Chapter 8 : Themes

```
library(ggplot2)
library(gridExtra)
```

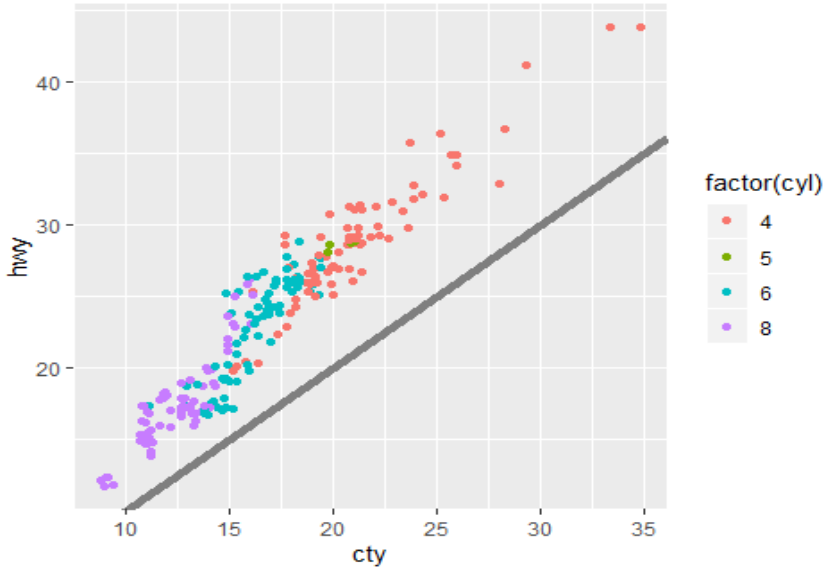
8.1 Introduction

give you control over things likes fonts, ticks, panel strips, and backgrounds. you use code like *plot + theme(element.name = element function())*.

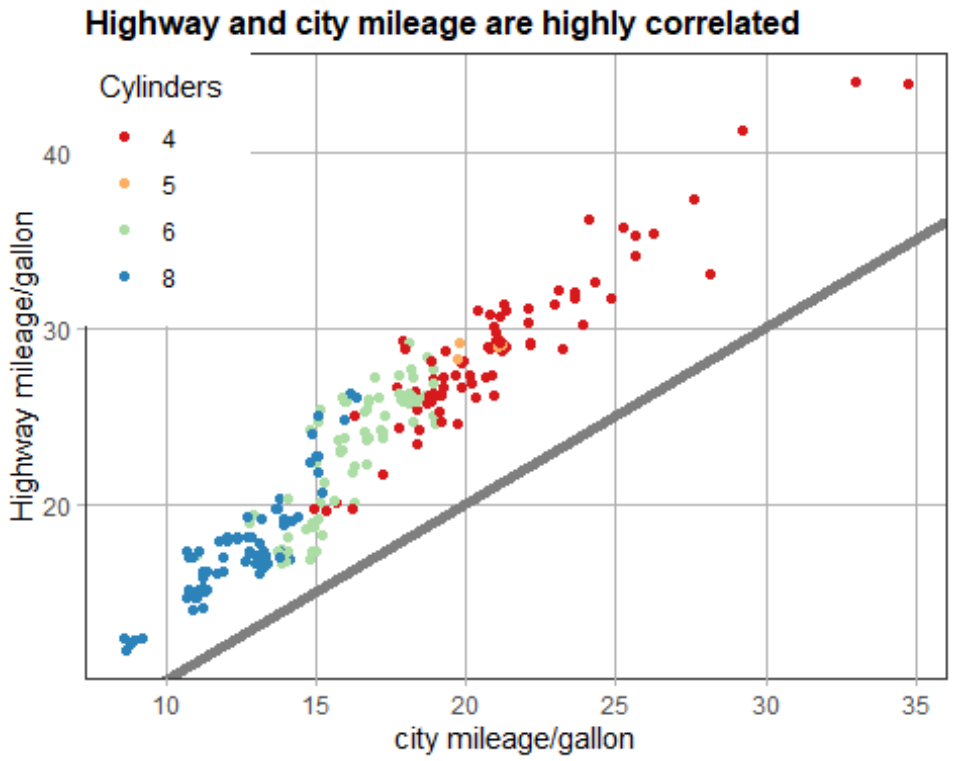
Theming system is composed of 4 main components.

- Theme **elements** specify the non-data elements.For example, the *plot.title* element controls the appearance of the plot title; *axis.ticks.x*, the ticks on the x axis;
- Each element is associated with an **element function** which describes the visual properties of the element. For example, *element_text()* sets the font size, colour and face of text elements like *plot.title*.
- The *theme()* function which allows you to override the default theme elements by calling element functions, like *theme(plot.title = element text(colour = "red")).*
- Complete **themes**, like *theme_grey()* set all of the theme elements to values designed to work together harmoniously.

```
base <- ggplot(mpg, aes(cty, hwy, color = factor(cyl))) +
  geom_jitter() +
  geom_abline(colour = "grey50", size = 2)
base
```



```r
base +
  labs(
    x = "city mileage/gallon",
    y = "Highway mileage/gallon",
    colour = "Cylinders",
    title = "Highway and city mileage are highly correlated"
  ) +
  scale_color_brewer(type = "seq", palette = "Spectral") +
  theme_bw() +
  theme(
    plot.title = element_text(face = "bold", size = 12),
    legend.background = element_rect(fill = "white", size = 4, colour = "whit
e"),
    legend.justification = c(0, 1),
    legend.position = c(0, 1),
    axis.ticks = element_line(colour = "grey70", size = 0.2),
    panel.grid.major = element_line(colour = "grey70", size = 0.2),
    panel.grid.minor = element_blank()
  )
```



8.2 Complete Themes

There are 7 other themes! (default : *theme_grey()*)

- *theme_bw()* : a variation on *theme_grey()* that uses a white background and thin grey grid lines.

- *theme_linedraw()* : A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawing.

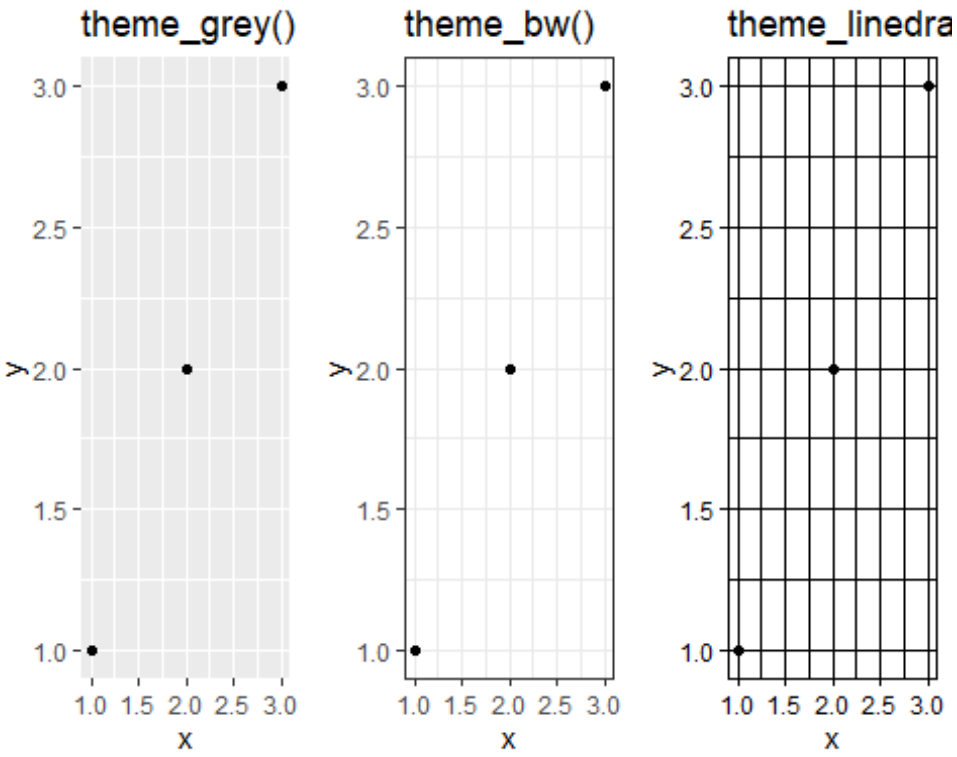
- *theme_light()* : similar to *theme_linedraw()* but with light grey lines and axes, to direct more attention towards the data.

- *theme_dark()* : the dark cousin of *theme_light()*, with similar line sizes but a dark background. Useful to make thin coloured lines pop out.

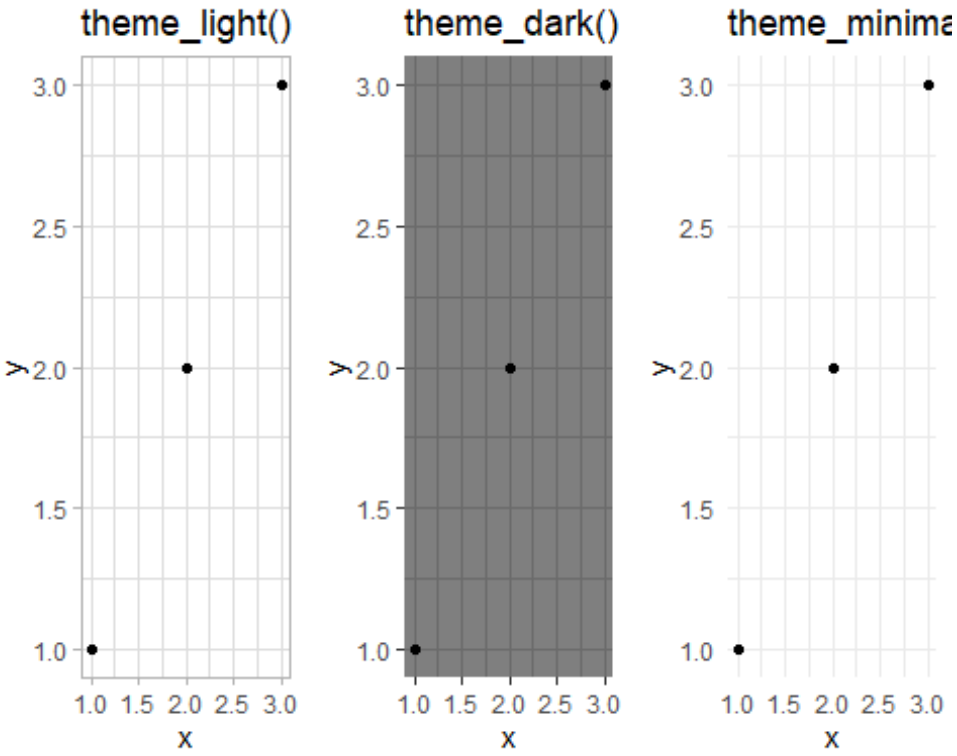
- *theme_minimal()* : A minimalistic theme with no background annotations.

- *theme_classic()* : A classic-looking theme, with x and y axis lines and no gridlines. = *theme_void()*: A completely empty theme.

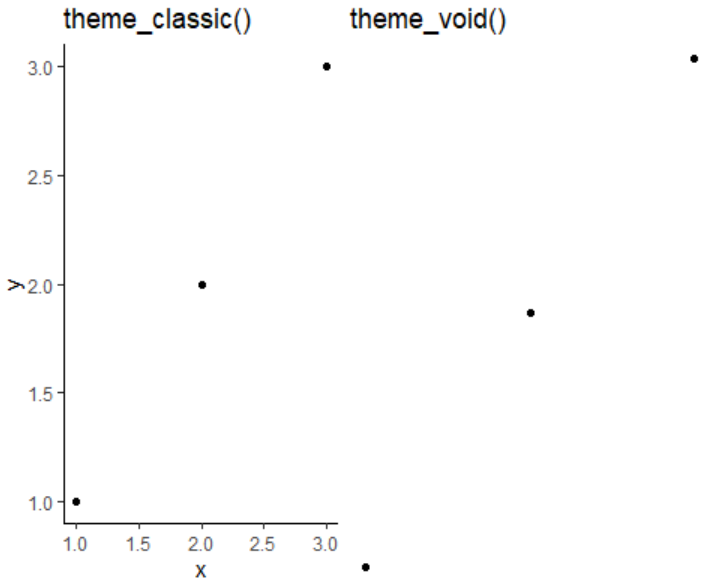
```
df <- data.frame(x = 1:3, y = 1:3)
base <- ggplot(df, aes(x, y)) + geom_point()

b1 = base + theme_grey() + ggtitle("theme_grey()")
b2 = base + theme_bw() + ggtitle("theme_bw()")
b3 = base + theme_linedraw() + ggtitle("theme_linedraw()")
b4 = base + theme_light() + ggtitle("theme_light()")
b5 = base + theme_dark() + ggtitle("theme_dark()")
b6 = base + theme_minimal() + ggtitle("theme_minimal()")
b7 = base + theme_classic() + ggtitle("theme_classic()")
b8 = base + theme_void() + ggtitle("theme_void()")

grid.arrange(b1,b2,b3, ncol = 3)
```




```
grid.arrange(b4,b5,b6, ncol = 3)
```



```
grid.arrange(b7,b8, ncol = 2)
```

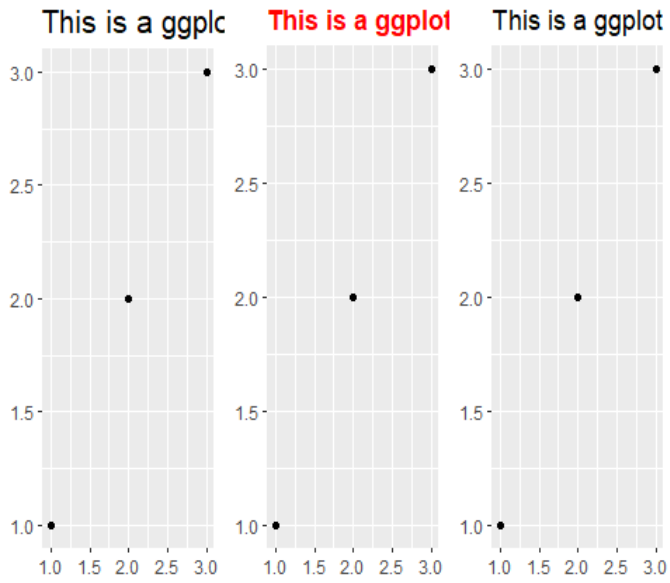


8.3 Modifying Theme Components

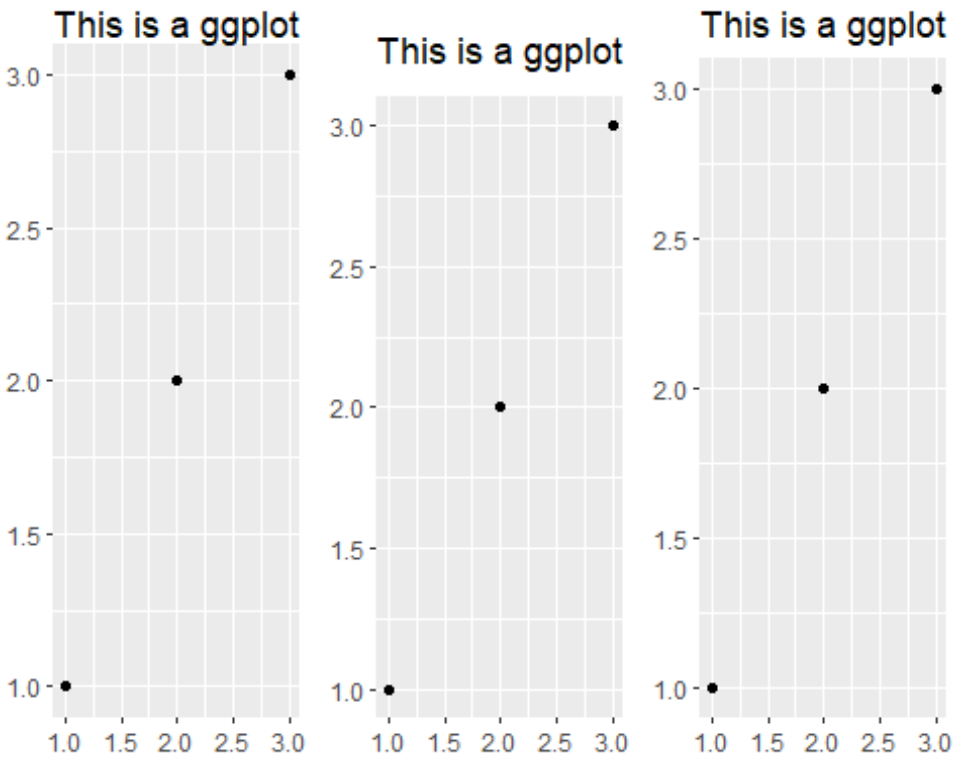
There are 4 basic types of built-in element functions : text, lines, rectangles, blank

- *element_text()* draws labels and headings.

```
base_t <- base + labs(title = "This is a ggplot") + xlab(NULL) + ylab(NULL)
b1 = base_t + theme(plot.title = element_text(size = 16))
b2 = base_t + theme(plot.title = element_text(face = "bold", colour = "red"))
b3 = base_t + theme(plot.title = element_text(hjust = 1))
grid.arrange(b1,b2,b3,ncol = 3)
```

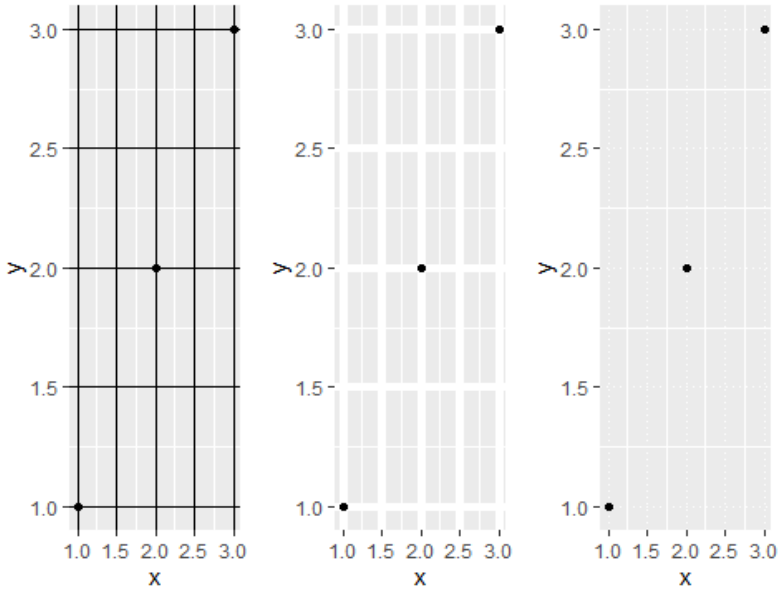


```
# The margins here look asymmetric because there are also plot margins
b1 = base_t + theme(plot.title = element_text(margin = margin()))
b2 = base_t + theme(plot.title = element_text(margin = margin(t = 10, b = 10
)))
b3 = base_t + theme(axis.title.y = element_text(margin = margin(r = 10)))

grid.arrange(b1,b2,b3,ncol = 3)
```



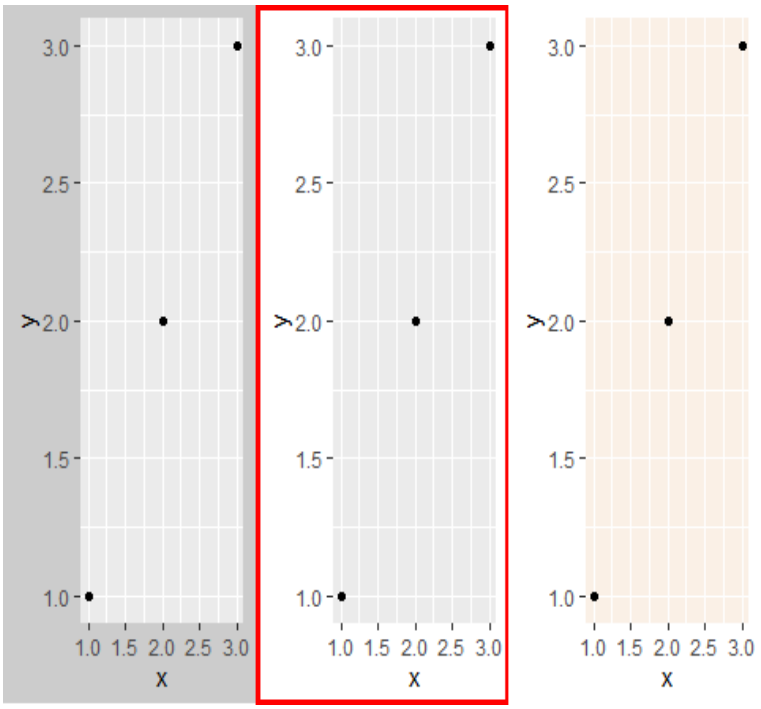
- *element_line()* draws lines parameterised by colour, size and linetype

```
b1 = base + theme(panel.grid.major = element_line(colour = "black"))
b2 = base + theme(panel.grid.major = element_line(size = 2))
b3 = base + theme(panel.grid.major = element_line(linetype = "dotted"))

grid.arrange(b1,b2,b3,ncol = 3)
```



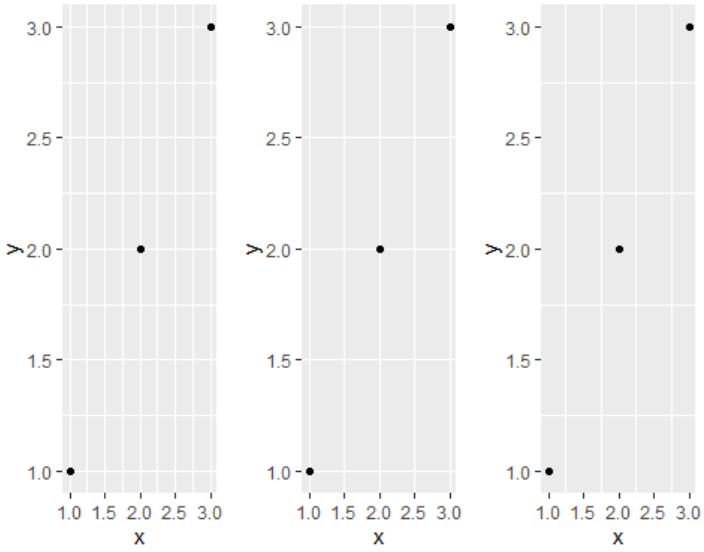
- *element_rect()* draws rectangles, mostly used for backgrounds, parameterised by *fill* colour and border *colour*, *size* and *linetype*.

```
b1 = base + theme(plot.background = element_rect(fill = "grey80", colour = NA
))
b2 = base + theme(plot.background = element_rect(colour = "red", size = 2))
b3 = base + theme(panel.background = element_rect(fill = "linen"))

grid.arrange(b1,b2,b3,ncol = 3)
```

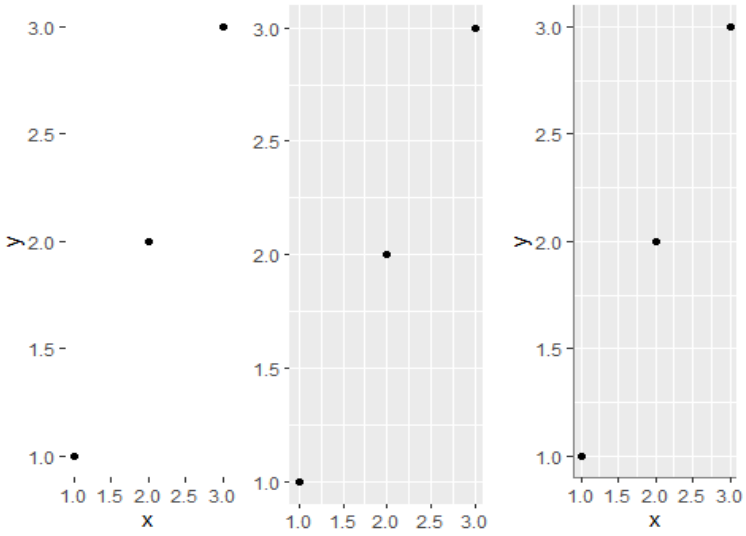


- *element_blank()* draws nothing.

```
b1 = base
b2 = base + theme(panel.grid.minor = element_blank())
b3 = base + theme(panel.grid.major = element_blank())

grid.arrange(b1,b2,b3,ncol = 3)
```




```
b1 = base + theme(panel.background = element_blank())
b2 = base + theme(
  axis.title.x = element_blank(),
  axis.title.y = element_blank()
)
b3 = base + theme(axis.line = element_line(colour = "grey50"))

grid.arrange(b1,b2,b3,ncol = 3)
```



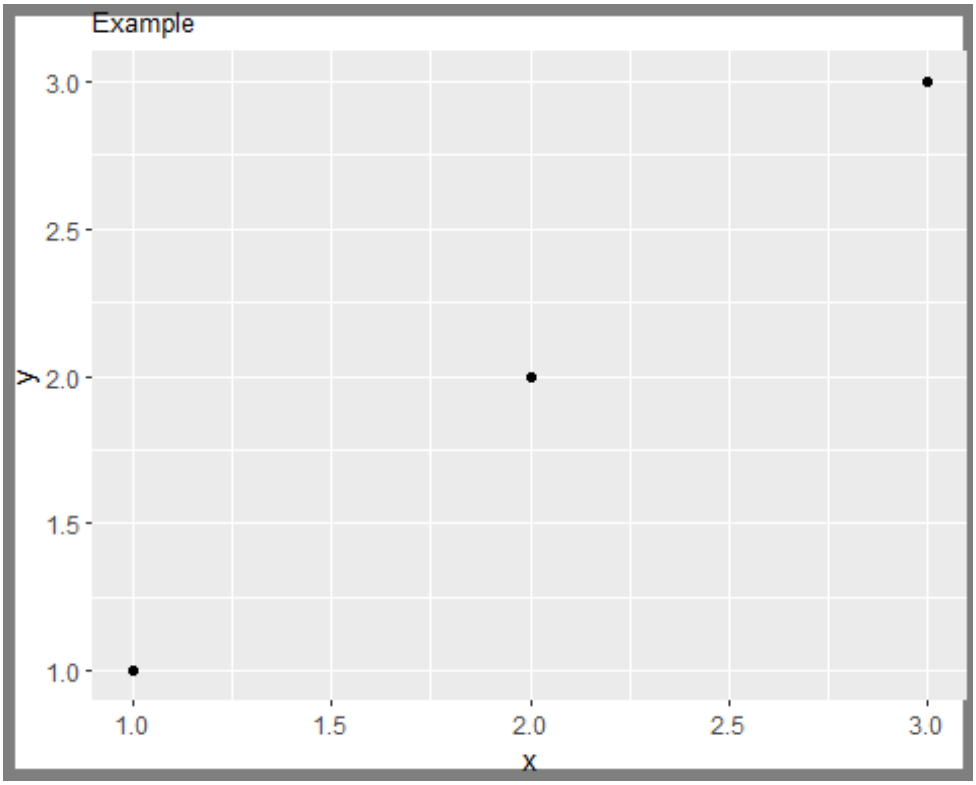
8.4 Theme Elements

There are so many elements. Roughly grouped into five category : plot, axis, legend, panel, facet

8.4.1 Plot Elements

Element	Setter	Description
plot.background	element_rect()	Plot backgroud
plot.title	element_text()	Plot title
plot.margin	margin()	Margins around plot

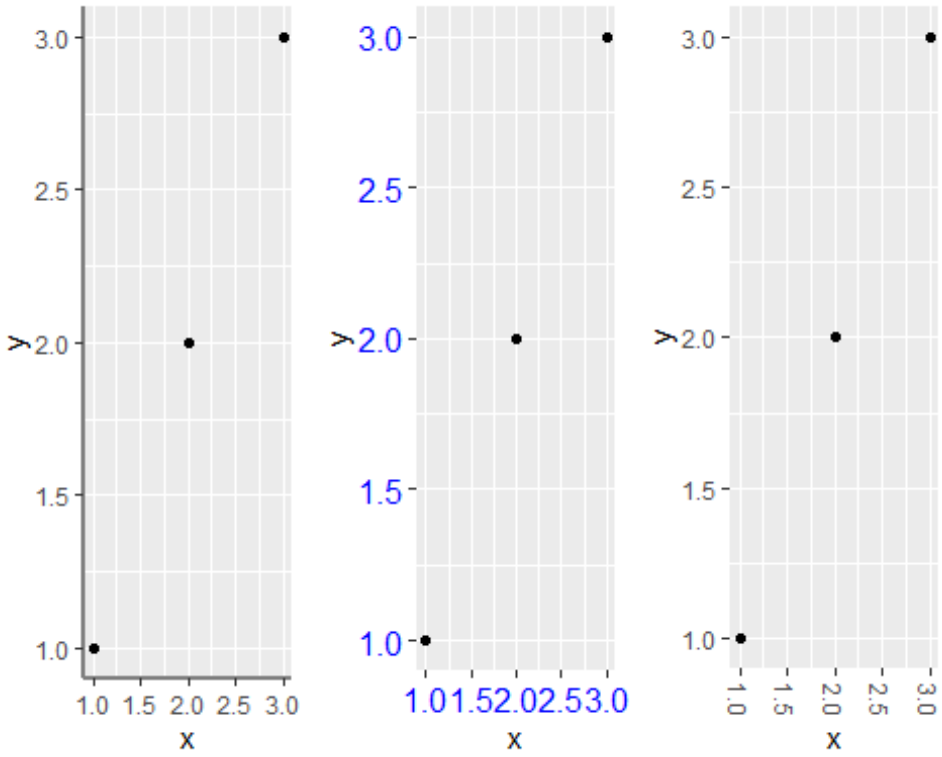
```
base + labs(title = "Example") +theme(
  plot.background = element_rect(colour = "grey50", size = 4),
  plot.title = element_text(size = 10),
  plot.margin = margin(2, 2, 2, 2)
)
```



8.4.2 Axis Elements

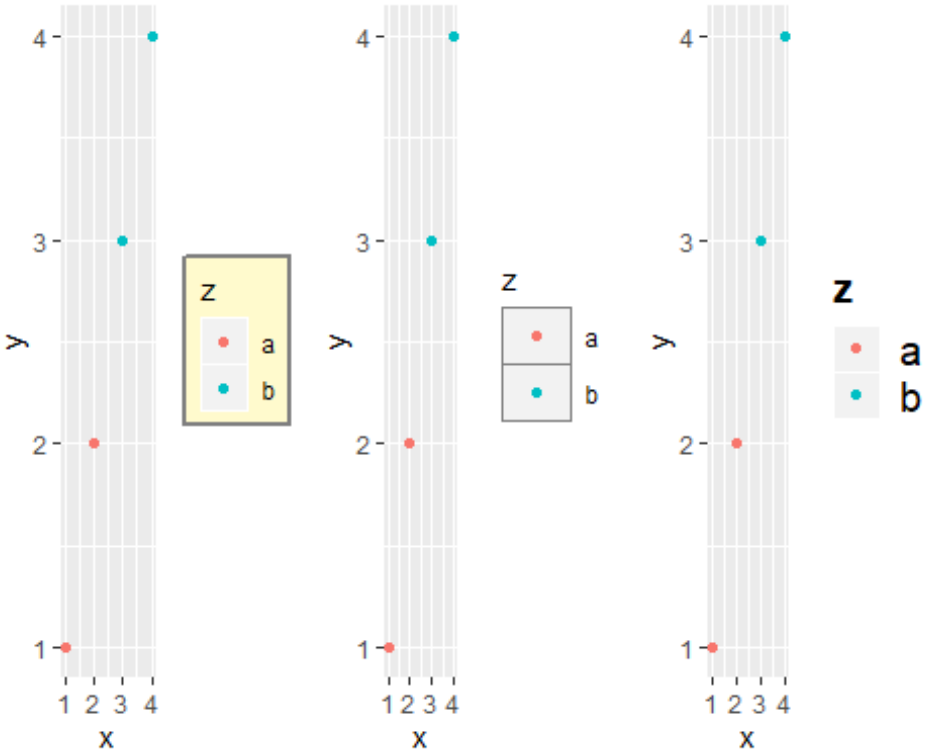
axis.text (and axis.title) comes in three forms: axis.text, axis.text.x, and axis.text.y.

```
df <- data.frame(x = 1:3, y = 1:3)
base <- ggplot(df, aes(x, y)) + geom_point()
# Accentuate the axes
b1 = base + theme(axis.line = element_line(colour = "grey50", size = 1))
# Style both x and y axis labels
b2 = base + theme(axis.text = element_text(color = "blue", size = 12))
# Useful for long labels
b3 = base + theme(axis.text.x = element_text(angle = -90, vjust = 0.5))

grid.arrange(b1,b2,b3,ncol=3)
```



8.4.3 Legend Elements

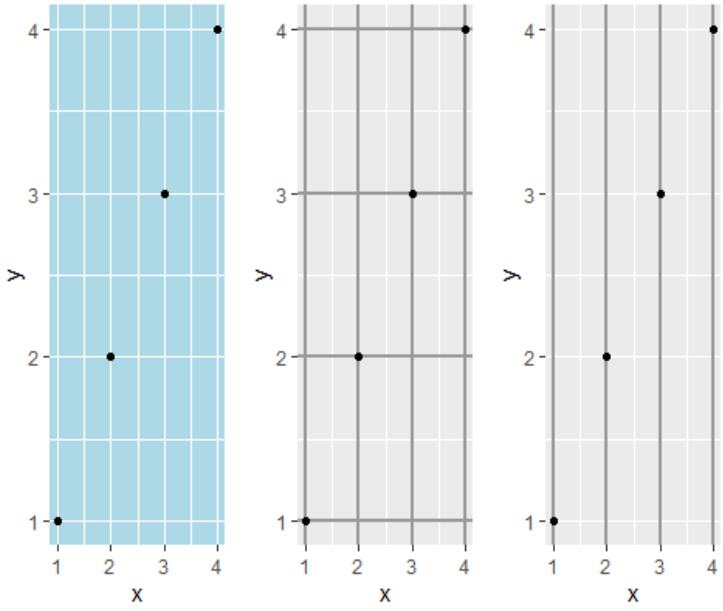
```
df <- data.frame(x = 1:4, y = 1:4, z = rep(c("a", "b"), each = 2))

base <- ggplot(df, aes(x, y, colour = z)) + geom_point()

b1 = base + theme(
  legend.background = element_rect(
    fill = "lemonchiffon",
    colour = "grey50",
    size = 1
)
)
b2 = base + theme(
  legend.key = element_rect(color = "grey50"),
  legend.key.width = unit(0.9, "cm"),
  legend.key.height = unit(0.75, "cm")
)
b3 = base + theme(
  legend.text = element_text(size = 15),
  legend.title = element_text(size = 15, face = "bold")
)
grid.arrange(b1,b2,b3,ncol=3)
```



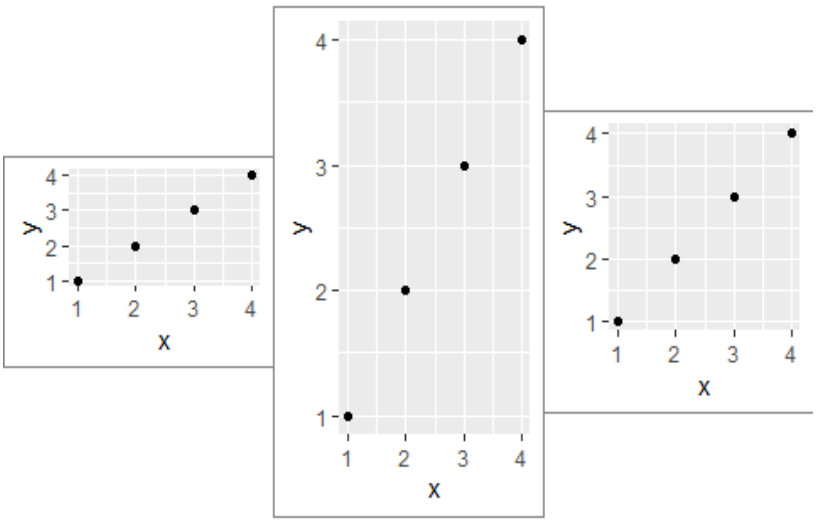
8.4.4 Panel Elements

The main difference between *panel.background* and *panel.border* is that the background is drawn underneath the data, and the border is drawn on top of it. For that reason, you'll always need to assign *fill = NA* when overriding *panel.border*.

```
base <- ggplot(df, aes(x, y)) + geom_point()
# Modify background
b1 = base + theme(panel.background = element_rect(fill = "lightblue"))
# Tweak major grid lines
b2 = base + theme(panel.grid.major = element_line(color = "gray60", size = 0.
8))
# Just in one direction
b3 = base + theme(panel.grid.major.x = element_line(color = "gray60", size =
0.8))

grid.arrange(b1,b2,b3,ncol=3)
```



```r
base2 <- base + theme(plot.background = element_rect(colour = "grey50"))
# Wide screen
b1 = base2 + theme(aspect.ratio = 9 / 16)
# Long and skiny
b2 = base2 + theme(aspect.ratio = 2 / 1)
# Square
b3 = base2 + theme(aspect.ratio = 1)

grid.arrange(b1,b2,b3,ncol=3)
```



8.4.5 Facetting Elements

Element *strip.text.x* affects both *facet_wrap()* or *facet grid(); strip.text.y* only affects *facet_grid().*

```
df <- data.frame(x = 1:4, y = 1:4, z = c("a", "a", "b", "b"))
base_f <- ggplot(df, aes(x, y)) + geom_point() + facet_wrap(~z)
b1 = base_f
b2 = base_f + theme(panel.margin = unit(0.5, "in"))

## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead

b3 = base_f + theme(
  strip.background = element_rect(fill = "grey20", color = "grey80", size = 1
),
  strip.text = element_text(colour = "white")
)

grid.arrange(b1,b2,b3,ncol=3)
```