Advanced graphics with ggplot2

Introduction

ggplot2 is an R package for producing publication quality graphics, giving a very professional appearance. Unlike most other graphics packages, it has an underlying *grammar* (see e.g. Wilkinson, 2005, *The Grammar of Graphics*, Springer, 2nd edn). It is included as part of the Tidyverse suite of packages.

One appealing feature of ggplot2 is that it can be used to build up plots in layers that you can stack and reorder easily. It automatically constructs the appropriate plot at the end of all commands, so no more worrying about order of commands for x/y axis scales, etc.

```r
# Either ...
#library("tidyverse")
# for all tidyverse packages
# OR, for just plotting
library("ggplot2")
```

ggplot()

Every plot starts with this function. Some optional arguments are:

- data to specify the data frame containing the variables we later reference in the plot
- mapping to specify what variables map to the x axis, y axis, colour legend, etc etc

Mappings are always specified by a call to aes(). For example, the following code chunk gives the plot on the next page.

```r
data("diamonds", package = "ggplot2")
ggplot(diamonds, aes(x = carat, y = price))
```

We use "Geoms" to specify how data is plotted by "adding" + to the plot. For example:

```r
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point()
```

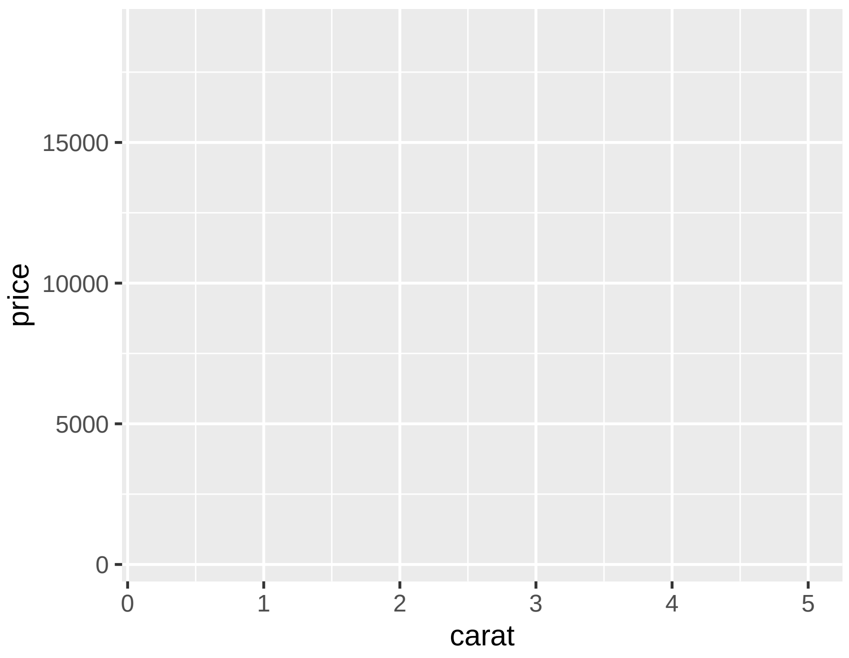


Figure 1: It's blank! We haven't specified what plot to make, only what data to use!

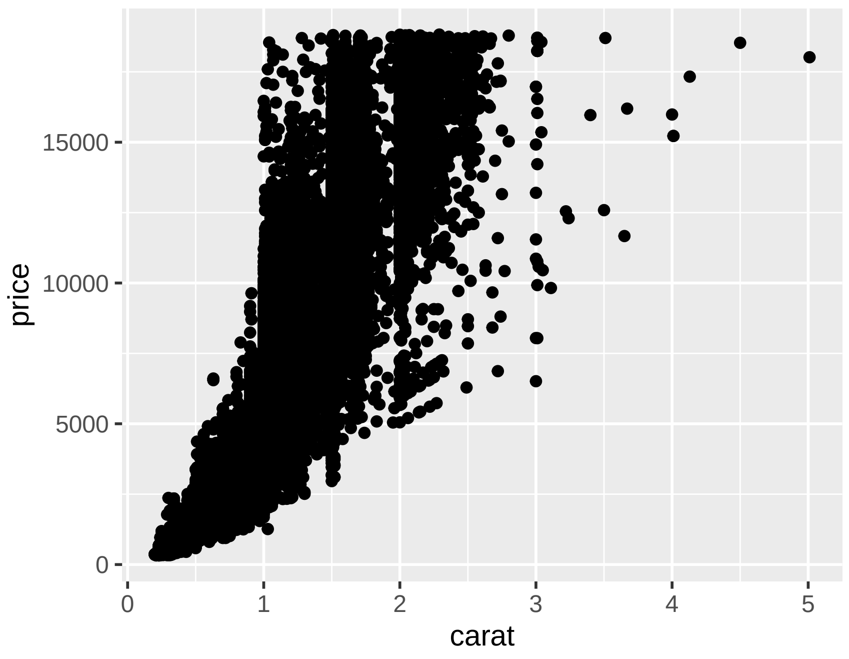


Figure 2: A scatter plot of price versus carat.

Geoms

Geoms are geometric objects that inherit the data and mapping from the original `ggplot()` call, but can be overridden (or added to with `aes`). Some Geoms have their own special set of options relevant to the plot type. Each Geom builds up layers in the order you call them (so to change any overplotting, change the order of addition to the plot).

Legends etc for colours, line types etc are all handled automatically. For example, consider the following code and resulting plot:

```r
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(aes(colour = cut), size = 0.2) +
  geom_smooth(aes(colour = cut)) +
  xlab("Number of carats") + ylab("Price in $")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

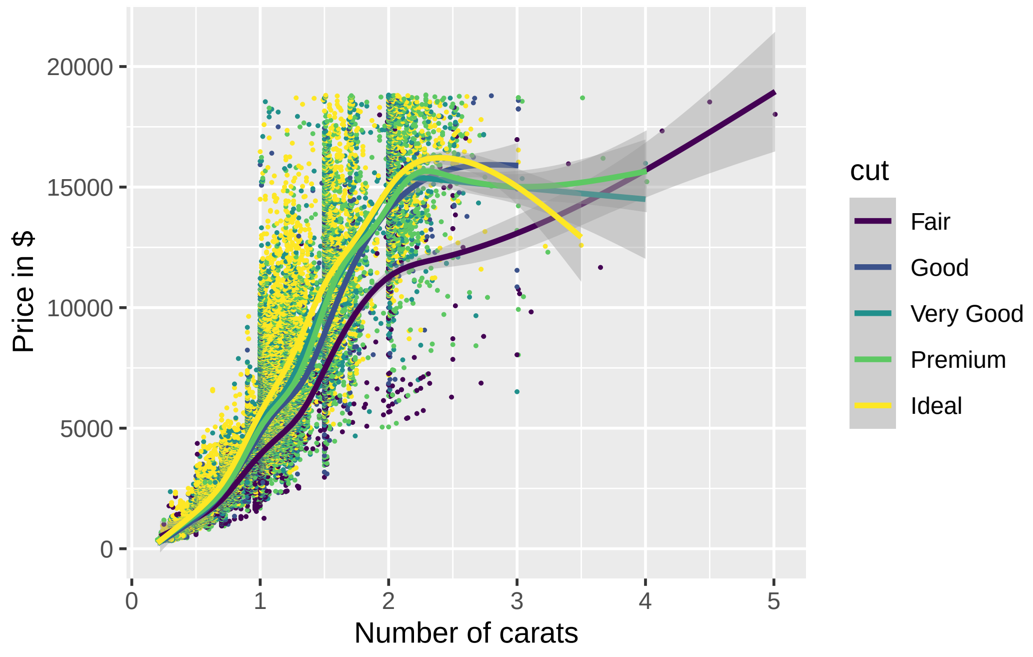


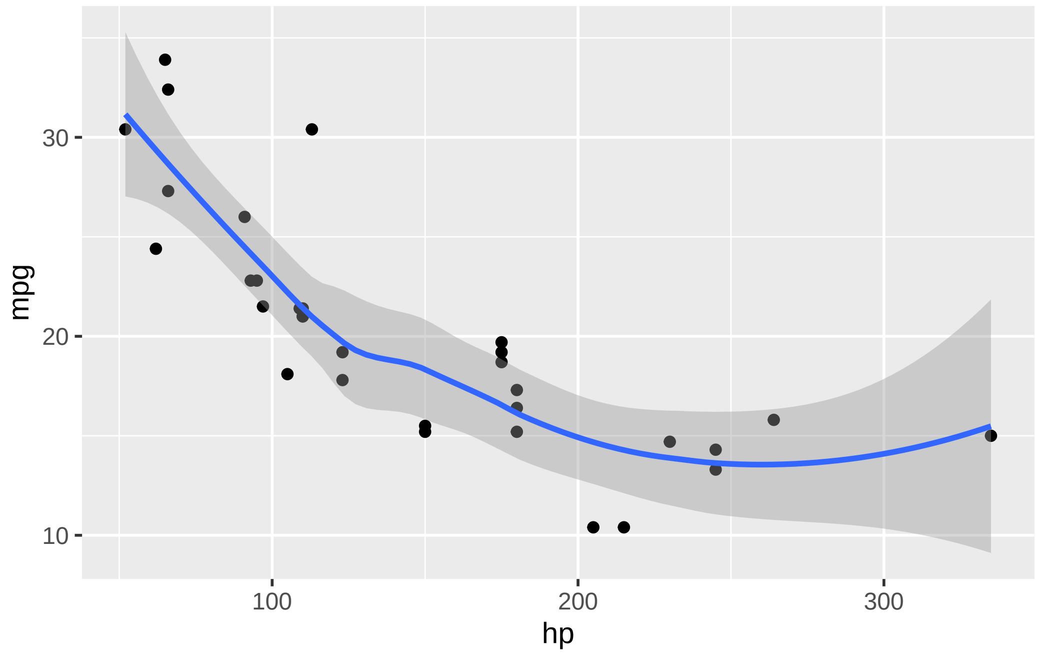
Figure 3: Scatter plot coloured by ‘cut’, added smoother coloured by ‘cut’, x and y labels updated.

Plots as variables

Most of the time you will create a plot object and immediately plot it, but you can also save a plot to a variable and manipulate it. For example:

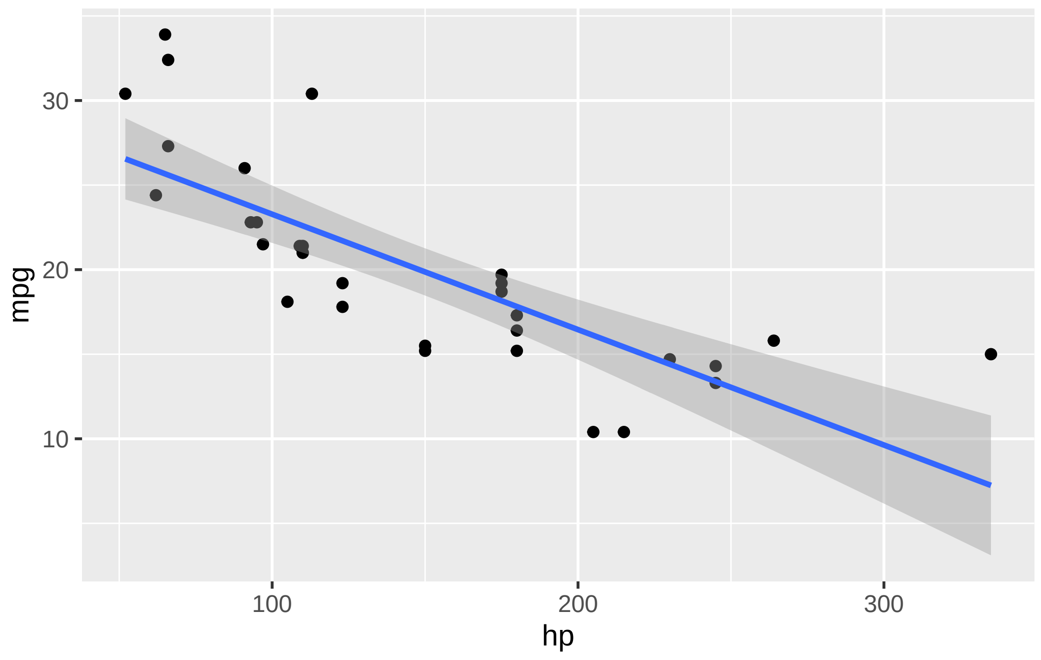
```
data("mtcars")
p <- ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point()
p + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

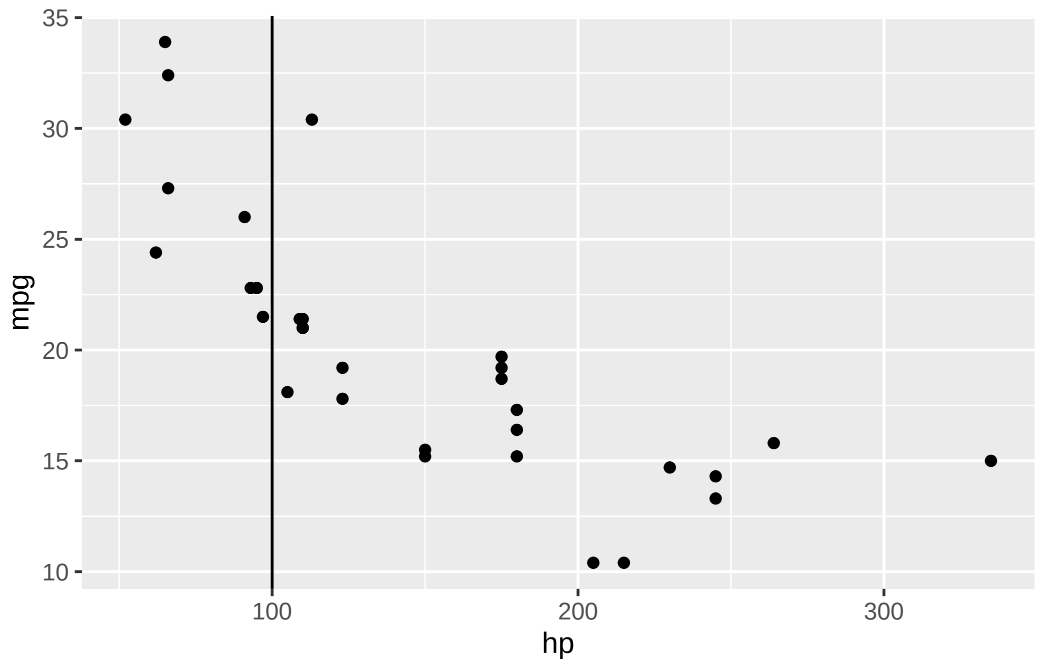


```
p + geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'
```



```
p + geom_vline(xintercept = 100)
```

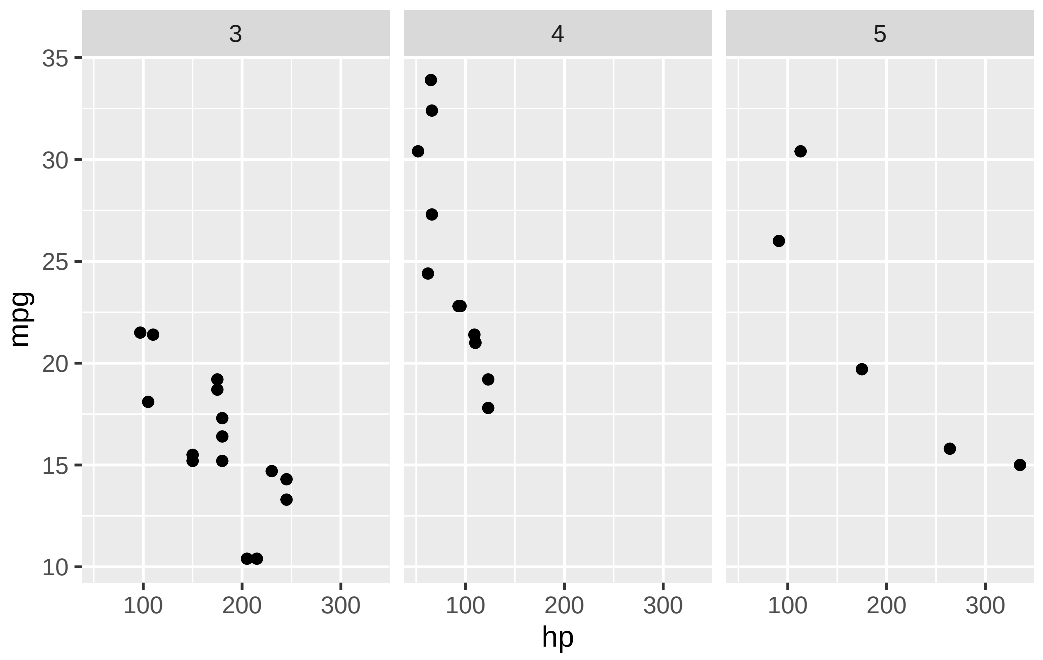


Faceting

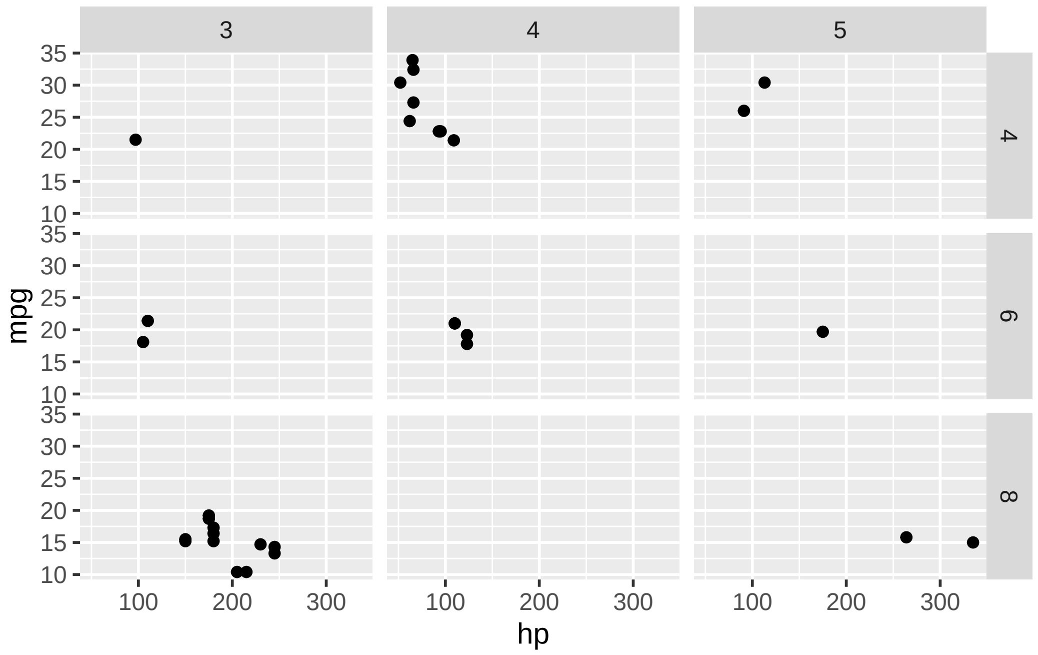
Faceting enables splitting your data into multiple plots according to a categorical variable.

- `facet_wrap()`
  - single variable split
  - formula notation to indicate splitting variable ∼ `var`
- `facet_grid()` two variable split
  - formula indicating both splitting variables `rows_var` ∼ `cols_var`

```r
ggplot(mtcars, aes(x = hp, y = mpg)) +
  facet_wrap(~ gear) +
  geom_point()
```



```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  facet_grid(cyl ~ gear) +
  geom_point()
```



Key resources

Documentation reference:

https://ggplot2.tidyverse.org/reference/index.html

Cheat sheet:

https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf

ggplot2 is a huge project so you need to read the docs to learn it! But, picking it up once you have the basics is easy, because of a very coherent interface.