Categorical x Continuous data (2 of 2)

Sep 23, 2023

1. This chapter explores **Categorical** x **Continuous** data using the `dplyr` and `ggplot2` packages.
2. **Data**: Suppose we run the following code to prepare the `mtcars` data for subsequent analysis and save it in a tibble called `tb`.

```
# Load the required libraries, suppressing annoying startup messages
library(dplyr, quietly = TRUE, warn.conflicts = FALSE)
library(tibble, quietly = TRUE, warn.conflicts = FALSE)
library(knitr) # For formatting tables
# Read the mtcars dataset into a tibble called tb
data(mtcars)
tb <- as_tibble(mtcars)
# Convert relevant columns into factor variables
tb$cyl <- as.factor(tb$cyl) # cyl = {4,6,8}, number of cylinders
tb$am <- as.factor(tb$am) # am = {0,1}, 0:automatic, 1: manual transmission
tb$vs <- as.factor(tb$vs) # vs = {0,1}, v-shaped engine, 0:no, 1:yes
tb$gear <- as.factor(tb$gear) # gear = {3,4,5}, number of gears
# Directly access the data columns of tb, without tb$mpg
attach(tb)
```

Visualizing Continuous Data using `ggplot2`

Let's take a closer look at some of the most effective ways of visualizing continuous data, across one Category, **using ggplot2**, including

(i) Bee Swarm plots, using ggplot2;

(ii) Histograms, using ggplot2;

(iii) PDF and CDF Density plots, using ggplot2;

(iv) Box plots, using ggplot2;

(v) Violin plots, using ggplot2;

Bee Swarm Plot across one Category using `ggbeeswarm`

- Visualizing Median using Box Plot – median weight of the cars broken down by cylinders (`cyl`=4,6,8)

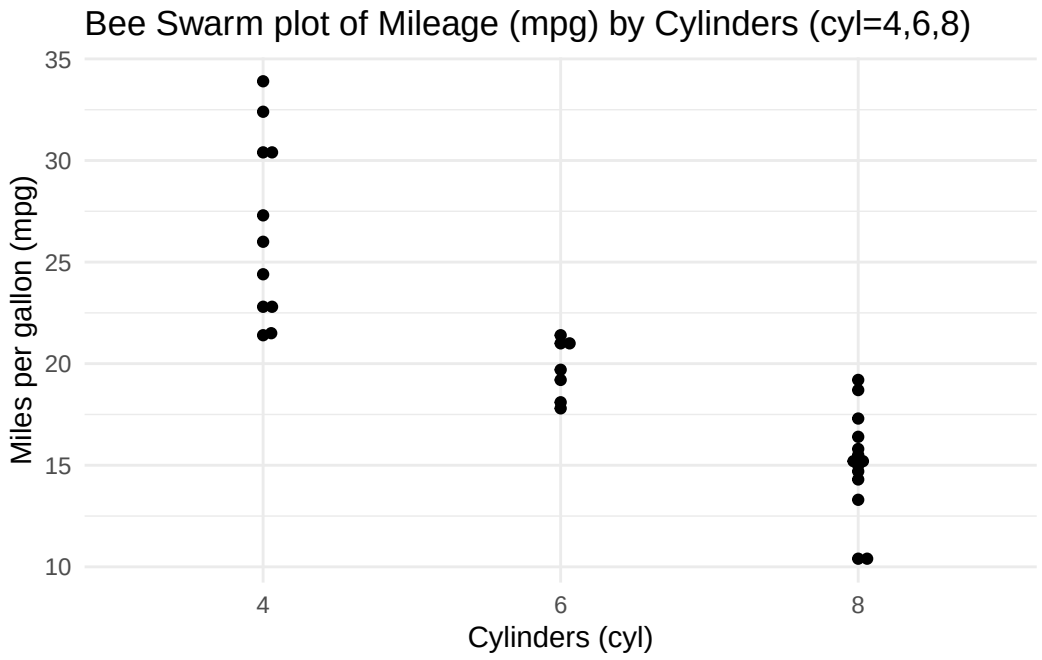
```
library(ggplot2)
```

```
Attaching package: 'ggplot2'
```

```
The following object is masked from 'tb':

    mpg
```

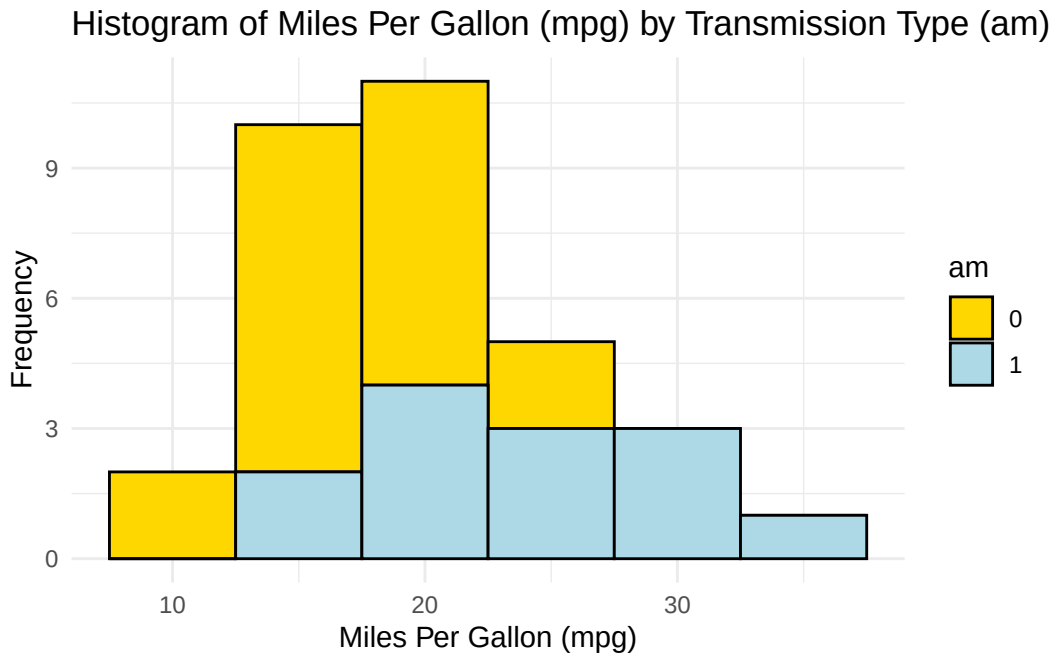
```
library(ggbeeswarm)
# Create the beeswarm plot
ggplot(mtcars,
       aes(x = factor(cyl),
           y = mpg)) +
  geom_beeswarm() +
  labs(title = "Bee Swarm plot of Mileage (mpg) by Cylinders (cyl=4,6,8)",
       x = "Cylinders (cyl)",
       y = "Miles per gallon (mpg)") +
  theme_minimal()
```



Histograms across one Category using `ggplot2`

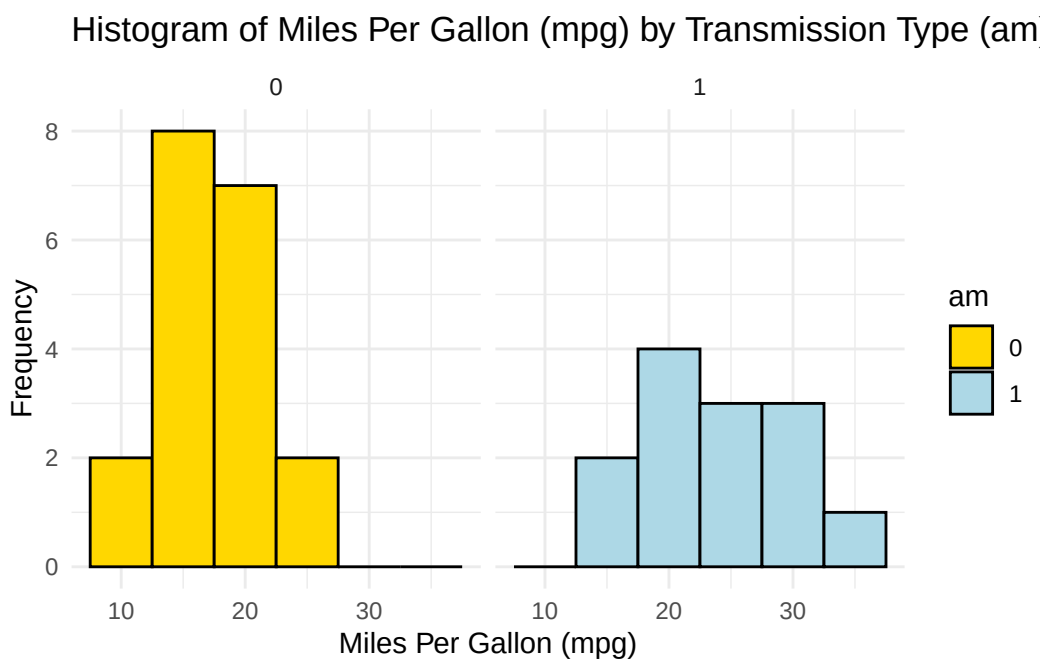
- Visualizing histograms of car milegage (`mpg`) broken down by transmission (`am`=0,1)

```
ggplot(tb, aes(x = mpg,
               fill = am)) +
  geom_histogram(binwidth = 5, color = "black") +
  scale_fill_manual(values = c("gold", "lightblue")) +
  theme_minimal() +
  labs(title = "Histogram of Miles Per Gallon (mpg) by Transmission Type (am)",
       x = "Miles Per Gallon (mpg)",
       y = "Frequency")
```



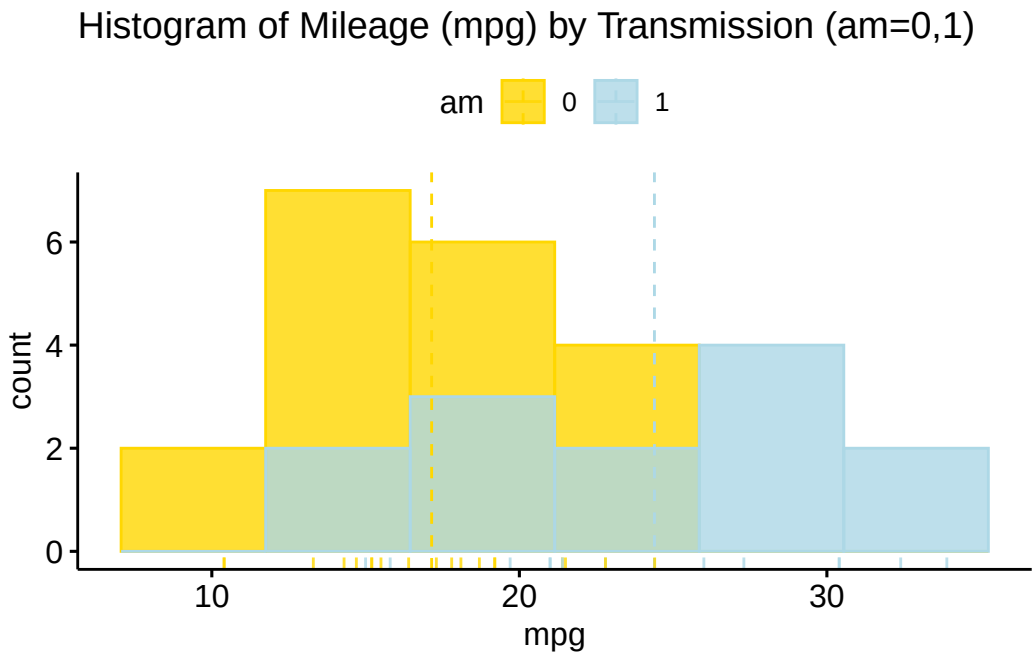
- **Discussion**: If we want separate histograms, we can set facet_wrap(~ am)

```
ggplot(tb, aes(x = mpg,
               fill = am)) +
  geom_histogram(binwidth = 5, color = "black") +
  scale_fill_manual(values = c("gold", "lightblue")) +
  facet_wrap(~ am) +
  theme_minimal() +
  labs(title = "Histogram of Miles Per Gallon (mpg) by Transmission Type (am)",
       x = "Miles Per Gallon (mpg)",
       y = "Frequency")
```



Histogram across one Category using `ggpubr`

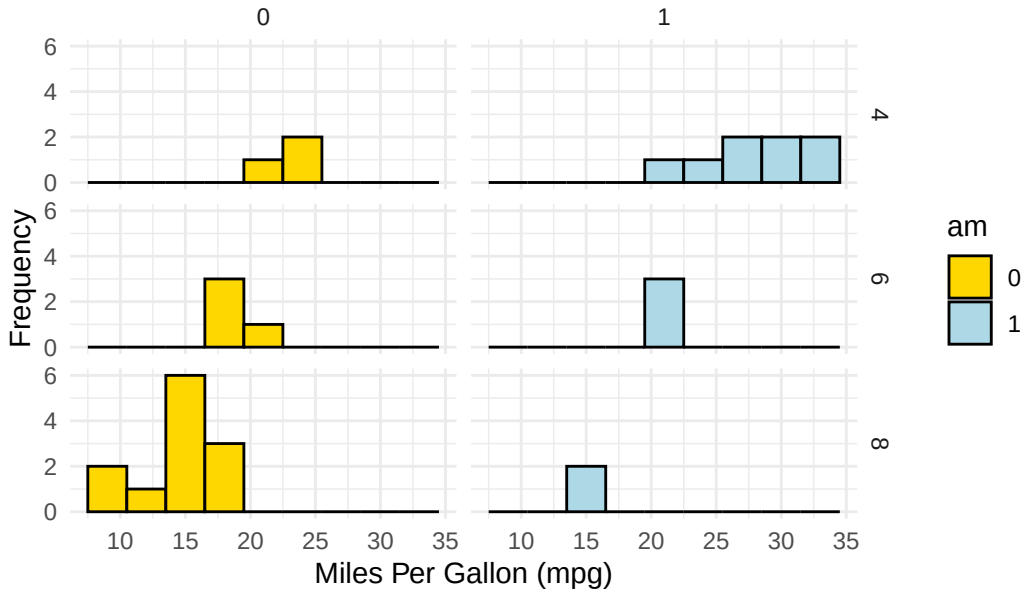
```
library(ggpubr)
gghistogram(tb,
            x = "mpg",
            bins = 6,
            add = "mean",
            rug = TRUE,
            color = "am",
            fill = "am",
            alpha = 0.8,
            palette = c("gold", "lightblue"),
            title = "Histogram of Mileage (mpg) by Transmission (am=0,1)")
```



Histograms across two Categories using `ggplot2`

- Visualizing histograms of car milegage (`mpg`) by transmission (`am`=0,1) and cylinders (`cyl`=4,6,8)

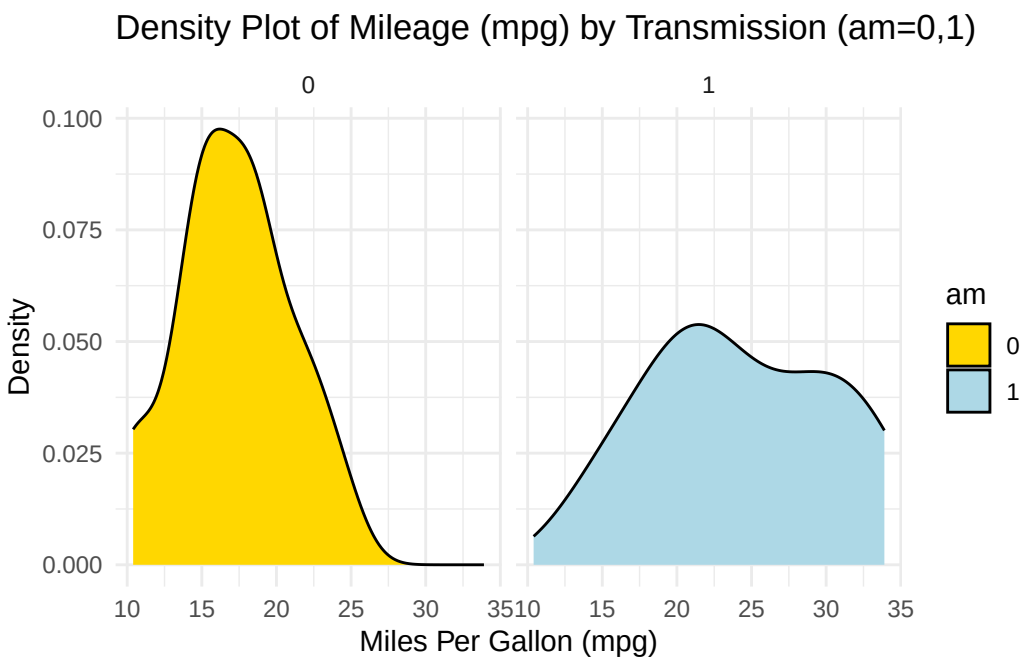
```
ggplot(tb, aes(x = mpg,
               fill = am)) +
  geom_histogram(binwidth = 3, color = "black") +
  scale_fill_manual(values = c("gold", "lightblue")) +
  facet_grid(cyl ~ am) +
  theme_minimal() +
  labs(title = "Histogram of Mileage (mpg) by Transmission (am=0,1) and Cylinders (cyl=4,6
       x = "Miles Per Gallon (mpg)",
       y = "Frequency")
```



PDF across one Category using `ggplot2`

- Visualizing the Probability Density Functions (PDF) of car milegage (`mpg`) by transmission (`am`=0,1)

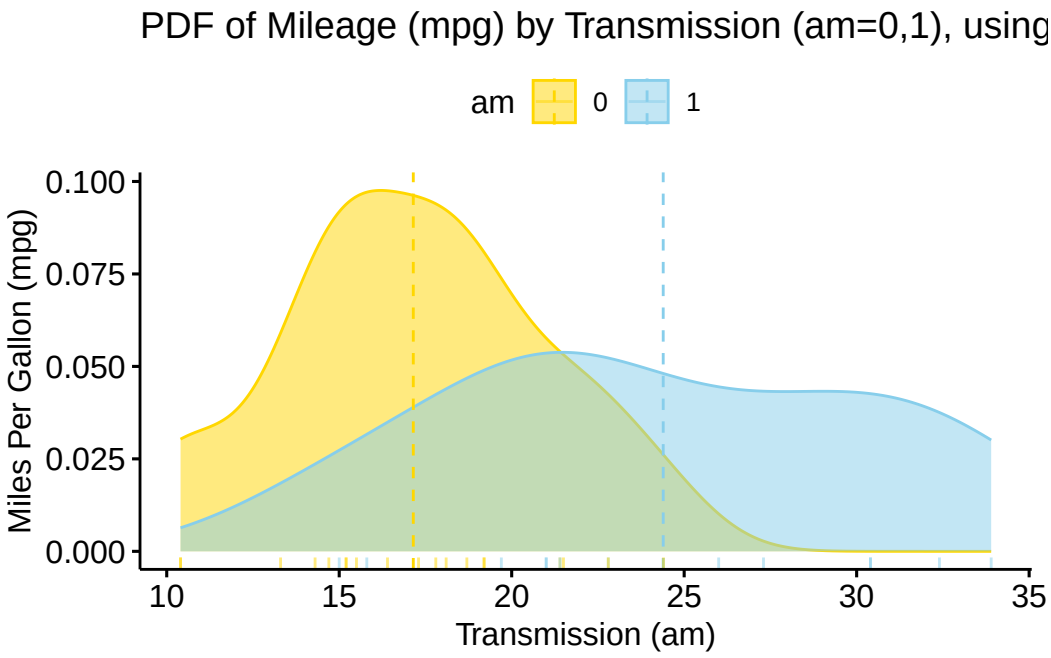
```
ggplot(tb, aes(x = mpg,
               fill = am)) +
  geom_density(color = "black") +
  scale_fill_manual(values = c("gold", "lightblue")) +
  facet_wrap(~ am) +
  theme_minimal() +
  labs(title = "Density Plot of Mileage (mpg) by Transmission (am=0,1)",
       x = "Miles Per Gallon (mpg)",
       y = "Density")
```



PDF across one Category using `ggpubr`

- The provided R code creates a Boxplot of the `mpg` (miles per gallon) variable in the `tb` dataset, using the `ggboxplot()` function from the `ggpubr` package.

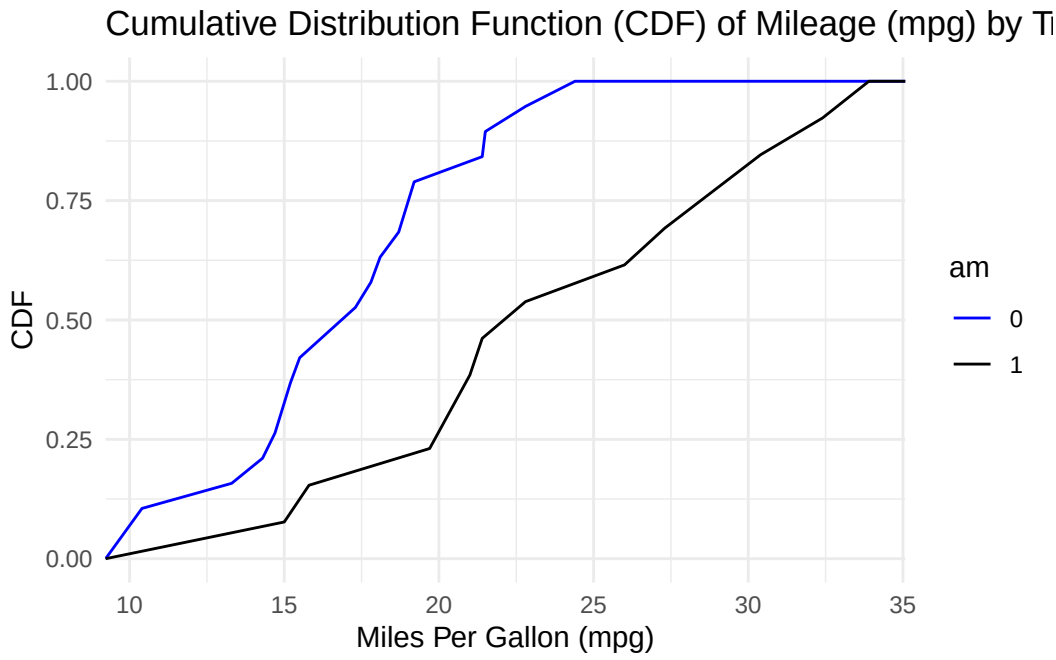
```
library(ggpubr)
ggdensity(tb,
          x = "mpg",
          color = "am" ,
          fill = "am",
          add = "mean",
          rug = TRUE,
          palette = c("gold", "skyblue"),
          title = "PDF of Mileage (mpg) by Transmission (am=0,1), using ggpubr::ggdensity(
          ylab = "Miles Per Gallon (mpg)",
          xlab = "Transmission (am)"
)
```



CDF across one Category using `ggplot2`

- Visualizing the Cumulative Density Functions (CDF) of car milegage (`mpg`) by transmission (`am`=0,1)

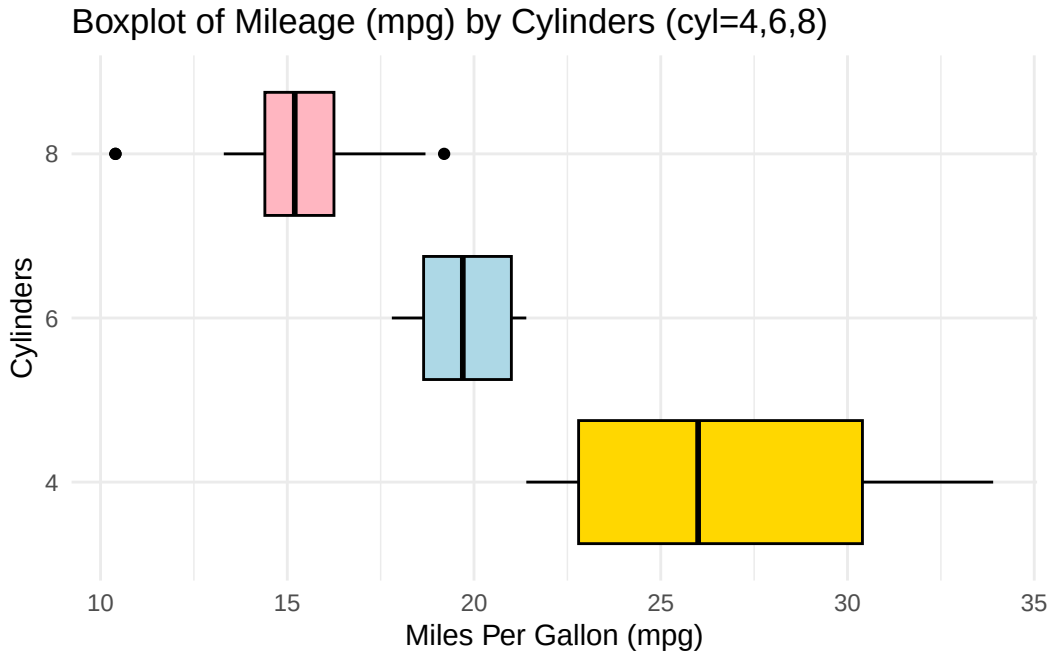
```
library(ggplot2)
# Create separate CDFs for 'am = 0' and 'am = 1'
ggplot(tb, aes(x = mpg,
               color = factor(am))) +
  stat_ecdf(geom = "line") +
  scale_color_manual(values = c("blue", "black")) +
  labs(x = "Miles Per Gallon (mpg)", y = "CDF",
       title = "Cumulative Distribution Function (CDF) of Mileage (mpg) by Transmission (a
       color = "am") +
  theme_minimal()
```



Box Plot across one Category using `ggplot2`

- Visualizing Boxplots of car milegage (`mpg`) broken down by cylinders (`cyl`=4,6,8)

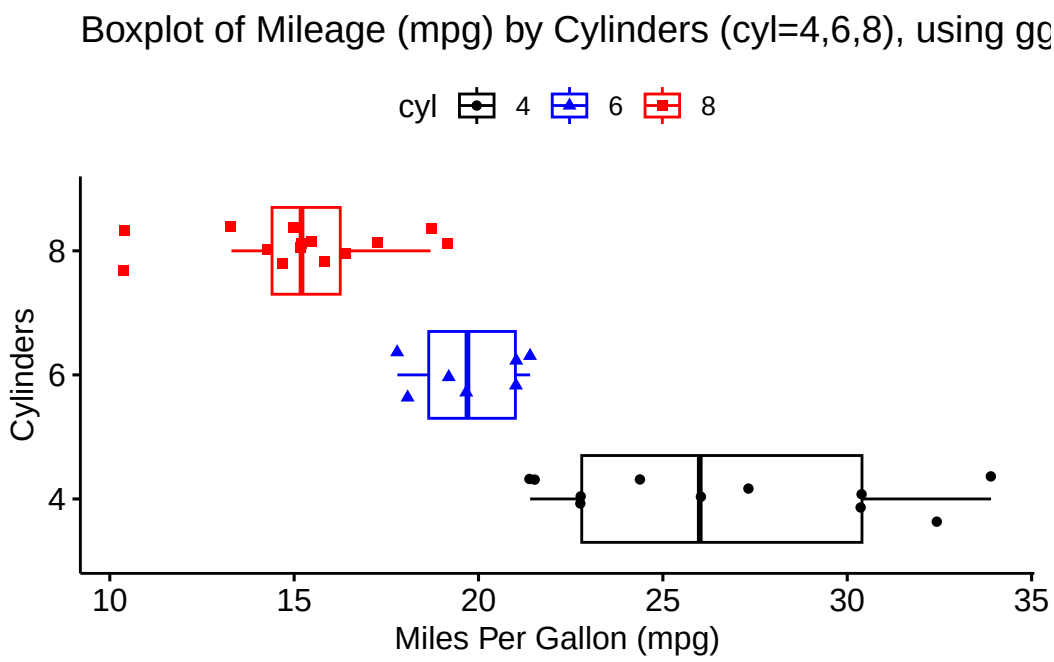
```
library(ggplot2)
ggplot(tb, aes(x = cyl,
               y = mpg)) +
  geom_boxplot(fill = c("gold","lightblue","lightpink"),
               color = "black") +
  coord_flip() +
  labs(title = "Boxplot of Mileage (mpg) by Cylinders (cyl=4,6,8)",
       y = "Miles Per Gallon (mpg)",
       x = "Cylinders") +
  theme_minimal()
```



Box Plot across one Category using `ggpubr`

- The provided R code creates a Boxplot of the `mpg` (miles per gallon) variable in the `tb` dataset, using the `ggboxplot()` function from the `ggpubr` package.

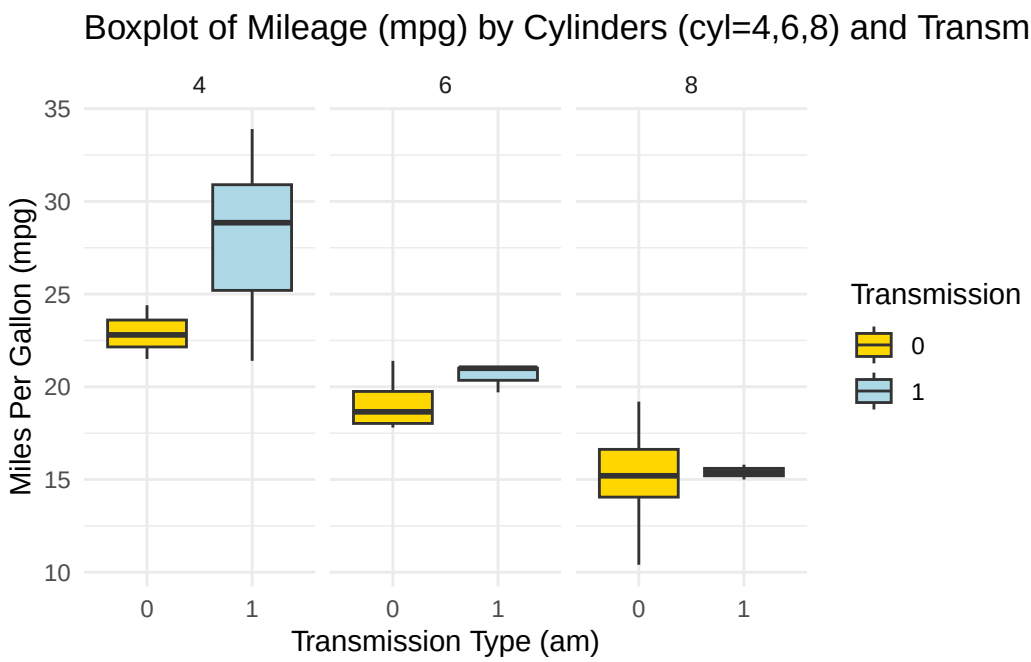
```
library(ggpubr)
ggboxplot(tb,
          y = "mpg",
          x = "cyl",
          color = "cyl",
          fill = "white",
          palette = c("black","blue","red"),
          shape = "cyl",
          orientation = "horizontal",
          add = "jitter", #jitter helps display the data points
          title = "Boxplot of Mileage (mpg) by Cylinders (cyl=4,6,8), using ggpubr::ggboxp
          ylab = "Miles Per Gallon (mpg)",
          xlab = "Cylinders"
)
```



Box Plot across two Categories using `ggplot2`

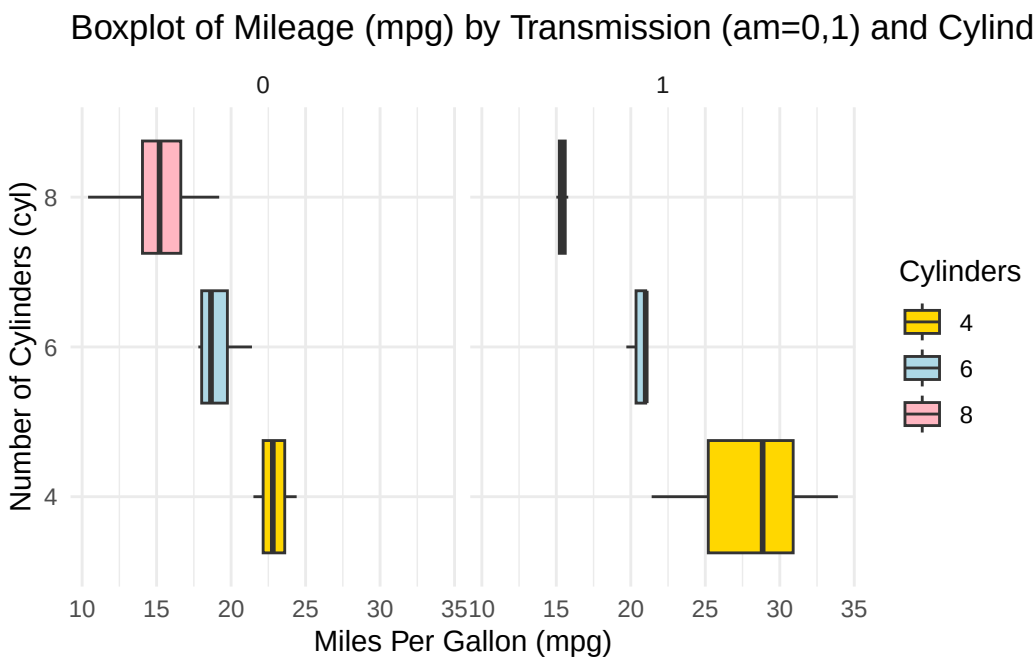
- Visualizing Boxplots of car milegage (`mpg`) broken down by cylinders (`cyl`=4,6,8) and Transmission (am=0,1)

```
ggplot(tb, aes(x = as.factor(am), y = mpg, fill = as.factor(am))) +
  geom_boxplot() +
  scale_fill_manual(values = c("gold", "lightblue"), name = "Transmission") +
  facet_grid(~ cyl) +
  theme_minimal() +
  labs(title = "Boxplot of Mileage (mpg) by Cylinders (cyl=4,6,8) and Transmission (am=0,1
       x = "Transmission Type (am)",
       y = "Miles Per Gallon (mpg)")
```



Alternately:

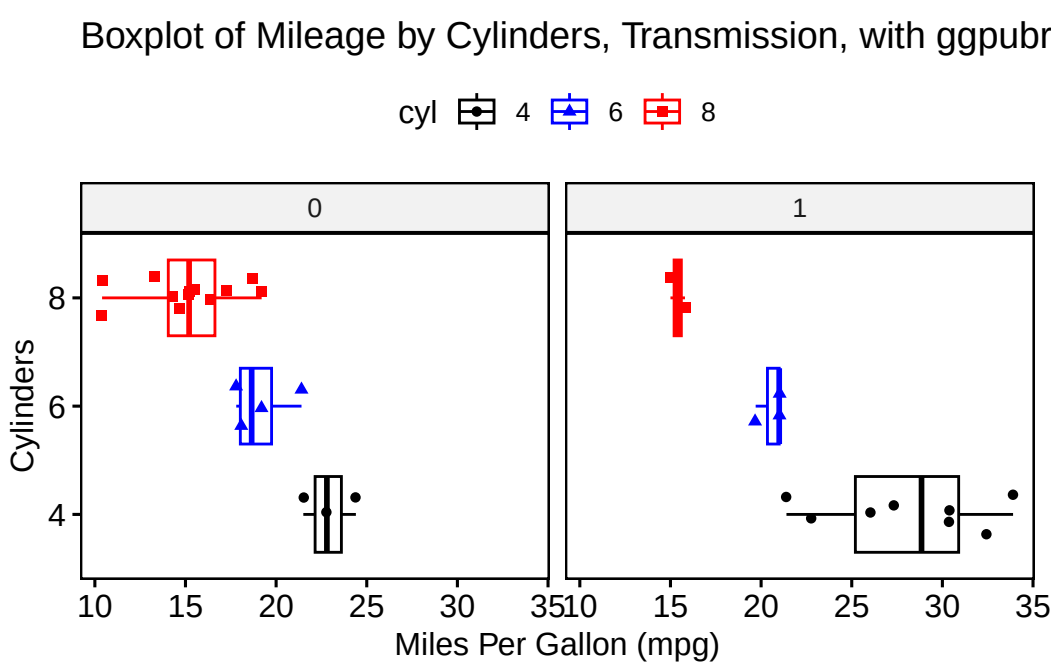
```
ggplot(tb, aes(x = as.factor(cyl), y = mpg, fill = as.factor(cyl))) +
  geom_boxplot() +
  scale_fill_manual(values = c("gold", "lightblue", "lightpink"), name = "Cylinders") +
  facet_grid(~ am) +
  theme_minimal() +
  coord_flip() +
  labs(title = "Boxplot of Mileage (mpg) by Transmission (am=0,1) and Cylinders (cyl=4,6,8
       x = "Number of Cylinders (cyl)",
       y = "Miles Per Gallon (mpg)")
```



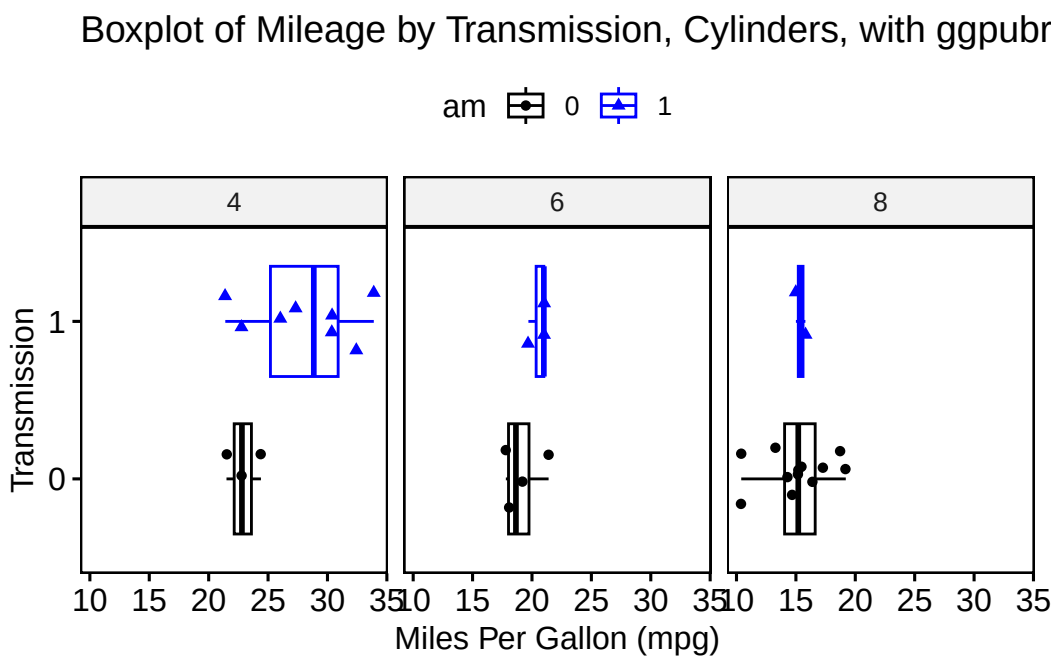
Box Plot across two Categories using `ggpubr`

- The provided R code creates Boxplots of the `mpg` (miles per gallon) variable in the `tb` dataset, using the `ggboxplot()` function from the `ggpubr` package.

```
library(ggpubr)
ggboxplot(tb,
          y = "mpg",
          x = "cyl",
          color = "cyl",
          fill = "white",
          palette = c("black","blue","red"),
          shape = "cyl",
          orientation = "horizontal",
          add = "jitter", #jitter helps display the data points
          facet.by = "am", #split data by "am"
          title = "Boxplot of Mileage by Cylinders, Transmission, with ggpubr::ggboxplot()
          ylab = "Miles Per Gallon (mpg)",
          xlab = "Cylinders"
)
```



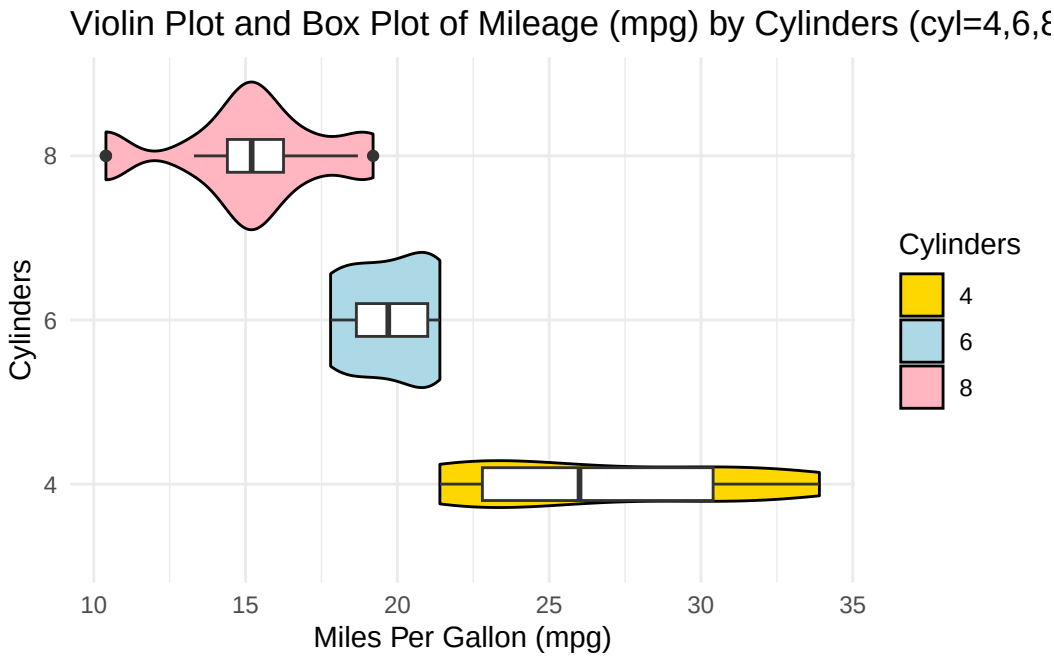

```
library(ggpubr)
ggboxplot(tb,
          y = "mpg",
          x = "am",
          color = "am",
          fill = "white",
          palette = c("black", "blue"),  # assuming am has 2 levels; adjust as needed
          shape = "am",
          orientation = "horizontal",
          add = "jitter", #jitter helps display the data points
          facet.by = "cyl", #split data by "cyl"
          title = "Boxplot of Mileage by Transmission, Cylinders, with ggpubr::ggboxplot()
          ylab = "Miles Per Gallon (mpg)",
          xlab = "Transmission"
)
```



Violin Plot across one Category using `ggplot2`

- We can embed boxplots within the above Violin plots, as follows.

```
library(ggplot2)
ggplot(tb, aes(x = factor(cyl),
               y = mpg)) +
  geom_violin(aes(fill = factor(cyl)),
              color = "black") +
  scale_fill_manual(values = c("gold","lightblue","lightpink"),
                    name = "Cylinders") +
  geom_boxplot(width=0.2,
               fill="white") +
  coord_flip() +
  labs(title = "Violin Plot and Box Plot of Mileage (mpg) by Cylinders (cyl=4,6,8)",
       y = "Miles Per Gallon (mpg)",
       x = "Cylinders") +
  theme_minimal()
```



Summarizing Continuous Data using `dplyr` and `ggplot2`

Across one Category using `dplyr` and `ggplot2`

1. Calculating the mean and standard deviation

- We demonstrate the bivariate relationship between Miles Per Gallon (`mpg`) and Cylinders (`cyl`) using `ggplot2`.

```
suppressPackageStartupMessages(library(dplyr))
s1 <- tb %>%
  group_by(cyl) %>%
  summarise(Mean_mpg = mean(mpg, na.rm = TRUE),
            SD_mpg = sd(mpg, na.rm = TRUE))

kable(s1, digits=2, caption = "Summary Statistics of Mileage (mpg) by Cylinders (cyl=4,6,8
```

Table 0.1: Summary Statistics of Mileage (mpg) by Cylinders (cyl=4,6,8)

cyl	Mean_mpg	SD_mpg
4	26.66	4.51

cyl	Mean_mpg	SD_mpg
6	19.74	1.45
8	15.10	2.56

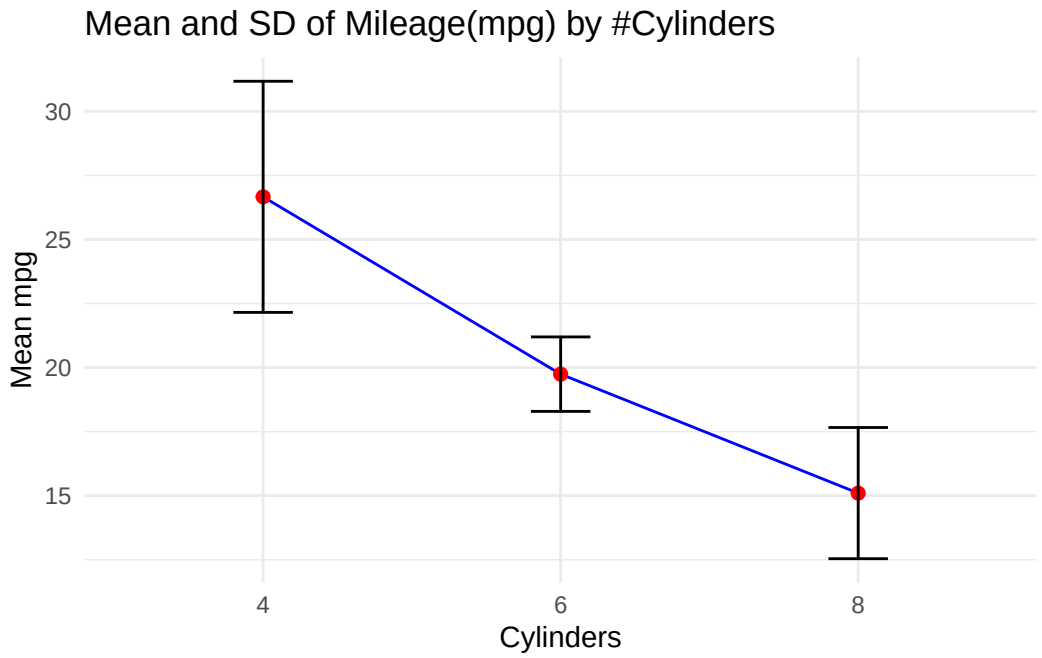
2. **Discussion:**

- In this code, we use the pipe operator `%\>%` to perform a series of operations. We first group the data by the `cyl` column using the `group_by()` function. We then use `summarise()` to apply the `mean()` and `sd()` functions to the `mpg` column.
- The results are stored in new columns, aptly named `Mean_mpg` and `SD_mpg`.
- We set `na.rm = TRUE` in both `mean()` and `sd()` function calls, to remove any missing values before calculation.
- The data resulting from the above code consists of grouped cylinder counts (`cyl`), their corresponding mean miles per gallon (`Mean_mpg`), and the standard deviation of miles per gallon (`SD_mpg`).[1]

3. **Visualizing the mean and standard deviation**

- A simple way to visualize this data is to create a **line plot** for the mean miles per gallon with **error bars** indicating the standard deviation. Here is an example of how we could do this with `ggplot2`:

```
suppressPackageStartupMessages(library(ggplot2))
ggplot(s1,
       aes(x = cyl, y = Mean_mpg)) +
  geom_line(group=1, color = "blue") +
  geom_point(size = 2, color = "red") +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg,
                    ymax = Mean_mpg + SD_mpg),
                width = .2, colour = "black") +
  labs(x = "Cylinders", y = "Mean mpg",
       title = "Mean and SD of Mileage(mpg) by #Cylinders") +
  theme_minimal()
```



4. **Discussion**:

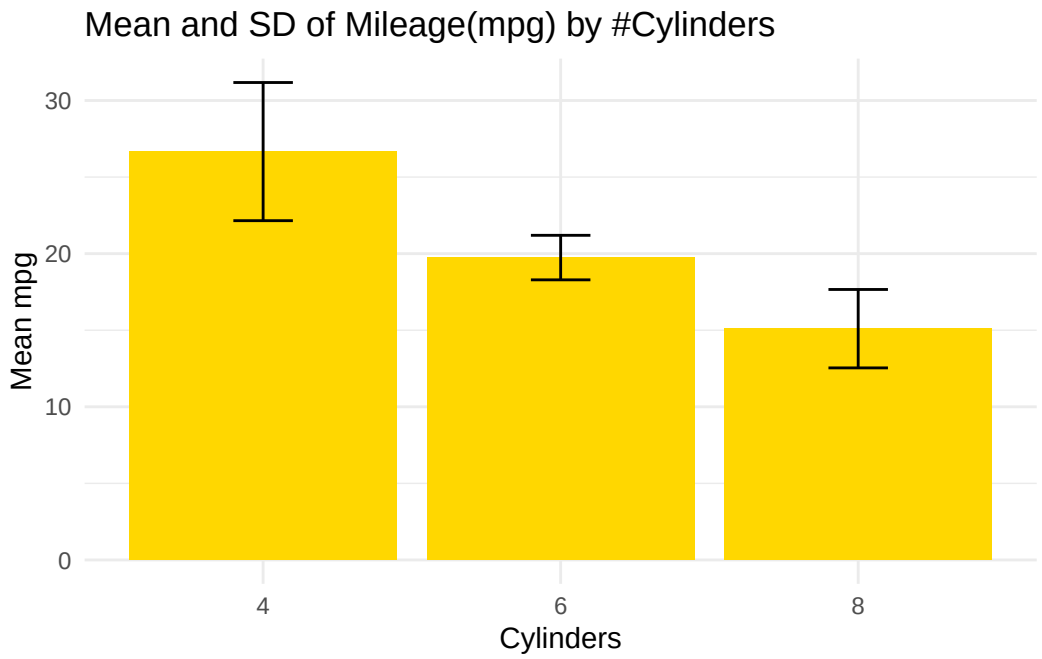
- `aes(x = cyl, y = Mean_mpg)` assigns the `cyl` values to the x-axis and `Mean_mpg` to the y-axis.
- `geom_line(group=1, color = "blue")` adds a blue line connecting the data points.
- `geom_point(size = 2, color = "red")` adds red points for each data point.
- `geom_errorbar(aes(ymin = Mean_mpg - SD_mpg, ymax = Mean_mpg + SD_mpg), width = .2, colour = "black")` adds error bars, where the error is the standard deviation.
- The `ymin` and `ymax` arguments define the range of the error bars.
- `labs(x = "Cylinders", y = "Mean mpg")` labels the x and y axes.
- `theme_minimal()` applies a minimal theme to the plot.

5. **Visualizing the mean and standard deviation - Alternate Method**

- An alternate method is to visualize this mean by creating a **bar plot**, with **error bars** indicating the standard deviation. Here is an example of how we could do this with `ggplot2`:

```
library(ggplot2)
# mpg plot
```

```r
ggplot(s1,
       aes(x = cyl,
           y = Mean_mpg)) +
  geom_bar(stat = "identity",
           fill = "gold") +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg,
                    ymax = Mean_mpg + SD_mpg),
                width = .2) +
  labs(x = "Cylinders", y = "Mean mpg",
       title = "Mean and SD of Mileage(mpg) by #Cylinders") +
  theme_minimal()
```



6. **Discussion**:

- `ggplot(s1, aes(x = cyl, y = Mean_mpg))`: The `ggplot()` function initializes a ggplot object using dataframe `s1` and mapping aesthetic elements. Here, `aes(x = cyl, y = Mean_mpg)` specifies that the x-axis represents `cyl` (number of cylinders) and the y-axis represents `Mean_mpg` (mean miles per gallon).
- `geom_bar(stat = "identity", fill = "gold")`: The `geom_bar()` function is used to create a bar chart. Setting `stat = "identity"` indicates that the heights of the bars represent the values in the data (in this case, `Mean_mpg`). The `fill = "gold"` argument sets the color of the bars.

- `geom_errorbar()` adds error bars to the plot. The arguments `aes(ymin = Mean_mpg - SD_mpg, ymax = Mean_mpg + SD_mpg)` set the bottom (`ymin`) and top (`ymax`) of the error bars to represent one standard deviation below and above the mean, respectively. `width = .2` sets the horizontal width of the error bars.
- `labs(x = "Cylinders", y = "Mean mpg")`: The `labs()` function is used to specify the labels for the x-axis and y-axis.
- `theme_minimal()`: The `theme_minimal()` function is used to set a minimalistic theme for the plot.

7. We extend this code to demonstrate how to measure the bivariate relationships between multiple continuous variables from the mtcars data and the categorical variable number of Cylinders (`cyl`), using `ggplot2`. Specifically, we want to measure the mean and SD of continuous variables (i) Miles Per Gallon (`mpg`); (ii) Weight (`wt`); (iii) Horsepower (`hp`) across the number of Cylinders (`cyl`).

```
library(dplyr)
s3 <- tb %>%
  group_by(cyl) %>%
  summarise(
    Mean_mpg = mean(mpg, na.rm = TRUE),
    SD_mpg = sd(mpg, na.rm = TRUE),
    Mean_wt = mean(wt, na.rm = TRUE),
    SD_wt = sd(wt, na.rm = TRUE),
    Mean_hp = mean(hp, na.rm = TRUE),
    SD_hp = sd(hp, na.rm = TRUE)
    )
kable(s3,
      digits=2,
      caption = "Summary of Mileage (mpg), Weight (wt), Horsepower (hp) by Cylinders (cyl=
```

Table 0.2: Summary of Mileage (mpg), Weight (wt), Horsepower (hp) by Cylinders (cyl=4,6,8)

cyl	Mean_mpg	SD_mpg	Mean_wt	SD_wt	Mean_hp	SD_hp
4	26.66	4.51	2.29	0.57	82.64	20.93
6	19.74	1.45	3.12	0.36	122.29	24.26
8	15.10	2.56	4.00	0.76	209.21	50.98

8. **Discussion**:

- With `tb %>%`, we indicate that we are going to perform a series of operations on the `tb` data frame. The `group_by(cyl)` groups the data by the `cyl` variable.

- The `summarise()` function calculates the mean and standard deviation (SD) of three variables (`mpg`, `wt`, and `hp`). The`na.rm = TRUE` argument inside `mean()` and `sd()` functions is used to exclude any NA values from these calculations.
- The resulting calculations are assigned to new variables (`Mean_mpg`, `SD_mpg`, `Mean_wt`, `SD_wt`, `Mean_hp`, and `SD_hp`) which will be the columns in the summarised data frame.
- To summarize, this script groups the data in the `tb` tibble by `cyl` and then calculates the mean and standard deviation of the `mpg`, `wt`, and `hp` variables for each group. [1]

Across two Categories using `ggplot2`

1. We demonstrate the relationship between Miles Per Gallon (`mpg`) and Cylinders (`cyl`) and Transmission type (`am`) using `ggplot2`. Recall that a car's transmission may be automatic (`am=0`) or manual (`am=1`).

```
s4 <- tb %>%
  group_by(cyl, am) %>%
  summarise(Mean_mpg = mean(mpg, na.rm = TRUE),
            SD_mpg = sd(mpg, na.rm = TRUE))
```

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

```
kable(s4,
      digits=2,
      caption = "Summary of Mileage (mpg) by Cylinders (cyl=4,6,8) and Transmission (am=0,
```

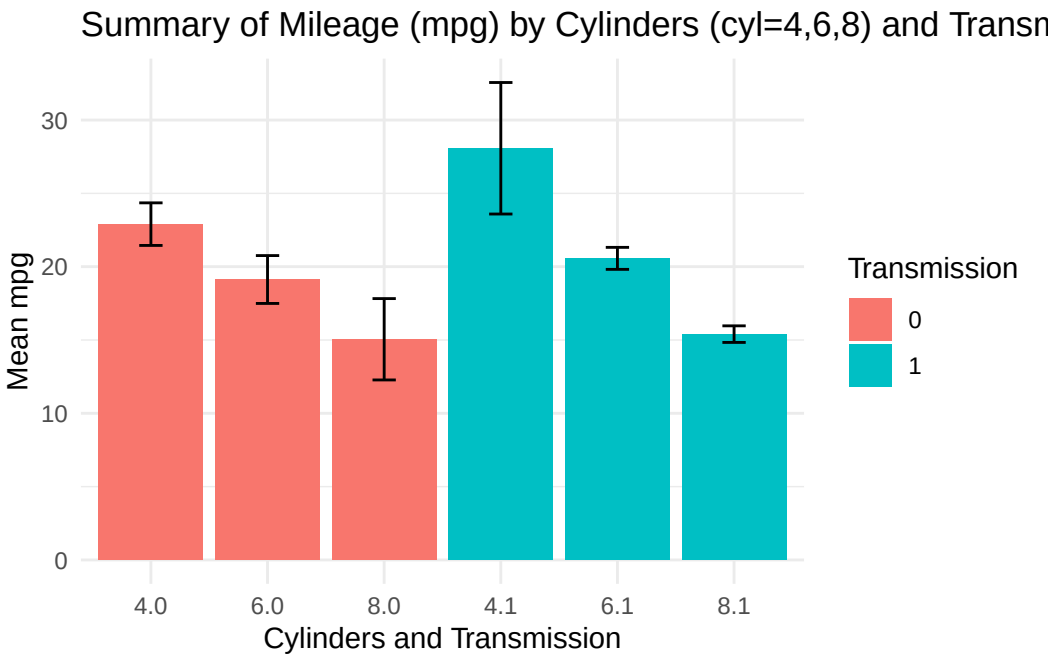
Table 0.3: Summary of Mileage (mpg) by Cylinders (cyl=4,6,8) and Transmission (am=0,1)

cyl	am	Mean_mpg	SD_mpg
4	0	22.90	1.45
4	1	28.08	4.48
6	0	19.12	1.63
6	1	20.57	0.75
8	0	15.05	2.77
8	1	15.40	0.57

2. **Discussion**:

- The above code provides the mean and standard deviation of `mpg` for each unique combination of `cyl` and `am`.
- Here is how it can be visualized:

```
# Create the plot
ggplot(s4, aes(x = interaction(cyl, am),
               y = Mean_mpg,
               fill = as.factor(am))) +
  geom_bar(stat = "identity",
           position = position_dodge()) +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg,
                    ymax = Mean_mpg + SD_mpg),
                width = .2,
                position = position_dodge(.9)) +
  labs(x = "Cylinders and Transmission",
       y = "Mean mpg",
       fill = "Transmission",
       title = "Summary of Mileage (mpg) by Cylinders (cyl=4,6,8) and Transmission (am=0,1
  theme_minimal()
```



3. In the below code, the order of the variables is reversed - the data is first grouped by `am`, then by `cyl`. So, the function first sorts the data by the `am` variable, and within each `am` group, it further groups the data by `cyl`.

```r
library(dplyr)
s5 <- tb %>%
  group_by(am, cyl) %>%
  summarise(Mean_mpg = mean(mpg, na.rm = TRUE),
            SD_mpg = sd(mpg, na.rm = TRUE))
```

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

```r
kable(s5,
      digits=2, caption = "Summary of Mileage (mpg) by Transmission (am=0,1) and Cylinders
```

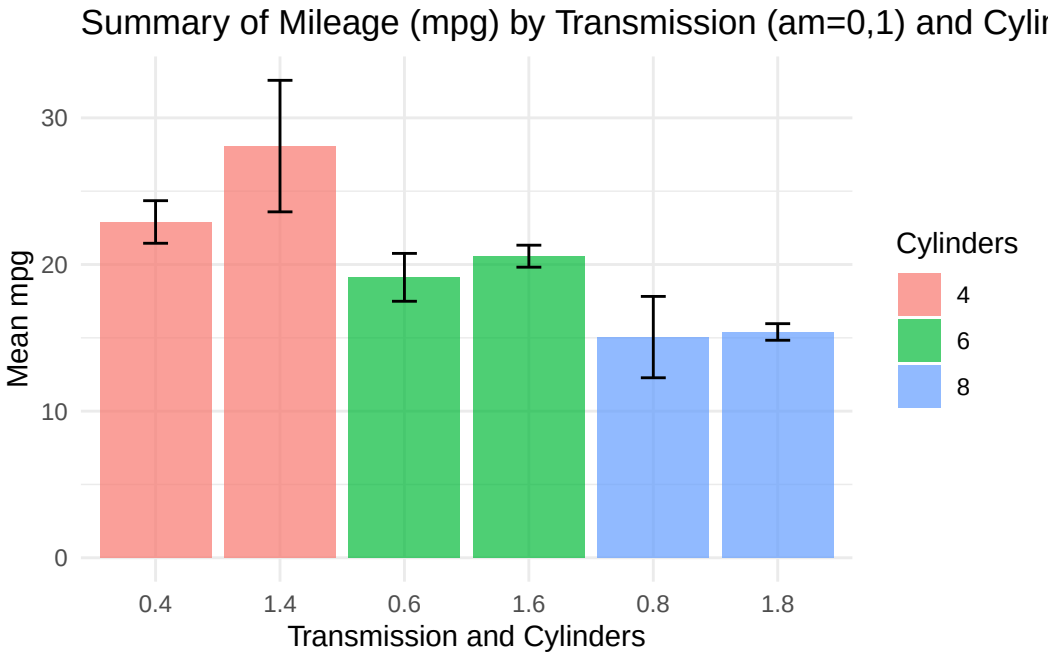
Table 0.4: Summary of Mileage (mpg) by Transmission (am=0,1) and Cylinders (cyl=4,6,8)

am	cyl	Mean_mpg	SD_mpg
0	4	22.90	1.45
0	6	19.12	1.63
0	8	15.05	2.77
1	4	28.08	4.48
1	6	20.57	0.75
1	8	15.40	0.57

- Here is how it can be visualized:

```r
# Create the plot
ggplot(s5,
       aes(x = interaction(am, cyl),
           y = Mean_mpg,
           fill = as.factor(cyl))) +
  geom_bar(stat = "identity",
           alpha = 0.7,
           position = position_dodge()) +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg,
                    ymax = Mean_mpg + SD_mpg),
                width = .2,
                position = position_dodge(.9)) +
  labs(x = "Transmission and Cylinders",
       y = "Mean mpg",
       fill = "Cylinders",
```

```
      title = "Summary of Mileage (mpg) by Transmission (am=0,1) and Cylinders (cyl=4,6,8
  theme_minimal()
```



4. The following code produces a new data frame that contains the mean and standard deviation of the continuous variables `mpg`, `wt`, and `hp` for each combination of the factor variables `am` and `cyl`. [1]

```
s6 <- tb %>%
  group_by(am, cyl) %>%
  summarise(
    Mean_mpg = mean(mpg, na.rm = TRUE),
    SD_mpg = sd(mpg, na.rm = TRUE),
    Mean_wt = mean(wt, na.rm = TRUE),
    SD_wt = sd(wt, na.rm = TRUE),
    Mean_hp = mean(hp, na.rm = TRUE),
    SD_hp = sd(hp, na.rm = TRUE)
    )
```

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

```
kable(s6,
      digits=2, caption = "Summary of Mileage (mpg) by Transmission (am=0,1) and Cylinder
```

Table 0.5: Summary of Mileage (mpg) by Transmission (am=0,1) and Cylinder (cyl=4,6,8)

am	cyl	Mean_mpg	SD_mpg	Mean_wt	SD_wt	Mean_hp	SD_hp
0	4	22.90	1.45	2.94	0.41	84.67	19.66
0	6	19.12	1.63	3.39	0.12	115.25	9.18
0	8	15.05	2.77	4.10	0.77	194.17	33.36
1	4	28.08	4.48	2.04	0.41	81.88	22.66
1	6	20.57	0.75	2.76	0.13	131.67	37.53
1	8	15.40	0.57	3.37	0.28	299.50	50.20

References

[1]

Wickham, H., François, R., Henry, L., & Müller, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. https://CRAN.R-project.org/package=dplyr

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, https://ggplot2.tidyverse.org.

[2]

Kassambara A (2023). ggpubr: 'ggplot2' Based Publication Ready Plots. R package version 0.6.0, https://rpkgs.datanovia.com/ggpubr/.

Appendix A

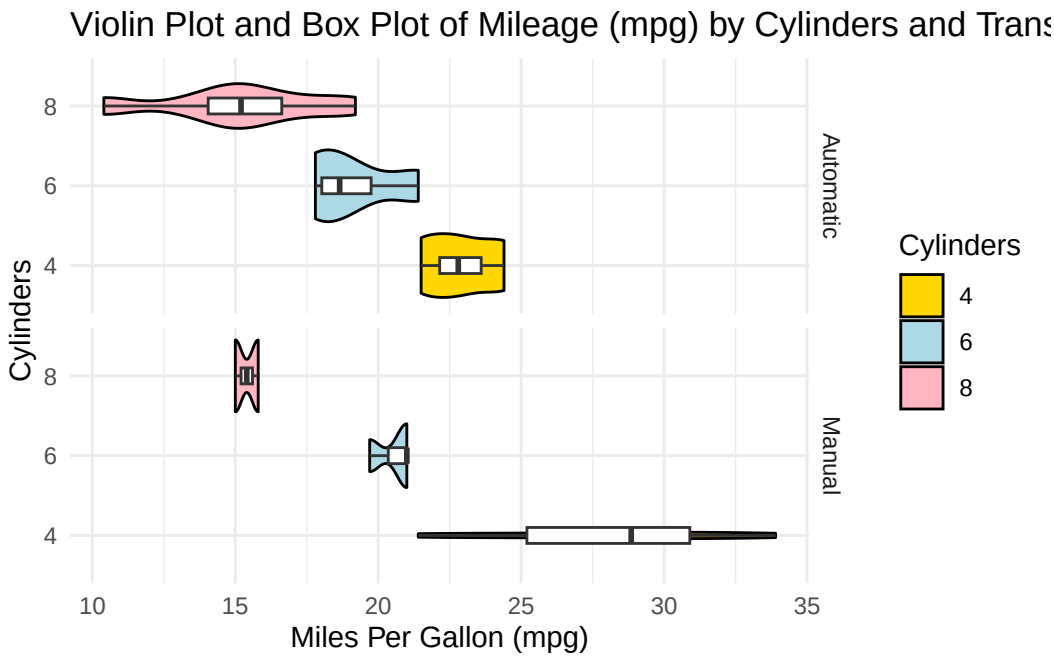
Appendix A1: Violin Plot across two Categories using `ggplot2`

- We can embed boxplots within the above Violin plots, as follows.

```
library(ggplot2)

ggplot(tb, aes(x = factor(cyl), y = mpg)) +
  geom_violin(aes(fill = factor(cyl)), color = "black") +
  scale_fill_manual(values = c("gold", "lightblue", "lightpink"), name = "Cylinders") +
```

```
  geom_boxplot(width = 0.2, fill = "white") +
  coord_flip() +
  labs(title = "Violin Plot and Box Plot of Mileage (mpg) by Cylinders and Transmission",
       y = "Miles Per Gallon (mpg)",
       x = "Cylinders") +
  facet_grid(am ~ ., scales = "free_y", space = "free_y", labeller = labeller(am = functio
  theme_minimal()
```



- Alternately, We can embed boxplots within the above Violin plots, as follows.

```
library(ggplot2)
library(dplyr)

# Modify the data first
tb_modified <- tb %>%
  mutate(am = factor(am, levels = c(0, 1), labels = c("Automatic", "Manual")))

# Create the plot
ggplot(tb_modified,
       aes(x = factor(cyl),
           y = mpg)) +
  geom_violin(aes(fill = factor(cyl)),
```

```
            color = "black") +
scale_fill_manual(values = c("gold", "lightblue", "lightpink"),
                  name = "Cylinders") +
geom_boxplot(width = 0.2, fill = "white") +
coord_flip() +
labs(title = "Violin Plot and Box Plot of Mileage (mpg) by Cylinders and Transmission",
     y = "Miles Per Gallon (mpg)",
     x = "Cylinders") +
facet_grid(am ~ .,
           scales = "free_y",
           space = "free_y") +
theme_minimal()
```