Datacamp_Data Visualization with ggplot2 (Part 1)___Data

dizhen

2019/4/8

Objects and Layers

Limitation of base package

1. Plot does not get redrawn
2. Plot is drawn as a image
3. Need to manually add legend
4. No unified framework for plotting

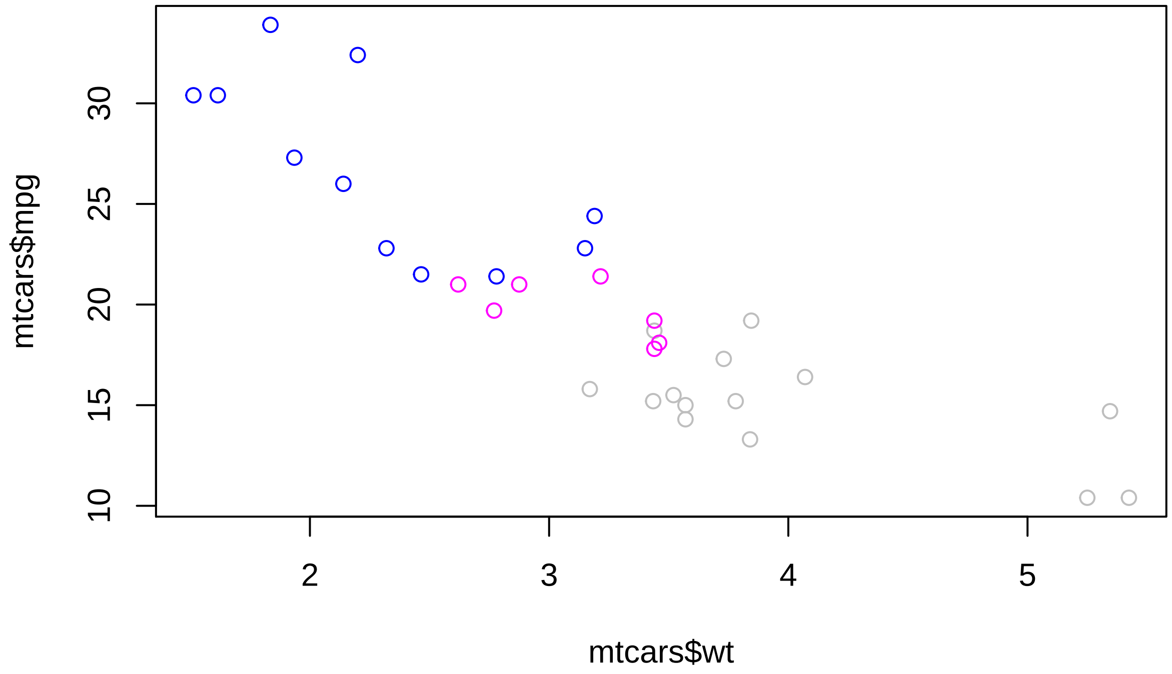
```
library(ggplot2)
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

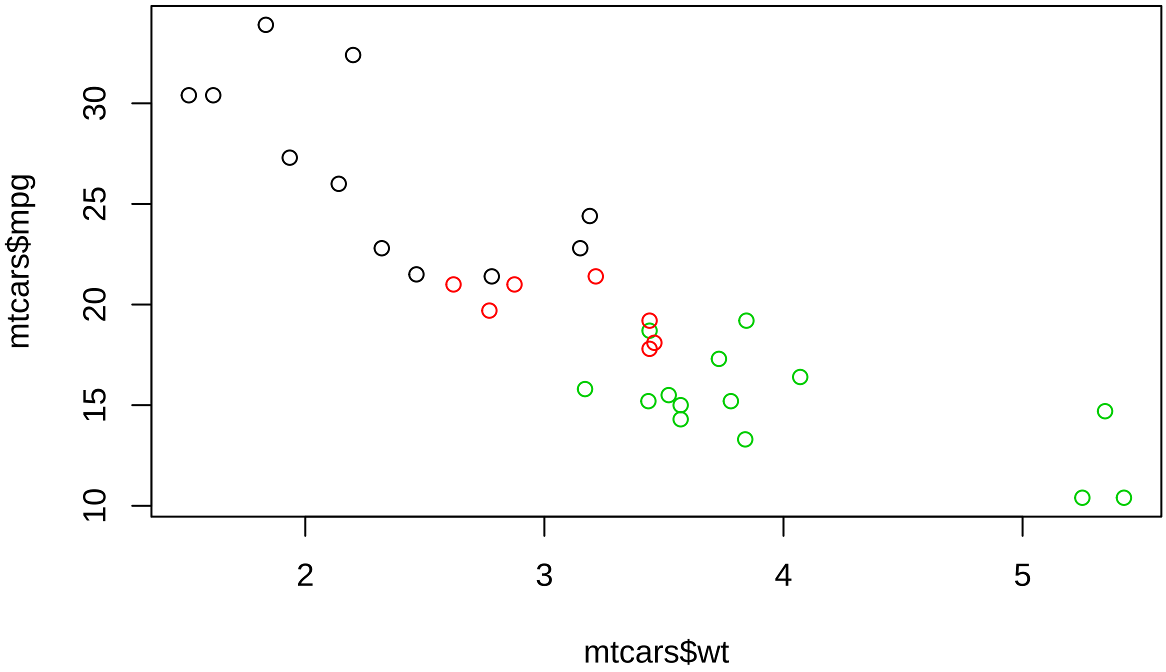
```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
# Plot the correct variables of mtcars
plot(mtcars$wt, mtcars$mpg, col = mtcars$cyl)
```

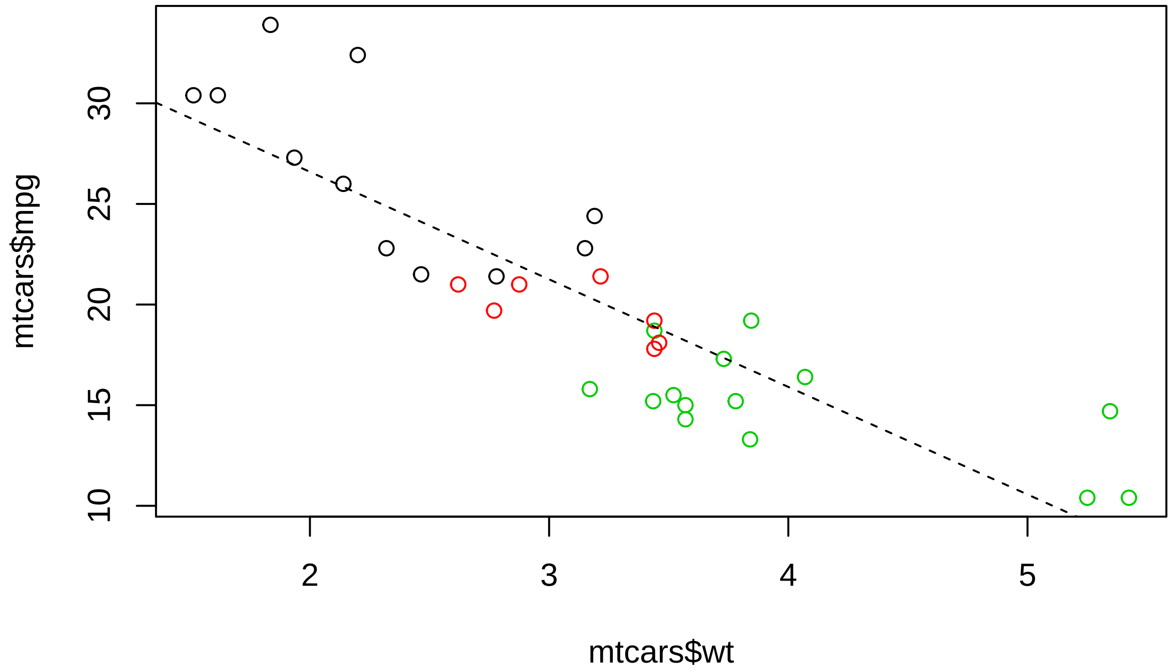


```
# Change cyl inside mtcars to a factor
mtcars$fcyl <- as.factor(mtcars$cyl)

# Make the same plot as in the first instruction
plot(mtcars$wt, mtcars$mpg, col = mtcars$fcyl)
```



lapply() applies the function it was given to each element of the vector and returns the results in a list.

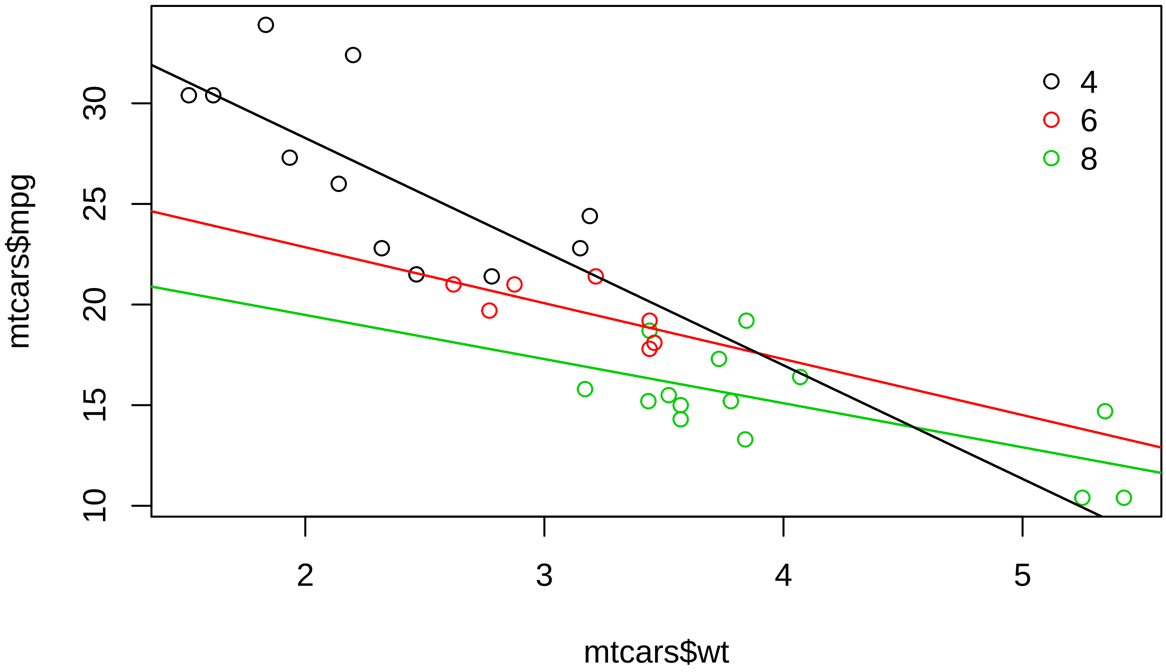
```r
# Use lm() to calculate a linear model and save it as carModel
carModel <- lm(mpg ~ wt, data = mtcars)

# Basic plot
mtcars$cyl <- as.factor(mtcars$cyl)
plot(mtcars$wt, mtcars$mpg, col = mtcars$cyl)

# Call abline() with carModel as first argument and set lty to 2 (lty: line types)
abline(carModel, lty = 2)
```



```
# Plot each subset efficiently with lapply
# subset according to cyl
plot(mtcars$wt, mtcars$mpg, col = mtcars$cyl)
lapply(mtcars$cyl, function(x) {
  abline(lm(mpg ~ wt, mtcars, subset = (cyl == x)), col = x)
  })
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
##
## [[19]]
## NULL
##
## [[20]]
## NULL
##
## [[21]]
## NULL
##
## [[22]]
## NULL
##
## [[23]]
## NULL
##
## [[24]]
## NULL
##
## [[25]]
## NULL
##
## [[26]]
## NULL
##
## [[27]]
## NULL
##
## [[28]]
## NULL
##
## [[29]]
## NULL
##
## [[30]]
## NULL
##
## [[31]]
## NULL
##
## [[32]]
## NULL
```

```
# draw the legend of the plot
# pch: legend bullet type
# bty: whether there is a frame for bty
legend(x = 5, y = 33, legend = levels(mtcars$cyl),
       col = 1:3, pch = 1, bty = "n")
```



base package and ggplot2, part 3

In this exercise you'll recreate the base package plot in ggplot2.

The code for base R plotting is given at the top. The first line of code already converts the cyl variable of mtcars to a factor.

Instructions

1. Plot 1: add geom_point() in order to make a scatter plot.
2. Plot 2: copy and paste Plot 1.
3. Add a linear model for each subset according to cyl by adding a geom_smooth() layer.
4. Inside this geom_smooth(), set method to "lm" and se to FALSE.
5. Note: geom_smooth() will automatically draw a line per cyl subset. It recognizes the groups you want to identify by color in the aes() call within the ggplot() command.
6. Plot 3: copy and paste Plot 2.
7. Plot a linear model for the entire dataset, do this by adding another geom_smooth() layer.
8. Set the group aesthetic inside this geom_smooth() layer to 1. This has to be set within the aes() function.

9. Set method to “lm”, se to FALSE and linetype to 2. These have to be set outside aes() of the geom_smooth().
10. Note: the group aesthetic will tell ggplot() to draw a single linear model through all the points.

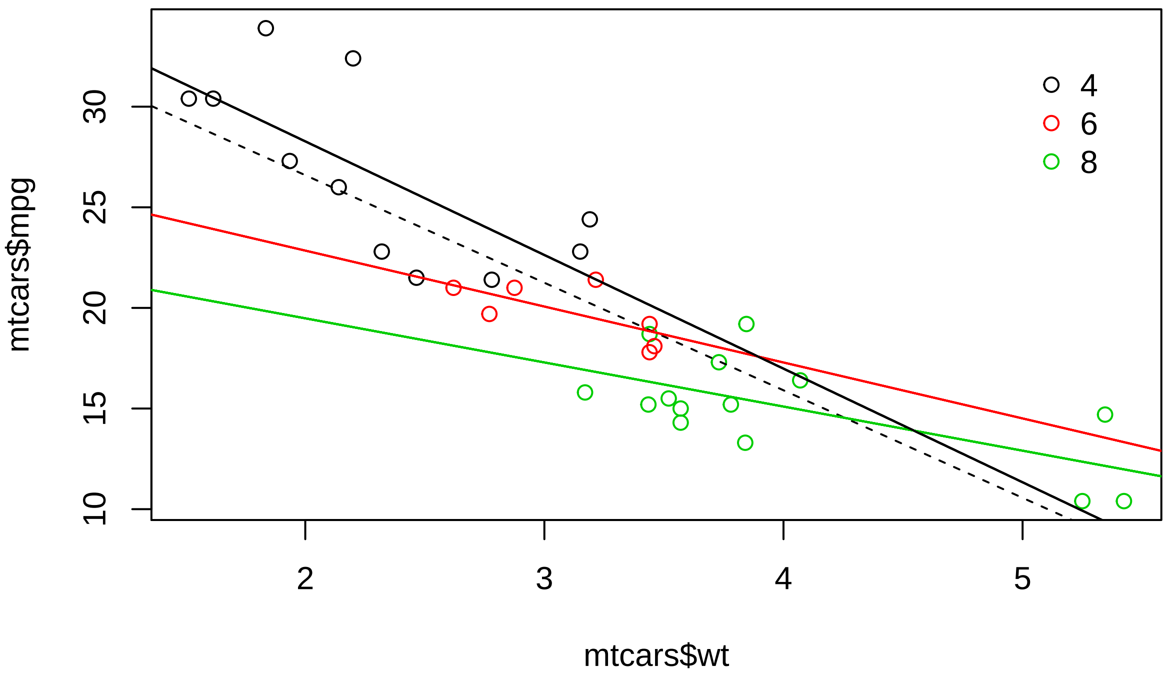
```
# Convert cyl to factor
mtcars$cyl <- as.factor(mtcars$cyl)

# Example from base R
plot(mtcars$wt, mtcars$mpg, col = mtcars$cyl)
abline(lm(mpg ~ wt, data = mtcars), lty = 2)

lapply(mtcars$cyl, function(x) {
  abline(lm(mpg ~ wt, mtcars, subset = (cyl == x)), col = x)
  })
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
```

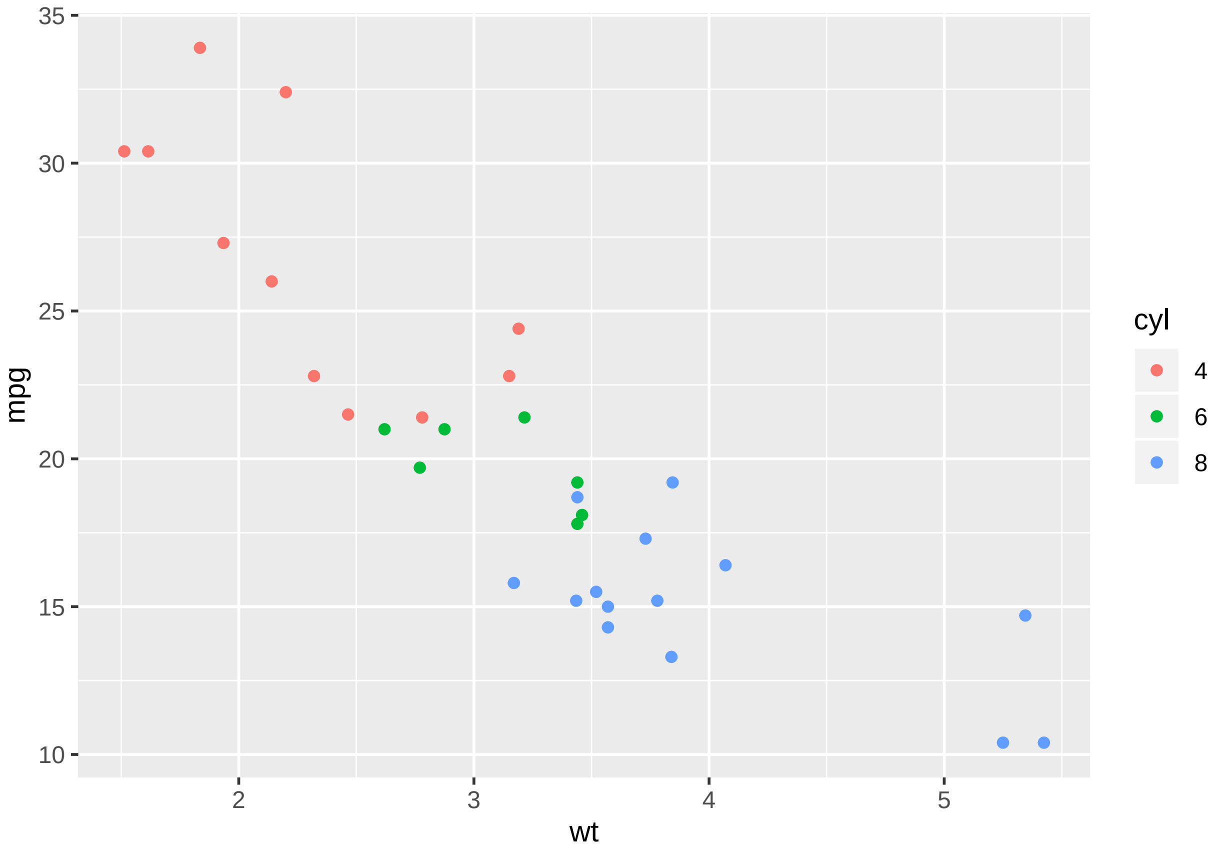
```
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
##
## [[19]]
## NULL
##
## [[20]]
## NULL
##
## [[21]]
## NULL
##
## [[22]]
## NULL
##
## [[23]]
## NULL
##
## [[24]]
## NULL
##
## [[25]]
## NULL
##
## [[26]]
## NULL
##
## [[27]]
## NULL
##
## [[28]]
## NULL
##
## [[29]]
## NULL
##
## [[30]]
## NULL
##
## [[31]]
## NULL
```

```
##
## [[32]]
## NULL
```

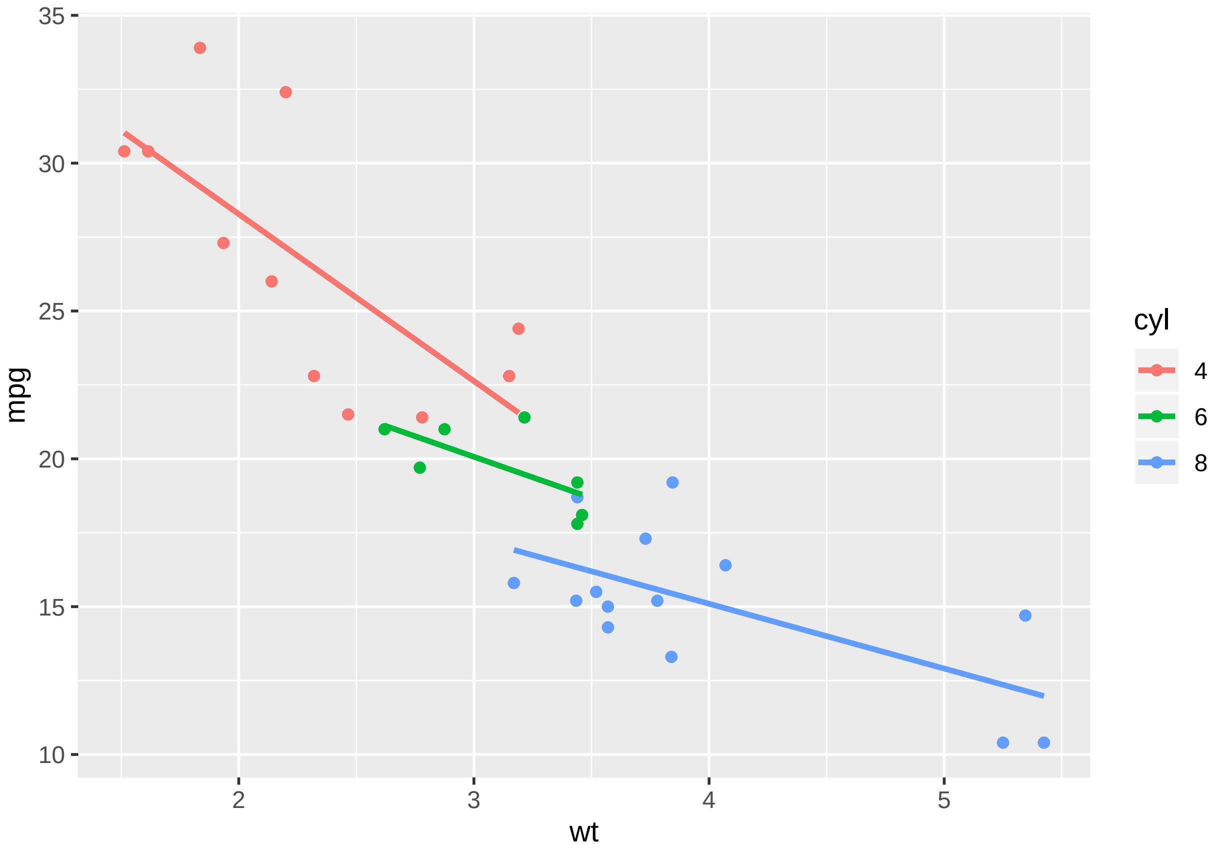
```
legend(x = 5, y = 33, legend = levels(mtcars$cyl),
       col = 1:3, pch = 1, bty = "n")
```



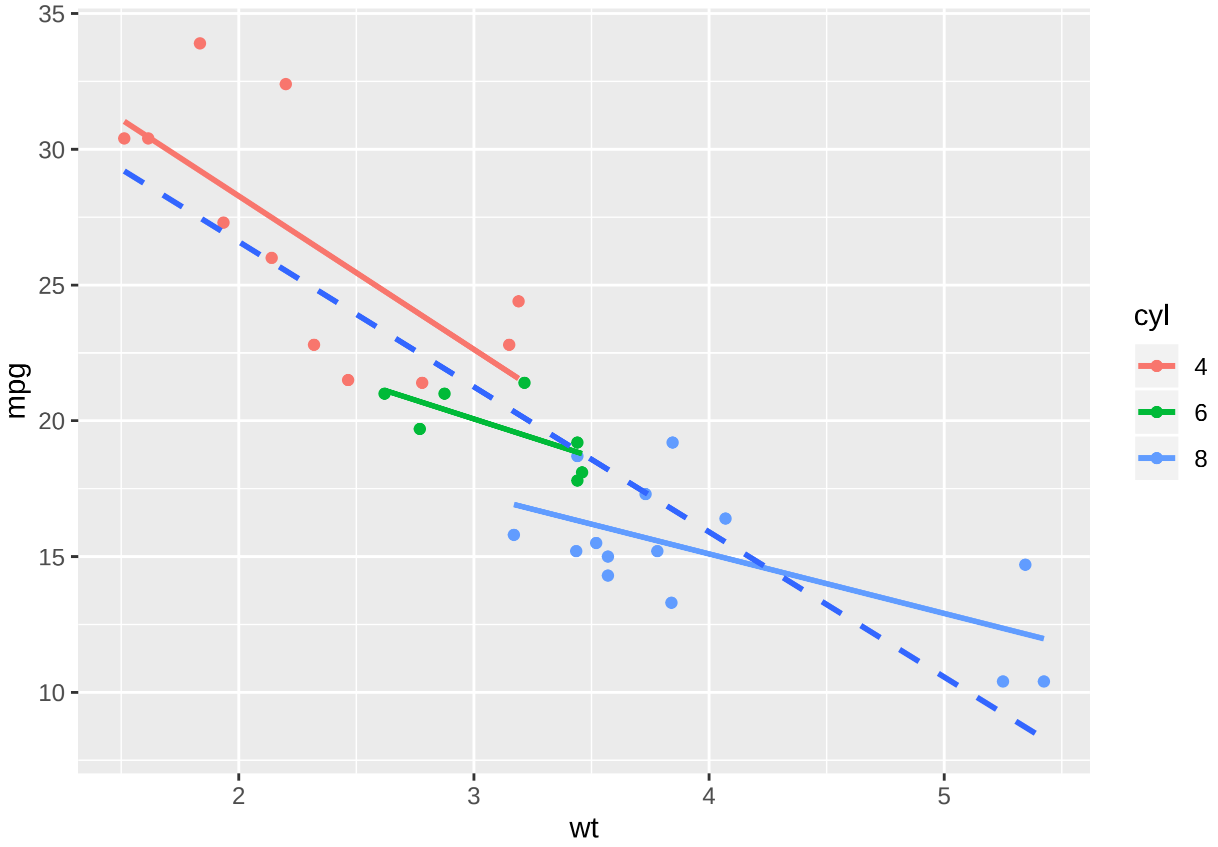
```
# Plot 1: add geom_point() to this command to create a scatter plot
ggplot(mtcars, aes(x = wt, y = mpg, col = cyl)) +
  geom_point()
```



```
# Plot 2: include the lines of the linear models, per cyl
ggplot(mtcars, aes(x = wt, y = mpg, col = cyl)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



```
# Plot 3: include a lm for the entire dataset in its whole
ggplot(mtcars, aes(x = wt, y = mpg, col = cyl)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(aes(group = 1), method = "lm", se = FALSE, linetype = 2)
```



Proper Data Format and Tidy Data

1. gather() rearranges the data frame by specifying the columns that are categorical variables with a - notation. In this case, Species and Flower are categorical.
2. separate() splits up the new key column, which contains the former headers, according to .. The new column names "Part" and "Measure" are given in a character vector.
3. use spread() to distribute the new Measure column and associated value column into two columns.

```
# Consider the structure of iris, iris.wide and iris.tidy
library(tidyr)
library(tidyverse)
```

```
## -- Attaching packages ------------------ tidyverse 1.2.1 --
```

```
## v tibble  2.1.3     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v purrr   0.3.2     v forcats 0.4.0
```

```
## -- Conflicts --------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
# head(iris)

# Add column with unique ids
iris$Flower <- 1:nrow(iris)

iris.wide <- iris %>%
  gather(key, value, -Flower, -Species) %>%
  separate(key, c("Part", "Measure"), "\\.") %>%
  spread(Measure, value)

iris.tidy <- iris %>%
  gather(key, Value, -Flower, -Species) %>%
  separate(key, c("Part", "Measure"), "\\.")

str(iris)
```

```
## 'data.frame':    150 obs. of  6 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Flower      : int  1 2 3 4 5 6 7 8 9 10 ...
```

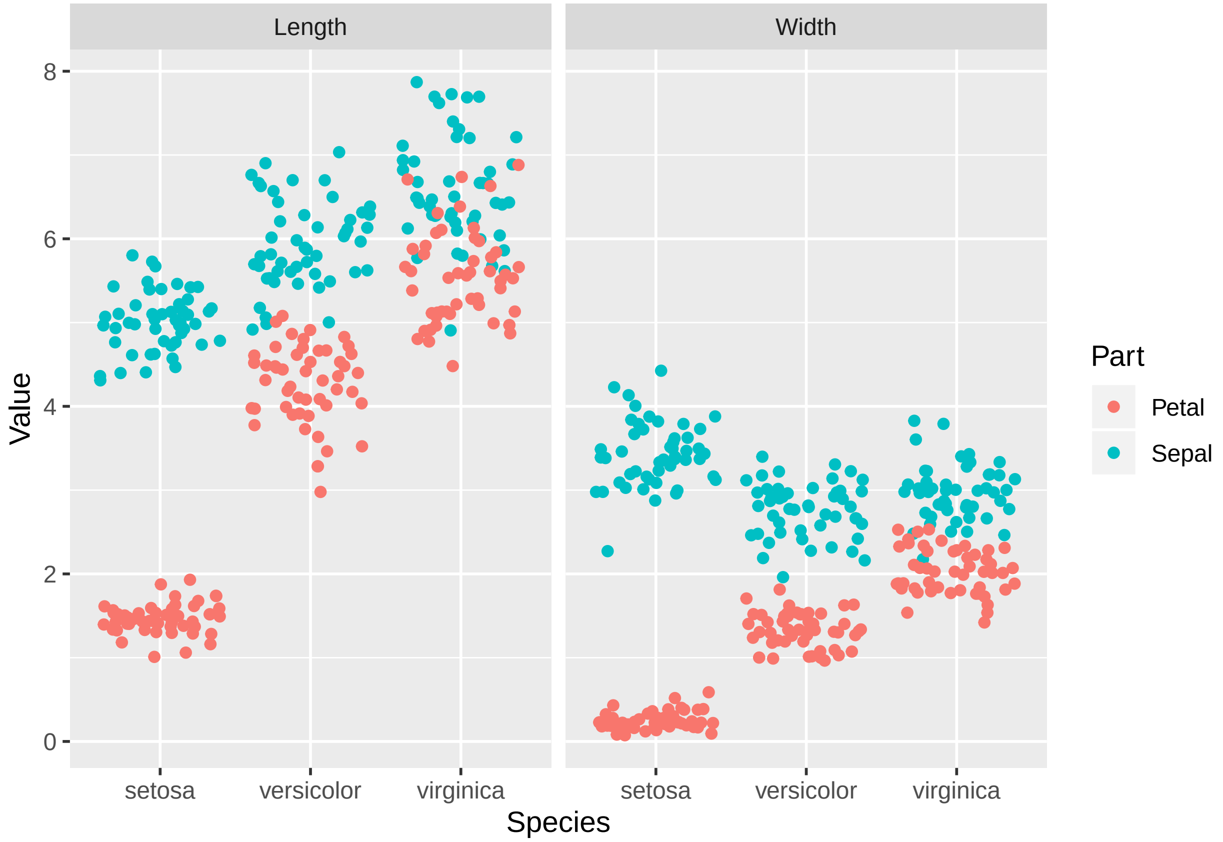
```r
str(iris.wide)
```

```
## 'data.frame':    300 obs. of  5 variables:
##  $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Flower : int  1 1 2 2 3 3 4 4 5 5 ...
##  $ Part   : chr  "Petal" "Sepal" "Petal" "Sepal" ...
##  $ Length : num  1.4 5.1 1.4 4.9 1.3 4.7 1.5 4.6 1.4 5 ...
##  $ Width  : num  0.2 3.5 0.2 3 0.2 3.2 0.2 3.1 0.2 3.6 ...
```

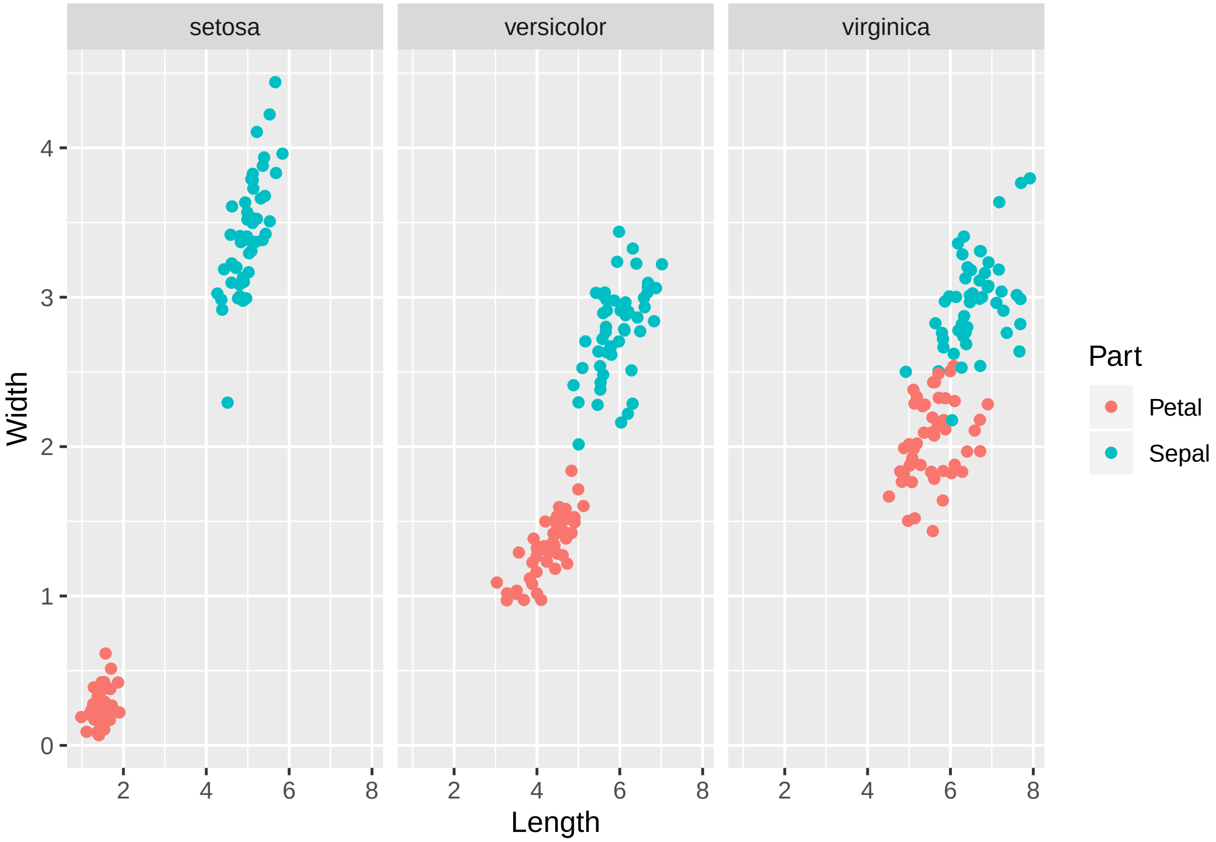
```r
str(iris.tidy)
```

```
## 'data.frame':    600 obs. of  5 variables:
##  $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Flower : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Part   : chr  "Sepal" "Sepal" "Sepal" "Sepal" ...
##  $ Measure: chr  "Length" "Length" "Length" "Length" ...
##  $ Value  : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

```r
# Think about which dataset you would use to get the plot shown right
ggplot(iris.tidy, aes(x = Species, y = Value, col = Part)) +
  geom_jitter() +
  facet_grid(. ~ Measure)
```



```r
# Think about which dataset you would use to get the plot shown right
ggplot(iris.wide, aes(x = Length, y = Width, color = Part)) +
  geom_jitter() +
  facet_grid(. ~ Species)
```


setosa
versicolor
virginica
Width
Length
0
1
2
3
4
2
4
6
8
Part
● Petal
● Sepal