A Useful Cheatsheet For Data Visualization

Wangtao Zheng
UNI: wz2618
EDAV Community Contribution
Fall 2022

Overview

- The purpose of this cheatsheet is to provide a quick and convenient coding glossary/library for data scientists to implement exploratory data analysis and visualization in R
- By looking at this cheatsheet, one can easily and quickly find the most commonly-used and the most convenient codes for doing different types of commonly-applied data visualization
- This cheatsheet also include different variations of a single type of plot

Table of Contents

Code Placeholders

- For convenience reasons, I placed various placeholders in the code.
- All placeholders in lines of code are highlighted in red so that you can identify them and change them according to your own dataset and naming conventions
- You can also change some of the code placeholders, such as color, to the type of your own like

1, Histogram

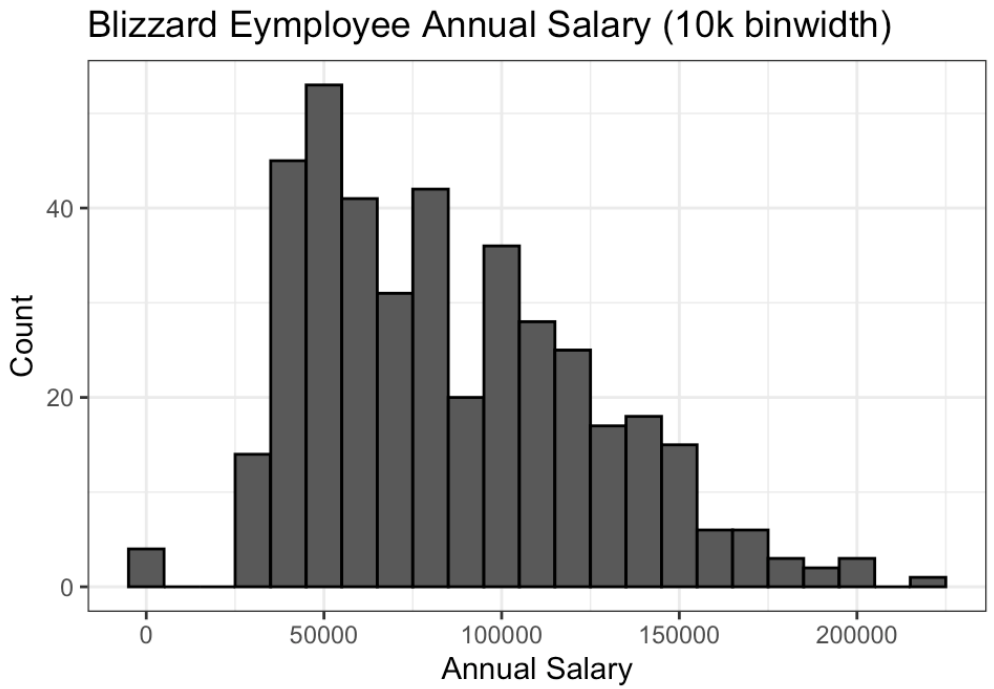
- We use ggplot2 package to create a classic histogram in R
- You can change the binwidth and the color of the histogram as you like
- **df %>% ggplot(aes(x = feature)) +**

  geom_histogram(binwidth=10000, color='black')+

  ggtitle('Title')+

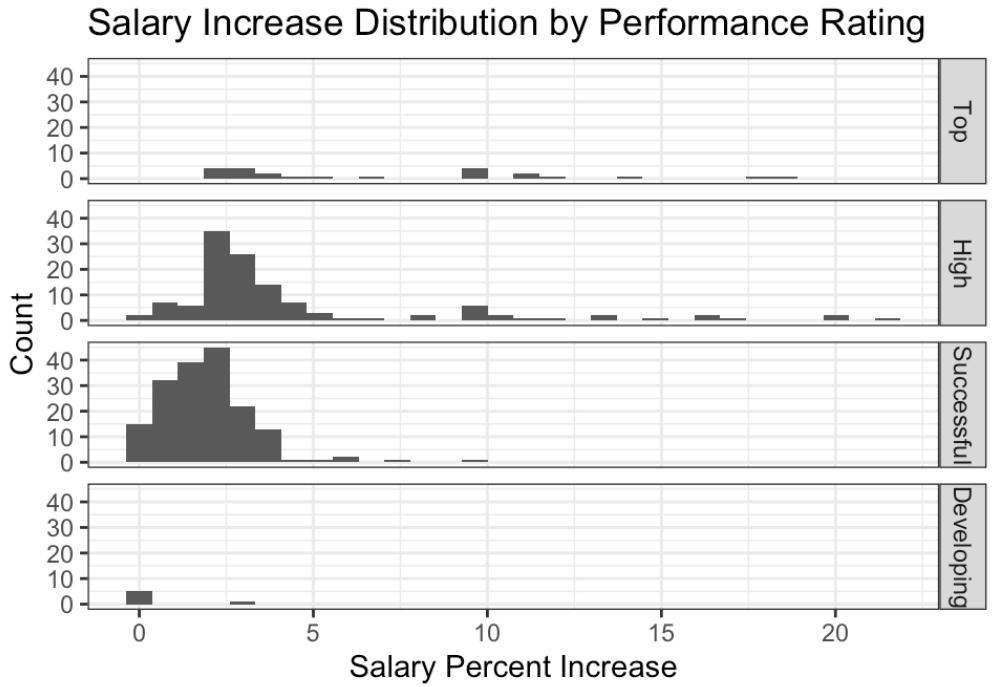
  labs(x='label', y='Count')+

  theme_bw()



1, Histogram

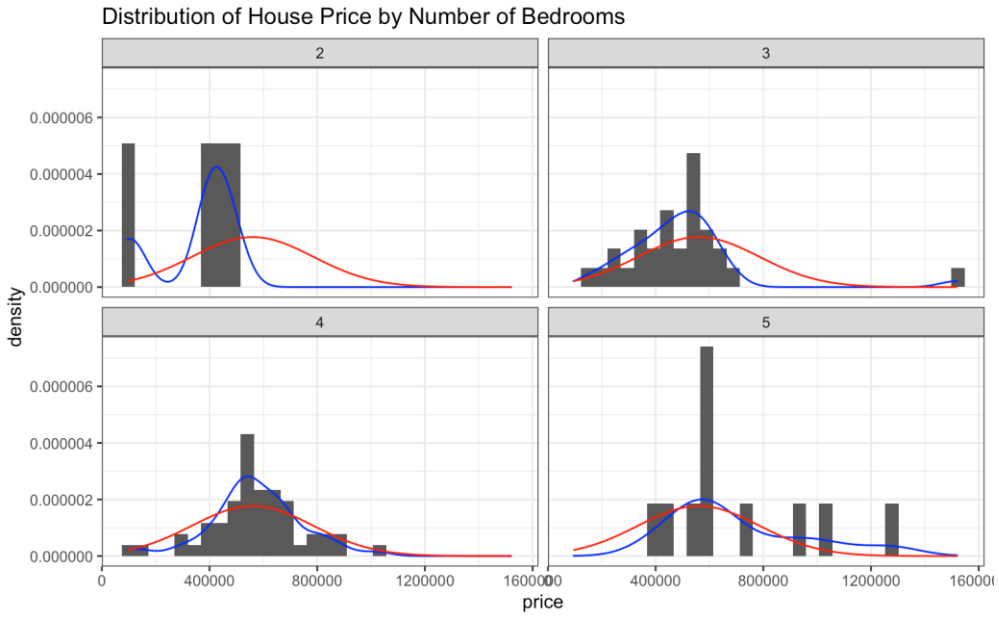
- If you would like to do a faceted histogram, simply include “facet_grid” in your code and name the categorical feature you would like to base on
- **ggplot(df, aes(x = feature)) +**
  geom_histogram() +
  facet_grid(rows = feature)))+
  ggtitle(‘Title’)+
  labs(x=‘label’, y=‘Count’)+
  theme_bw()



1, Histogram

- If you would like to combine histogram with density curves, use the following code:
- df %>%

```
  filter(feature != a)  %>%
  ggplot(aes(x = feature)) +
  geom_histogram(aes(y = stat(density))) +
  geom_density(color = 'blue') +
  stat_function(fun = dnorm, args =
          list(mean = mean(df$feature), sd =
sd(df$feature)), col = 'red') +
  facet_wrap(~feature)+
  ggtitle('Title')+
  theme_bw()
```



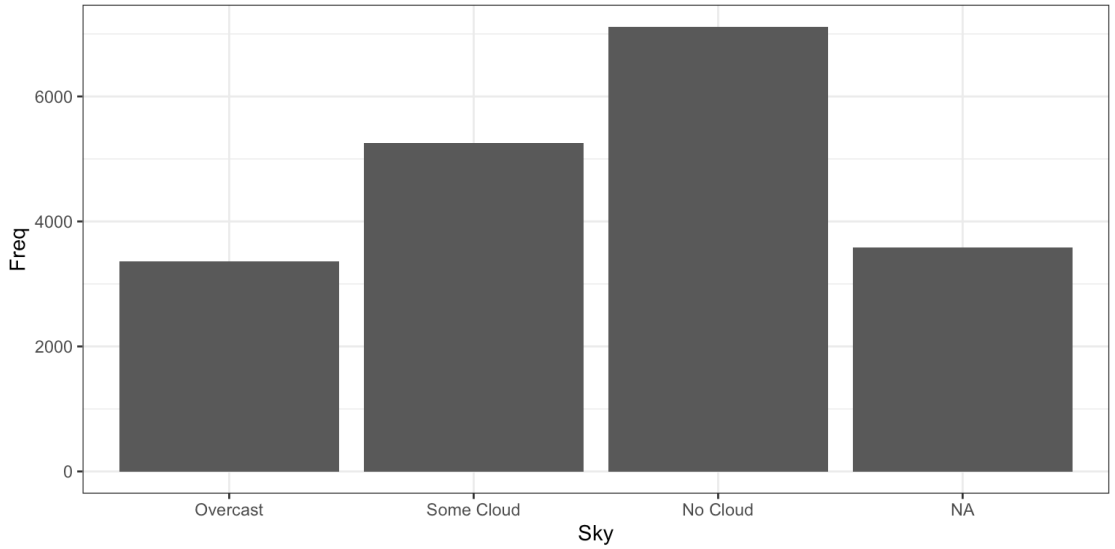
2, Bar Chart

- To draw a vanilla bar chart, you write the following code:
- **ggplot(df, aes(x=feature))+**

  geom_bar()+

  labs(x='label', y='Freq')+

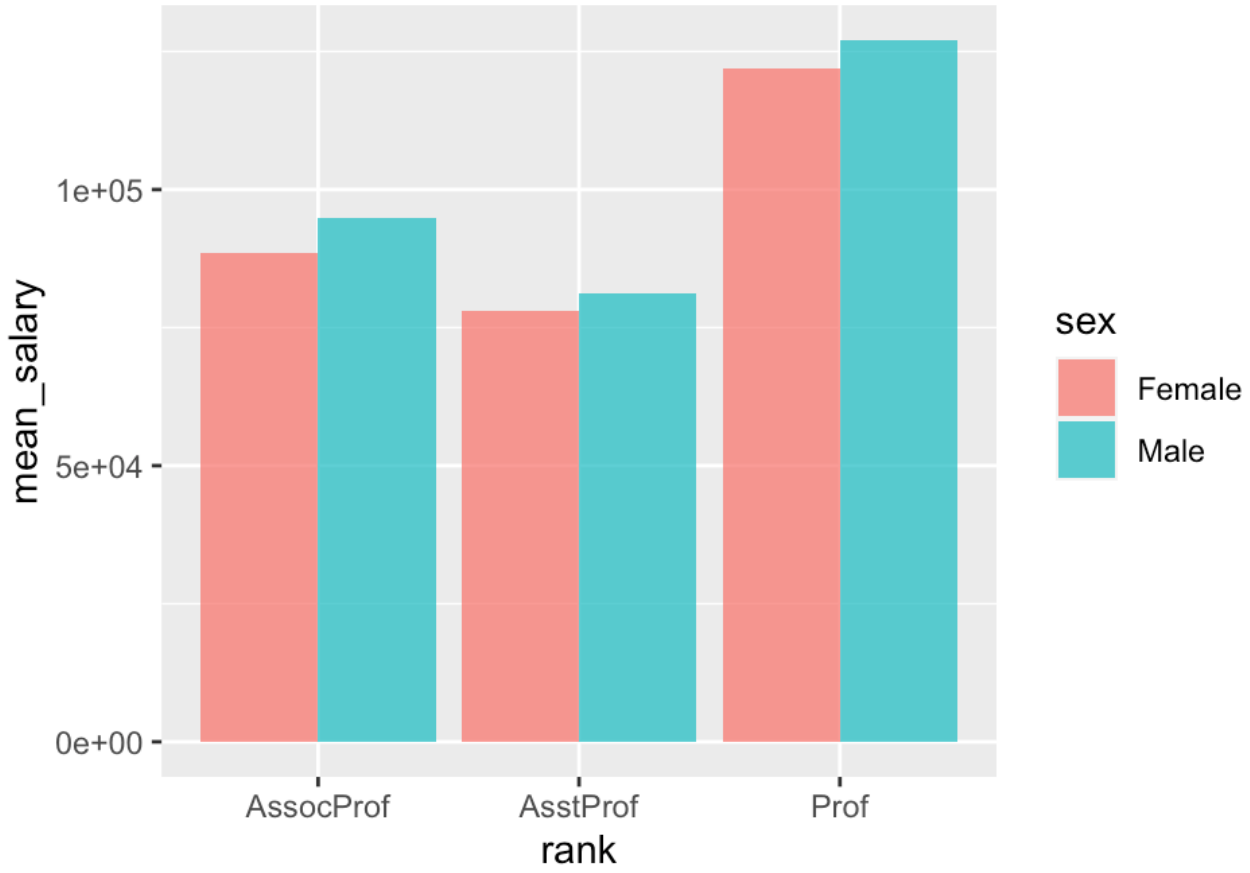
  theme_bw()



2, Bar Chart

- To draw a side-by-side bar chart, we introduce “fill” and use position_dodge():
- **ggplot(data = df, aes(x = feature, y = feature, fill = feature)) +**

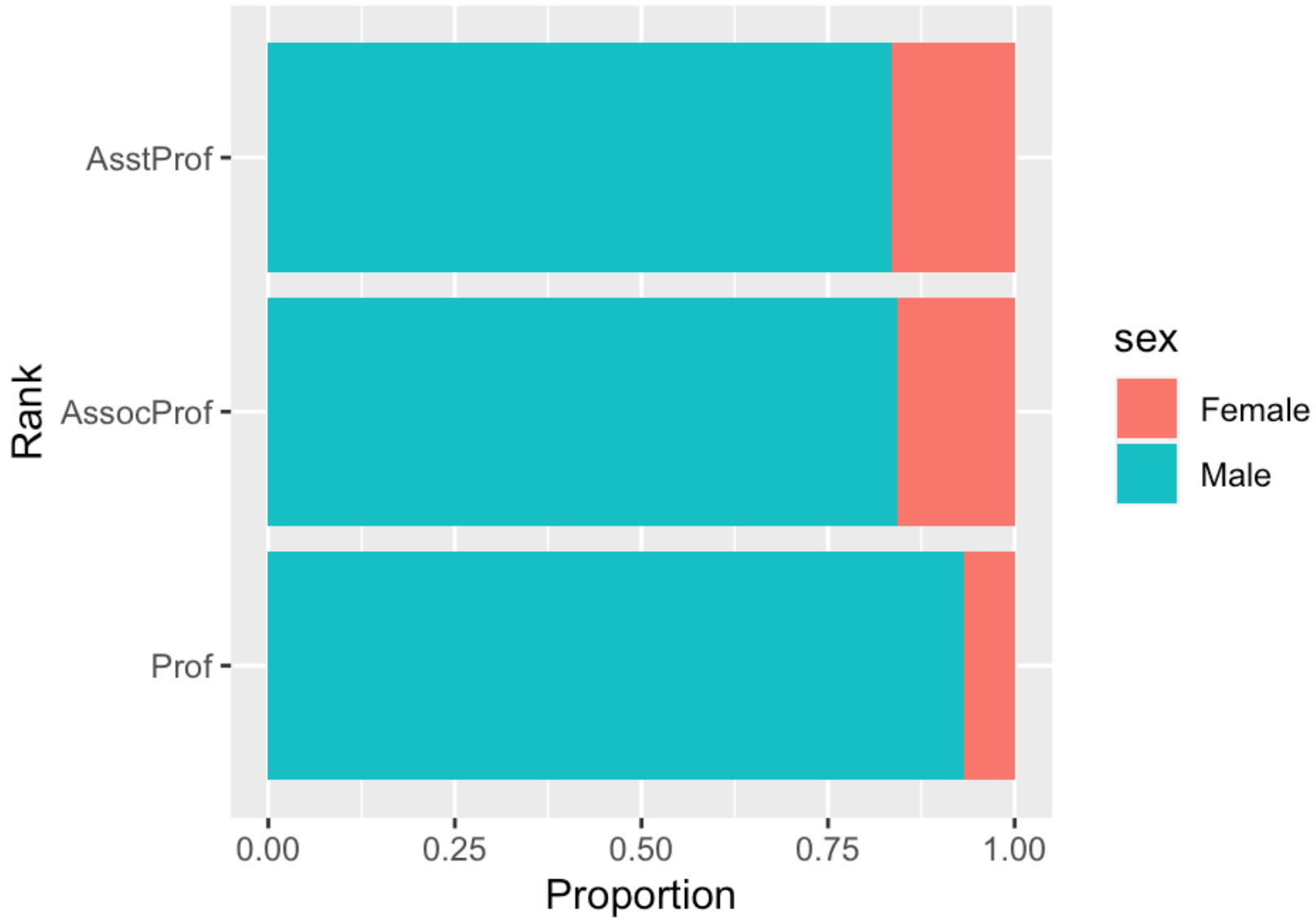
geom_bar(stat = "identity", position = position_dodge(), alpha = 0.75)



2, Bar Chart

- To draw a horizontal bar chart, we use coord_flip()
- If you would like to order the bar chart by proportion, you apply fct_reorder2
- Based on this, if you want to draw a horizontal percent stacked bar chart, you write the following code:
- **ggplot(df, aes(fill= feature,**

```
                x=fct_reorder2(feature,
                               feature==‘category’,
                               prop,.desc = FALSE),
                y=feature proportion)) +
geom_bar(position="fill", stat="identity") +
coord_flip() +
labs(x=‘label', y = 'Proportion')
```


Rank
AsstProf
AssocProf
Prof
0.00
0.25
0.50
0.75
1.00
Proportion
sex
Female
Male

3, Scatterplot

- To draw a classic scatterplot that shows a x-y relationship, you do the coding as follows:
- **ggplot(df, aes(x=x_feature, y=y_feature)) +**

  geom_point() +

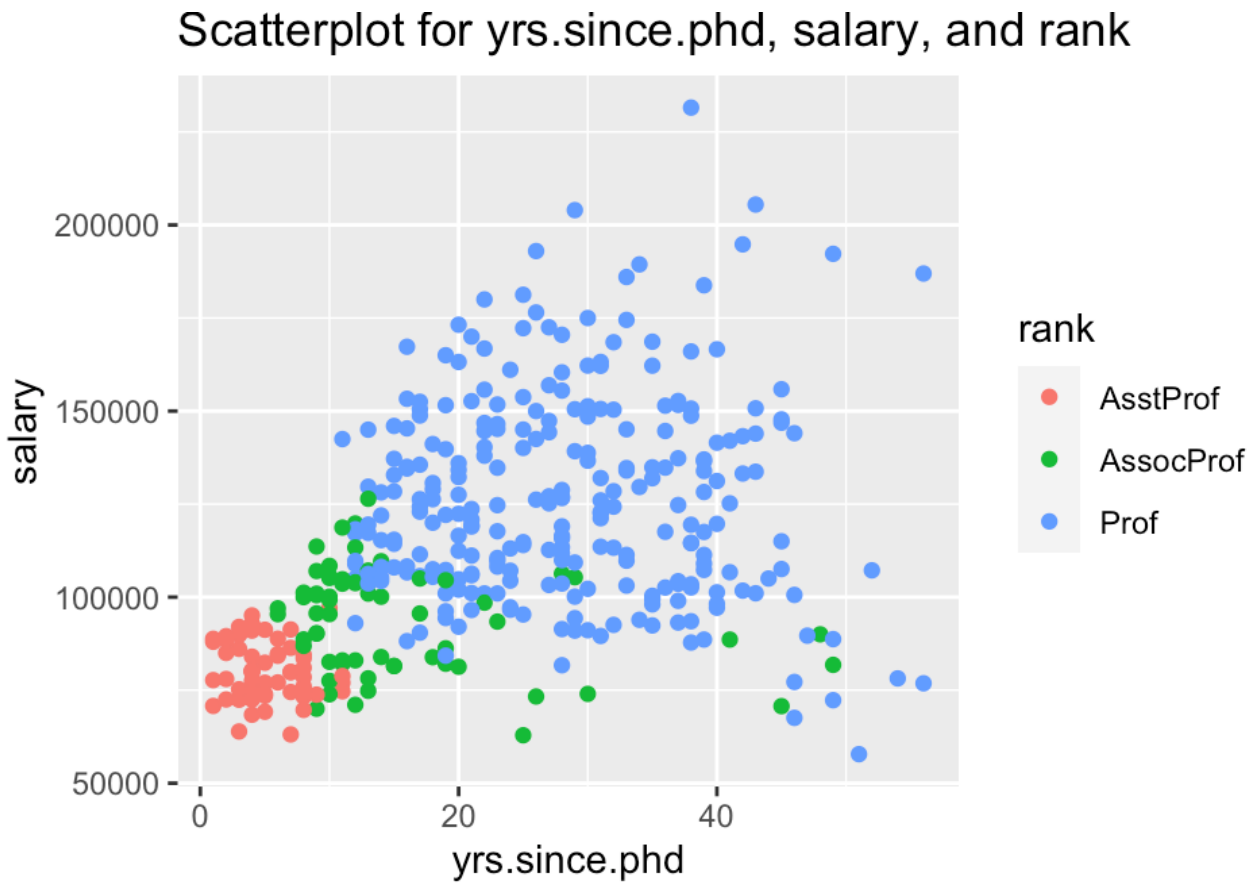
  ggtitle('Title')

3, Scatterplot

- If you want to draw a categorized scatterplot that assign different colors to different categorical entries, simply use “fill”:
- **ggplot(df, aes(x=x_feature, y=y_feature,color=categorical_feature)) +**

  geom_point() +

  ggtitle(‘Title’)



3, Scatterplot

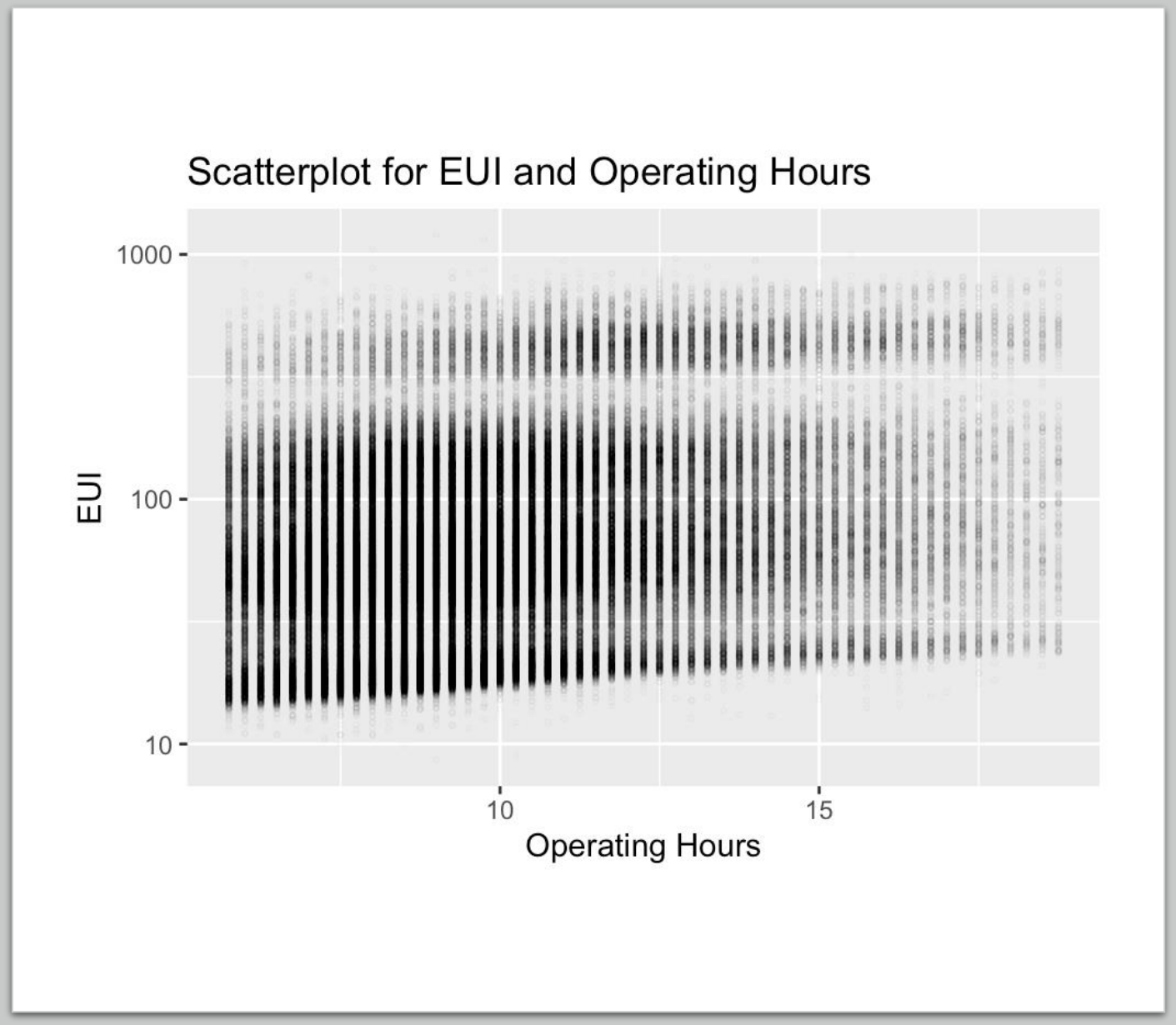
- If you want to draw a log-scaled scatterplot to visualize data trends, write the following:
- **ggplot(df, aes(x=x_feature,y=y_feature)) +**

  geom_point(alpha = 0.03, pch=21, size=0.5) +

  scale_y_continuous(trans='log10') +

  labs(x='x_label', y='y_label') +

  ggtitle('Title')



4, Q-Q plot

- If you would like to draw a Q-Q plot with lines of best fit, use stat_qq and geom_qq_line:
- **ggplot(aes(sample=feature))+**

  stat_qq()+

  geom_qq_line(color ='red')+

  ggtitle('Title')+

  theme_bw()

4, Q-Q plot

- Similar to previous graphs listed, you can add facet_wrap to create a multi-dimensional Q-Q plot:
- **ggplot(aes(sample=feature))+**

  stat_qq()+

  geom_qq_line(color ='red')+

  facet_wrap(~feature)+

  ggtitle('Title')+

  theme_bw()

5, Boxplot

- To draw paralleled boxplots, use the code:
- **df %>% ggplot(aes(x = x_feature, y = y_feature)) +**

  geom_boxplot() +

  theme_bw()
- Here, x_feature is the categorical feature that differentiate the boxplots, and y_feature is the numerical feature on which the boxplot is based

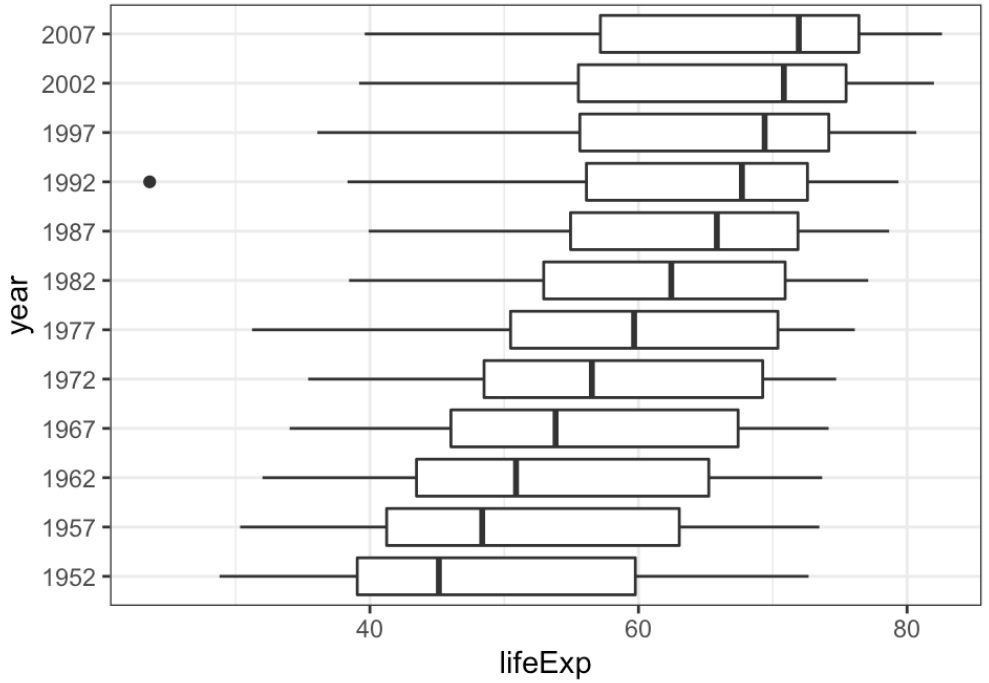
5, Boxplot

- If you want to draw horizontal paralleled boxplots, add coord_flip():
- **df %>% ggplot(aes(x = x_feature, y = y_feature)) +**

  geom_boxplot() +

  coord_flip() +

  theme_bw()

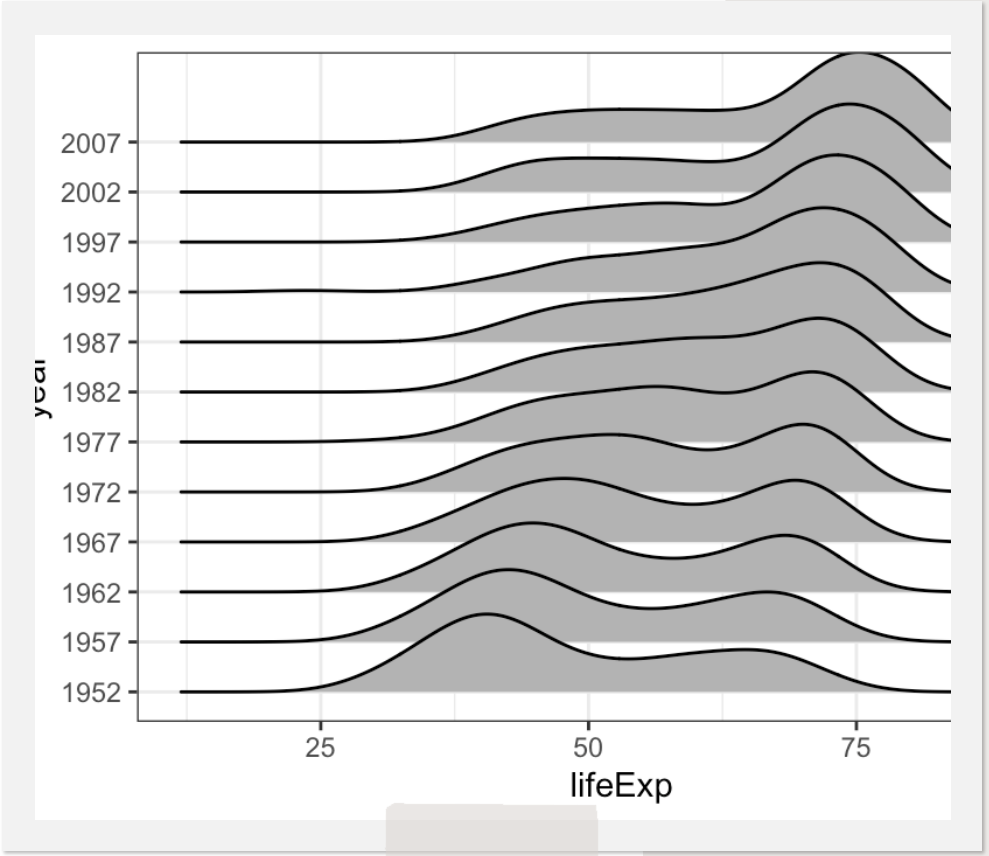


6, Ridgeline plot

- The code that is used to draw ridgeline plot is very similar to that of paralleled boxplots. You simply change geom_boxplot() to geom_density_ridges():
- **df %>% ggplot(aes(x = x_feature, y = y_feature)) +**

  geom_density_ridges() +

  theme_bw()



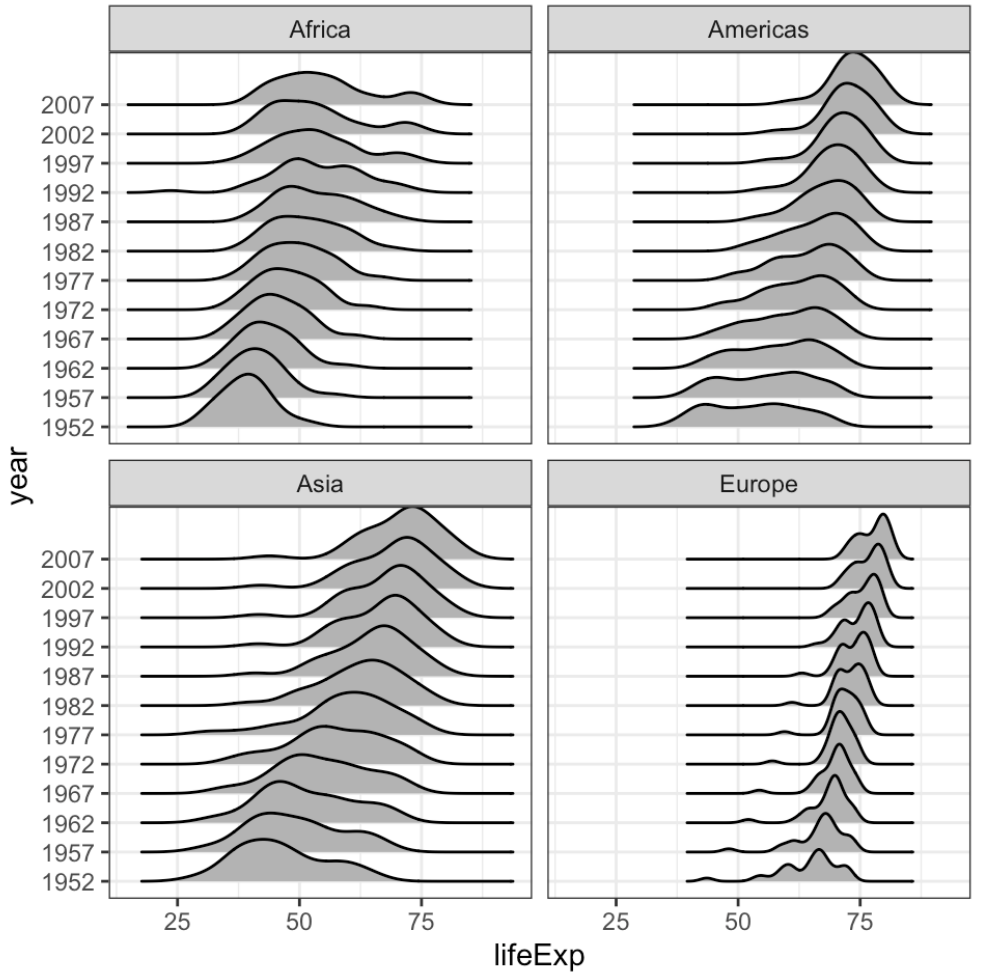
6, Ridgeline plot

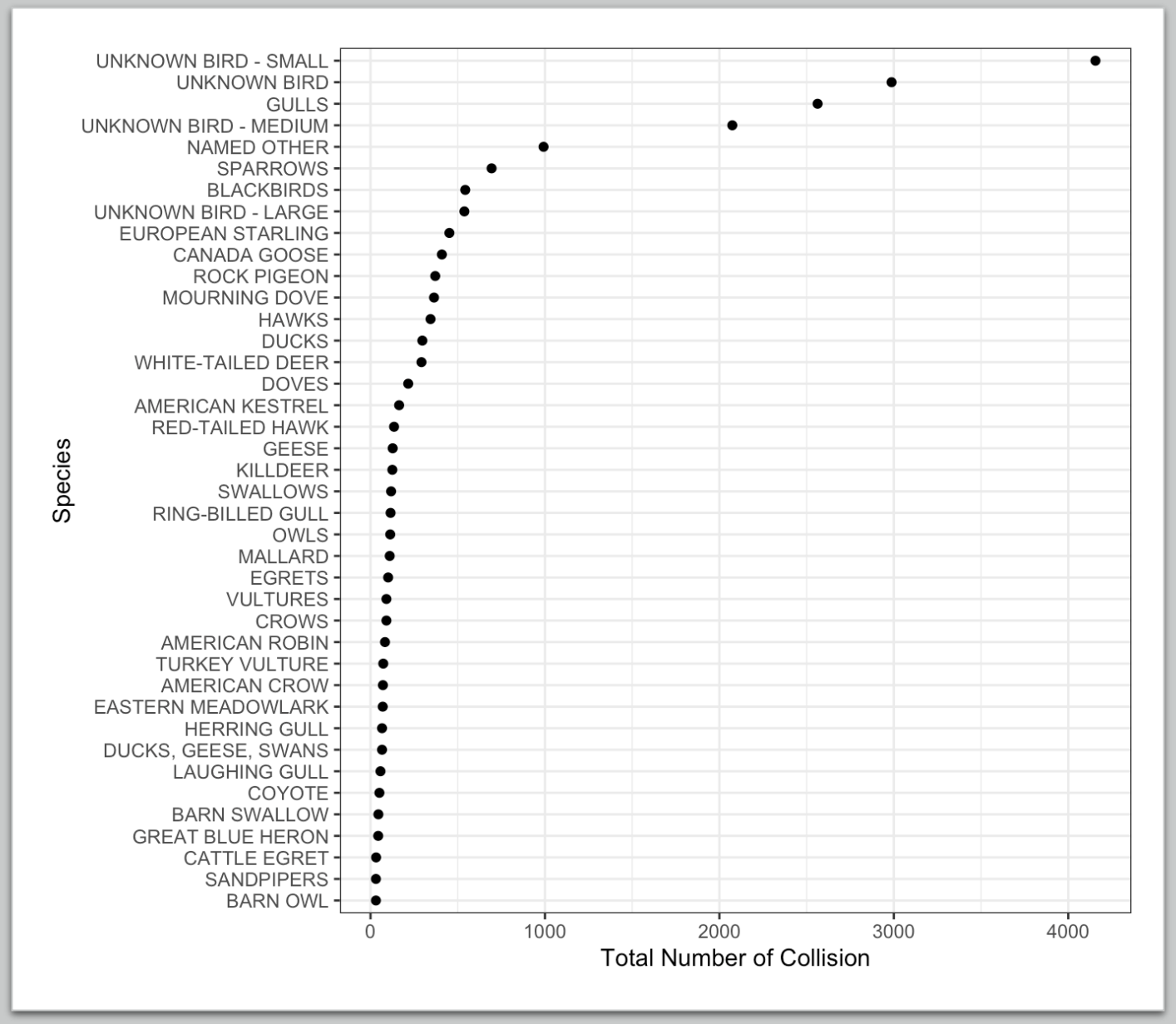
- Similar to previously-shown graphs, we can draw multi-facet ridgeline plot using facet_wrap():
- **df %>% ggplot(aes(x= x_feature, y=y_feature, group = feature)) +**

  geom_density_ridges() +

  facet_wrap(~feature) +

  theme_bw()





7, Cleveland dot plot

- To draw a Cleveland dot plot, you use the following code:
- **ggplot(df, aes(y = reorder(y_feature, x_feature), x = x_feature)) +**

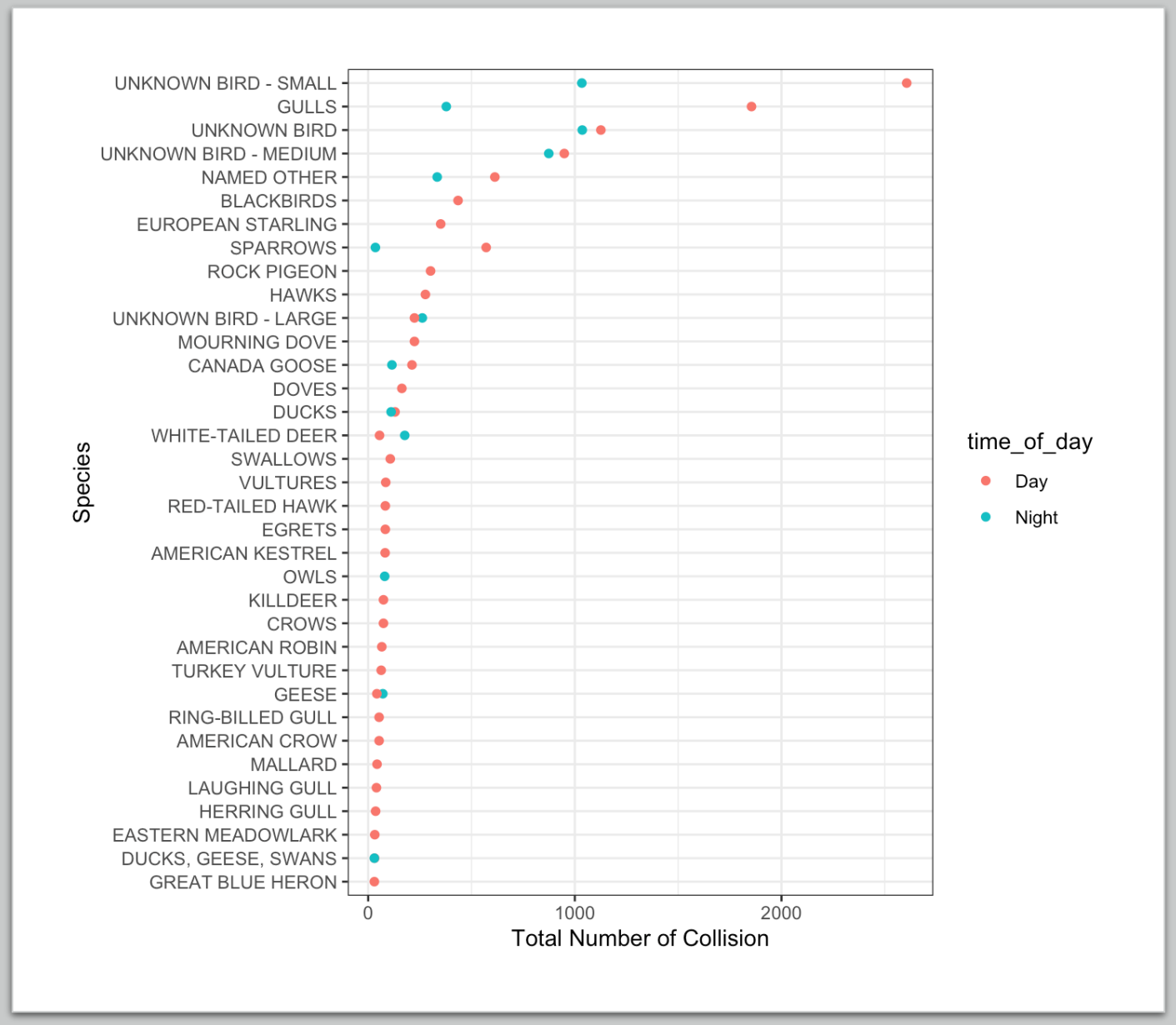
  geom_point()+

  labs(x='x_label', y='y_label')+

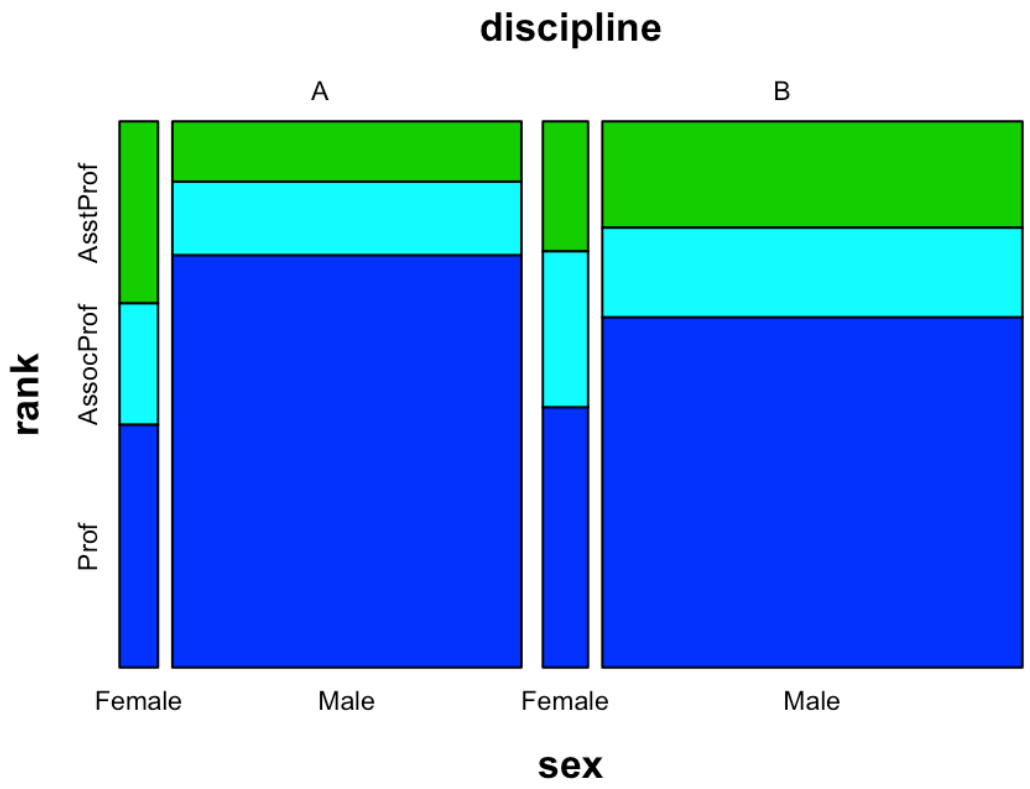
  theme_bw()
- Here y_feature are the divided categories, and x_feature is the numerical feature that we want to measure using Cleveland dot plot

7, Cleveland dot plot

- If you would like to draw a bicolor Cleveland dot plot that visualize the data in two parts, simply add aes(colour = feature) inside geom_point()
- Here, feature is the feature by which you would like to divide the Cleveland dot plot



8, Mosaic plot



- To visualize the inter-relationship between at least three parameters, we draw a mosaic plot:
- **mosaic(feature_C~ feature_A+feature_B, data = df,**

  direction = c("v", "v", "h"),

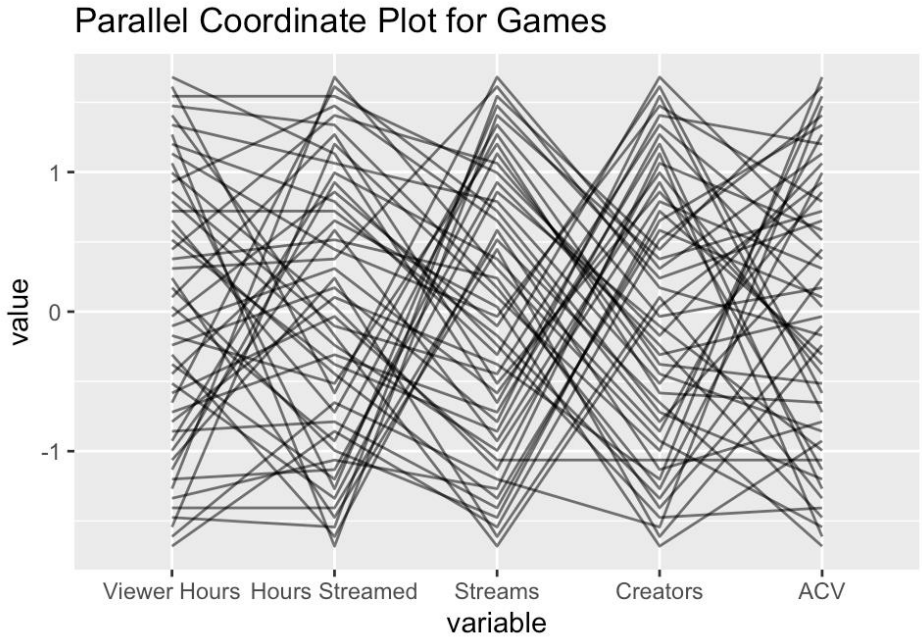
  highlighting_fill =c(3,5,4),

  gp_labels = gpar(fontsize = 8))
- Remember to use R packages “grid” and “vcd”

9, Parallel coordinate plot

- We use parallel coordinate plot to visualize relationship between multiple columns in a dataset:
- **ggparcoord(df, columns=c(3,4,7,6,5), alpha = 0.6,**

  title = "Title")
- Remember to use R packages "rvest" and "GGally"



Thanks and Appreciation

- The code and graphs included in this cheatsheet are cited from the first two problem sets of the EDAV class Fall 2022. I must hereby give credit to two of my brilliant teammates, Soheil Fakhrieh Kashan and Yuxin Lin (my teammates for PSet 1 and 2, respectively) for their contribution.
- All materials included in this cheatsheet are intellectual works authorized for use and dissemination within and ONLY within the EDAV class community.