Datacamp_Data Visualization with ggplot2 (Part 2)_Themes

dizhen

2019/4/11

Themes

1. Themes Layer

- All the non-data ink
- Visual elements not part of data
- Three types

text: element_text()

line: element_line()

rectangle: element_rect()

```
# Inheritance
theme(
  text = element_text()
    title
      plot.title
      legend.title
    axis.title
      axis.title.x
      axis.title.y
    legend.text
    axis.text
      axis.text.x
      axis.text.y
    strip.text
      strip.text.x
      strip.text.y
)

theme(
  line = element_line()
    axis.ticks
      axis.ticks.x
      axis.ticks.y
    axis.line
      axis.line.x
      axis.line.y
    panel.grid
      panel.grid.major
        panel.grid.major.x
        panel.grid.major.y
```

```
      panel.grid.minor
        panel.grid.minor.x
        panel.grid.minor.y
)

theme(
  rect = element_rect()
    legend.background
    legend.key
    panel.background
    panel.border
    plot.background
    strip.background
)
```

2. element_blank

```
theme(
  text = element_blank()
  line = element_blank()
  rect = element_blank()
  )
```

Practice

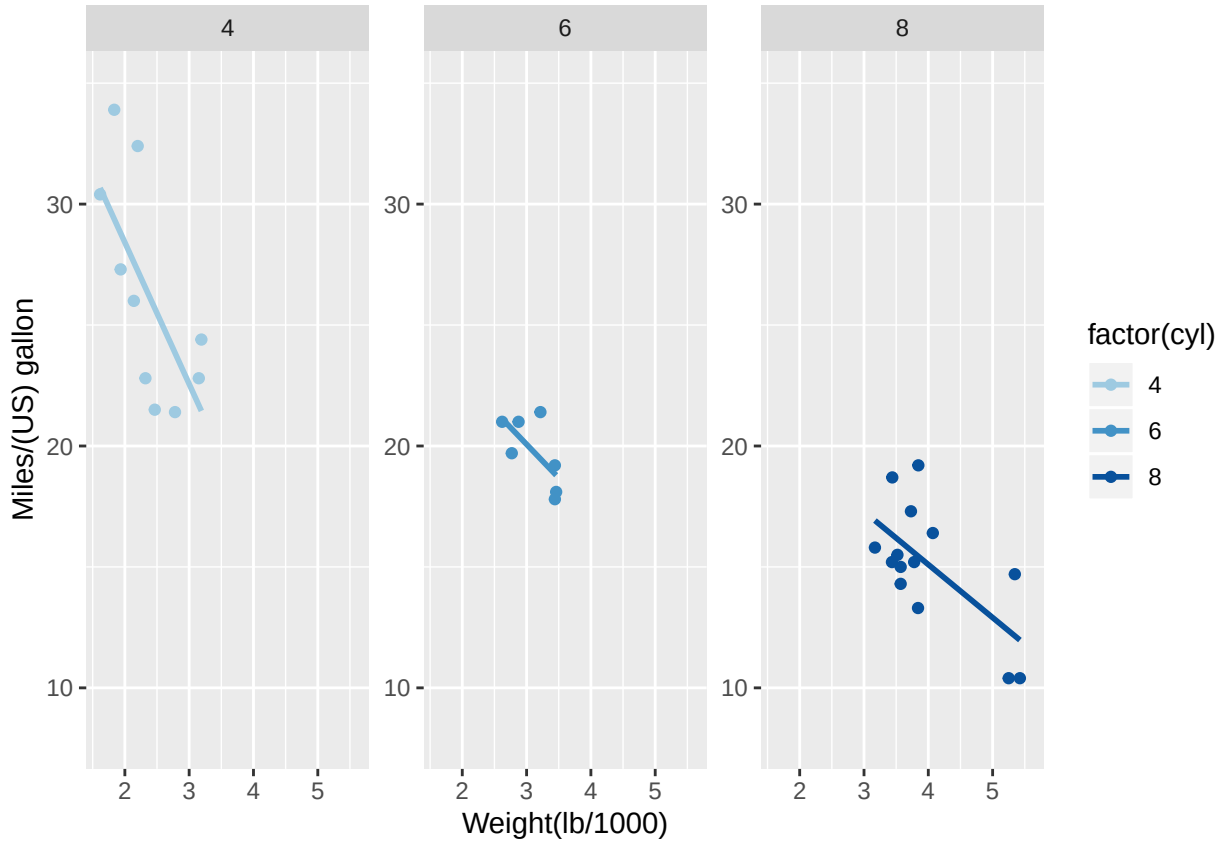
```
# Basic scatter plot
library("ggplot2")
library("RColorBrewer")

mycol <- brewer.pal(9,"Blues")[c(4,6,8)]

z<-ggplot(mtcars, aes(x = wt, y = mpg, col = factor(cyl))) +
  geom_point() +
  facet_wrap(~ cyl,scale = "free_y") +
  scale_color_brewer("Cylinders") +
  scale_color_manual(values = mycol) +
  geom_smooth(se = FALSE, method = "lm") +
  scale_y_continuous("Miles/(US) gallon", limits = c(8,35)) +
  scale_x_continuous("Weight(lb/1000)", limits = c(1.6,5.6))
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

```
z
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

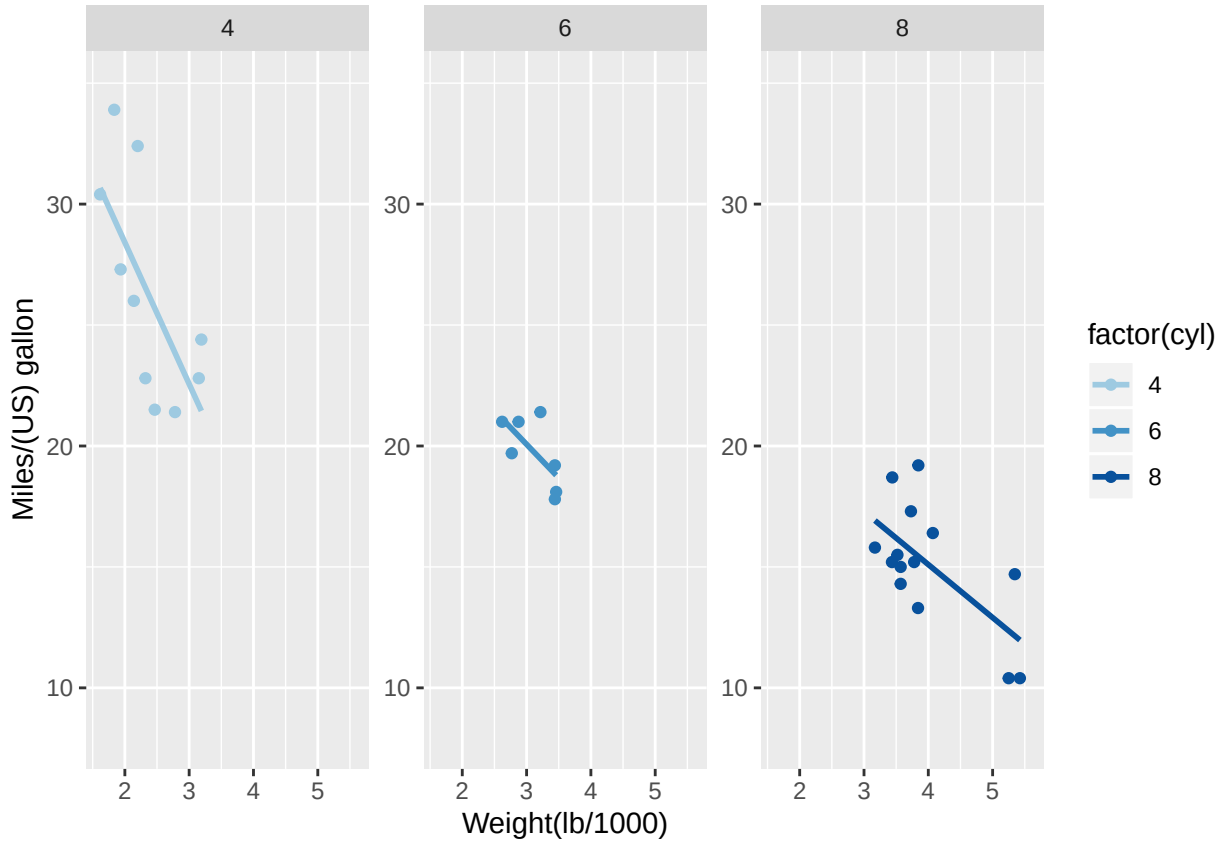


Rectangle

```
# Starting point
z
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

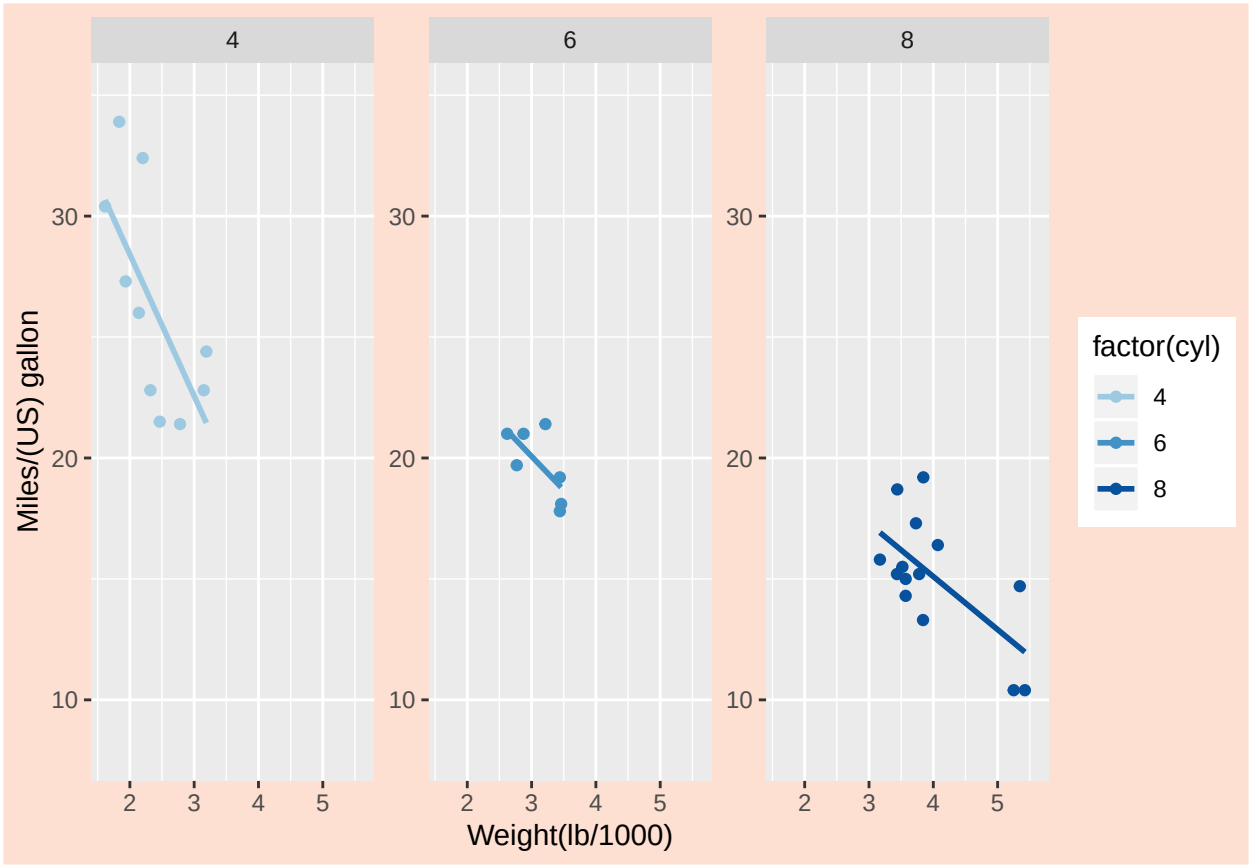
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
myPink <- "#FEE0D2"
# Plot 1: Change the plot background fill to myPink
z +
  theme(plot.background = element_rect(fill = myPink))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

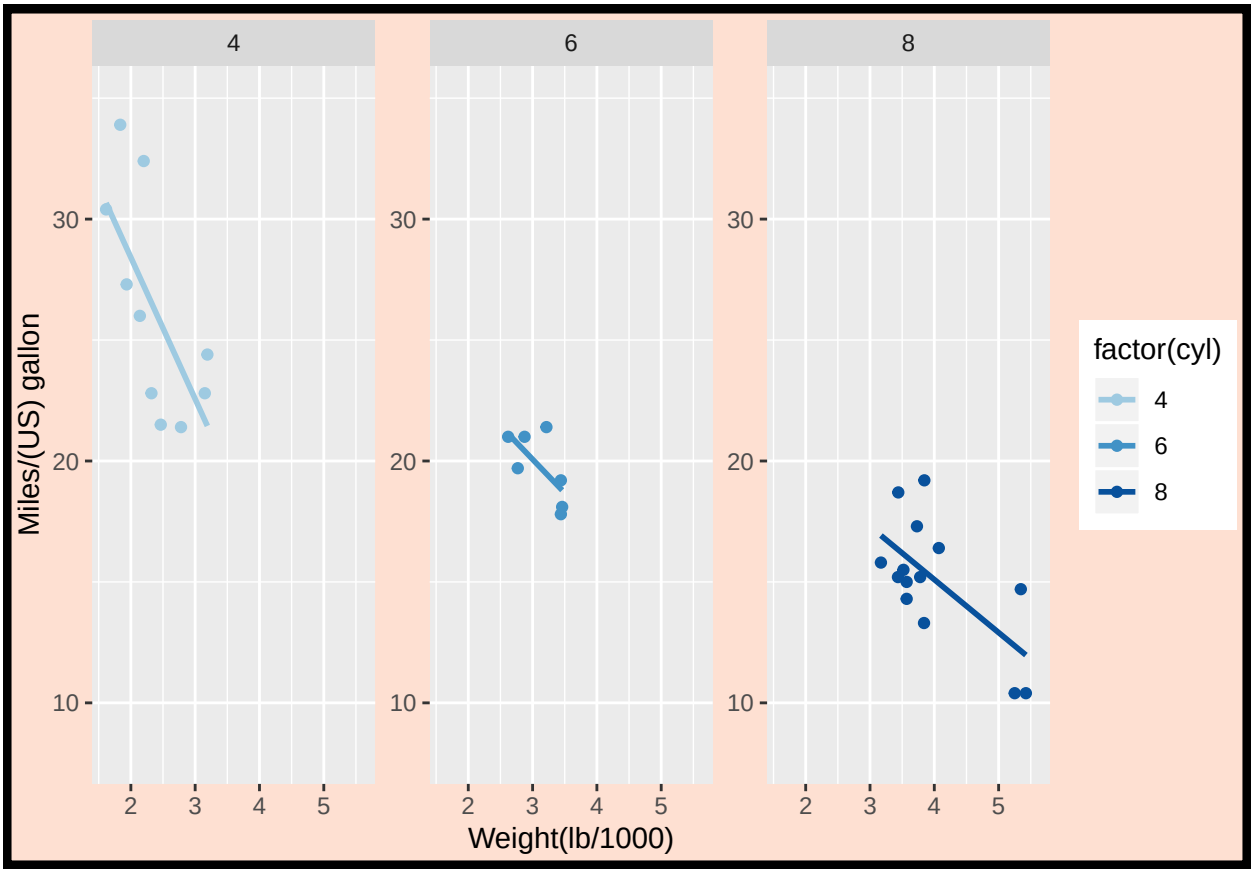
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Plot 2: Adjust the border to be a black line of size 3
z +
  theme(plot.background = element_rect(fill = myPink,color = "black", size = 3)) # expanded from plot 1
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Theme to remove all rectangles
no_panels <- theme(rect = element_blank())

# Plot 3: Combine custom themes
z<-z +
  no_panels +
  theme(plot.background = element_rect(fill = myPink,color = "black", size = 3)) # from plot 2
z
```

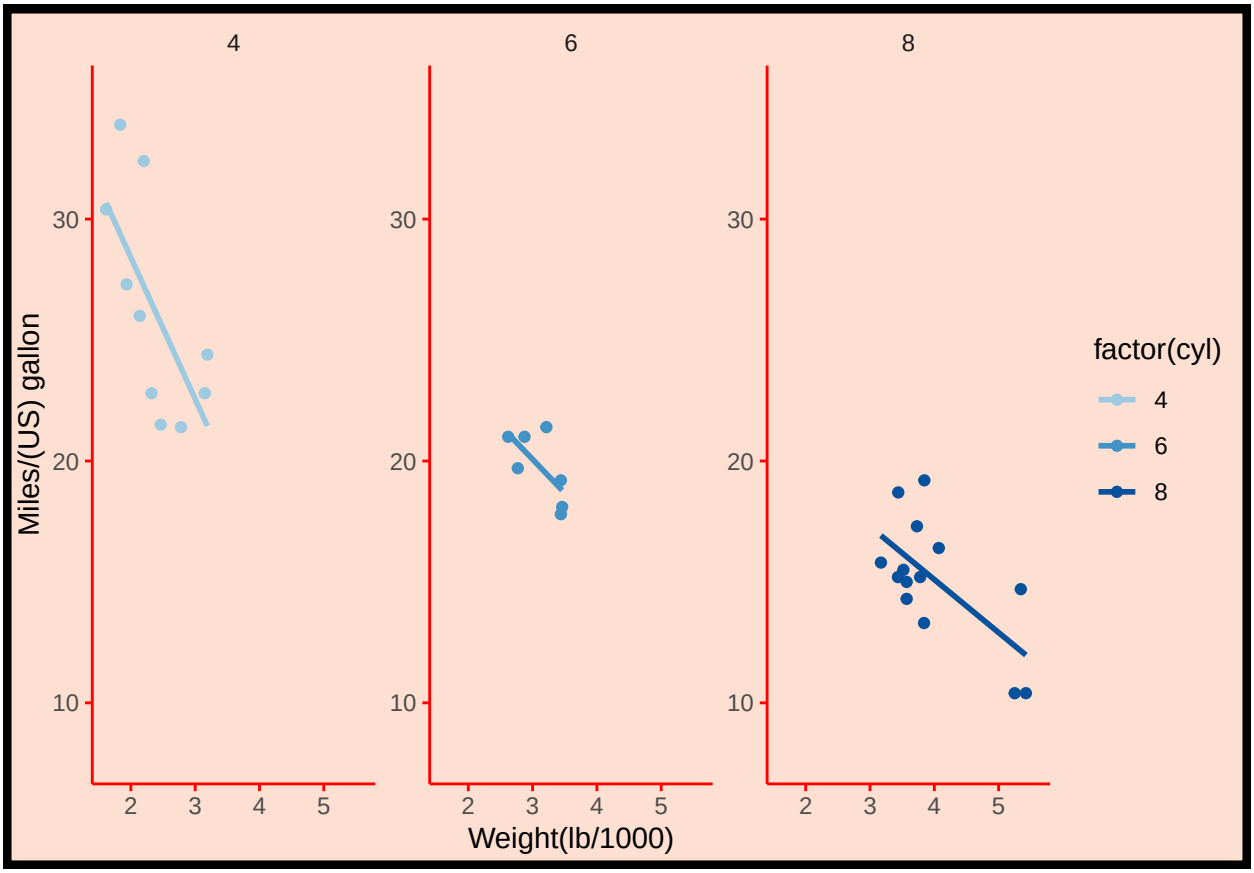
```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```

Line

```
# Extend z using theme() function and 3 args
z<- z + theme(panel.grid = element_blank(),
          axis.line = element_line(color = "red"),
          axis.ticks = element_line(color = "red"))
z
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



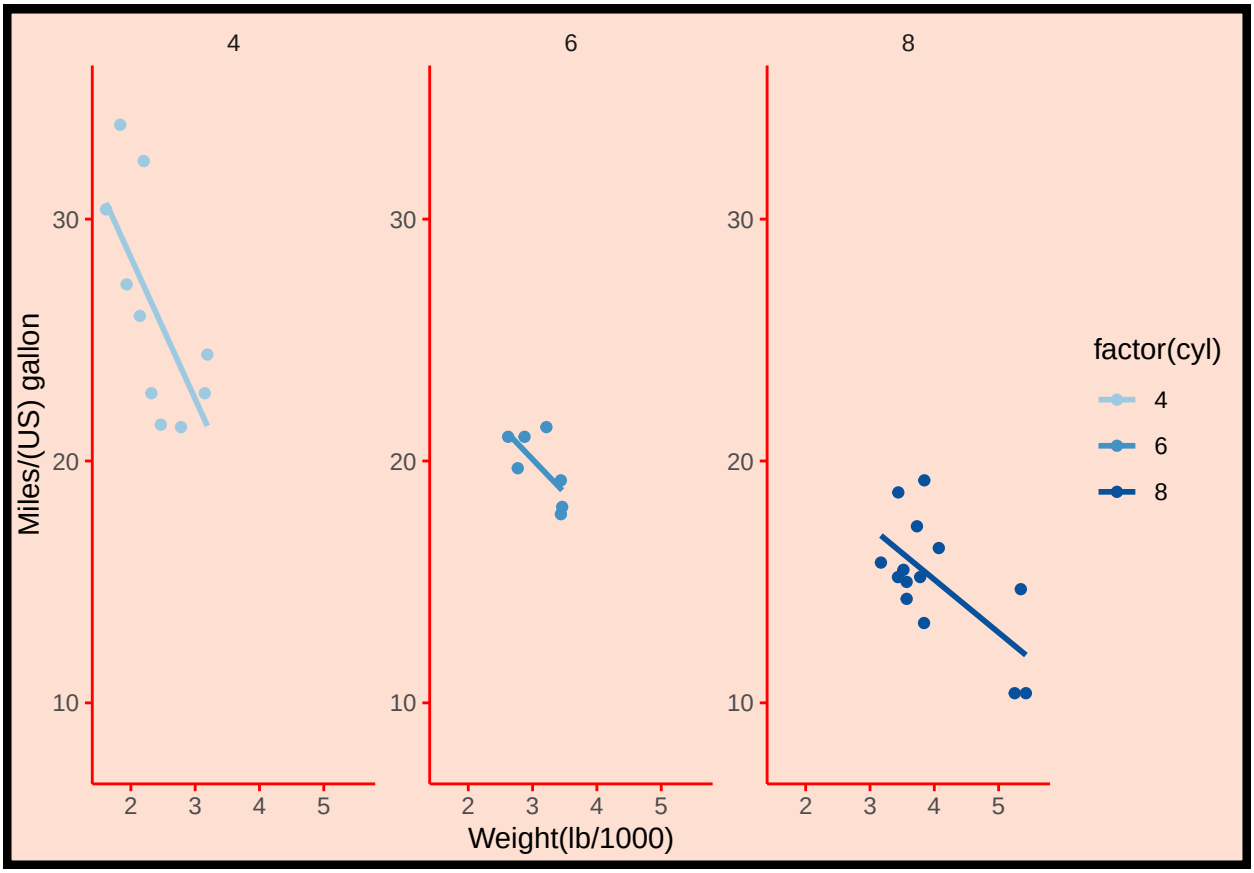
Text

1. Change the appearance of the strip text, that is the text in the facet strips. Specify strip.text with element_text()
2. Change the axis titles. Specify both axes with the axis.title argument and use element_text() to set the parameters: color = myRed, hjust = 0 (to put the text in the bottom left corner) and face = "italic".
3. Make the axis text black using the axis.text argument to do so.

```
# Original plot, color provided
z
```

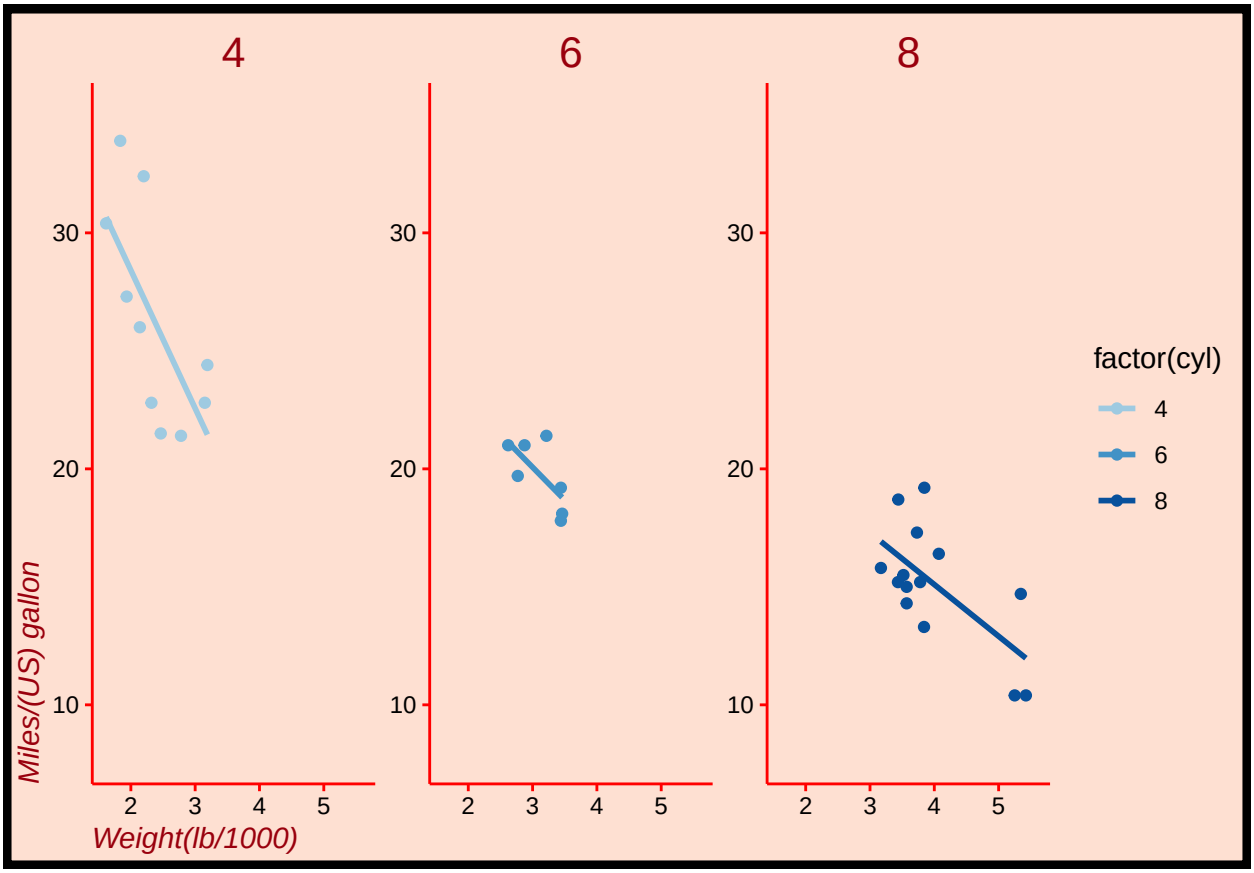
```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
myRed <- "#99000D"

# Extend z with theme() function and 3 args
z<- z +
  theme(strip.text = element_text(size = 16, color = myRed),
        axis.title = element_text(color = myRed, hjust = 0, face = "italic"),
        axis.text = element_text(color = "black"))
z
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```



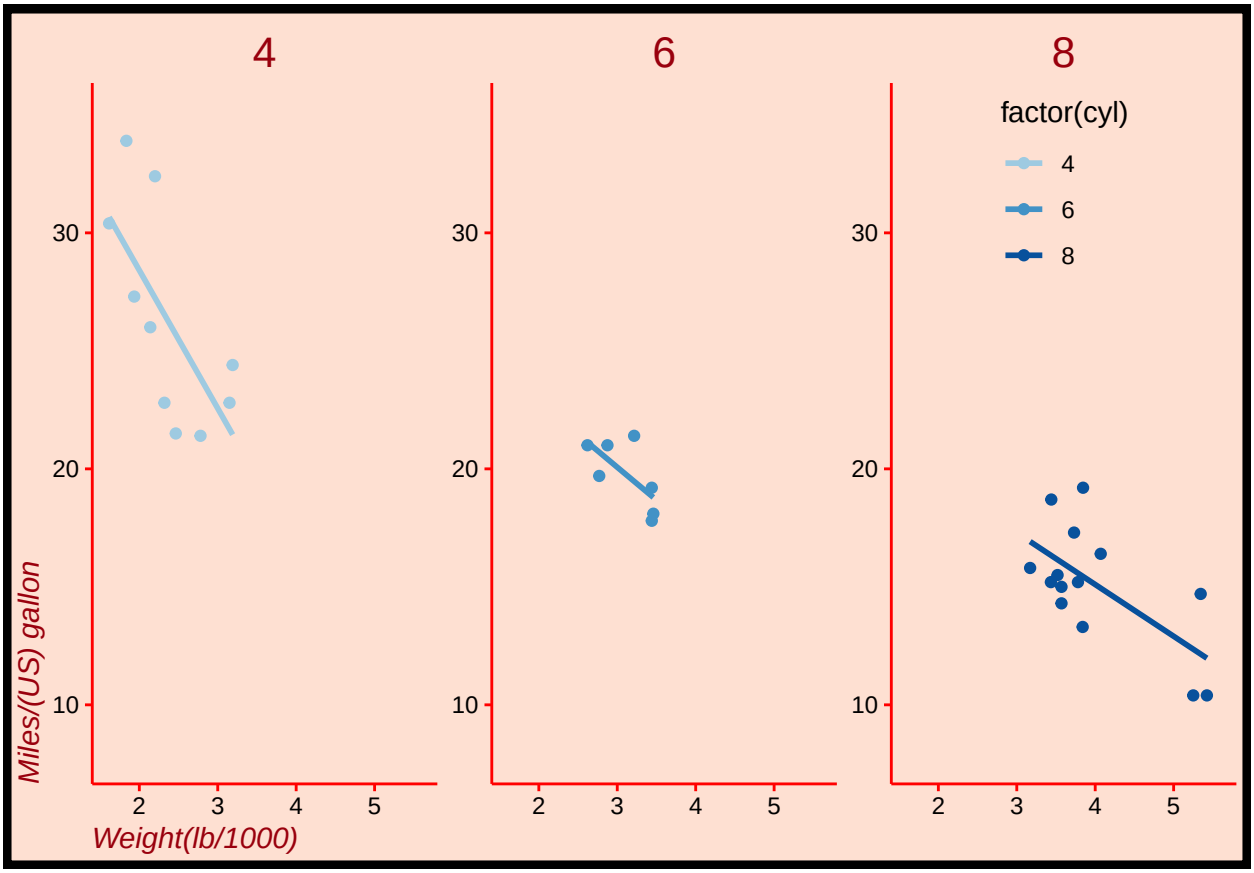
Legends

1. Add a theme() function to z to change the legend's location. Do this by specifying the legend.position argument to be c(0.85, 0.85). This will make the legend appear in the top right of the plot, inside the third facet.
2. Instead of a vertical list of legend entries, you might want to have the different entries next to each other. Starting from z, add a theme() function in which you specify legend.direction to be "horizontal".
3. You can also change the locations of legends by name: set legend.position to "bottom".
4. Finally, you can remove the legend entirely, by setting legend.position to "none".

```
# Move legend by position
z +
  theme(legend.position = c(0.85, 0.85))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

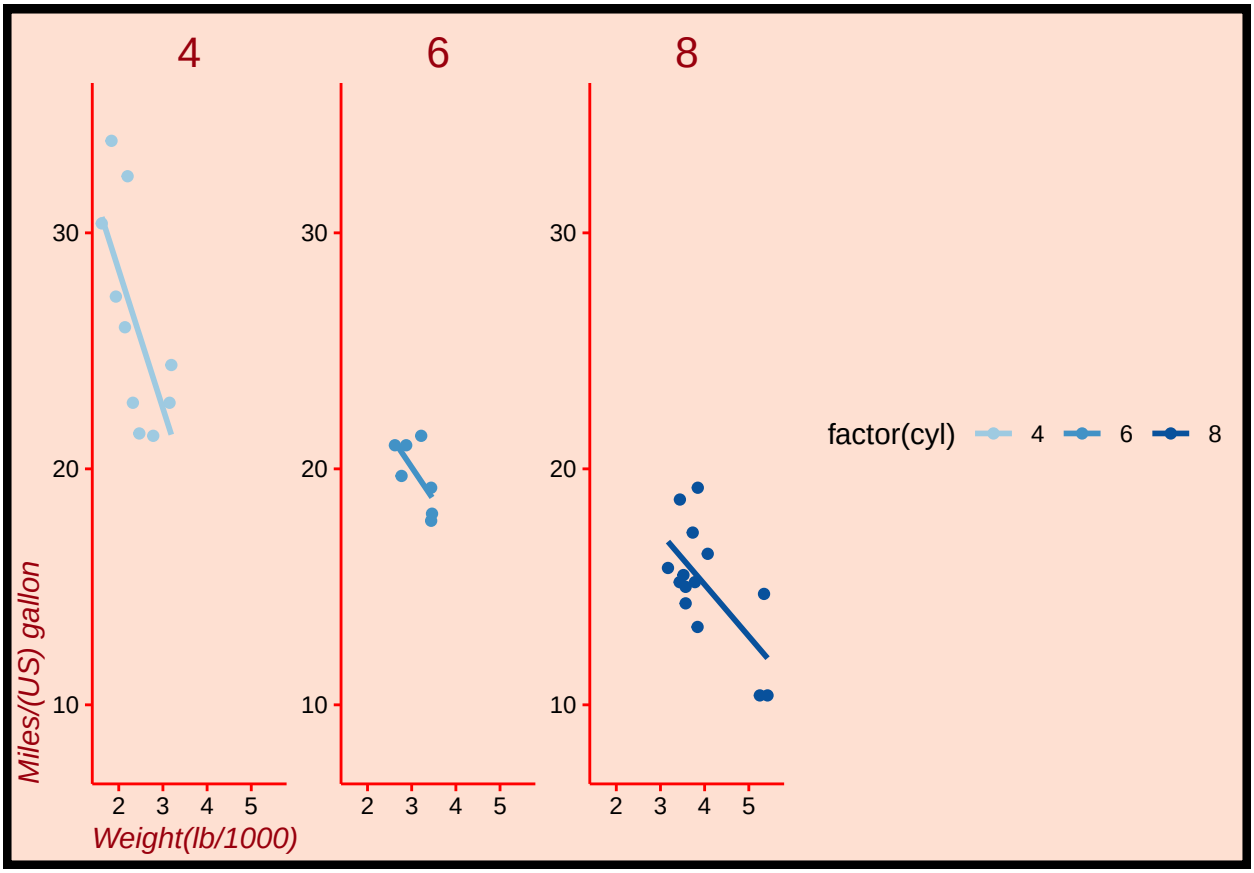
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Change direction
z +
  theme(legend.direction = "horizontal")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

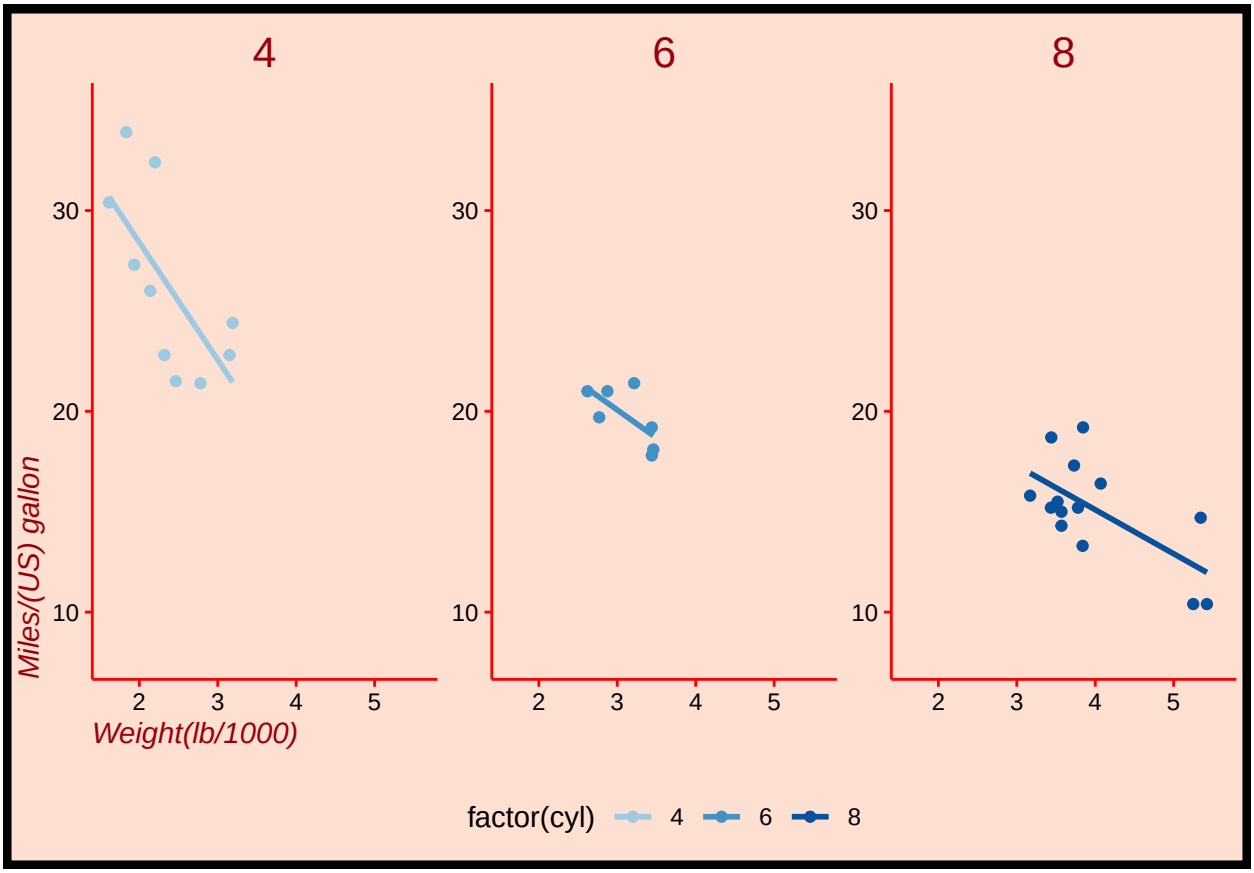
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Change location by name
z +
  theme(legend.position = "bottom")
```

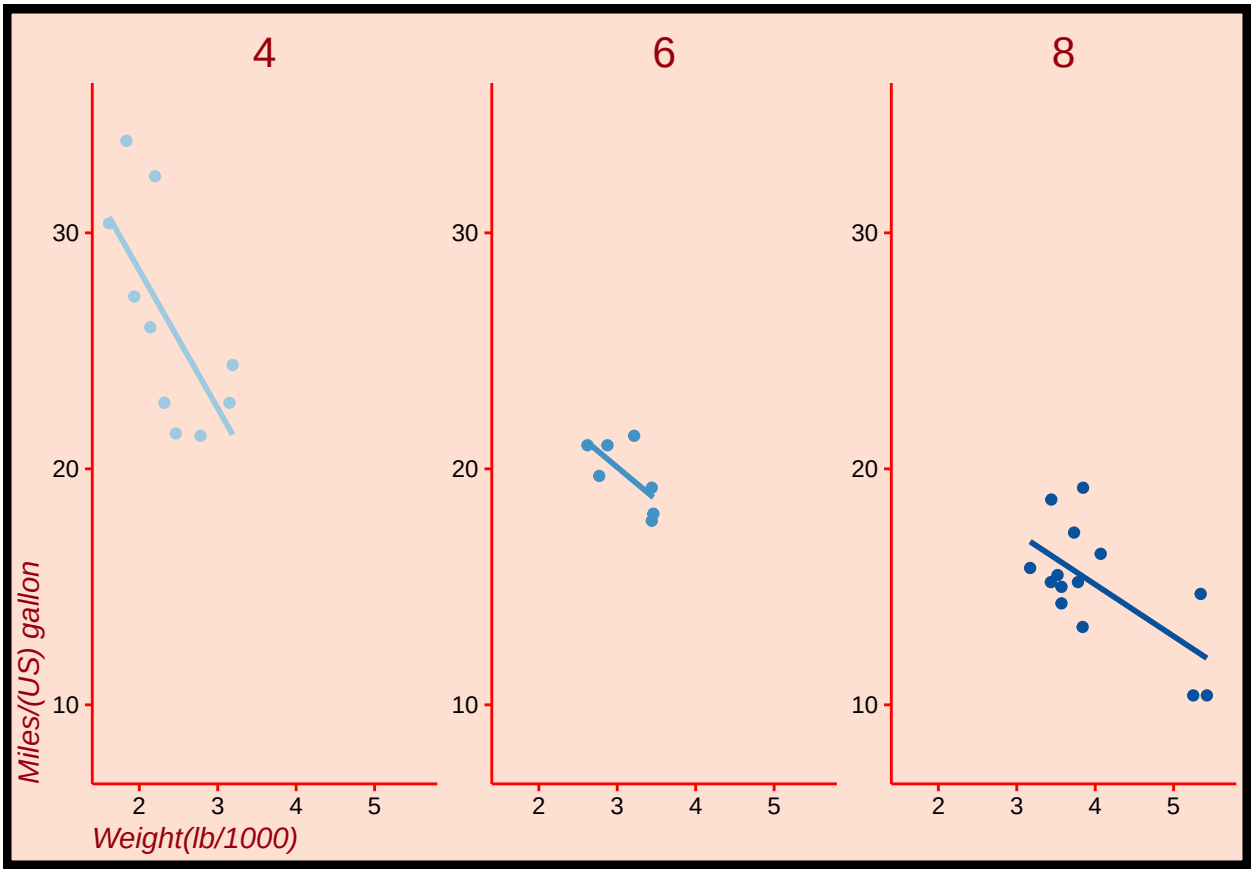
```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Remove legend entirely
z<- z +
  theme(legend.position = "none")
z
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```



Position

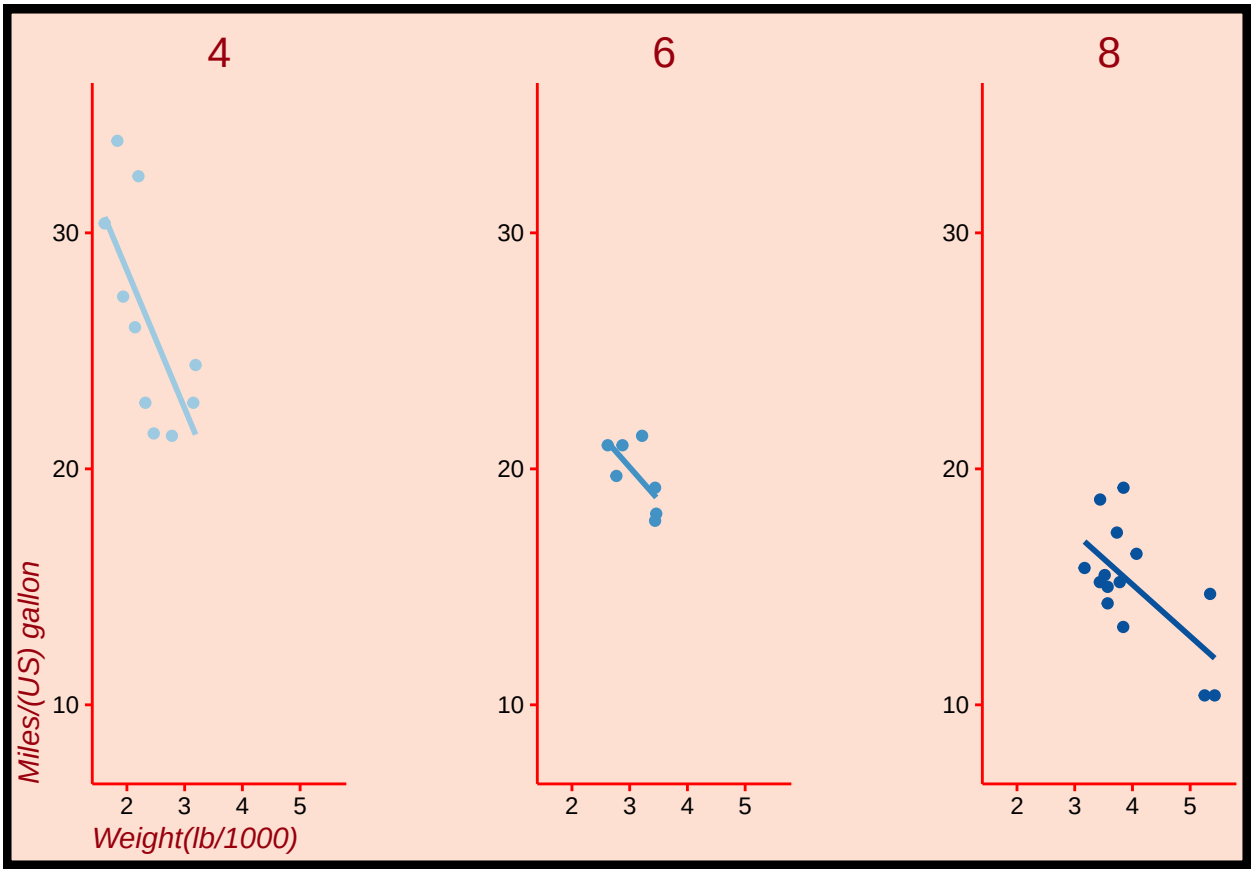
The different rectangles of your plot have spacing between them. There's spacing between the facets, between the axis labels and the plot rectangle, between the plot rectangle and the entire panel background, etc.

1. Suppose you want to have more spacing between the different facets. You can control this by specifying panel.spacing.x inside a theme() function you add to z. For the argument value, you should pass a unit object.
2. To adjust the plot margin, set plot.margin to unit(c(1,2,1,1), "cm") (spacing for top, right, bottom, and left margins).

```
# Increase spacing between facets
library(grid)
z + theme(panel.spacing.x = unit(2,"cm"))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

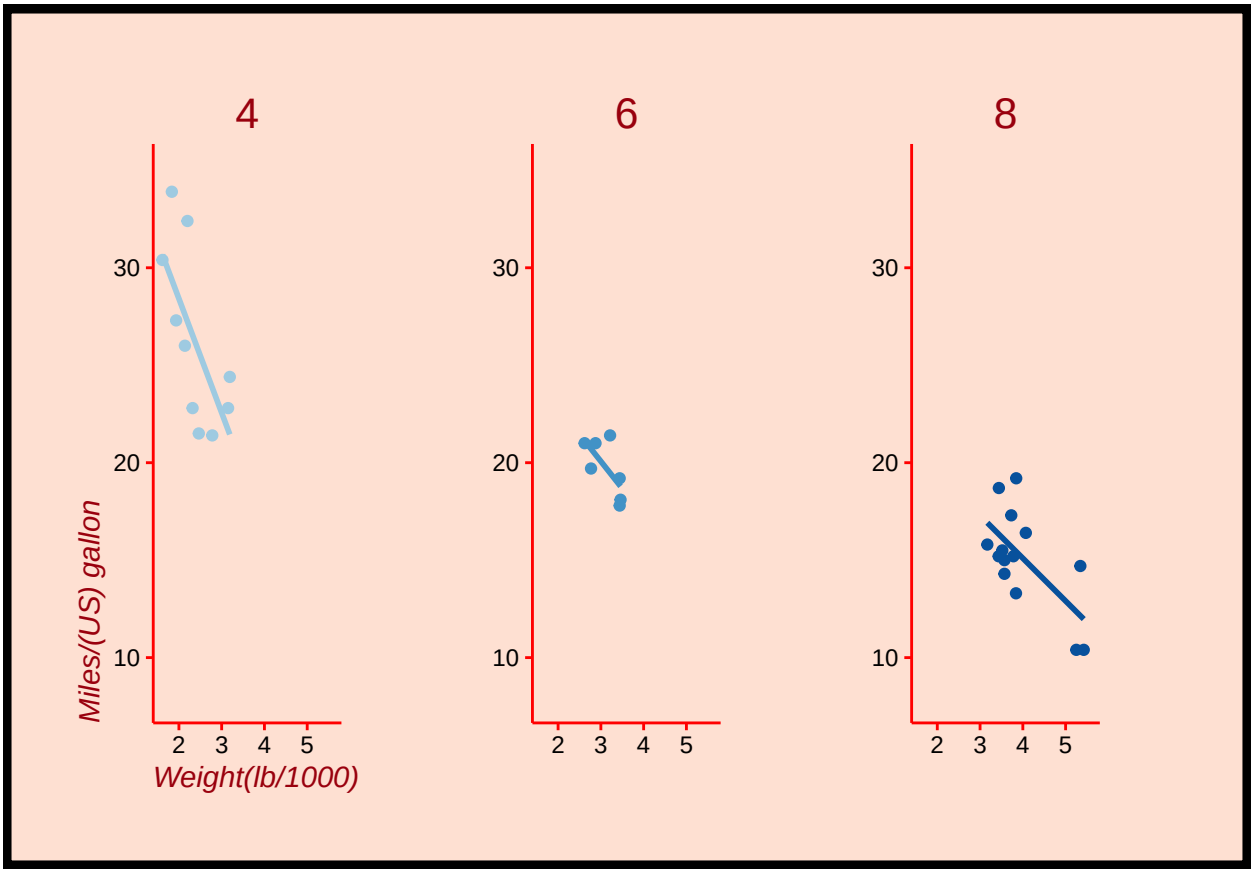
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Adjust the plot margin
z + theme(panel.spacing.x = unit(2,"cm"), plot.margin = unit(c(1,2,1,1),"cm"))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

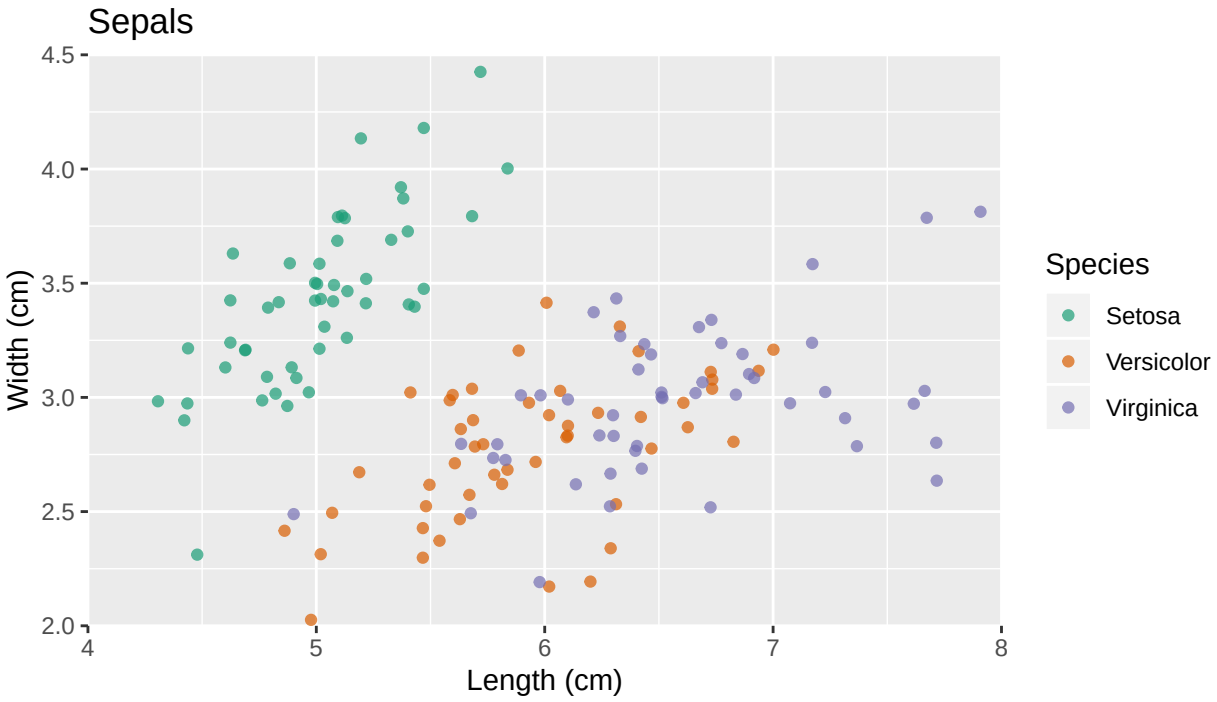
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



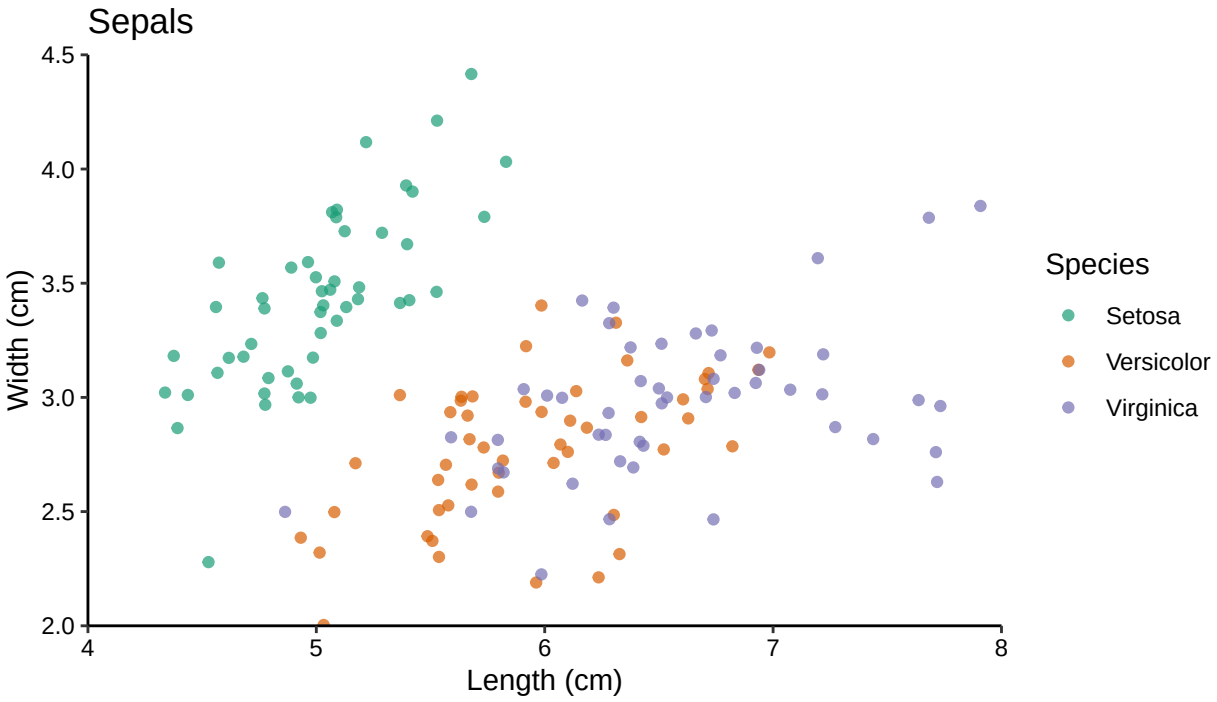
Recycling Themes

- Many plots
- Consistency in style
- Apply specific theme everywhere

```
z <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
  geom_jitter(alpha = 0.7) +
  scale_color_brewer("Species", palette = "Dark2", labels = c("Setosa", "Versicolor", "Virginica")) +
  scale_y_continuous("Width (cm)", limits = c(2, 4.5), expand = c(0, 0)) +
  scale_x_continuous("Length (cm)", limits = c(4, 8), expand = c(0, 0)) +
  ggtitle("Sepals") +
  coord_fixed(1)
z
```

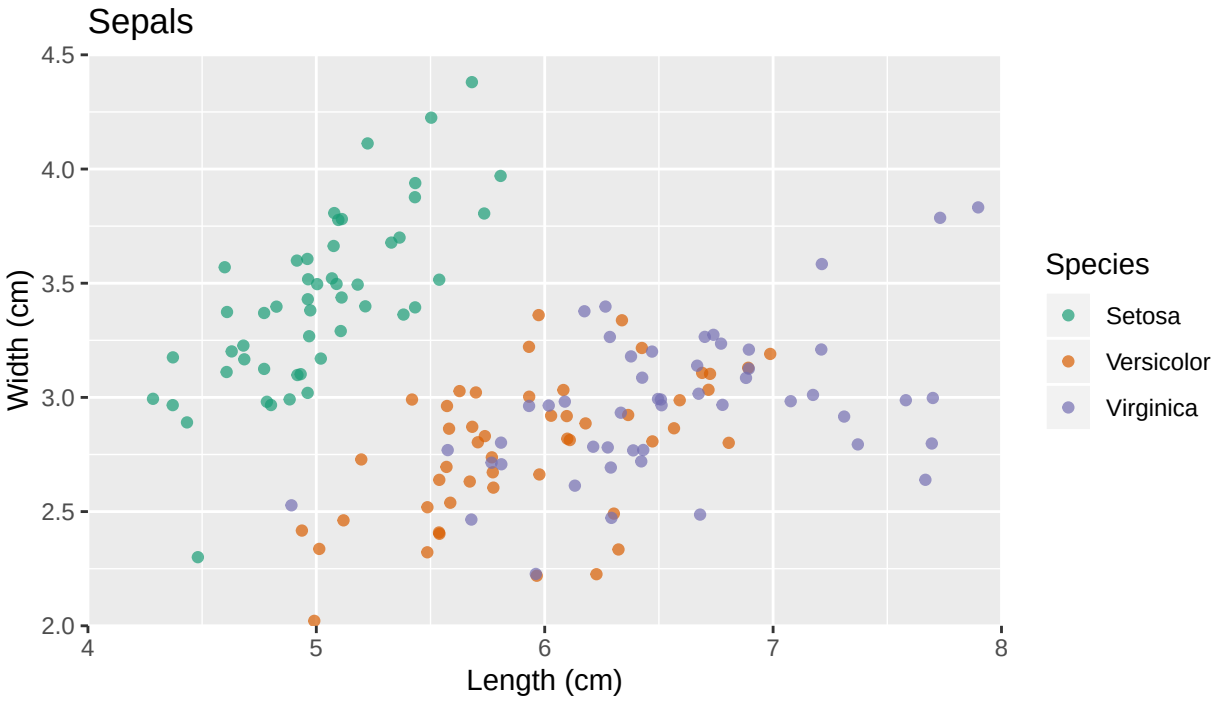



```
z + theme(panel.background = element_blank(),
          legend.background = element_blank(),
          legend.key = element_blank(),
          panel.grid = element_blank(),
          axis.text = element_text(colour = "black"),
          axis.line = element_line(colour = "black"))
```



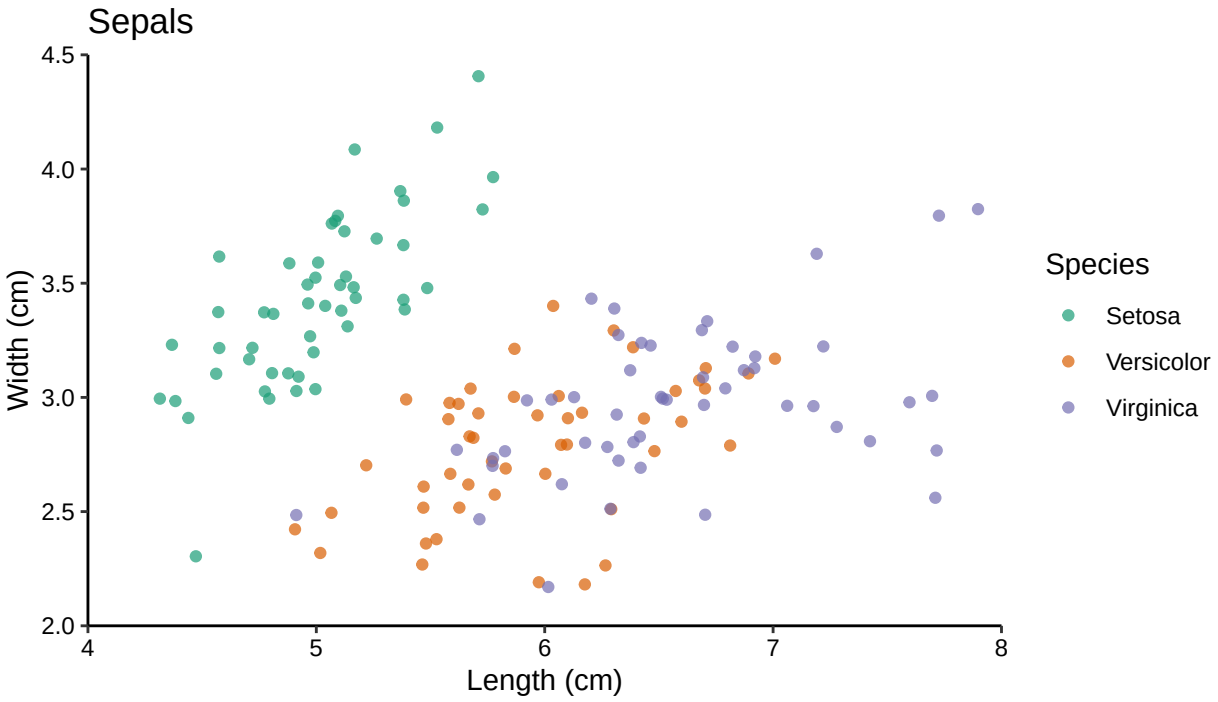
Save theme

```
theme_iris <- theme(panel.background = element_blank(),
                    legend.background = element_blank(),
                    legend.key = element_blank(),
                    panel.grid = element_blank(),
                    axis.text = element_text(colour = "black"),
                    axis.line = element_line(colour = "black"))
z
```




```
z+ theme_iris
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

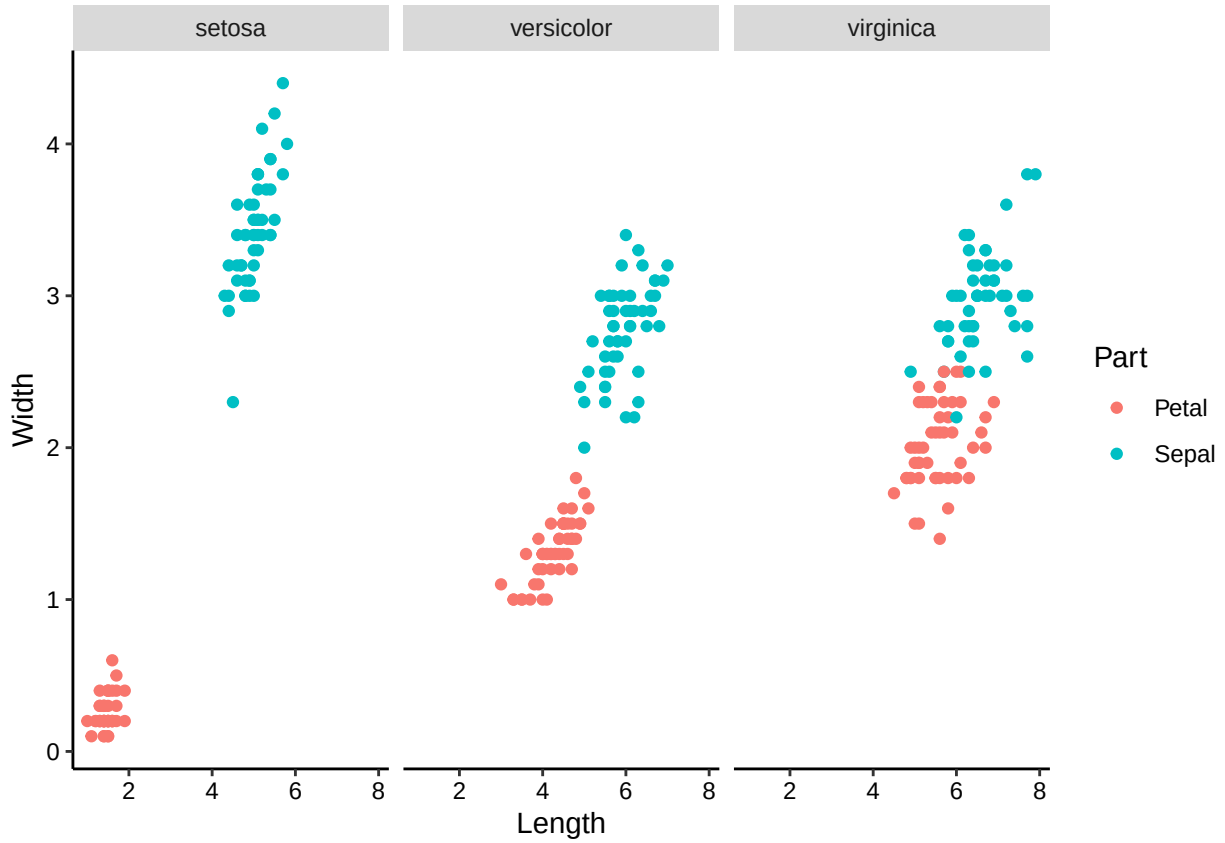


Reuse theme

```
library(tidyr)
library(tidyverse)
```

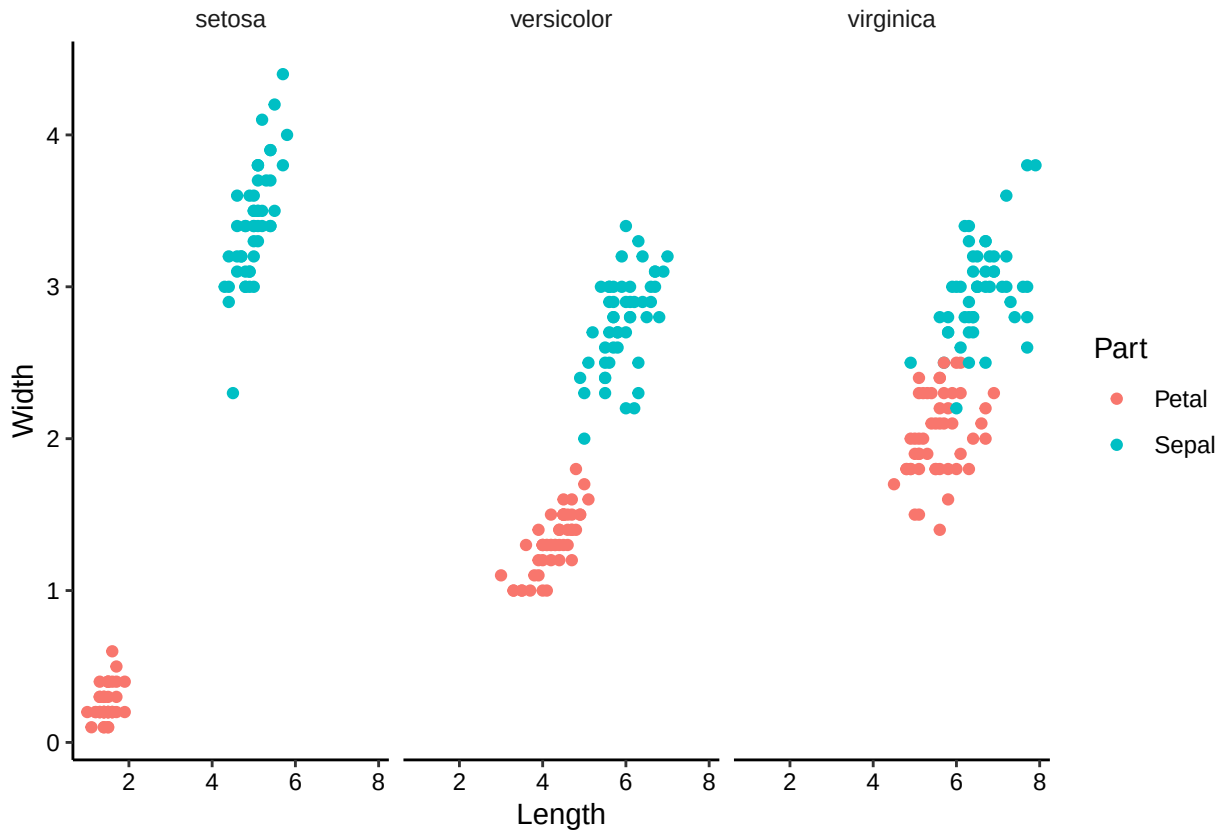
```
## -- Attaching packages ------------

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

```
iris$Flower <- 1:nrow(iris)

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

m <- ggplot(iris.wide, aes(x = Length, y = Width, col = Part)) +
  geom_point() +
  facet_grid(. ~ Species)
m + theme_iris
```



Extend theme

```
theme_iris <- theme_iris + theme(strip.background = element_blank())
m + theme_iris
```

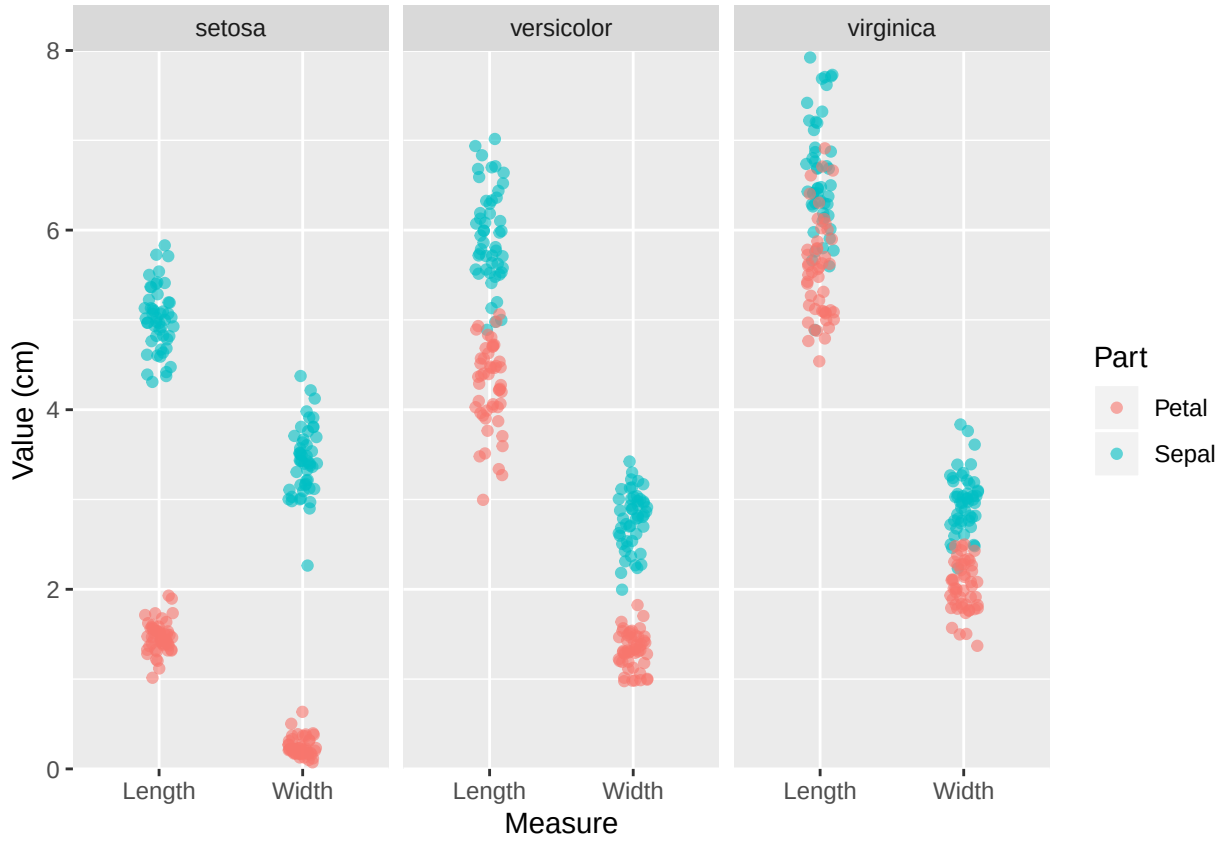


Discrete x-axis

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

p <- ggplot(iris.tidy, aes(x = Measure, y = Value, col = Part)) +
  geom_point(position = position_jitter(0.1), alpha = 0.6,
  width = 0.4) +
  scale_y_continuous("Value (cm)", limits = c(0, 8),
  expand = c(0, 0)) +
  facet_grid(. ~ Species)
```

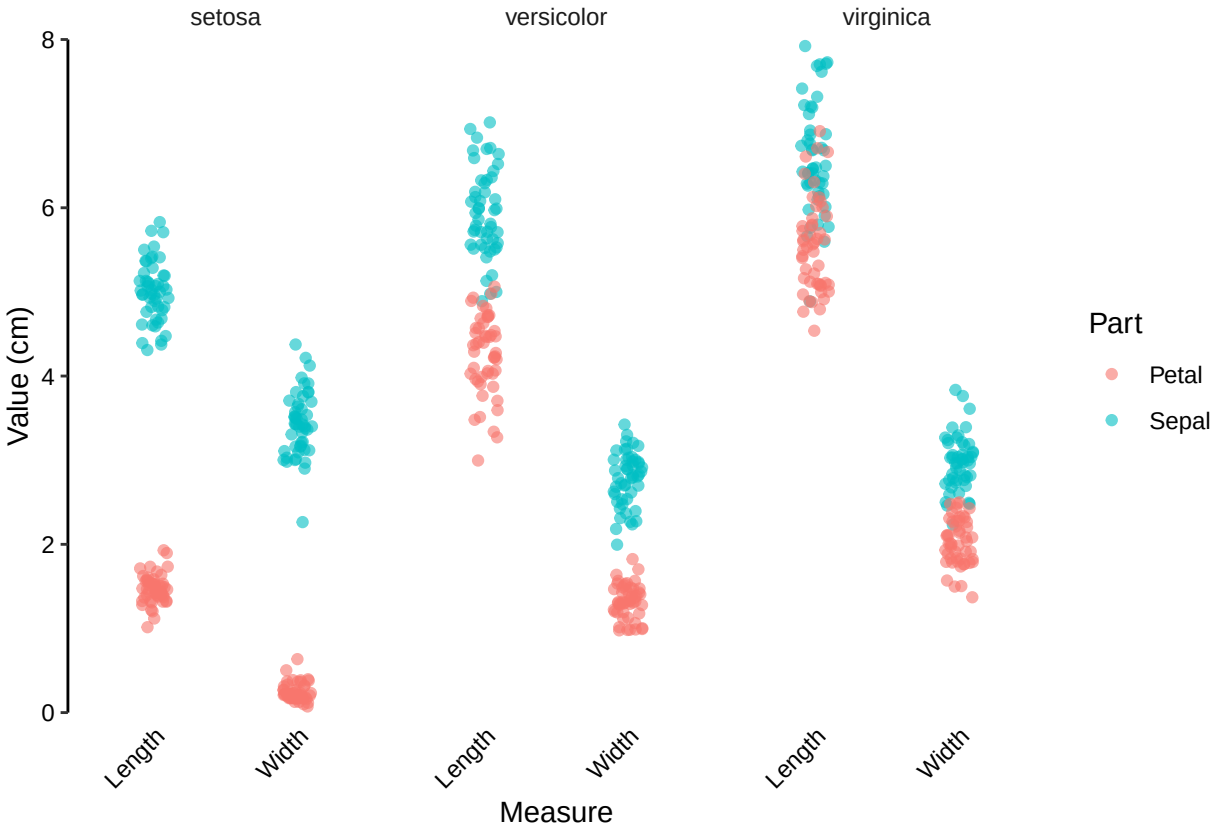
```
## Warning: Ignoring unknown parameters: width
```

```
p
```



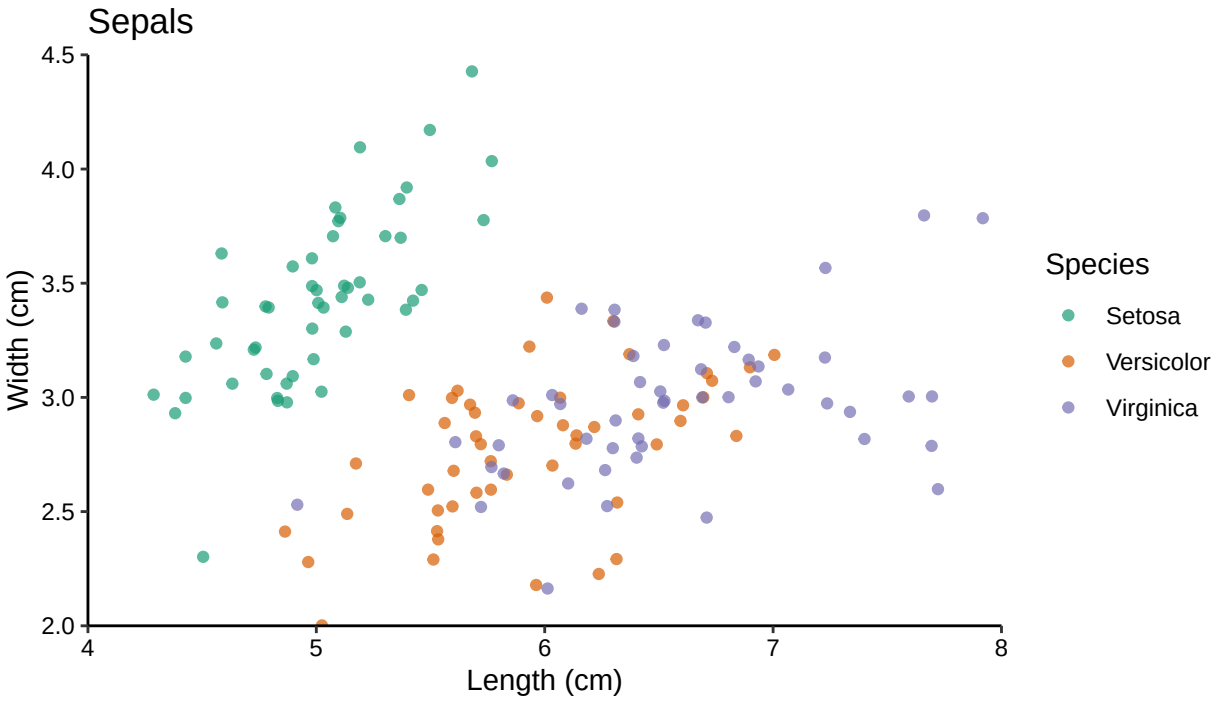
Derivative theme

```
theme_iris_disX <- theme_iris +
  theme(axis.line.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1))

p + theme_iris_disX
```

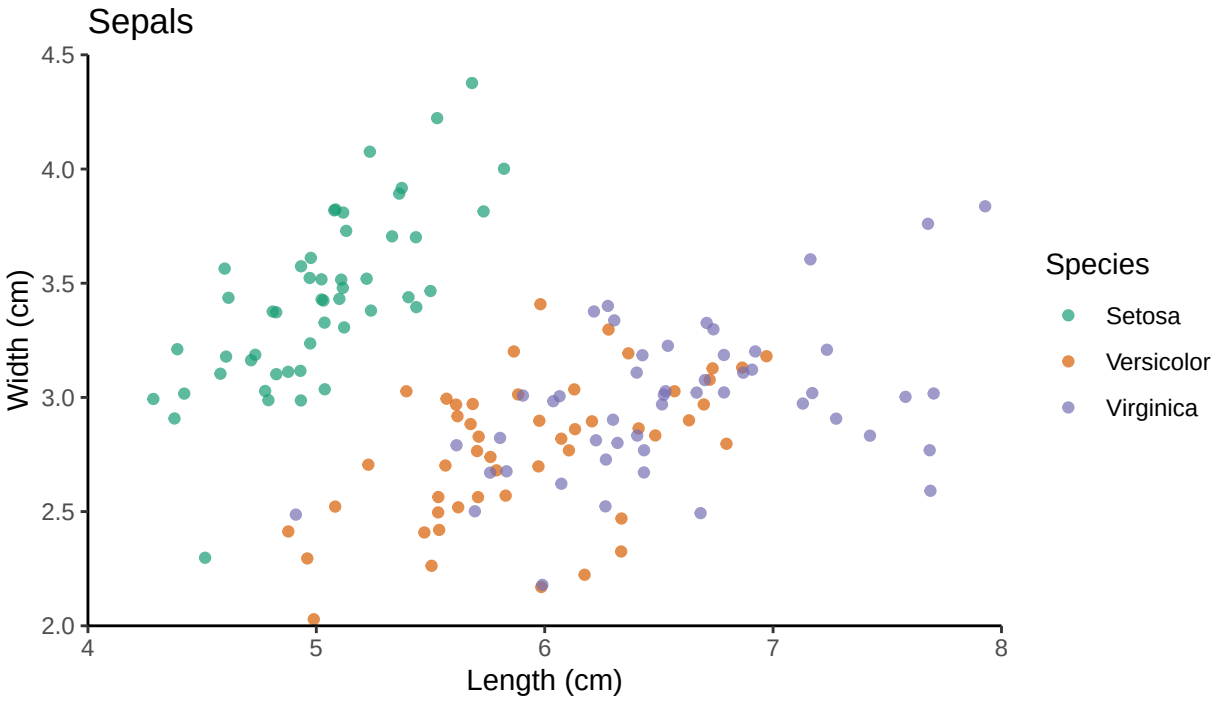


Built-in theme templates

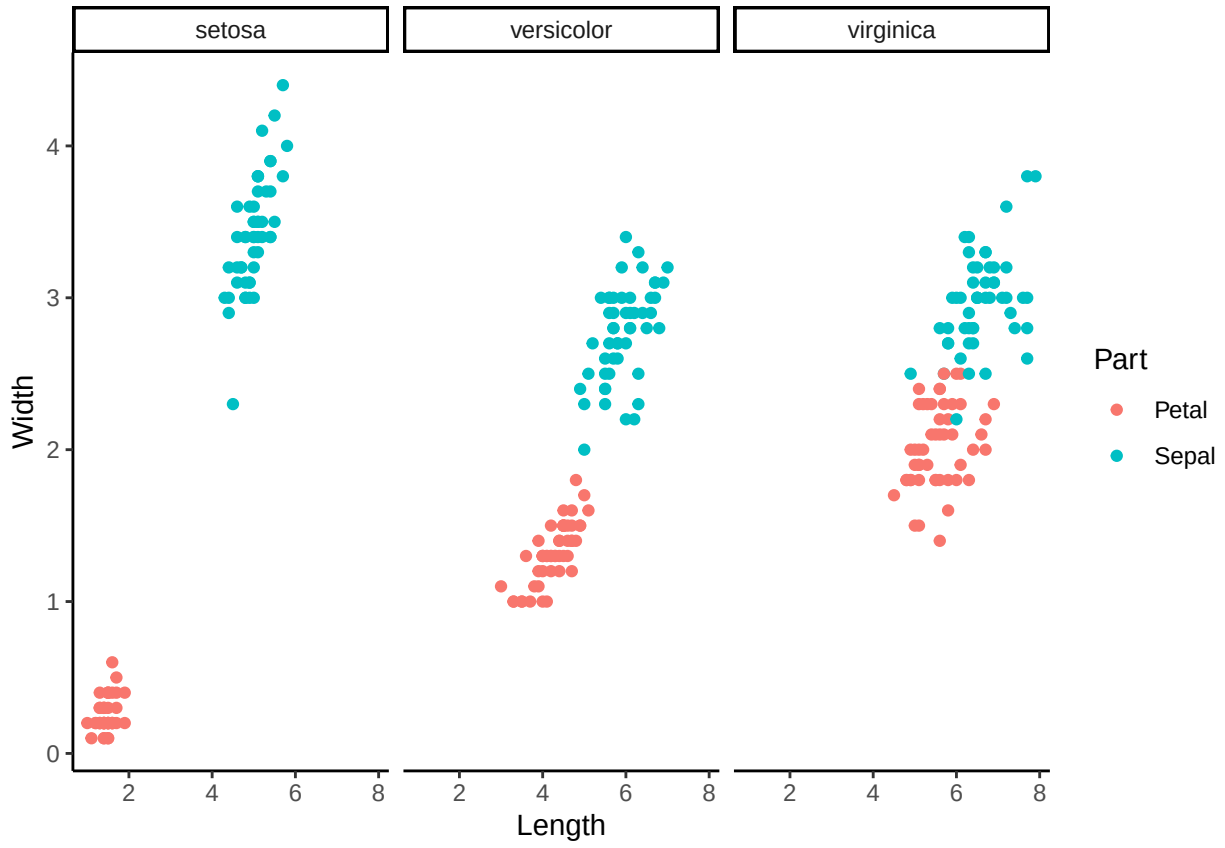
```
z + theme_iris
```



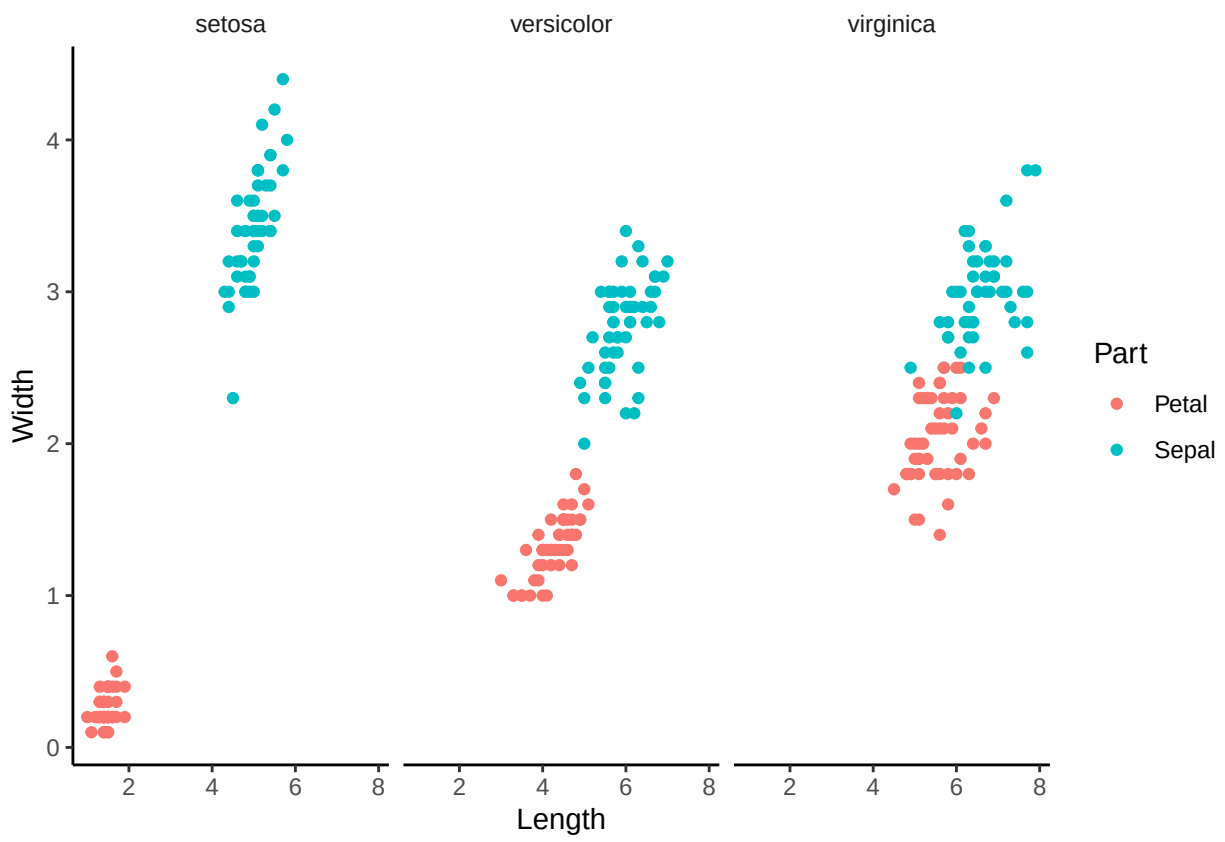

```
z + theme_classic()
```




```
m + theme_classic()
```




```
m + theme_classic() +
  theme(strip.background = element_blank())
```

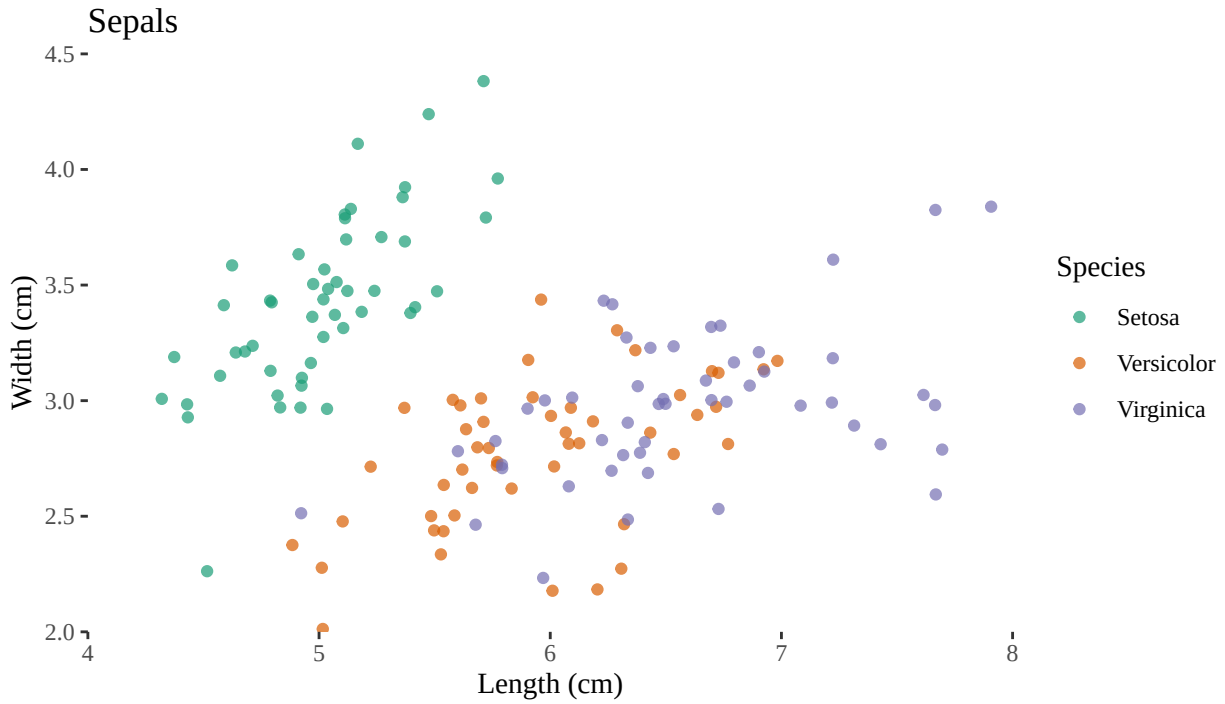


ggthemes

```
# install.packages("ggthemes")
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 3.6.3
```

```
z + theme_tufte()
```



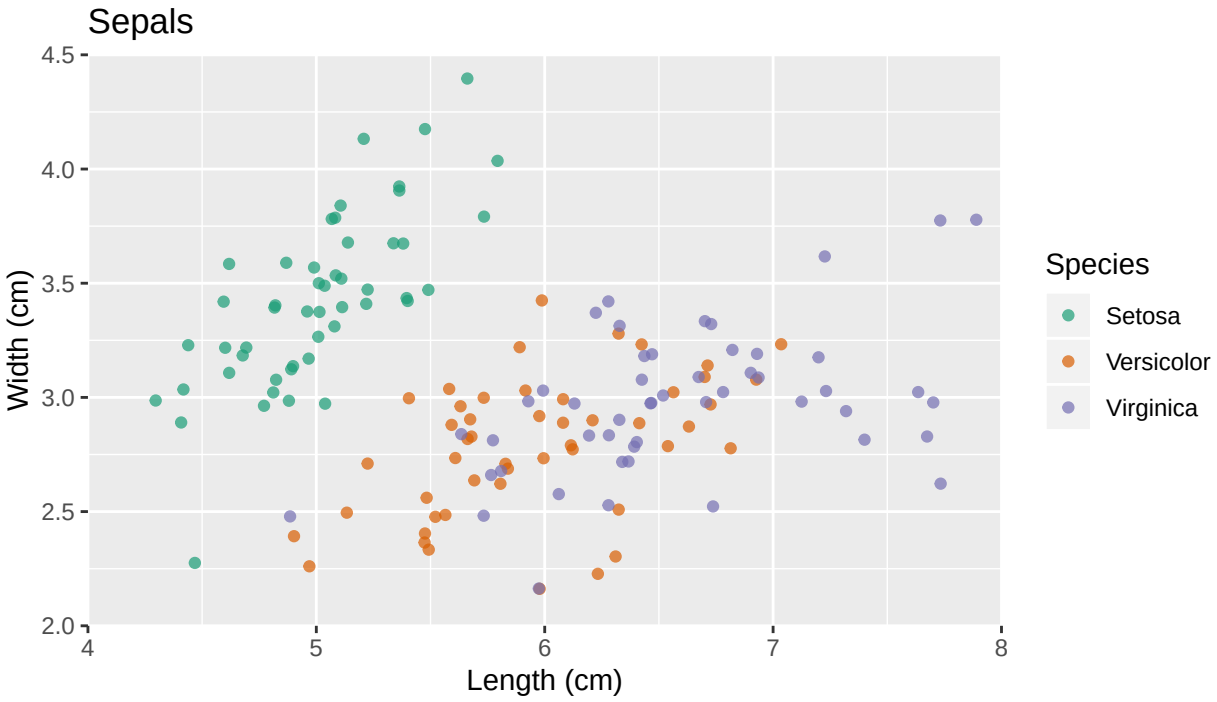
Theme update

```
original <- theme_update(panel.background = element_blank(),
   legend.background = element_blank(),
   legend.key = element_blank(),
   panel.grid = element_blank(),
   axis.text = element_text(colour = "black"),
   axis.line = element_line(colour = "black"),
   axis.ticks = element_line(colour = "black"),
   strip.background = element_blank())
```

theme_set

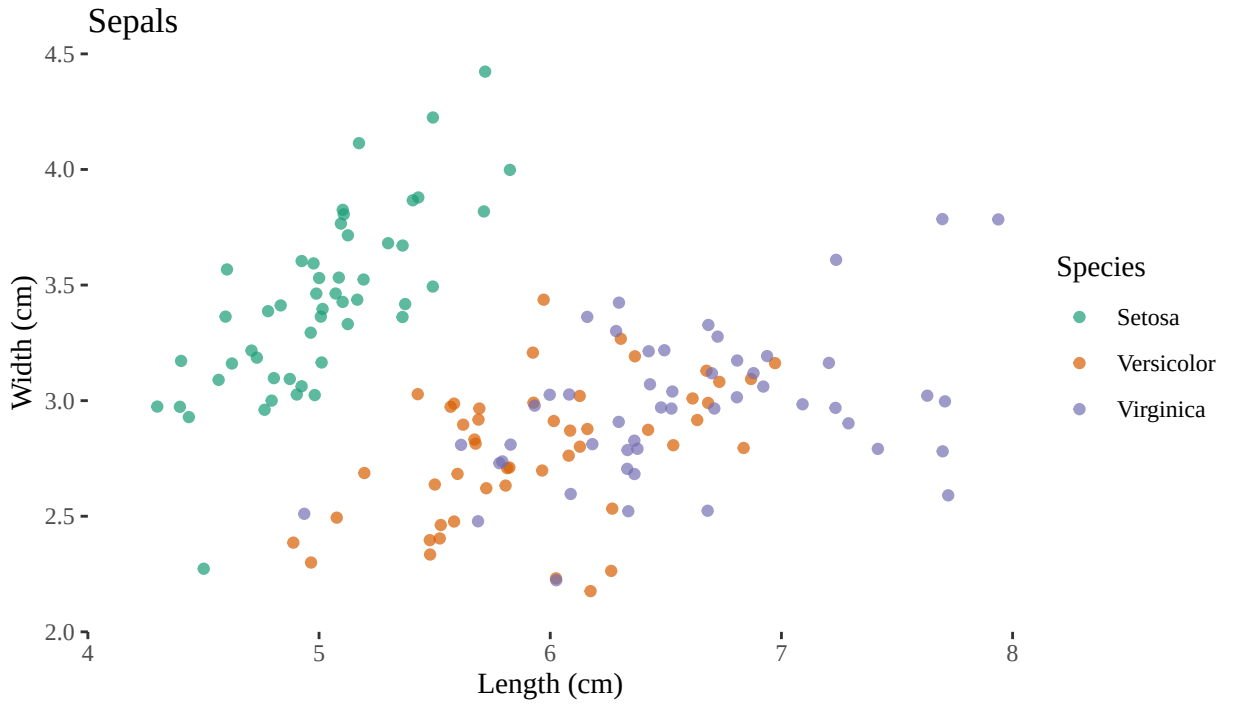
```
theme_set(original)
z
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

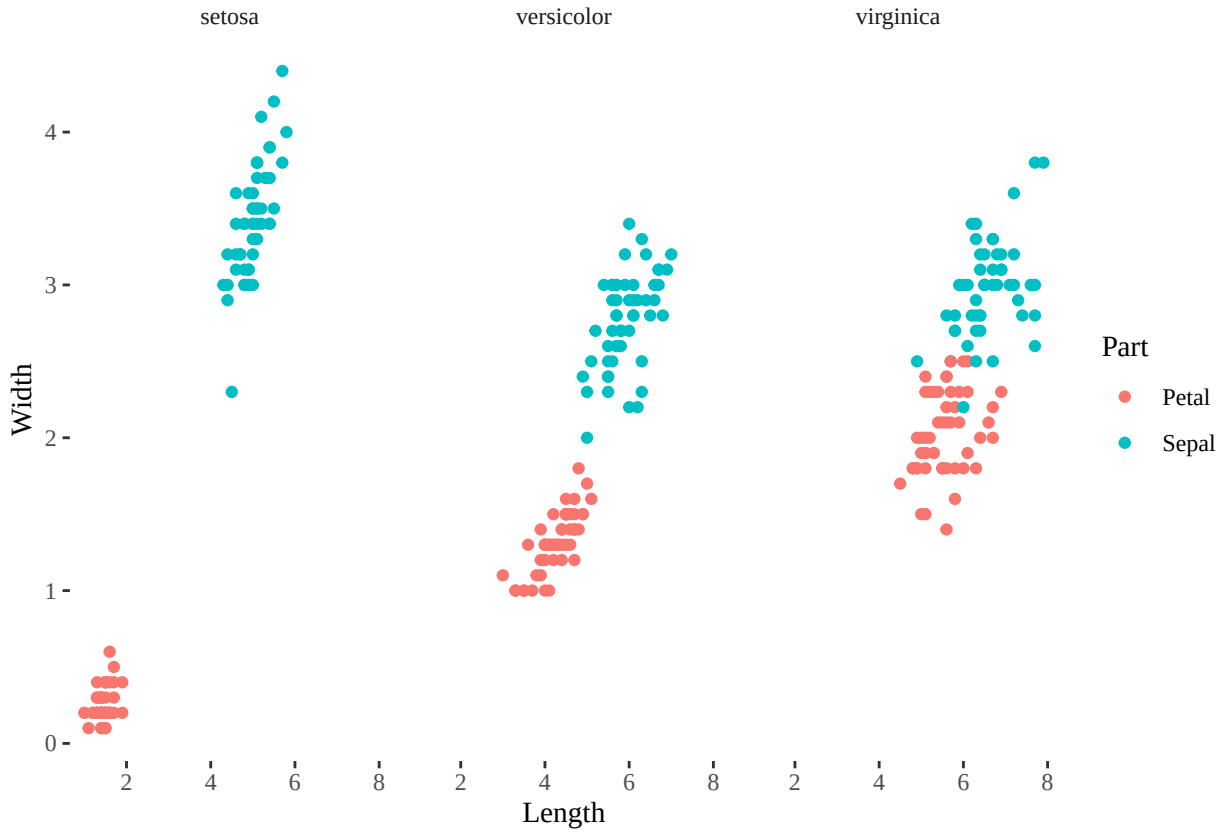



```
theme_set(theme_tufte())
z
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

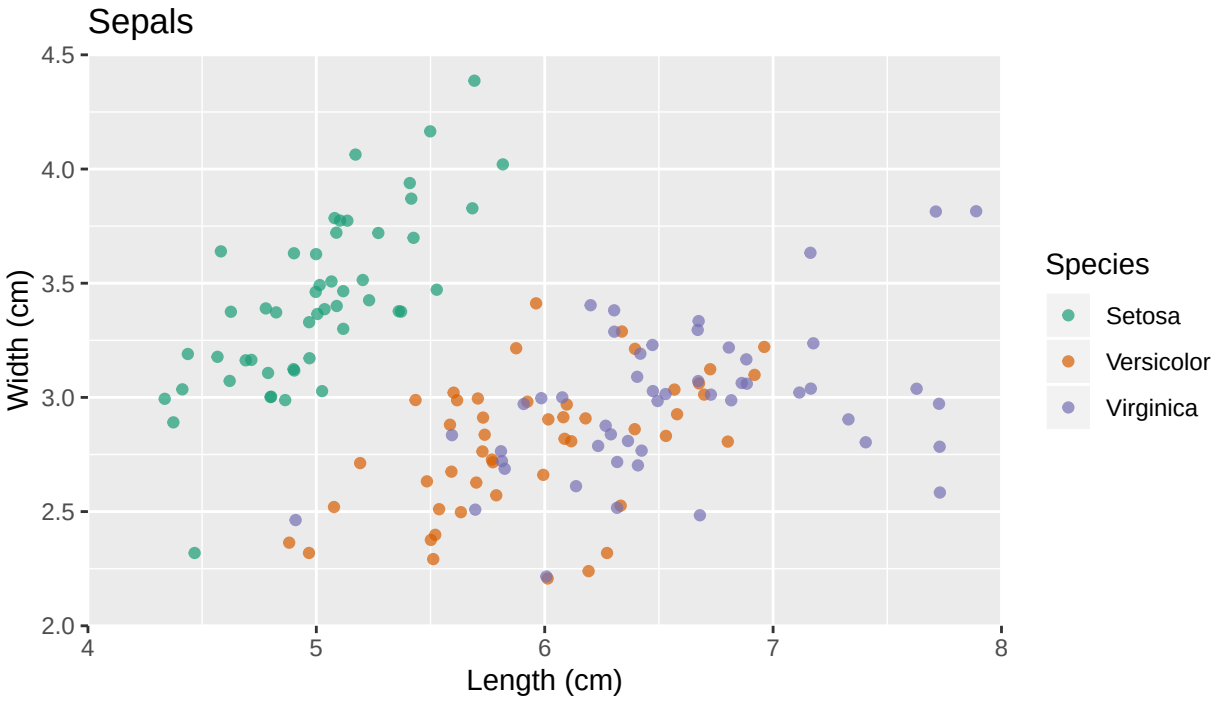

Sepals
Width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4
5
6
7
8
Length (cm)
Species
Setosa
Versicolor
Virginica

```
m
```




```
# back to original
theme_set(original) # saved earlier using theme_update()
z
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Practice

Updating Themes

In the previous exercises you've already customized the rectangles, lines and text on the plot. This theme layer is now separately stored as theme_pink, as shown in the sample code.

theme_update() updates the default theme used by ggplot2. The arguments for theme_update() are the same as for theme(). When you call theme_update() and assign it to an object (e.g. called old), that object stores the current default theme, and the arguments update the default theme. If you want to restore the previous default theme, you can get it back by using theme_update() again.

1 - "Apply" theme_pink to z2 to carry out all customizations.

2 - Instead of applying theme_pink, use theme_update(). This function returns an object that contains the previous theme settings, so that you can restore it later. Assign the output of theme_update() to an object called old.

3 - Plot z2 again, after the theme_update() call. The resulting plot has the same appearance as the previous one - but now you don't need to call theme() explictly.

4 - Restore the old theme using theme_set(old) and plot z2 again. It's back to the original default theme.

```
# Original plot
z2 <- ggplot(mtcars, aes(x = wt, y = mpg, col = factor(cyl))) +
  geom_point() +
  facet_wrap(~ cyl,scale = "free_y") +
  scale_color_brewer("Cylinders") +
```

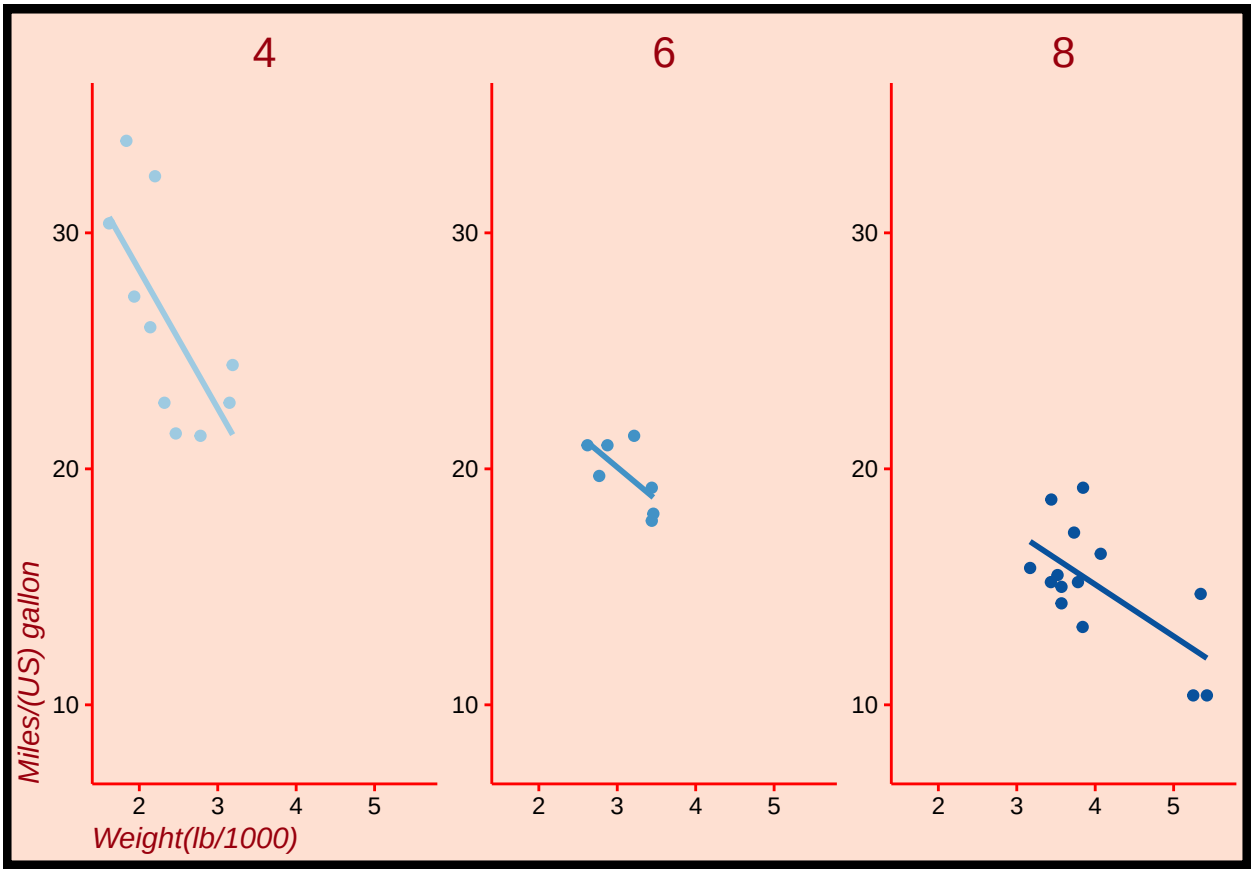
```
  scale_color_manual(values = mycol) +
  geom_smooth(se = FALSE, method = "lm") +
  scale_y_continuous("Miles/(US) gallon", limits = c(8,35)) +
  scale_x_continuous("Weight(lb/1000)", limits = c(1.6,5.6))
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

```
# Theme layer saved as an object, theme_pink
theme_pink <- theme(panel.background = element_blank(),
                    legend.key = element_blank(),
                    legend.background = element_blank(),
                    strip.background = element_blank(),
                    plot.background = element_rect(fill = myPink, color = "black", size = 3),
                    panel.grid = element_blank(),
                    axis.line = element_line(color = "red"),
                    axis.ticks = element_line(color = "red"),
                    strip.text = element_text(size = 16, color = myRed),
                    axis.title.y = element_text(color = myRed, hjust = 0, face = "italic"),
                    axis.title.x = element_text(color = myRed, hjust = 0, face = "italic"),
                    axis.text = element_text(color = "black"),
                    legend.position = "none")

# 1 - Apply theme_pink to z2
z2 +
  theme_pink
```

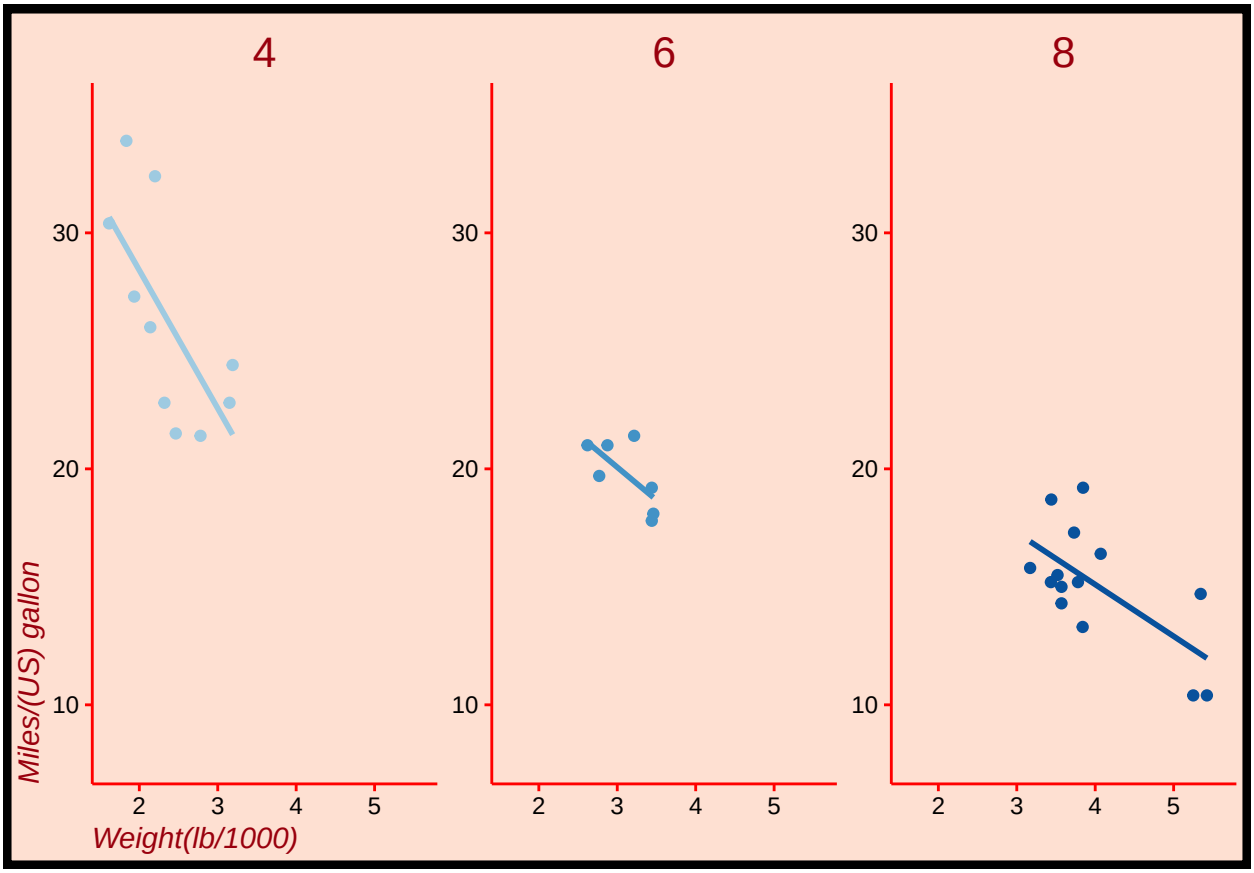
```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



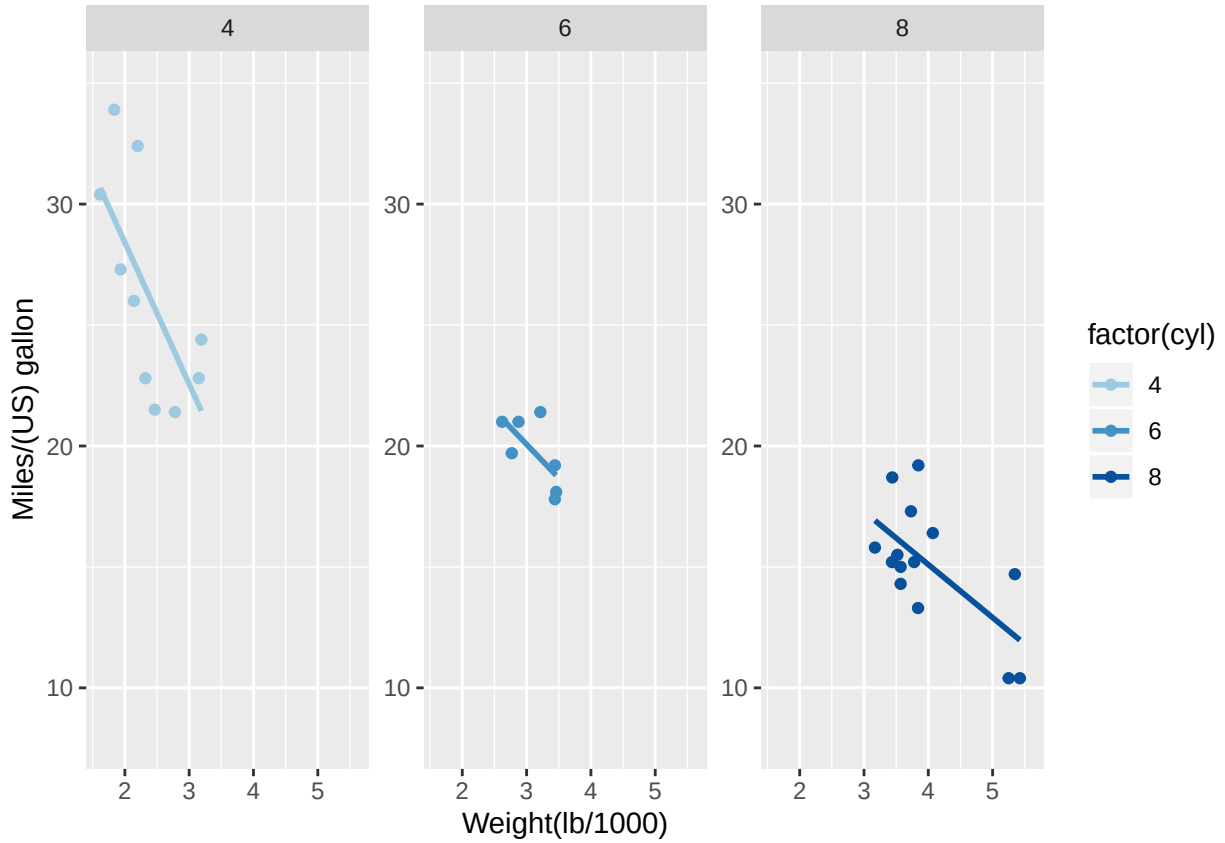

```
# 2 - Update the default theme, and at the same time
# assign the old theme to the object old.
old <- theme_update(panel.background = element_blank(),
             legend.key = element_blank(),
             legend.background = element_blank(),
             strip.background = element_blank(),
             plot.background = element_rect(fill = myPink, color = "black", size = 3),
             panel.grid = element_blank(),
             axis.line = element_line(color = "red"),
             axis.ticks = element_line(color = "red"),
             strip.text = element_text(size = 16, color = myRed),
             axis.title.y = element_text(color = myRed, hjust = 0, face = "italic"),
             axis.title.x = element_text(color = myRed, hjust = 0, face = "italic"),
             axis.text = element_text(color = "black"),
             legend.position = "none")

# 3 - Display the plot z2 - new default theme used
z2 + theme_update()
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# 4 - Restore the old default theme
theme_set(old)

# Display the plot z2 - old theme restored
z2 + theme_set(old)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).
```



Exploring ggthemes

There are many themes available by default in ggplot2: theme_bw(), theme_classic(), theme_gray(), etc. In the previous exercise, you saw that you can apply these themes to all following plots, with theme_set():

theme_set(theme_bw())

But you can also apply them on an individual plot, with:

... + theme_bw()

```
# Original plot
z2 <- ggplot(mtcars, aes(x = wt, y = mpg, col = factor(cyl))) +
  geom_point() +
  # facet_wrap(~ cyl,scale = "free_y") +
  scale_color_brewer("Cylinders") +
  scale_color_manual(values = mycol) +
  geom_smooth(se = FALSE, method = "lm") +
  scale_y_continuous("Miles/(US) gallon", limits = c(8,35)) +
  scale_x_continuous("Weight(lb/1000)", limits = c(1.6,5.6))
```

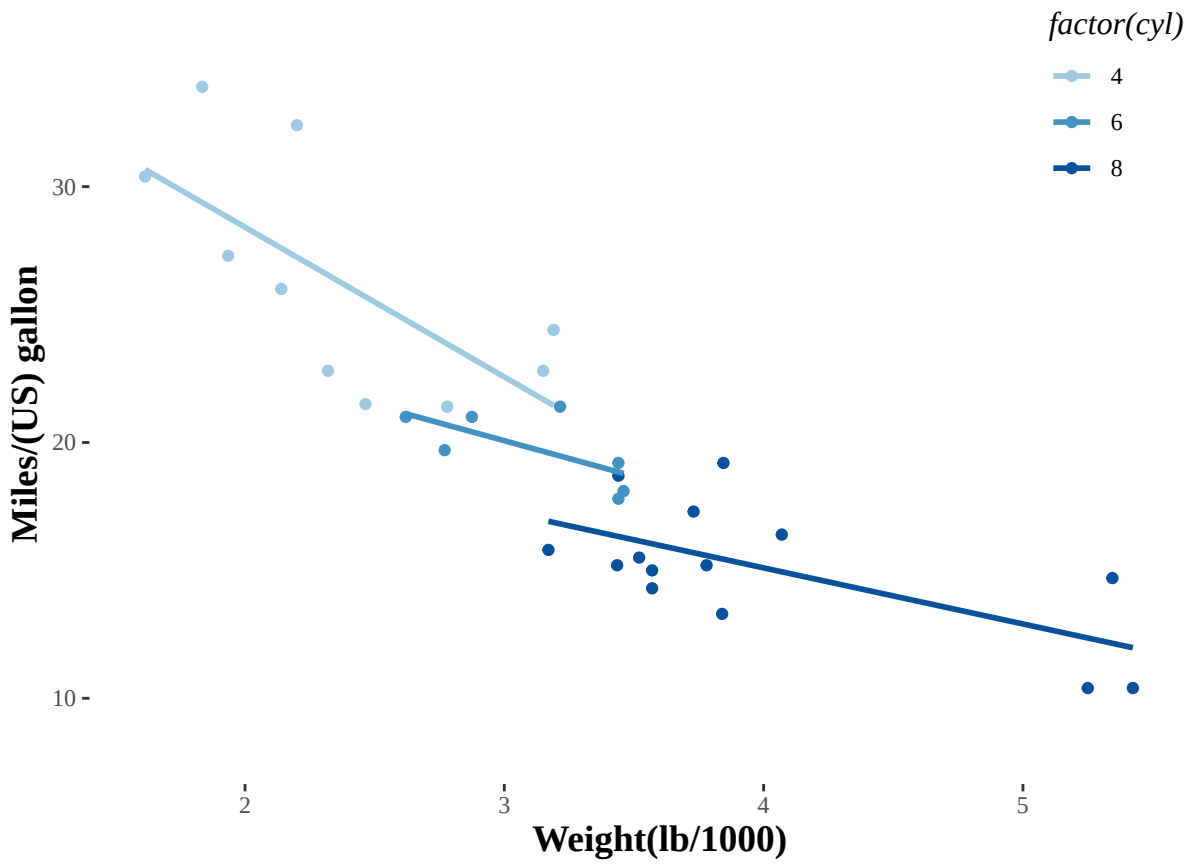
```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

```
# Load ggthemes
library(ggthemes)

# Apply theme_tufte(), plot additional modifications
```

```
custom_theme <- theme_tufte() +
  theme(legend.position = c(0.9,0.9),legend.title = element_text(
        face = "italic",
        size = 12), axis.title = element_text(face = "bold", size = 14))

# Draw the customized plot
z2 + custom_theme
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

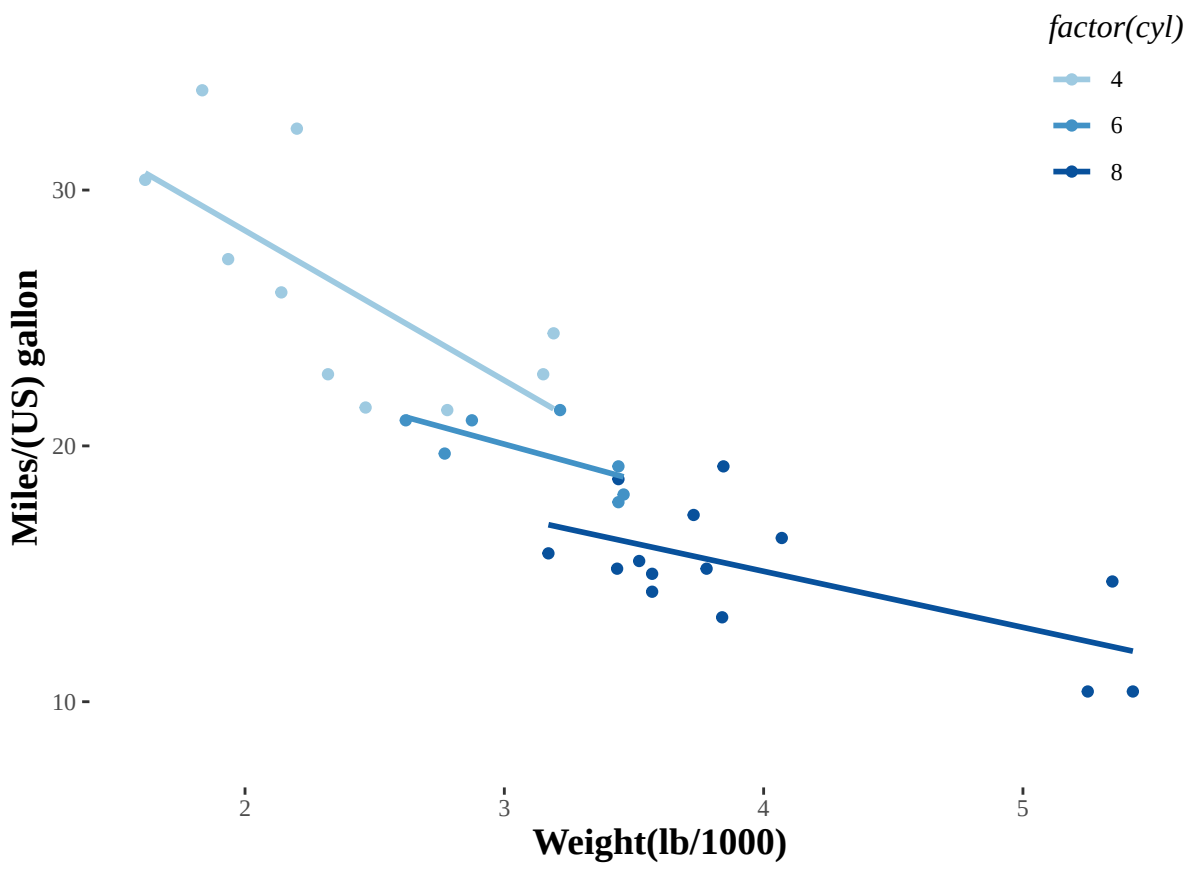
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# Use theme set to set custom theme as default
theme_set(custom_theme)

# Plot z2 again
z2 + theme_set(custom_theme)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```


factor(cyl)
4
6
8
30
20
10
Miles/(US) gallon
2
3
4
5
Weight(lb/1000)