Part 2: Customizing the look and feel

In this tutorial, I discuss how to customize the looks of the 6 most important aesthetics of a plot. Put together, it provides a fairly comprehensive list of how to accomplish your plot customization tasks in detail.

1. Adding Plot and Axis Titles
2. Modifying Legend
- How to Change Legend Title
- How to Change Legend Labels and Point Color
- How to Change Order of Legend
- How to Style the Legend Title, Text and Key
- How to Change Legend Positions
3. Adding Text, Label and Annotation
- How to Adding Text and Label around the Points
- How to Adding Custom Annotation Anywhere inside Plot
4. Flipping and Reversing X and Y Axis
5. Faceting: Draw multiple plots within one figure
- Facet Wrap
- Facet Grid
6. Modifying Plot Background, Major and Minor Axis
- How to Change Plot Background
- How to Removing Major and Minor Grid, Border, Axis Title, Text and Ticks
- How to Add an Image in Background
- Inheritance Structure of Theme Components

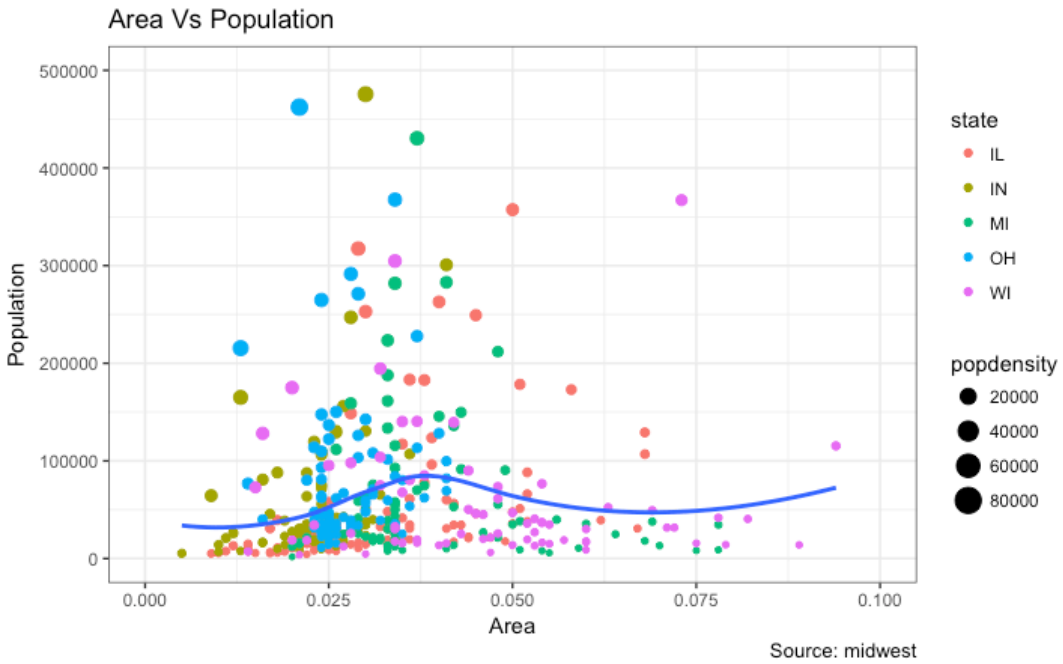
Let's begin with a scatterplot of Population against Area from `midwest` dataset. The point's color and size vary based on `state` (categorical) and `popdensity` (continuous) columns respectively. We have done something similar in the previous ggplot2 tutorial already.

The below plot has the essential components such as the title, axis labels and legend setup nicely. But how to modify the looks?

Most of the requirements related to look and feel can be achieved using the `theme()` function. It accepts a large number of arguments. Type `?theme` in the R console and see for yourself.

```
# Setup
options(scipen=999)
library(ggplot2)
data("midwest", package = "ggplot2")
theme_set(theme_bw())
# midwest <- read.csv("http://goo.gl/G1K41K")  # bkup data source


# Add plot components -------------------------------
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
```

```
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


# Call plot ----------------------------------------
plot(gg)
```



The arguments passed to `theme()` components require to be set using special `element_type()` functions. They are of 4 major types.

1. `element_text()`: Since the title, subtitle and captions are textual items, `element_text()` function is used to set it.
2. `element_line()`: Likewise `element_line()` is use to modify line based components such as the axis lines, major and minor grid lines, etc.
3. `element_rect()`: Modifies rectangle components such as plot and panel background.
4. `element_blank()`: Turns off displaying the theme item.

More on this follows in upcoming discussion.

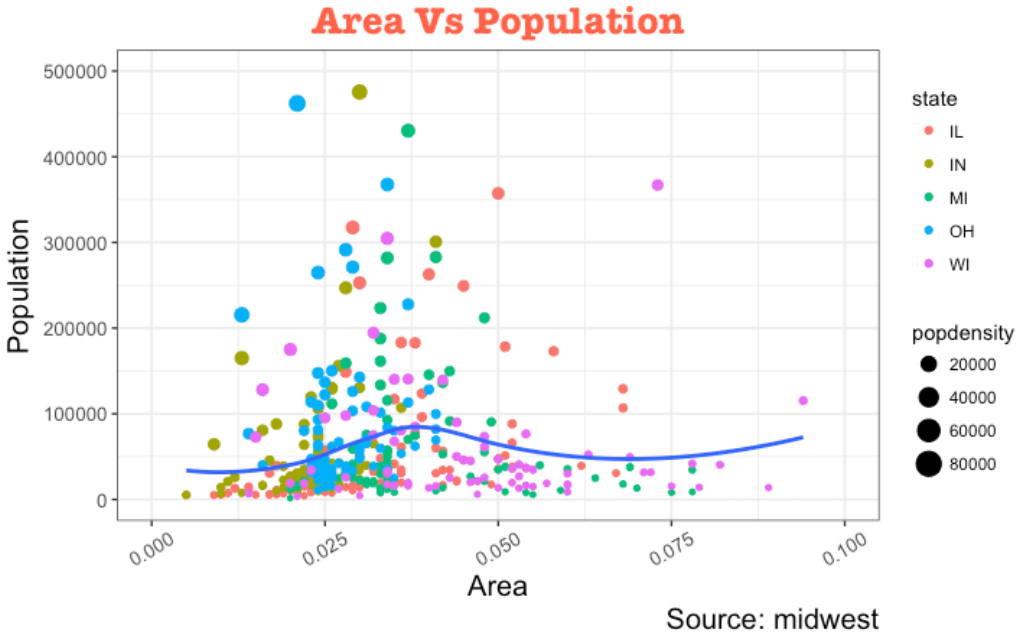
Let's discuss a number of tasks related to changing the plot output, starting with modifying the title and axis texts.

1. Adding Plot and Axis Titles

Plot and axis titles and the axis text are part of the plot's theme. Therefore, it can be modified using the `theme()` function. The `theme()` function accepts one of the four `element_type()` functions mentioned above as arguments. Since the plot and axis titles are textual components, `element_text()` is used to modify them.

Below, I have changed the size, color, face and line-height. The axis text can be rotated by changing the `angle`.

```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


# Modify theme components -------------------------------------------
gg + theme(plot.title=element_text(size=20,
                                   face="bold",
                                   family="American Typewriter",
                                   color="tomato",
                                   hjust=0.5,
                                   lineheight=1.2),  # title
           plot.subtitle=element_text(size=15,
                                      family="American Typewriter",
                                      face="bold",
                                      hjust=0.5),  # subtitle
           plot.caption=element_text(size=15),  # caption
           axis.title.x=element_text(vjust=10,
                                     size=15),  # X axis title
           axis.title.y=element_text(size=15),  # Y axis title
           axis.text.x=element_text(size=10,
                                    angle = 30,
                                    vjust=.5),  # X axis text
           axis.text.y=element_text(size=10))  # Y axis text
```



- vjust, controls the vertical spacing between title (or label) and plot.
- hjust, controls the horizontal spacing. Setting it to 0.5 centers the title.
- family, is used to set a new font
- face, sets the font face ("plain", "italic", "bold", "bold.italic")

Above example covers some of the frequently used theme modifications and the actual list is too long. So ?theme is the first place you want to look at if you want to change the look and feel of any component.

2. Modifying Legend

Whenever your plot's geom (like points, lines, bars, etc) is set to change the aesthetics (fill, size, col, shape or stroke) based on another column, as in geom_point(aes(col=state, size=popdensity)), a legend is automatically drawn.

If you are creating a geom where the aesthetics are static, a legend is *not* drawn by default. In such cases you might want to create your own legend manually. The below examples are for cases where you have the legend created automatically.

How to Change the Legend Title

Let's now change the legend title. We have two legends, one each for color and size. The size is based on a continuous variable while the color is based on a categorical(discrete) variable.

There are 3 ways to change the legend title.

Method 1: Using labs()

```
library(ggplot2)


# Base Plot
```

```
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")

gg + labs(color="State", size="Density")  # modify legend title
```

Method 2: Using guides()

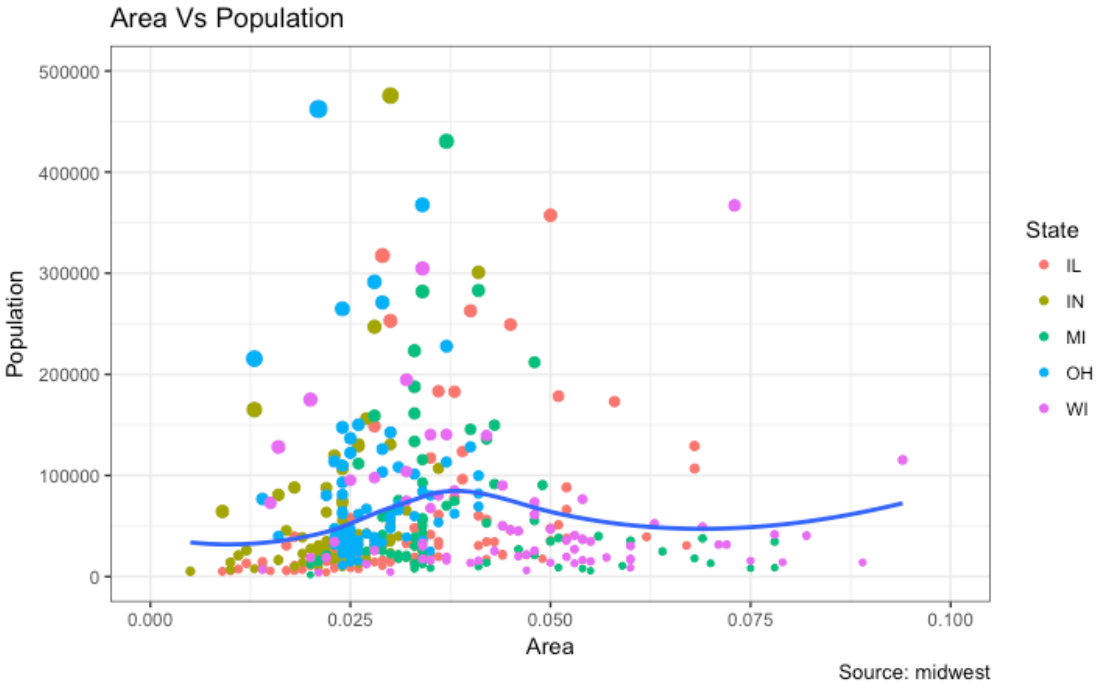
```
library(ggplot2)

# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")

gg <- gg + guides(color=guide_legend("State"), size=guide_legend("Density"))  # modify legend title
plot(gg)
```

Method 3: Using scale_aesthetic_vartype() format

The format of scale_aestheic_vartype() allows you to turn off legend for one particular aesthetic, leaving the rest in place. This can be done just by setting guide=FALSE. For example, if the legend is for size of points based on a continuous variable, then scale_size_continuous() would be the right function to use.

Can you guess what function to use if you have a legend for shape and is based on a categorical variable?

```
library(ggplot2)

# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")

# Modify Legend
gg + scale_color_discrete(name="State") + scale_size_continuous(name = "Density", guide = FALSE)  # turn off legend for size
```

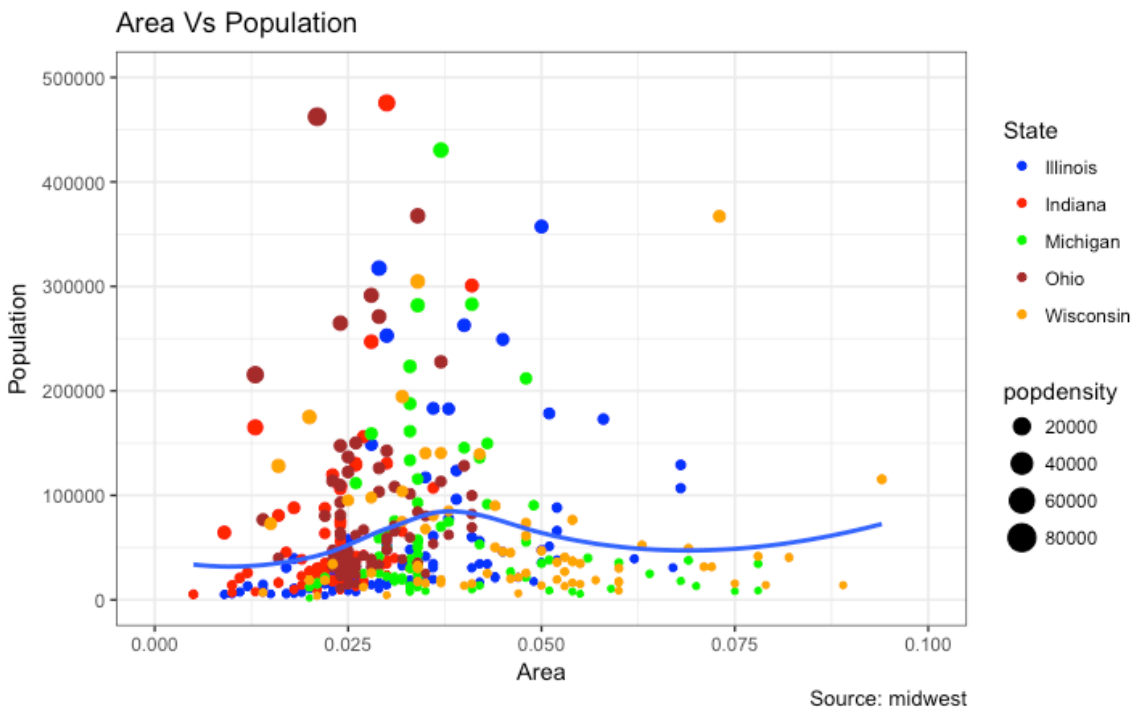


How to Change Legend Labels and Point Colors for Categories

This can be done using the respective `scale_aesthetic_manual()` function. The new legend labels are supplied as a character vector to the `labels` argument. If you want to change the color of the categories, it can be assigned to the `values` argument as shown in below example.

```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


gg + scale_color_manual(name="State",
                        labels = c("Illinois",
                                   "Indiana",
                                   "Michigan",
                                   "Ohio",
                                   "Wisconsin"),
                        values = c("IL"="blue",
                                   "IN"="red",
                                   "MI"="green",
                                   "OH"="brown",
```

```
                    "WI"="orange"))
```

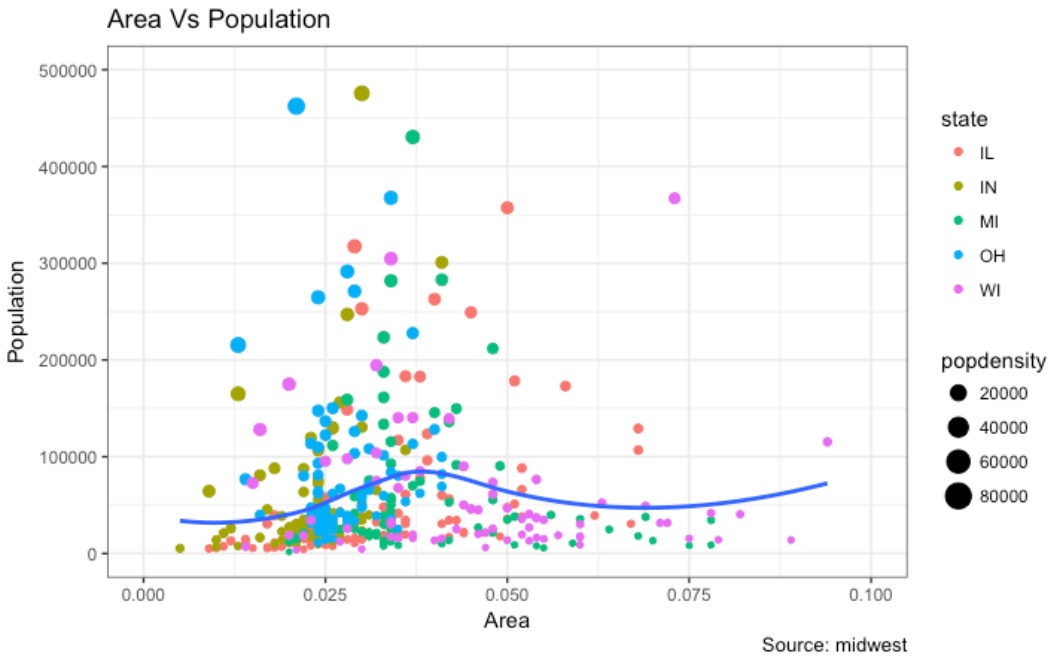


Change the Order of Legend

In case you want to show the legend for color (State) before size (Density), it can be done with the guides() function. The order of the legend has to be set as desired.

If you want to change the position of the labels inside the legend, set it in the required order as seen in previous example.

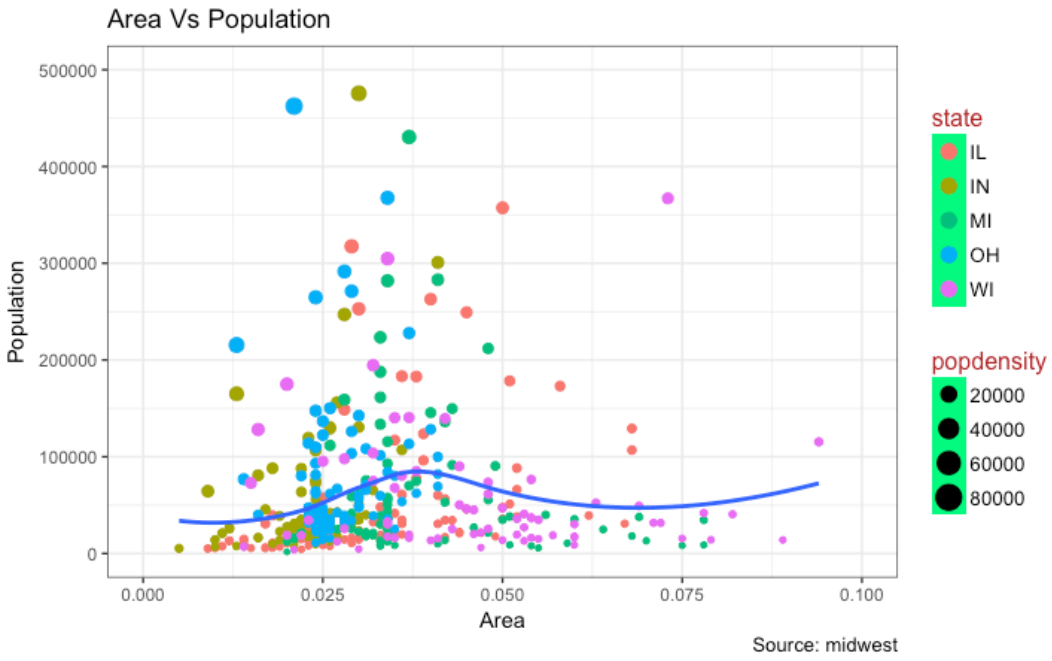
```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


gg + guides(colour = guide_legend(order = 1),
         size = guide_legend(order = 2))
```



How to Style the Legend Title, Text and Key

The styling of legend title, text, key and the guide can also be adjusted. The legend's key is a figure like element, so it has to be set using `element_rect()` function.

```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


gg + theme(legend.title = element_text(size=12, color = "firebrick"),
          legend.text = element_text(size=10),
          legend.key=element_rect(fill='springgreen')) +
  guides(colour = guide_legend(override.aes = list(size=2, stroke=1.5)))
```



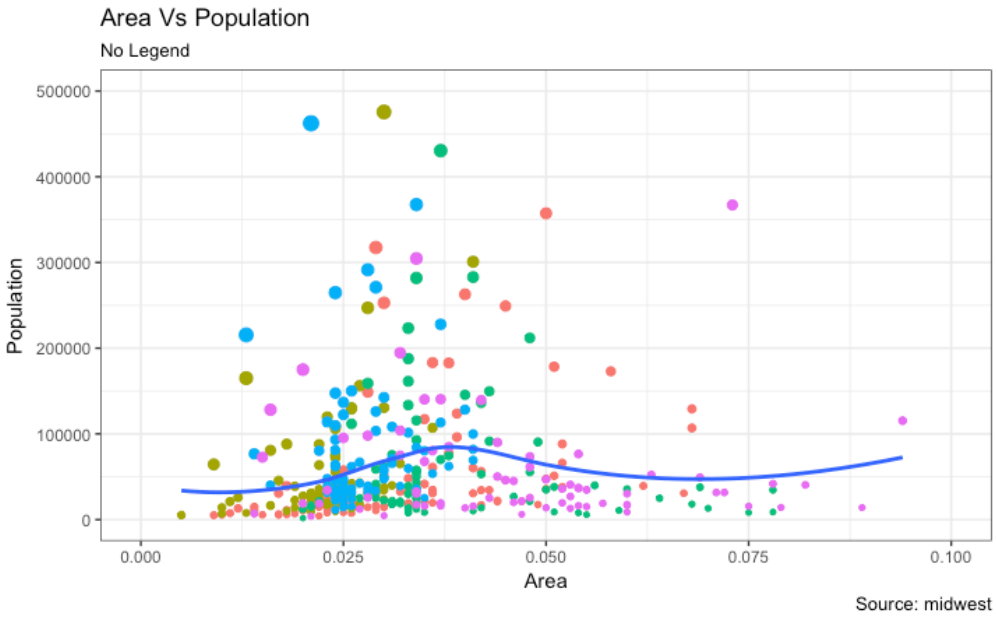
How to Remove the Legend and Change Legend Positions

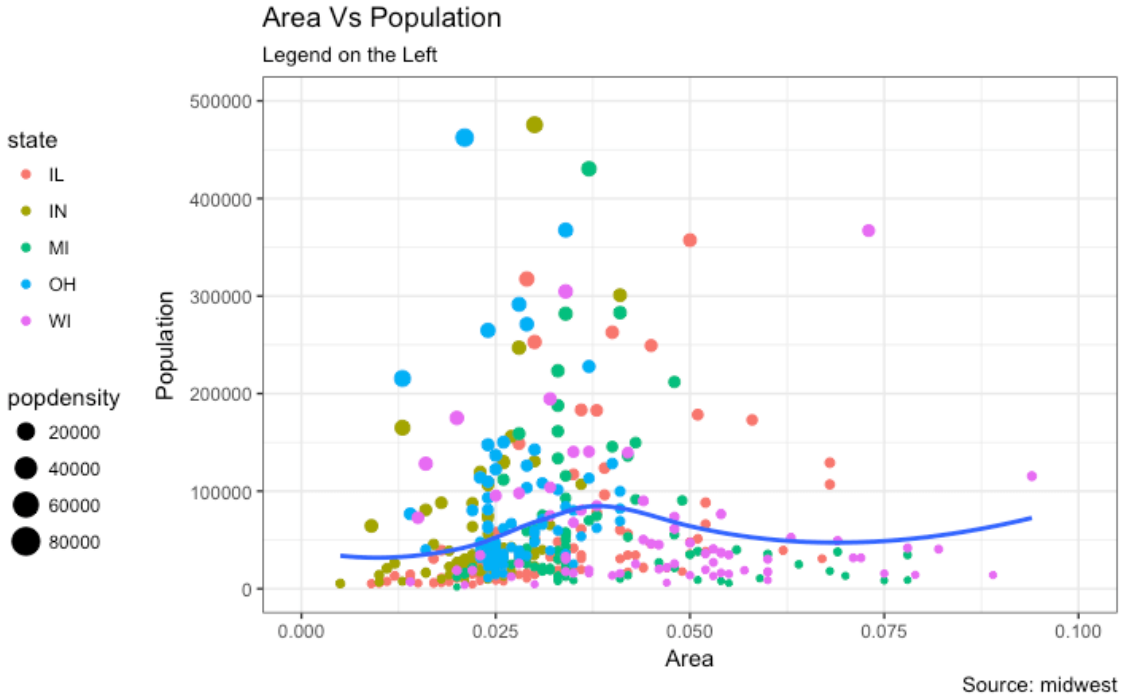
The legend's position inside the plot is an aspect of the theme. So it can be modified using the `theme()`function. If you want to place the legend inside the plot, you can additionally control the hinge point of the legend using `legend.justification`.

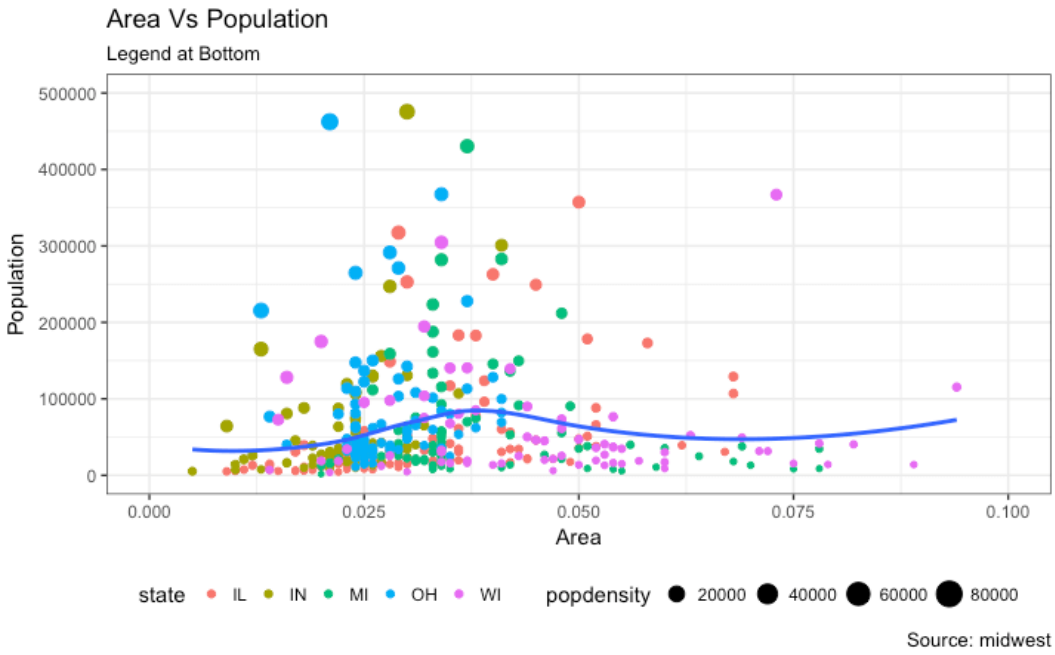
The `legend.position` is the x and y axis position in chart area, where `(0,0)` is bottom left of the chart and `(1,1)` is top right. Likewise, `legend.justification` refers to the hinge point inside the legend.

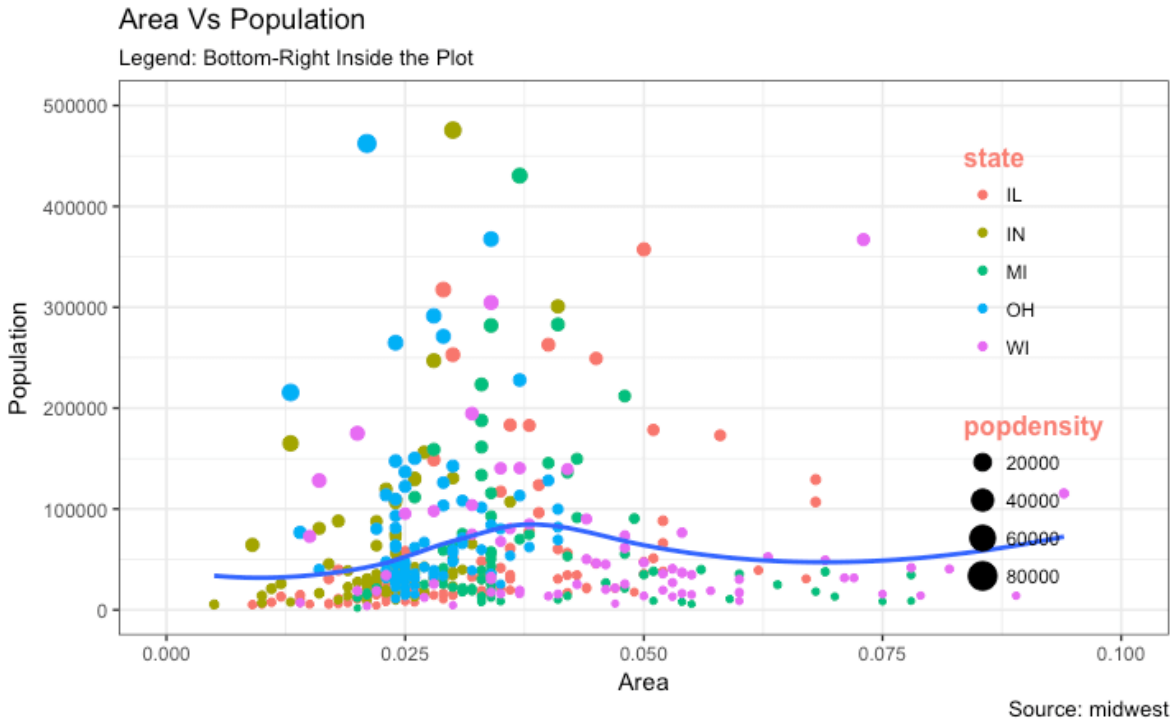
```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


# No legend ----------------------------------------------------
gg + theme(legend.position="None") + labs(subtitle="No Legend")


# Legend to the left -------------------------------------------
gg + theme(legend.position="left") + labs(subtitle="Legend on the Left")


# legend at the bottom and horizontal ----------------------
gg + theme(legend.position="bottom", legend.box = "horizontal") + labs(subtitle="Legend at Bottom")
```

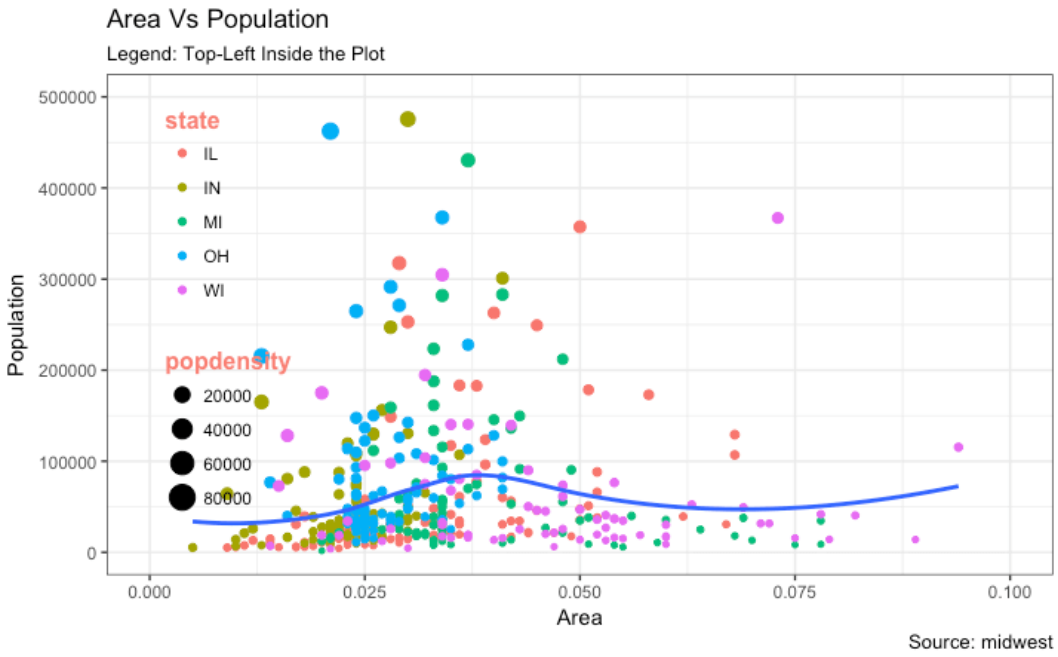
```
# legend at bottom-right, inside the plot ----------------
gg + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
          legend.justification=c(1,0),
          legend.position=c(0.95, 0.05),
          legend.background = element_blank(),
          legend.key = element_blank()) +
  labs(subtitle="Legend: Bottom-Right Inside the Plot")

# legend at top-left, inside the plot ----------------------
gg + theme(legend.title = element_text(size=12, color = "salmon", face="bold"),
          legend.justification=c(0,1),
          legend.position=c(0.05, 0.95),
          legend.background = element_blank(),
          legend.key = element_blank()) +
  labs(subtitle="Legend: Top-Left Inside the Plot")
```


Area Vs Population
No Legend
500000
400000
300000
200000
100000
0
Population
0.000
0.025
0.050
0.075
0.100
Area
Source: midwest


Area Vs Population
Legend on the Left
state
IL
IN
MI
OH
WI
popdensity
20000
40000
60000
80000
500000
400000
300000
200000
100000
0
Population
0.000
0.025
0.050
0.075
0.100
Area
Source: midwest


Area Vs Population
Legend at Bottom
Population
500000
400000
300000
200000
100000
0
0.000
0.025
0.050
0.075
0.100
Area
state IL IN MI OH WI
popdensity 20000 40000 60000 80000
Source: midwest


Area Vs Population
Legend: Bottom-Right Inside the Plot
Population
500000
400000
300000
200000
100000
0
0.000
0.025
0.050
0.075
0.100
Area
state
IL
IN
MI
OH
WI
popdensity
20000
40000
60000
80000
Source: midwest



3. Adding Text, Label and Annotation

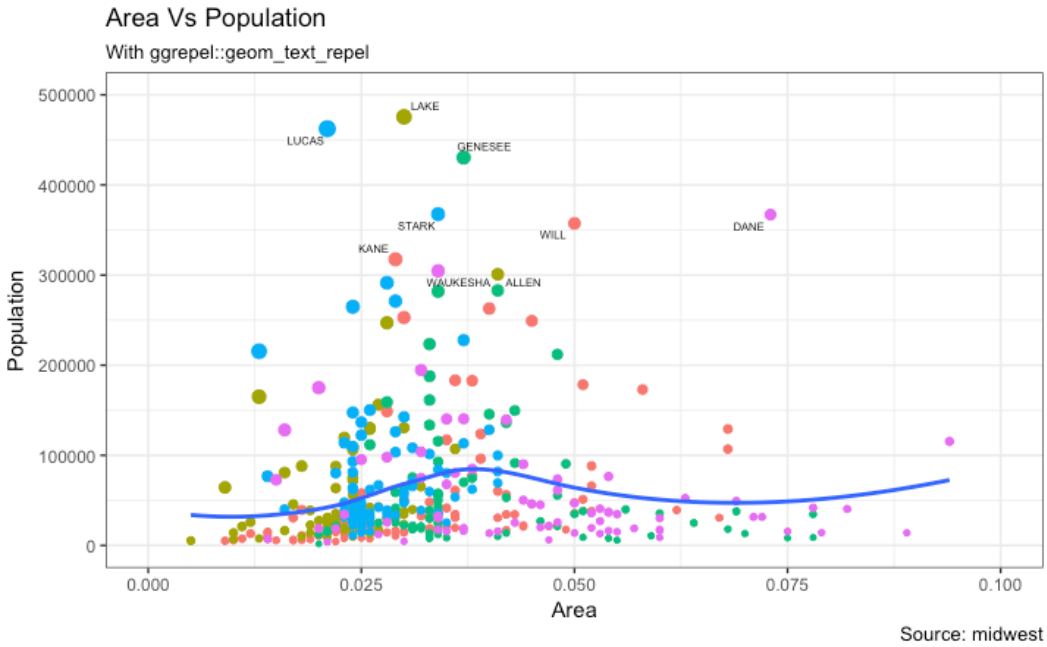
How to Add Text and Label around the Points

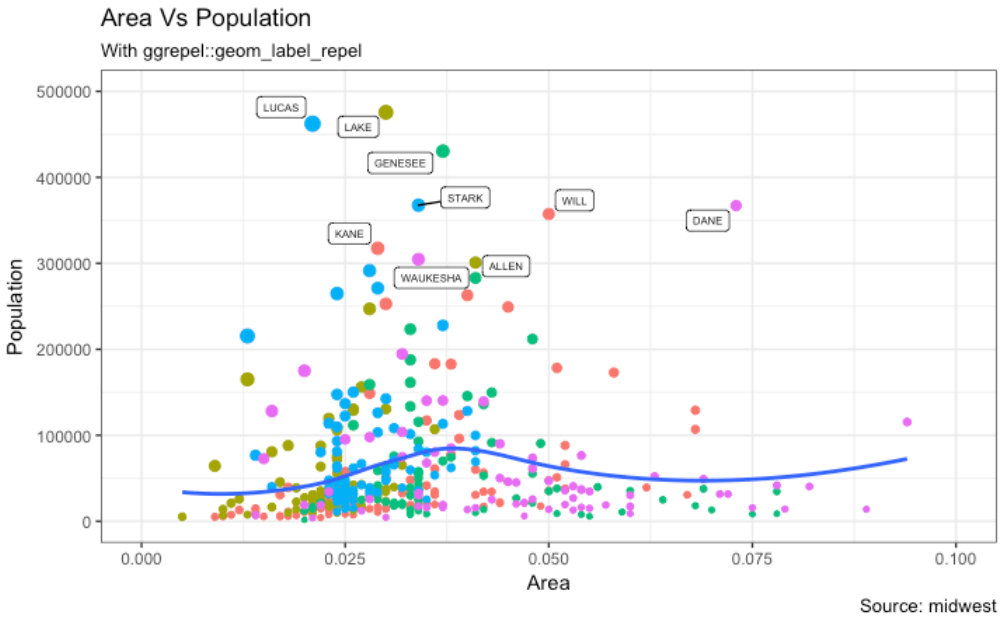
Let's try adding some text. We will add text to only those counties that have population greater than 400K. In order to achieve this, I create another subsetted dataframe (`midwest_sub`) that contains only the counties that qualifies the said condition.

Then, draw the `geom_text` and `geom_label` with this new dataframe as the `data` source. This will ensure that labels (`geom_label`) are added only for the points contained in the new dataframe.

```
library(ggplot2)


# Filter required rows.
midwest_sub <- midwest[midwest$poptotal > 300000, ]
midwest_sub$large_county <- ifelse(midwest_sub$poptotal > 300000, midwest_sub$county, "")


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


# Plot text and label ----------------------------------------------------
gg + geom_text(aes(label=large_county), size=2, data=midwest_sub) + labs(subtitle="With ggplot2::geom_text") + theme(legend.position = "None")  # text
```

```
gg + geom_label(aes(label=large_county), size=2, data=midwest_sub, alpha=0.25) + labs(subtitle="With ggplot2::geom_label") + theme(legend.position = "None")  # label


# Plot text and label that REPELS eachother (using ggrepel pkg) ------------

library(ggrepel)

gg + geom_text_repel(aes(label=large_county), size=2, data=midwest_sub) + labs(subtitle="With ggrepel::geom_text_repel") + theme(legend.position = "None")   # text


gg + geom_label_repel(aes(label=large_county), size=2, data=midwest_sub) + labs(subtitle="With ggrepel::geom_label_repel") + theme(legend.position = "None")   # label
```



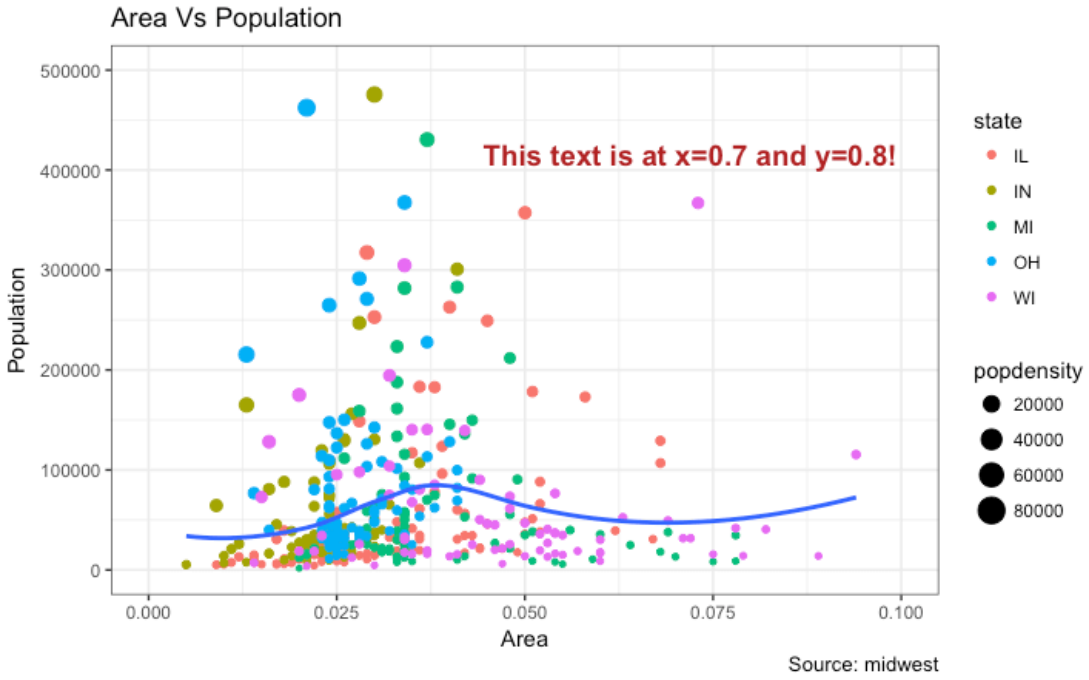


Since the label is looked up from a different dataframe, we need to set the data argument.

How to Add Annotations Anywhere inside Plot

Let's see how to add annotation to any specific point of the chart. It can be done with the `annotation_custom()` function which takes in a `grob` as the argument. So, let's create a grob the holds the text you want to display using the `grid` package.

```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest")


# Define and add annotation ----------------------------------
library(grid)
my_text <- "This text is at x=0.7 and y=0.8!"
my_grob = grid.text(my_text, x=0.7,  y=0.8, gp=gpar(col="firebrick", fontsize=14, fontface="bold"))
gg + annotation_custom(my_grob)
```



4. Flipping and Reversing X and Y Axis

How to flip the X and Y axis?

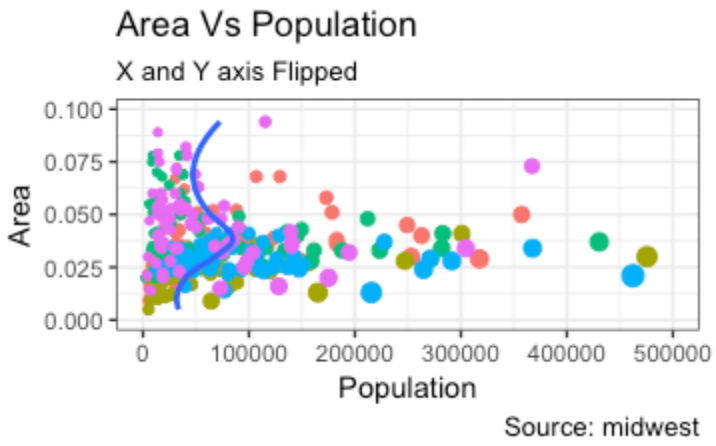
Just add `coord_flip()`.

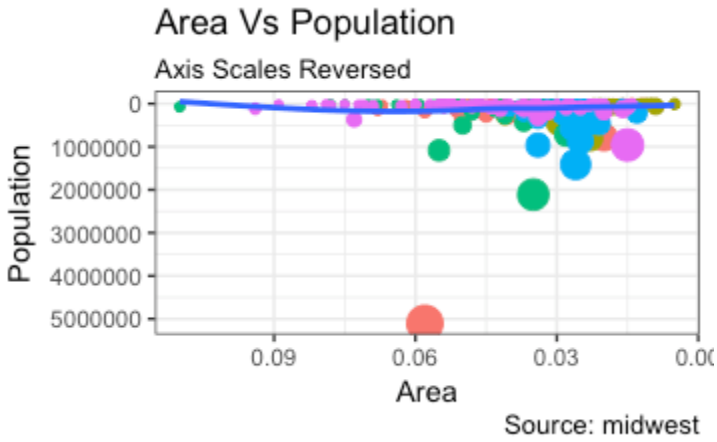
```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest", subtitle="X and Y axis Flipped") + theme(l
egend.position = "None")


# Flip the X and Y axis ----------------------------------------------
gg + coord_flip()
```

How to reverse the scale of an axis?

This is quite simple. Use `scale_x_reverse()` for X axis and `scale_y_reverse()` for Y axis.

```
library(ggplot2)


# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: midwest", subtitle="Axis Scales Reversed") + theme(
legend.position = "None")


# Reverse the X and Y Axis ---------------------------
gg + scale_x_reverse() + scale_y_reverse()
```

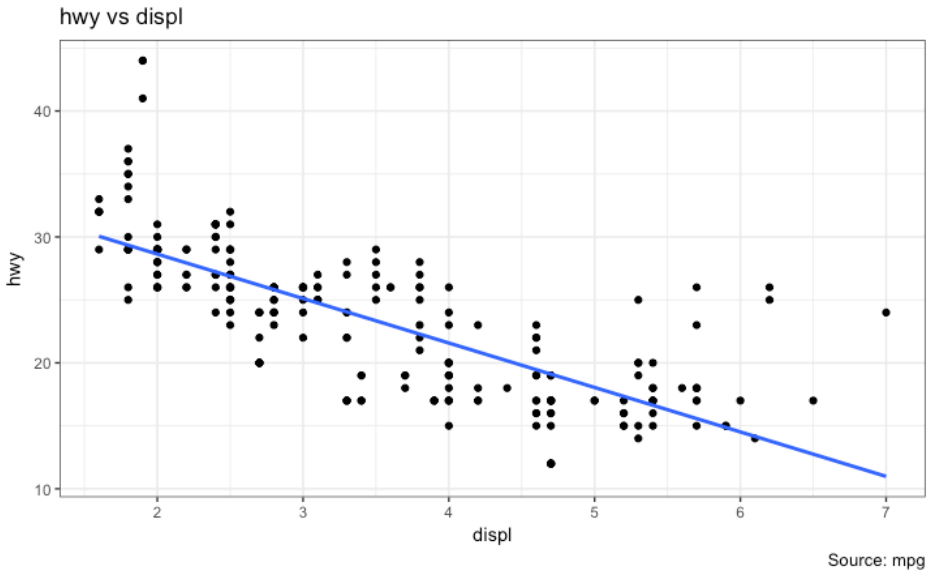




5. Faceting: Draw multiple plots within one figure

Let's use a the `mpg` dataset for this one. It is available in the ggplot2 package, or you can import it from this link.

```
library(ggplot2)
data(mpg, package="ggplot2")  # load data
# mpg <- read.csv("http://goo.gl/uEeRGu")  # alt data source

g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    labs(title="hwy vs displ", caption = "Source: mpg") +
    geom_smooth(method="lm", se=FALSE) +
    theme_bw()  # apply bw theme
plot(g)
```



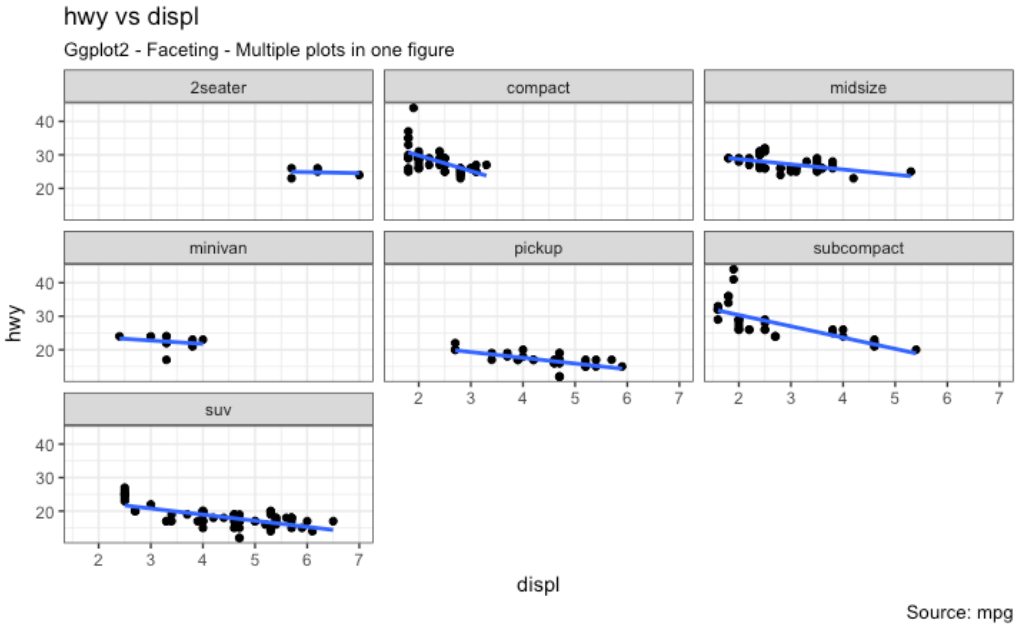
We have a simple chart of highway mileage `(hwy)` against the engine displacement `(displ)` for the whole dataset. But what if you want to study how this relationship varies for different classes of vehicles?

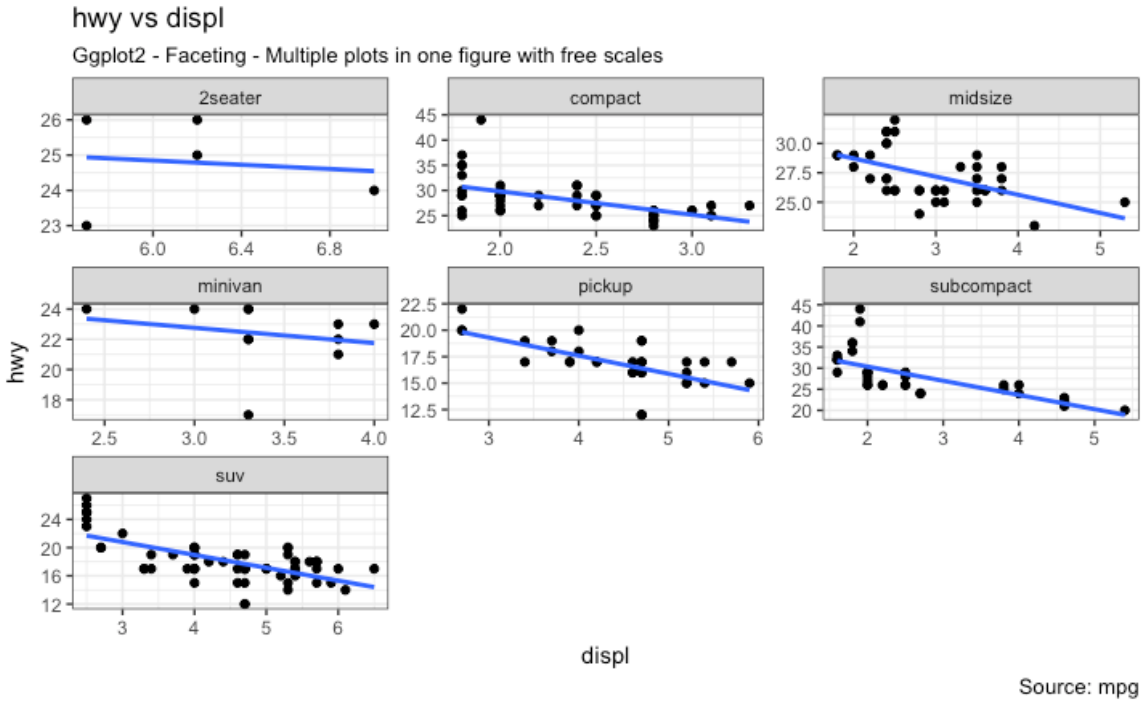
Facet Wrap

The `facet_wrap()` is used to break down a large plot into multiple small plots for individual categories. It takes a formula as the main argument. The items to the left of `~` forms the rows while those to the right form the columns.

By default, all the plots share the same scale in both X and Y axis. You can set them free by setting `scales='free'` but this way it could be harder to compare between groups.

```
library(ggplot2)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    geom_smooth(method="lm", se=FALSE) +
    theme_bw()  # apply bw theme


# Facet wrap with common scales
g + facet_wrap( ~ class, nrow=3) + labs(title="hwy vs displ", caption = "Source: mpg", subtitle="Ggplot2 - Faceting - Multiple plots in one figure")  # Shared scales


# Facet wrap with free scales
g + facet_wrap( ~ class, scales = "free") + labs(title="hwy vs displ", caption = "Source: mpg", subtitle="Ggplot2 - Faceting - Multiple plots in one figure with free scales")  # Scales free
```





So, What do you infer from this? For one, most 2 seater cars have higher engine displacement while the minivan and compact vehicles are on the lower side. This is evident from where the points are placed along the X-axis.

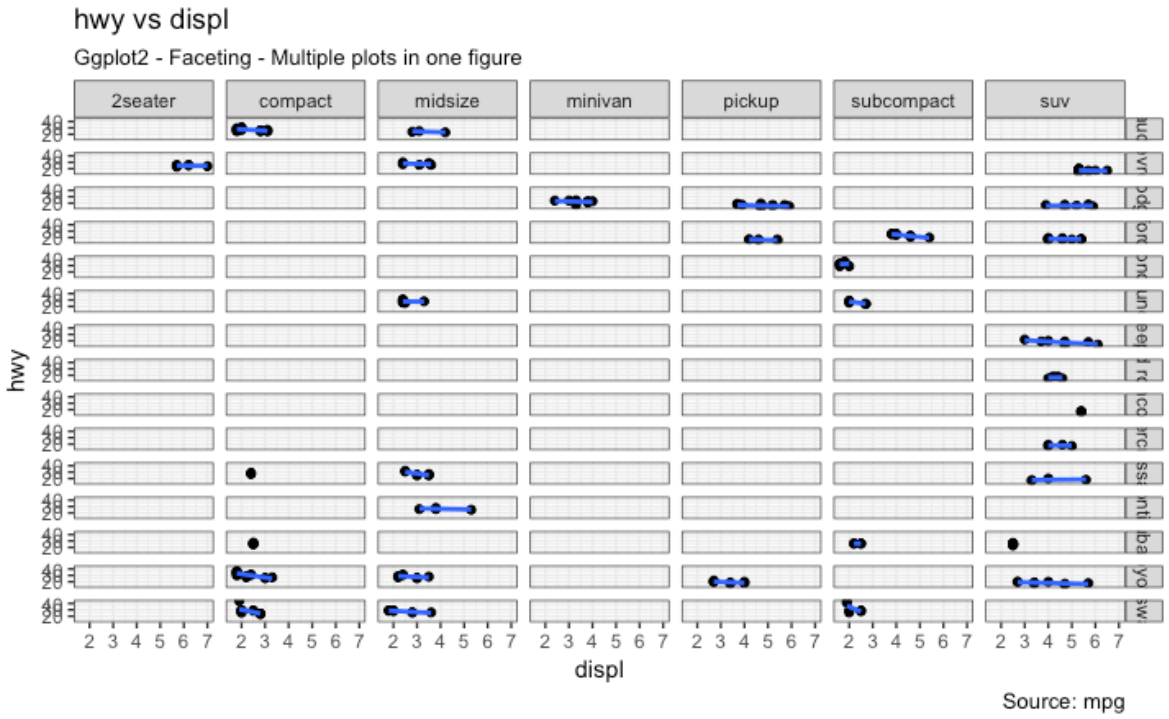
Also, the highway mileage drops across all segments as the engine displacement increases. This drop seems more pronounced in compact and subcompact vehicles.

Facet Grid

The headings of the middle and bottom rows take up significant space. The facet_grid() would get rid of it and give more area to the charts. The main difference with facet_grid is that it is not possible to choose the number of rows and columns in the grid.

Alright, Let's create a grid to see how it varies with manufacturer.

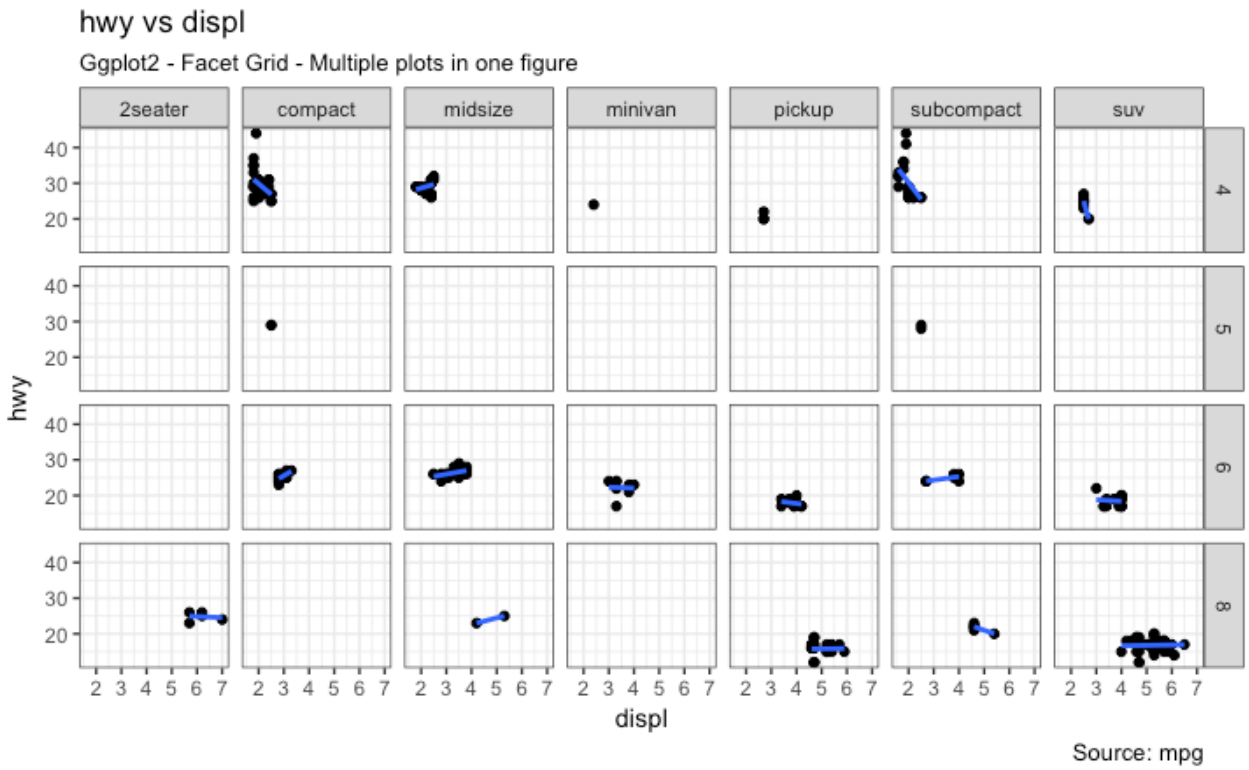
```
library(ggplot2)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    labs(title="hwy vs displ", caption = "Source: mpg", subtitle="Ggplot2 - Faceting - Multiple plots in one figure") +
    geom_smooth(method="lm", se=FALSE) +
    theme_bw()  # apply bw theme


# Add Facet Grid
g1 <- g + facet_grid(manufacturer ~ class)  # manufacturer in rows and class in columns
plot(g1)
```



Let's make one more to vary by cylinder.

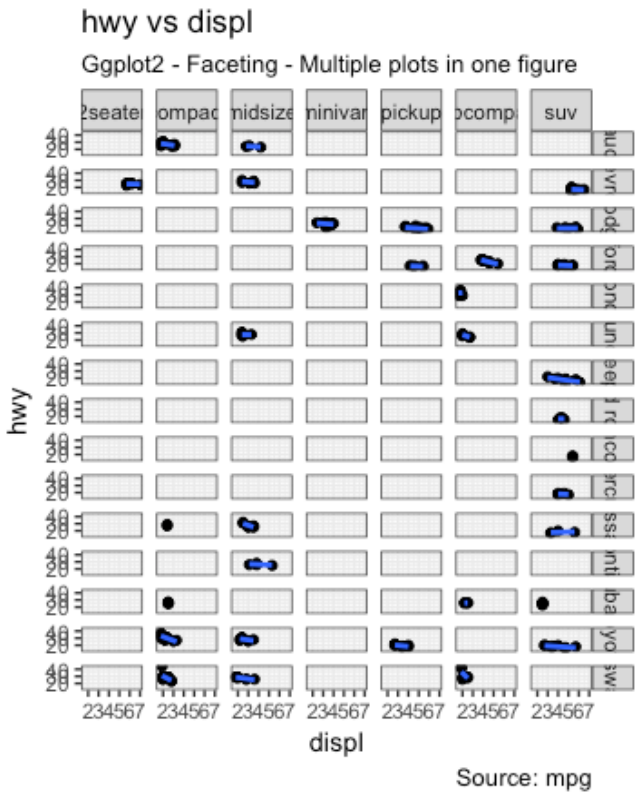
```
library(ggplot2)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    geom_smooth(method="lm", se=FALSE) +
    labs(title="hwy vs displ", caption = "Source: mpg", subtitle="Ggplot2 - Facet Grid - Multiple plots in one figure") +
    theme_bw()  # apply bw theme
```

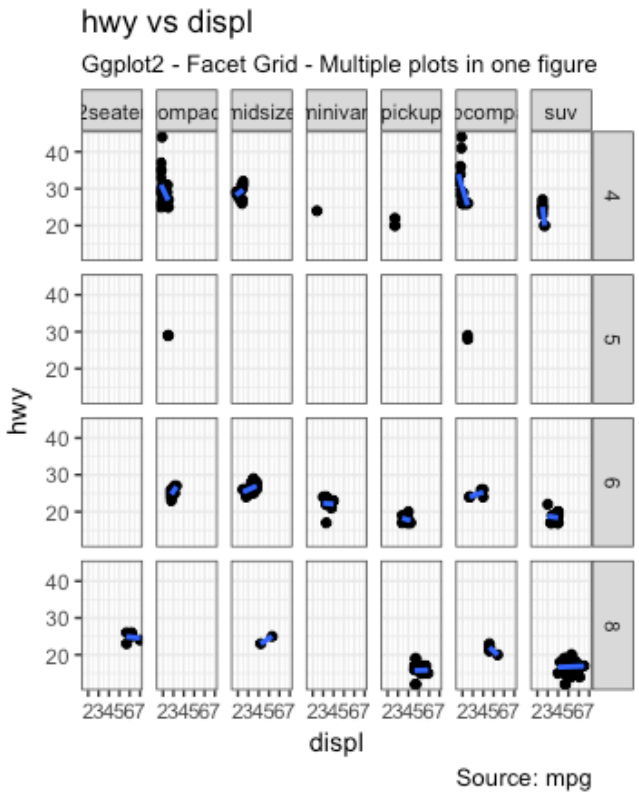
```
# Add Facet Grid
g2 <- g + facet_grid(cyl ~ class)  # cyl in rows and class in columns.
plot(g2)
```



Great!. It is possible to layout both these charts in the sample panel. I prefer the `gridExtra()` package for this.

```
# Draw Multiple plots in same figure.
library(gridExtra)
gridExtra::grid.arrange(g1, g2, ncol=2)
```



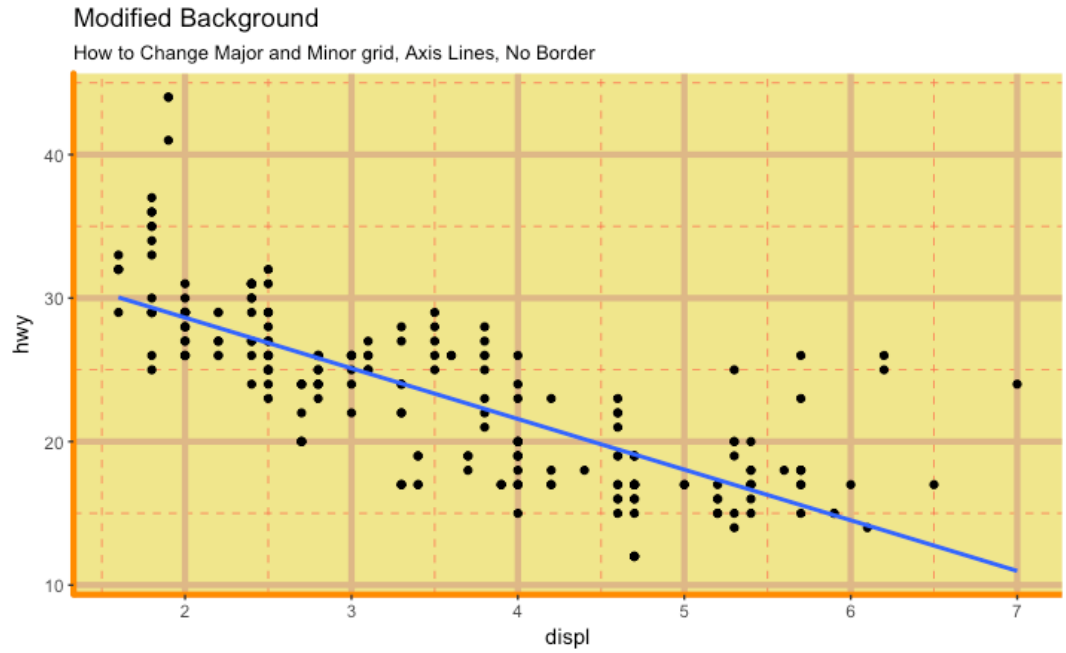


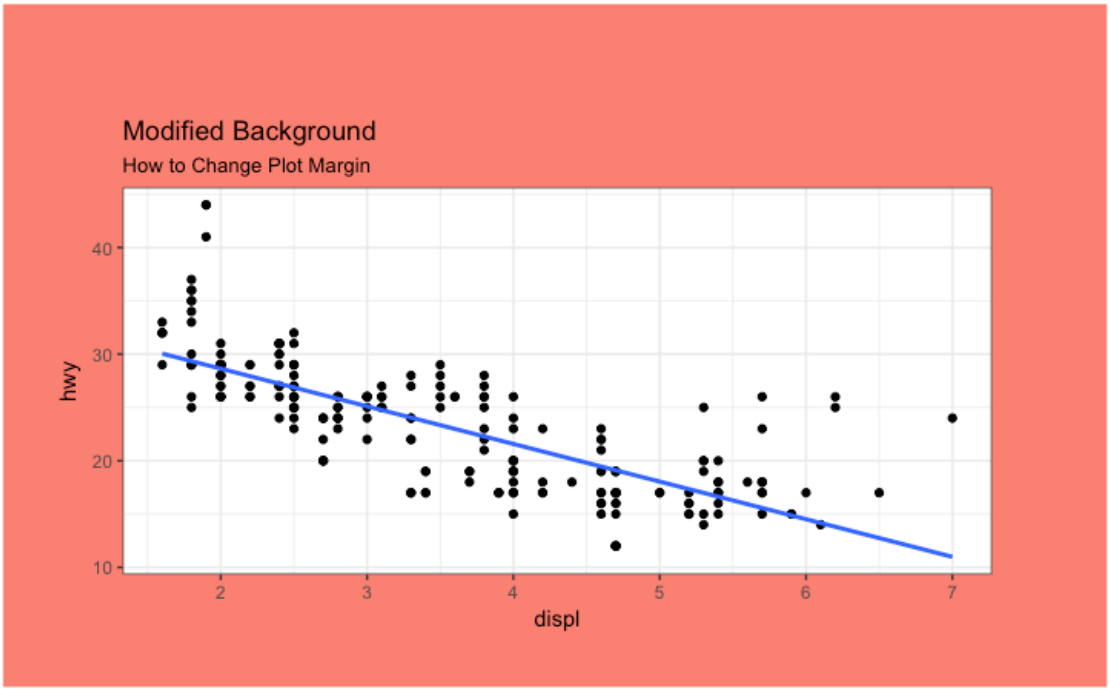
6. Modifying Plot Background, Major and Minor Axis

How to Change Plot background

```
library(ggplot2)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    geom_smooth(method="lm", se=FALSE) +
    theme_bw()  # apply bw theme


# Change Plot Background elements ----------------------------------
g + theme(panel.background = element_rect(fill = 'khaki'),
        panel.grid.major = element_line(colour = "burlywood", size=1.5),
        panel.grid.minor = element_line(colour = "tomato",
                                        size=.25,
                                        linetype = "dashed"),
        panel.border = element_blank(),
        axis.line.x = element_line(colour = "darkorange",
```

```
                        size=1.5,
                        lineend = "butt"),
      axis.line.y = element_line(colour = "darkorange",
                        size=1.5)) +
  labs(title="Modified Background",
       subtitle="How to Change Major and Minor grid, Axis Lines, No Border")


# Change Plot Margins ----------------------------------------------
g + theme(plot.background=element_rect(fill="salmon"),
       plot.margin = unit(c(2, 2, 1, 1), "cm")) +  # top, right, bottom, left
  labs(title="Modified Background", subtitle="How to Change Plot Margin")
```

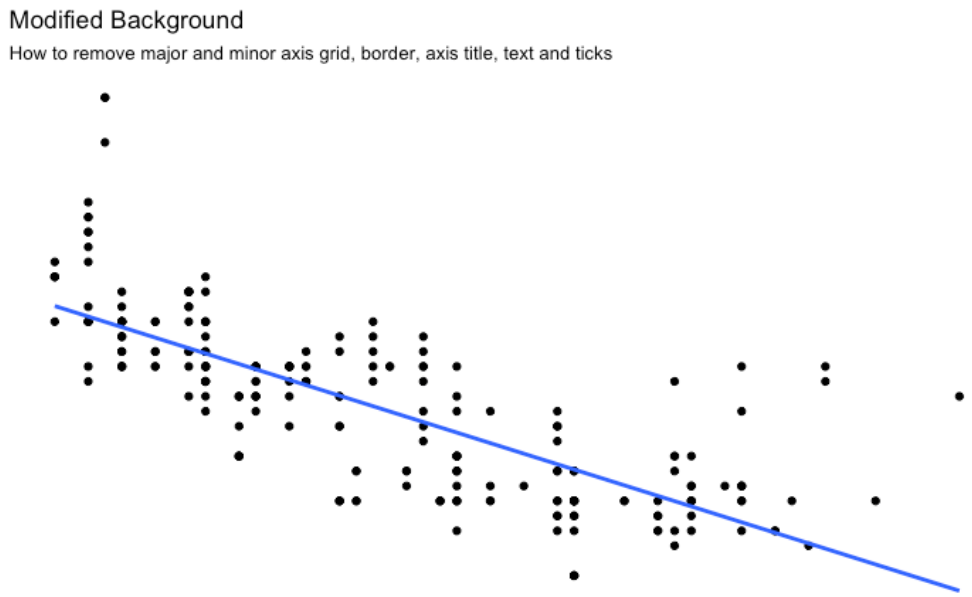




How to Remove Major and Minor Grid, Change Border, Axis Title, Text and Ticks

```
library(ggplot2)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    geom_smooth(method="lm", se=FALSE) +
```

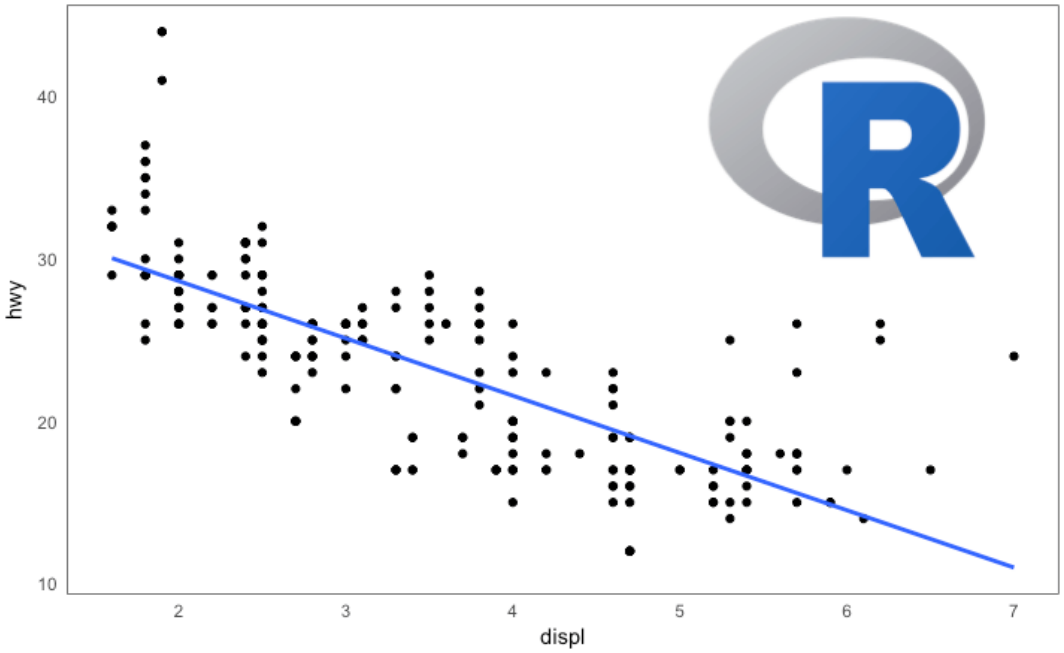
```
    theme_bw()  # apply bw theme


g + theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          panel.border = element_blank(),
          axis.title = element_blank(),
          axis.text = element_blank(),
          axis.ticks = element_blank()) +
  labs(title="Modified Background", subtitle="How to remove major and minor axis grid, border, axis title, text and ticks")
```



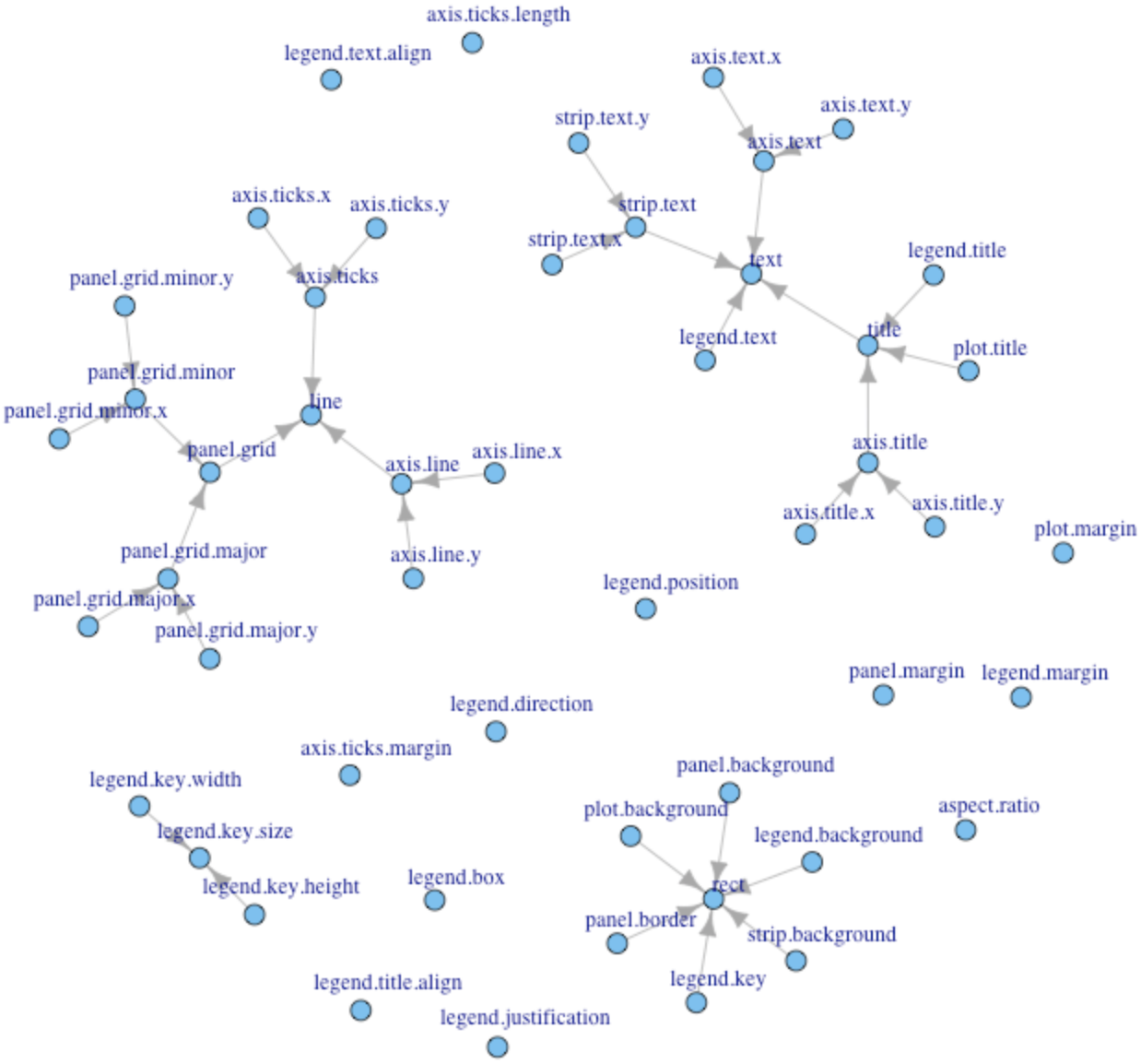
Add an Image in Background

```
library(ggplot2)
library(grid)
library(png)


img <- png::readPNG("screenshots/Rlogo.png")  # source: https://www.r-project.org/
g_pic <- rasterGrob(img, interpolate=TRUE)


# Base Plot
g <- ggplot(mpg, aes(x=displ, y=hwy)) +
    geom_point() +
    geom_smooth(method="lm", se=FALSE) +
    theme_bw()  # apply bw theme
```

```
g + theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          plot.title = element_text(size = rel(1.5), face = "bold"),
          axis.ticks = element_blank()) +
  annotation_custom(g_pic, xmin=5, xmax=7, ymin=30, ymax=45)
```



Inheritance Structure of Theme Components



source: http://docs.ggplot2.org/dev/vignettes/themes.html