Continuous Data (4 of 6)

Aug 4, 2023. V1.4

1. THIS CHAPTER explores Continuous x Categorical data using **ggplot2**. Specifically, it demonstrates the use of the popular **ggplot2** package to further explore *bivariate continuous data across categories*.
2. **Data**: Let us work with the same `mtcars` data from the previous chapter. Suppose we run the following code to prepare the data for subsequent analysis. The data is now in a tibble called `tb`:

```
# Load the required libraries, suppressing annoying startup messages
library(tibble)
suppressPackageStartupMessages(library(dplyr))
# Read the mtcars dataset into a tibble called tb
data(mtcars)
tb <- as_tibble(mtcars)
# Convert several numeric columns into factor variables
tb$cyl <- as.factor(tb$cyl)
tb$vs <- as.factor(tb$vs)
tb$am <- as.factor(tb$am)
tb$gear <- as.factor(tb$gear)
# Directly access the data columns of tb, without tb$mpg
attach(tb)
```

Summarizing Continuous Data across one Category, using `ggplot2`

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

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

```
            SD_mpg = sd(mpg, na.rm = TRUE))
print(s1)
```

```
# A tibble: 3 x 3
  cyl   Mean_mpg SD_mpg
  <fct>    <dbl>  <dbl>
1 4         26.7   4.51
2 6         19.7   1.45
3 8         15.1   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. [1]

3. Visualizing the mean and standard deviation

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

```
library(ggplot2)
```

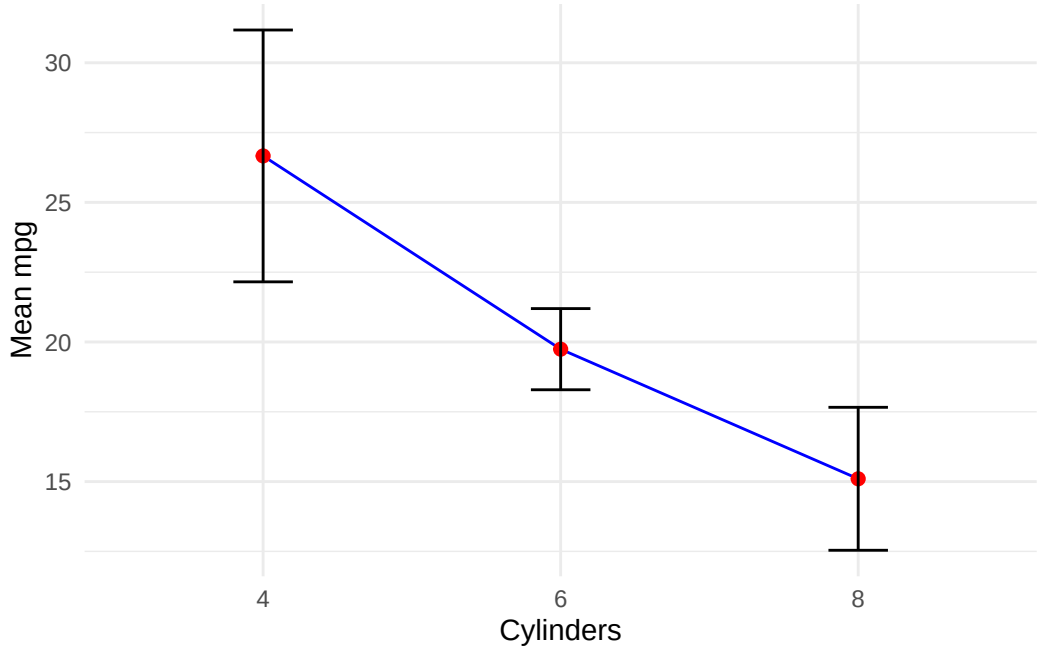
```
Attaching package: 'ggplot2'
```

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

    mpg
```

```
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") +
  theme_minimal()
```



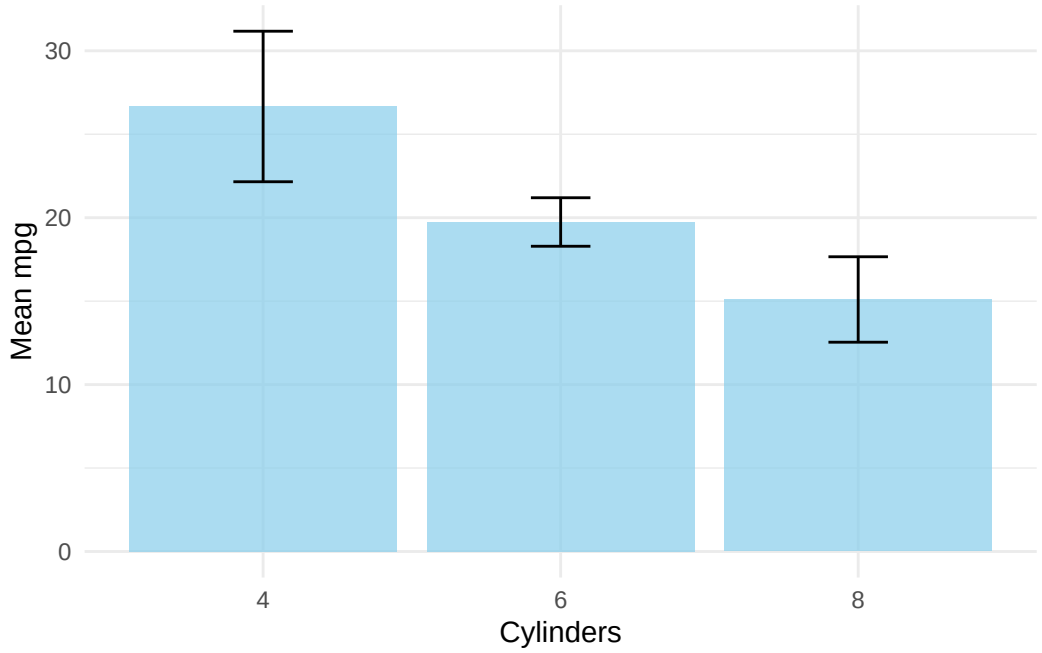
4. Discussion:

- `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. Alternate visualization:

```
library(ggplot2)

# mpg plot
ggplot(s1, aes(x = cyl, y = Mean_mpg)) +
  geom_bar(stat = "identity",
           fill = "skyblue",
           alpha = 0.7) +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg,
                    ymax = Mean_mpg + SD_mpg),
                width = .2) +
  labs(x = "Cylinders", y = "Mean mpg") +
  theme_minimal()
```



5. 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 consider the continuous variables (i) Miles Per Gallon (`mpg`); (ii) Weight (`wt`); (iii) Horsepower (`hp`) across the number of Cylinders (`cyl`).

```r
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)
    )
print(s3)
```

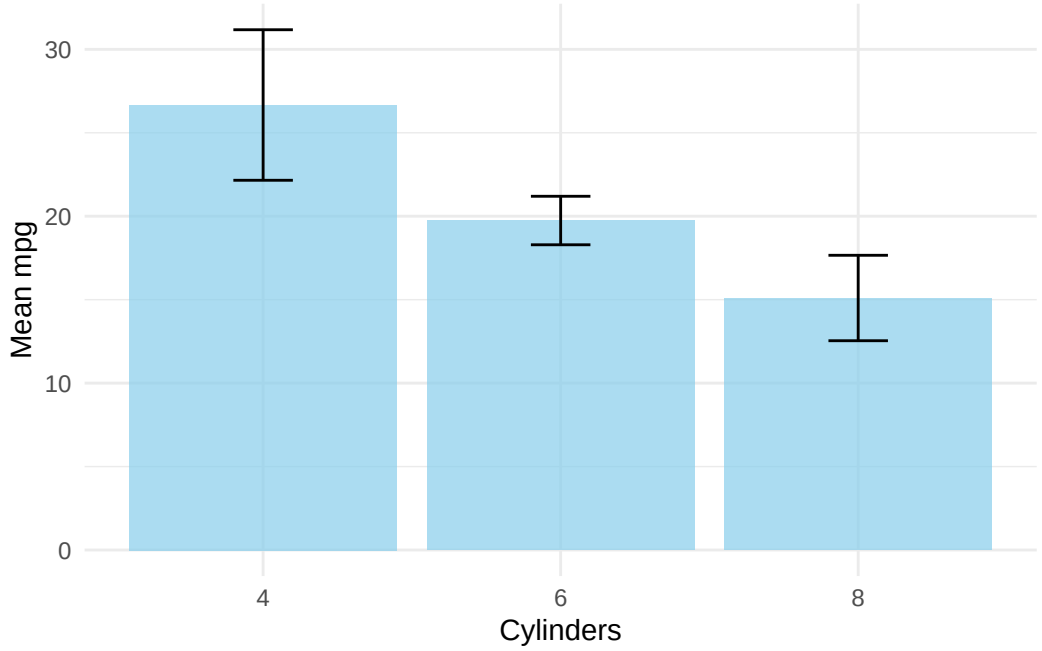
```
# A tibble: 3 x 7
  cyl   Mean_mpg SD_mpg Mean_wt SD_wt Mean_hp SD_hp
  <fct>    <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
1 4         26.7   4.51    2.29 0.570    82.6  20.9
2 6         19.7   1.45    3.12 0.356   122.   24.3
3 8         15.1   2.56    4.00 0.759   209.   51.0
```

6. Discussion:

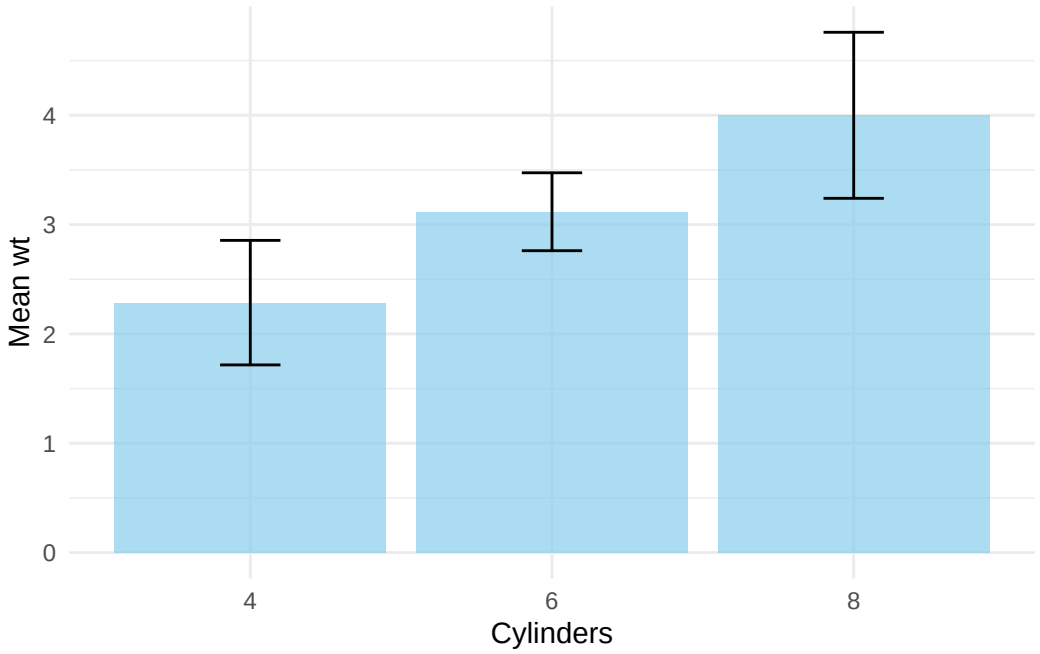
- With `tb %>%`, we indicate that we are going to perform a series of operations on the `tb` data frame. The next operation is `group_by(cyl)`, which groups the data by the `cyl` variable.
- The `summarise()` function is then used to create a new data frame that summarizes the grouped data. Inside `summarise()`, we calculate 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. The summarised data will contain one row for each group (in this case, each unique value of `cyl`), and columns for each of the summary statistics.
- 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]
- The following code visualizes the three summary statistics:

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

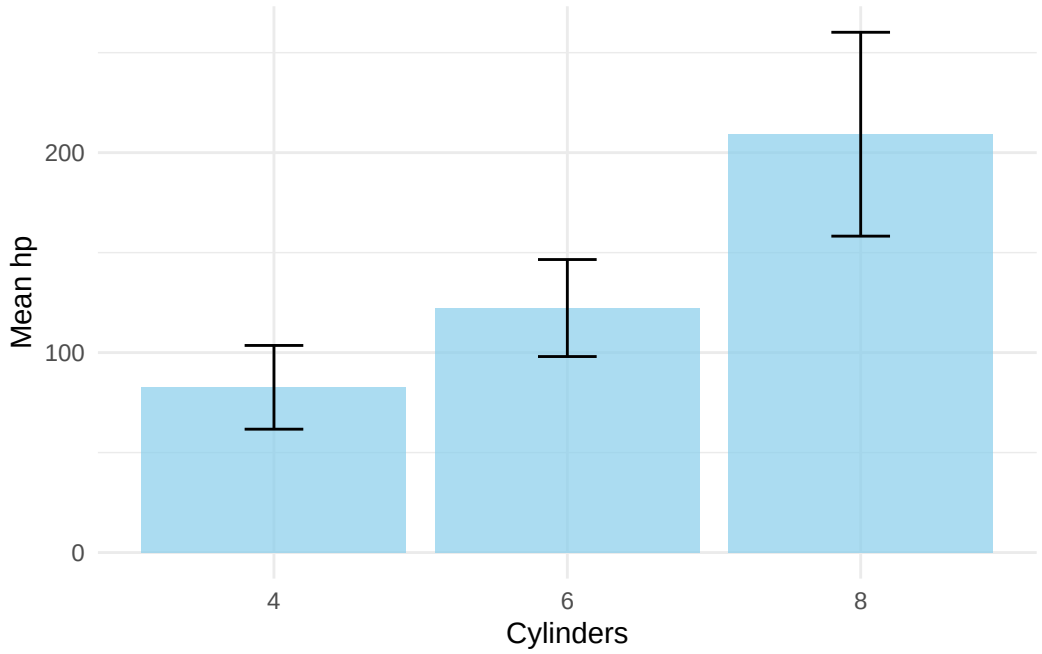
```r
ggplot(s3, aes(x = cyl, y = Mean_mpg)) +
  geom_bar(stat = "identity", fill = "skyblue", alpha = 0.7) +
  geom_errorbar(aes(ymin = Mean_mpg - SD_mpg, ymax = Mean_mpg + SD_mpg), width = .2) +
  labs(x = "Cylinders", y = "Mean mpg") +
  theme_minimal()
```




```r
# wt plot
ggplot(s3, aes(x = cyl, y = Mean_wt)) +
  geom_bar(stat = "identity", fill = "skyblue", alpha = 0.7) +
  geom_errorbar(aes(ymin = Mean_wt - SD_wt, ymax = Mean_wt + SD_wt), width = .2) +
  labs(x = "Cylinders", y = "Mean wt") +
  theme_minimal()
```




```
# hp plot
ggplot(s3, aes(x = cyl, y = Mean_hp)) +
  geom_bar(stat = "identity", fill = "skyblue", alpha = 0.7) +
  geom_errorbar(aes(ymin = Mean_hp - SD_hp, ymax = Mean_hp + SD_hp), width = .2) +
  labs(x = "Cylinders", y = "Mean hp") +
  theme_minimal()
```



Visualizing Continuous Data across one Category, 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) Histograms, using ggplot2;

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

(iii) Box plots, using ggplot2;

(iv) Bee Swarm plots, using ggplot2;

(v) Violin plots, using ggplot2;

(vi) Q-Q plots, using ggplot2.

Summarizing Continuous Data 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`).

```r
library(dplyr)
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.
```

```
# A tibble: 6 x 4
# Groups:   cyl [3]
  cyl   am    Mean_mpg SD_mpg
  <fct> <fct>    <dbl>  <dbl>
1 4     0         22.9  1.45
2 4     1         28.1  4.48
3 6     0         19.1  1.63
4 6     1         20.6  0.751
5 8     0         15.0  2.77
6 8     1         15.4  0.566
```

2. Discussion:

- In the above code, we are grouping by both `cyl` and `am` before summarizing. This will provide the mean and standard deviation of `mpg` for each unique combination of `cyl` and `am`.
- 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)
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.
```

```
# A tibble: 6 x 4
# Groups:   am [2]
  am    cyl   Mean_mpg SD_mpg
  <fct> <fct>    <dbl>  <dbl>
1 0     4         22.9  1.45
2 0     6         19.1  1.63
3 0     8         15.0  2.77
4 1     4         28.1  4.48
5 1     6         20.6  0.751
6 1     8         15.4  0.566
```

3. 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]

```
library(dplyr)
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.
```

```
# A tibble: 6 x 8
# Groups:   am [2]
  am    cyl   Mean_mpg SD_mpg Mean_wt SD_wt Mean_hp SD_hp
  <fct> <fct>    <dbl>  <dbl>   <dbl> <dbl>   <dbl> <dbl>
1 0     4         22.9  1.45     2.94 0.408    84.7 19.7
2 0     6         19.1  1.63     3.39 0.116   115.   9.18
3 0     8         15.0  2.77     4.10 0.768   194.  33.4
4 1     4         28.1  4.48     2.04 0.409    81.9 22.7
5 1     6         20.6  0.751    2.76 0.128   132.  37.5
6 1     8         15.4  0.566    3.37 0.283   300.  50.2
```

Visualizing Continuous Data across two Categories using `ggplot2`

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.