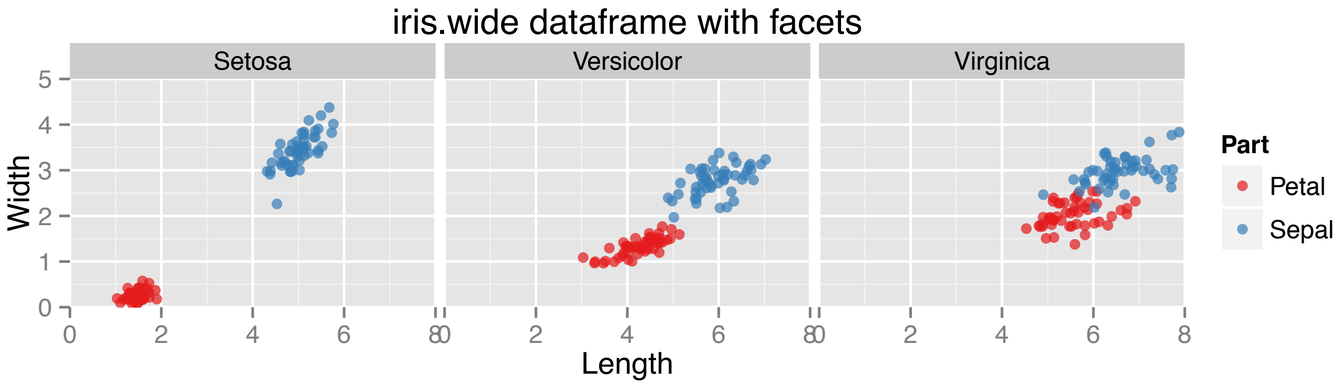
DATA VISUALIZATION WITH GGPLOT2

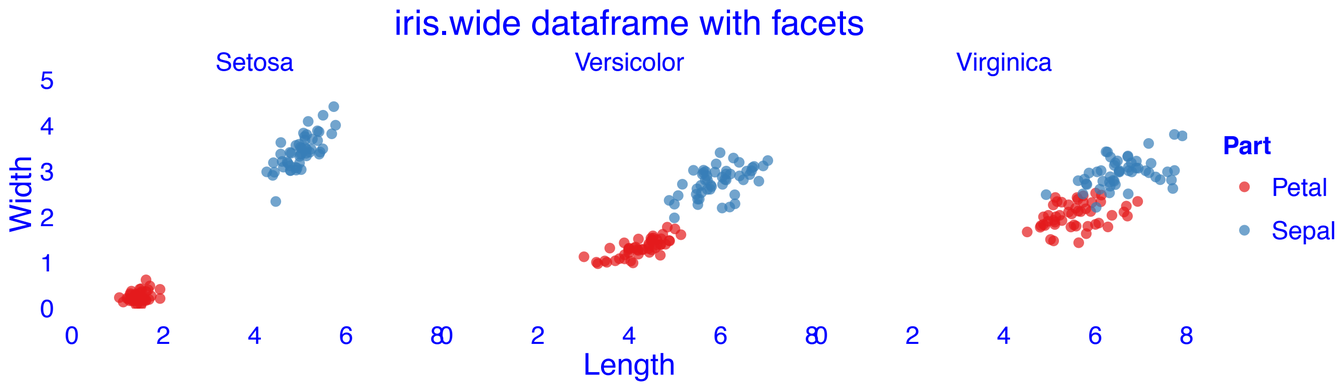
Themes

Themes Layer

- All the non-data ink
- Visual elements not part of data
- Three types
  - text element_text()
  - line element_line()
  - rectangle element_rect()

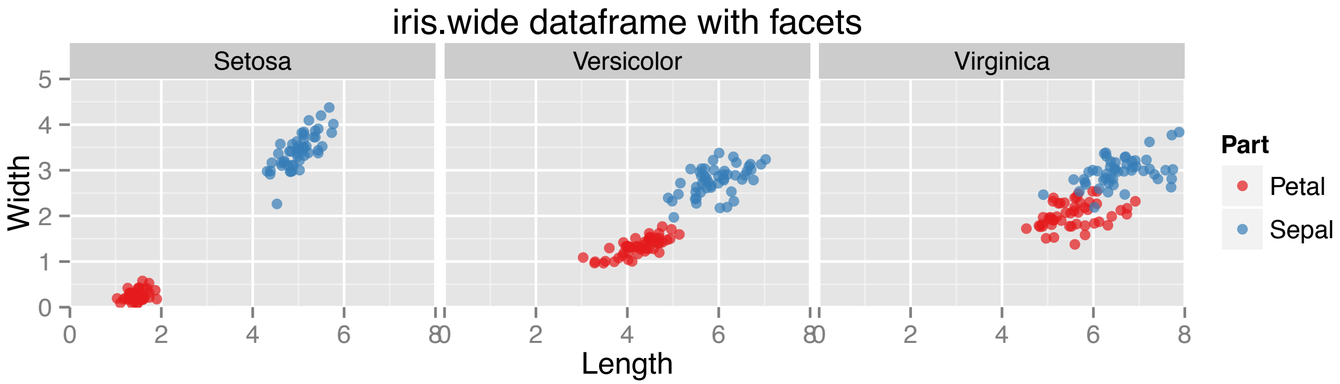

iris.wide dataframe with facets
Setosa
Versicolor
Virginica
Width
Length
0
1
2
3
4
5
0
2
4
6
8
Part
Petal
Sepal

text

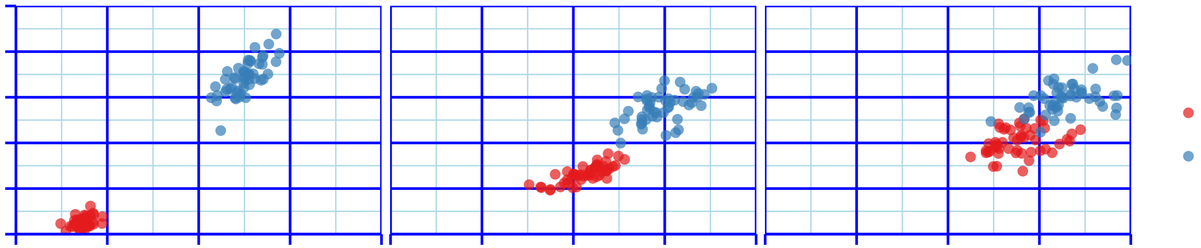



```
theme( text = element_text()
       title =
       plot.title =
       legend.text =
       legend.title =
       axis.title =
       axis.title.x =
       axis.title.y =
       axis.text =
       axis.text.x =
       axis.text.y =
       strip.text =
       strip.text.x =
       strip.text.y =
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

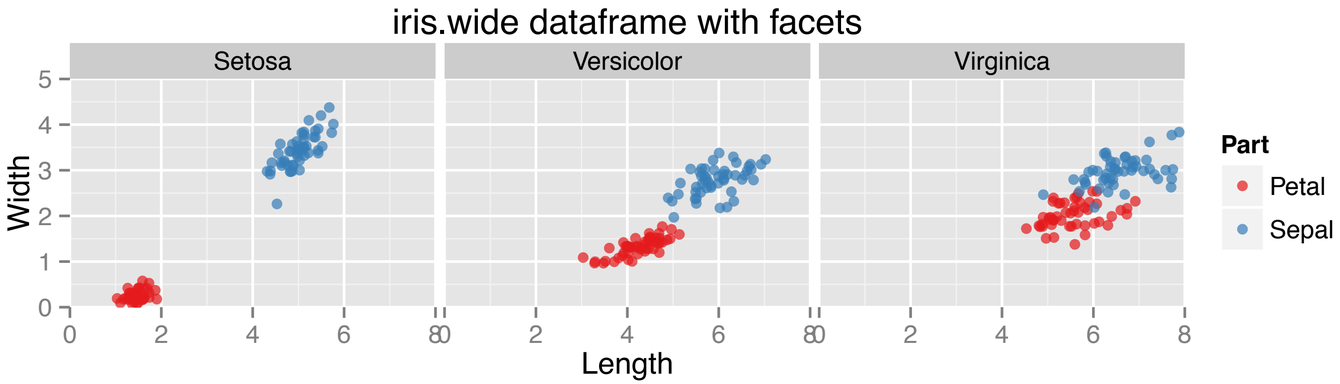
strip.text is for facet names


iris.wide dataframe with facets
Setosa
Versicolor
Virginica
Width
Length
0
1
2
3
4
5
0
2
4
6
8
Part
Petal
Sepal

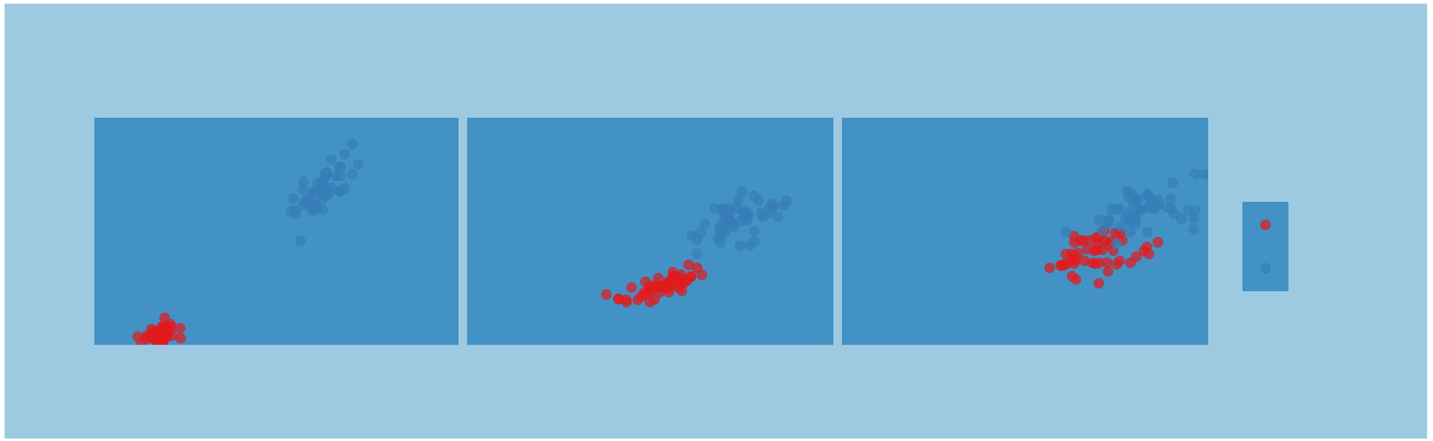
line



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


iris.wide dataframe with facets
Setosa
Versicolor
Virginica
Width
Length
0
1
2
3
4
5
0
2
4
6
8
Part
Petal
Sepal

rect



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

that's why we don't need to modify each element separately

Inheritance

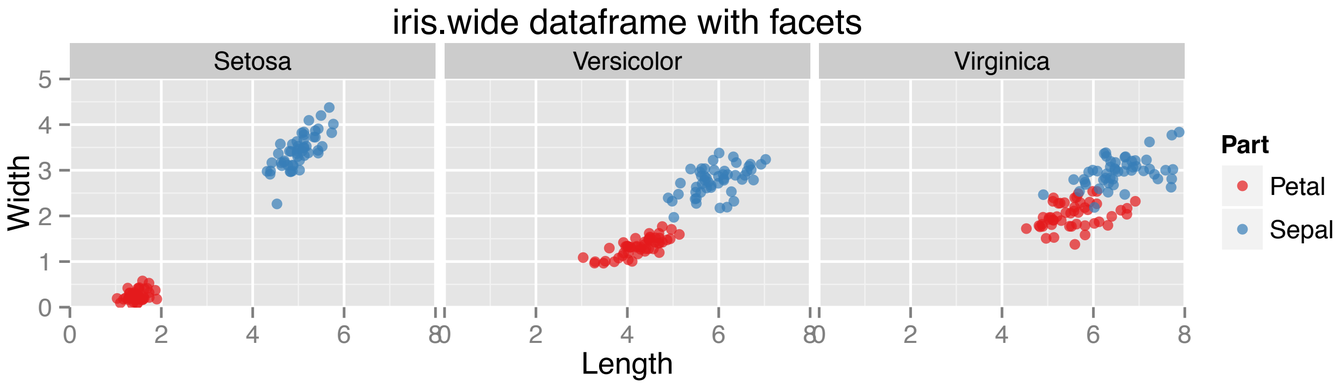
text
title
plot.title
legend.title
axis.title
axis.title.x
axis.title.y
legend.text
axis.text
axis.text.x
axis.text.y
strip.text
strip.text.x
strip.text.y

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

rect
legend.background
legend.key
panel.background
panel.border
plot.background
strip.background

Inheritance

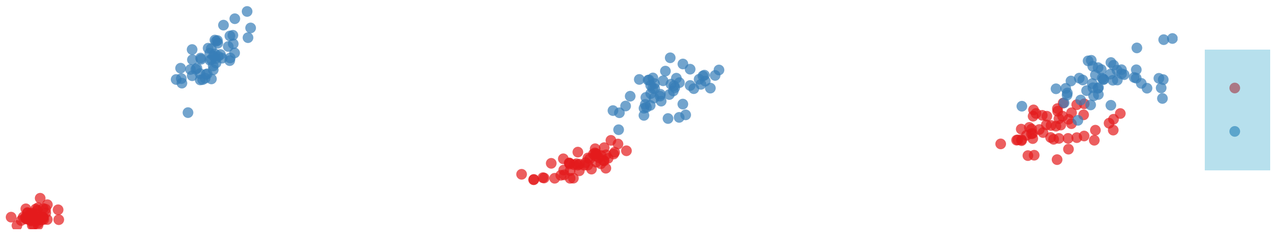
- text
  - title
    - plot.title
    - legend.title
  - axis.title
    - axis.title.x
    - axis.title.y
  - legend.text
  - axis.text
    - axis.text.x
    - axis.text.y
  - strip.text
    - strip.text.x
    - strip.text.y
- line
  - axis.ticks
    - axis.ticks.x
    - axis.ticks.y
  - axis.line
    - 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
- rect
  - legend.background
  - legend.key
  - panel.background
  - panel.border
  - plot.background
  - strip.background


iris.wide dataframe with facets
Setosa
Versicolor
Virginica
Width
Length
0
1
2
3
4
5
0
2
4
6
8
Part
Petal
Sepal

element_blank

to delete / hide an element from the plot

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



DATA VISUALIZATION WITH GGPLOT2

Recycling Themes

this is the one I've been waiting for :))))

Recycling Themes

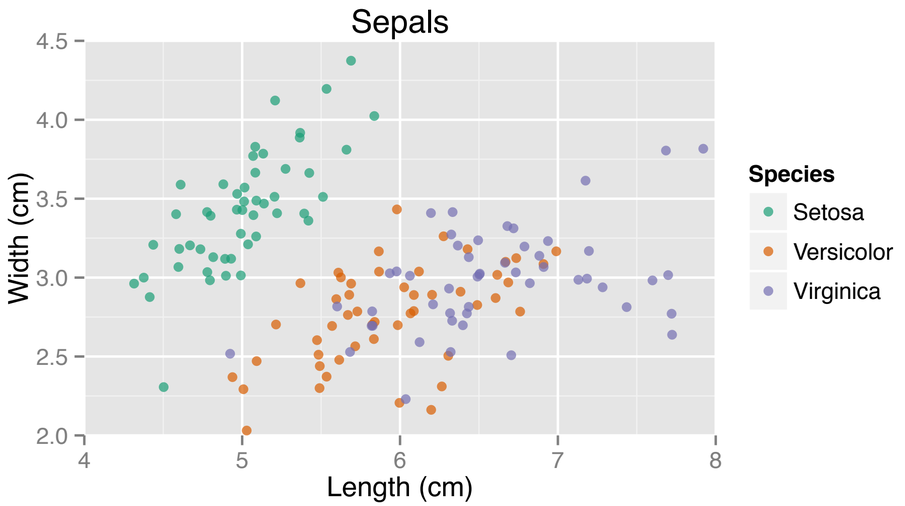
- Many plots
- Consistency in style
- Apply specific theme everywhere

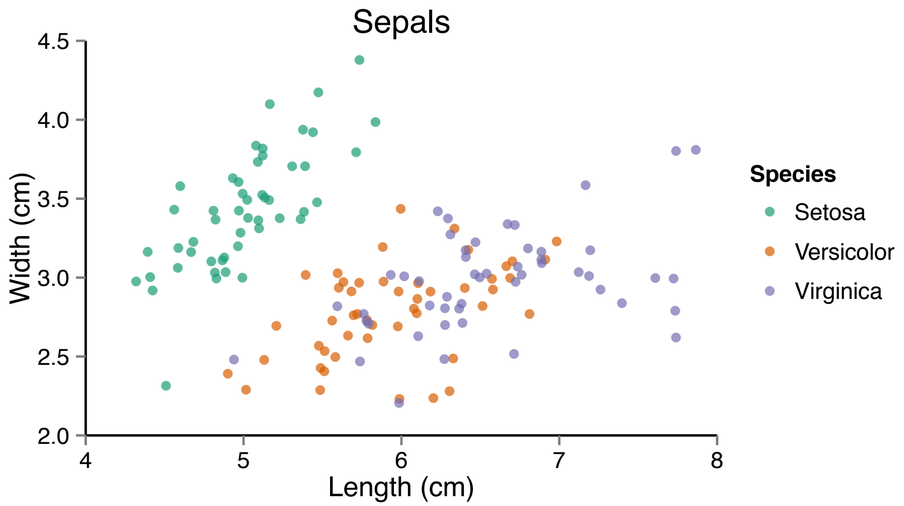
Z

```
> 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

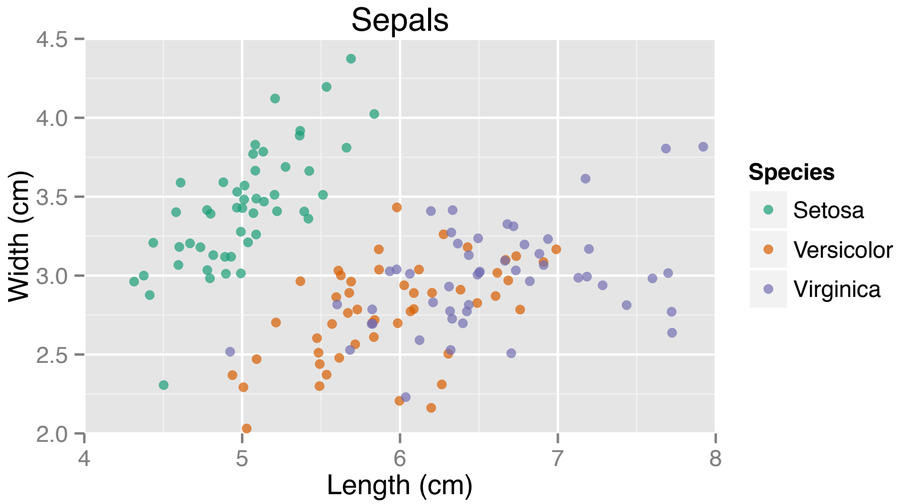
```
> z
> z + theme(panel.background = element_blank(),
            legend.background = element_blank(),
            legend.key = element_blank(),
            panel.grid = element_blank(),
            axis.text = element_text(colour = "black"),
            axis.line = element_line(colour = "black"))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

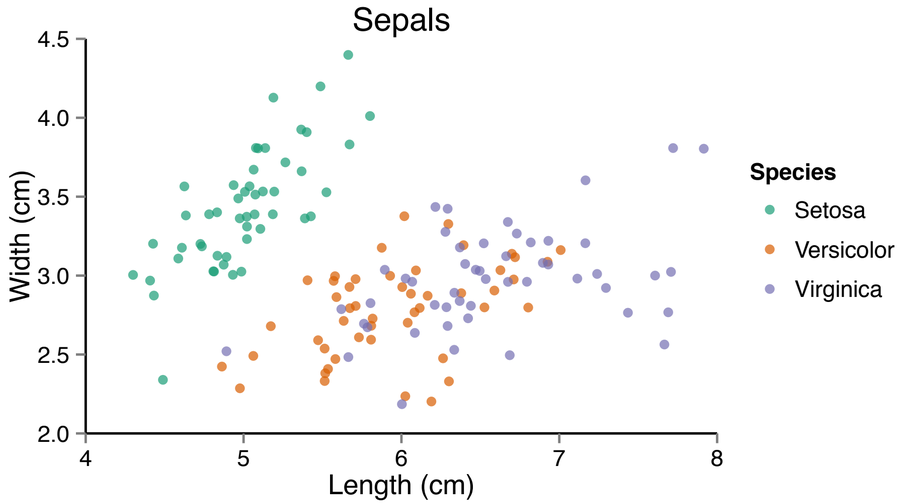




Save theme

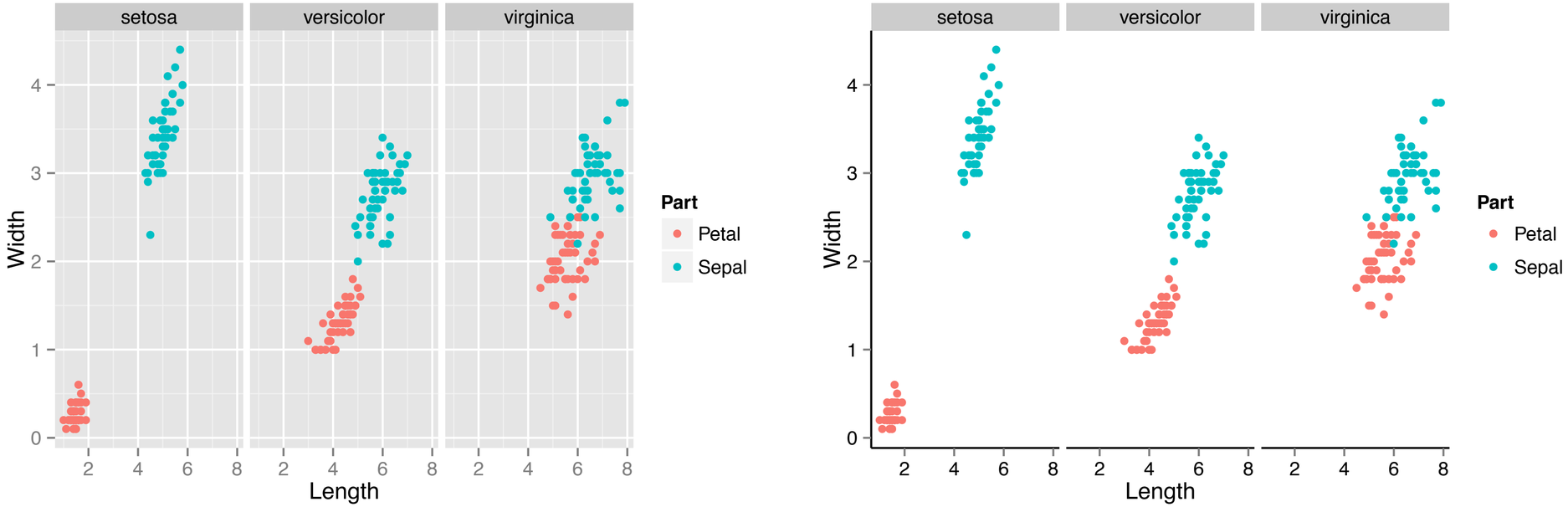
```
> theme_iris <- theme(panel.background = element_blank(),
                      legend.background = element_blank(),
                      legend.key = element_blank(),
                      panel.grid = element_blank(),
                      axis.text = element_text(colour = "black"),
         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





Reuse theme

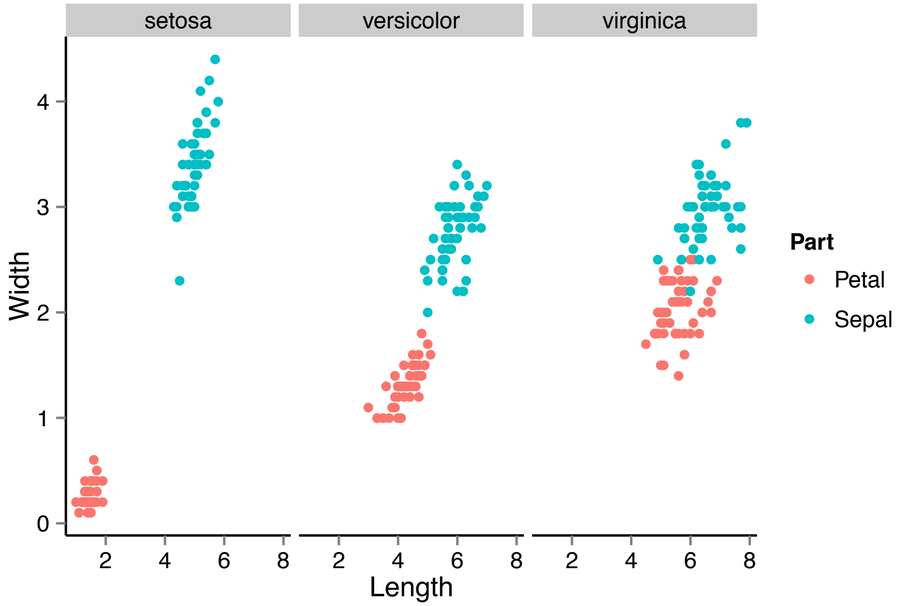
```
> m <- ggplot(iris.wide, aes(x = Length, y = Width, col = Part)) +
    geom_point() +
    facet_grid(. ~ Species)
> m
> m + theme_iris
```

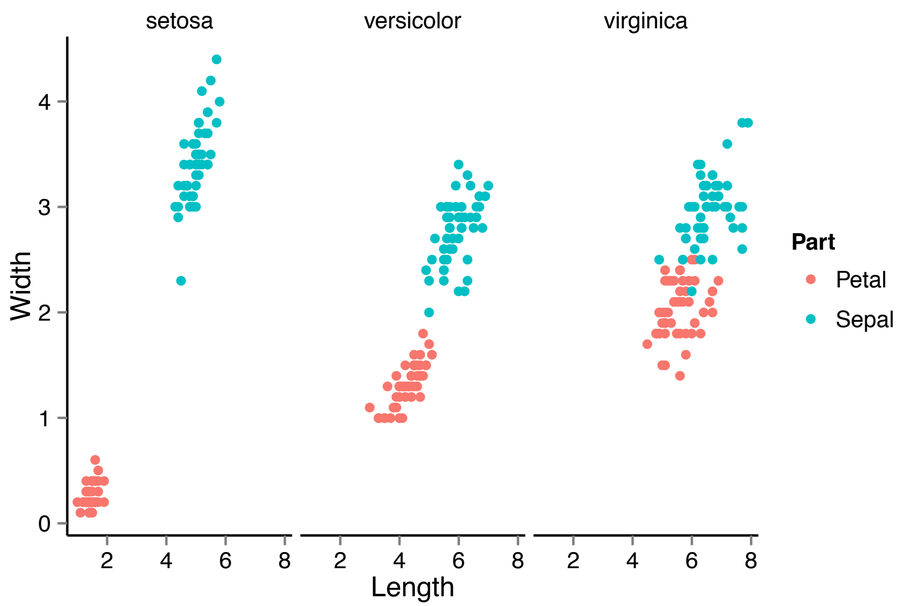


Extend theme

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

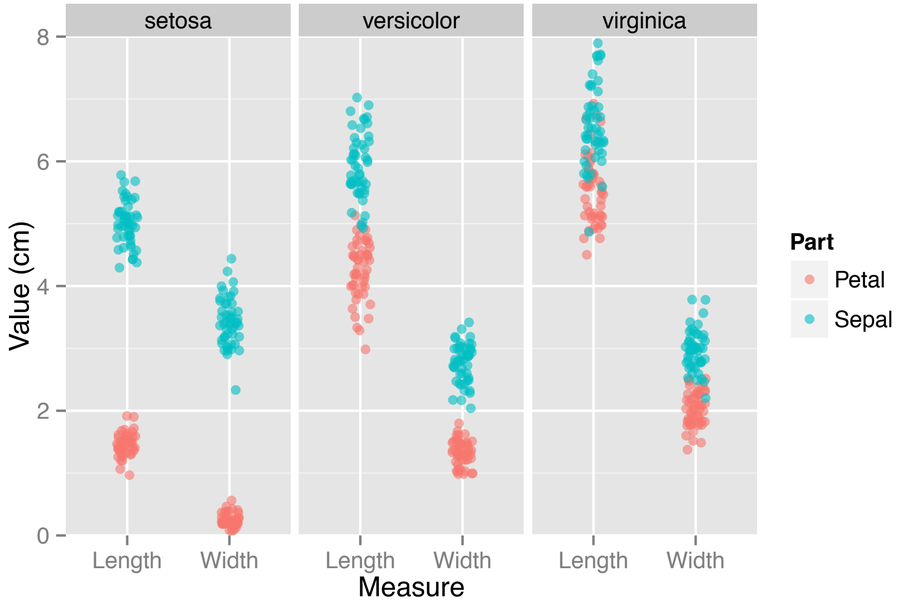
previous plot





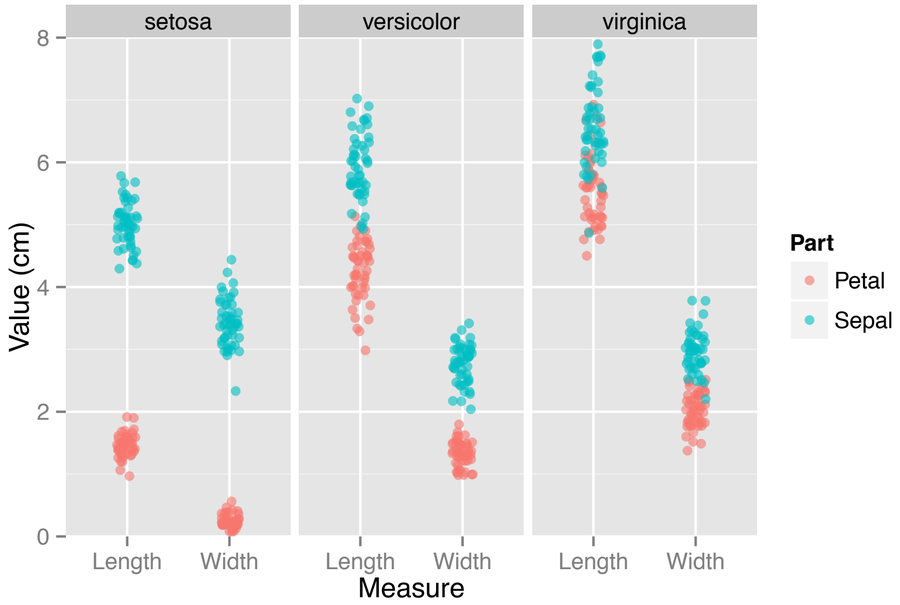
Discrete x-axis

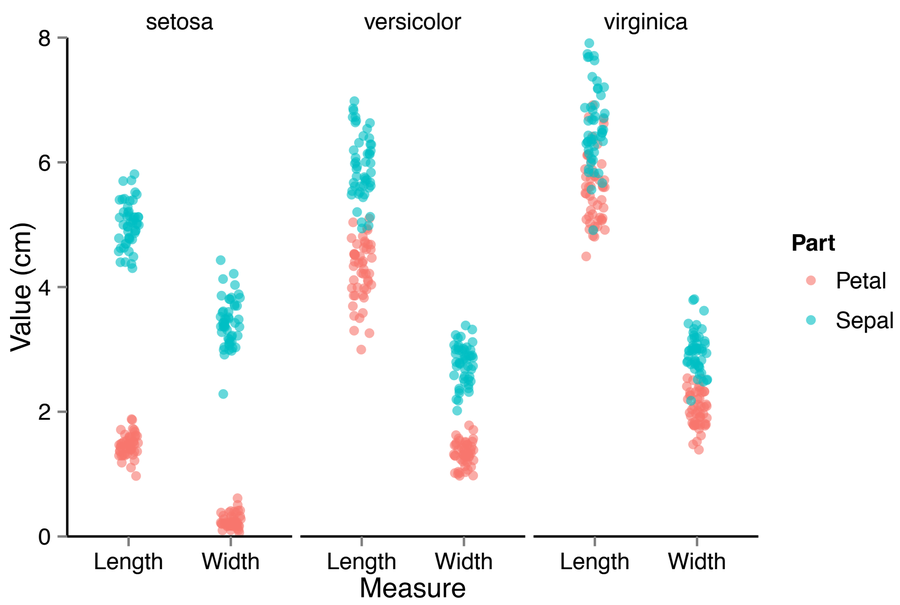
```
> p <- ggplot(iris.tidy, aes(x = Measure, y = Value, col = Part)) +
        geom_point(position = position_jitter(0.1), alpha = 0.6,
                   width = 0.4) +
        scale_y_continuous("Value (cm)", limits = c(0, 8),
                           expand = c(0, 0)) +
        facet_grid(. ~ Species)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



Discrete x-axis

```
> p
> p + theme_iris
```



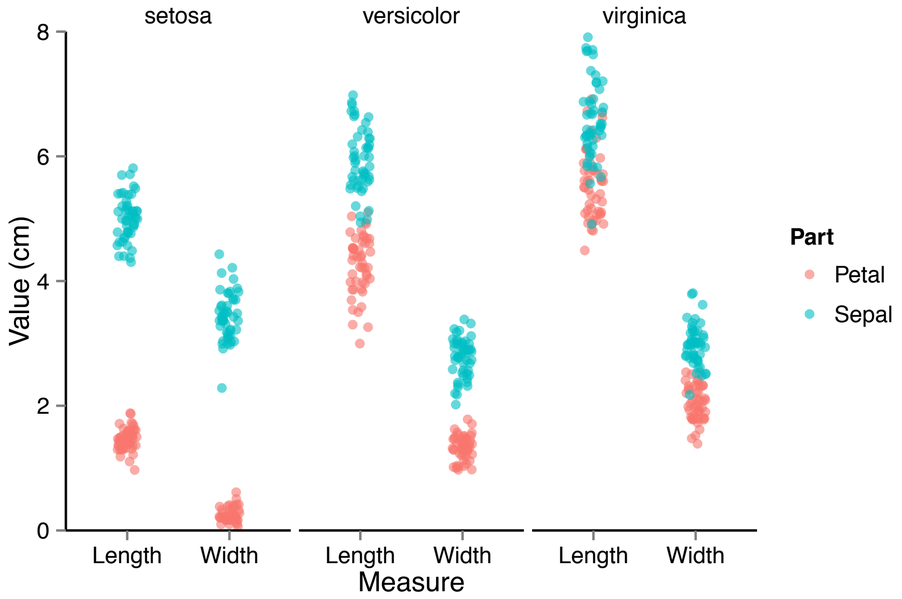


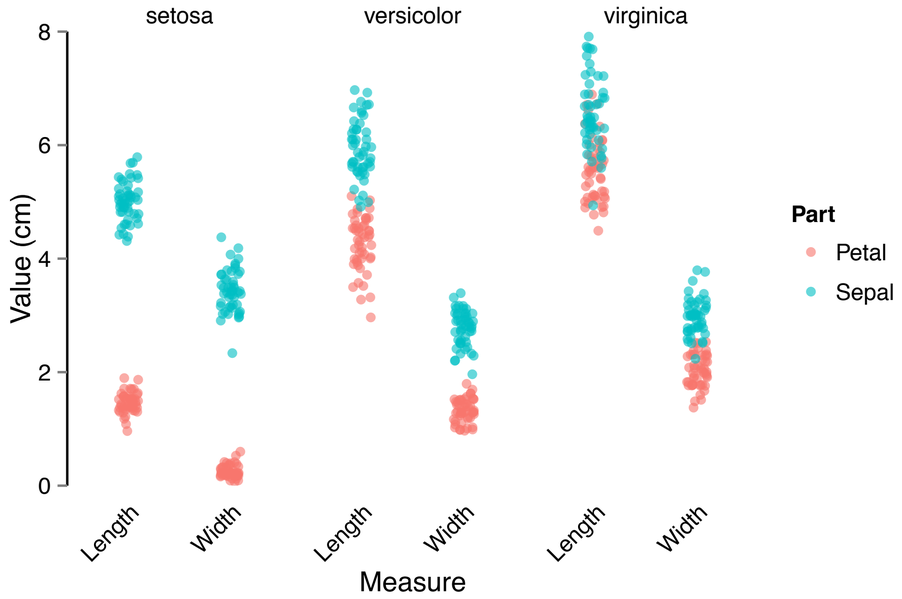
Derivative theme

```
> 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
```

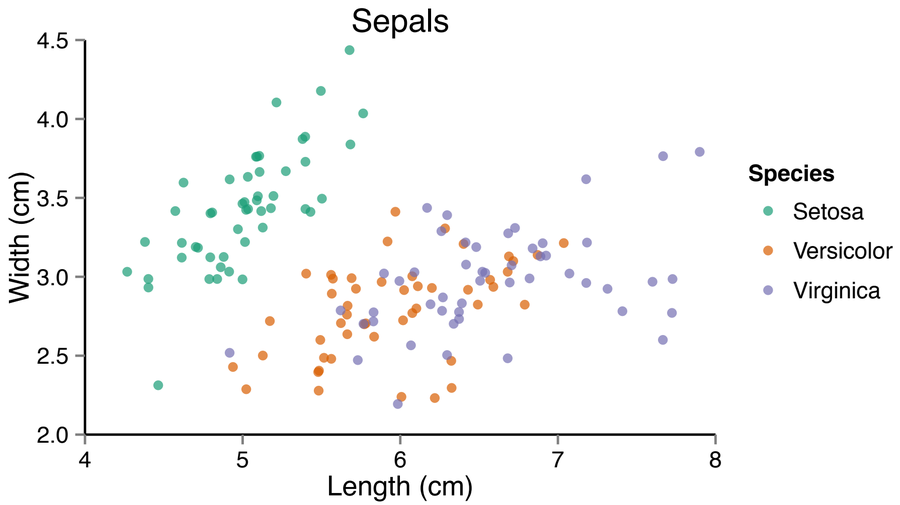
previous plot

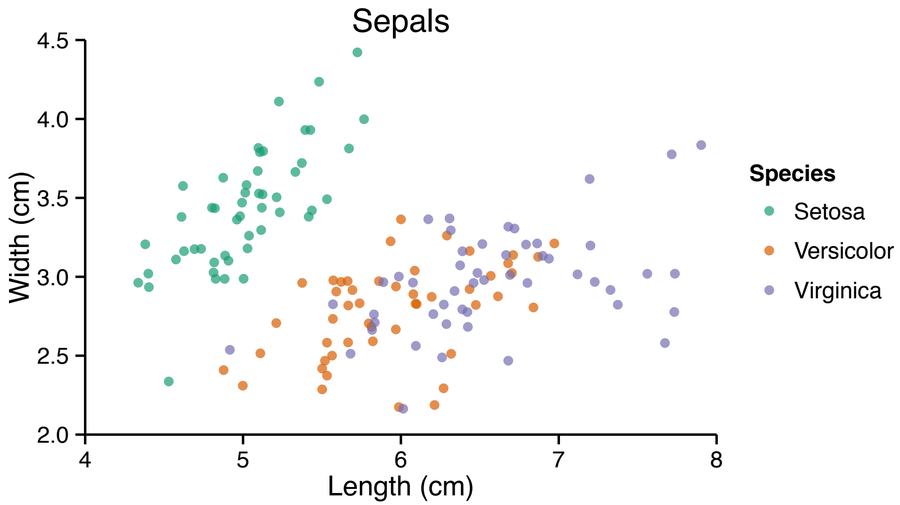




Built-in theme templates

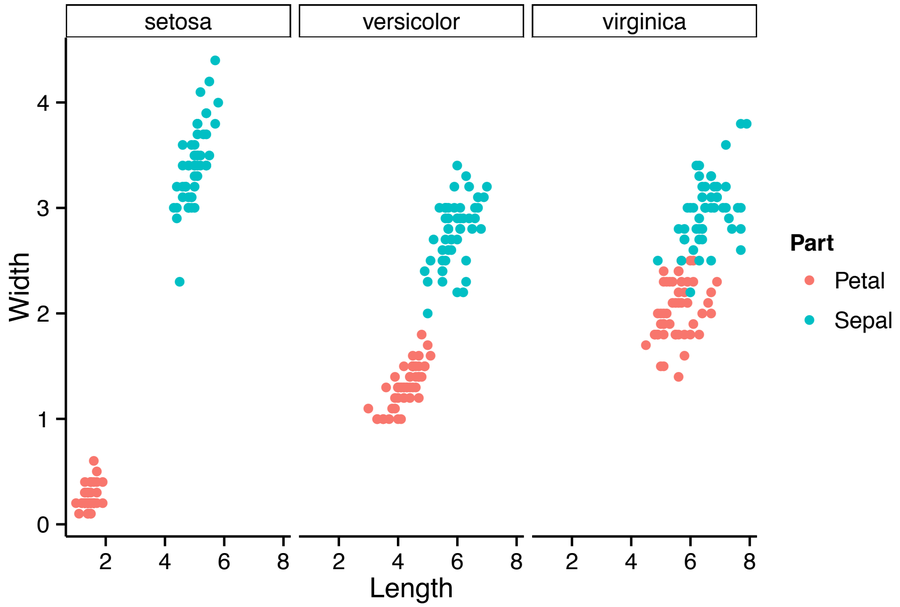
```
> z + theme_iris
> z + theme_classic()
```

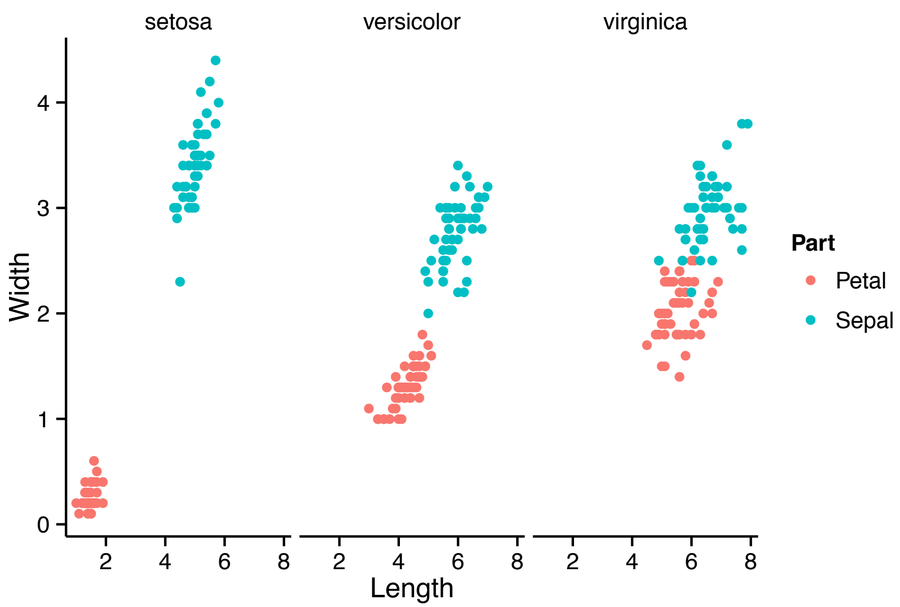




Built-in theme templates

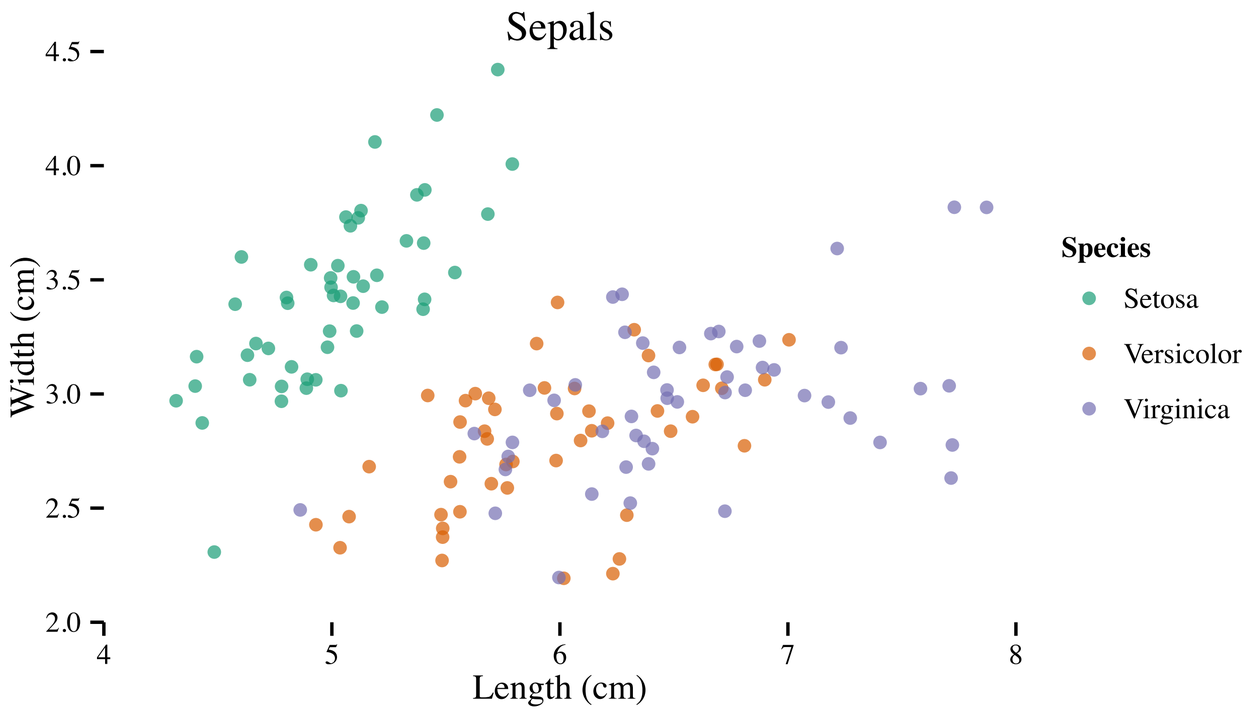
```
> m + theme_classic()
> m + theme_classic() +
    theme(strip.background = element_blank())
```





ggthemes

```
> library(ggthemes)
> z + theme_tufte()
```

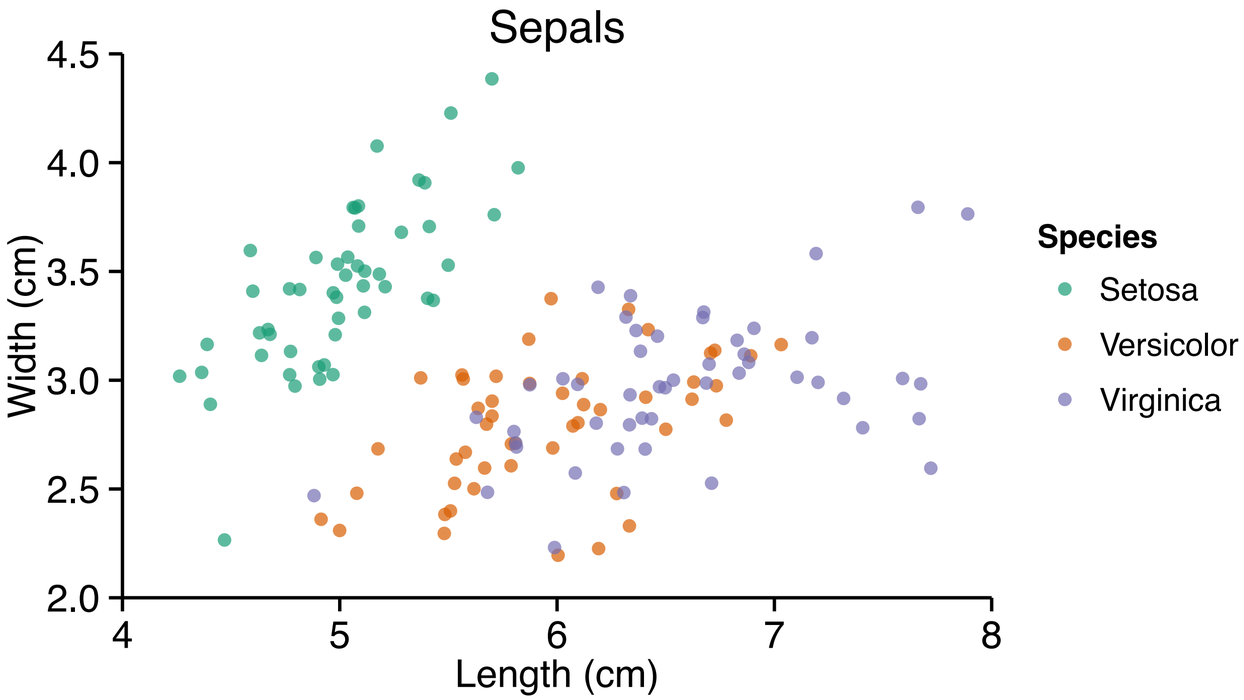


Theme update

```
> 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())
```

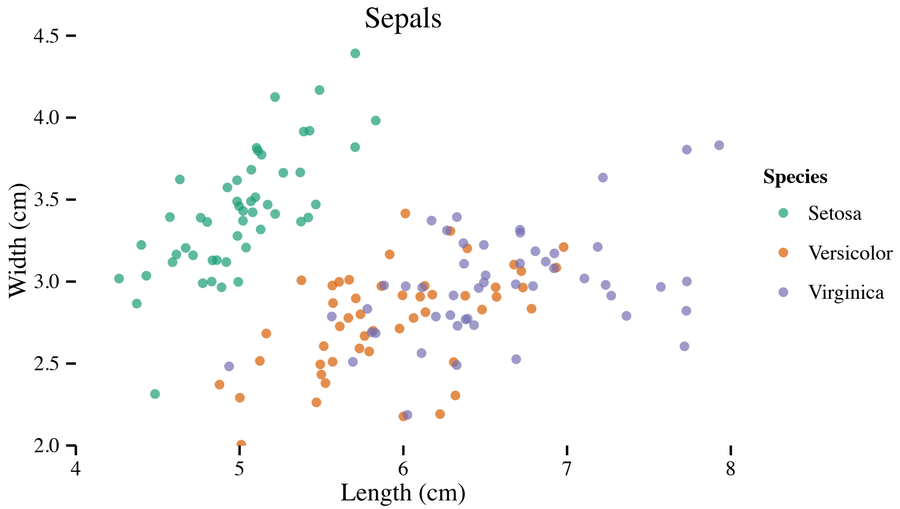
Theme applied everywhere

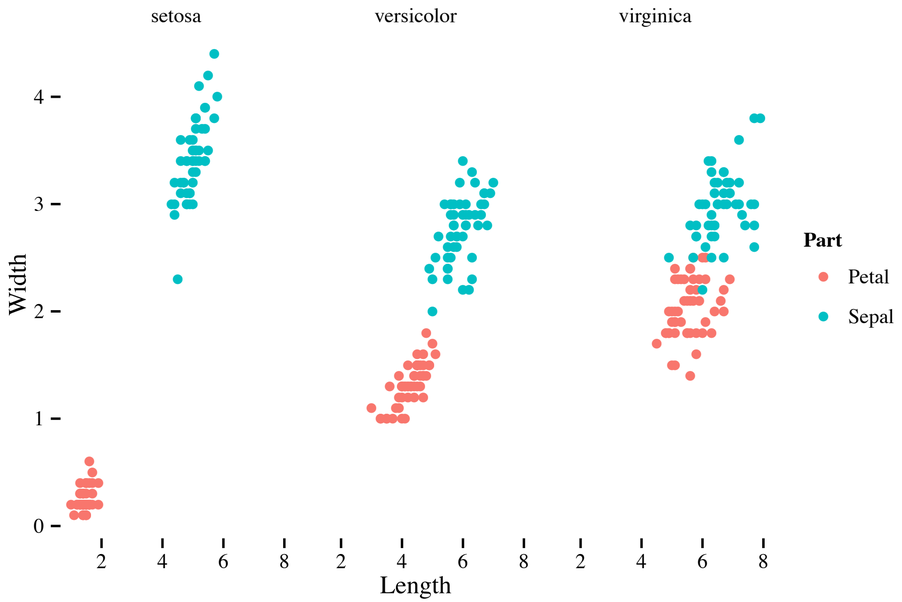
```
> z
```



theme_set

```
> theme_set(theme_tufte())
> z
> m
```





Back to original

theme_grey() is the default

```
> theme_set(original) # saved earlier using theme_update()
> z
```