Research Article

A conceptual introduction to ggplot2

Arindam Basu[1]

1. University of Canterbury, New Zealand

This is a conceptual introduction to ggplot2 in Rstudio. We take an available data set in R and work step by step to identify patterns and produce publishable quality graphics using the principles of grammar of graphics.

Introduction to ggplot2

GGplot stands for grammar of graphics plotting function. This is included in the tidyverse package of Rstudio. Here, let's learn a conceptual introduction to how to use ggplot2 in R to produce graphics and plots where you can generate insights into your data and present the plots for publications. In general, as Hadley Wickham paraphrases in ggplot2 book, in the grammar of graphics that Leland Wilkinson wrote a grammar that statistical graphic maps data onto aesthetic aspects of graphics such as colour of the graphic, the shape of the graphic, and size of geometric objects mainly points, lines, and bars. Besides this, the plot will contain statistical information and drawn on a coordinate system. If the data need to be subsetted and viewed as such, it need to be subsetted. In this scheme, all plots consist of five elements:

1. Data must be tidy and long form

Data are the source from where you will draw the graph/plot. Regardless of what data you will deal with, for ggplot to work, your data should be organised in **tidy format** and **long form**. Tidy format is where the data frame has one row per observation and one column per variable under study and each cell in the data contain only one information, as follows (Table 1):

Table 1. Organisation of tidy data for age and gender

ID	Age in years	Sex
Individual 1	20	Male
Individual 2	40	Female

As can be seen in the table, this data is organised such that ID as a column contains only information for one individual in the row. 'Age in years' and 'Sex' are two variables and the cells contain only one value in them. When you plot your graphs in ggplot2, make sure your data conform to the tidy format. For more information on tidy data format, see this entry on tidy data

Data must also be in long format (Table 2a and Table 2b)

Table 2a. Data in wide format for rate of disease D

Country	1999	2000	2001
New Zealand	20	21	22

Table 2b. Data in Table 2a organised in long format

Country	Year	Disease_rate
New Zealand	1999	20
New Zealand	2000	21
New Zealand	2001	22

Here you can see that the same data as in Table 2a as in Table 2b, but they are arranged differently. In the first table, some repeated measurement of the rate of disease for New Zealand was collected and presented in the years as columns. But in the second table (Table 2b), the same information on the rates were presented using a longer format where the name of the country was repeated three times to indicate three years for which the data were available. For more information on long and wide format of data, see the blog post by Karen Grace-Martin.

2. The plot will be built in layers

After the data, comes the considerations of the **geometry** and the **statistical information** that you want to see. What geometric features, in the speak of ggplot2, *geom*, would you like to plot? Would you like to plot points, lines, or bars, or polygons? These are the fundamental building blocks of building a plot. Learn more about geoms from the online ggplot2 book. All plots plot statistical information, in the language of ggplot2, these are referred to as **"stat"**; every geom is associated with some stat, and every stat has a corresponding geom. For example, if you want to plot a barplot of a categorical variable, you will first summarise the variable in counts or proportions and then you will plot these numbers. The counts and proportions are the stat for the geom of bar. Likewise, if you want to plot a scatterplot and then superimpose a regression line on the top of the scatter, your geom of line will have the associated stat of a smoothing procedure, such as linear regression or ("lm") or another form of smoothing function. The following entry in the manual of ggplot2 describes the stat function. Now we see that in this scheme, data is the **FIRST** layer in ggplot.

3. Plot must have scale

The word scale here means that there are data points in the database, these are now transformed as visual elements in the plot. Scale maps the data in the data set to the visual elements in the plot, so that one can be referenced with another. This means, using scale, you can specify the colour, size, and shape of a point, or draw the legend or axis, or set how the axis will be drawn (will it be drawn as a continuous scale or will it undergo log transformation?). Or you can use the scale function to modify the legend of a plot. For more information on scales, follow the link in ggplot2 book's online version

4. All plots must have coordinates

Every plot must have at least a pair of coordinates or a coordinate system. The coordinate system in ggplot2 is **coord**. This provides the axes and guidelines that are drawn on the plane of the graphic.

5. Plots can be drawn on small multiples, facets in ggplot speak

Edward Tufte introduced the concept of small multiples. Zach Gemignani in Juice Analytics blog paraphrased and cited Edward Tufte thus:

> Illustrations of postage-stamp size are indexed by category or a label, sequenced over time like the frames of a movie, or ordered by a quantitative variablenot used in the single image itself

This means you can create the same plot and repeat over several levels of another categorical variable. This will help to show how the relationship between two variables or say distribution of a single variable _vary_over different levels of another categorical variable. In ggplot, this is referred to as "facets" and facets can be wrapped or be presented in the form of a grid. We also refer to facetted charts as trellis chart or lattice chart or grid chart.

Small multiples in ggplot are referred to as "facets". In ggplot, it is not necessary that all plots will have to have facets, but if you have to show multivariate relationships, then facetting the relationship helps to reduce clutter and makes the charts easy to follow.

6. Plots are themed.

Themes control the appearance of the plot. For example, the background colour, the font size, whether there should be axes etc. For more on Tufte's principles, see Rajesh Sigdel's Medium Post in Nightingale.

Implementation of these principles in ggplot2 to make your life easier

This is the first of several posts, so in this post, we will take a simple dataset already within R and we will explore some of those principles. We will use Rstudio, tidyverse, and ggplot2 for our visualisations. If you want to work along with the codes I have presented here, please install the following in the order I have presented:

1. R for statistical computing
2. Rstudio
3. Tidyverse

R is a free open-source software programme for statistical data analysis and graphics (indeed statistical programming). You can obtain R from the link I indicated here. **After** you install R, please install Rstudio. If you do not want to install Rstudio, then, you can access rstudio in the cloud by visiting the following site:

https://rstudio.cloud

And creating a free account there.

Finally, after you have installed Rstudio or accessed Rstudio cloud, install the tidyverse package. Tidyverse is a versatile package that will enable you to conduct data sciene, data analysis and create graphics; it comes with ggplot2 package. You can install tidyverse in Rstudio by visiting the console or opening a code window and typing

```
install.packages("tidyverse")
```

Now you can get started. If you need more information and need to learn about how to use Rstudio, I recommend this dataquest.io tutorial

Step by step in creating graphics using ggplot2

Step 1. Load the packages Load `tidyverse` and `ggplot2` as shown in the code block to get started.

```
# load the library library(tidyverse) library(ggplot2)
```

Step 2. Get some data Here we will work with the `mtcars` data set on car brands and their performances and properties. In a later tutorial, how you can import data stored elsewhere and load your own data for plotting.

```
# start with a built in data set

head(mtcars)

# mtcars is a built in data set in R # we will use mtcars data set to show ggplot

# structure of mtcars str(mtcars)
```

Step 3. Set up a basic plot

Let’s start with a very basic plot.

```
# basic structure of ggplot

ggplot(mtcars, aes(x = wt, y = mpg)) +  geom_point() +

labs(title = "Miles per gallon for car weight",

x = "Weight of car",

y = "Miles per gallon") +
ggsave("plot01.png")
```

This results in the following plot:

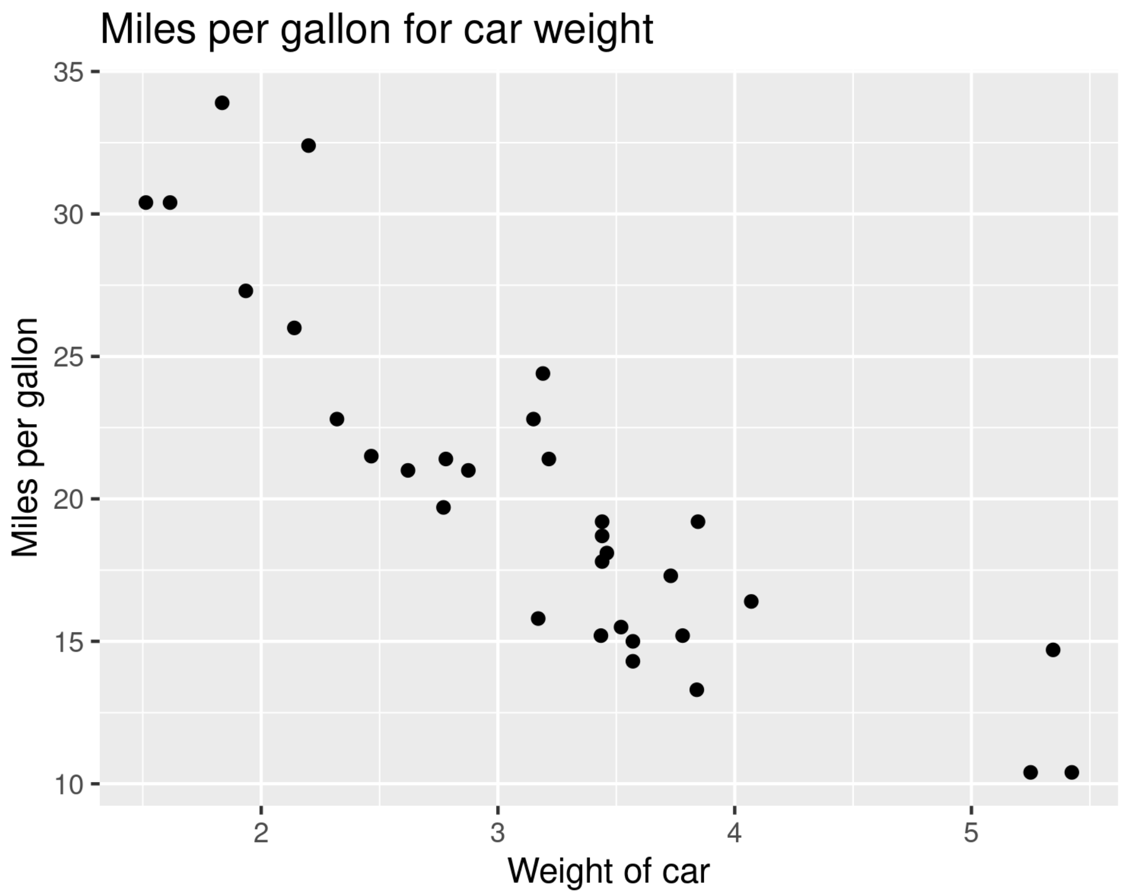


Figure 1. First image, a basic version

Note,

1. First line calls the `ggplot()` function, it must have two *required* things:
2. It must have the data, in this case `mtcars`, and
3. It must have the `mapping` which is an aes() function
4. The `aes()` function must include the information as to the x and y variables (note that we do not have to include the name of the dataset)
5. The first line ends with a `+` sign. The sign must be placed at the end of the first line, otherwise it will fail!

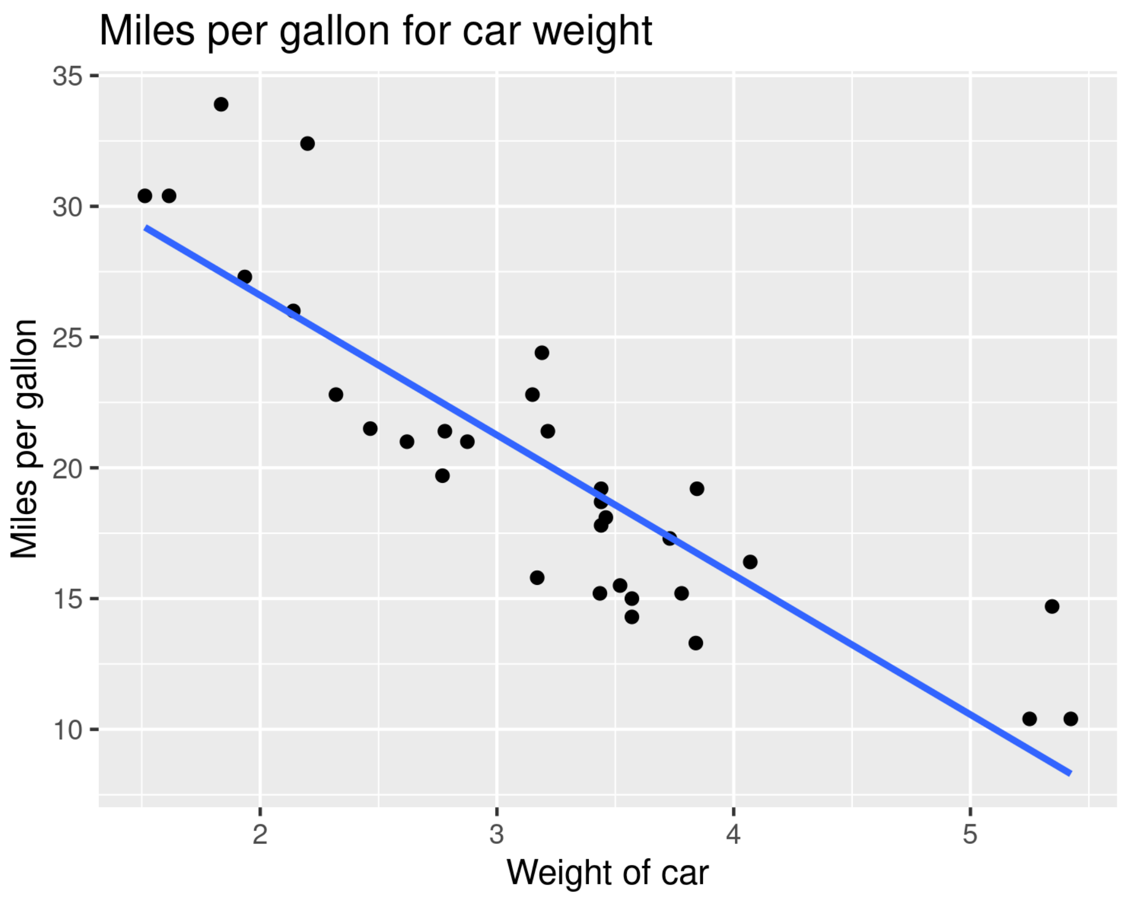
6. Line 2 (or Layer 2) is the **second layer** and specifies the geometry, given by `geom_` and here we want points as this is a scatterplot
7. The layer 3, we have added the labels that will go with the plot.

This is a minimalist plot, and it suggests that for all cars, as their weight increase, their milage drops. But there are different types of cars in the database, is it true for all of them? Besides, what is the nature of the relationship? For this we will add a regression line to the plot.

Step 4. Add a regression line

```
ggplot(mtcars, aes(x = wt, y = mpg)) +  geom_point() +

 geom_smooth(method = "lm",

            se = F) +

 labs(title = "Miles per gallon for car weight",       x = "Weight of car",

      y = "Miles per gallon") +  ggsave("plot02.png")
```

This produces



Note,

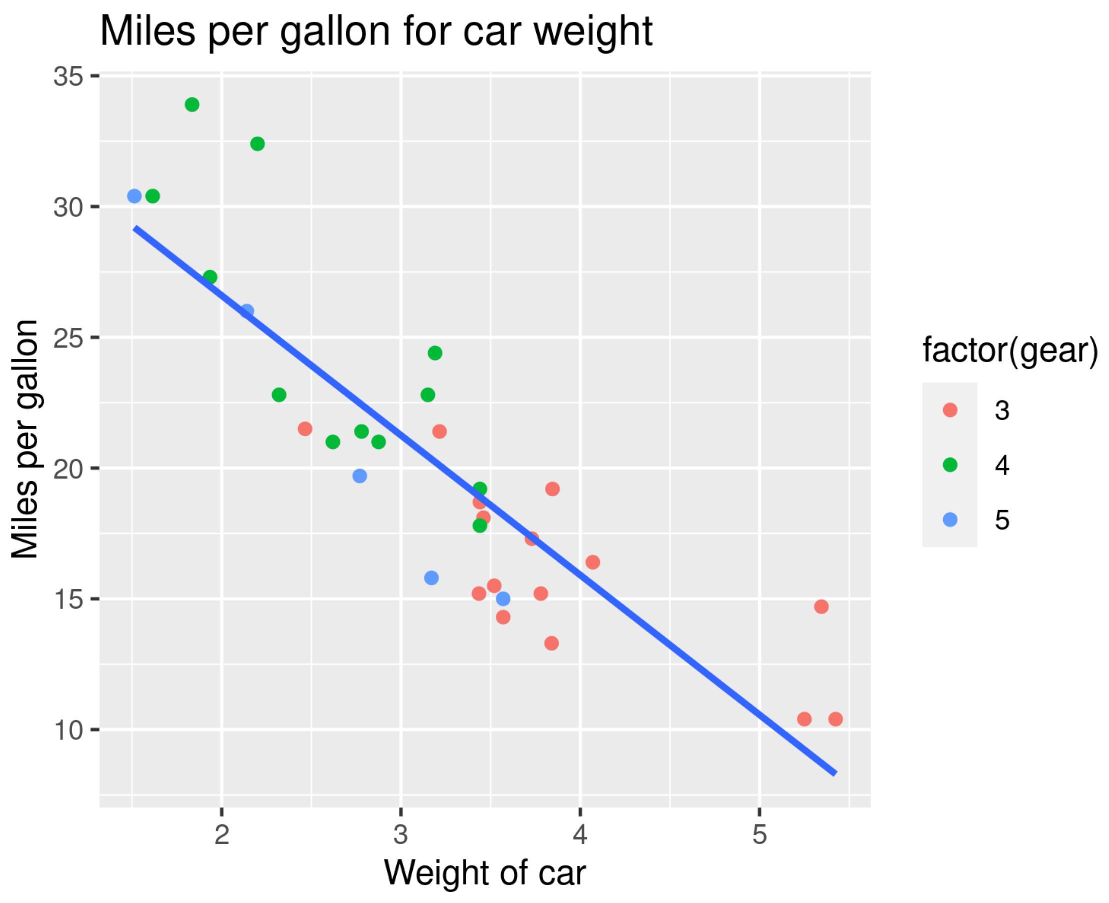
1. In layer 3, we added geom_smooth() because we wanted to see how smoothing the points, what summary figure we would be able to visualise.
2. We specified that it be linear model, hence "lm". For details, see this tutorial
3. We did not want the standard error band, so we asked `se = F`, `F` stands for `false`.

Step 5. Subset the data and view again

Now we want to see how are the points distributed for the gear types in the car, as we believe that cars with different gears may have different relationships between milage and weightage. So we do:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +  geom_point(aes(color = factor(gear))) +
 geom_smooth(method = "lm", se = F) +  labs(title = "Miles per gallon for car
weight",       x = "Weight of car",
      y = "Miles per gallon") +  ggsave("plot03.png")
```

This produces:



Note:

In Layer 2, we added an aes() in the geom_point() and there we had specified `color = factor(gear)`.

A few things:

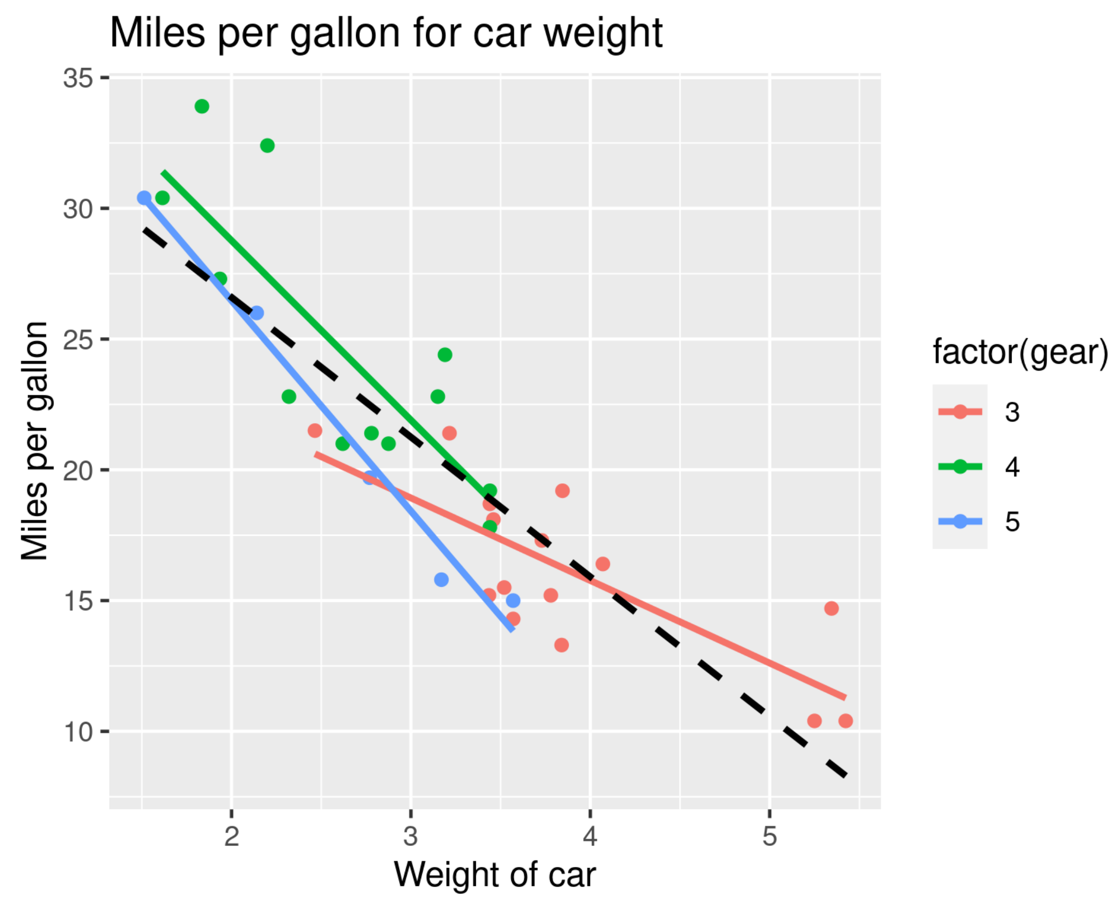
First, note that the colours are best if the classifying variable, in this case `gear` is a categorical variable. Otherwise, you will see a range of colours. This is fine but in this context, we want a categorical variable to bring out the difference in the colours in the points. But because the variable `gear` in our dataset was a continuous variable, we had to convert the variable to a categorical variable using the `factor()` function. This converted the variable to a categorical/factor variable. For more about factors in R, read this tutorial. Second, we specified that as we colour the points, we will colour them based on the "factorised" gear variable. This made sure that for the three categories in the factorised gear variable, they had three colours in the points.

Step 6. View the regression line for separate groups

At this stage, we want to view the different coloured points, then the regression lines for each subset of the points, and superimpose on that pattern a regression line for the entire set of data points.

```
ggplot(mtcars, aes(x = wt, y = mpg)) +   geom_point(aes(color = factor(gear))) +
 geom_smooth(method = "lm",
            se = F,
            aes(color = factor(gear))) +  geom_smooth(method = "lm",
            color = "black",
            se = F,
            linetype = "dashed") +
  labs(title = "Miles per gallon for car   weight",
x = "Weight of car",
y = "Miles per gallon") +
ggsave("plot04.png")
```

This produces:



Note:

1. We have added two `geom_smooth()` layers! One on top of the other. The sequence does not matter
2. In Layer 3 (or line 3), in the `geom_smooth()`, we added an `aes()`; why? this is because we wanted these lines to be at the variable level
3. In layer 4 (or line 4), in the `geom_smooth()` was for EVERY POINTS in the scatterplot, this is why we did not use the `aes()` function. But we made the line a dashed line with `linetype = "dashed"`.

So now you see the scatterplot changed with points, and four lines showing how the relationship between car weight and car milage varied with the gears.

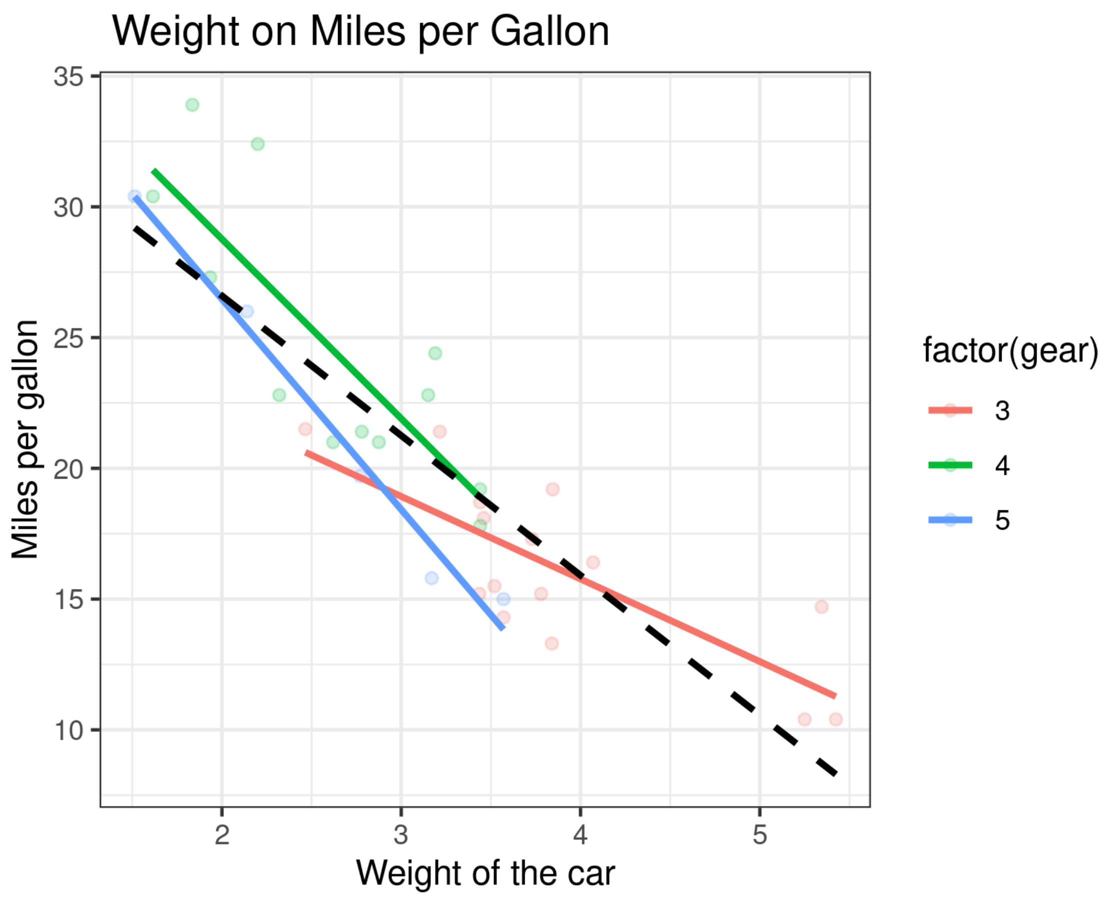
Step 7. We make these lines more prominent

By now, we are seeing that while in *general*, as the weight of the cars increase, there is a corresponding drop in milage, such slope of the drop is different for different number of gears. We are interested in studying the changes in the slope, but the gray background and colourful points are distracting us. So, now we will tidy up the graph by making the background white, making the points somewhat fade.

So we do this:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +  geom_point(alpha = 0.2,

  aes(color = factor(gear))) +  geom_smooth(method = "lm", se = F,

 aes(color = factor(gear))) +  geom_smooth(method = "lm",

          color = "black", se = F,                        linetype = "dashed") +

 labs(x = "Weight of the car",

      y = "Miles per gallon",

      title = " Weight on Miles per Gallon") +  theme_bw() +

ggsave("plot05.png")
```

This produces



Notes:

- The first thing we wanted to do was to make the background grey theme magically transform into white, and this we did with changing the `theme` of the plot. Hence we selected `theme_bw()` as the last layer
- We added an `alpha` channel to the `geom_point()` such that it would apply to ALL the points in the plot the same way. The `alpha` controls the transparency of the points so that an **alpha value of 1 is opaque** and an **alpha value of 0** is transparent, and the lower the alpha value, the more transparent the points.

Step 8. Remove the gridlines and make the legend legible

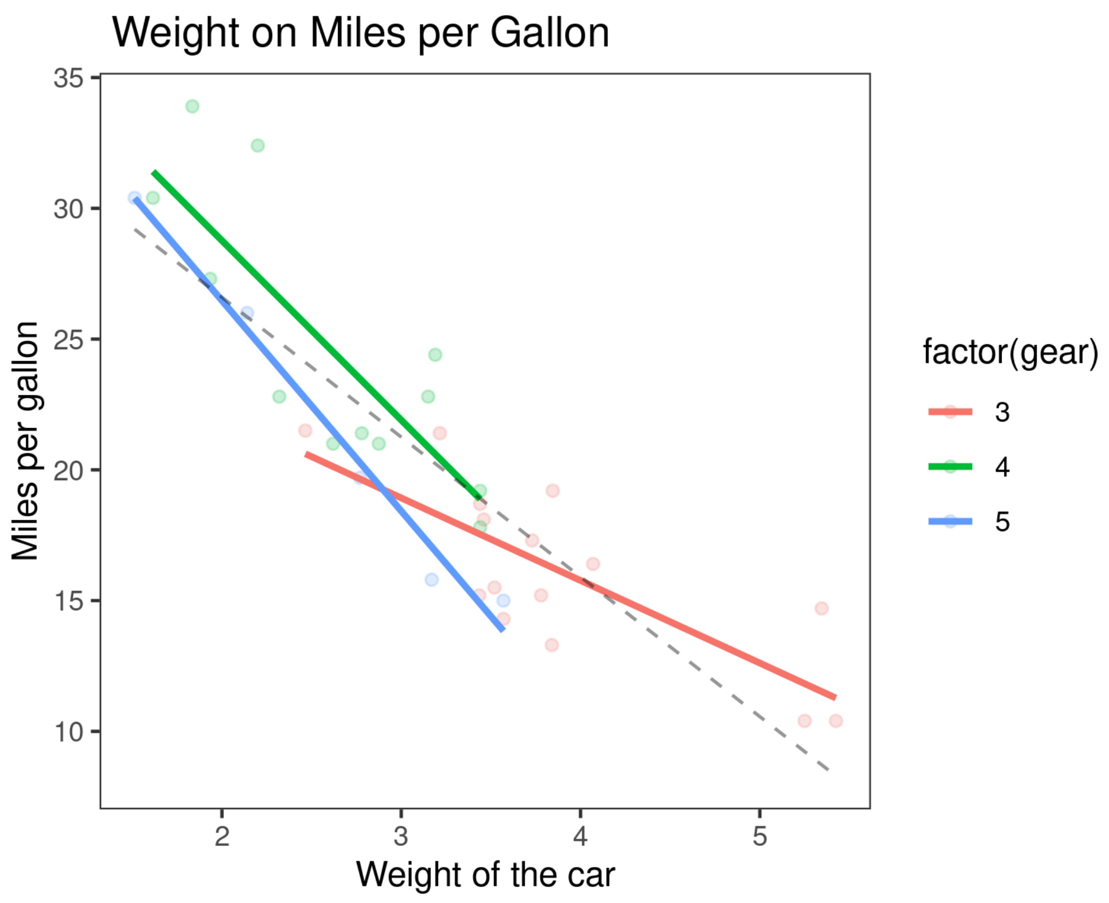
At this stage, the plot is informative and shows that while there is a drop in milage with car weight, the slope varies with the gear size of the cars. We want to make this information stand out further by:

- Removing the major and minor gridlines to minimise visual distraction
- Make the overall regression line (the dashed line) more transparent

So we do as follows

```
  ggplot(mtcars, aes(x = wt, y = mpg)) +          geom_point(alpha = 0.2, aes(color
= factor(gear))) +

geom_smooth(method = "lm", se = F,             aes(color = factor(gear))) +

geom_line(stat = "smooth",            method = "lm", color = "black", se = F,
       linetype = "dashed",

alpha = 0.4) +

labs(x = "Weight of the car",      y = "Miles per gallon",      title = " Weight on
Miles per Gallon") +  theme_bw() +

theme(panel.grid.major = element_blank(),       panel.grid.minor = element_blank())
+  ggsave("plot06.png")
```

This produces:



Notes:

- We have two themes in two different layers.
- The first is a black and white theme that we selected and stays
- The second theme is where we work with the major and minor grids in the panel. The `element_blank()` function makes the grid elements disappear
- Also, note that as before, we add an `alpha = 0.4` to blur out the dashed regression line

Step 9. Make the legend legible

So far, we have achieved that

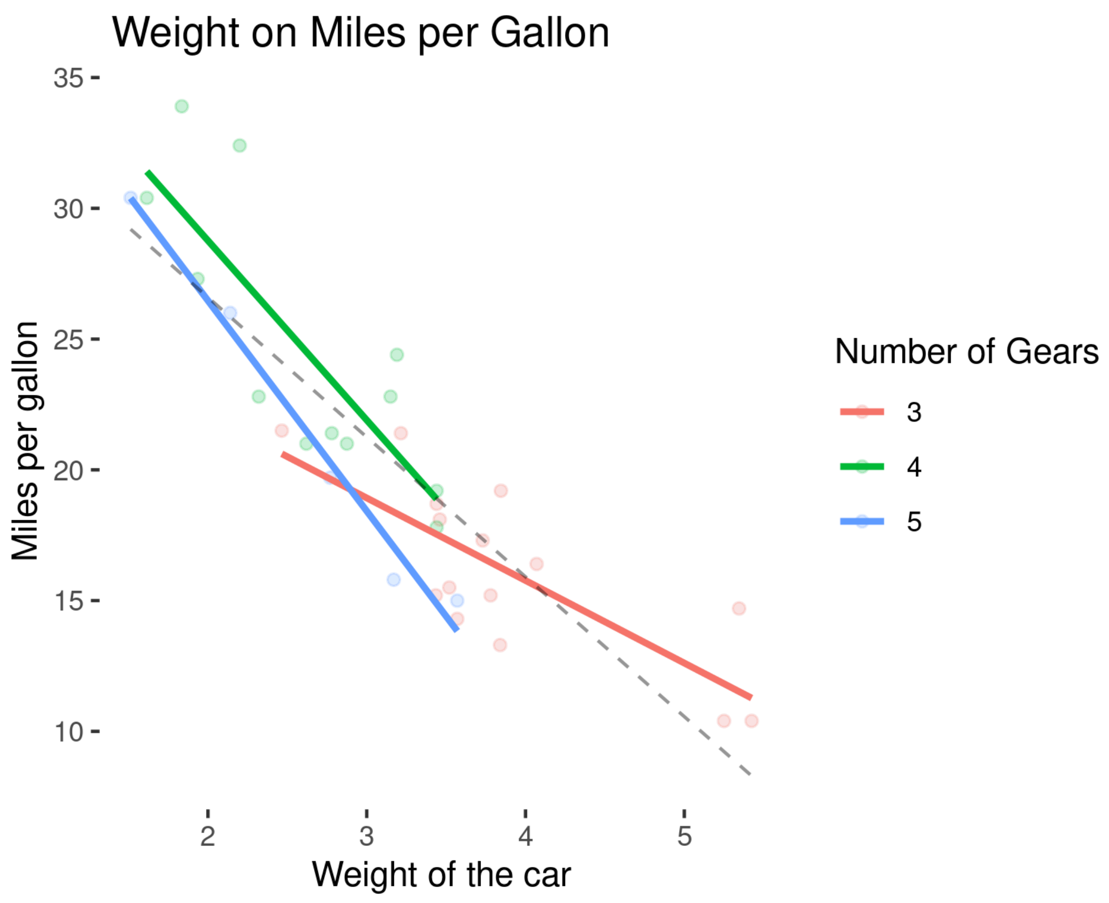
- We have made the three regression lines prominent
- We have faded out the data points and the overall regression line
- The graph has highlighted the points that we wanted to highlight

However, the legend on the right hand side has the headline `factor(gear)` makes no sense to anyone who does not have an appreciation of what factor means. To help the reader further:

- We will change the headline there
- And we will take out the border around the plot
- So we do:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +   geom_point(alpha = 0.2, aes(color =
factor(gear))) +

geom_smooth(method = "lm", se = F,              aes(color = factor(gear))) +

geom_line(stat = "smooth",

method = "lm", color = "black", se = F,              linetype = "dashed",


alpha = 0.4) +

labs(x = "Weight of the car",      y = "Miles per gallon",      title = " Weight on
Miles per Gallon") +  theme_bw() +

theme(panel.grid.major = element_blank(),        panel.grid.minor = element_blank(),
         panel.border = element_blank()) + scale_color_discrete(name = "Number of
Gears") +  ggsave("plot07.png")
```

This produces:



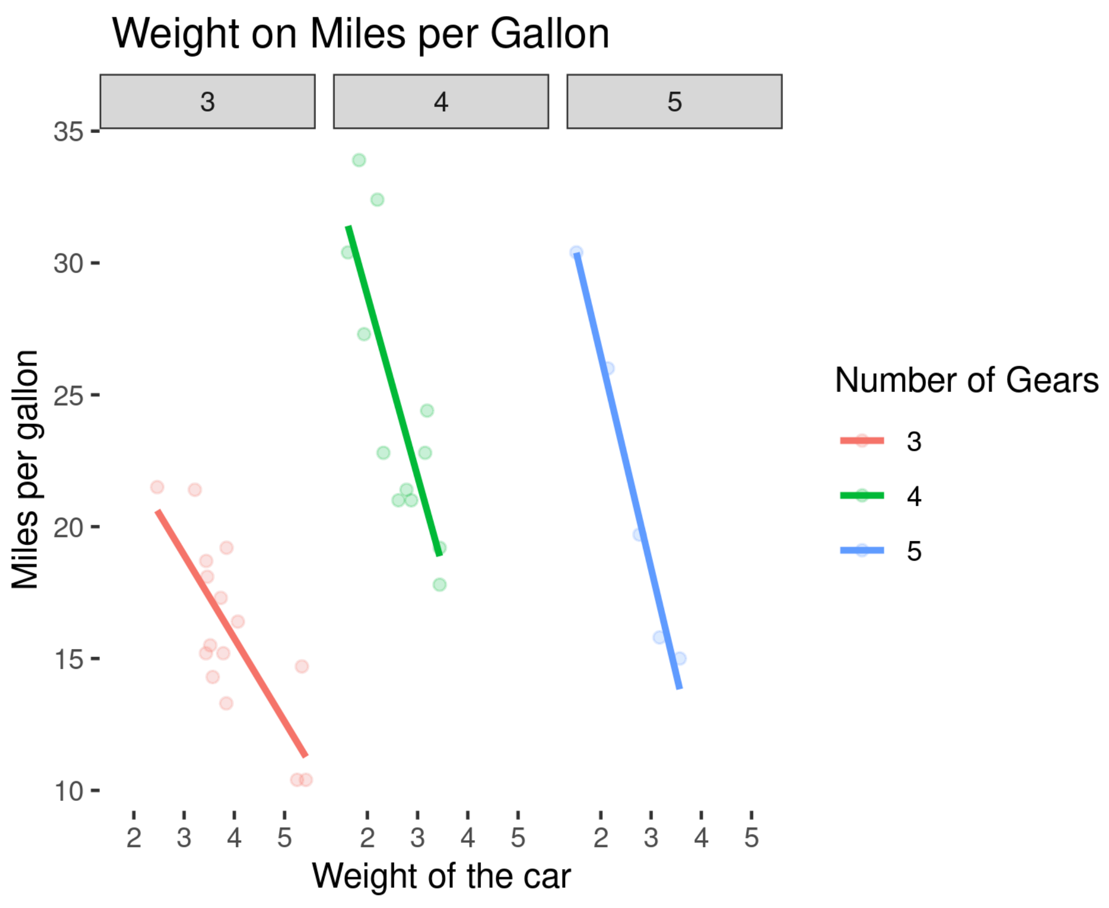
Note:

- Now we introduce another concept, that of **scales**, so
- `scale_color_discrete()` is where we map from data space the discretised gear variable factor(gear) to the graph aesthetic space and we change the name of the legend as the legend was based on "color" property

However, while the graph now clearly shows the three regression lines, this plot is still quite complex. It'd be useful to use the concept of "small multiples" to see this in a panel of three plots separately side by side. So, we do this:

```
 ggplot(mtcars, aes(x = wt, y = mpg)) +  geom_point(alpha = 0.2, aes(color =
factor(gear))) +

geom_smooth(method = "lm", se = F,              aes(color = factor(gear))) +

labs(x = "Weight of the car",      y = "Miles per gallon",      title = " Weight on
Miles per Gallon") +
theme_bw() +

theme(panel.grid.major = element_blank(),        panel.grid.minor = element_blank(),

         panel.border = element_blank()) + scale_color_discrete(name = "Number of

Gears") +  facet_wrap(~factor(gear)) +  ggsave("plot08.png")
```

Now we get:



Note:

- We have the concept of faceting where we have divided the single plot based on categorical variable into small multiples of panels
- Our categorical variable came from converting the continuous variable `gear` to a factor with three levels. Hence we get to see three panels in the frame.
- We used `facet_wrap` to wrap around the three variables. However, this is a flexible approach, where we could use `facet_grid()` function if we wanted to make even more complex subdivisions of panels.

Summary

In this introduction, we have used `ggplot()` within tidyverse to take a dataset and started with a barebones plot of two variables and using a step by step method, we converted that plot to a plot of three panels that clearly showed how the relationship between the milage of a car varies with the weightage of the car but varies with the number of gears. This example shows that `ggplot()` can be used to develop refined plots in preparation for testing and working for your models. We will continue to explore this topic with other types of plotting as we discuss modelling and machine learning in data science.

Declarations

Funding: The author(s) received no specific funding for this work.

Potential competing interests: The author(s) declared that no potential competing interests exist.