wide

```
dm_long |> pivot_wider(names_from = "sex", values_from = "mortality")
```

```
## # A tibble: 10,989 x 5
##     Year Age    Female    Male   Total
##    <dbl> <chr>   <dbl>   <dbl>   <dbl>
##  1  1921 0     0.0978  0.129   0.114
##  2  1921 1     0.0129  0.0144  0.0137
##  3  1921 2     0.00521 0.00737 0.00631
##  4  1921 3     0.00471 0.00457 0.00464
##  5  1921 4     0.00461 0.00433 0.00447
##  6  1921 5     0.00372 0.00361 0.00367
##  7  1921 6     0.00265 0.00393 0.00330
##  8  1921 7     0.00295 0.00351 0.00323
##  9  1921 8     0.00237 0.00285 0.00262
## 10  1921 9     0.00198 0.00255 0.00227
## # ... with 10,979 more rows
```

```
head(dm_long)
```

```
## # A tibble: 6 x 4
##    Year Age   sex    mortality
```

```
##   <dbl> <chr> <chr>      <dbl>
## 1  1921 0     Female    0.0978
## 2  1921 0     Male      0.129
## 3  1921 0     Total     0.114
## 4  1921 1     Female    0.0129
## 5  1921 1     Male      0.0144
## 6  1921 1     Total     0.0137
```

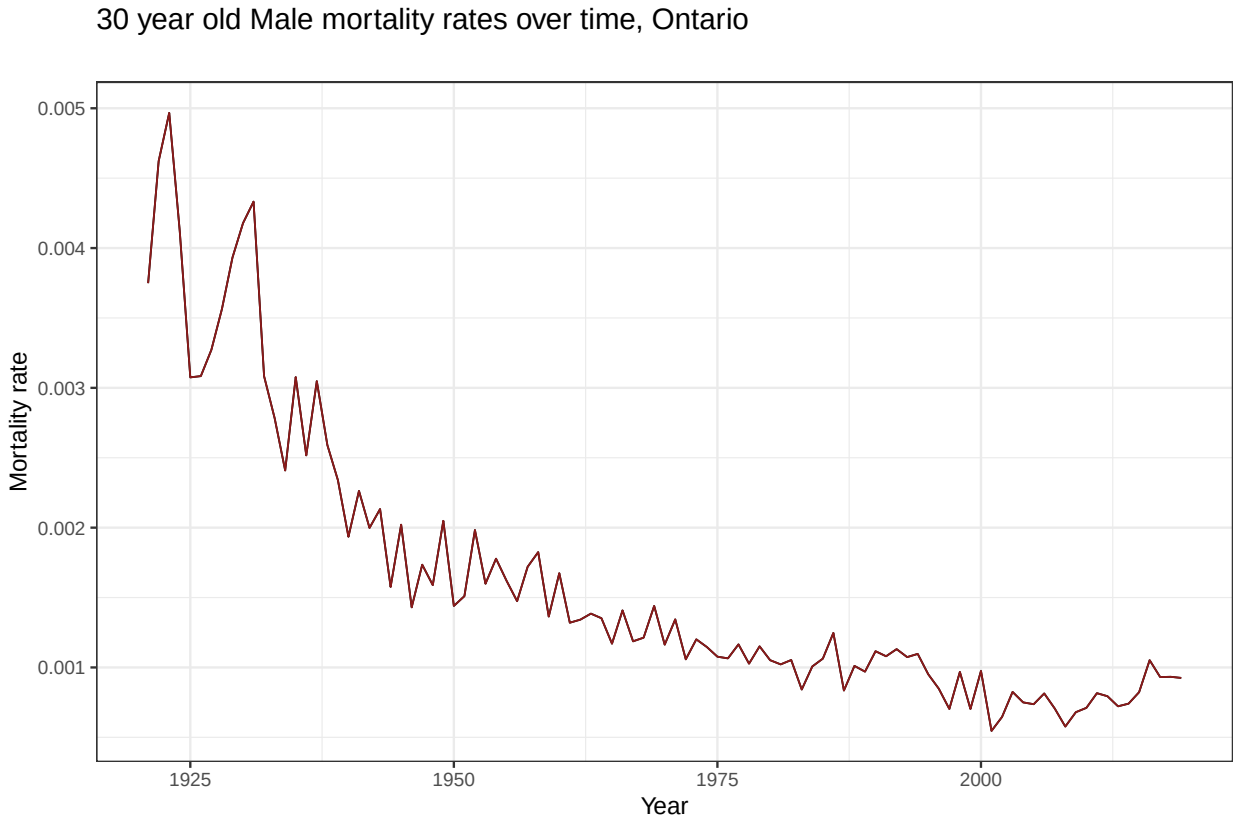
## 2. ggplot()

e.g. mortality rates for 30 year old males over time

```
d_to_plot <- dm |> filter(Age==30) |> select(Year, Male)
head(d_to_plot)
```

```
## # A tibble: 6 x 2
##    Year    Male
##   <dbl>   <dbl>
## 1  1921 0.00375
## 2  1922 0.00462
## 3  1923 0.00497
## 4  1924 0.00412
## 5  1925 0.00308
## 6  1926 0.00308
```

```
p <- ggplot(data = d_to_plot, aes(x = Year, y = Male))
p + # an empty box
  geom_line() + # specify that we want a line plot
  geom_line(color = "firebrick4") + # color of the line
  labs(title = "30 year old Male mortality rates over time, Ontario",
       subtitle = "", y = "Mortality rate") +
  theme_bw(base_size = 14)
```




e.g. trends for 30-year old males and females on the one plot

```
dp <- dm |> filter(Age==30) |> select(Year:Male) |>
            pivot_longer(Female:Male, names_to = "Sex", values_to = "Mortality")
head(dp)
```

```
## # A tibble: 6 x 4
##    Year Age   Sex    Mortality
##   <dbl> <chr> <chr>      <dbl>
## 1  1921 30    Female   0.00486
## 2  1921 30    Male     0.00375
## 3  1922 30    Female   0.00510
## 4  1922 30    Male     0.00462
## 5  1923 30    Female   0.00429
## 6  1923 30    Male     0.00497
```

# Week 2: Exploratory data analysis (EDA) and data visualization

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

```r
all_data <- list_packages(limit = 500) # to look whats available
head(all_data)
```

```
## # A tibble: 6 x 11
##   title     id    topics civic~1 publi~2 excerpt datas~3 num_r~4 formats refre~5
##   <chr>     <chr> <chr>  <chr>   <chr>   <chr>   <chr>     <int> <chr>   <chr>
## 1 COVID-19~ d3f2~ Health <NA>    Toront~ "This ~ Map          13 SHP,GP~ Daily
## 2 Short Te~ 2ab2~ Permi~ Afford~ Munici~ "This ~ Table         4 JSON,C~ Daily
## 3 Traffic ~ a330~ Trans~ <NA>    Transp~ "This ~ Map          12 XSD,SH~ As ava~
## 4 Polls co~ 7bce~ City ~ <NA>    City C~ "Polls~ Table         5 JSON,X~ Daily
## 5 Rain Gau~ f293~ Locat~ Climat~ Toront~ "This ~ Docume~      11 DOCX,C~ Monthly
## 6 Developm~ 0aa7~ <NA>   <NA>    City P~ "This ~ Table         4 JSON,C~ Monthly
## # ... with 1 more variable: last_refreshed <date>, and abbreviated variable
## #   names 1: civic_issues, 2: publisher, 3: dataset_category, 4: num_resources,
## #   5: refresh_rate
```

```r
# obtained code from searching data frame above
res <- list_package_resources("996cfe8d-fb35-40ce-b569-698d51fc683b")
 # extracts the first complete match from each string
res <- res |> mutate(year = str_extract(name, "202.?"))
head(res)
```

```
## # A tibble: 6 x 5
##   name                                id                format last_mod~1 year
##   <chr>                               <chr>             <chr>  <date>     <chr>
## 1 ttc-subway-delay-codes              3900e649-f31e-4b~ XLSX   2022-04-06 <NA>
## 2 ttc-subway-delay-data-readme        ca43ac3d-3940-43~ XLSX   2022-04-06 <NA>
## 3 ttc-subway-delay-jan-2014-april-2017 8ca4a6ed-5e7e-4b~ XLSX   2022-04-06 <NA>
## 4 ttc-subway-delay-may-december-2017  e2ee9f63-3130-4d~ XLSX   2022-04-06 <NA>
## 5 ttc-subway-delay-data-2018          32bd0973-e83d-4d~ XLSX   2022-04-06 <NA>
## 6 ttc-subway-delay-data-2019          1df6aace-fa16-40~ XLSX   2022-04-06 <NA>
## # ... with abbreviated variable name 1: last_modified
```

```r
delay_2022_ids <- res |> filter(year==2022)|> select(id) |>
                        pull() # Extract a single column
delay_2022 <- get_resource(delay_2022_ids) # download a resource into R
# from janitor to make the column names nicer to work with
delay_2022 <- clean_names(delay_2022)
```

```r
# download the delay code and readme, as reference.
delay_codes <- get_resource("3900e649-f31e-4b79-9f20-4731bbfd94f7")
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_d~1 min_gap bound line
##   <dttm>              <chr> <chr>    <chr>     <chr>   <dbl>   <dbl> <chr> <chr>
## 1 2022-01-01 00:00:00 15:59 Saturday LAWRENCE~ SRDP        0       0 N     SRT
## 2 2022-01-01 00:00:00 02:23 Saturday SPADINA ~ MUIS        0       0 <NA>  BD
## 3 2022-01-01 00:00:00 22:00 Saturday KENNEDY ~ MRO         0       0 <NA>  SRT
## 4 2022-01-01 00:00:00 02:28 Saturday VAUGHAN ~ MUIS        0       0 <NA>  YU
## 5 2022-01-01 00:00:00 02:34 Saturday EGLINTON~ MUATC       0       0 S     YU
## 6 2022-01-01 00:00:00 05:40 Saturday QUEEN ST~ MUNCA       0       0 <NA>  YU
## # ... with 1 more variable: vehicle <dbl>, and abbreviated variable name
```

```
## #   1: min_delay
```

## 1. EDA and data vizualization

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.

## a) 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?)

```
unique(delay_2022$day)  # check days of week
```

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

```
unique(delay_2022$line) # some have obvious recodes, others, not so much.
```

```
##  [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"
```

```
# skim a data frame, getting useful summary statistics
# skim(delay_2022)
```

**Missing values:**

```
delay_2022 |>
           summarize(across(everything(), ~ sum(is.na(.)))) # Calculate number of NAs by column
```

```
## # 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  4975    36       0
```

```
# vis_dat(delay_2022)  visualises a data.frame
# vis_miss(delay_2022) to see how missing values are distributed
```

**Duplicates:**

```
get_dupes(delay_2022) # from janitor package
```

```
## # A tibble: 26 x 11
##    date                time  day       station code  min_d~1 min_gap bound line
##    <dttm>              <chr> <chr>     <chr>   <chr>   <dbl>   <dbl> <chr> <chr>
##  1 2022-01-12 00:00:00 13:27 Wednesday FINCH ~ TUNOA       3       6 S     YU
##  2 2022-01-12 00:00:00 13:27 Wednesday FINCH ~ TUNOA       3       6 S     YU
```

```
##  3 2022-01-12 00:00:00 17:49 Wednesday FINCH ~ TUNOA       3       6 S     YU
##  4 2022-01-12 00:00:00 17:49 Wednesday 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 Thursday  YONGE ~ TUST        0       0 <NA>  YU
##  8 2022-01-20 00:00:00 02:30 Thursday  YONGE ~ TUST        0       0 <NA>  YU
##  9 2022-01-20 00:00:00 08:51 Thursday  WILSON~ TUNOA       3       6 S     YU
## 10 2022-01-20 00:00:00 08:51 Thursday  WILSON~ TUNOA       3       6 S     YU
## # ... with 16 more rows, 2 more variables: vehicle <dbl>, dupe_count <int>, and
## #   abbreviated variable name 1: min_delay
```

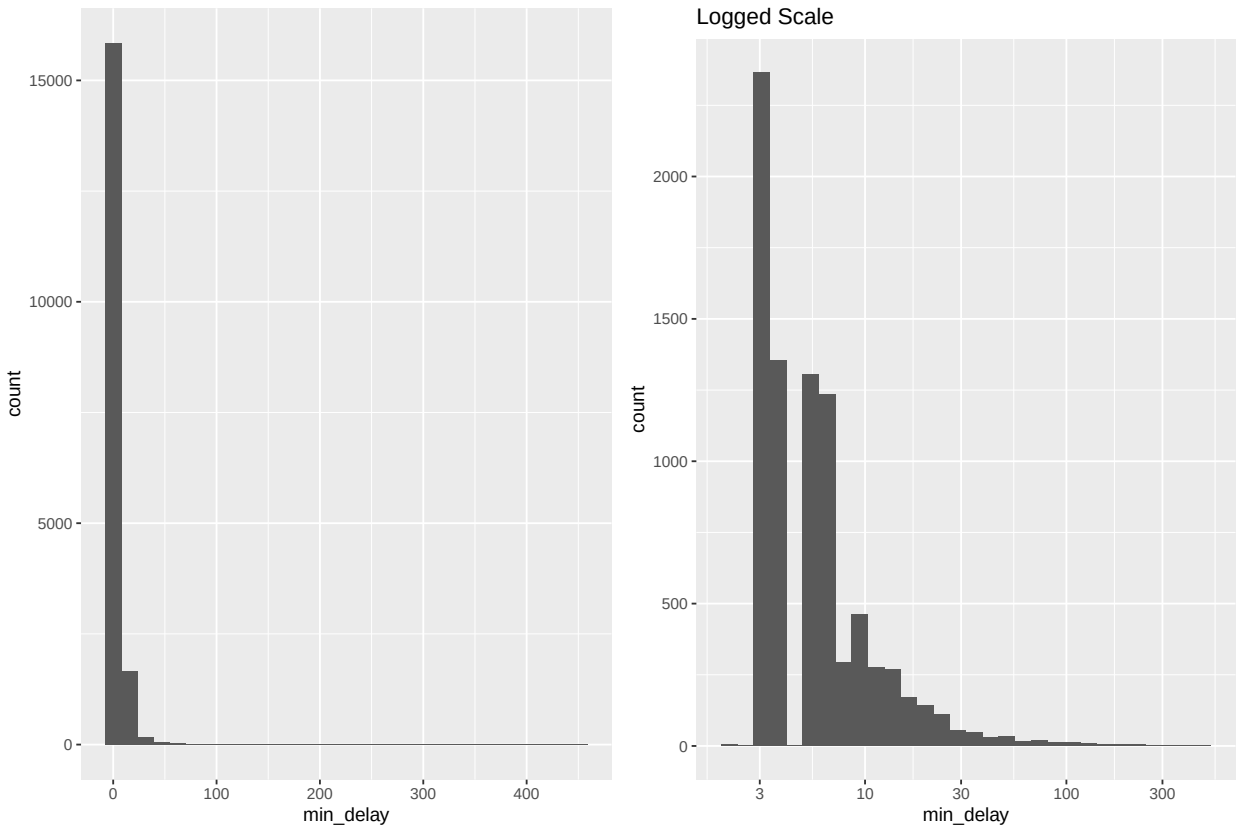
```
delay_2022 <-delay_2022 |> distinct() # subset distinct/unique rows
```

## b) Visualizing distributions

```
## Removing the observations that have non-standardized lines
delay_2022 <- delay_2022 |> filter(line %in% c("BD", "YU", "SHP", "SRT"))

ggp1 <- ggplot(data = delay_2022) + geom_histogram(aes(x = min_delay))

ggp2 <- ggplot(data = delay_2022) + geom_histogram(aes(x = min_delay)) +
        labs(title = "Logged Scale") +
        scale_x_log10() # on logged scale
require(gridExtra)
grid.arrange(ggp1, ggp2, ncol=2)
```




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...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 "Signals Other".

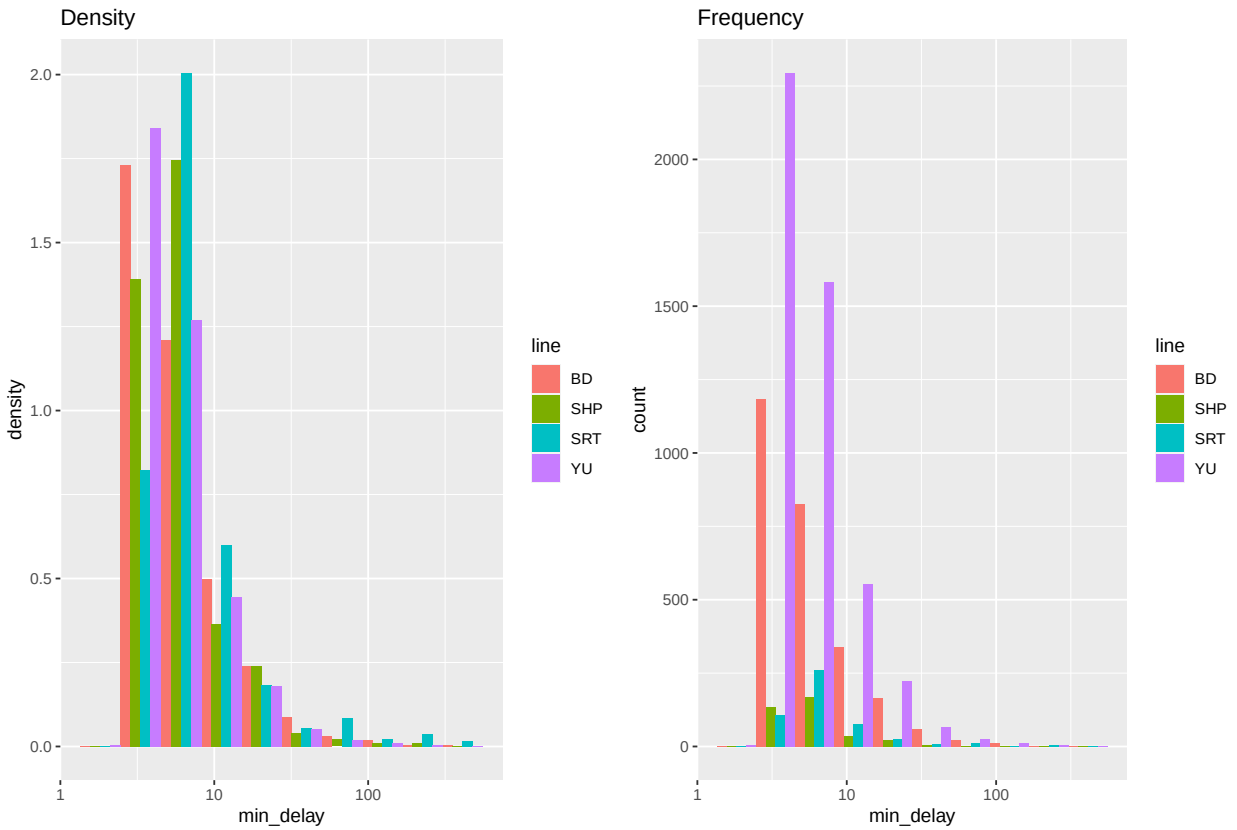
```
delay_2022 |>
           left_join(delay_codes |>
           rename(code = `SUB RMENU CODE`, code_desc = `CODE DESCRIPTION...3`) |>
           select(code, code_desc)) |>
           arrange(-min_delay) |>
           select(date, time, station, line, min_delay, code, code_desc)
```

```
## # A tibble: 17,819 x 7
##    date                time  station                line  min_de~1 code  code_~2
##    <dttm>              <chr> <chr>                  <chr>     <dbl> <chr> <chr>
##  1 2022-08-22 00:00:00 12:20 SRT LINE               SRT         451 PRSO  Signal~
##  2 2022-04-28 00:00:00 06:02 JANE STATION           BD          388 PUTR  Rail R~
##  3 2022-07-26 00:00:00 07:06 YONGE BD STATION       BD          382 MUPLB Fire/S~
##  4 2022-08-15 00:00:00 12:57 DUFFERIN STATION       BD          327 MUPR1 Priori~
##  5 2022-01-26 00:00:00 20:15 KENNEDY SRT STATION    SRT         315 MRWEA Weathe~
##  6 2022-08-02 00:00:00 21:23 HIGHWAY 407 STATION    YU          312 MUPR1 Priori~
##  7 2022-01-17 00:00:00 21:30 SHEPPARD WEST TO UNION YU          291 MUFM  Force ~
##  8 2022-01-25 00:00:00 21:03 SCARBOROUGH CTR STATIO SRT         285 PRSL  Loop R~
##  9 2022-06-17 00:00:00 12:25 KIPLING STATION        BD          241 SUUT  Unauth~
## 10 2022-02-09 00:00:00 06:06 DUPONT STATION         YU          240 SUAE  Assaul~
## # ... with 17,809 more rows, and abbreviated variable names 1: min_delay,
## #   2: code_desc
```

## c) Grouping and small multiples

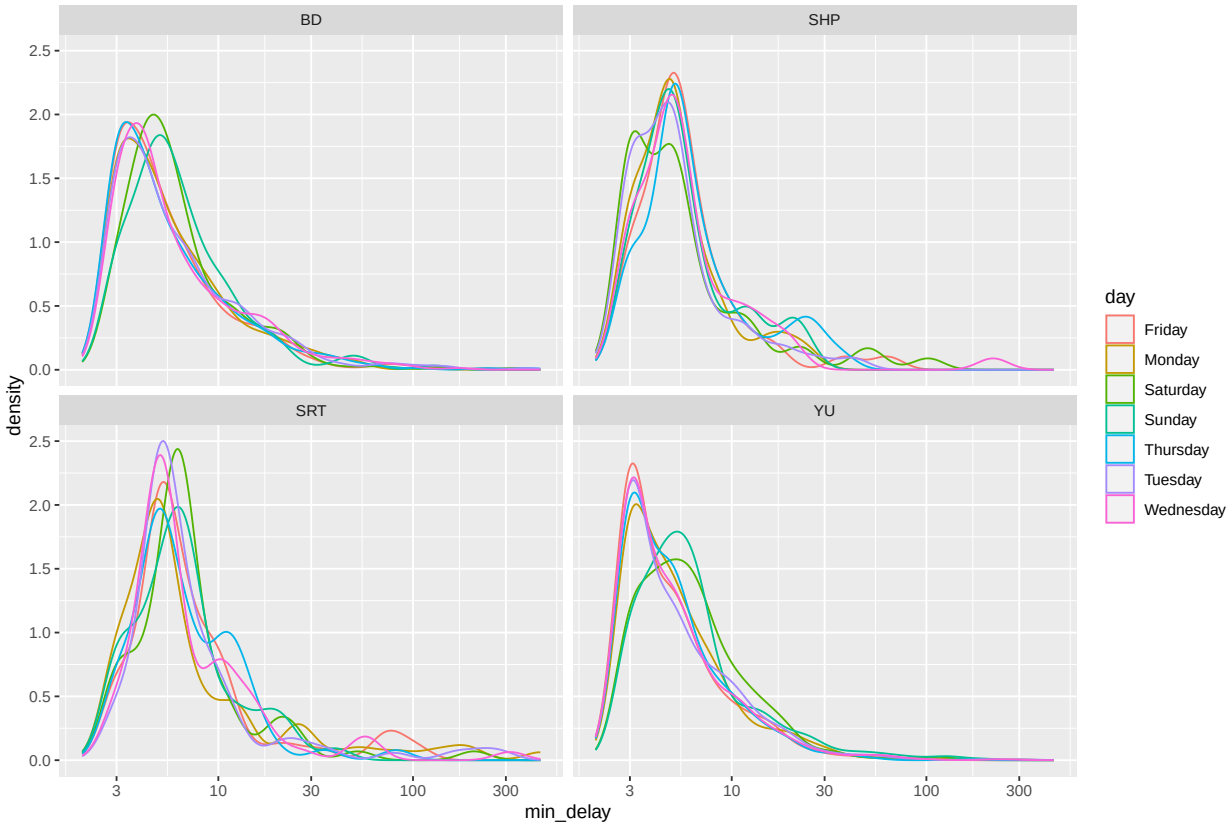
A quick and powerful visualization technique is to group the data by a variable of interest, e.g. `line`

```
ggp3 <- ggplot(data = delay_2022) +
        geom_histogram(aes(x = min_delay, y = ..density.., fill = line), position = 'dodge', bins = 10)
        labs(title = "Density") +
        scale_x_log10()
ggp4 <- ggplot(data = delay_2022) +
        geom_histogram(aes(x = min_delay, fill = line), position = 'dodge', bins = 10) +
        labs(title = "Frequency") +
        scale_x_log10()
require(gridExtra)
grid.arrange(ggp3, ggp4, ncol = 2)
```




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)
```




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(station, 1)))
head(delay_2022)
```

```
## # A tibble: 6 x 12
##   date                time  day      station   code  min_d~1 min_gap bound line
##   <dttm>              <chr> <chr>    <chr>     <chr>   <dbl>   <dbl> <chr> <chr>
## 1 2022-01-01 00:00:00 15:59 Saturday LAWRENCE~ SRDP        0       0 N     SRT
## 2 2022-01-01 00:00:00 02:23 Saturday SPADINA ~ MUIS        0       0 <NA>  BD
## 3 2022-01-01 00:00:00 22:00 Saturday KENNEDY ~ MRO         0       0 <NA>  SRT
## 4 2022-01-01 00:00:00 02:28 Saturday VAUGHAN ~ MUIS        0       0 <NA>  YU
## 5 2022-01-01 00:00:00 02:34 Saturday EGLINTON~ MUATC       0       0 S     YU
## 6 2022-01-01 00:00:00 05:40 Saturday QUEEN ST~ MUNCA       0       0 <NA>  YU
## # ... with 3 more variables: vehicle <dbl>, code_desc <chr>,
## #   station_clean <chr>, and abbreviated variable name 1: min_delay
```

## d) Visualizing time series

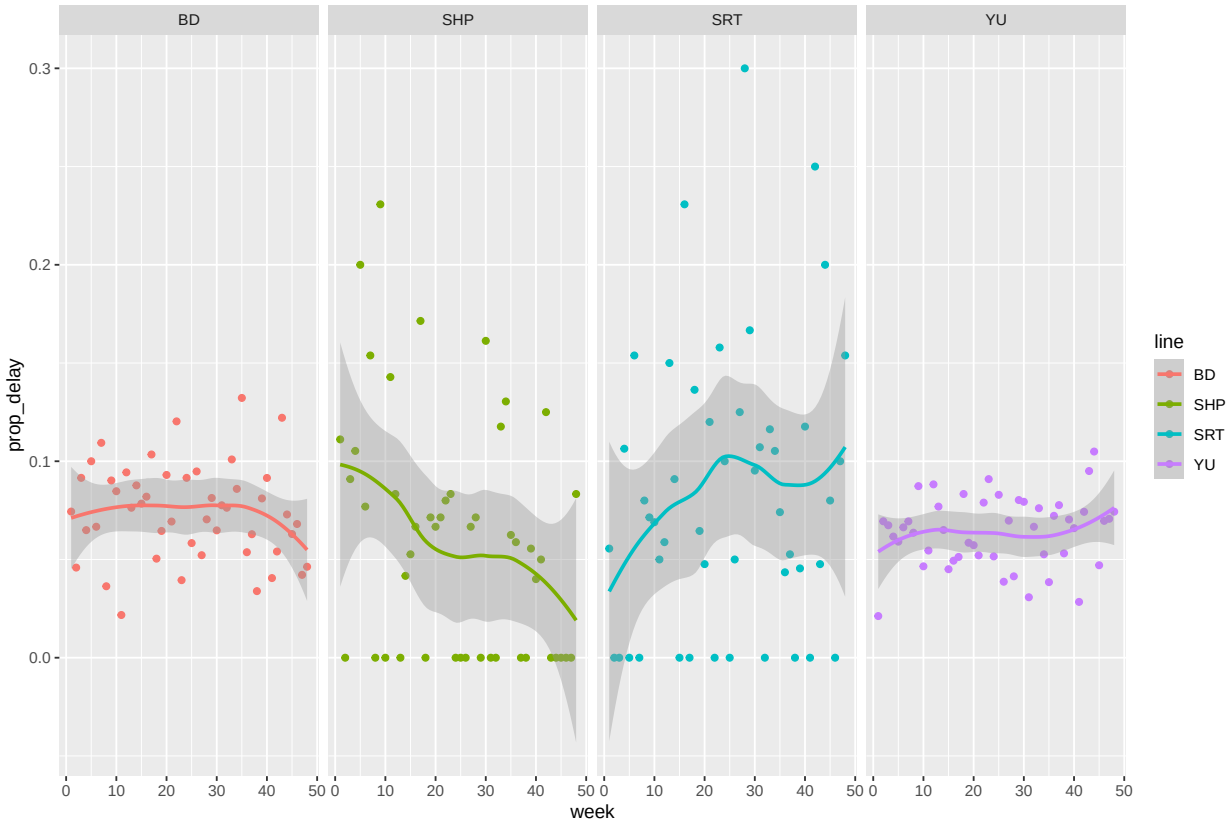
Daily plot is messy (you can check for yourself). Let's look by week to see if there's any seasonality. The `lubridate` package has lots of helpful functions that deal with date variables. First, mean delay (of those that were delayed more than 0 mins):

```
delay_2022 |> filter(min_delay>0) |>
              mutate(week = week(date)) |> # Get/set weeks component of a date-time
```

```
              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?

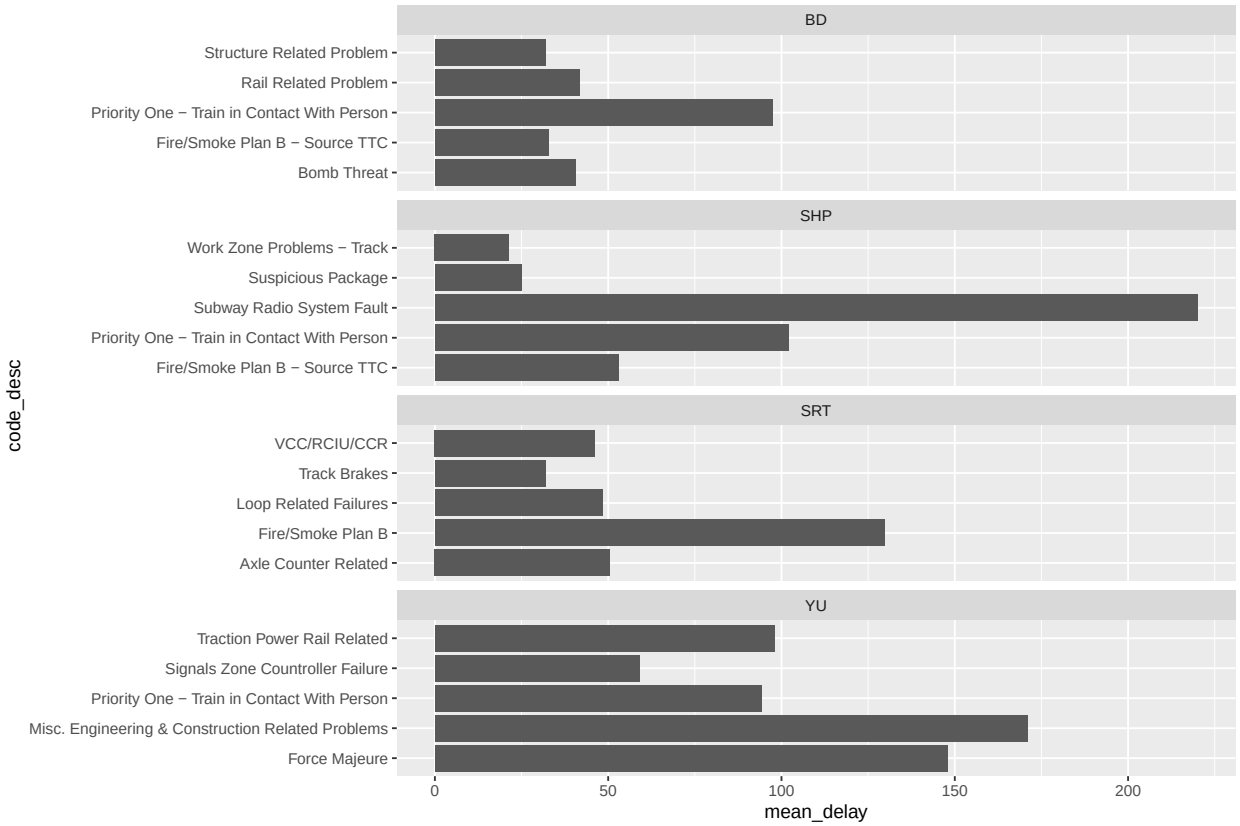
```
delay_2022 |> mutate(week = week(date)) |>
              group_by(week, line) |>
              summarise(prop_delay = sum(min_delay>10)/n()) |>
              ggplot(aes(week, prop_delay, color = line)) +
              geom_point() +
              geom_smooth() +
              facet_grid(~line)
```




## e) 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) |> # subset rows using their positions
              ggplot(aes(x = code_desc, y = mean_delay)) +
              geom_col() +
              facet_wrap(vars(line), scales = "free_y", nrow = 4) +
              coord_flip()
```




## f) 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:

```
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:

```
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 36
## # Groups:   line, station_clean [6]
##   line  station_c~1    PC1    PC2   PC3   PC4   PC5    PC6     PC7     PC8   PC9
##   <chr> <chr>        <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>   <dbl> <dbl>
## 1 BD    BATHURST     -4.63 -21.1  -9.64  9.11  3.58 -5.09   0.291  -10.5    6.68
## 2 BD    BAY          16.0   -9.30 -7.46  7.13  1.11  2.32  -5.94    -0.286  2.04
## 3 BD    BLOOR        36.4   27.2  33.2  16.8  -8.87 -8.52   0.0401   0.991  3.32
## 4 BD    BROADVIEW   -11.7  -23.3  -8.84 10.2  -2.35  3.60  -2.72     7.19  -5.05
## 5 BD    CASTLE       21.5   -5.02 -4.29  6.35  4.24  0.609 -1.29    -5.76   1.36
## 6 BD    CHESTER      26.8   -1.63 -2.97  5.79  3.88  2.78  -3.07    -1.29  -4.07
## # ... with 25 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>, and abbreviated variable name
## #   1: station_clean
```

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

```
ggp5 <- 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?

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

```
## # A tibble: 34 x 35
##         PC1      PC2      PC3      PC4     PC5     PC6      PC7      PC8     PC9
##       <dbl>    <dbl>    <dbl>    <dbl>   <dbl>   <dbl>    <dbl>    <dbl>   <dbl>
##  1 -0.127   -0.0459  -2.36e-2  0.0280  -0.0305  0.0853 -0.0811   0.0334  -0.182
##  2 -0.289   -0.146   -8.14e-2  0.0439  -0.101   0.243  -0.0534  -0.747    0.322
##  3 -0.0483  -0.0165   3.43e-2  0.0385  -0.0584 -0.0268  0.0710  -0.0804  -0.323
##  4 -0.0113  -0.0156  -1.25e-2 -0.00572 -0.0433  0.0421  0.00534 -0.0100   0.0686
##  5 -0.00986 -0.00341 -5.26e-4  0.00908  0.0191  0.0262  0.0409  -0.0626  -0.0706
##  6 -0.0860  -0.0388   5.06e-2 -0.0114   0.0295  0.0858 -0.0377   0.274   -0.188
##  7 -0.0128  -0.00567 -5.83e-5  0.00341 -0.0149  0.0376  0.00710  0.00733  0.0442
##  8 -0.678   -0.421   -7.40e-2  0.0849   0.119  -0.448   0.267    0.205    0.0927
##  9 -0.263    0.373    6.99e-1  0.310   -0.419  -0.0906  0.0264   0.0106   0.0750
## 10 -0.0427  -0.00255  1.11e-1 -0.0263   0.0915  0.457   0.228    0.261    0.148
## # ... with 24 more rows, and 26 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>, feature <chr>
```

```
ggp6 <- ggplot(df_out_r,aes(x=PC1,y=PC2,label=feature )) + geom_text_repel()
require(gridExtra)
```

```
grid.arrange(ggp5, ggp6, ncol = 2)
```