Data Visualisation Using R

Lecture-3

Suman Rakshit

School of EECMS, Curtin University

February 9, 2024

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Outline

1. A quick review
2. Deep dive into themes layer
3. Key theme element-1: Text elements
4. Key theme element-2: Line elements
5. Key theme element-3: Rectangles
6. Creating custom themes
7. Ggplot-2 and Ggthemes in-built themes
8. Important scale functions
9. Summary

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

A quick recap of Ggplot2 basics

- Three most important components for any visualisation are: (i) data (ii) aesthetic mapping and (iii) geometry.
- Aesthetic mapping is performed using the function `aes(x, y, color, fill, shape, size, alpha)`.
- Geometry corresponds to *chart type*, and is specified using `geom_⋆()`, where a chart type is specified in place of ⋆.
- Visible attributes are specified outside `aes()` and, most often, inside the `geom_⋆()` function.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

A quick recap of Ggplot2 basics

- Three most important components for any visualisation are: (i) data (ii) aesthetic mapping and (iii) geometry.
- Aesthetic mapping is performed using the function `aes(x, y, color, fill, shape, size, alpha)`.
- Geometry corresponds to *chart type*, and is specified using `geom_*()`, where a chart type is specified in place of $\star$.
- Visible attributes are specified outside `aes()` and, most often, inside the `geom_*()` function.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

A quick recap of Ggplot2 basics

- Three most important components for any visualisation are: (i) data (ii) aesthetic mapping and (iii) geometry.
- Aesthetic mapping is performed using the function `aes(x, y, color, fill, shape, size, alpha)`.
- Geometry corresponds to *chart type*, and is specified using `geom_⋆()`, where a chart type is specified in place of ⋆.
- Visible attributes are specified outside `aes()` and, most often, inside the `geom_⋆()` function.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

A quick recap of Ggplot2 basics

- Three most important components for any visualisation are: (i) data (ii) aesthetic mapping and (iii) geometry.
- Aesthetic mapping is performed using the function `aes(x, y, color, fill, shape, size, alpha)`.
- Geometry corresponds to *chart type*, and is specified using `geom_⋆()`, where a chart type is specified in place of ⋆.
- Visible attributes are specified outside `aes()` and, most often, inside the `geom_⋆()` function.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Deep dive into themes layer

- The themes layer provides all non-data related visible attributes – `theme()` allows you to modify all non-data ink in your visualisation.

- Three main visual elements of themes layer:

  1. `text`: modify using `element_text()`
  2. `line`: modify using `element_line()`
  3. `rectangle`: modify `element_rect()`

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Important: element_⋆() functions!

Element	R-Function	Example
Text	element_text()	element_text(face = "bold")
Line	element_line()	element_line(color = "red")
Rectangle	element_rect()	element_rect(fill = "grey")

One important element_⋆() function is element_blank() — this function removes any themes-based visual element.

Example: theme(panel.grid = element_blank()) removes all panel grid-lines.

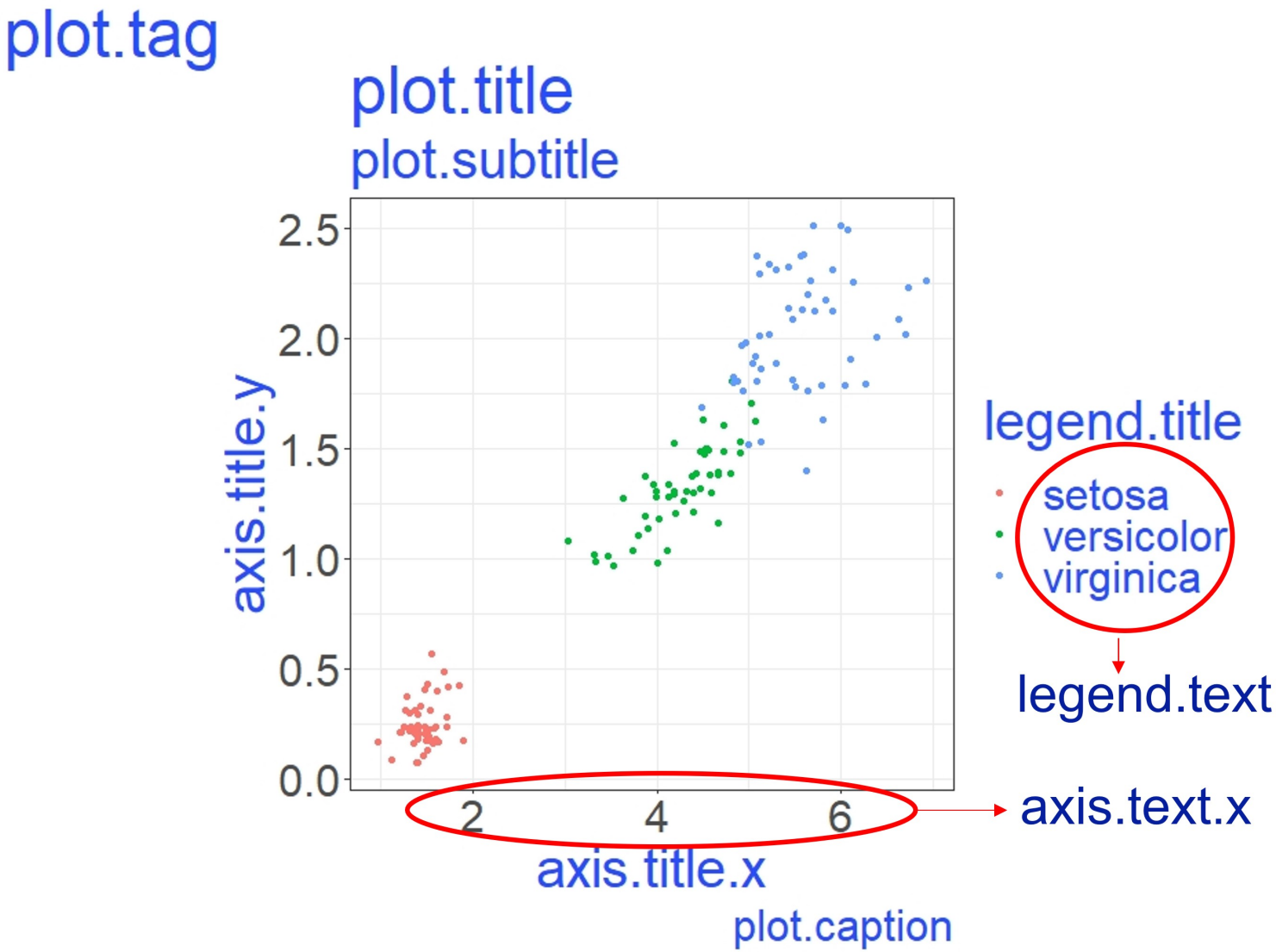
Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

text element hierarchy structure

- axis.title
  - axis.title.x
    - axis.title.x.top
    - axis.title.x.bottom
  - axis.title.y
    - axis.title.y.left
    - axis.title.y.right
- title
  - legend.title
  - plot.title
  - plot.subtitle
  - plot.caption
  - plot.tag
- axis.text
  - axis.text.x
    - axis.text.x.top
    - axis.text.x.bottom
  - axis.text.y
    - axis.text.y.left
    - axis.text.y.right
- legend.text
- strip.text
  - strip.text.x
  - strip.text.y

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Visualize `text` elements in a plot



Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode to specify text elements

```
# Standard scatter plot
ggplot(iris, aes(Petal.Length, Petal.
    Width, color=Species)) +
geom_point() + theme_bw() +
# Specify text elements using labs()
labs(color = "legend.title",
      title = "plot.title",
      subtitle = "plot.subtitle",
      caption = "plot.caption",
      tag = "plot.tag",
      x = "axis.title.x",
      y = "axis.title.y") +
# Modify text size and color
theme(text = element_text(size = 30,
                          color = "#2A4DFA"))
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Visualize `strip.text`

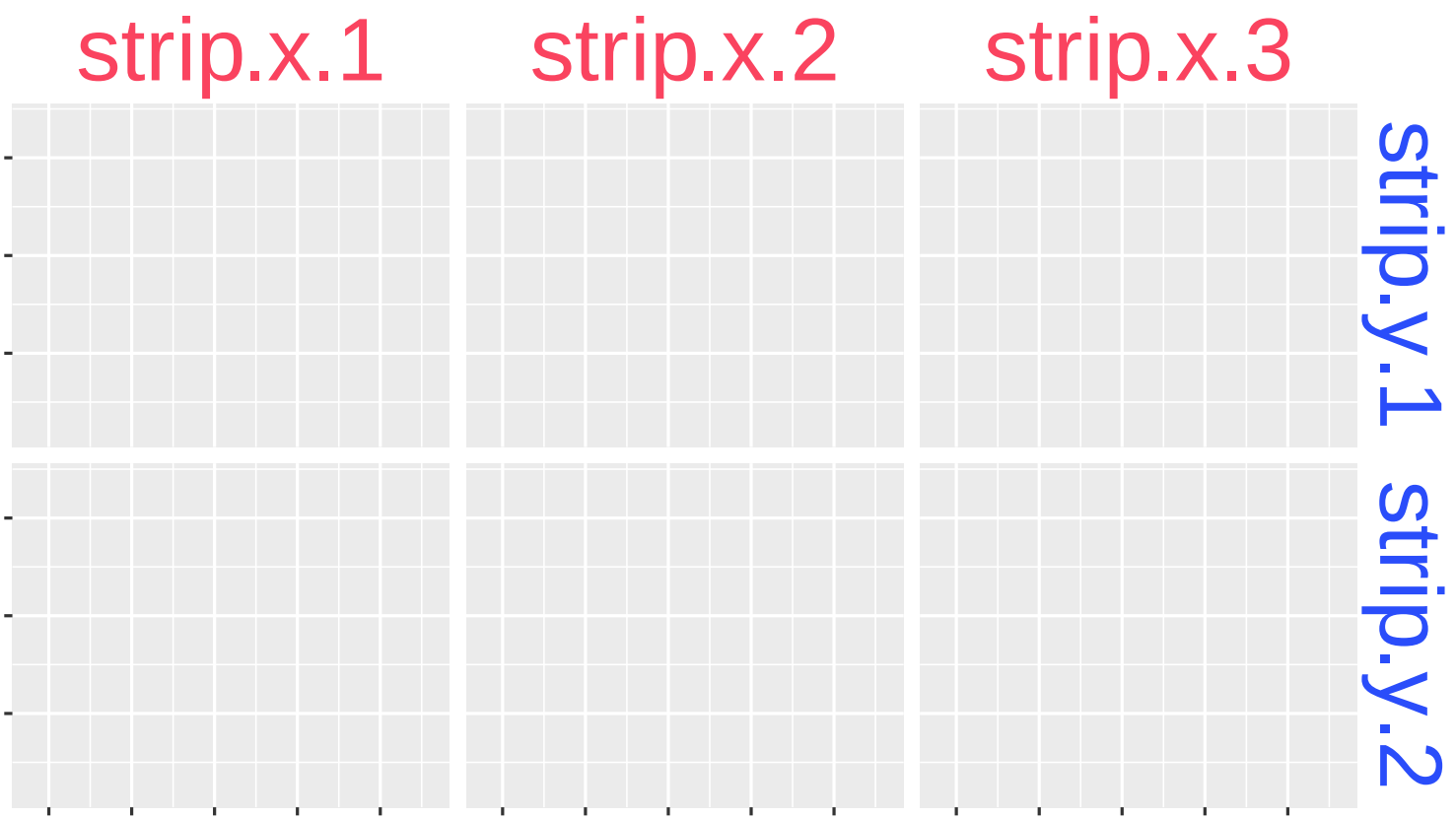


Figure 1: `strip.text.x` and `strip.text.y`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modified `strip.text`

We modified the font family, color and size of the `strip.text` element in the plot.

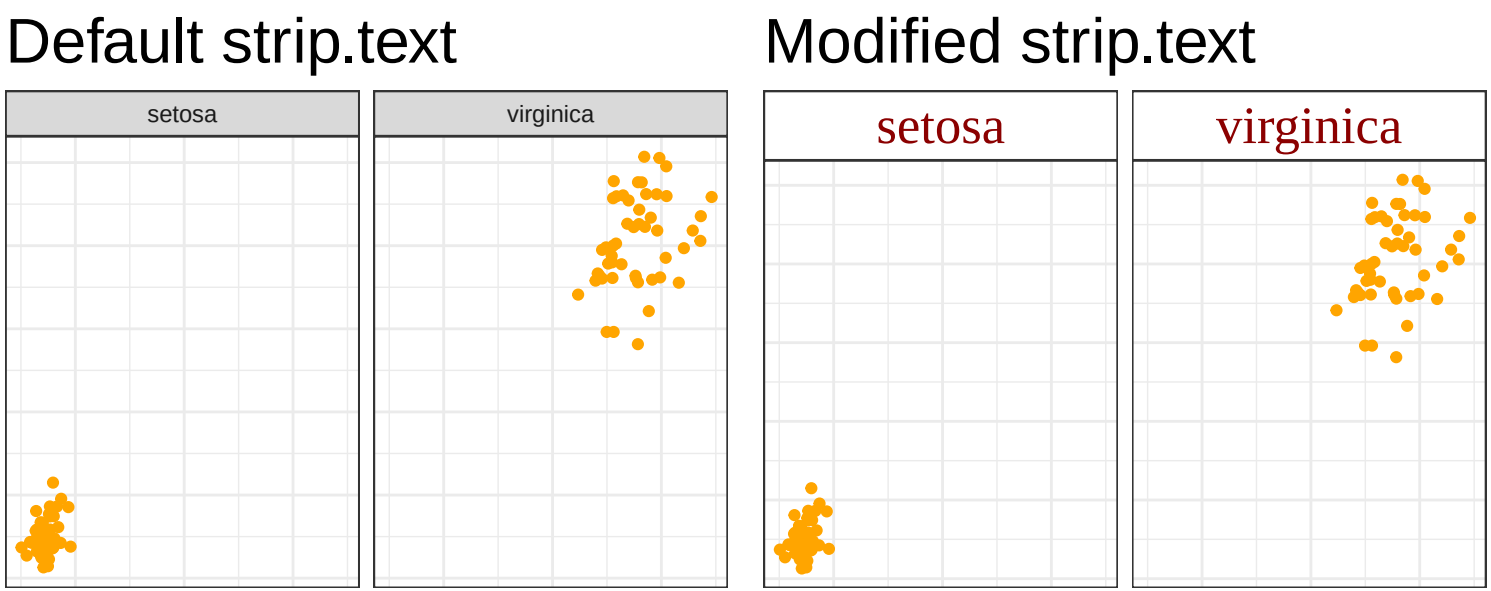


Figure 2: *Left*: Default `strip.text`; *Right*: Font family, color and size are modified of `strip.text`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode to modify strip.text

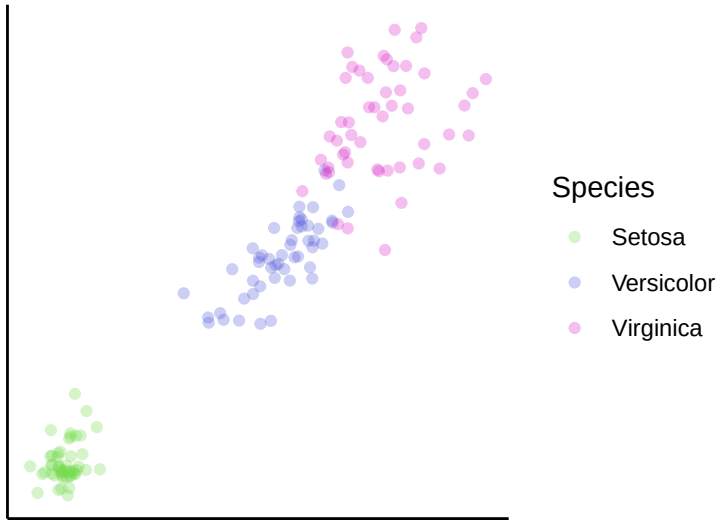
```
# Modify font family, size, and color
theme(strip.text = element_text(
                      family = "serif",
                        size = 20,
                       color = "#8b0000"))
```

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

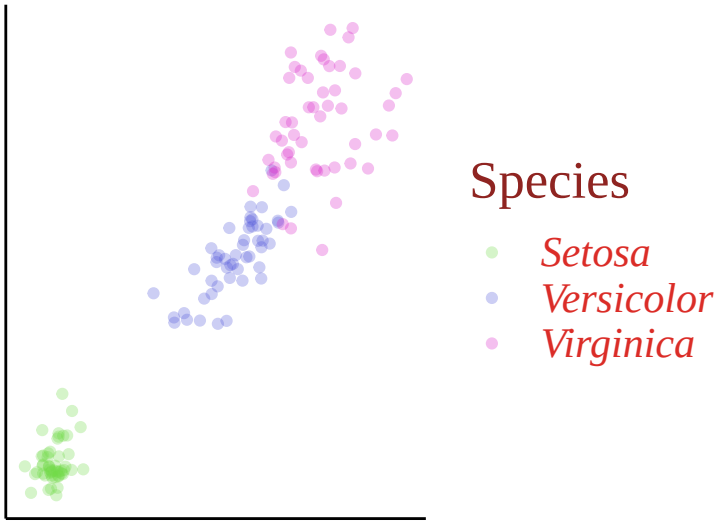
Modified `legend.title` and `legend.text`

We modified the font family, color and size of `legend.title` and `legend.text`.

Default legend.text and legend.title



Modified legend.text and legend.title



Can you spot the problem that still remains with the `legend guide`?

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: modify legend text elements

```
# Modify font family, size, and color
theme( # modify legend.title
legend.title = element_text(
family = "serif", size = 20,
      color = "#8F2421"),

        # modify legend.text
 legend.text = element_text(
 family = "serif", face = "italic",
 size = 16, color = "#DB2C27"))
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Fix legend guide using `guides` layer

- Sometimes we want the "*geoms in the legend*" to display differently to the "*geoms in the plot*".
- This is particularly important when the size of the points is small or the transparency of the points is high.
- To modify legend guides, we can use the `override.aes` parameter of `guide_legend()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Fix legend guide using `guides` layer

- Sometimes we want the "*geoms in the legend*" to display differently to the "*geoms in the plot*".
- This is particularly important when the size of the points is small or the transparency of the points is high.
- To modify legend guides, we can use the `override.aes` parameter of `guide_legend()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Fix legend guide using `guides` layer

- Sometimes we want the "*geoms in the legend*" to display differently to the "*geoms in the plot*".
- This is particularly important when the size of the points is small or the transparency of the points is high.
- To modify legend guides, we can use the `override.aes` parameter of `guide_legend()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example: modify legend guides for clarity

We have increased the size and the value of the alpha attribute of the points used in the legend guide.

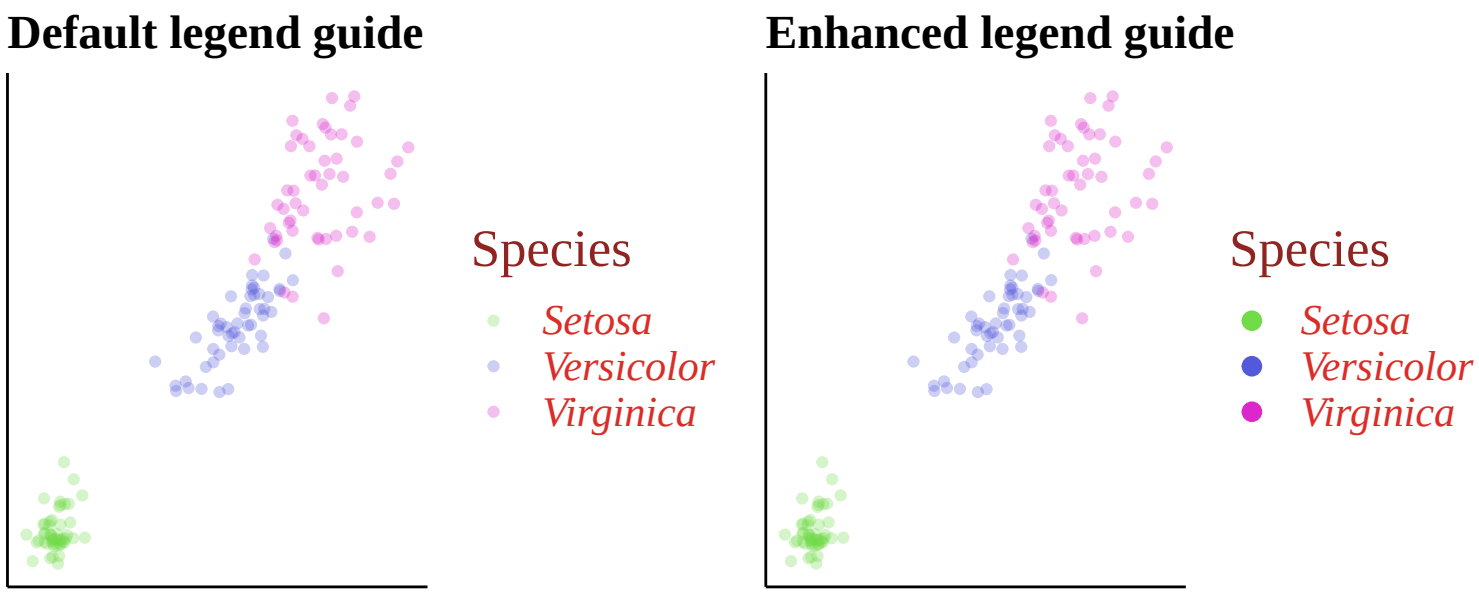


Figure 3: *Left*: default legend guide; *Right*: Legend point size has been increased to 3 and alpha value to 1.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode to modify legend guide

```
# Create new color guide object
col_guide <- guide_legend(
        override.aes = list(alpha = 1,
                            size = 3))

# Modify size and alpha of legend points
ggplot(iris, aes(Petal.Length,
                 Petal.Width,
                 color = Species)) +
# Call geom_point() with alpha 0.30
geom_point(alpha = 0.30) +
# Modify the points in the legend guide
guides(color = col_guide)
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Specify legend position

- One of the key tricks to enhance the visualisation is to be creative with the legend position.
- The easiest way to modify the legend position is to use the `legend.position` argument in `theme()` layer.
- It is easy to put the legend at the left (`legend.position = "left"`), right (`legend.position = "right"`), top (`legend.position = "top"`), or bottom (`legend.position = "bottom"`) of the plot.
- Another useful call is `legend.position = "none"` – it removes the legend from the plot.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Specify legend position

- One of the key tricks to enhance the visualisation is to be creative with the legend position.
- The easiest way to modify the legend position is to use the `legend.position` argument in `theme()` layer.
- It is easy to put the legend at the left (`legend.position = "left"`), right (`legend.position = "right"`), top (`legend.position = "top"`), or bottom (`legend.position = "bottom"`) of the plot.
- Another useful call is `legend.position = "none"` – it removes the legend from the plot.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Specify legend position

- One of the key tricks to enhance the visualisation is to be creative with the legend position.
- The easiest way to modify the legend position is to use the `legend.position` argument in `theme()` layer.
- It is easy to put the legend at the left (`legend.position = "left"`), right (`legend.position = "right"`), top (`legend.position = "top"`), or bottom (`legend.position = "bottom"`) of the plot.
- Another useful call is `legend.position = "none"` – it removes the legend from the plot.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

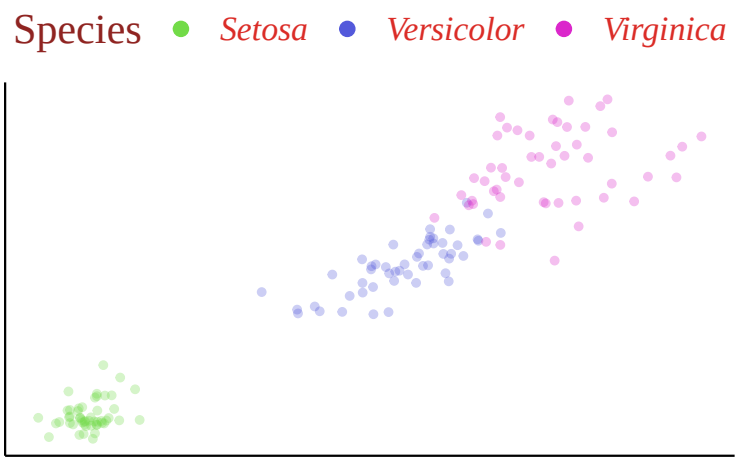
Specify legend position

- One of the key tricks to enhance the visualisation is to be creative with the legend position.
- The easiest way to modify the legend position is to use the `legend.position` argument in `theme()` layer.
- It is easy to put the legend at the left (`legend.position = "left"`), right (`legend.position = "right"`), top (`legend.position = "top"`), or bottom (`legend.position = "bottom"`) of the plot.
- Another useful call is `legend.position = "none"` – it removes the legend from the plot.

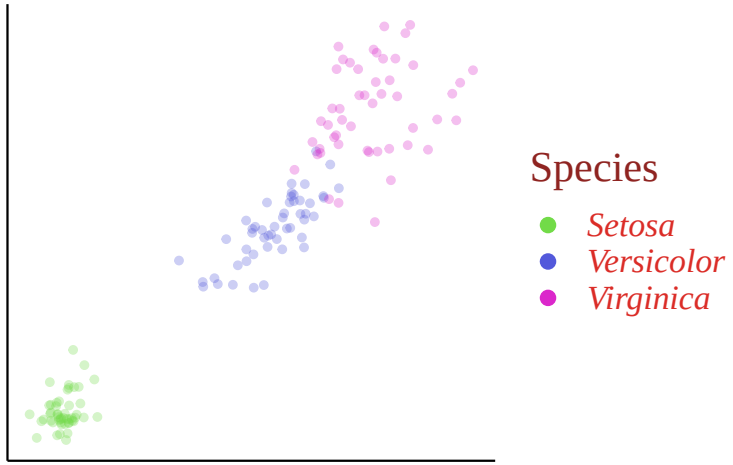
Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example of different legend positions

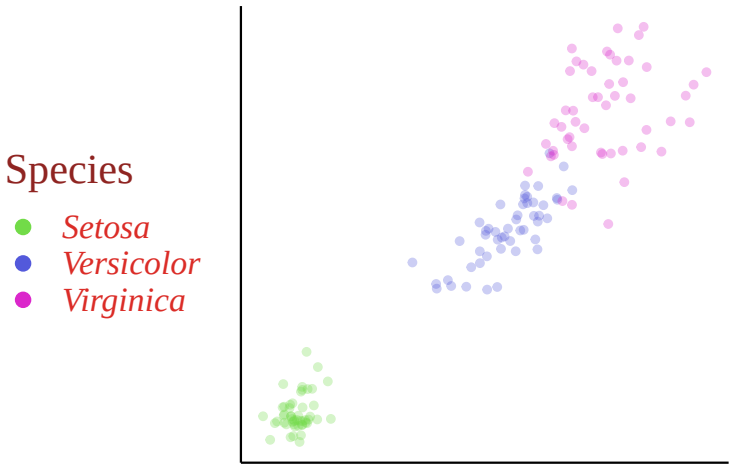
legend.position = "top"



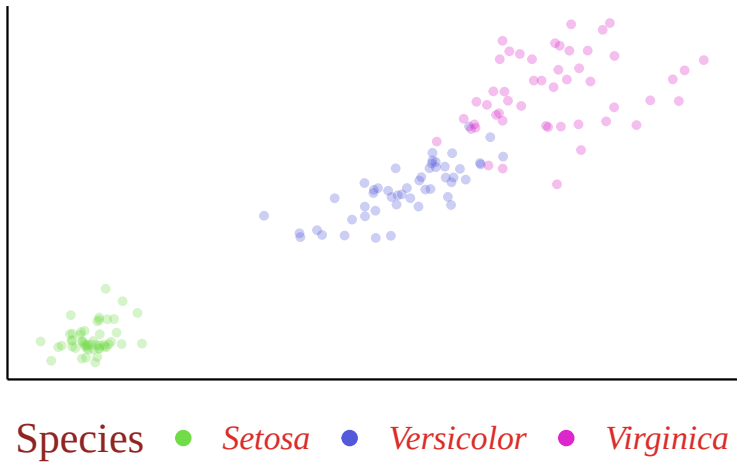
legend.position = "right"



legend.position = "left"



legend.position = "bottom"



Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode to specify legend.position

```
# Legend position at the top
theme(legend.position = "top")

# Legend position at the right
theme(legend.position = "right")

# Legend position at the left
theme(legend.position = "left")

# Legend position at the bottom
theme(legend.position = "bottom")

# Remove legend completely
theme(legend.position = "none")
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Putting legend inside the plot

- In most academic publication, the legends are required to be placed inside the plot.
- This is particularly useful when you have a lot of blank space in your plot.
- This can be achieved by passing a numeric vector with x and y coordinates to the `legend.position` parameter in the `theme()` layer.
- The x and y coordinates represent a relative location in the panel:
  1. `c(0, 0)` is bottom left;
  2. `c(0, 1)` is top left;
  3. `c(1, 0)` is bottom right;
  4. `c(1, 1)` is top right.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Putting legend inside the plot

- In most academic publication, the legends are required to be placed inside the plot.
- This is particularly useful when you have a lot of blank space in your plot.
- This can be achieved by passing a numeric vector with x and y coordinates to the `legend.position` parameter in the `theme()` layer.
- The x and y coordinates represent a relative location in the panel:
  1. `c(0, 0)` is bottom left;
  2. `c(0, 1)` is top left;
  3. `c(1, 0)` is bottom right;
  4. `c(1, 1)` is top right.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Putting legend inside the plot

- In most academic publication, the legends are required to be placed inside the plot.
- This is particularly useful when you have a lot of blank space in your plot.
- This can be achieved by passing a numeric vector with x and y coordinates to the `legend.position` parameter in the `theme()` layer.
- The x and y coordinates represent a relative location in the panel:
  1. `c(0, 0)` is bottom left;
  2. `c(0, 1)` is top left;
  3. `c(1, 0)` is bottom right;
  4. `c(1, 1)` is top right.

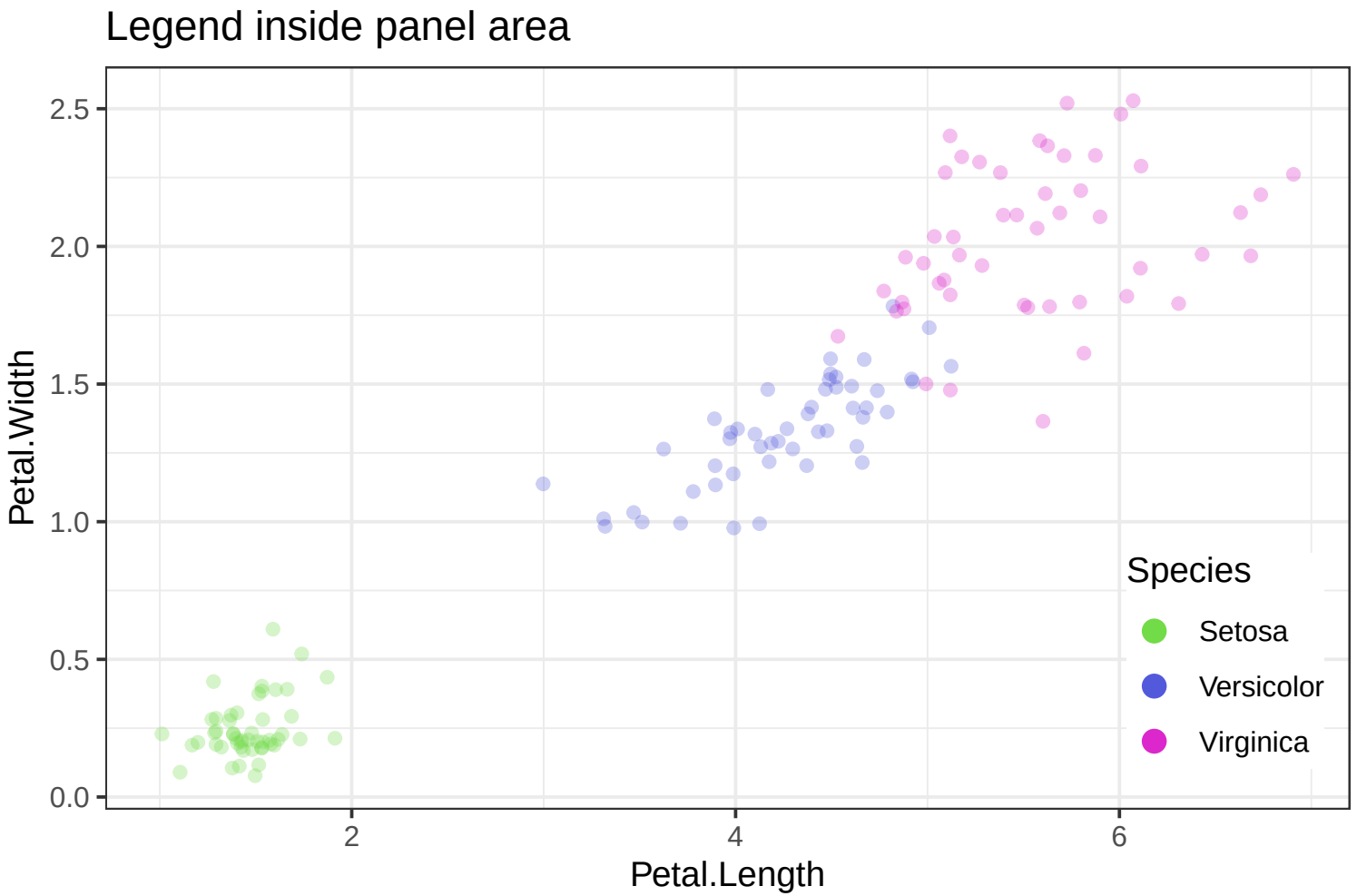
Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Putting legend inside the plot

- In most academic publication, the legends are required to be placed inside the plot.
- This is particularly useful when you have a lot of blank space in your plot.
- This can be achieved by passing a numeric vector with x and y coordinates to the `legend.position` parameter in the `theme()` layer.
- The x and y coordinates represent a relative location in the panel:
  1. `c(0, 0)` is bottom left;
  2. `c(0, 1)` is top left;
  3. `c(1, 0)` is bottom right;
  4. `c(1, 1)` is top right.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example: legend inside the plot



Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: legend inside the plot

```
##################################
# Specify legend position at the #
# bottom right corner of the plot #
##################################
theme(legend.position = c(0.9, 0.2),
##################################
# Remove the legend margin for   #
# an enhanced visualisation      #
##################################
legend.margin = margin(0,0,0,0,  "mm"))
```

Note: Specifying a suitable legend position this way would often require a lot of trial and error on your part.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

line element hierarchy structure

- panel.grid
  - panel.grid.major
    - panel.grid.major.x
    - panel.grid.major.y
  - panel.grid.minor
    - panel.grid.minor.x
    - panel.grid.minor.y
- axis.ticks
  - axis.ticks.x
    - axis.ticks.x.top
    - axis.ticks.x.bottom
- axis.line
  - axis.line.x
    - axis.line.x.top
    - axis.line.x.bottom
  - axis.line.y
    - axis.line.y.left
    - axis.line.y.right
  - axis.ticks.y
    - axis.ticks.y.left
    - axis.ticks.y.right

Figure 4: Use `element_line()` to modify all line elements.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modifying `panel.grid` lines

We changed the color, size and linetype attributes of `panel.grid` lines.

Figure 5: Use `panel.grid` parameter inside `theme()` to modify both major and minor panel grid-lines.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: modify panel.grid lines

```
##########################################
# Specify panel grid line attributes#
##########################################
mygrid <- element_line(color = "blue",
                       size = 1,
                       linetype = "dashed")
##########################################
# Use panel.grid parameter inside    #
# theme() layer to modify grid lines#
##########################################
theme(panel.grid = mygrid)
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modify height of `axis.ticks`

- Because axis ticks are treated as line elements, these can be modified using `element_line()` like any other line elements.
- However, to modify the height of the axis ticks, we need to use the `axis.ticks.length` parameter of the `theme()` layer.
- The `axis.ticks.length` parameter accepts an object of the class `unit()`.
- The first argument of `unit()` is a length value, and the second argument is a length unit. For example, to specify 1 cm, we shall use `unit(1, "cm")`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modify height of `axis.ticks`

- Because axis ticks are treated as line elements, these can be modified using `element_line()` like any other line elements.
- However, to modify the height of the axis ticks, we need to use the `axis.ticks.length` parameter of the `theme()` layer.
- The `axis.ticks.length` parameter accepts an object of the class `unit()`.
- The first argument of `unit()` is a length value, and the second argument is a length unit. For example, to specify 1 cm, we shall use `unit(1, "cm")`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modify height of `axis.ticks`

- Because axis ticks are treated as line elements, these can be modified using `element_line()` like any other line elements.
- However, to modify the height of the axis ticks, we need to use the `axis.ticks.length` parameter of the `theme()` layer.
- The `axis.ticks.length` parameter accepts an object of the class `unit()`.
- The first argument of `unit()` is a length value, and the second argument is a length unit. For example, to specify 1 cm, we shall use `unit(1, "cm")`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modify height of `axis.ticks`

- Because axis ticks are treated as line elements, these can be modified using `element_line()` like any other line elements.
- However, to modify the height of the axis ticks, we need to use the `axis.ticks.length` parameter of the `theme()` layer.
- The `axis.ticks.length` parameter accepts an object of the class `unit()`.
- The first argument of `unit()` is a length value, and the second argument is a length unit. For example, to specify 1 cm, we shall use `unit(1, "cm")`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Modifying `axis.ticks.length`

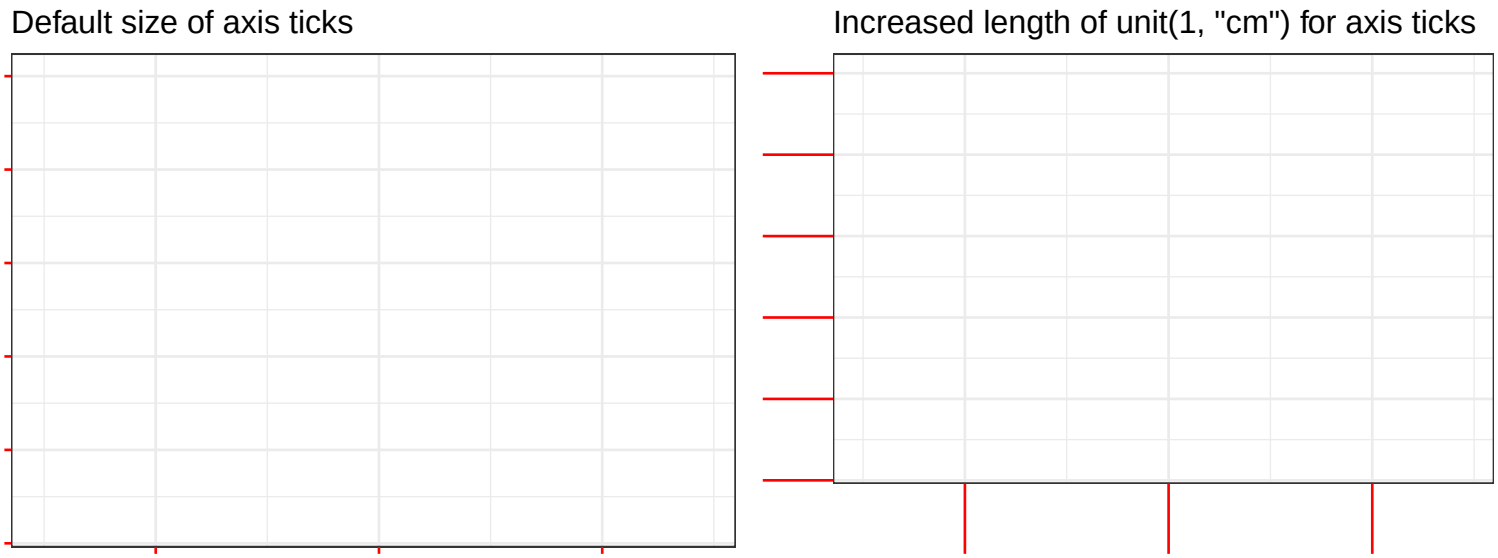


Figure 6: `axis.ticks.length = unit(1, "cm")` is used inside the `theme()` layer to modify the length of axis ticks.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

`rect` element hierarchy structure

- `legend.background`
- `legend.key`
- `legend.box.background`
- `panel.background`
- `panel.border`
- `plot.background`
- `strip.background`
  - `strip.background.x`
  - `strip.background.y`

Figure 7: Use `element_rect()` to modify all rectangular elements.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Create customised theme – Curtin theme

We have created a customised theme inspired by the colors prominent in the Curtin University logo.

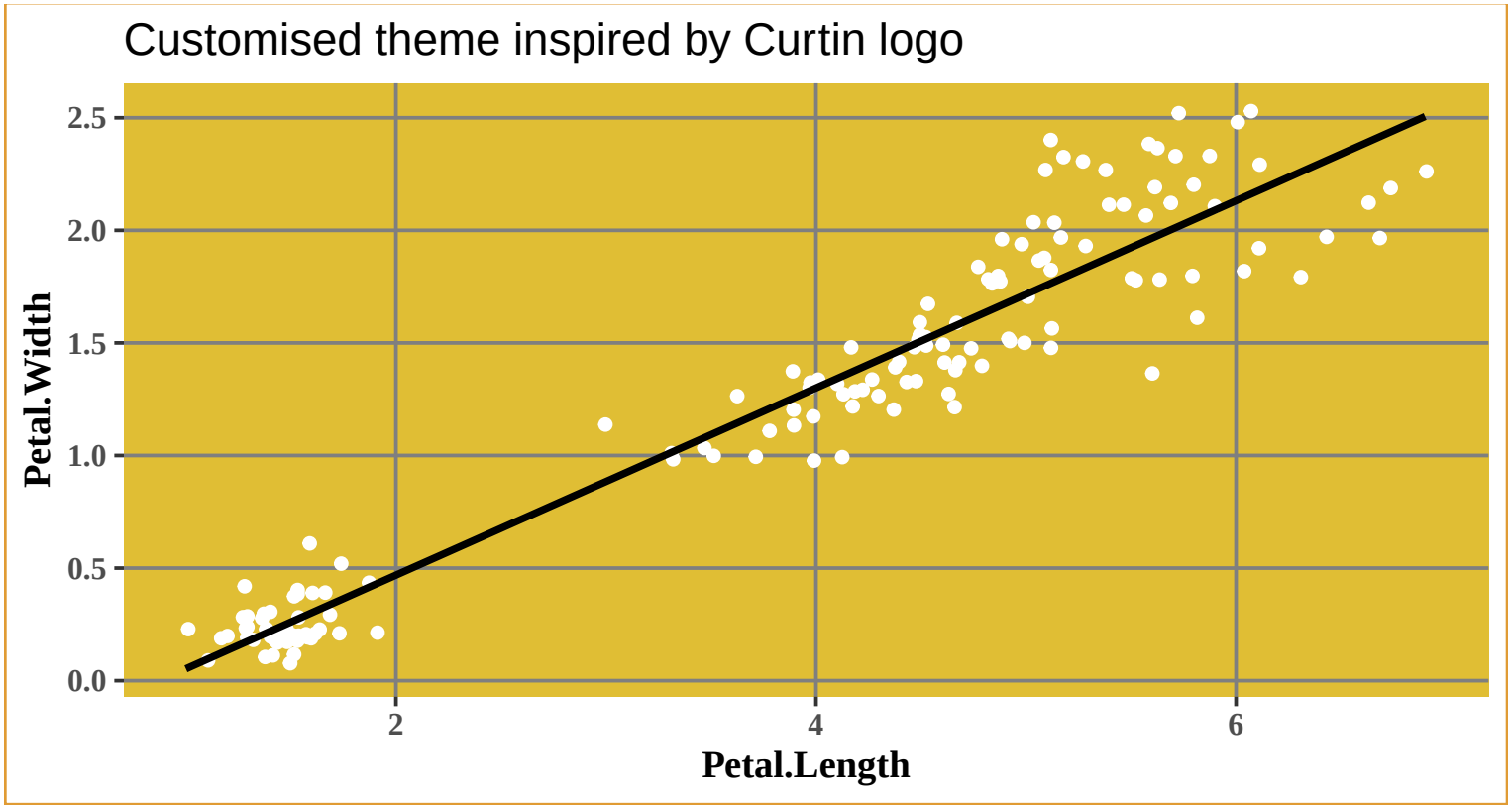


Figure 8: Relationship between iris Petal.Width and Petal.Length, fitted using the linear regression model.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: create Curtin theme

```
# Create custom theme
curtin_theme <-
theme(plot.background =
        element_rect(color = "#E09B34"),
      panel.background =
        element_rect(fill = "#E0BE34"),
      panel.grid.major =
        element_line(color = "gray50"),
      panel.grid.minor = element_blank(),
      axis.text =
        element_text(face = "bold",
                     family = "serif"),
      axis.title =
        element_text(face = "bold",
                     family = "serif"))
```

Curtin University

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

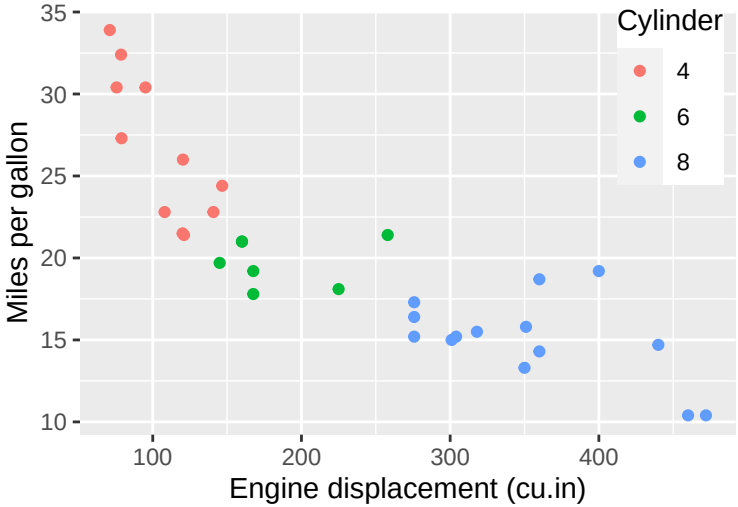
RCode: use Curtin theme to plot

```
# Plot Petal.Width vs Petal.Length
ggplot(iris,
       aes(x = Petal.Length,
           y = Petal.Width)) +
# Add jittered points
geom_point(position =
             position_jitter(seed = 123),
             color = "white",
             size = 1.3) +
# Add the line of best fit
geom_smooth(method = "lm",
             se = FALSE,
             color = "black") +
# Add Curtin theme object
curtin_theme
```

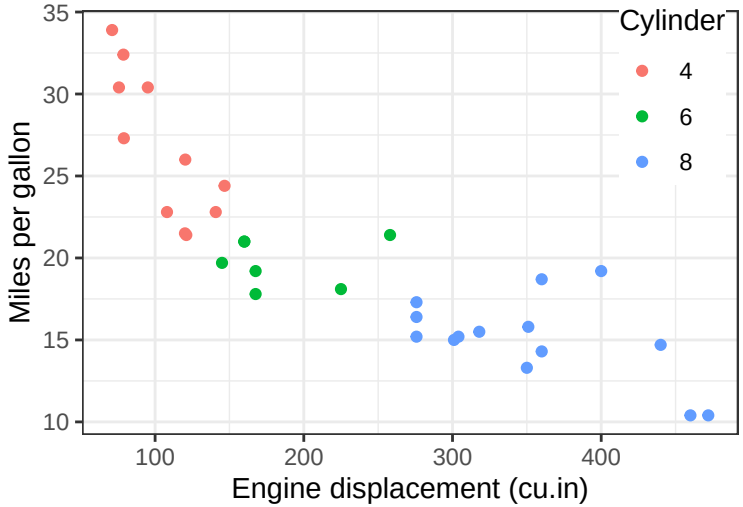

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Use ggplot2 in-built themes

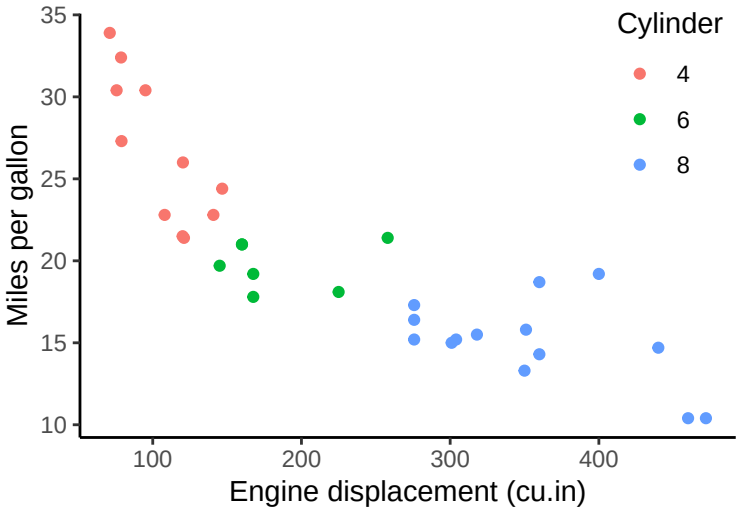
Default: theme_grey()



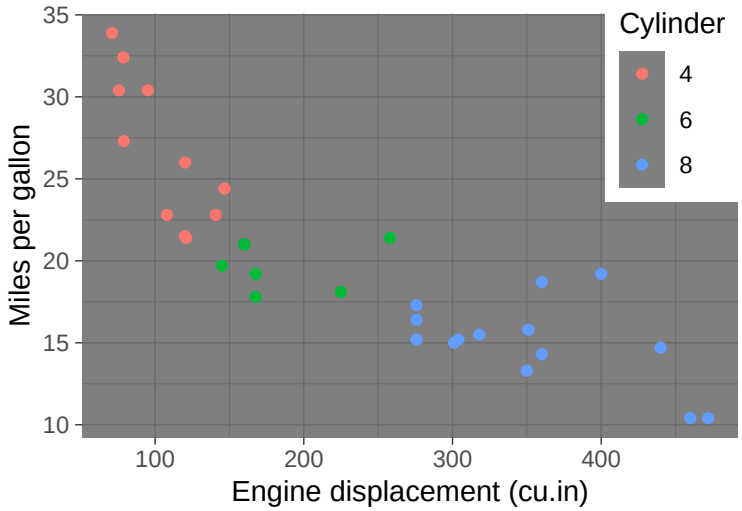
theme_bw()



theme_classic()

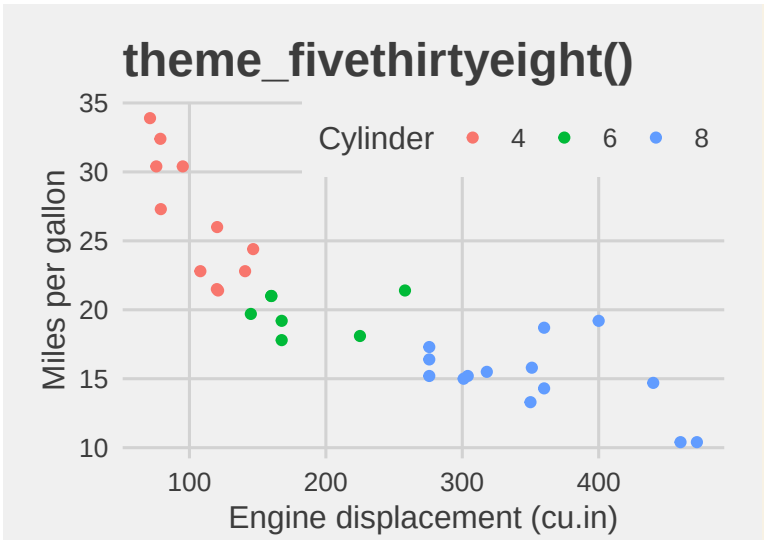


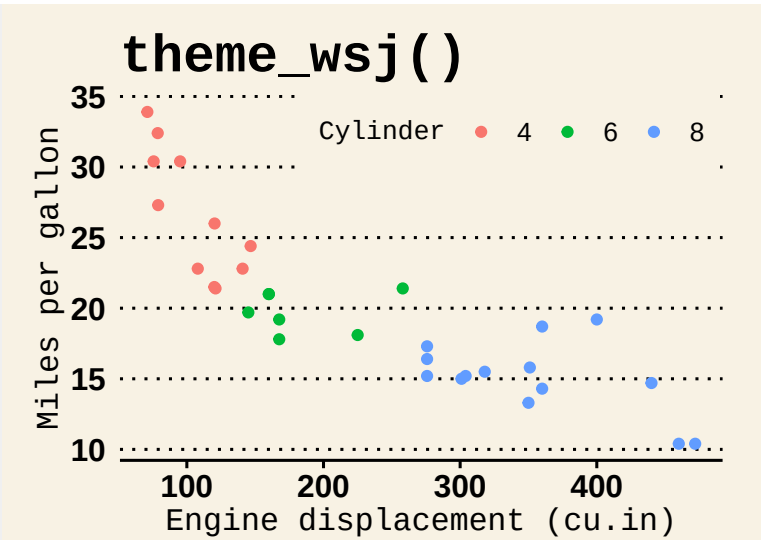
theme_dark()

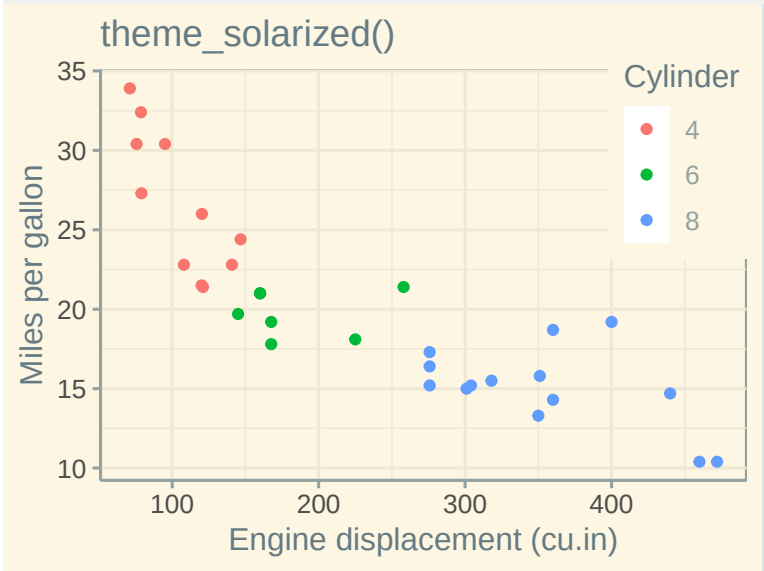


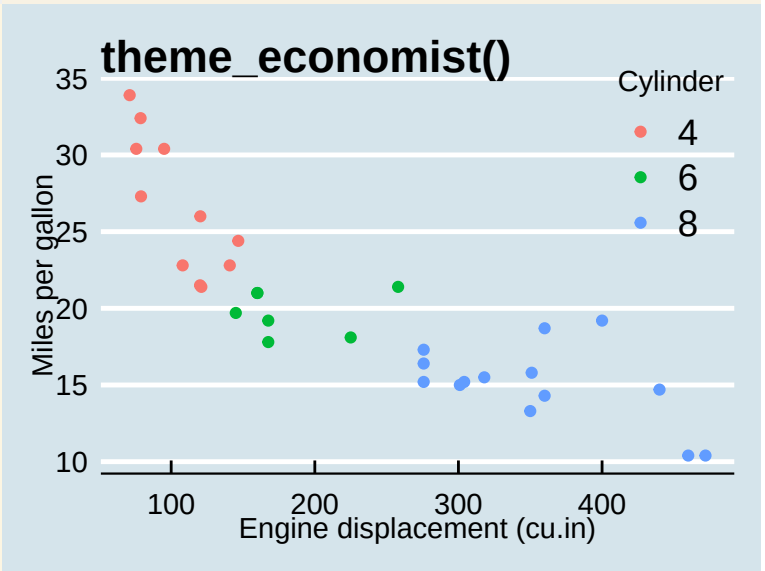
Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Use ggthemes package for more themes









Curtin University

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Important `scale_*_*()` class of functions

- We have seen that `aes()` is used to map variables to visual aesthetics. The `scale` functions are used to modify or enhance this mapping.

- For every aesthetic, we have corresponding `scale` functions. For example, for the `x` and `y` aesthetics, the corresponding scales functions are of the forms `scale_x_*()` and `scale_y_*()`, respectively.

- The last $\star$ corresponds to the type of variable that is mapped to the visual aesthetic. For example, if a continuous variable is mapped to the `x` axis, then the corresponding scale function is `scale_x_continuous()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Important `scale_*_*()` class of functions

- We have seen that `aes()` is used to map variables to visual aesthetics. The `scale` functions are used to modify or enhance this mapping.

- For every aesthetic, we have corresponding `scale` functions. For example, for the `x` and `y` aesthetics, the corresponding scales functions are of the forms `scale_x_*()` and `scale_y_*()`, respectively.

- The last $\star$ corresponds to the type of variable that is mapped to the visual aesthetic. For example, if a continuous variable is mapped to the `x` axis, then the corresponding scale function is `scale_x_continuous()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Important `scale_*_*()` class of functions

- We have seen that `aes()` is used to map variables to visual aesthetics. The `scale` functions are used to modify or enhance this mapping.
- For every aesthetic, we have corresponding `scale` functions. For example, for the `x` and `y` aesthetics, the corresponding scales functions are of the forms `scale_x_*()` and `scale_y_*()`, respectively.
- The last $\star$ corresponds to the type of variable that is mapped to the visual aesthetic. For example, if a continuous variable is mapped to the `x` axis, then the corresponding scale function is `scale_x_continuous()`.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Useful arguments of `scale_x_continuous()`

- `trans`: name of the transformation, e.g., `"log10"`, `"sqrt"`, and `"log"`.
- `breaks`: a numeric vector of positions for axis labels to appear.
- `labels`: a character vector giving labels (must be same length as `breaks`).
- `position`: the position of the axis, e.g., `"top"` or `"bottom"` for the `x`-axis.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example: Diamond price vs log 10(carat)

We used `scale_x_continuous()` to transform the x-axis into log 10 scale, and used the `scale_y_continuous()` to change the y-axis labels.

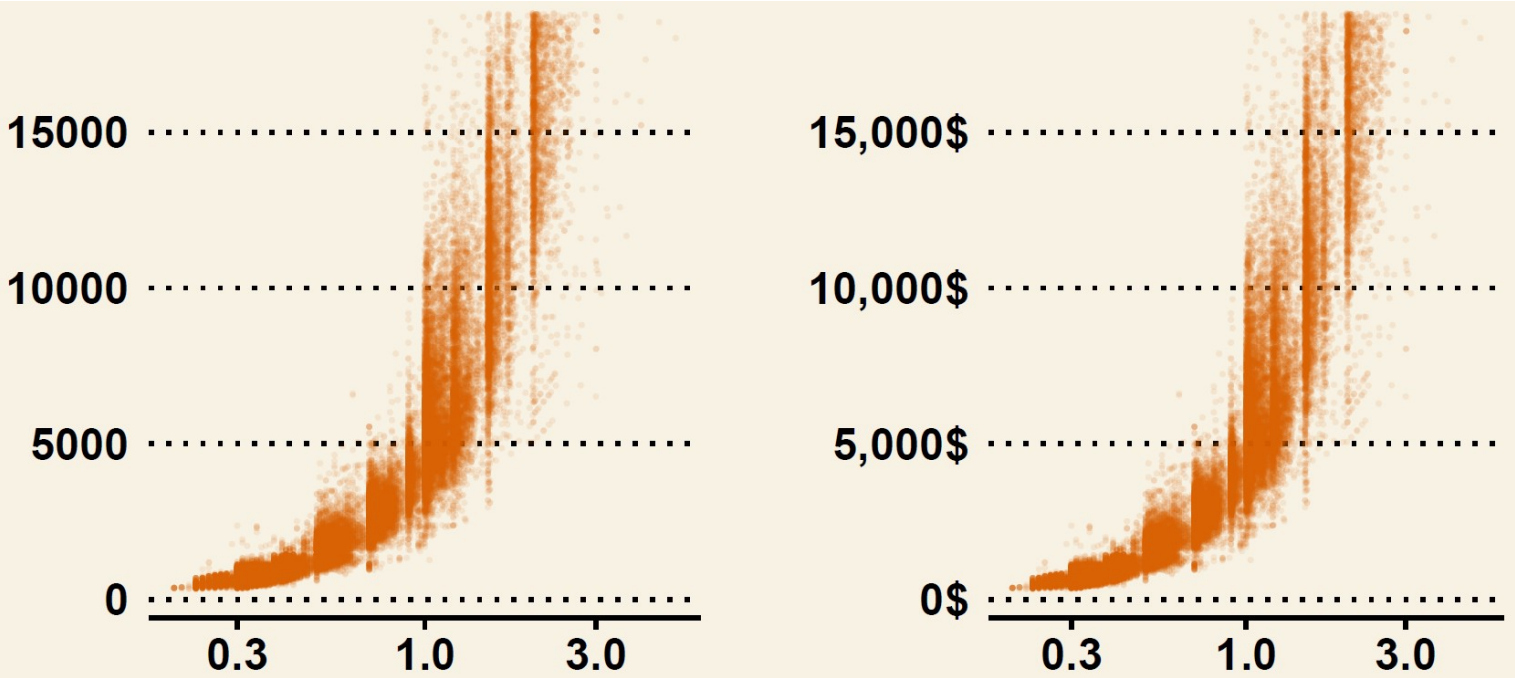


Figure 9: *Left*: Default y-axis labels; *Right*: Dollar sign and comma are added to the price for visual enhancement

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: scale_y_continuous() example

```
# Plot diamond price vs log10(carat)
ggplot(diamonds, aes(carat, price)) +
geom_point(size = 0.1,
           alpha=0.10,
           color = "#D55E00") +
# Change x-axis to log-scale
scale_x_continuous(
    "log10(carat)",
    trans = "log10") +
# Change y-axis labels
scale_y_continuous(
    breaks = c(0, 5000, 10000, 15000),
    labels = c("0$", "5,000$",
               "10,000$", "15,000$")) +
theme_wsj()
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

scale_color_⋆() functions

- scale_color_brewer() and scale_color_manual() are two useful functions that allow us to change the default color produced by mapping to the color aesthetics.
- The scale_color_brewer() provides sequential, diverging and qualitative color schemes from the R-package RColorBrewer.
- Modify the type and palette argument inside scale_color_brewer():
  - type – one of "seq" (sequential), "div" (divergent), and "qual" (qualitative).
  - palette – If a string, will use the named palette. If a number, will index into the list of palettes of appropriate type.
    - Diverging: BrBg, PiYG, RdYlBu, RdYlGr, Spectral.
    - Qualitative: Accent, Set1, Set2, Pastel1.
    - Sequential: Blues, YlOrRd, YlGnBu, Purples.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

scale_color_⋆() functions

- scale_color_brewer() and scale_color_manual() are two useful functions that allow us to change the default color produced by mapping to the color aesthetics.
- The scale_color_brewer() provides sequential, diverging and qualitative color schemes from the R-package RColorBrewer.
- Modify the type and palette argument inside scale_color_brewer():
  - type – one of "seq" (sequential), "div" (divergent), and "qual" (qualitative).
  - palette – If a string, will use the named palette. If a number, will index into the list of palettes of appropriate type.
    - Diverging: BrBg, PiYG, RdYlBu, RdYlGr, Spectral.
    - Qualitative: Accent, Set1, Set2, Pastel1.
    - Sequential: Blues, YlOrRd, YlGnBu, Purples.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

scale_color_⋆() functions

- scale_color_brewer() and scale_color_manual() are two useful functions that allow us to change the default color produced by mapping to the color aesthetics.
- The scale_color_brewer() provides sequential, diverging and qualitative color schemes from the R-package RColorBrewer.
- Modify the type and palette argument inside scale_color_brewer():
  - type – one of "seq" (sequential), "div" (divergent), and "qual" (qualitative).
  - palette – If a string, will use the named palette. If a number, will index into the list of palettes of appropriate type.
    - Diverging: BrBg, PiYG, RdYlBu, RdYlGr, Spectral.
    - Qualitative: Accent, Set1, Set2, Pastel1.
    - Sequential: Blues, YlOrRd, YlGnBu, Purples.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example: `scale_color_brewer()`

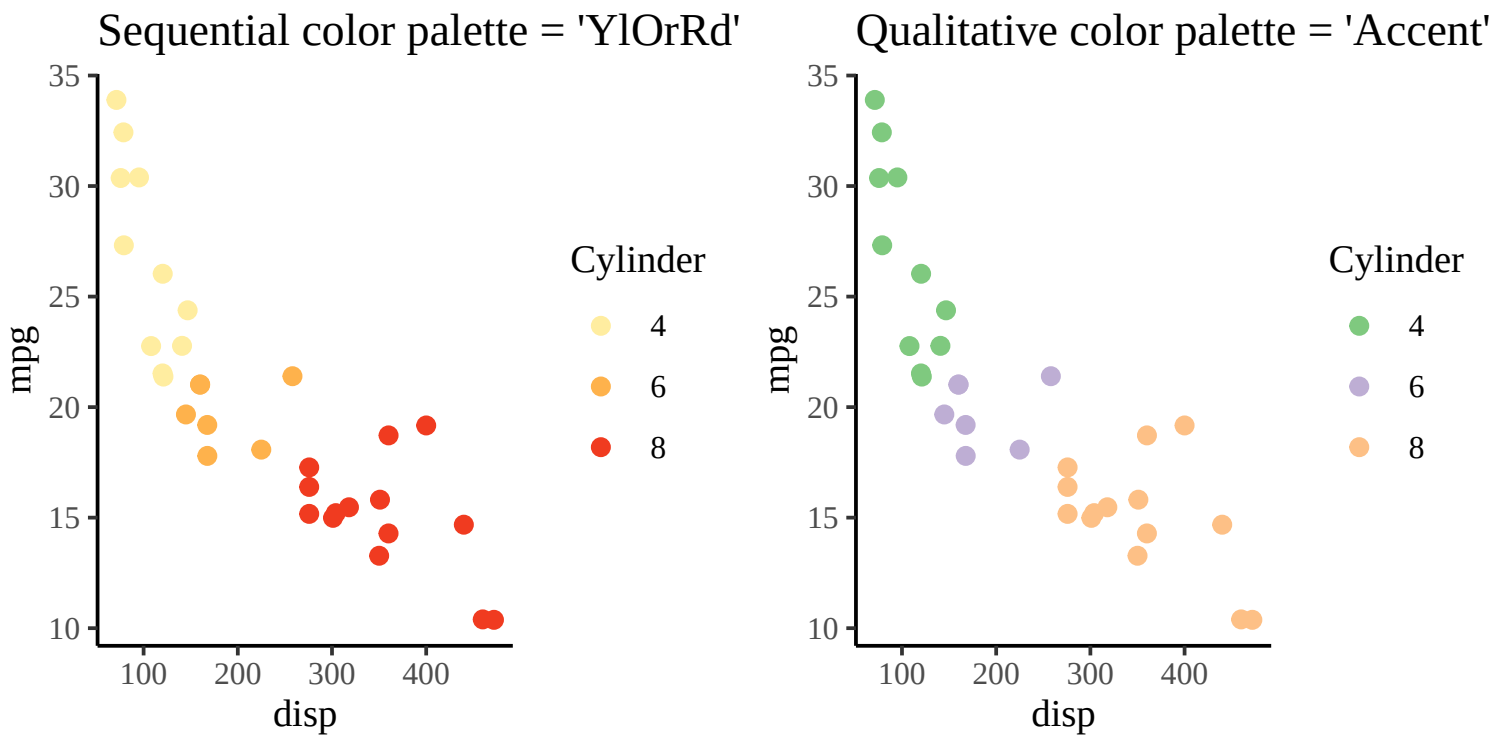


Figure 10: *Left*: Sequential color palette used; *Right*: Qualitative color palette used

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: scale_color_brewer() example

```
baseplt <- ggplot(mtcars,aes(disp, mpg,
                color = factor(cyl))) +
              geom_point()
# Use a sequential palette
baseplt +
scale_color_brewer("Cylinder",
                   type = "seq",
                   palette = "YlOrRd")
# Use a qualitative palette
baseplt +
scale_color_brewer("Cylinder",
                  type = "qual",
                  palette = "Accent")
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

`scale_color_manual()` function

- `scale_color_manual()` provides the most flexibility, as it allows you to create own discrete color scale.

- The two most important parameters are `values` and `breaks`:
  - `values`: for a categorical variable, provide a vector specifying colors for each category; for a continuous variable, provide a vector specifying colors for number of categories created by the number of `breaks`;
  - `breaks`: levels for which a color band is specified.

- Similarly, for `fill` aesthetics, we have `scale_fill_brewer()` and `scale_fill_manual()` functions.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

`scale_color_manual()` function

- `scale_color_manual()` provides the most flexibility, as it allows you to create own discrete color scale.
- The two most important parameters are `values` and `breaks`:
  - `values`: for a categorical variable, provide a vector specifying colors for each category; for a continuous variable, provide a vector specifying colors for number of categories created by the number of `breaks`;
  - `breaks`: levels for which a color band is specified.
- Similarly, for `fill` aesthetics, we have `scale_fill_brewer()` and `scale_fill_manual()` functions.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

`scale_color_manual()` function

- `scale_color_manual()` provides the most flexibility, as it allows you to create own discrete color scale.

- The two most important parameters are `values` and `breaks`:
  - `values`: for a categorical variable, provide a vector specifying colors for each category; for a continuous variable, provide a vector specifying colors for number of categories created by the number of `breaks`;
  - `breaks`: levels for which a color band is specified.

- Similarly, for `fill` aesthetics, we have `scale_fill_brewer()` and `scale_fill_manual()` functions.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Example: `scale_color_manual()`

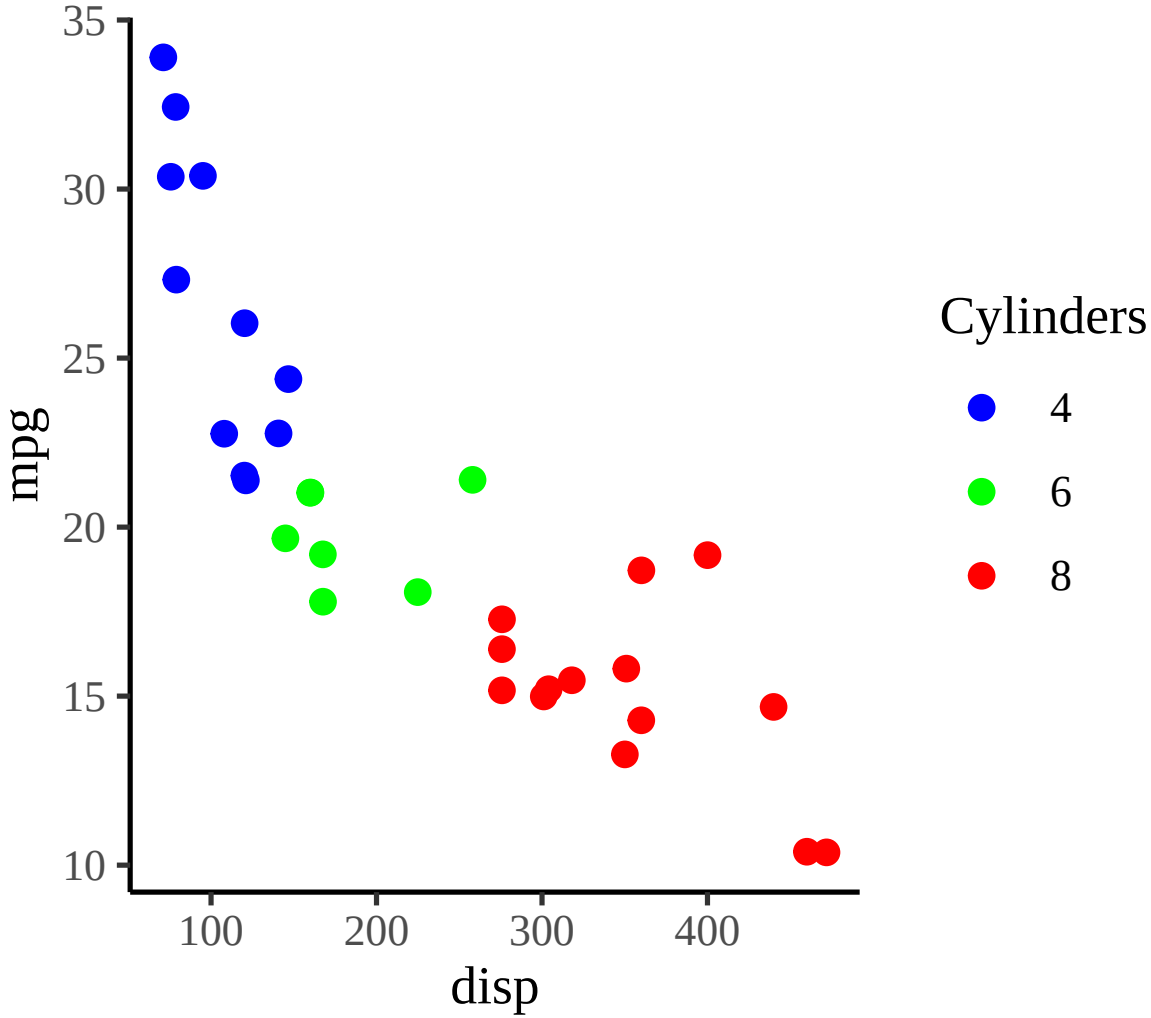


Figure 11: User-specified discrete color scale for three cylinder numbers in the mtcars dataset.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

RCode: scale_color_manual() example

```
baseplt <- ggplot(mtcars,aes(disp, mpg,
           color = factor(cyl))) +
           geom_point()

# Define a named vector with your
    favourite colors
col_vec <- c("4" = "blue",
             "6" = "green",
             "8" = "red")

# Use your favourite colors
baseplt +
 scale_color_manual("Cylinders",
                    values = col_vec)
```


Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary

Summary

- Theme elements are all non-data ink in the plot.
- Use `element_text()`, `element_line()`, and `element_rect()` to modify three main theme elements.
- You can specify plot title, tag, caption, legend title, axis title in the `labs()` function.
- Use `override.aes` parameter of `guide_legend()` to modify legend guides.
- You can build custom themes or use many in-built themes from different R-packages.
- Use `scale` functions to modify the default aesthetic mappings.
- Use `scale_color_manual()` to select customised colors for the plot.

Outline
Quick review
Deep dive into themes
Text elements
Line elements
Rectangular elements
Custom theme creation
Ggplot2 in-built themes and others
Important scale functions
Summary