Post 2: Themes in ggplot2

Themes in ggplot

```
#load packages
library(ggplot2)
```

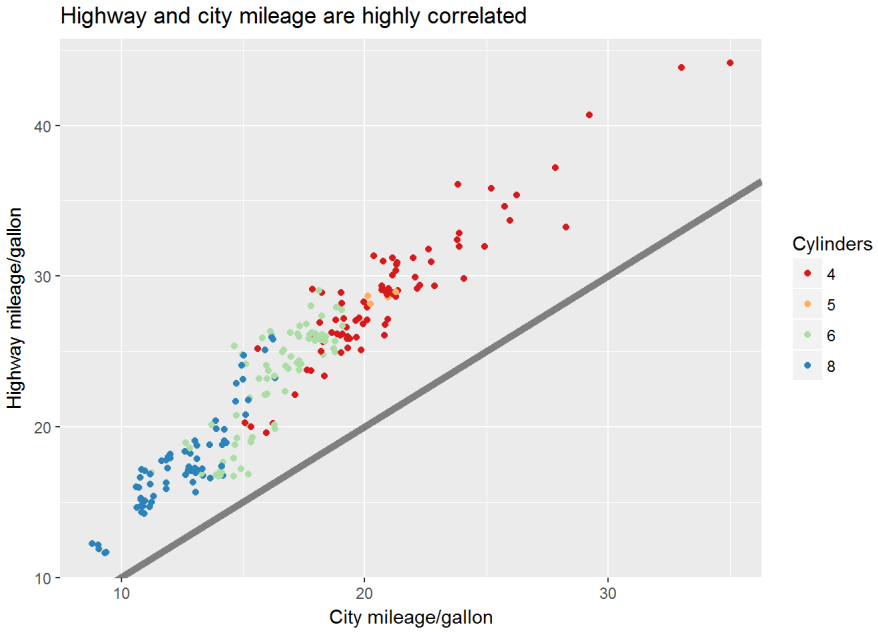
```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

Knowing the basics of the R package “ggplot2” can give you a really nice graph for your data analysis, however, getting a professional graph will require an extra bit of work. Data representation is essential to any analysis work. Even if you have great data, graph may hinder some reader’s understanding because it is not represented correctly. The theme system of the ggplot package is great to help you customize your graph to fit your needs. It lets you precisely control the non-data elements of your graph and makes your plot more aesthetically pleasing to look at.

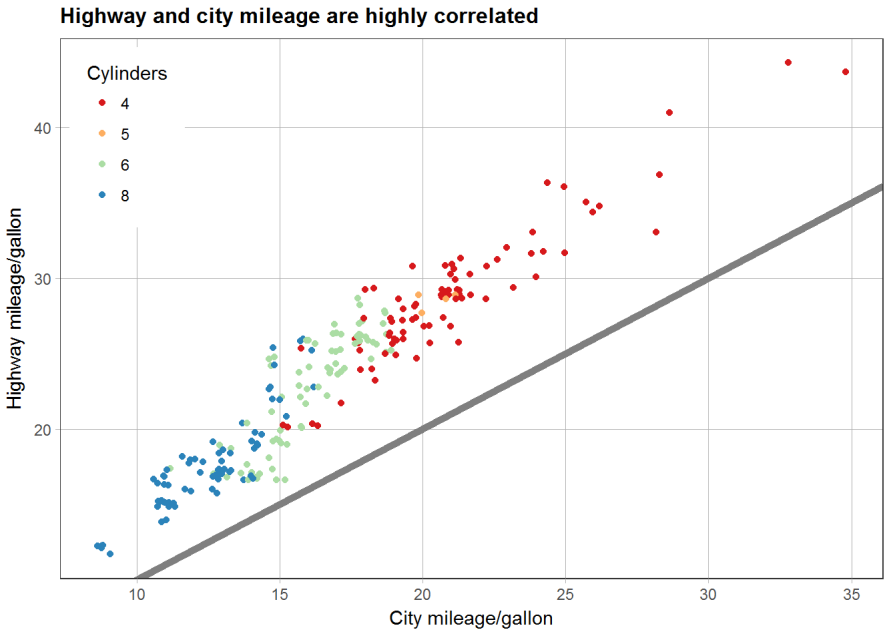
The theme system makes the ggplot2 package different from other graphical packages, such as base and lattice graphics, because it allows for a separation of control over the data and non-data parts of a graph. You can only change the theme after you’ve already decided how your data is to be displayed. Below is an example comparing graphs with and without using the theme system:

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

labelled <- base +
labs(
x = "City mileage/gallon",
y = "Highway mileage/gallon",
colour = "Cylinders",
title = "Highway and city mileage are highly correlated"
) +
scale_colour_brewer(type = "seq", palette = "Spectral")
labelled
```




```
styled <- labelled +
theme_bw() +
theme(
plot.title = element_text(face = "bold", size = 12),
legend.background = element_rect(fill = "white", size = 4, colour = "white"),
legend.justification = c(0, 1),
legend.position = c(.02, .97),
axis.ticks = element_line(colour = "grey70", size = 0.2),
panel.grid.major = element_line(colour = "grey70", size = 0.2),
panel.grid.minor = element_blank()
)
styled
```



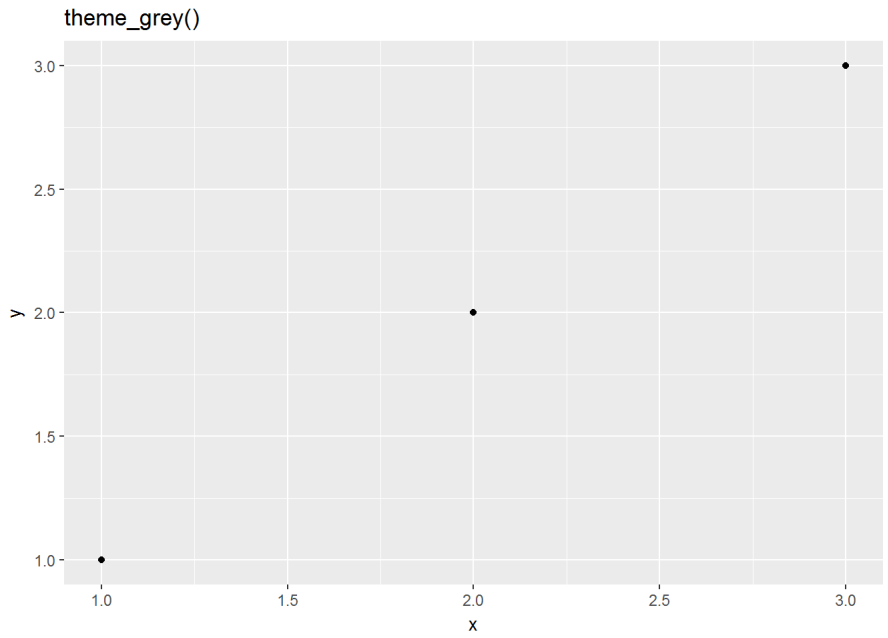
Complete Themes

The previous code for theme system may seem to be too much, so there are complete themes that you can use with one short code. The most used theme is perhaps `theme_grey()` which produces the signature ggplot2 theme with a light grey background and white grid lines. The theme is designed to to present the graph with a smooth flow from text to image: the grey background gives the plot a similar typographic color to the text, ensuring that the graphics fit in with the rest of the document without a sudden bright white background to the reader's eyes.

Here is a view on most of the complete themes: * `theme_bw()` : similar to `theme_grey()` that uses a white background and thin grey grid lines.

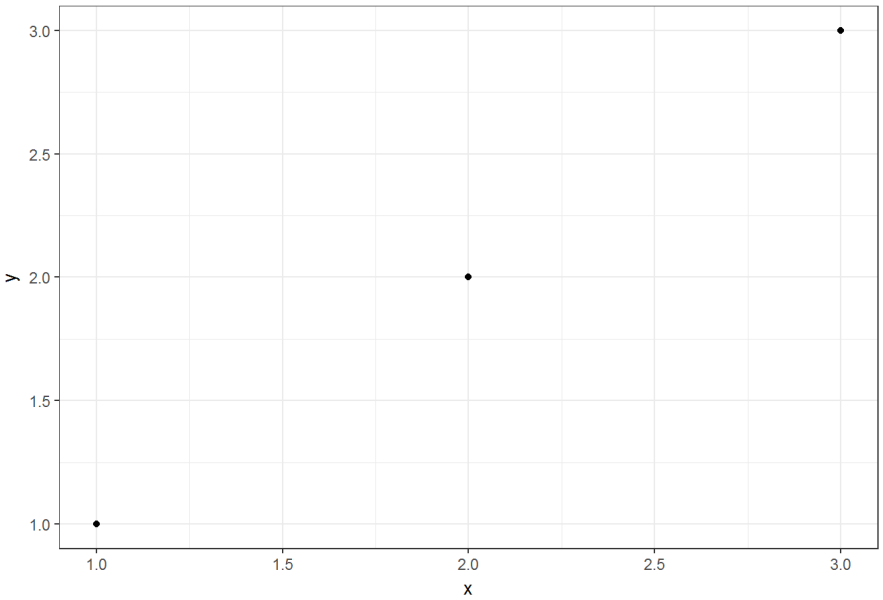
- `theme_linedraw()` : A theme with only black lines on white backgrounds, reminiscent of a line drawing.
- `theme_light()` : similar to `theme_linedraw()` but with light grey lines and axes, to direct more attention towards the data.
- `theme_dark()` : a darker version of theme_light()`, with similar line sizes but a dark background. Very useful to make thin colored lines pop out.
- `theme_minimal()` : A minimalist theme with no background annotations.
- `theme_classic()` : A classic-looking theme, with x and y axis lines and no grid lines.
- `theme_void()` : A completely empty theme.

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



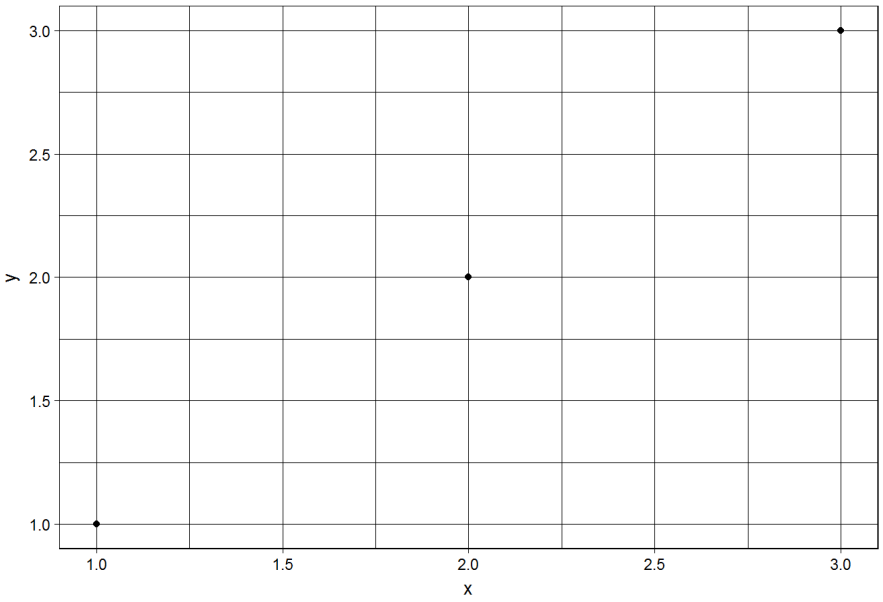

```
base + theme_bw() + ggtitle("theme_bw()")
```

theme_bw()



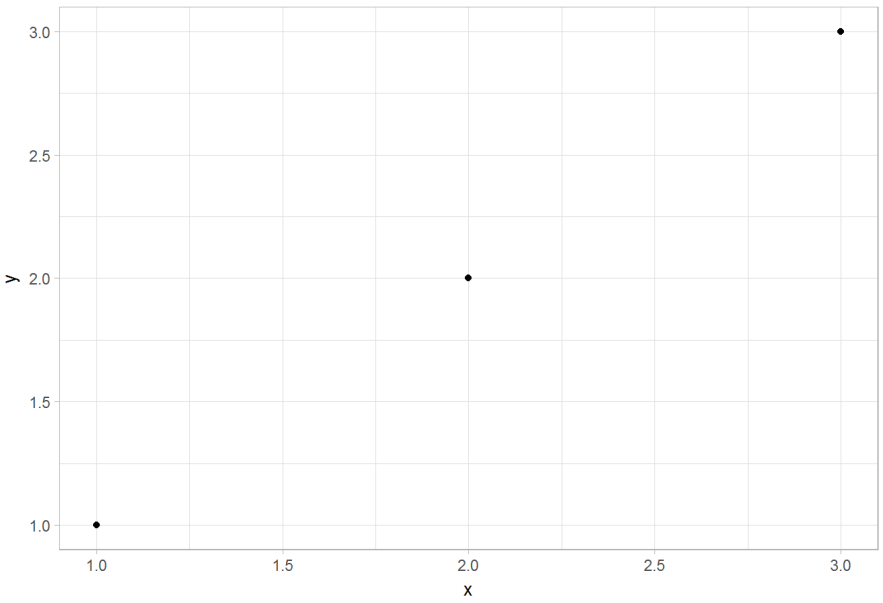

```
base + theme_linedraw() + ggtitle("theme_linedraw()")
```

theme_linedraw()

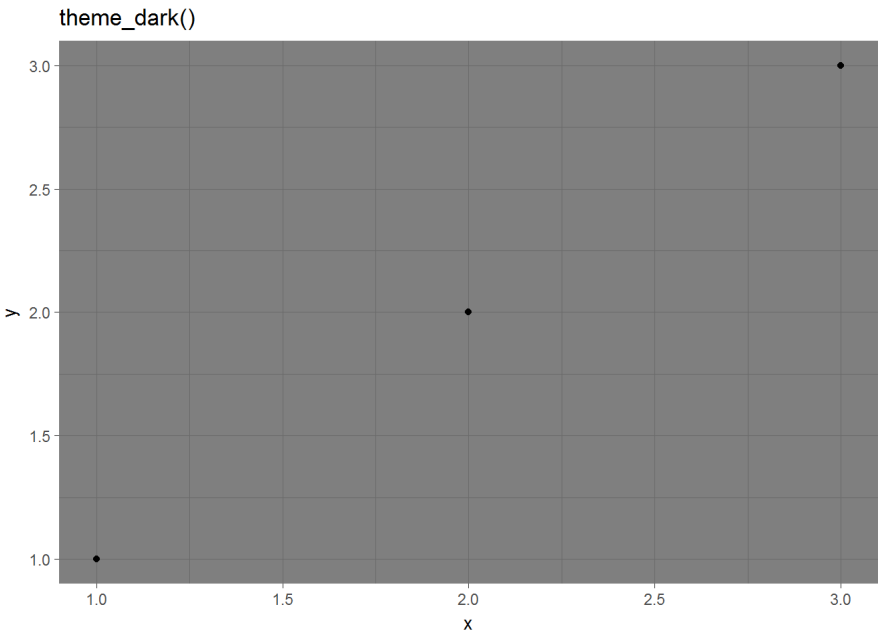



```
base + theme_light() + ggtitle("theme_light()")
```

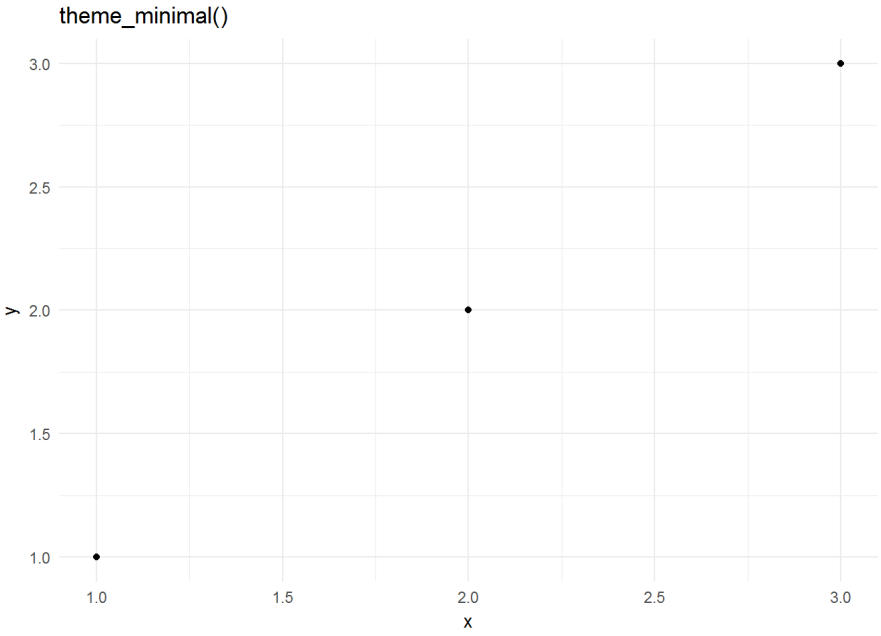
theme_light()



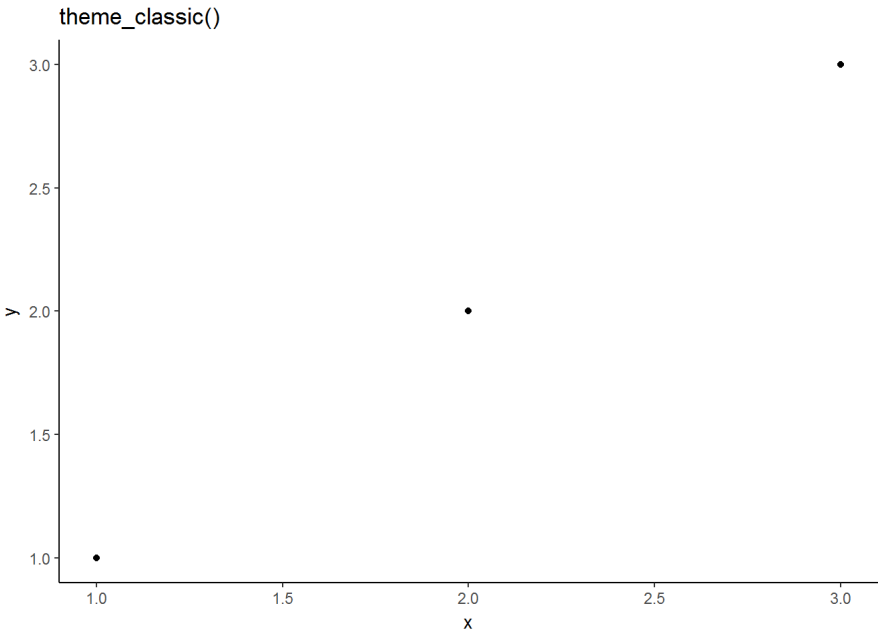
```
base + theme_dark() + ggtitle("theme_dark()")
```



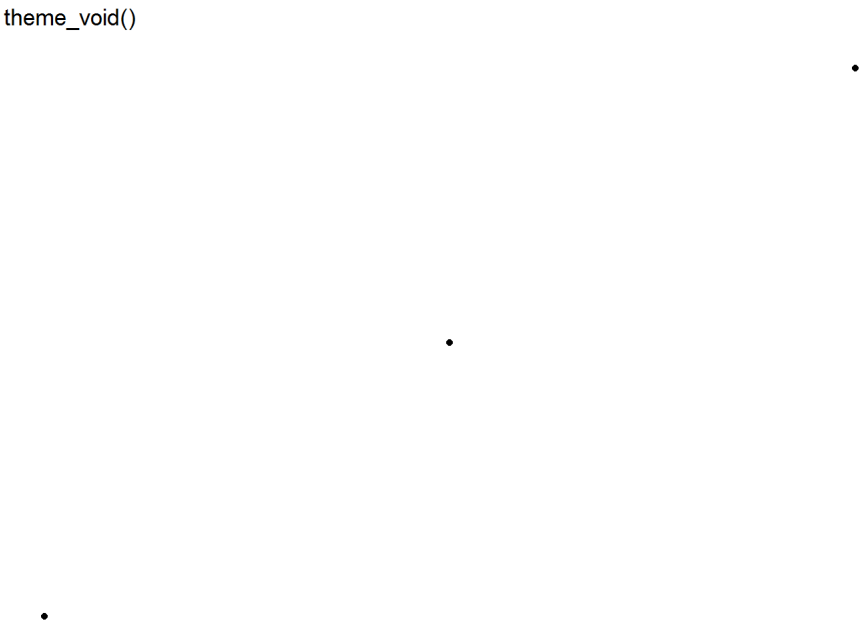

```
base + theme_minimal() + ggtitle("theme_minimal()")
```




```
base + theme_classic() + ggtitle("theme_classic()")
```




```
base + theme_void() + ggtitle("theme_void()")
```



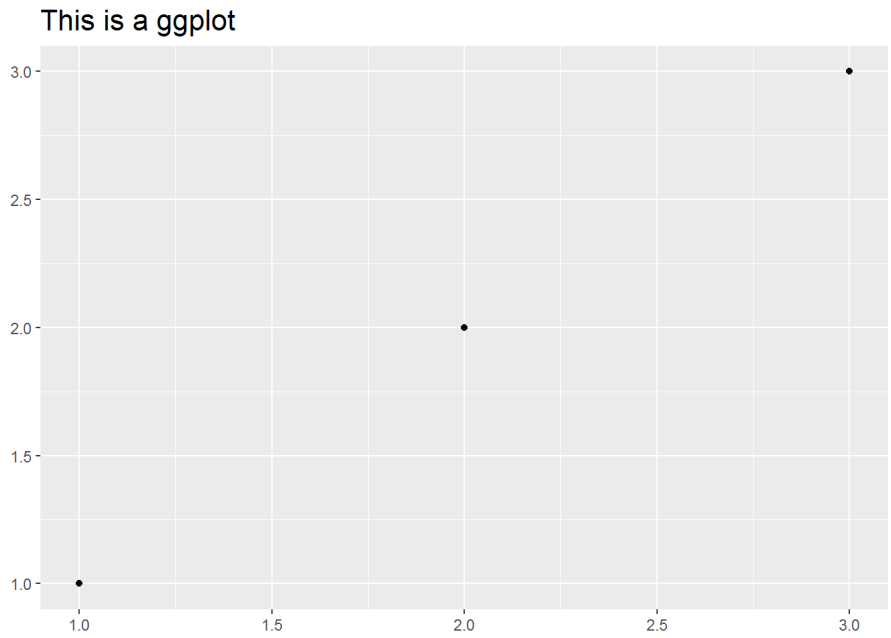
The complete themes is great for initial explorations but does not give you much control over over individual elements so you need to use `theme()` to override the default settings.

Modifying Theme Components

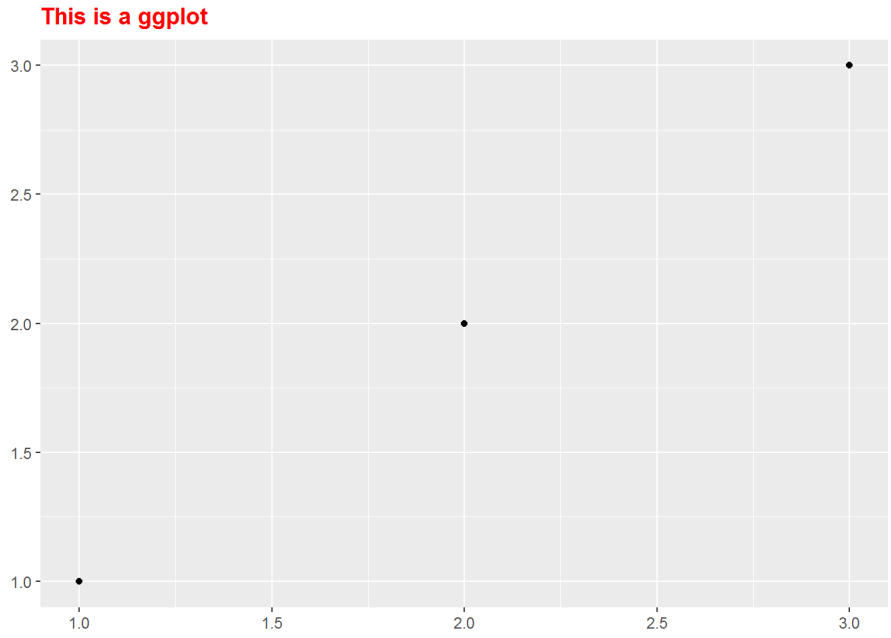
With the `theme()` function, you can modify four basic types of built-in element functions: text, lines, rectangles, and blank. You can change the parameters for each of the element functions to customize your graph. This is done with a code like `plot + theme(element.name = element function())`. We will only customize a few theme elements in the examples below, but there are 40 elements that you can modify. They can be roughly grouped into five categories: plot, axis, legend, panel and facet. You can find them all in the help document.

`element_text()` draws labels and headings. You can modify the font family, face, color, size (in points), hjust, vjust, angle (in degrees) and line height (as ratio of font case):

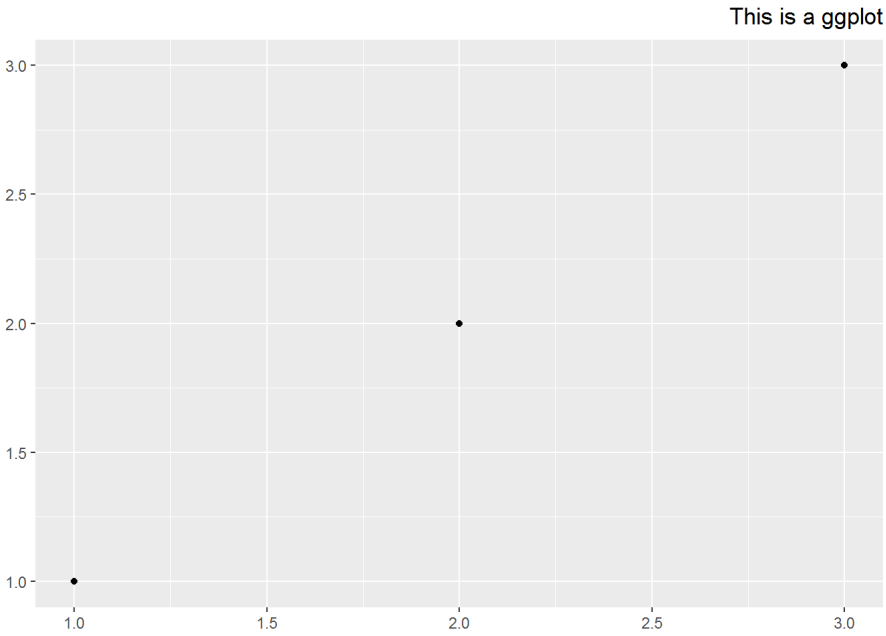
```
base_t <- base + labs(title = "This is a ggplot") + xlab(NULL) + ylab(NULL)
base_t + theme(plot.title = element_text(size = 16))   #increasing the size of the plot title
```




```
base_t + theme(plot.title = element_text(face = "bold", colour = "red"))   #bolding and changing the plot title to red
```

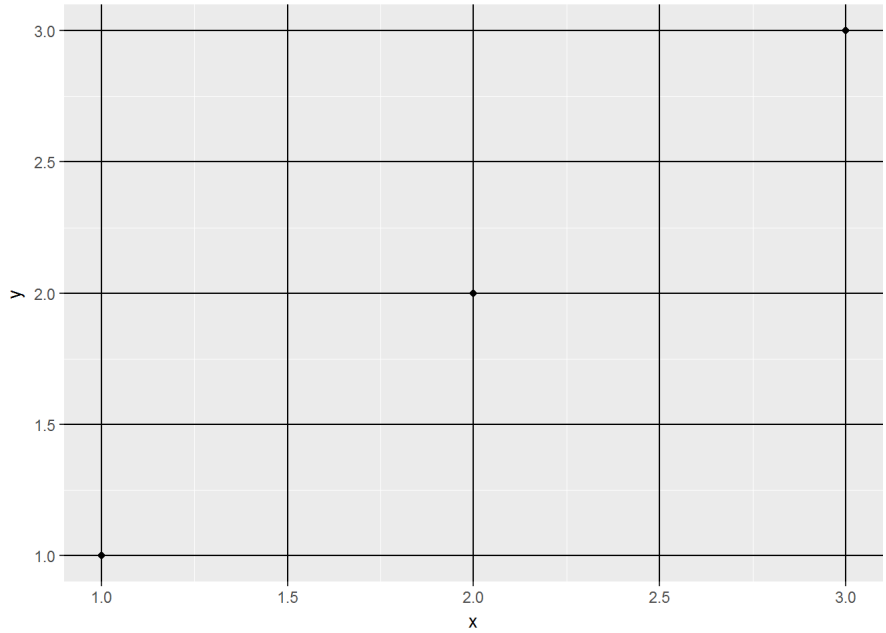



```
base_t + theme(plot.title = element_text(hjust = 1))   #aligning plot title to the right
```

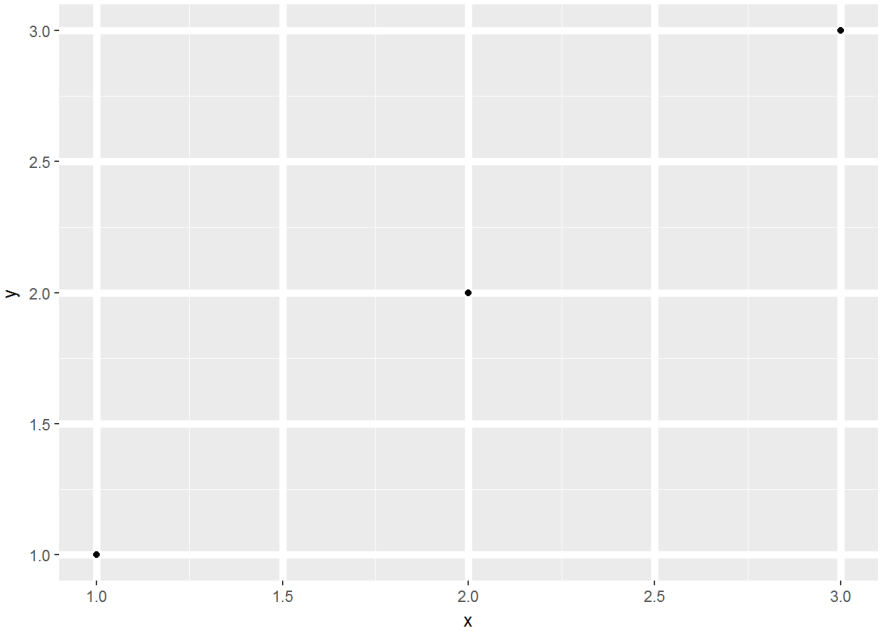


`element_line()` draws lines parameterized by color, size and line type:

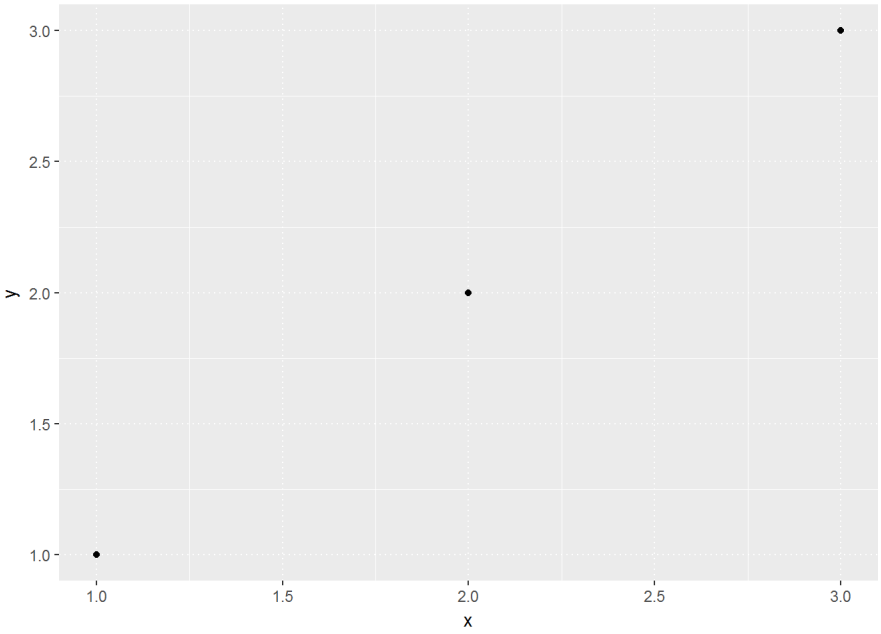
```
base + theme(panel.grid.major = element_line(colour = "black"))   #black grid lines
```




```
base + theme(panel.grid.major = element_line(size = 2))   #thick grid lines
```

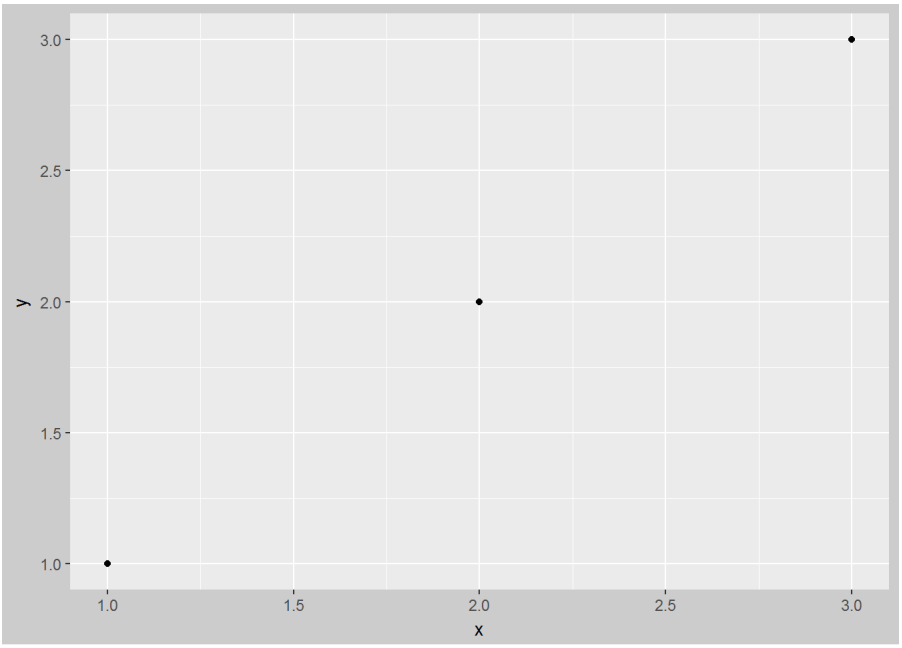



```
base + theme(panel.grid.major = element_line(linetype = "dotted"))   #dotted grid lines
```

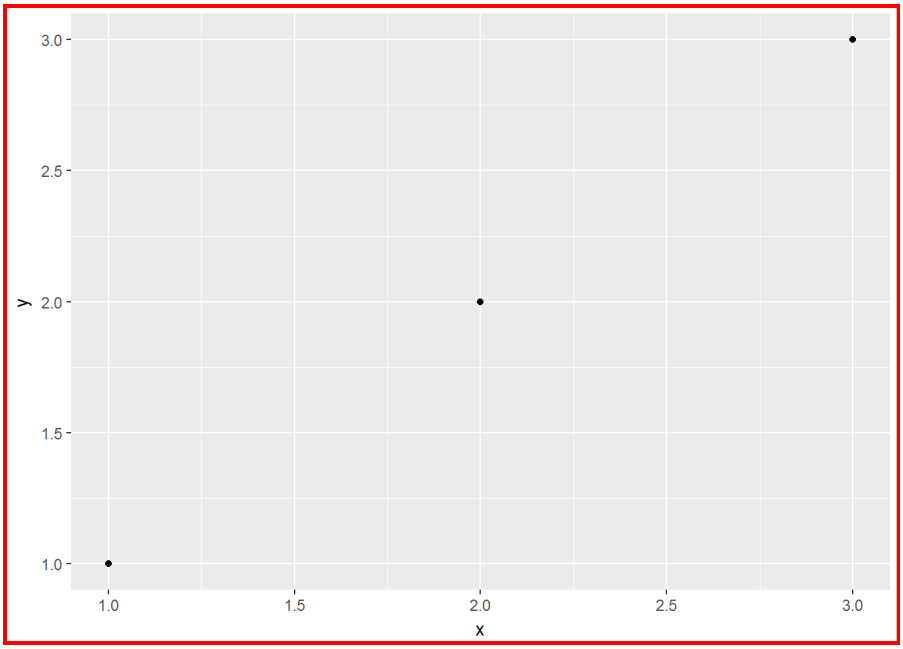


`element_rect()` draws rectangles, mostly used for backgrounds, defined by fill color and border color, size and line type.

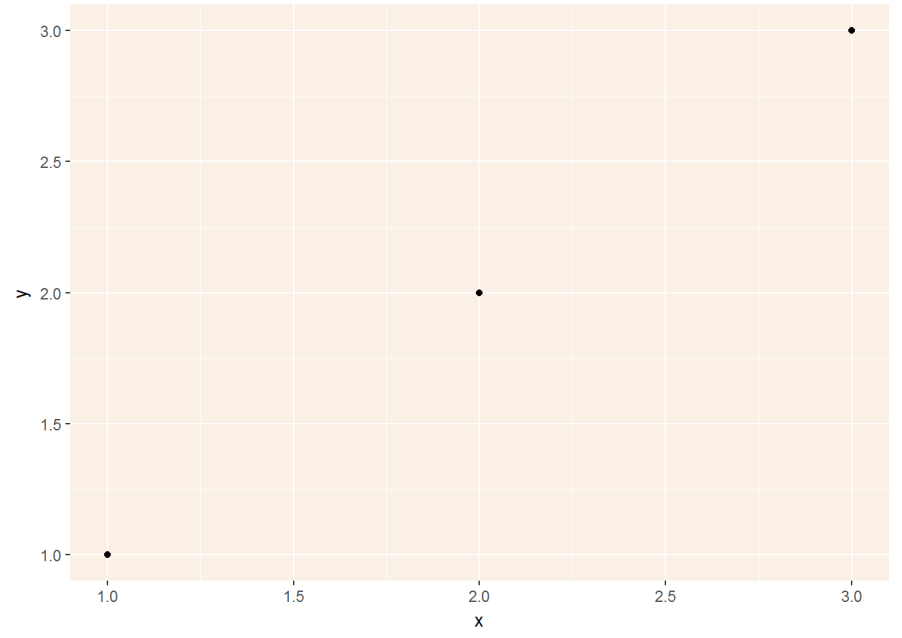
```
base + theme(plot.background = element_rect(fill = "grey80", colour = NA))   #grey rectangle fill
```




```
base + theme(plot.background = element_rect(colour = "red", size = 2))   #red rectangle line
```

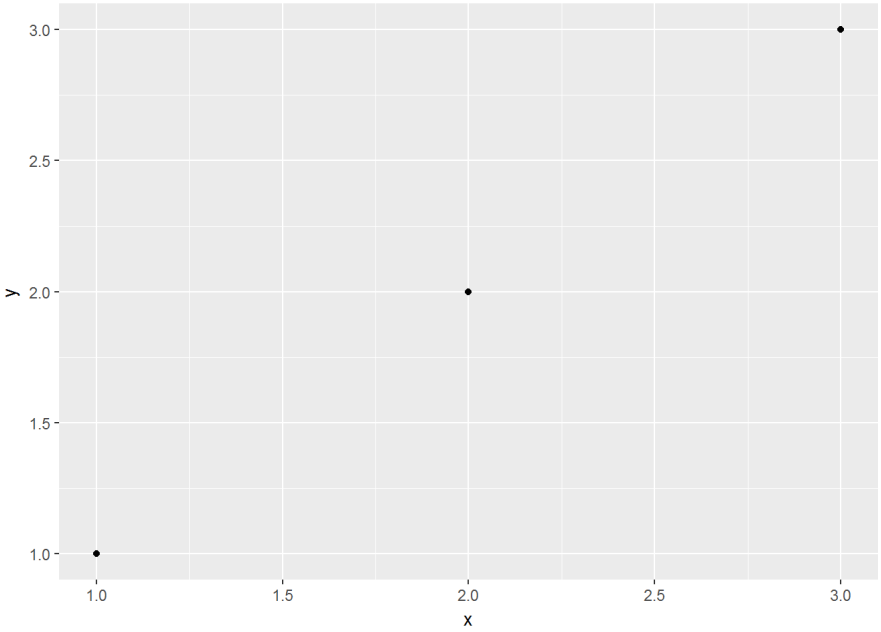



```
base + theme(panel.background = element_rect(fill = "linen"))   #changing panel background color
```

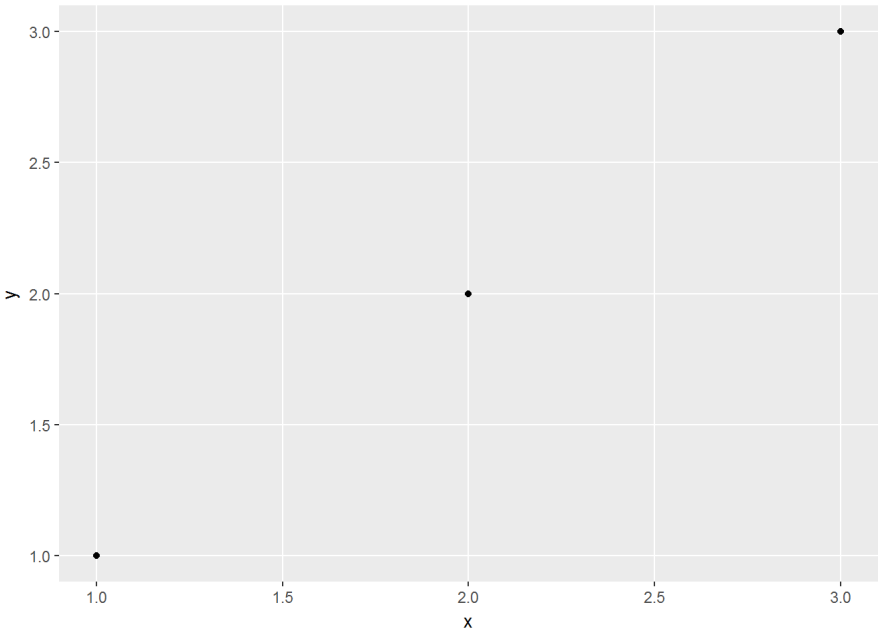


`element_blank()` draws nothing. Only use this if you want to hide certain elements.

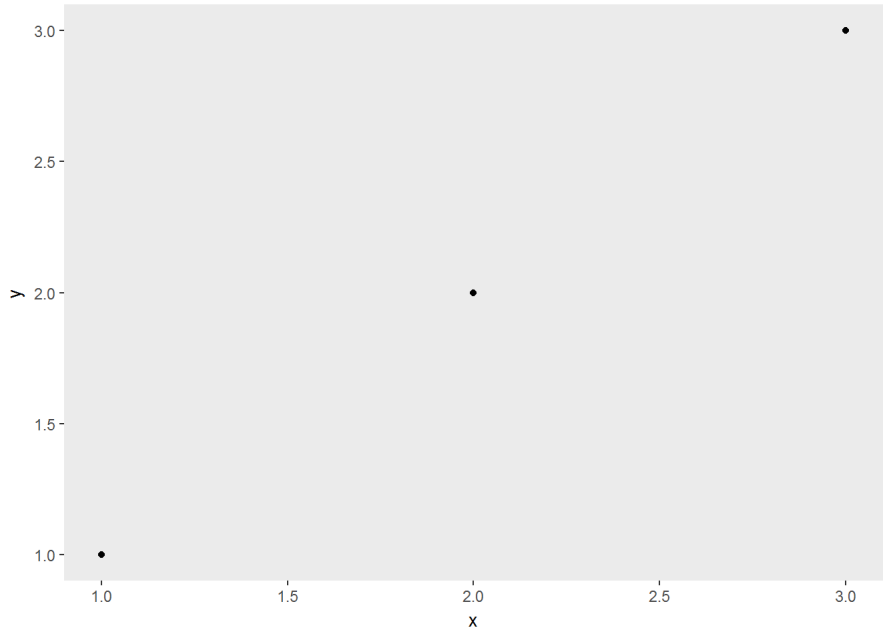
```
base
```



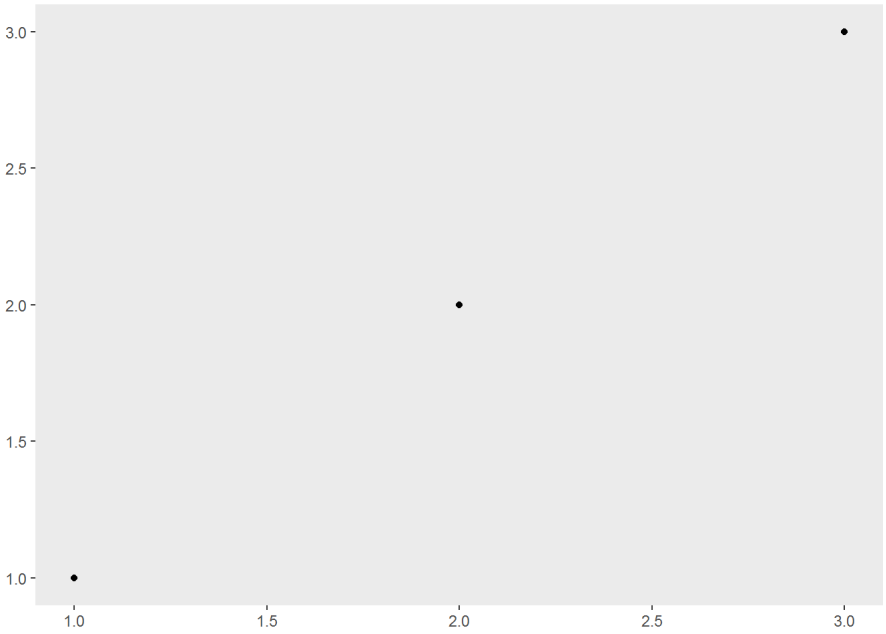

```
last_plot() + theme(panel.grid.minor = element_blank())   #removes some grid lines
```




```
last_plot() + theme(panel.grid.major = element_blank())   #removes all grid lines
```



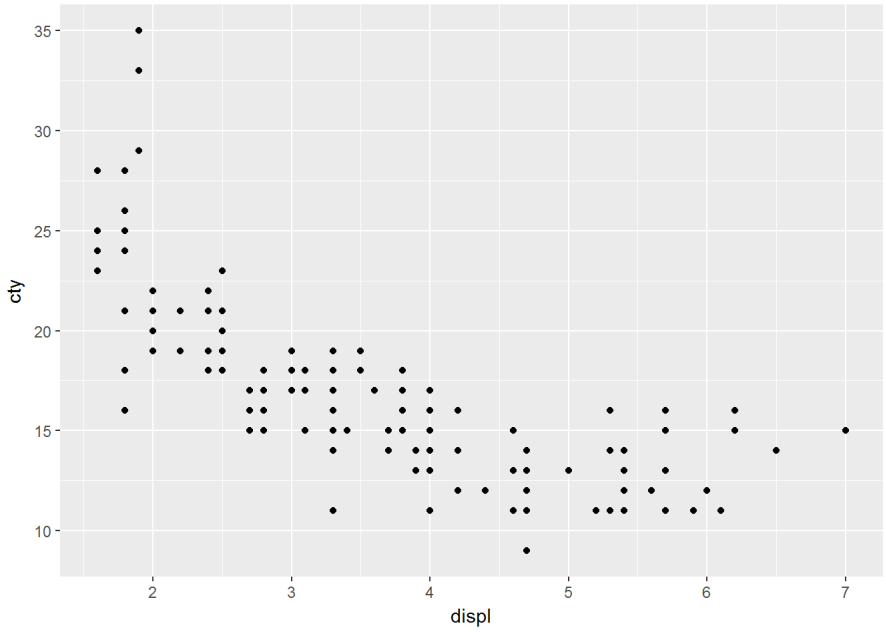

```
last_plot() + theme(
axis.title.x = element_blank(),
axis.title.y = element_blank()
)   #hides x and y axis labels
```



Saving Your Output

ggplot2 package also provides a convenient way of saving your graph with a shorthand `ggsave()`:

```
ggplot(mpg, aes(displ, cty)) + geom_point()
```




```
ggsave("output.pdf")
```

```
## Saving 7 x 5 in image
```

`ggsave()` is used directly after you've drawn your graph and it takes the following arguments:

- The first argument, path, specifies the path where the image should be saved. The file extension will be used to automatically select the correct graphics device. ggsave() can produce .eps, .pdf, .svg, .wmf, .png, .jpg, .bmp, and .tiff.
- Width and height control the output size, specified in inches. If left blank, they'll use the size of the on-screen graphics device.
- If you're saving your graph as a picture (i.e. .png, .jpg), the dpi argument controls the resolution of the plot. The default is 300 but you can increase the number for a higher resolution output, or change it to 96 for on-screen display.

Conclusion

ggplot2 is a powerful package to help creating graphics, however, you can't rely on the basic functions for a graph. The graphic features should be specific to your data and the display should highlight the parts you'd like to show. The theme system is an easy way to make the graph like your own and you should take time to explore the different options you have with the theme element commands to find one that best fits your need.

Referencese:

DataCamp, director. Learn R: An Introduction to ggplot2. Youtube, 9 Nov. 2016. http://www.youtube.com/watch?v=YxKr2a-Y1WE&list=PLjgj6kdf_snaBCTJEi53DvRVgOuVbzyk

Prabhakaran, Selva. "The Complete ggplot2 Tutorial." r-Statistics.co. http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

Smith, David. "10 Tips for Making Your R Graphics Look Their Best." Revolutions. http://blog.revolutionanalytics.com/2009/01/10-tips-for-making-your-r-graphics-look-their-best.html

Wickham, Hadley, and Carson Sievert. ggplot2: Elegant Graphics for Data Analysis. Springer, 2016.

Wickham , Hadley. "Create Elegant Data Visualisations Using the Grammar of Graphics . ggplot2." Create Elegant Data Visualisations Using the Grammar of Graphics . ggplot2, Tidyverse. <ggplot2.tidyverse.org> http://ggplot2.tidyverse.org/

Wickham, Hadley. "R Graph Catalog." R Graph Catalog. http://shiny.stat.ubc.ca/r-graph-catalog/

Woolf, Max. "How to Make High Quality Data Visualizations for Websites With R and ggplot2." Minimaxir | Max Woolf's Blog. http://minimaxir.com/2017/08/ggplot2-web/