DS 116 Data Visualization

Introduction to ggplot and layered grammar of graphics

Habet Madoyan

American University of Armenia

Grammar of graphics

There are few definitions of what grammar is
"the whole system and structure of a language or of languages in general, usually taken as consisting of syntax and morphology (including inflections) and sometimes also phonology and semantics"
grammar is "the fundamental principles or rules of an art or science"

Grammar of graphics

Grammar gives language rules. The word stems from the Greek noun for letter or mark. And that derives from the Greek verb for writing, which is the source of English word graph. Grammar means, more generally, rules for art and science, as in the richly illustrated The Grammar of Ornament (Jones, 1856), and Karl Pearson's The Grammar of Science (Pearson, 1892)
(Leland Wilkinson)

Grammar of graphics

Grammar has a technical meaning in linguistics. In the transformational theory of Chomsky (1956), a grammar is a formal system of rules for generating lawful statements in a language.
(Leland Wilkinson)

Grammar of graphics

Grammar makes language expressive. A language consisting of words and no grammar (statement = word) expresses only as many ideas as there are words. By specifying how words are combined in statements, a grammar expands a language's scope.

Grammar of graphics

(Leland Wilkinson)

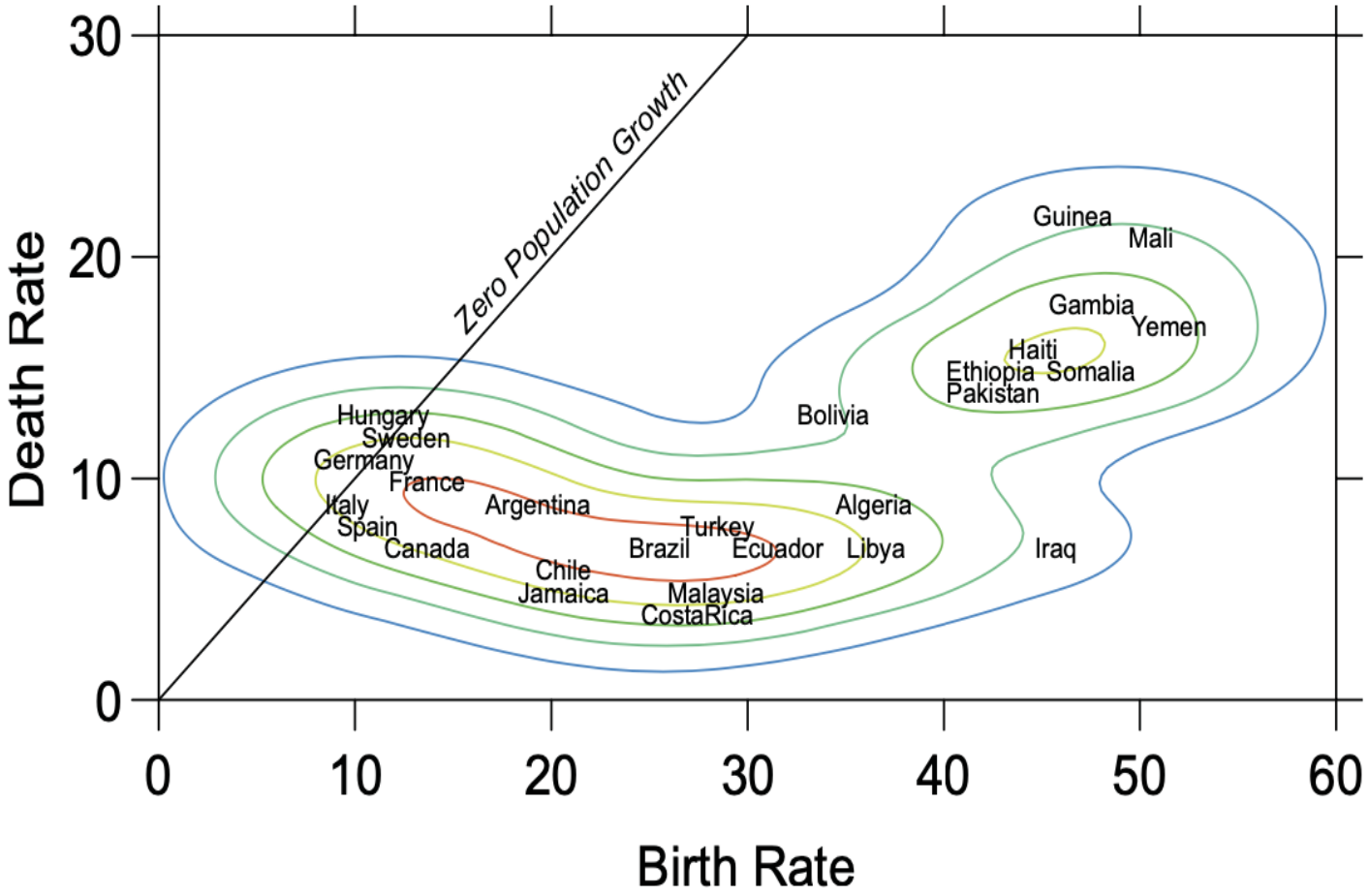


Figure 1.1 *Plot of death rates against birth rates for selected countries*

Grammar of graphics

- ELEMENT: point(position(birth*death), size(0), label(country))
- ELEMENT: contour(position(smooth.density.kernel.epanechnikov.joint(birth*death)), color.hue())
- GUIDE: form.line(position((0,0),(30,30)), label("Zero Population Growth"))
- GUIDE: axis(dim(1), label("Birth Rate"))
- GUIDE: axis(dim(2), label("Death Rate"))

(Leland Wilkinson)

Grammar of graphics

What Are some other elements and guides that you see on the graph ?

Grammar of graphics

ggplot2

ggplot2 is an R package for producing statistical, or data, graphics, but it is unlike most other graphics packages because it has a deep underlying grammar. This grammar, based on the Grammar of Graphics (Wilkinson, 2005), is composed of a set of independent components that can be composed in many different ways. This makes ggplot2 very powerful, because you are not limited to a set of pre-specified graphics, but you can create new graphics that are precisely tailored for your problem. This may sound overwhelming, but because there is a simple set of core principles and very few special cases, ggplot2 is also easy to learn (although it may take a little time to forget your preconceptions from other graphics tools).
(Hadley Wickham)

Grammar of graphics

- Learning the grammar will help you not only create graphics that you know about now, but will also help you to think about new graphics that would be even better.
- Without the grammar, there is no underlying theory and existing graphics packages are just a big collection of special cases.

Grammar of graphics

Layered grammar of graphics

ggplot2 is designed to work in a layered fashion, starting with a layer showing the raw data then adding layers of annotations and statistical summaries.

ggplot2

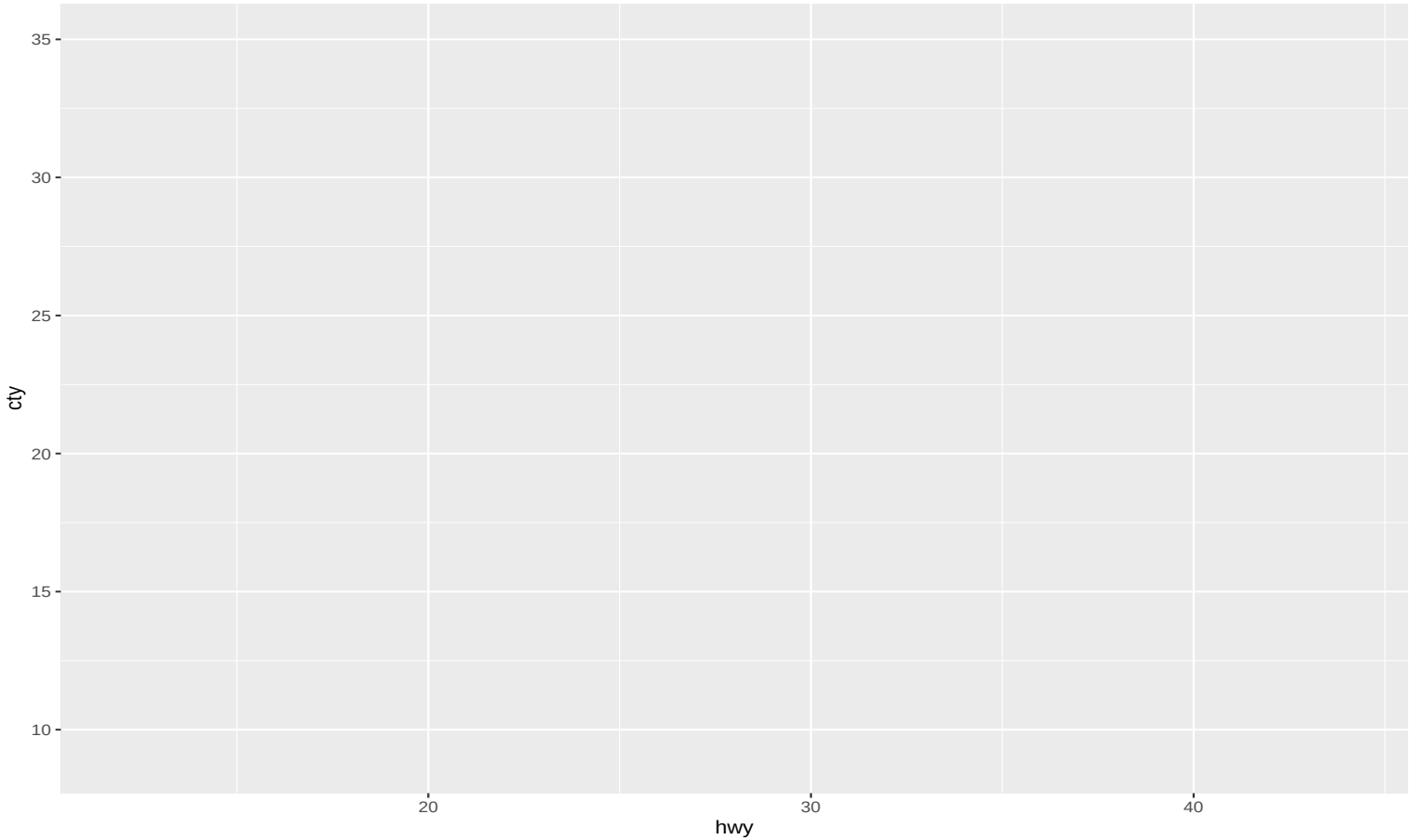
In ggplot the graph building process always starts with **ggplot()** layer
Get the data mpg from the library ggplot2

```
data(mpg)
```

ggplot2

The result is an empty space

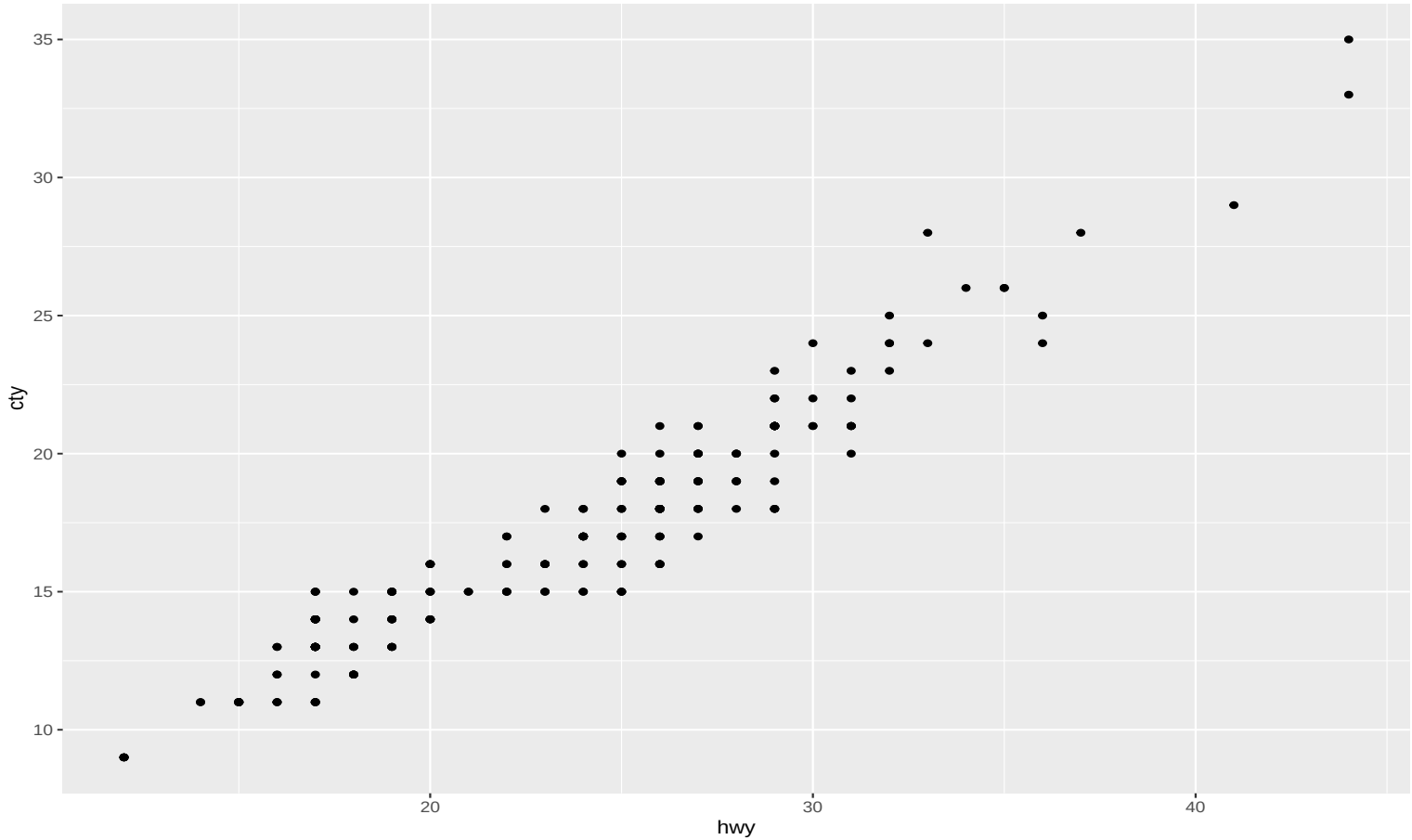
```
ggplot(data = mpg, aes(x = hwy, y = cty))
```



ggplot2

If you want to have a chart, add additional *layer*

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_point()
```



ggplot2

Elements of grammar of graphics

- The **data** that you want to visualise and a set of aesthetic mappings describing how variables in the data are mapped to aesthetic attributes that you can perceive.
- **Geometric objects**, geoms for short, represent what you actually see on the plot: points, lines, polygons, etc.
- **Statistical transformations**, stats for short, summarise data in many useful ways. For example, binning and counting observations to create a histogram, or summarising a 2d relationship with a linear model. Stats are optional, but very useful.
- The **scales** map values in the data space to values in an aesthetic space, whether it be colour, or size, or shape. Scales draw a legend or axes, which provide an inverse mapping to make it possible to read the original data values from the graph.

ggplot2 (continued)

- A **coordinate system**, coord for short, describes how data coordinates are mapped to the plane of the graphic. It also provides axes and gridlines to make it possible to read the graph. We normally use a Cartesian coordinate system, but a number of others are available, including polar coordinates and map projections.
- A **faceting** specification describes how to break up the data into subsets and how to display those subsets as small multiples. This is also known as conditioning or latticing/trellising.

ggplot2

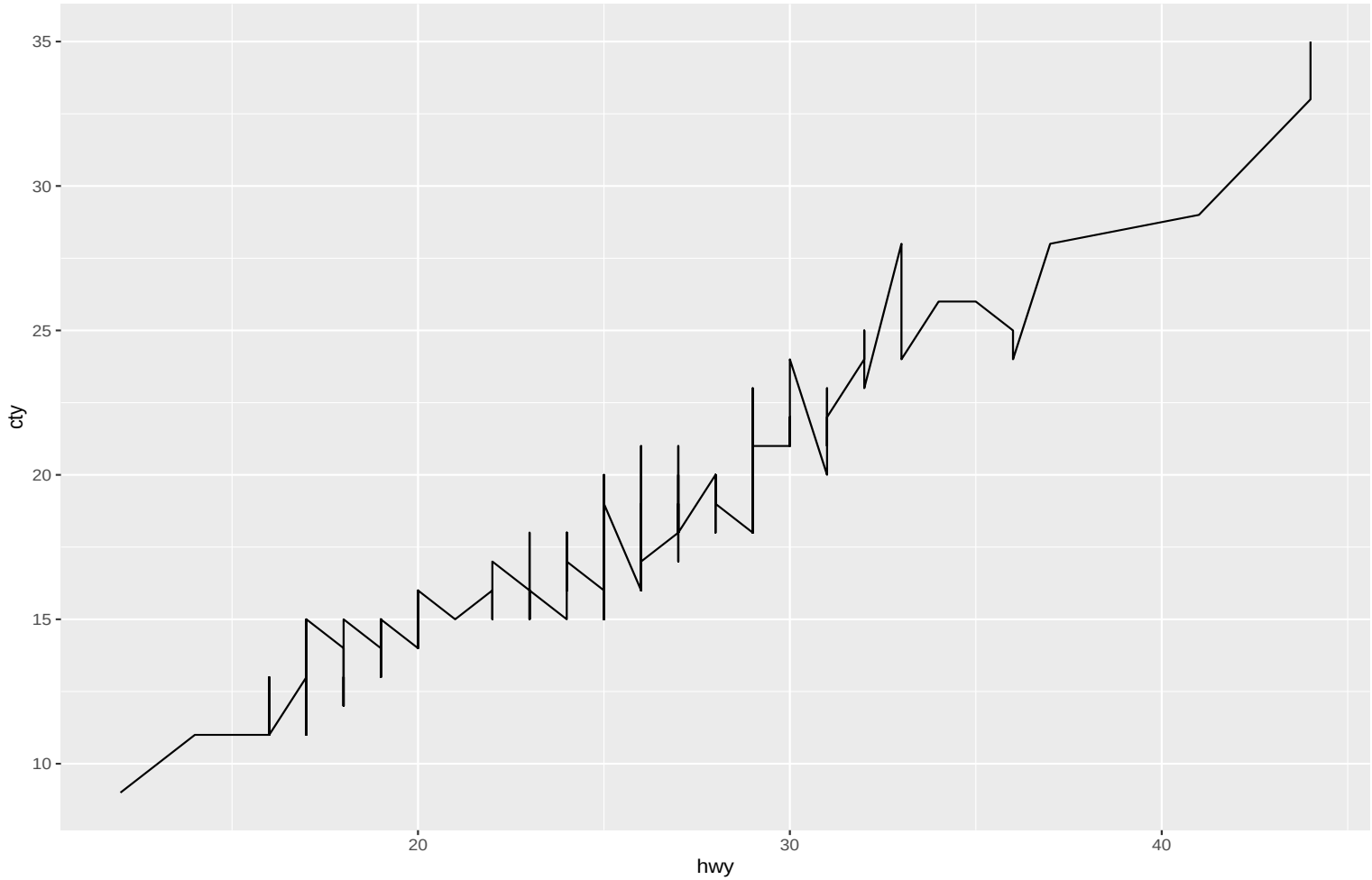
ggplot2 uses shortcuts for the elements

- geom_XXX - for geometric objecs
- stat_XXX - for statistical transformations
- scale_ - for scales
- coord_XXX for coordinate systems
- facet_ - for faceting

ggplot2

If you want to see what are the geometric objects available as a layers for ggplot, just type geom_ . *line chart*

```
ggplot(data = mpg, aes(x = hwy, y = cty)) + geom_line()
```



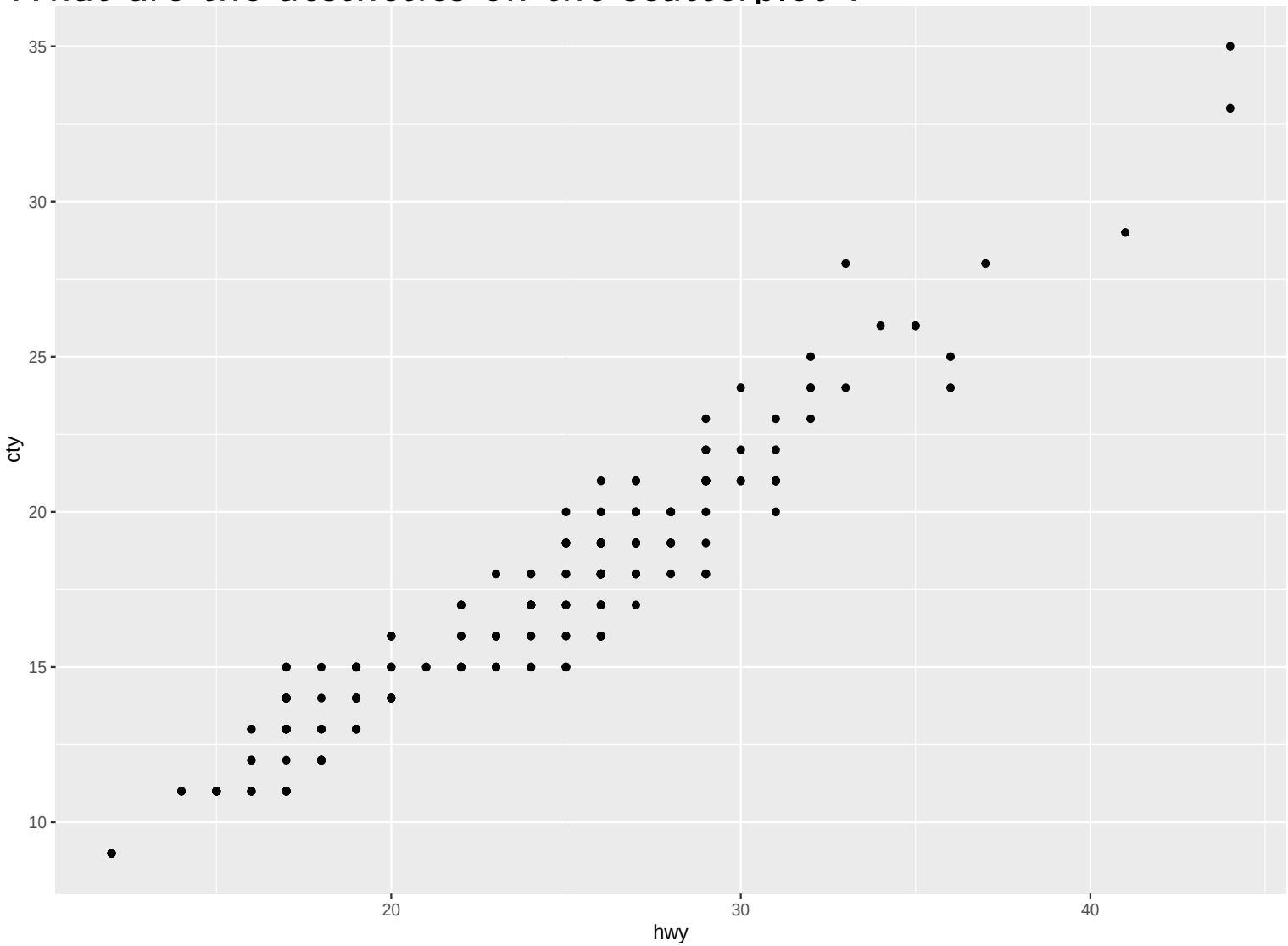
ggplot2

Aesthetics

- Aesthetics, in the original Greek sense, offers principles for relating sensory attributes to abstractions.
- In ggplot and data visualization in general, aesthetics map data to the plot. In other words, whatever can be perceived on the graph, is an aesthetics.

ggplot2

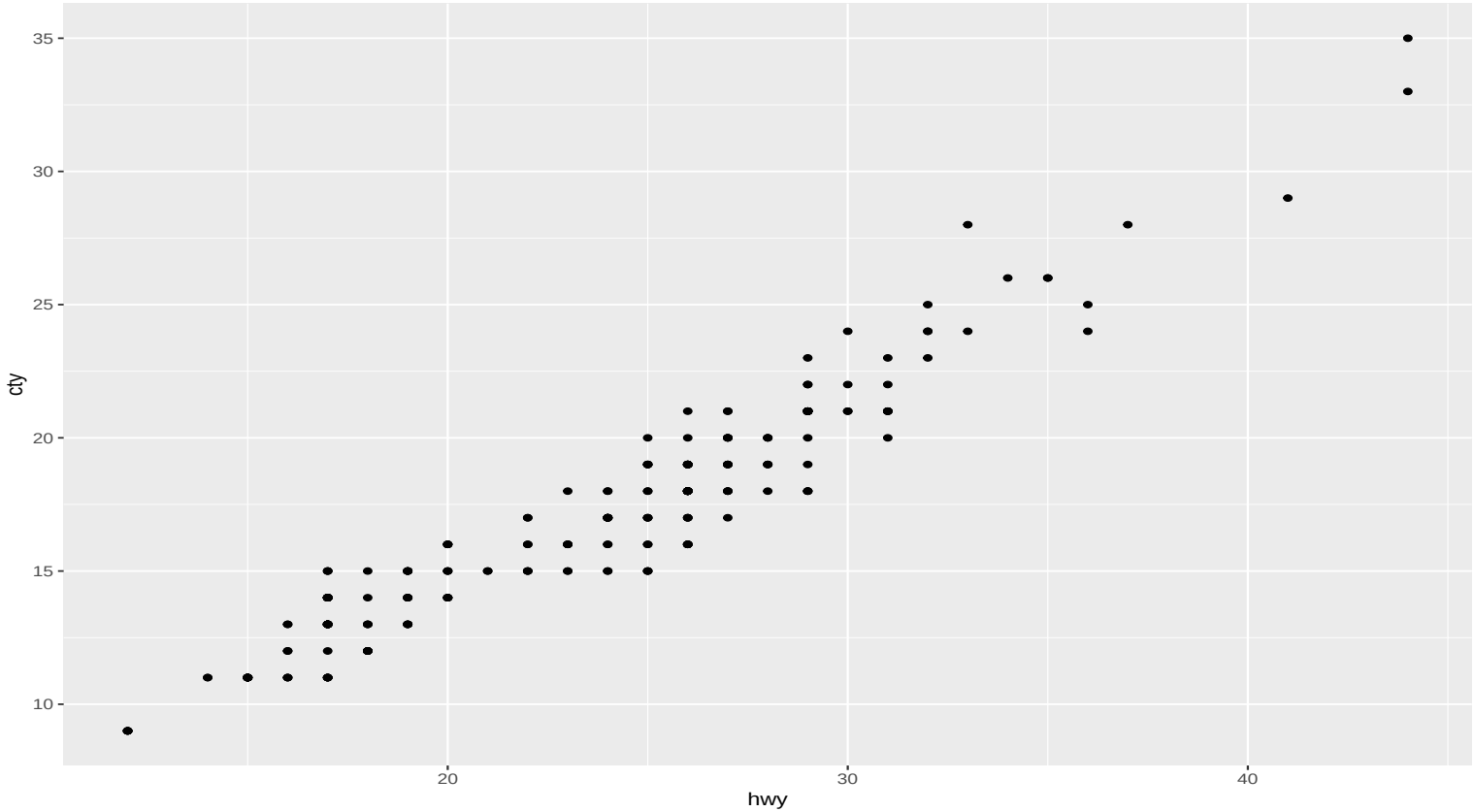
What are the aesthetics on the scatterplot ?



ggplot2

If you define data and aesthetics in ggplot(), all other layers are going to inherit them
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) + geom_point()
However you can define data and aesthetics in the layer as well

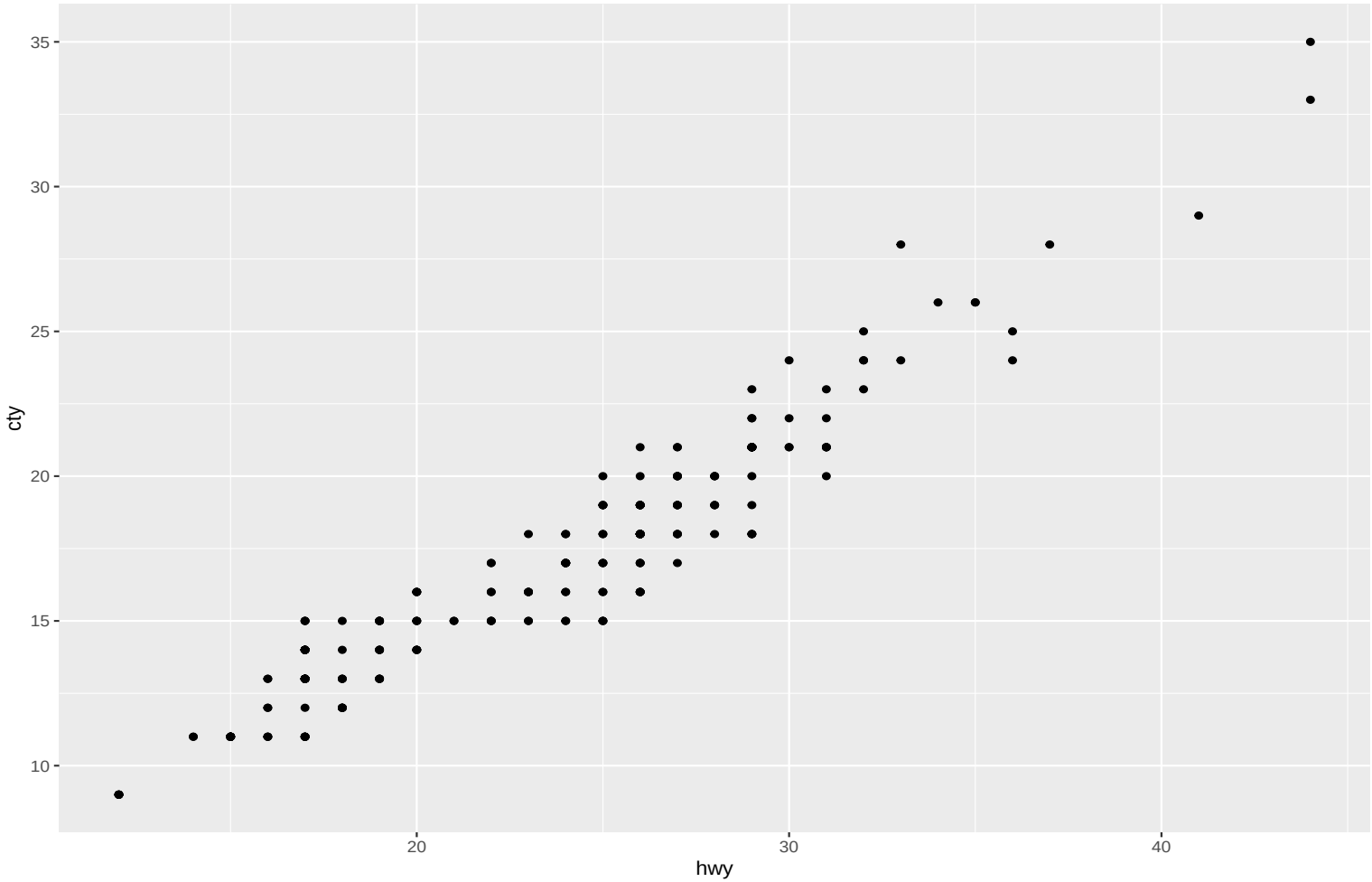
```
ggplot() + geom_point(data = mpg, mapping = aes(x = hwy, y = cty))
```



ggplot2

The difference becomes apparent when you add another layer

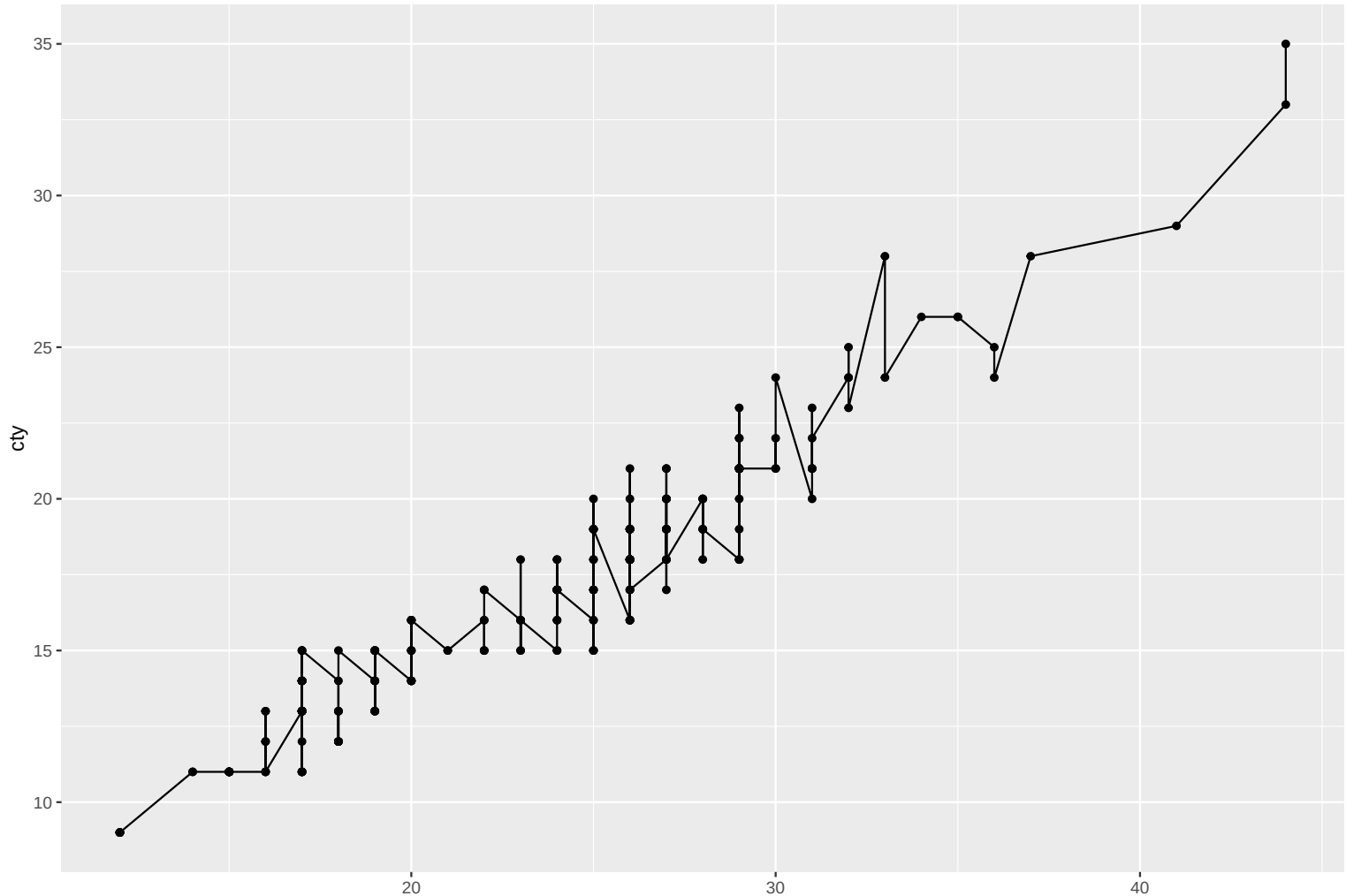
```
ggplot() + geom_point(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_line()
```



ggplot2

Now you need to specify data and aesthetics in each layer

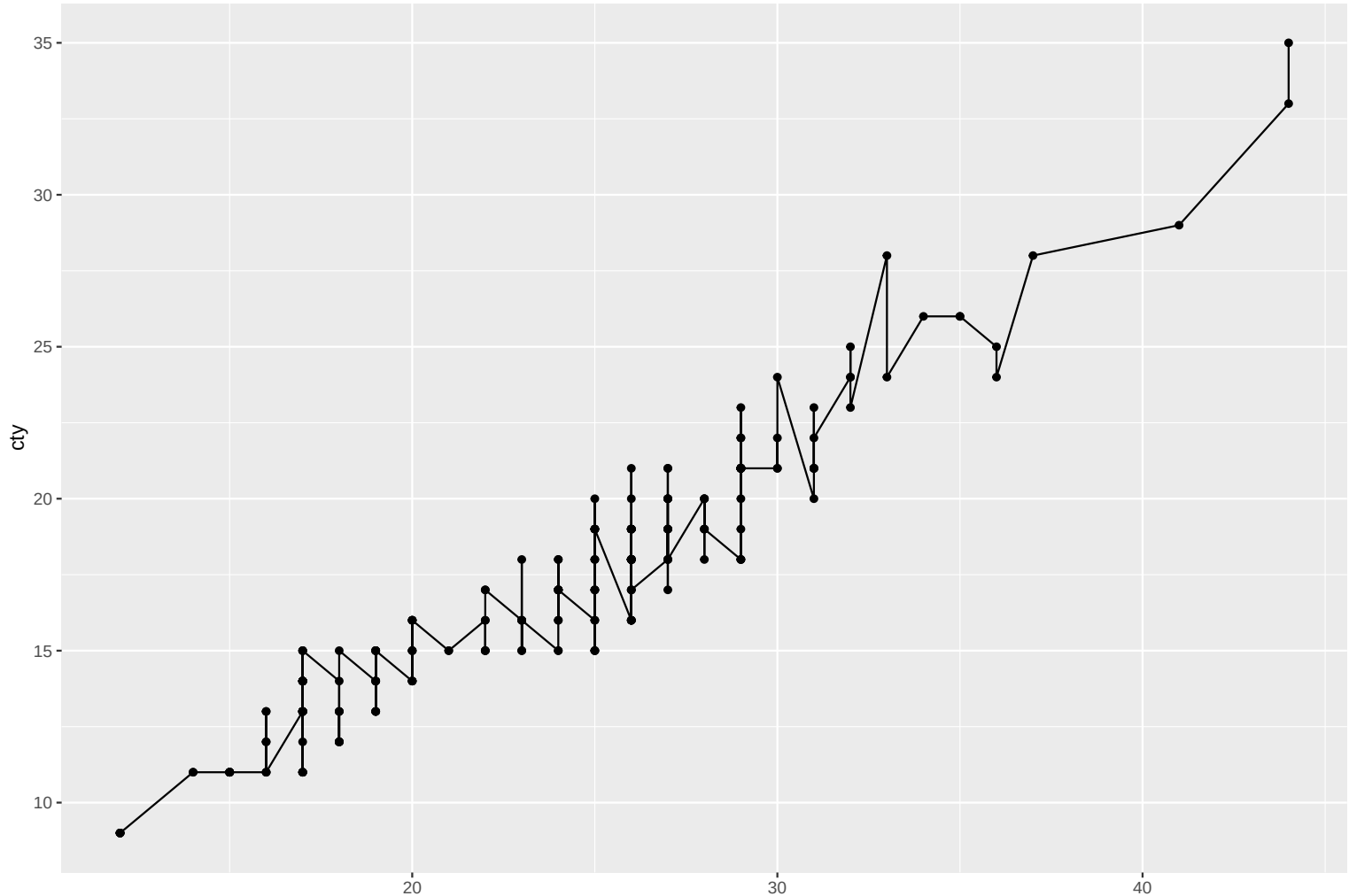
```
ggplot() + geom_point(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_line(data = mpg, mapping = aes(x = hwy, y = cty))
```



ggplot2

Or

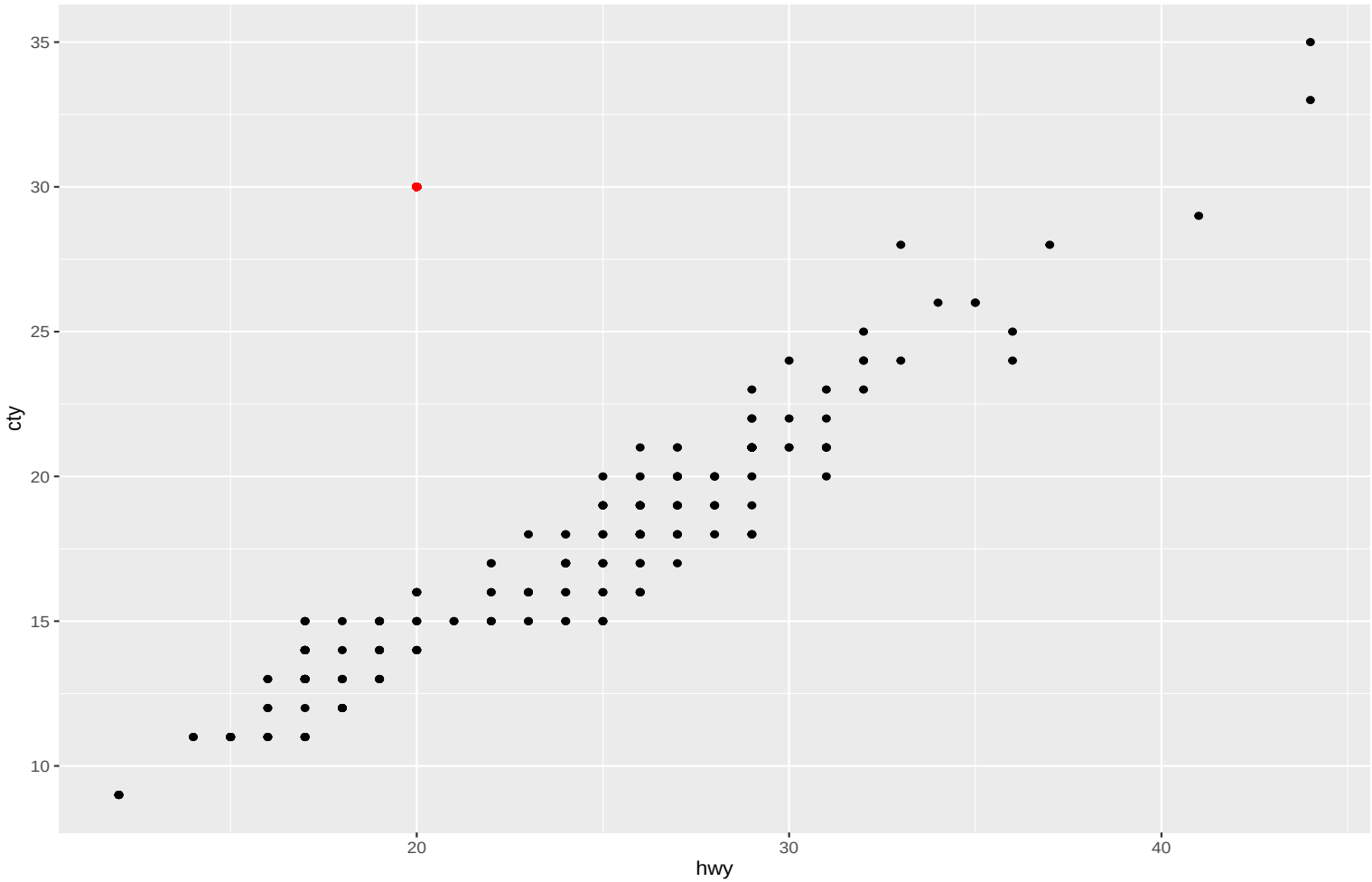
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) + geom_point() +
  geom_line()
```



ggplot2

Add additional point with a new layer

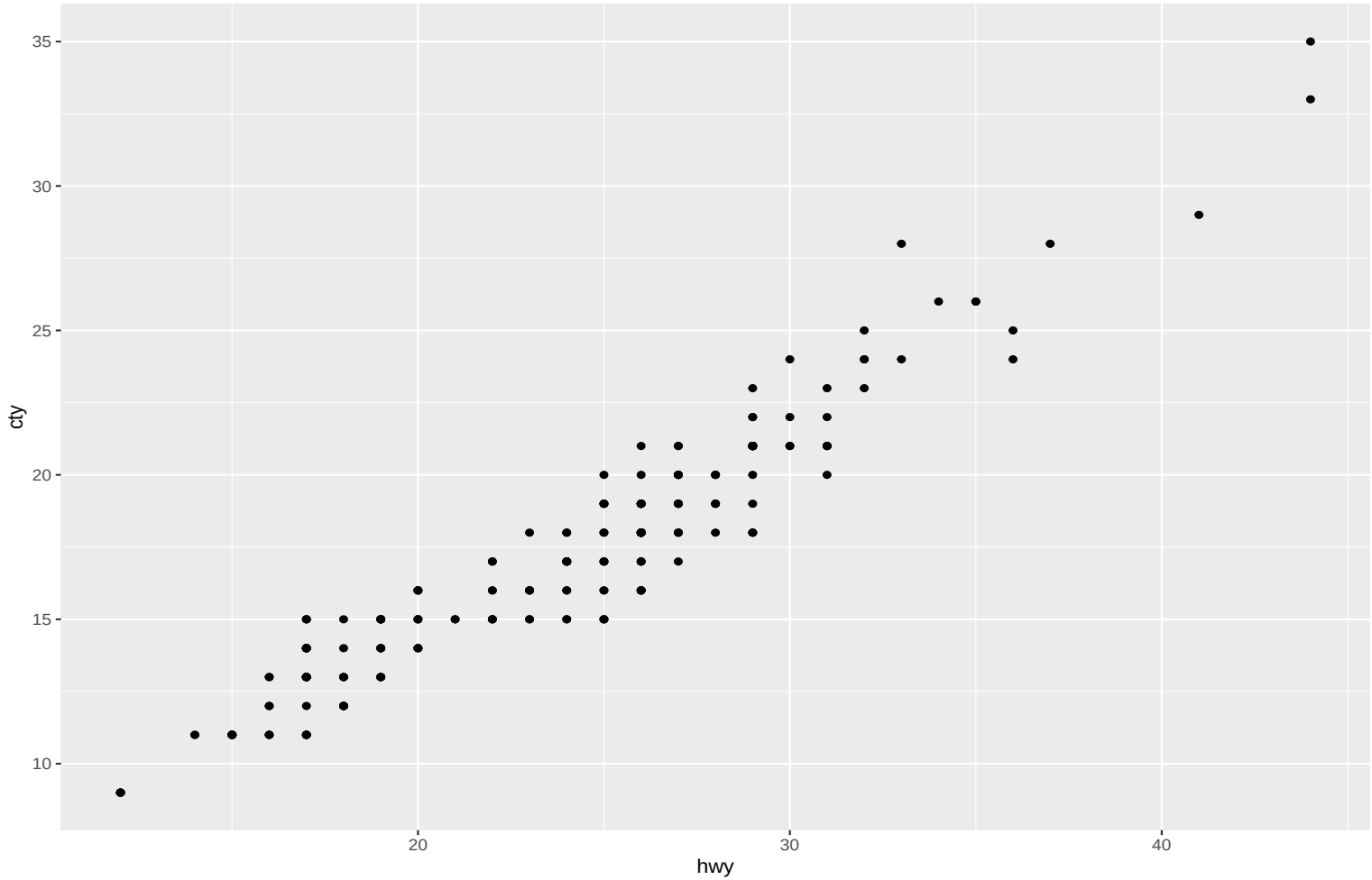
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_point() + geom_point(mapping = aes(x = 20, y = 30), color = "red")
```



ggplot2

Actually you can save ggplot output and then add layers to it

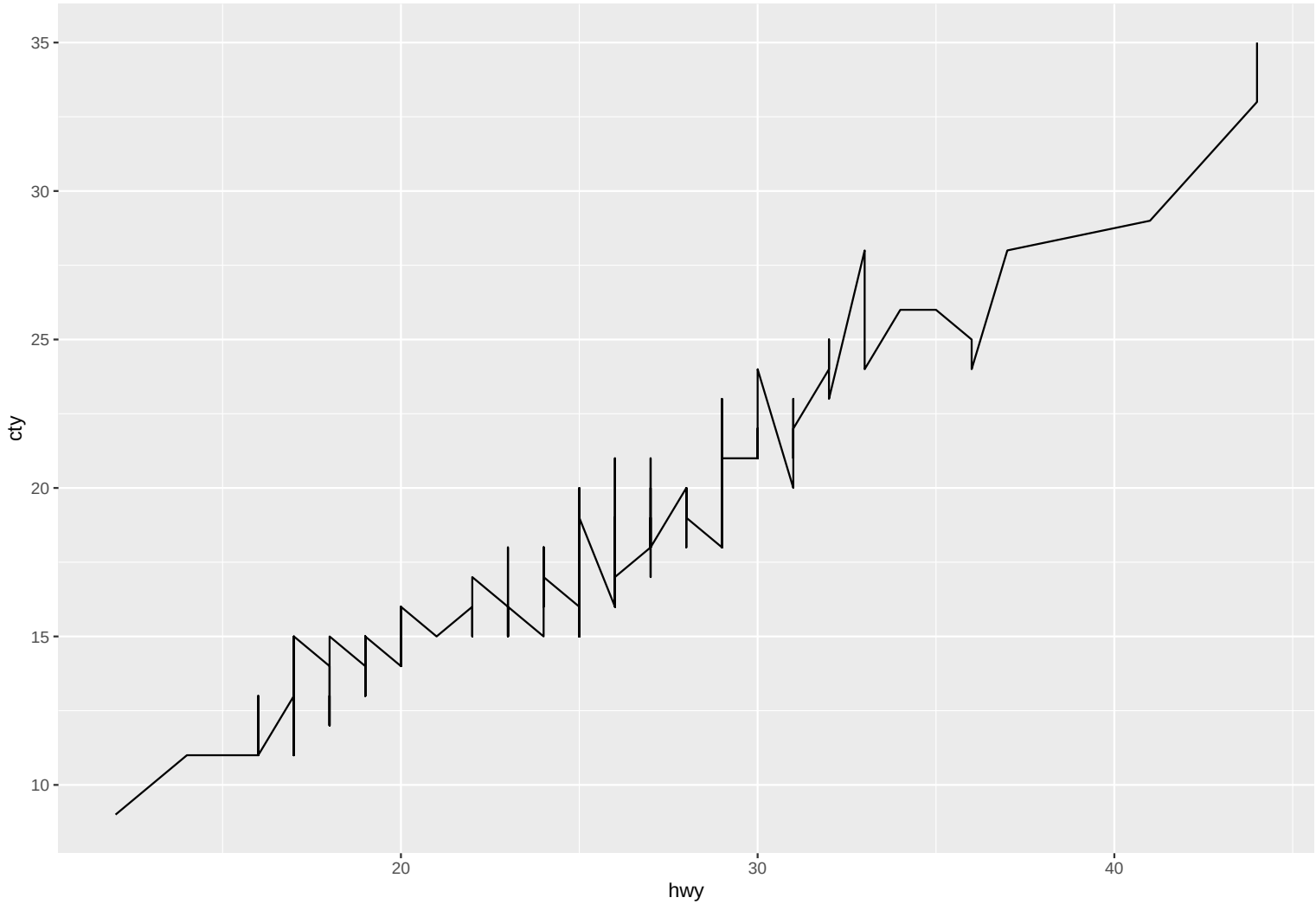
```
p <- ggplot(data = mpg, mapping = aes(x = hwy, y = cty))
p + geom_point()
```



ggplot2

Line chart

```
p + geom_line()
```



ggplot2

If you are planning to break the line, leave + at the end of the line
Like this

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_point() +
  geom_line()
```

Not Like this

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty))
  + geom_point() +
  geom_line()
```

ggplot2

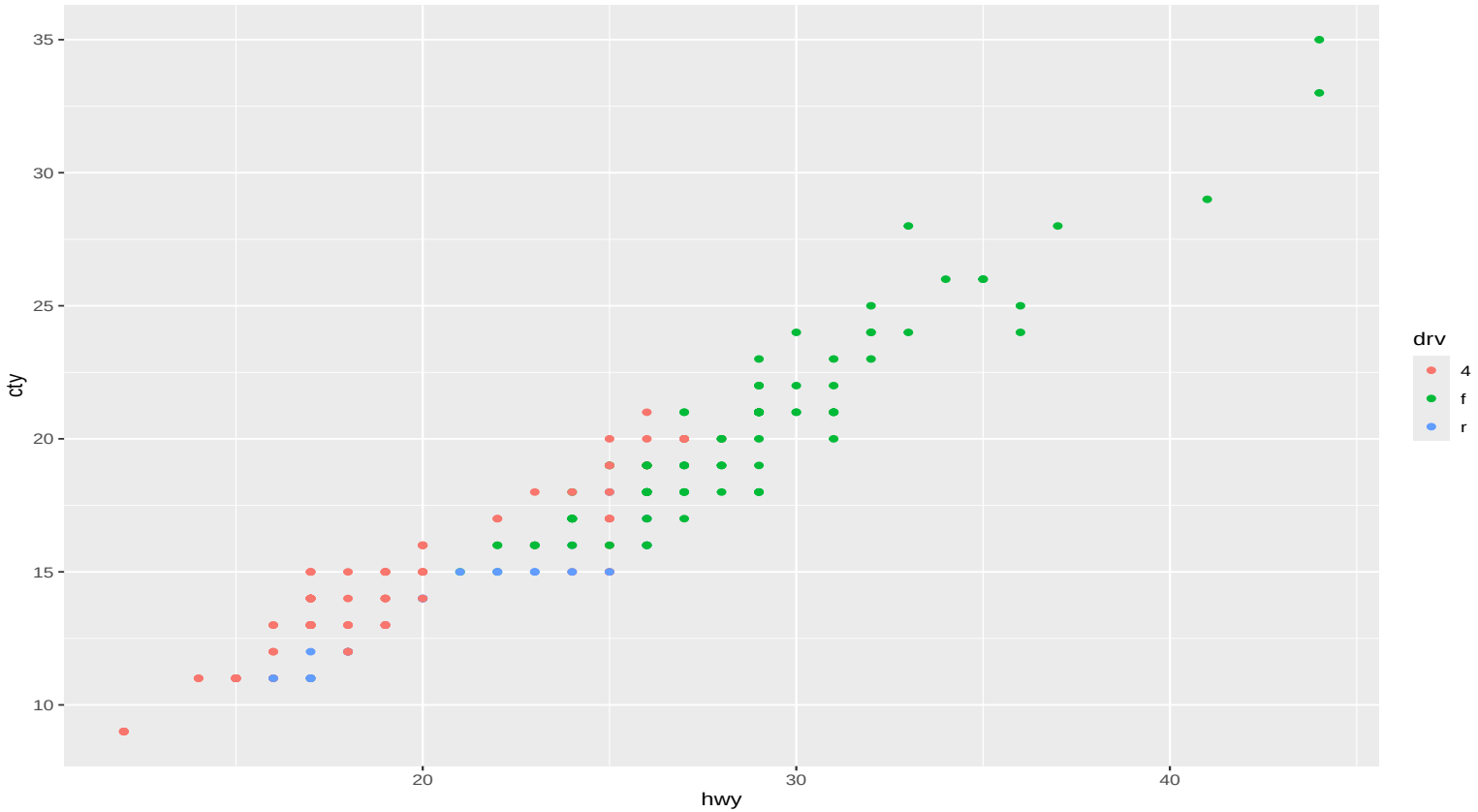
aesthetics
For the scatterplot:

- x coordinate
- y coordinate
- size
- shape
- color

ggplot2

The aesthetics can be either a constant, or a variable from the data.
variable

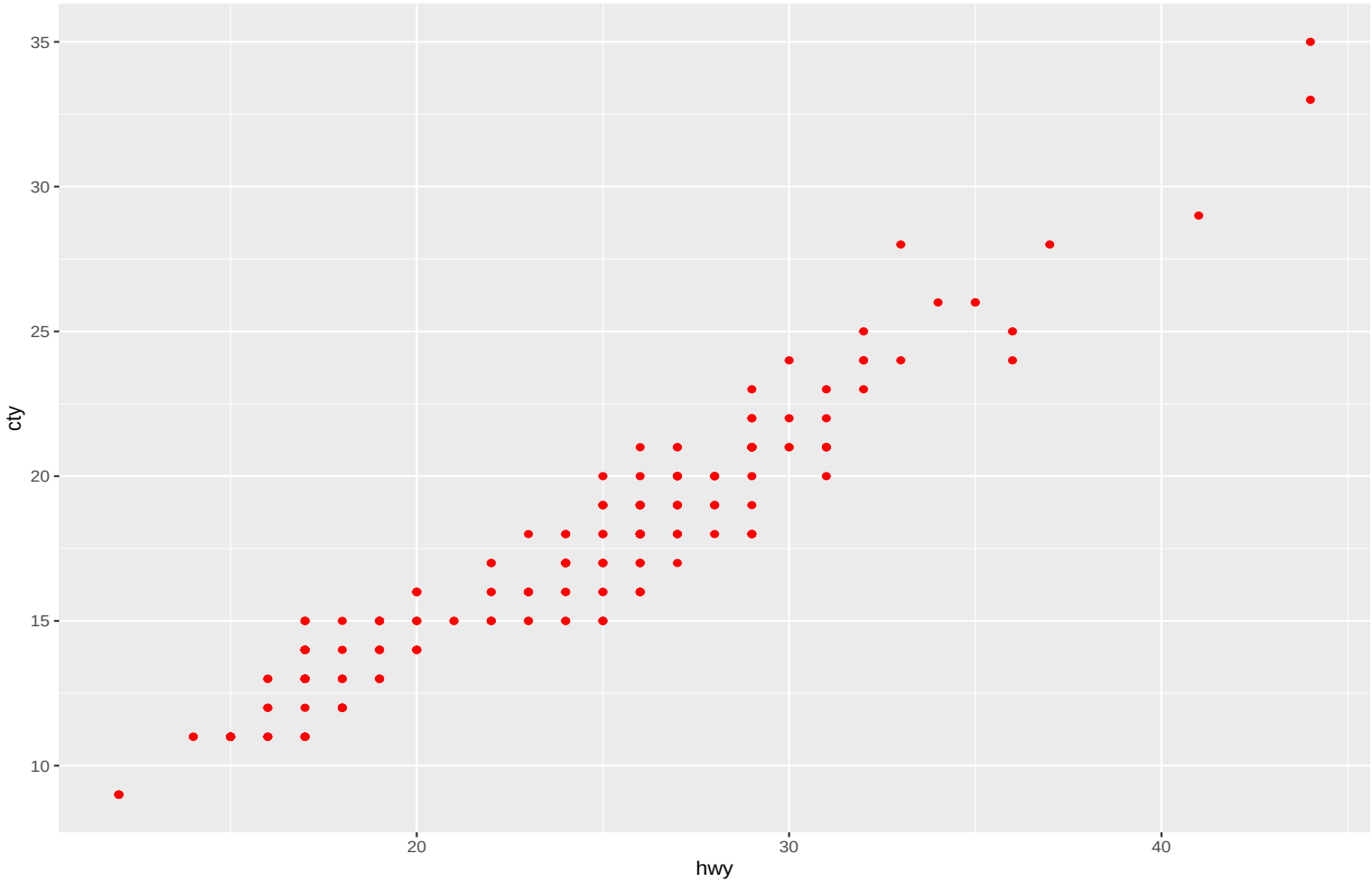
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, color = drv)) +
  geom_point()
```



ggplot2

If you want aesthetics as a constant, don't include it in aesthetics

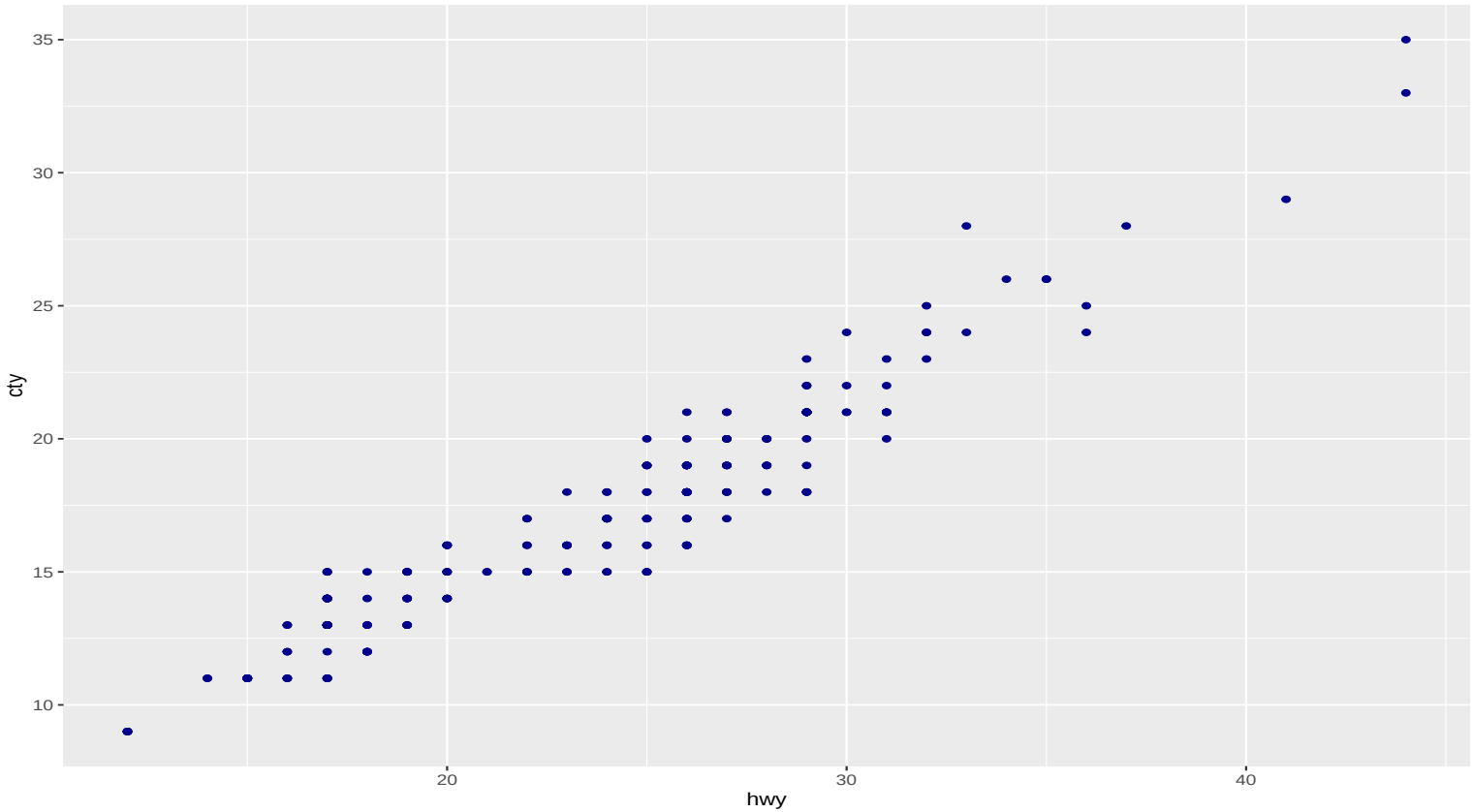
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_point(color = 'red')
```



ggplot2

Lets see the difference between *mapping* the aesthetics and *setting* it.
Setting

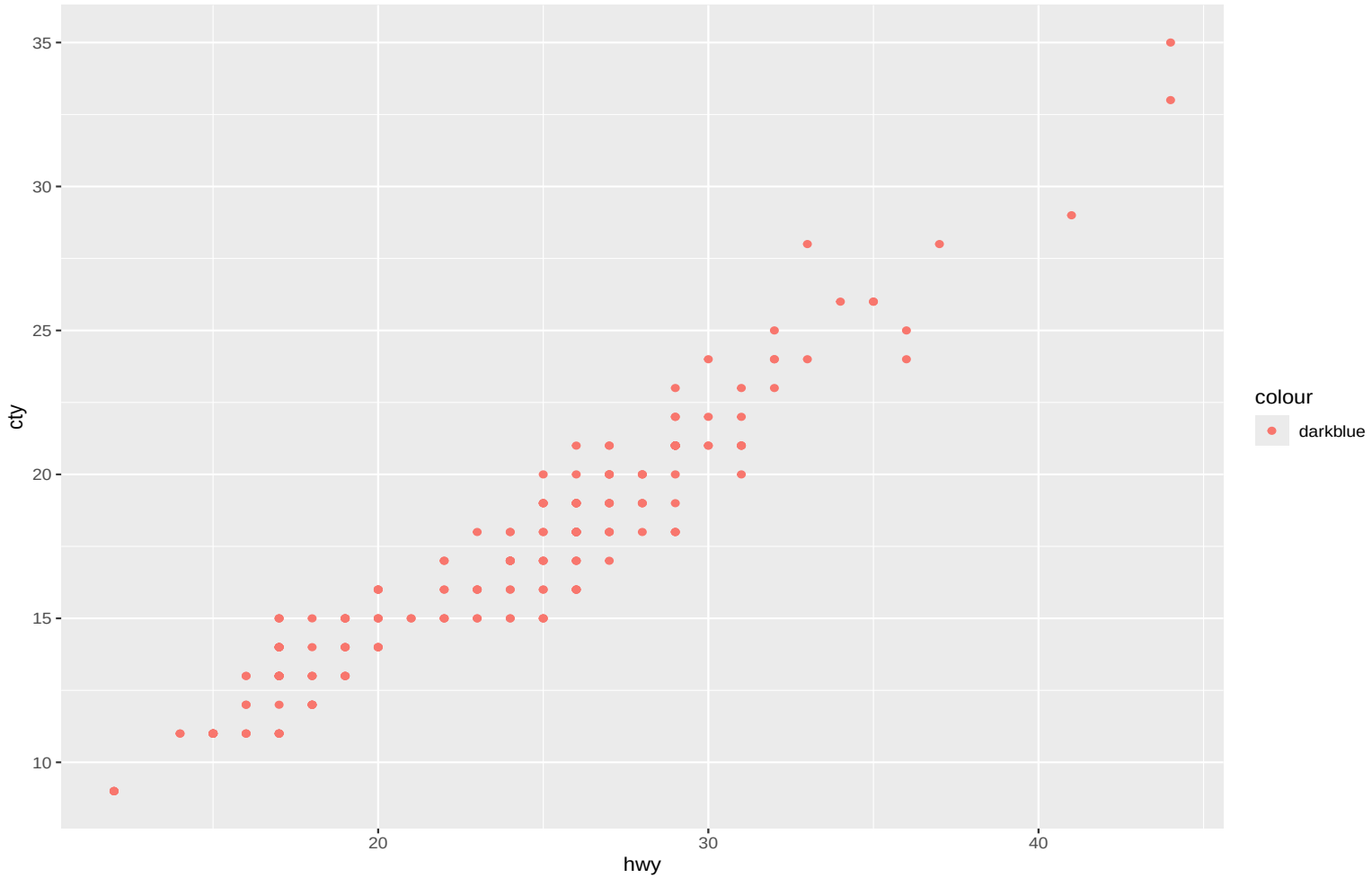
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty)) +
  geom_point(color = 'darkblue')
```



ggplot2

Mapping

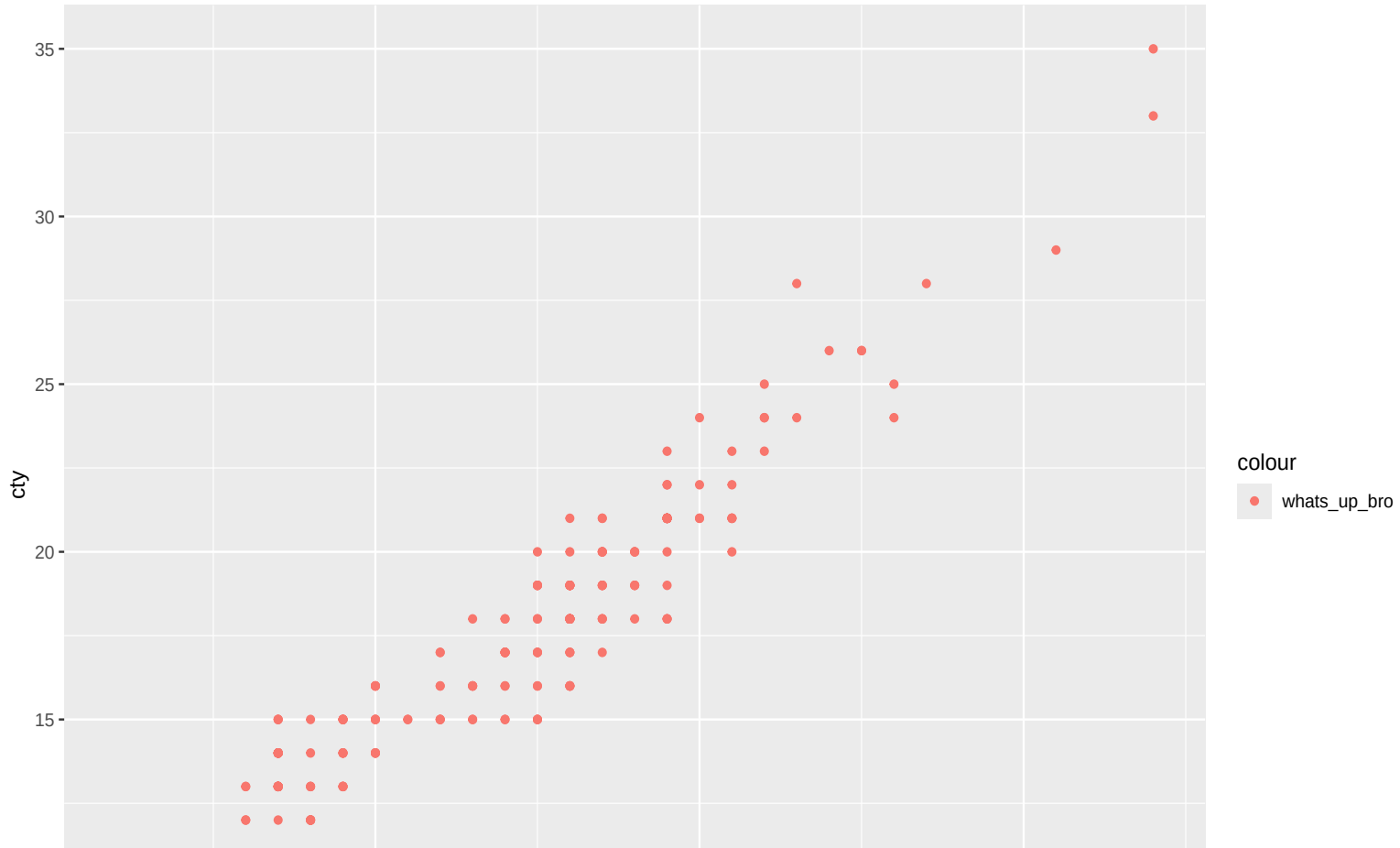
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, color = 'darkblue')) +
  geom_point()
```



ggplot2

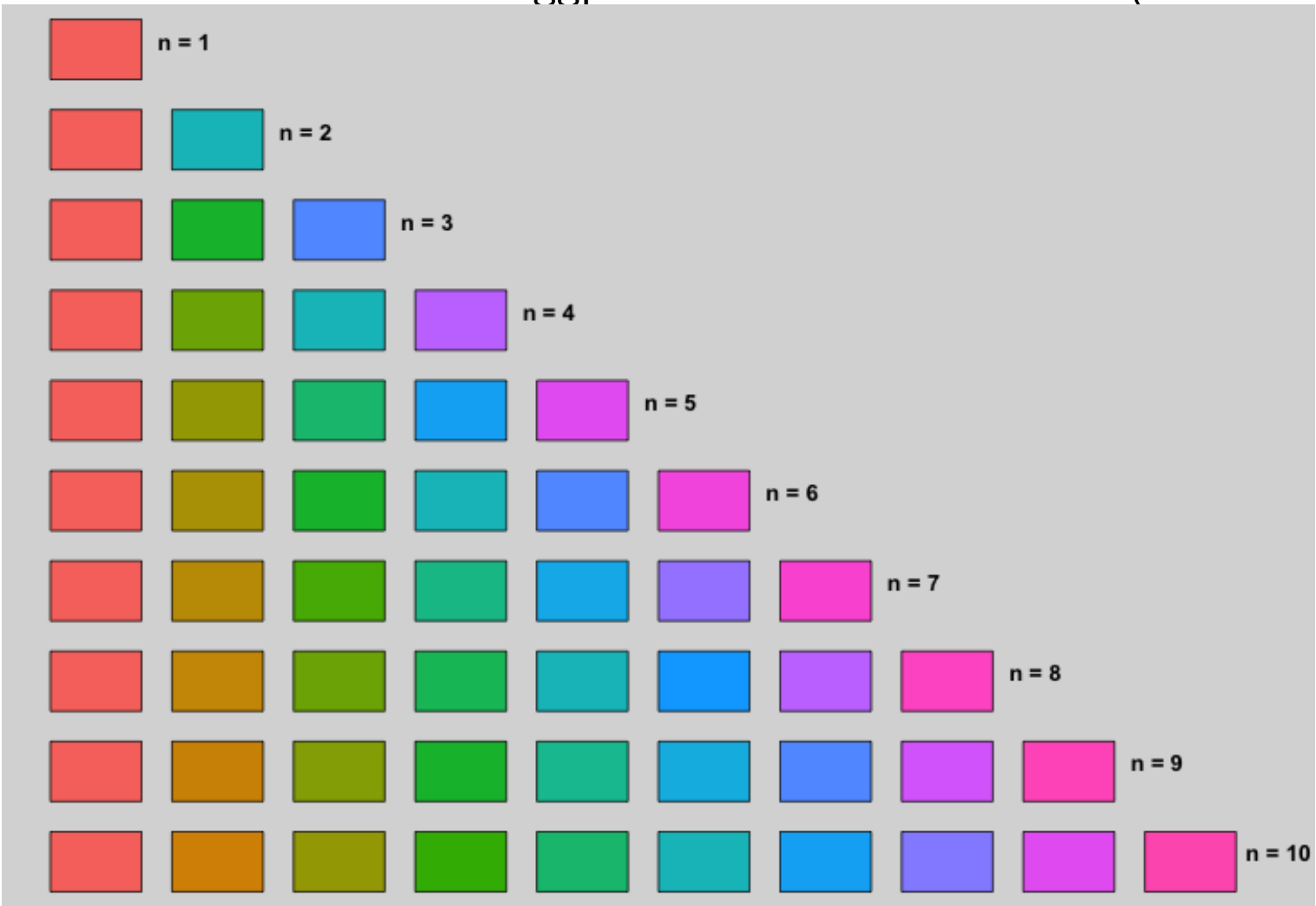
Basically if you map the aesthetics to a constant, it does not matter what you write there, the result is going to be the same

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, color = 'whats_up_bro'))
  geom_point()
```



ggplot2

The colors are taken from ggplot's standard color wheel (more about it later)



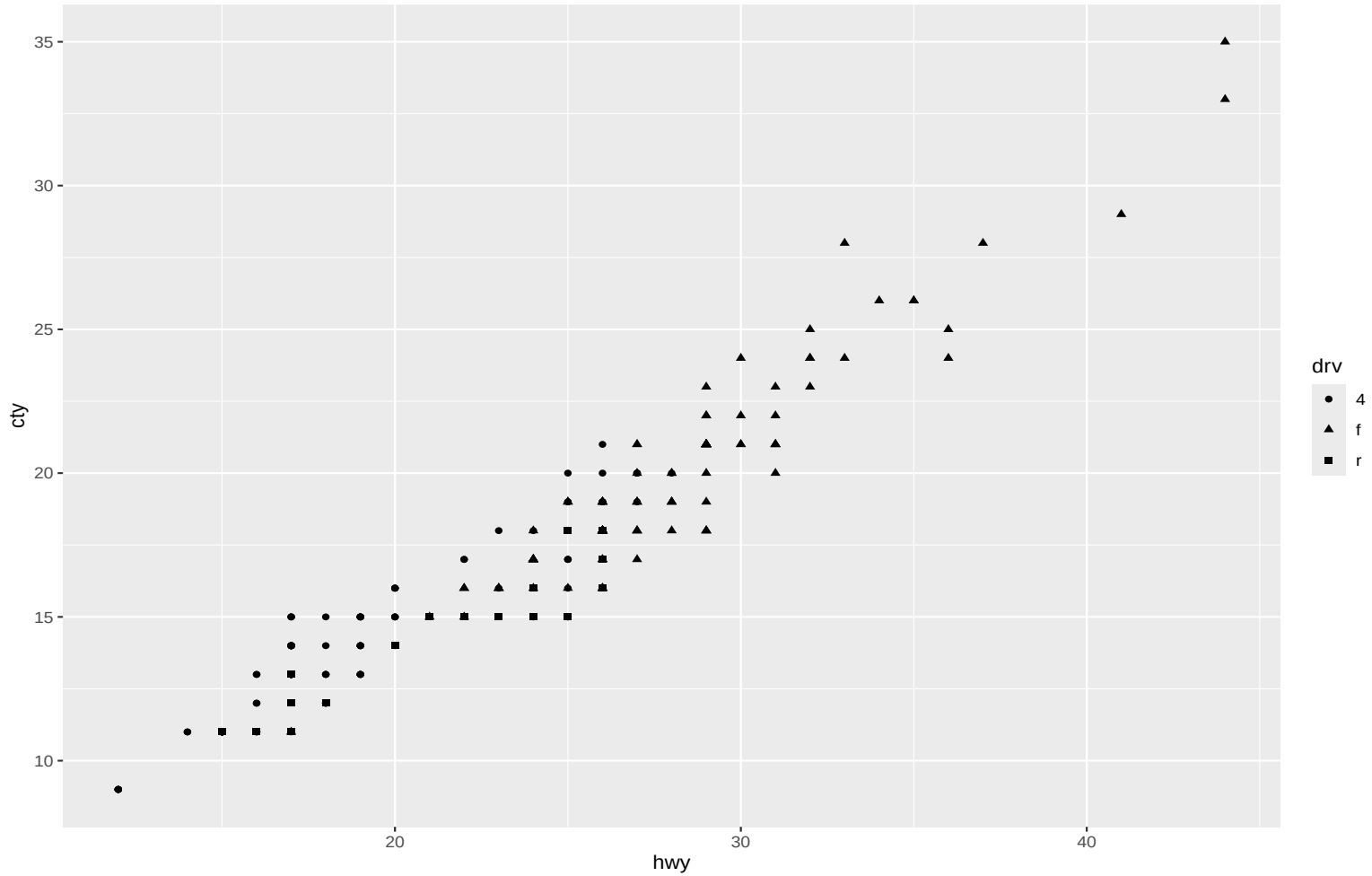
ggplot2

Hadley Wickham clarifies here
p + geom_point(aes(colour = "darkblue"))
This maps (not sets) the colour to the value "darkblue". This effectively creates a new variable containing only the value "darkblue" and then maps colour to that new variable. Because this value is discrete, the default colour scale uses evenly spaced colours on the colour wheel, and since there is only one value this colour is pinkish (n=1 from color scale of ggplot)

ggplot2

Other aesthetics: shape

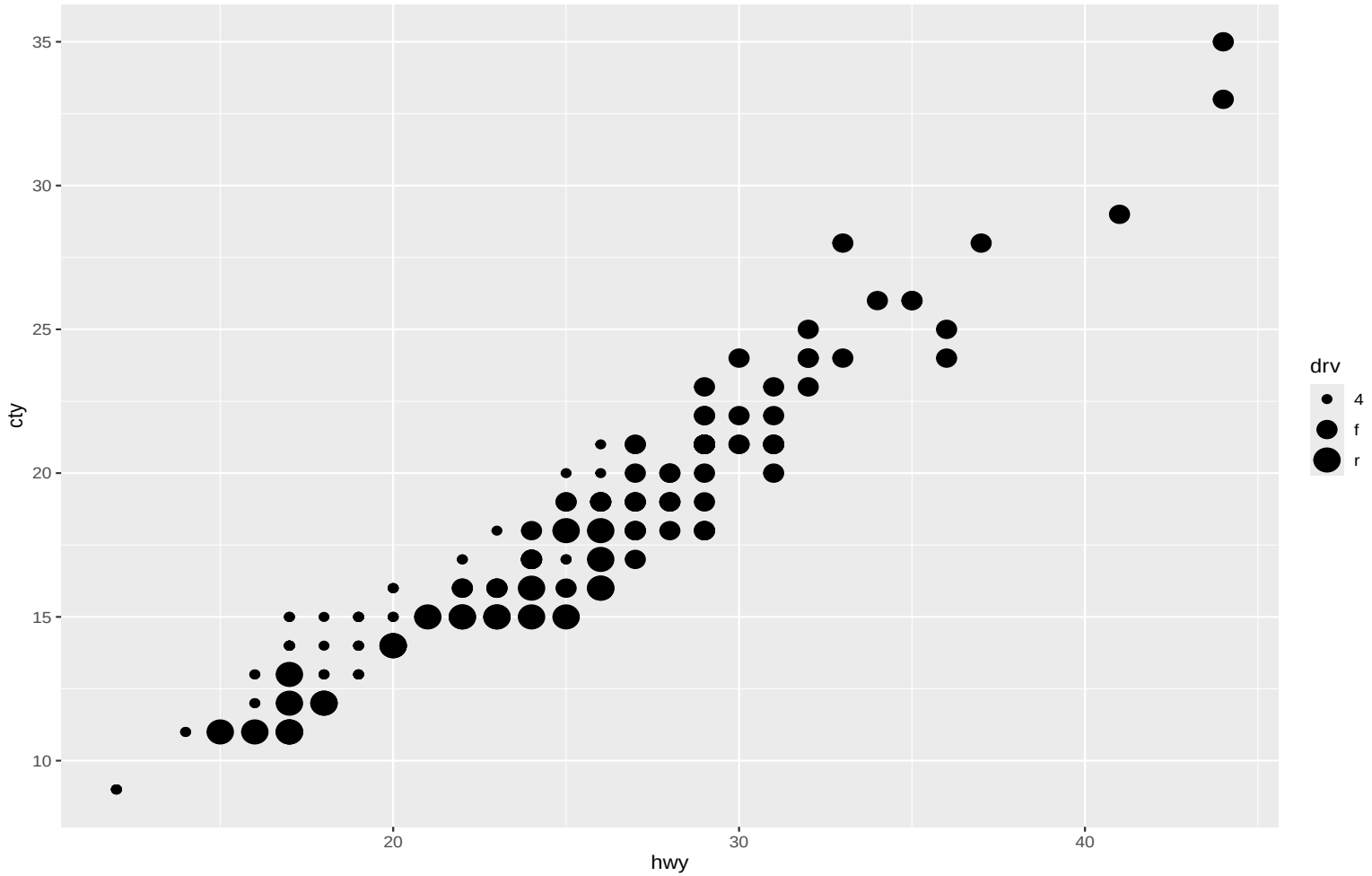
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, shape = drv)) +
  geom_point()
```



ggplot2

Size aesthetics

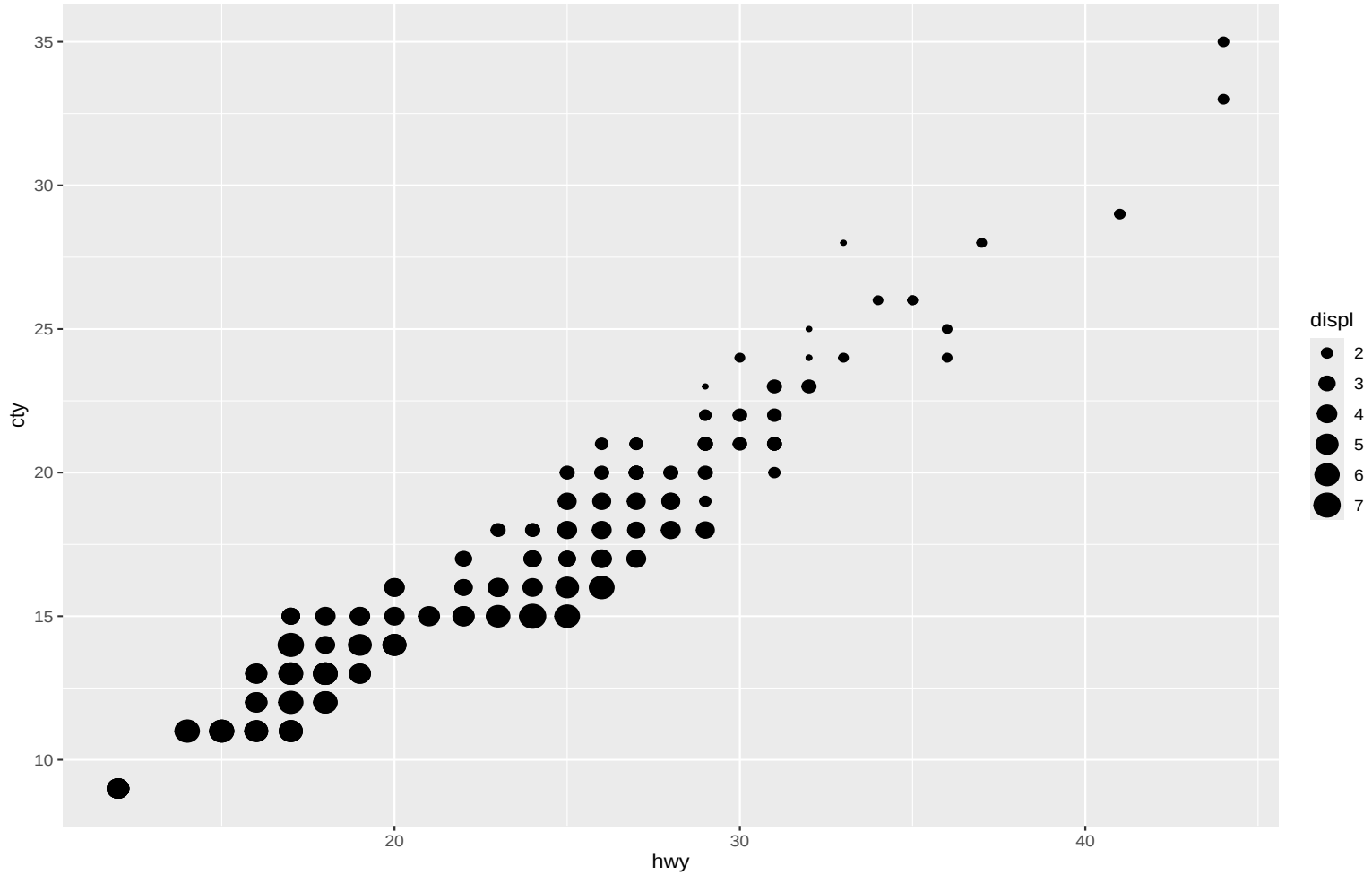
```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, size = drv)) +
  geom_point()
```



ggplot2

Size as a continuous variable

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, size = displ)) +
  geom_point()
```



ggplot2

Shape as a continuous variable will raise an error

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, shape = displ)) +
  geom_point()
```

ggplot2

ggplot has a wide range of geometric object

- geom_bar(): for bar chart
- geom_histogram(): for histogram
- geom_boxplot(): for boxplots etc

ggplot2

Different geometric objects ask for different aesthetics: geom_point() understands the following aesthetics

- x
- y
- alpha
- colour
- fill
- group
- shape
- size
- stroke

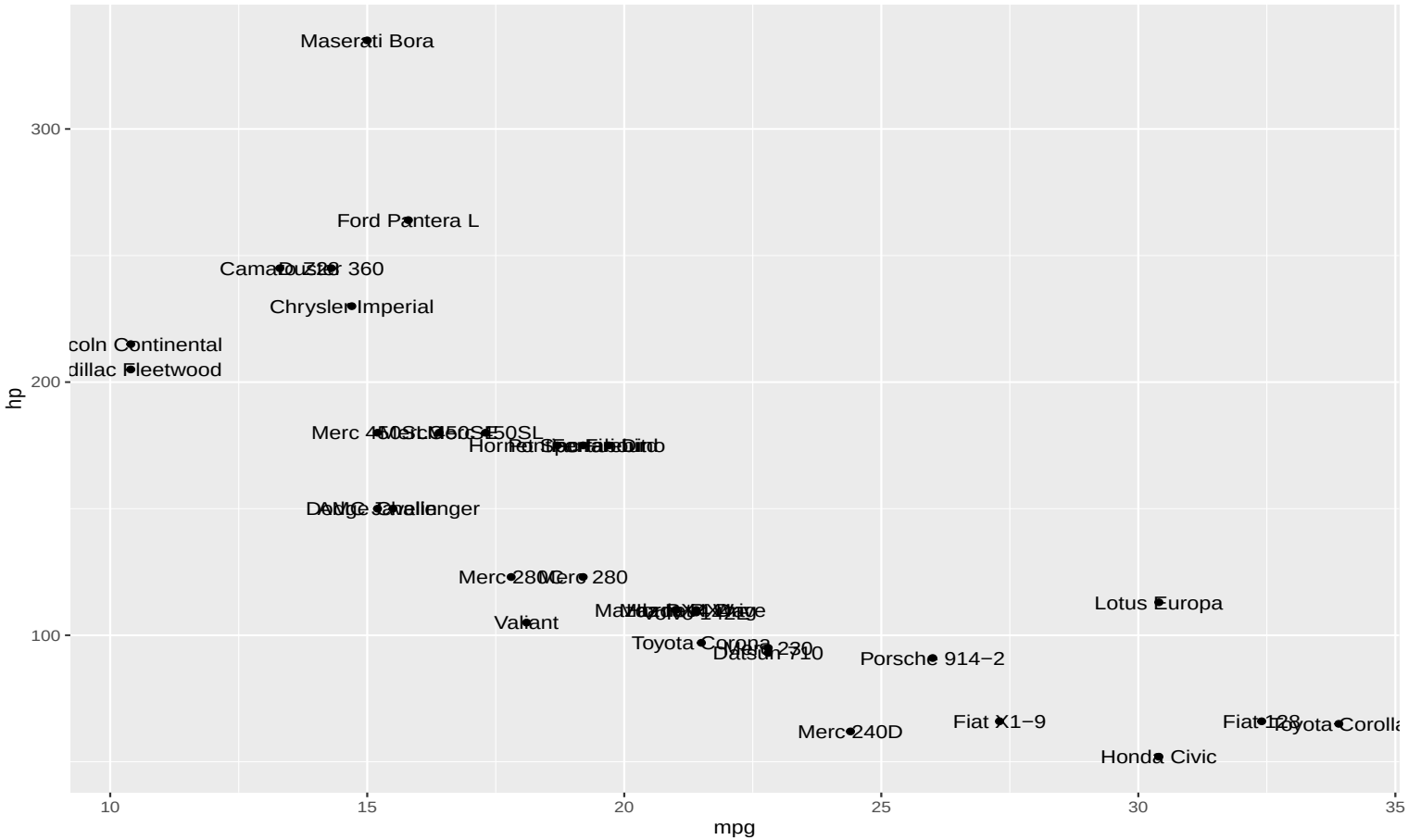
ggplot2

geom_text/geom_label is used to label plots. geom_text() understands the following aesthetics

- x
- y
- label
- alpha
- angle
- colour
- family
- fontface
- group
- hjust
- vjust
- lineheight
- size

ggplot2

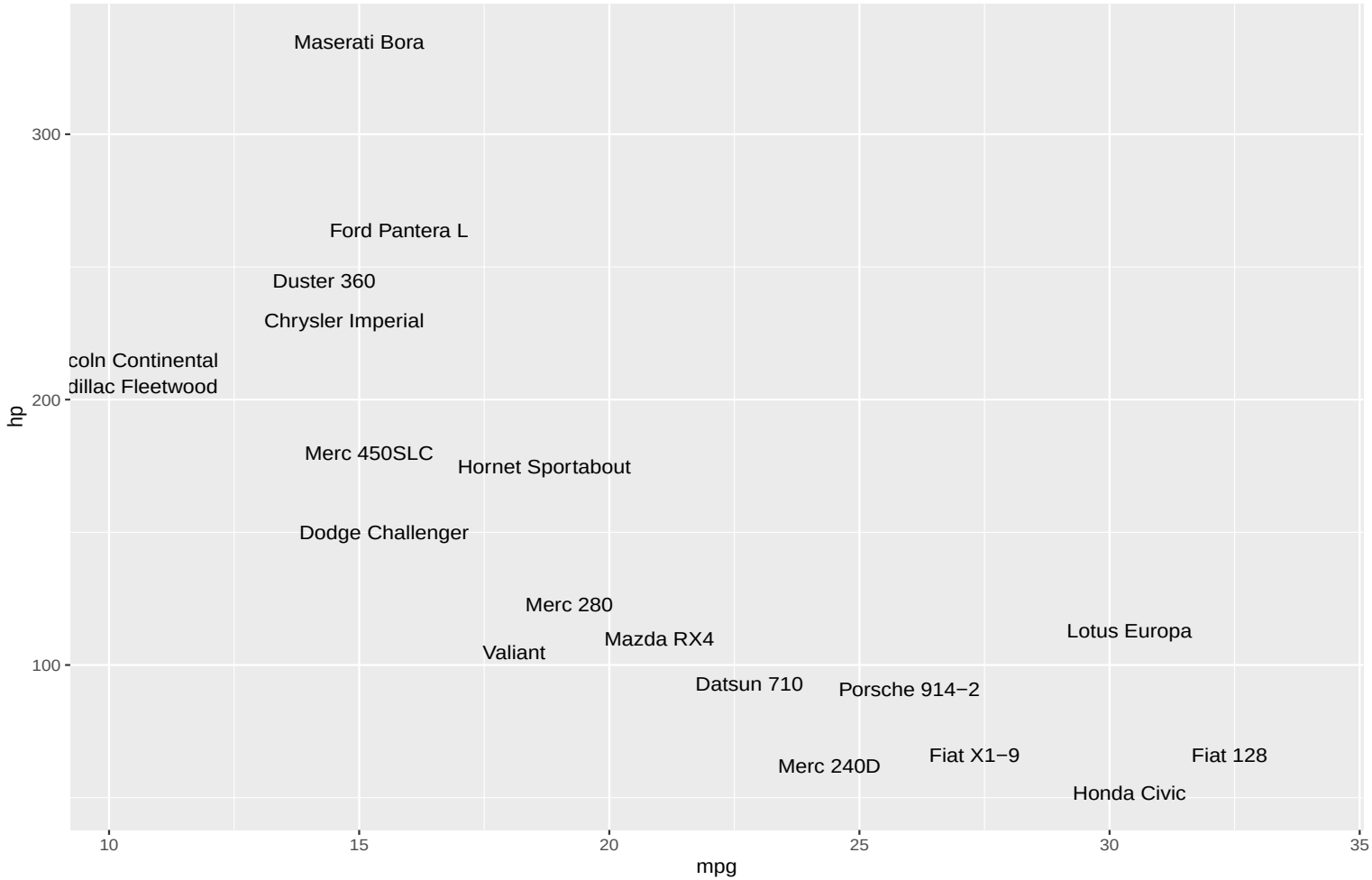
```
mtcars$car <- rownames(mtcars)
ggplot(mtcars, aes(x = mpg, y = hp, label = car)) +
  geom_point() + geom_text()
```



ggplot2

Or just use geom_text()

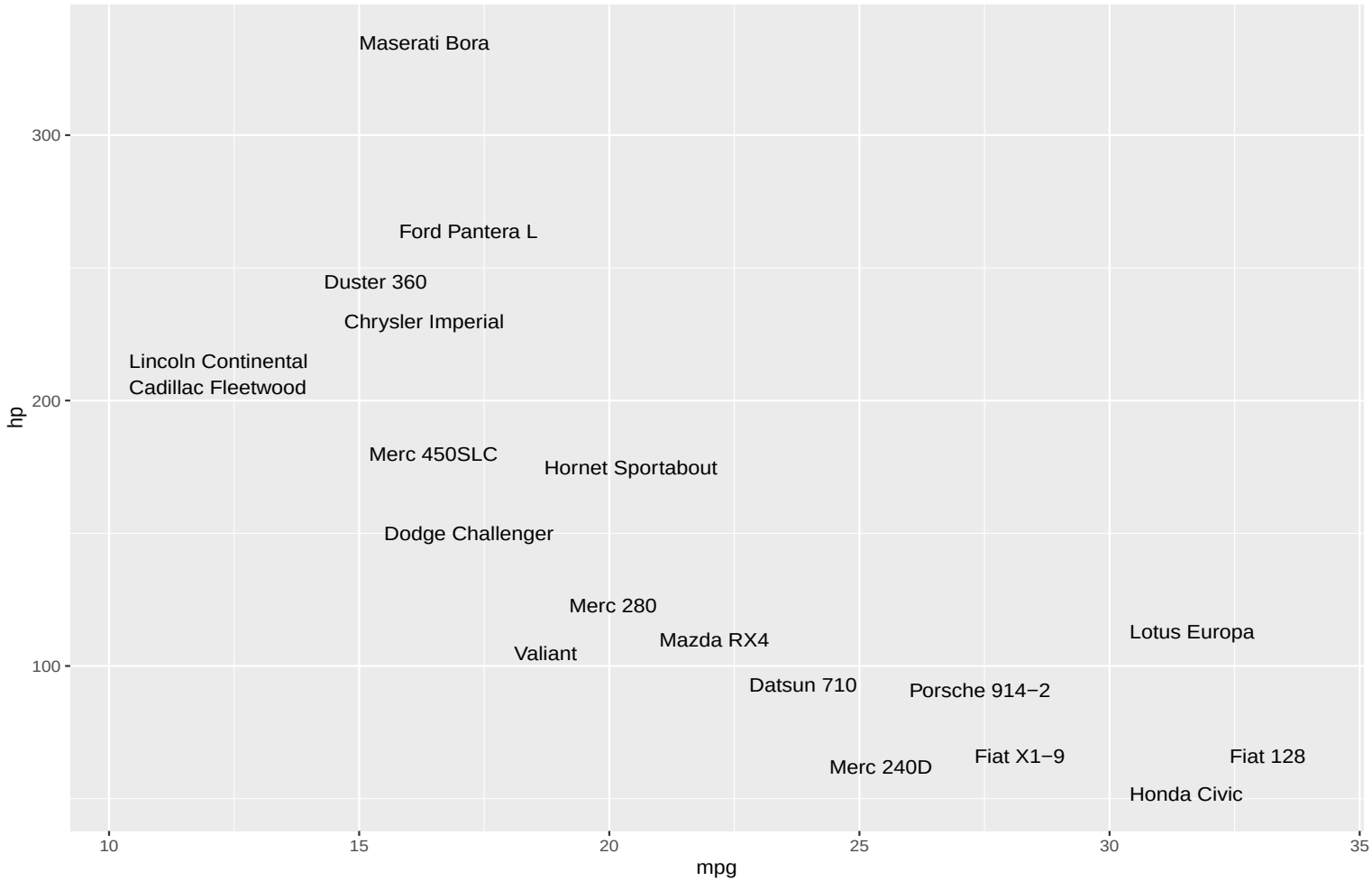
```
ggplot(mtcars, aes(x = mpg, y = hp, label = car)) +
  geom_text(check_overlap = T)
```



ggplot2

control positions of the labels (horizontal)

```
ggplot(mtcars, aes(x = mpg, y = hp, label = car )) +
  geom_text(check_overlap = T, hjust = 0)
```



ggplot2: statistical transformations

Statistical transformations, stats for short, summarise data in many useful ways.
Every geom has default statistical transformation.
geom_point(mapping = NULL, data = NULL, **stat = "identity"**, position = "identity", na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
geom_histogram(mapping = NULL, data = NULL, **stat = "bin"**, position = "stack", binwidth = NULL, bins = NULL, na.rm = FALSE, orientation = NA, show.legend = NA, inherit.aes = TRUE)

ggplot2: statistical transformations

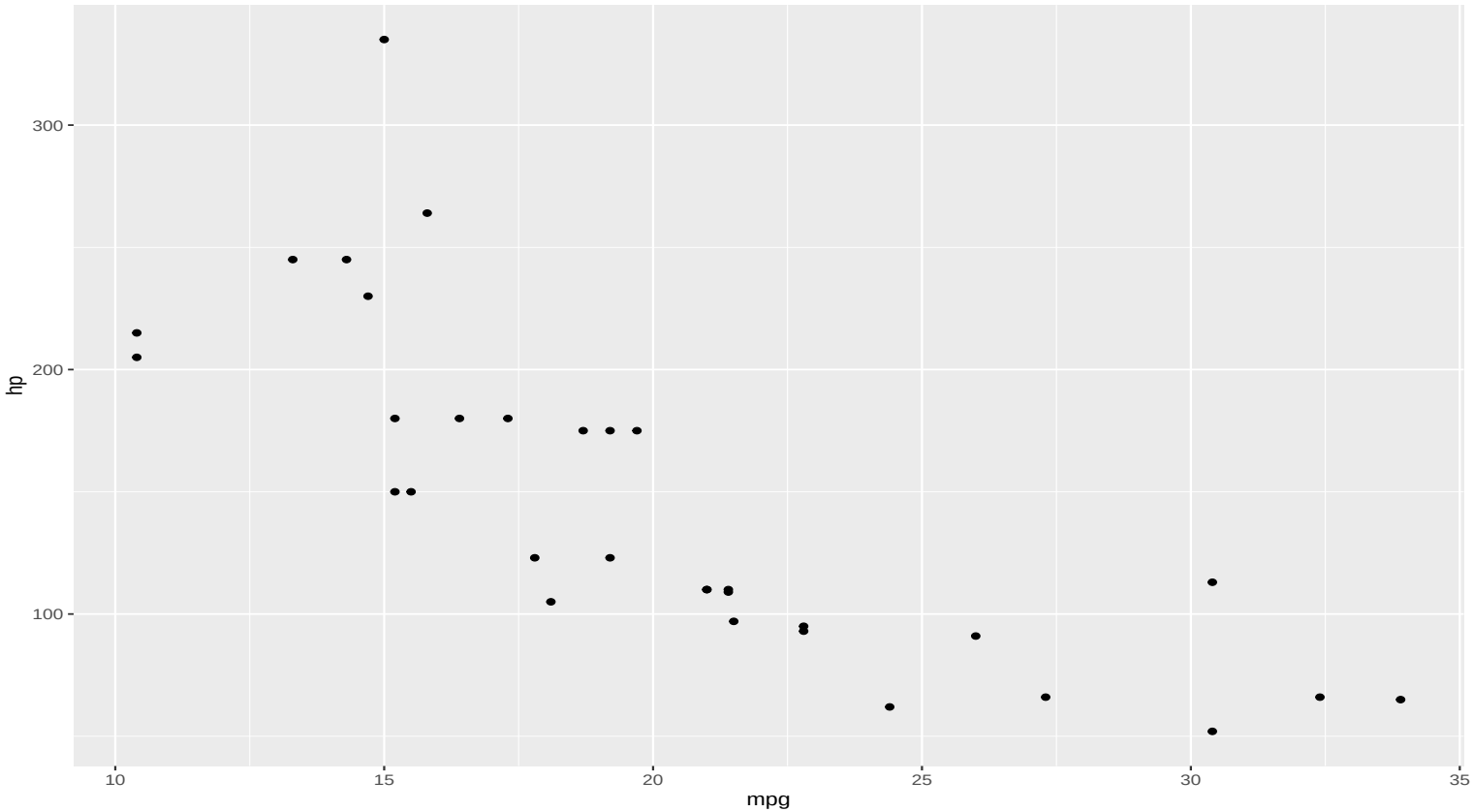
Identity transformation

$$f(x) = x$$

ggplot2: statistical transformations

stat_XXX and geom_XXX can be used interchangeably
The same: ggplot(mtcars, aes(mpg, hp)) + geom_point(stat = 'identity')

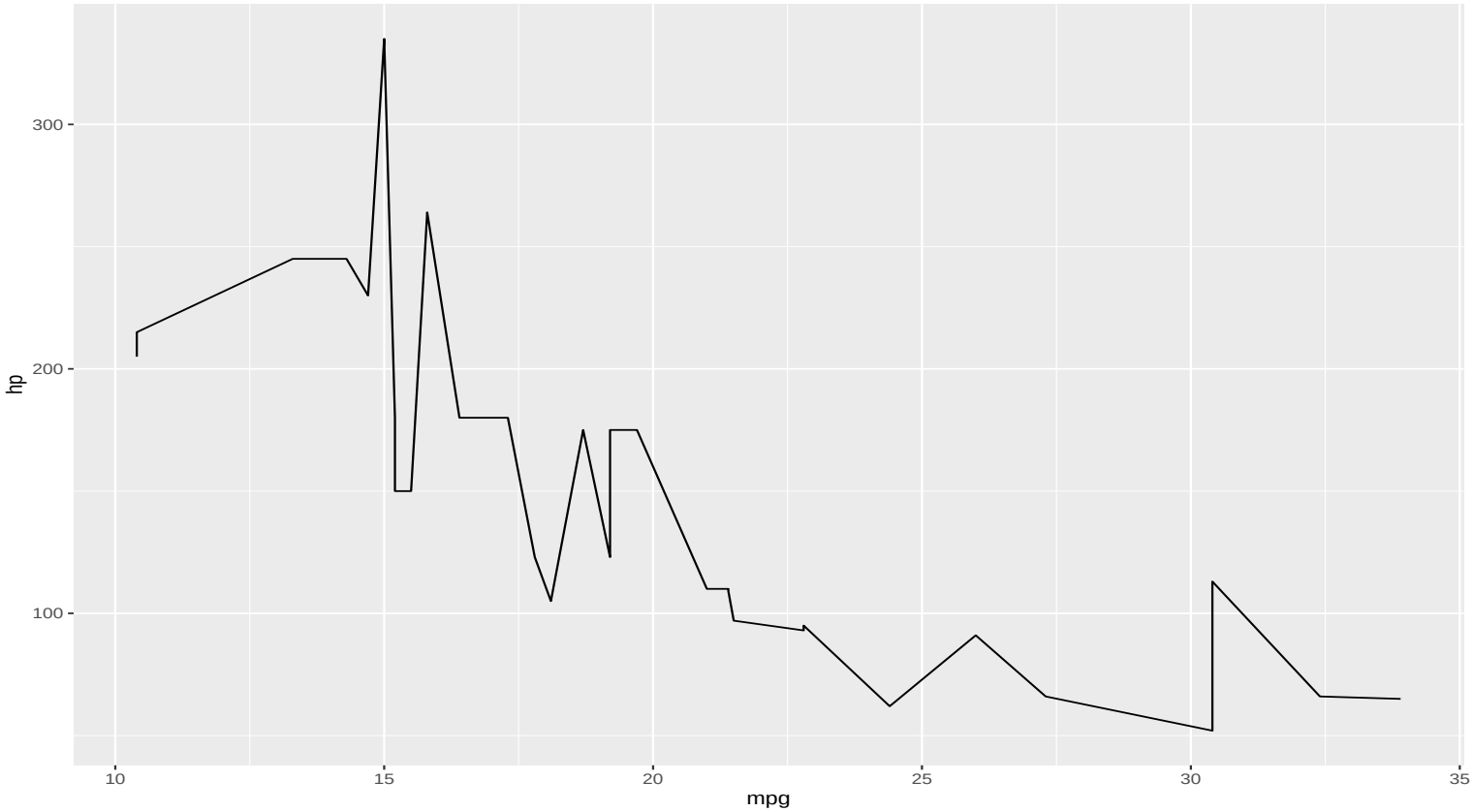
```
ggplot(mtcars, aes(mpg, hp)) + stat_identity(geom = 'point')
```



ggplot2: statistical transformations

line chart

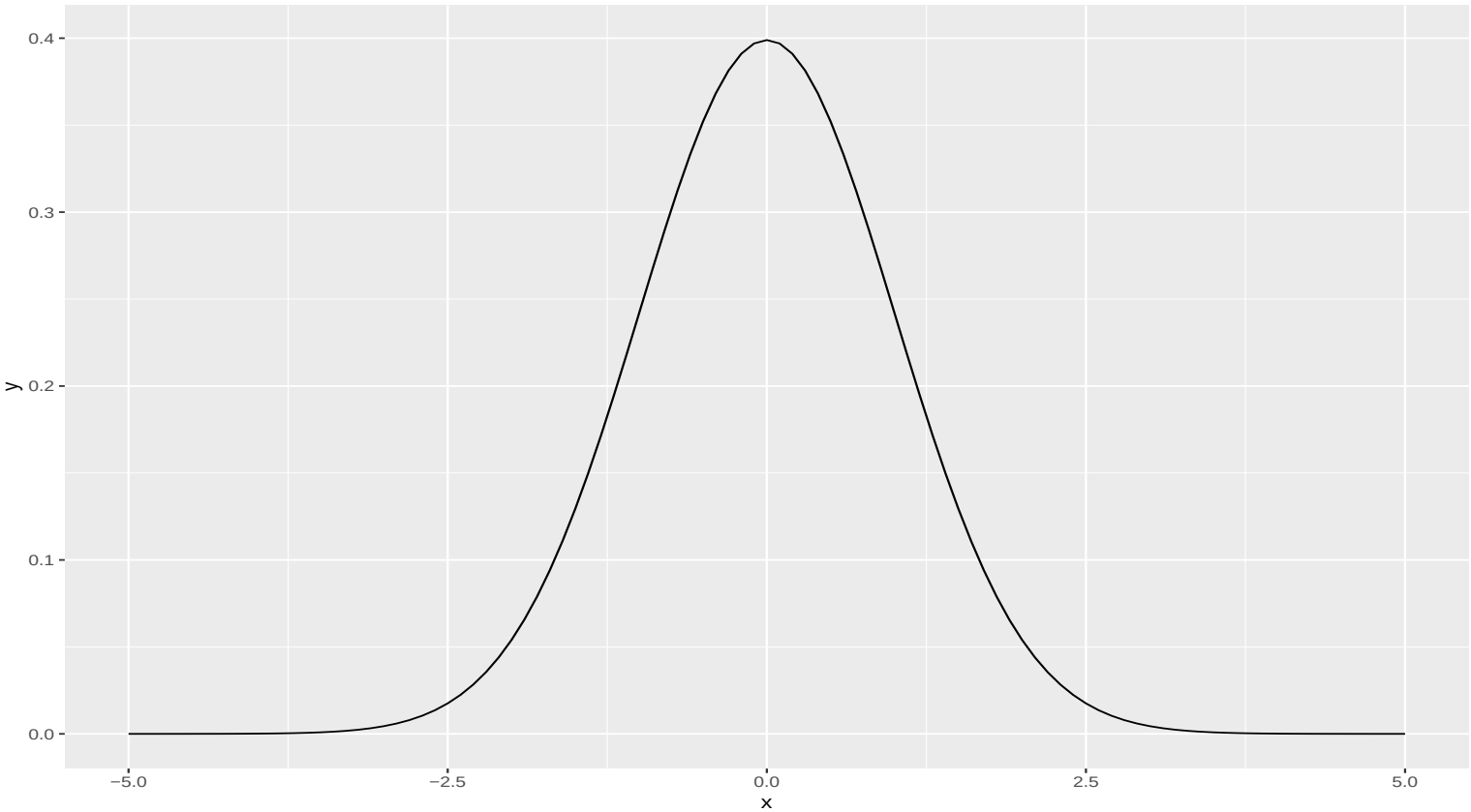
```
ggplot(mtcars, aes(mpg, hp)) + stat_identity(geom = 'line')
```



ggplot2: statistical transformations

use stat_function to draw any function you want Normal distribution, values along x axis (-5,5)

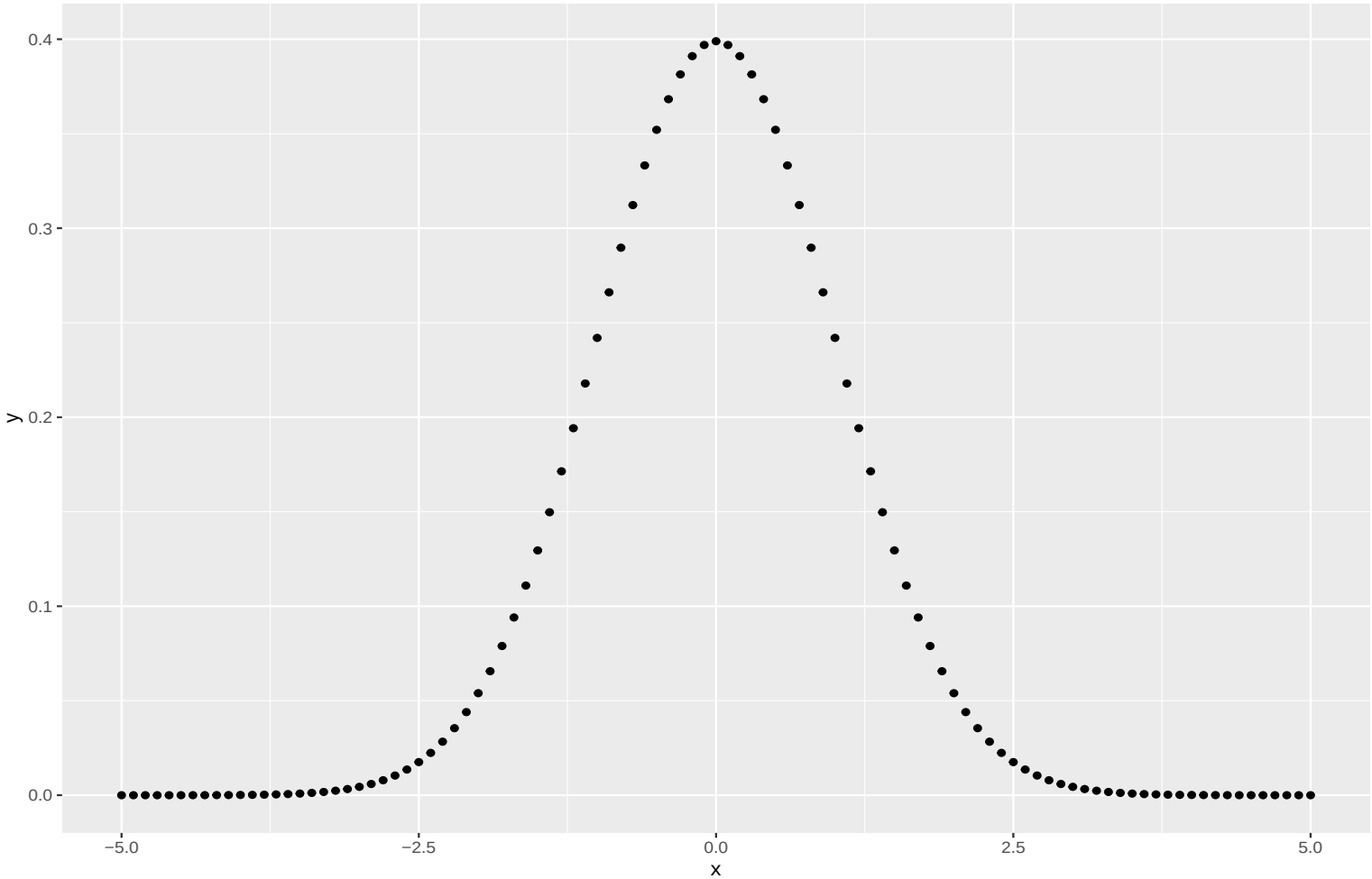
```
ggplot(data.frame(x = c(-5, 5)), aes(x)) + stat_function(fun = dnorm)
```



ggplot2: statistical transformations

change geom

```
ggplot(data.frame(x = c(-5, 5)), aes(x)) + stat_function(fun = dnorm, geom
```



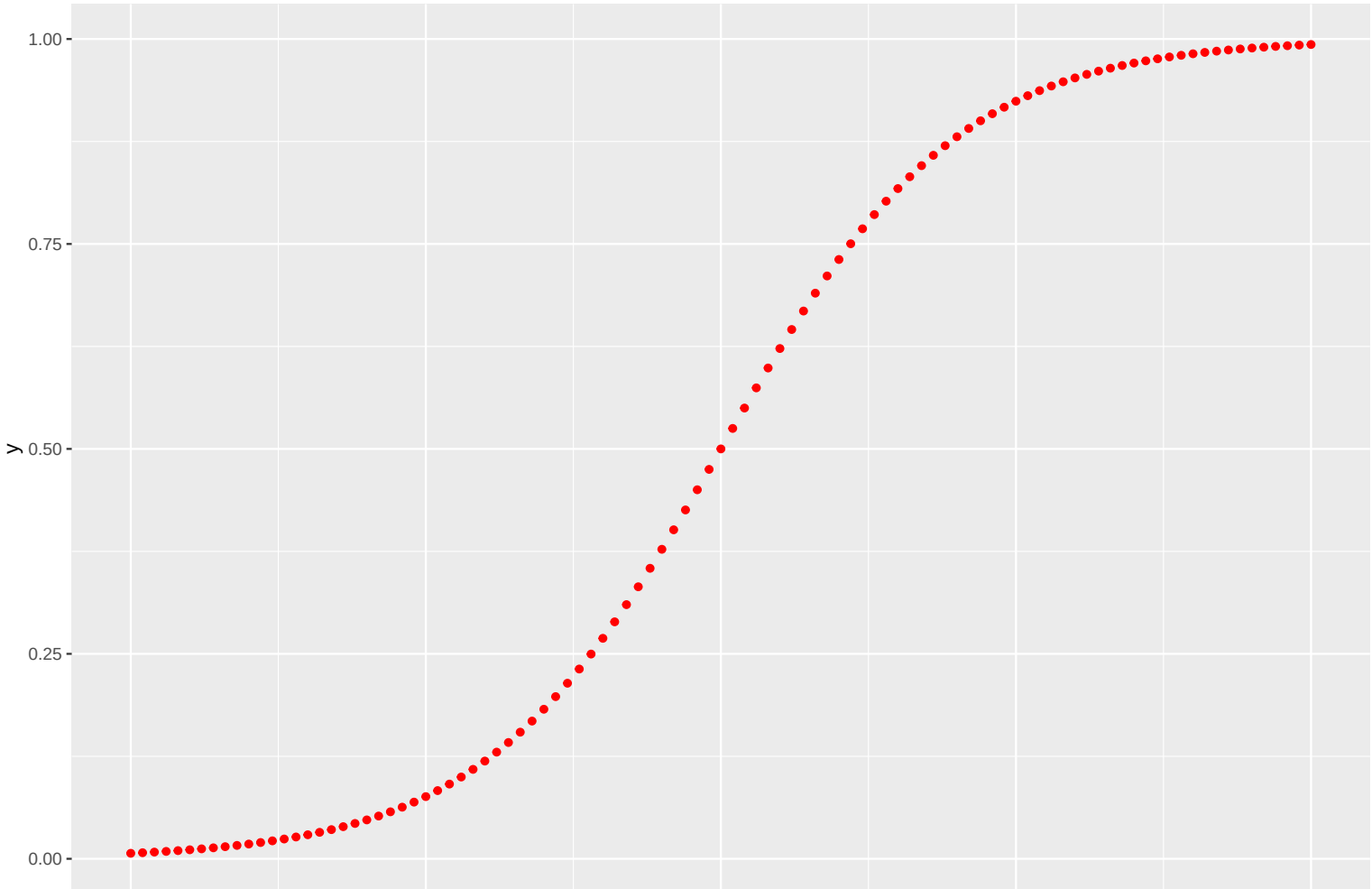
ggplot2: statistical transformations

With user defined functions: Logistic function

$$f(x) = \frac{1}{1 + e^{-x}}$$

ggplot2: statistical transformations

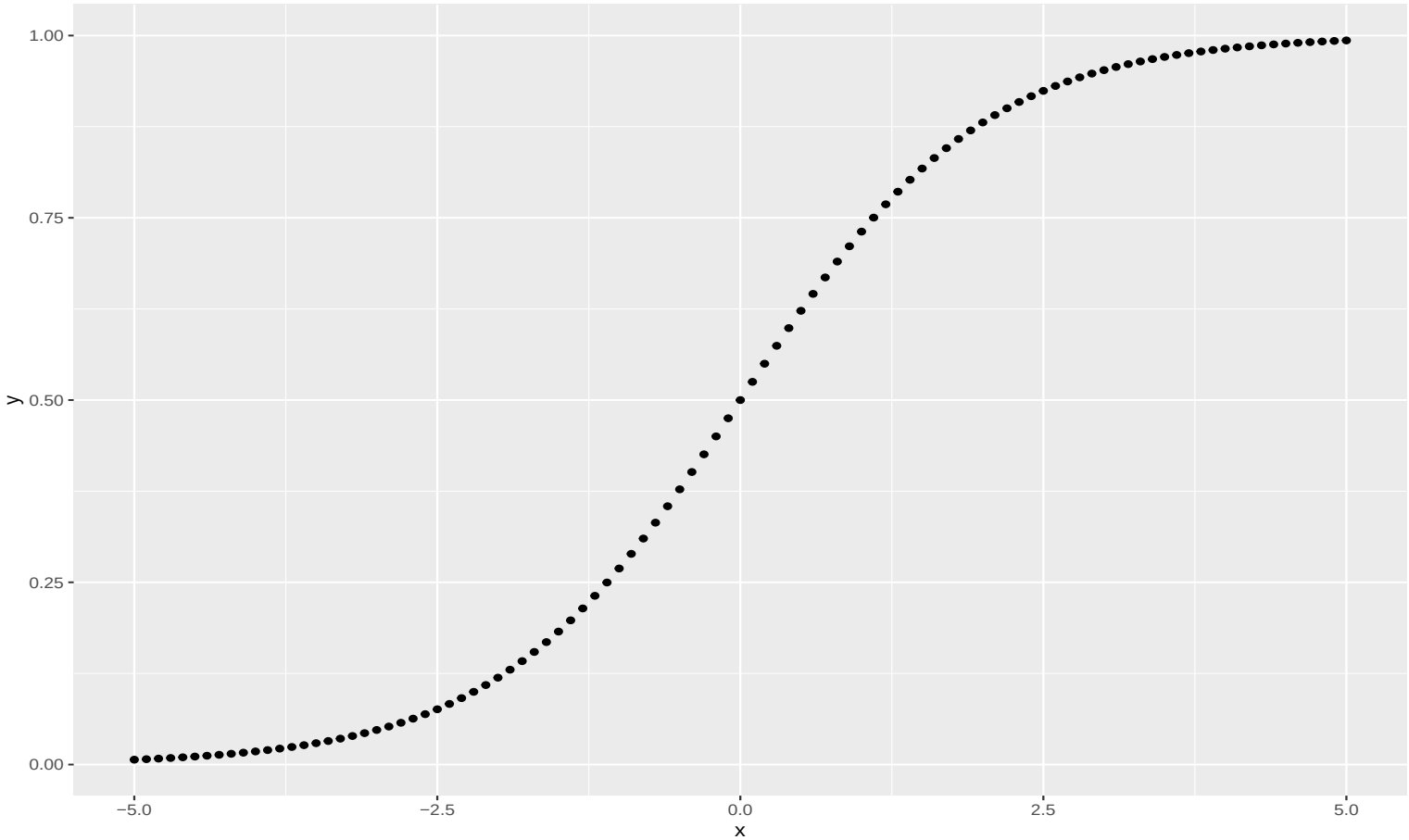
```
logistic <- function(x){1/(1 + exp(-x))}
ggplot(data.frame(x = c(-5, 5)), aes(x)) +
  stat_function(fun = logistic, geom = 'point', color = 'red')
```



ggplot2: statistical transformations

Define the function inside stat_function

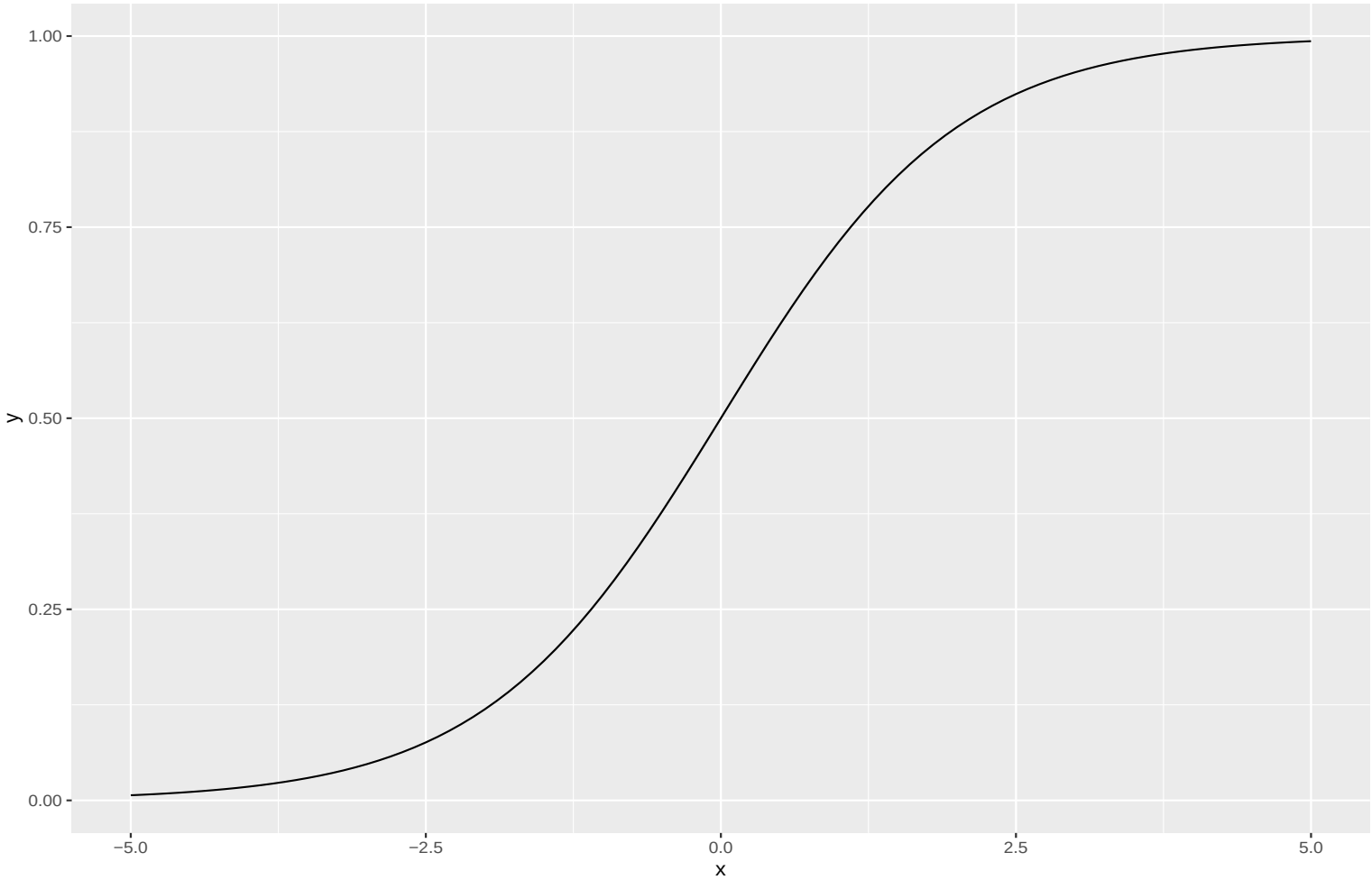
```
ggplot(data.frame(x = c(-5, 5)), aes(x)) +
  stat_function(fun = function(x) 1/(1 + exp(-x)), geom = 'point')
```



ggplot2: statistical transformations

geom_function() does the same but with defined geometry - line

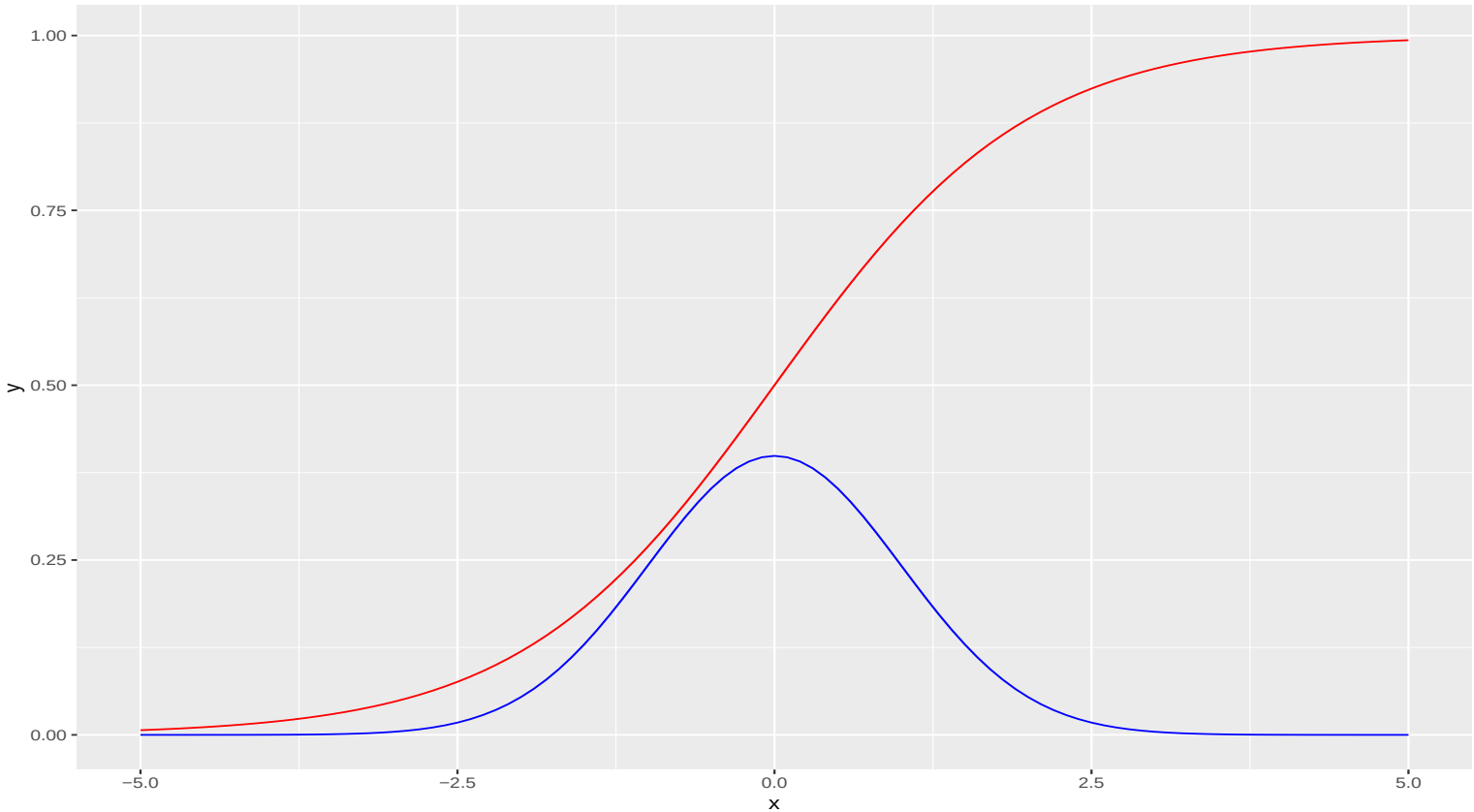
```
ggplot(data.frame(x = c(-5, 5)), aes(x)) + geom_function(fun = logistic)
```



ggplot2: statistical transformations

Plot two functions together

```
ggplot(data.frame(x = c(-5, 5)), aes(x)) +
  stat_function(fun = logistic, color = 'red') +
  stat_function(fun = dnorm, color = 'blue')
```



ggplot2: scales

A scale controls the mapping from data to aesthetic attributes, and we need a scale for every aesthetic used on a plot. Each scale operates across all the data in the plot, ensuring a consistent mapping from data to aesthetics. *(Hadley Wickham)*

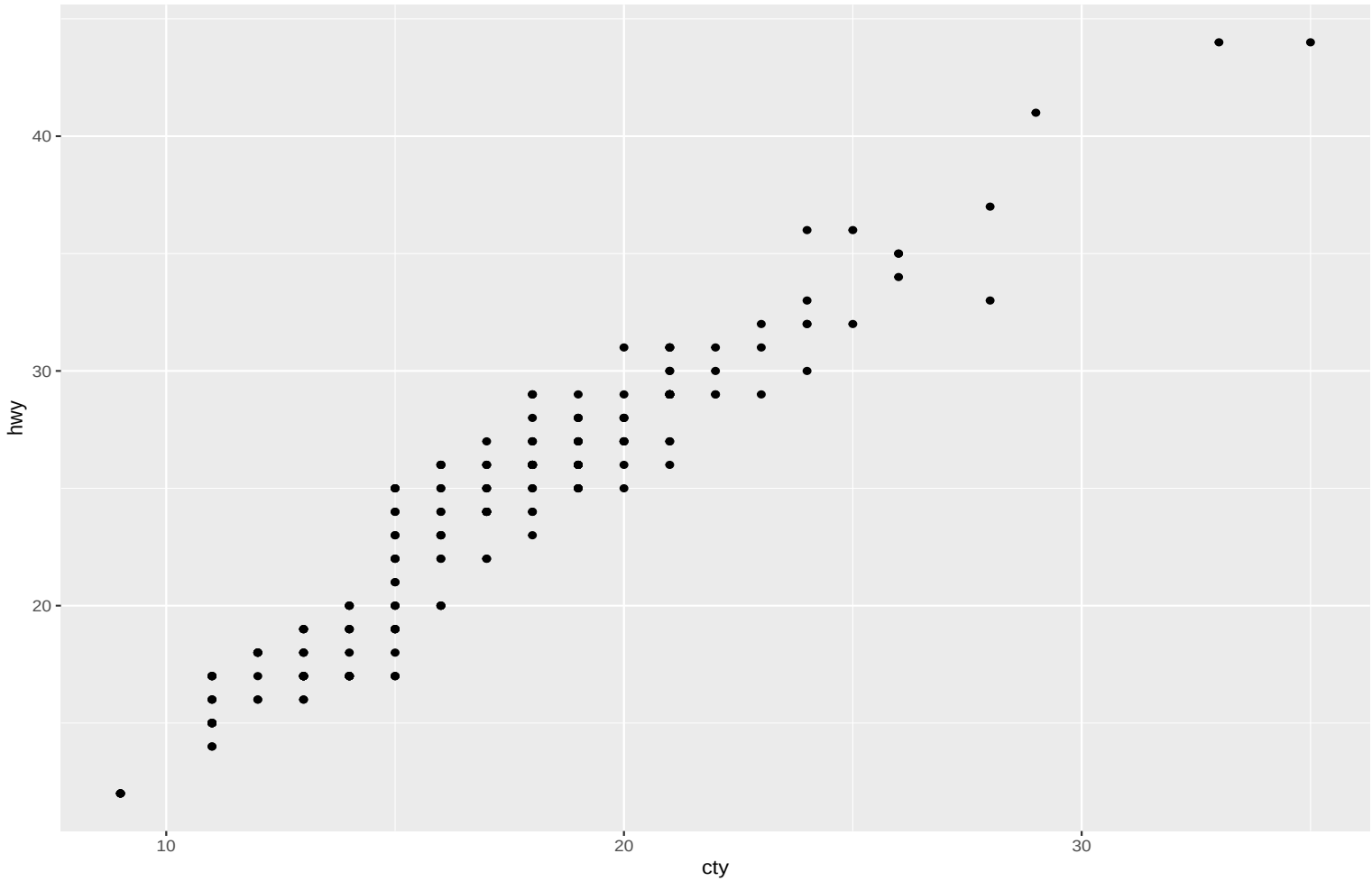
ggplot2: scales

They start with scale_, followed by the name of the aesthetic (e.g., colour_, shape_ or x_), and finally by the name of the scale (e.g., gradient, hue, manual, discrete, etc).
All aesthetics come with the default scale, you can override them with scale_ layers
Example: x, y aesthetics can be either continuous or discrete, thus we have scale_x_continuous, scale_x_discrete, scale_y_continuous, scale_y_discrete (and many more)

ggplot2: scales

Set your own breaks

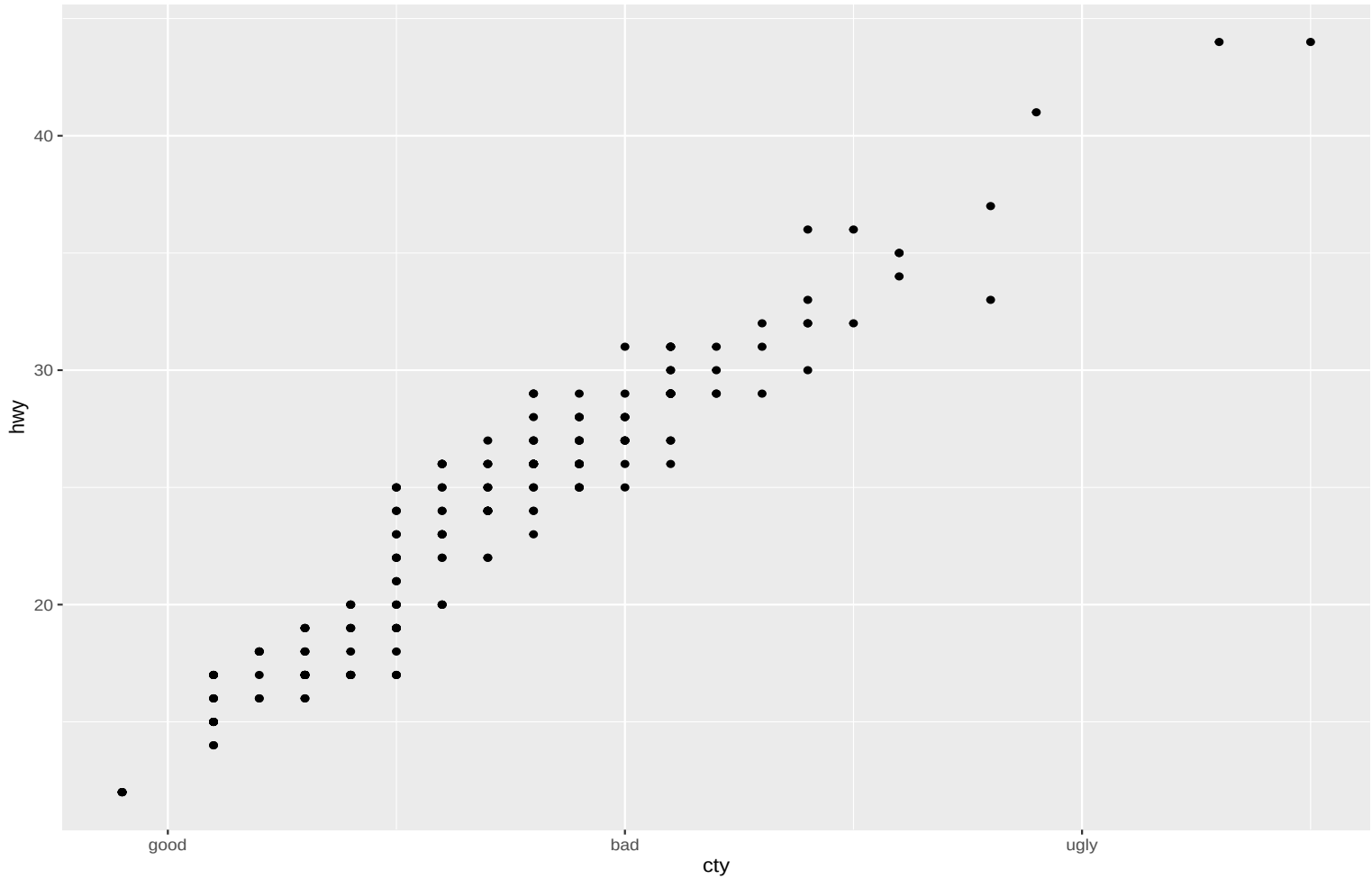
```
ggplot(mpg, aes(x = cty, y = hwy)) + geom_point() +
  scale_x_continuous(breaks = c(10,20,30))
```



ggplot2: scales

Adding labels to the breaks

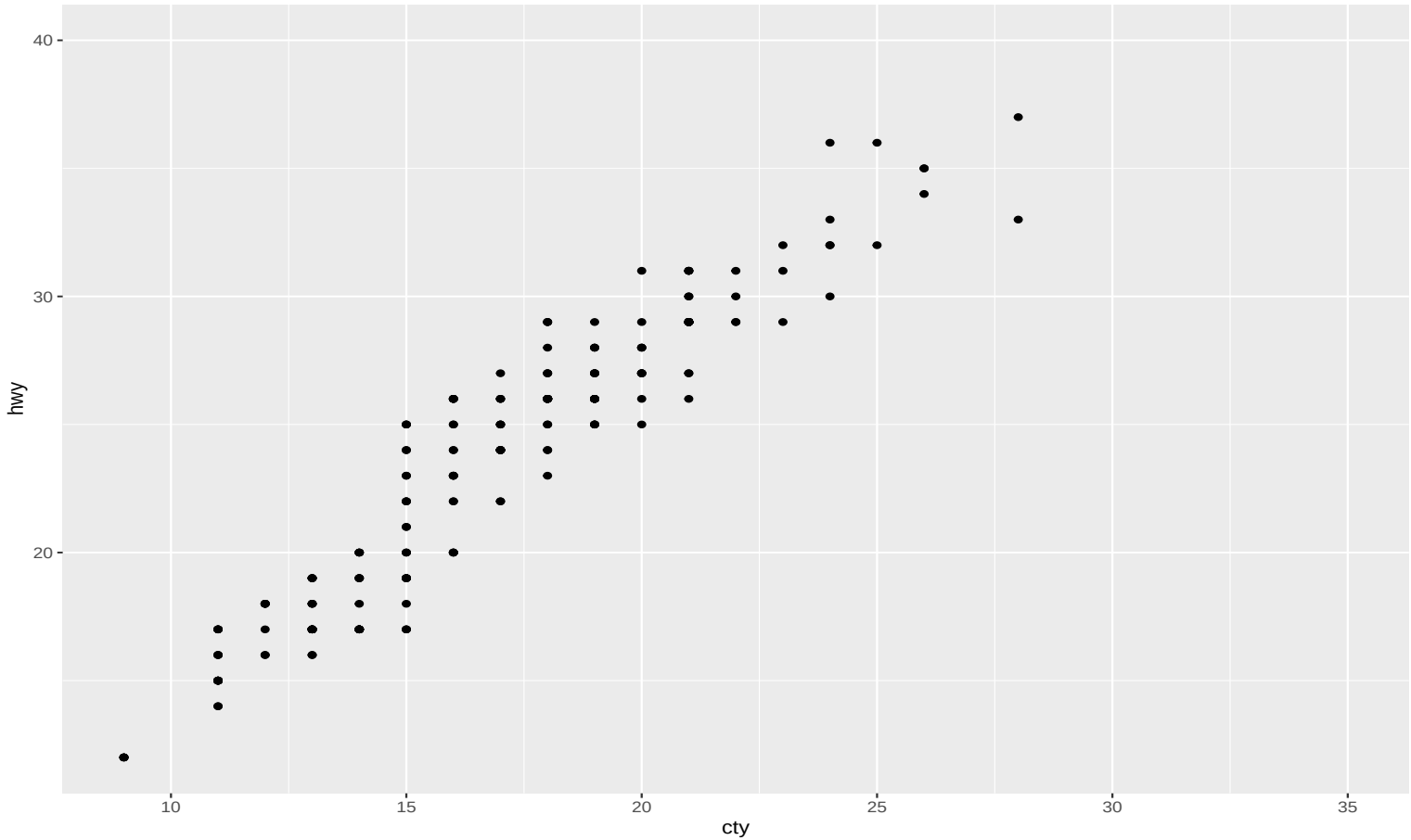
```
ggplot(mpg, aes(x = cty, y = hwy)) + geom_point() +
  scale_x_continuous(breaks = c(10,20,30), labels = c('good', 'bad', 'ugly'
```



ggplot2: scales

Setting the limits for y axis (note the minimum value is NA, thus decided by ggplot2)

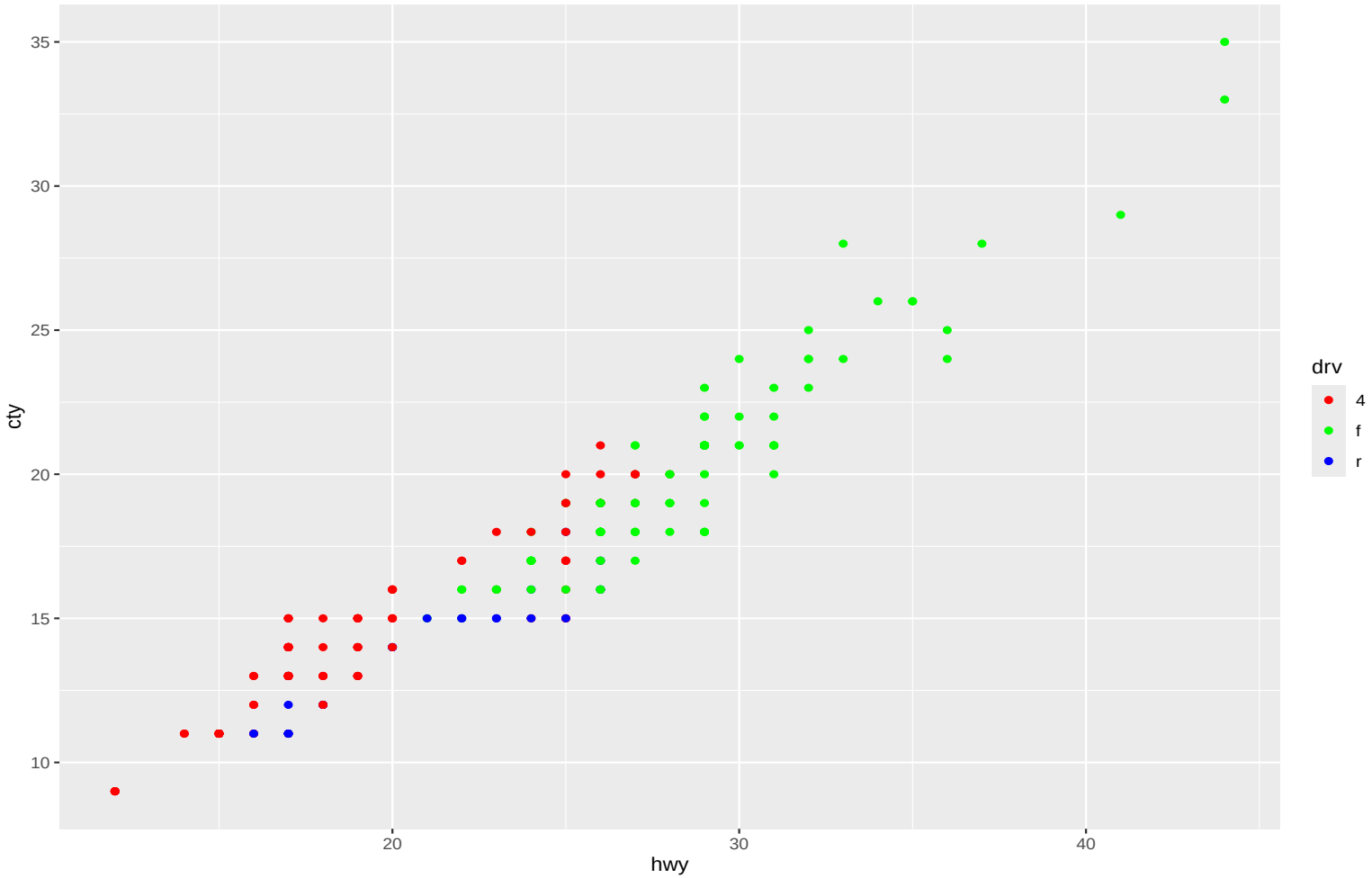
```
ggplot(mpg, aes(x = cty, y = hwy)) + geom_point() +
  scale_y_continuous(limits = c(NA,40))
```



ggplot2: scales

Using scale with color aesthetics Set color manually

```
ggplot(data = mpg, mapping = aes(x = hwy, y = cty, color = drv)) +
  geom_point() + scale_color_manual(values = c('red', 'green', 'blue'))
```



ggplot2: coordinate system

Coordinate systems tie together the two position scales to produce a 2d location. Currently, ggplot2 comes with six different coordinate systems. All these coordinate systems are two dimensional.
Coordinate systems in ggplot

- cartesian
- equal
- flip
- trans
- map
- polar

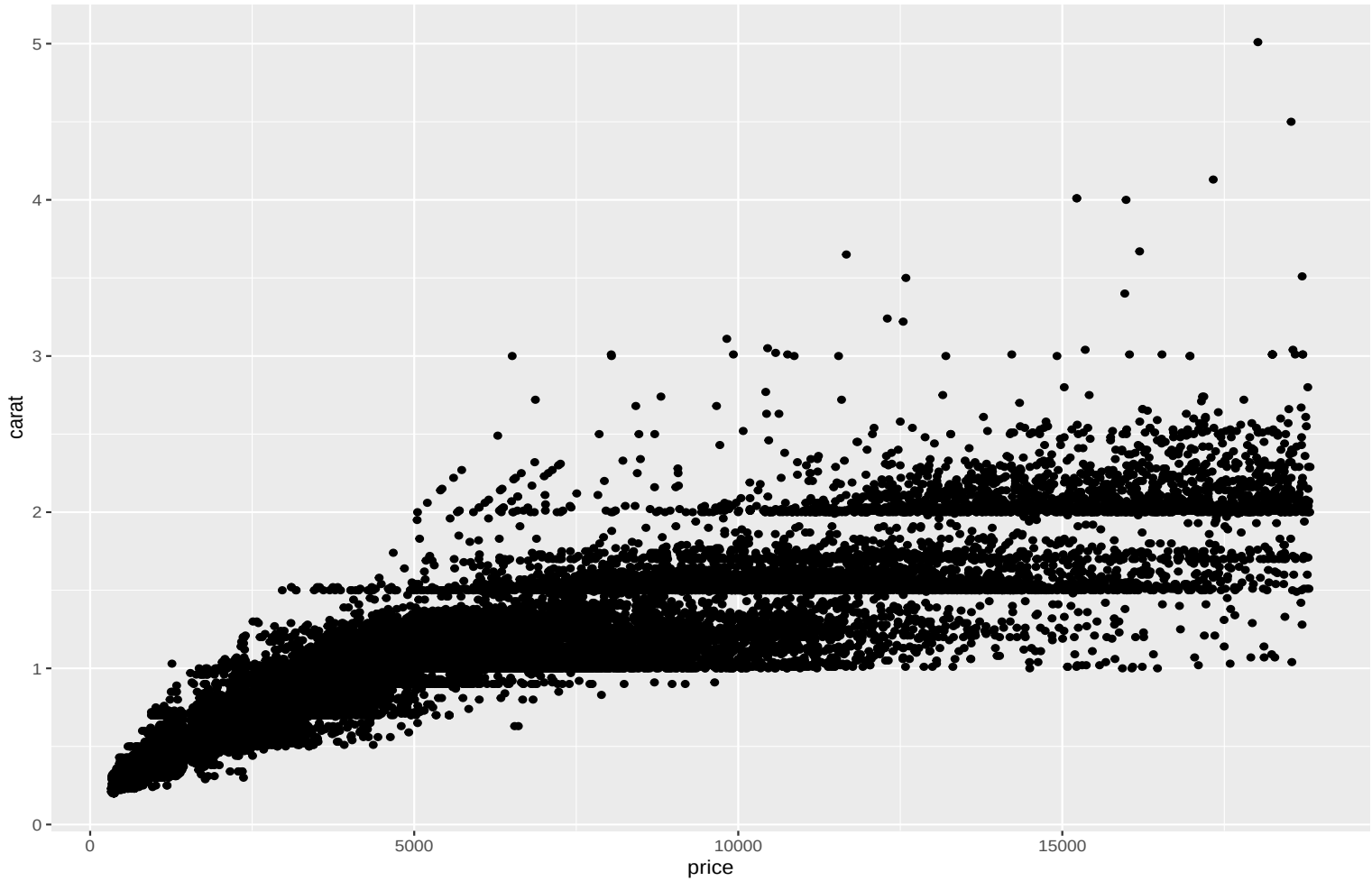
ggplot2: coordinate system

- The four Cartesian-based coordinate systems, coord_cartesian, coord_equal, coord_flip and coord_trans, share a number of common features.
- They are still essentially Cartesian because the x and y positions map orthogonally to x and y positions on the plot.

ggplot2: coordinate system

Diamonds dataset

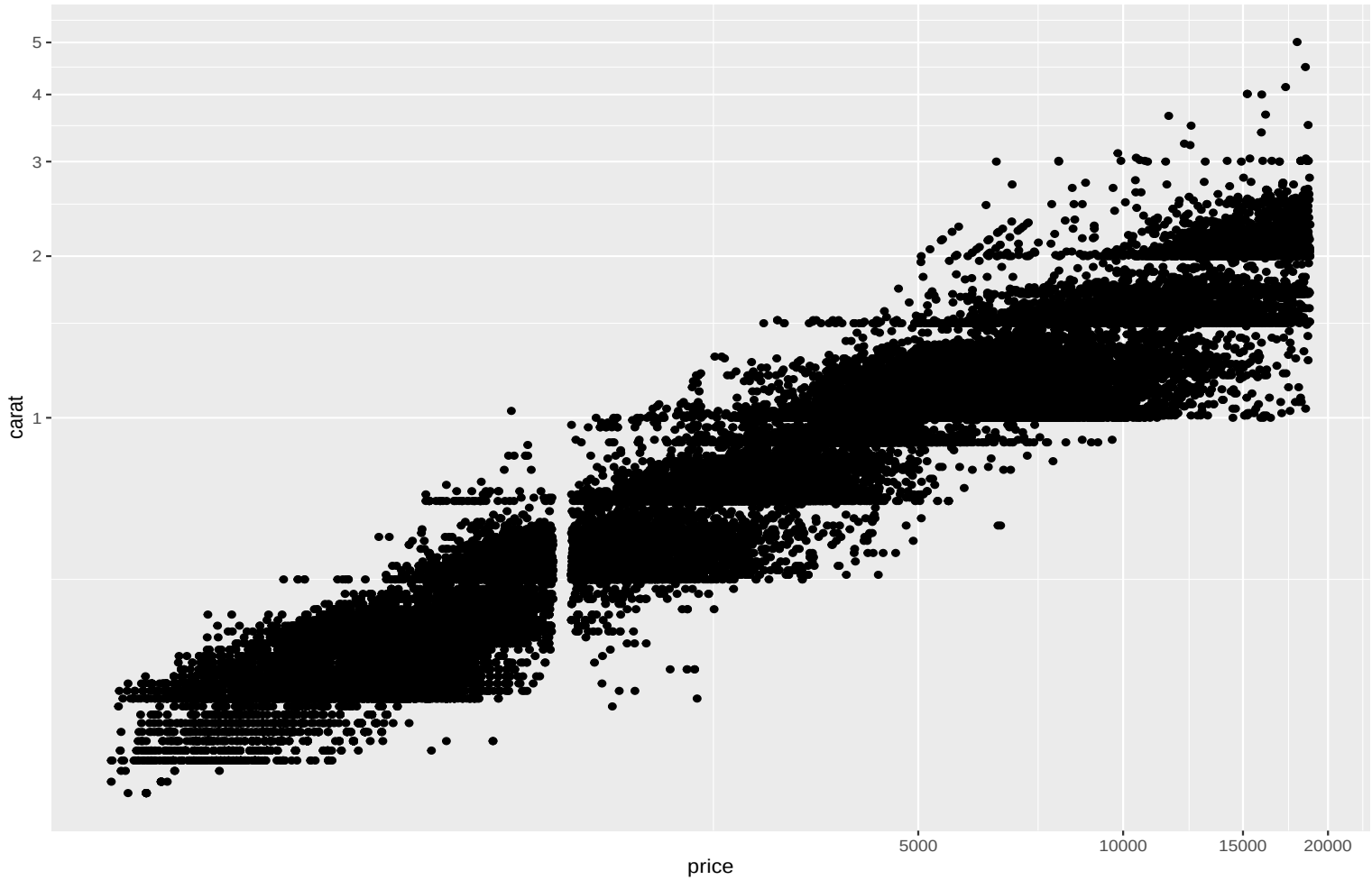
```
ggplot(data = diamonds, aes(price, carat)) + geom_point()
```



ggplot2: coordinate system

Transforming the axes: Log-Log plot

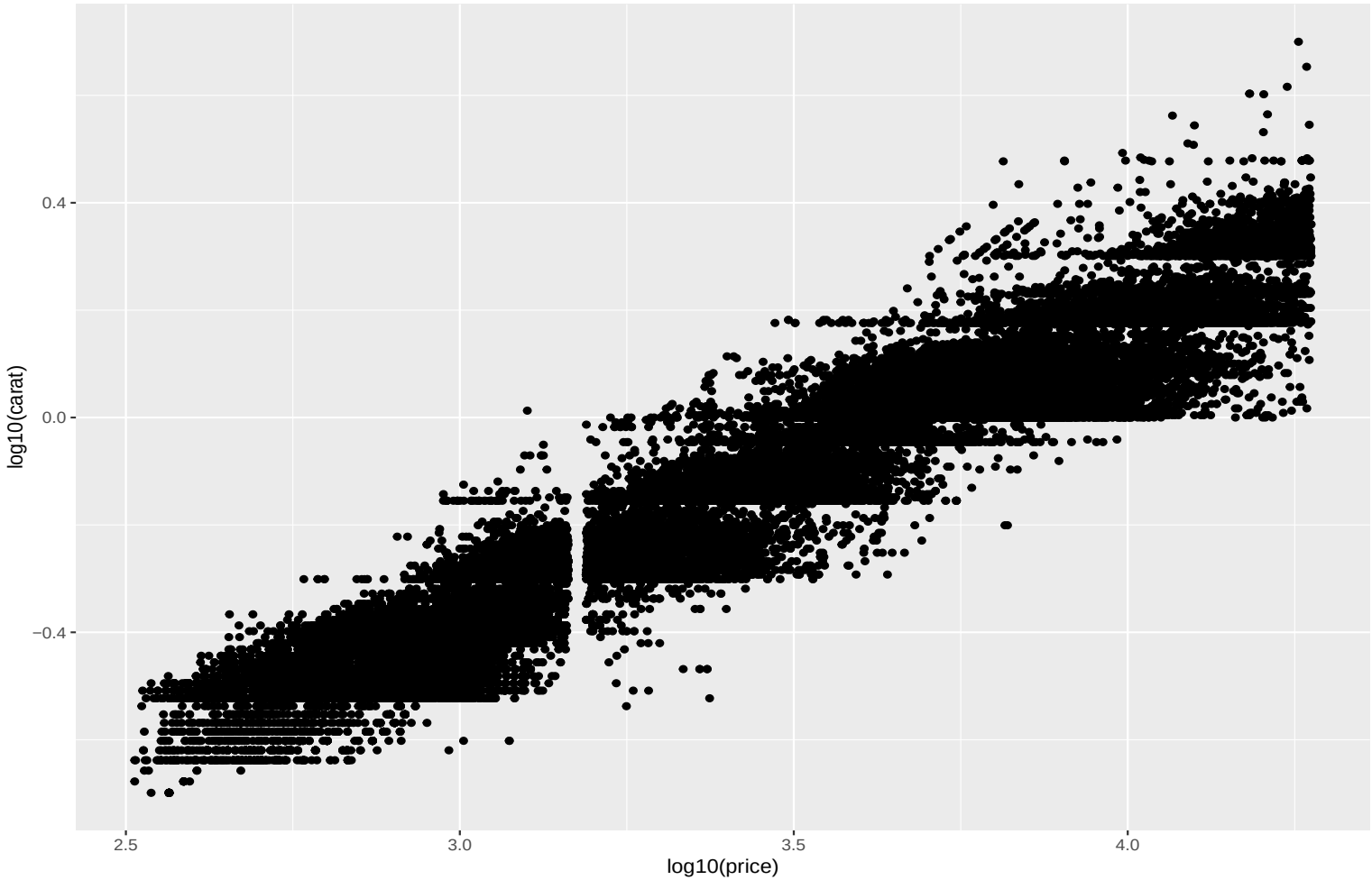
```
ggplot(data = diamonds, aes(price, carat)) + geom_point() +
  coord_trans(x = 'log10', y = 'log10')
```



ggplot2: coordinate system

You can get "almost" the same result with this

```
ggplot(data = diamonds, aes(log10(price), log10(carat))) +
  geom_point()
```

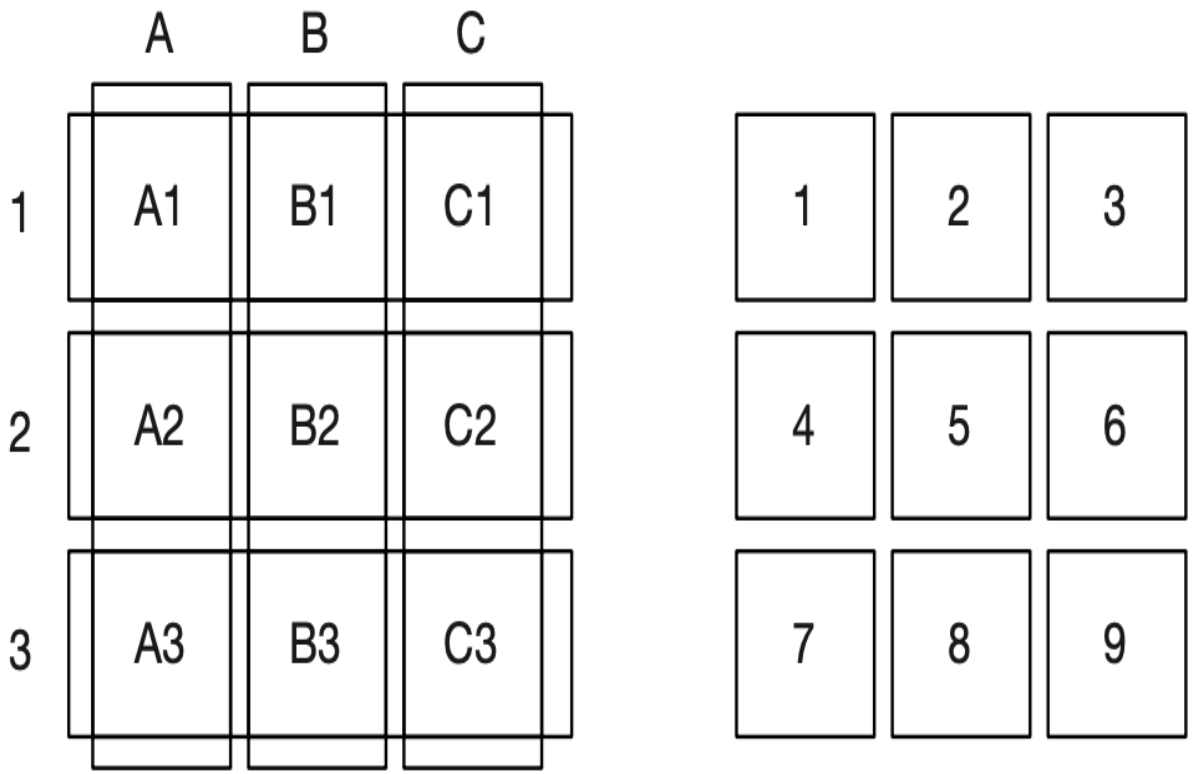


ggplot2: faceting

- Faceting creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset in an arrangement that facilitates comparison.
- ggplot2 has two layers for faceting: facet_grid and facet_wrap

ggplot2: faceting

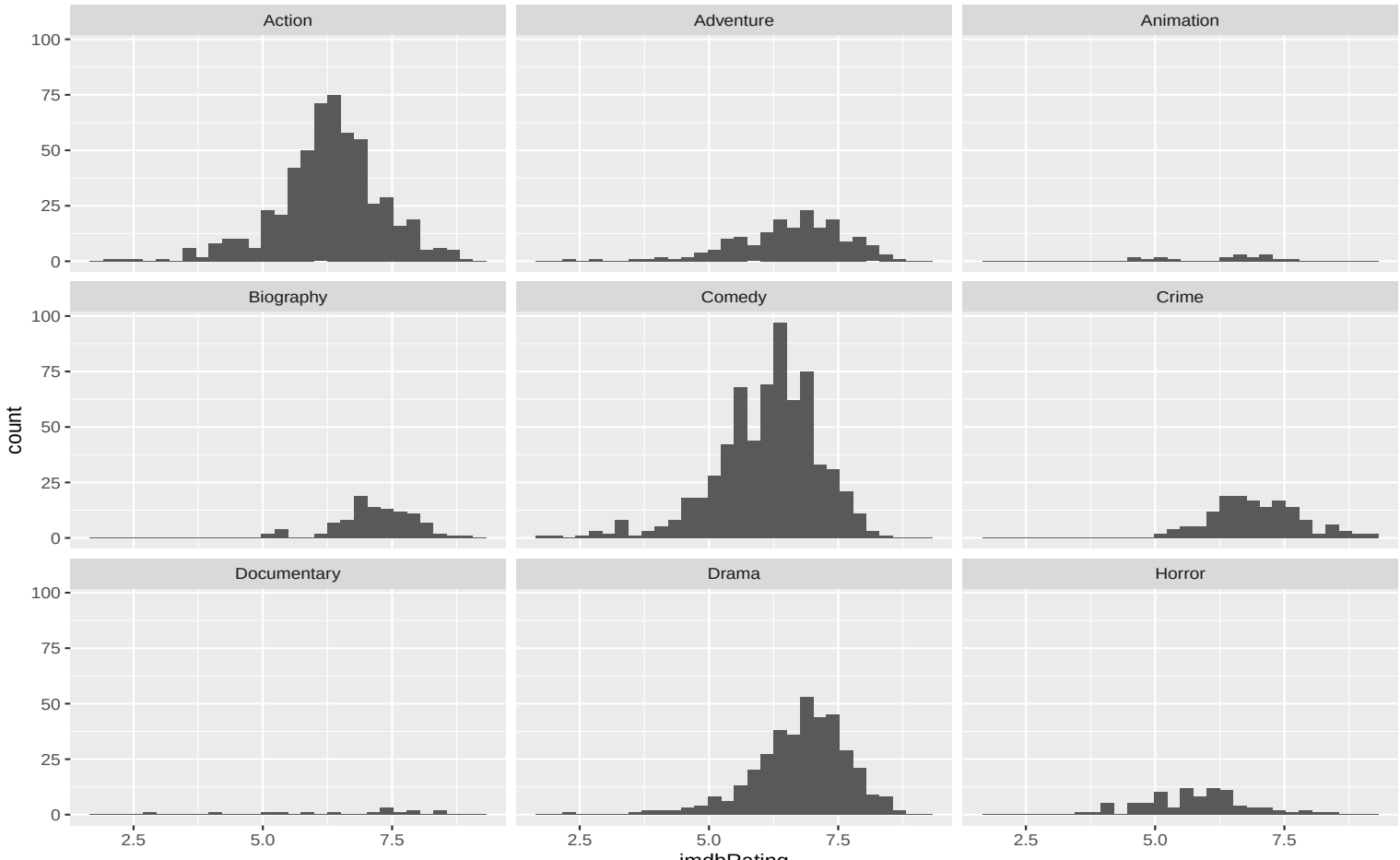
- facet_grid() produces a 2d grid of panels defined by variables which form the rows and columns,
- facet_wrap() produces a 1d ribbon of panels that is wrapped into 2d.



ggplot2: faceting

Get the histogram of imdbRating by genre_first

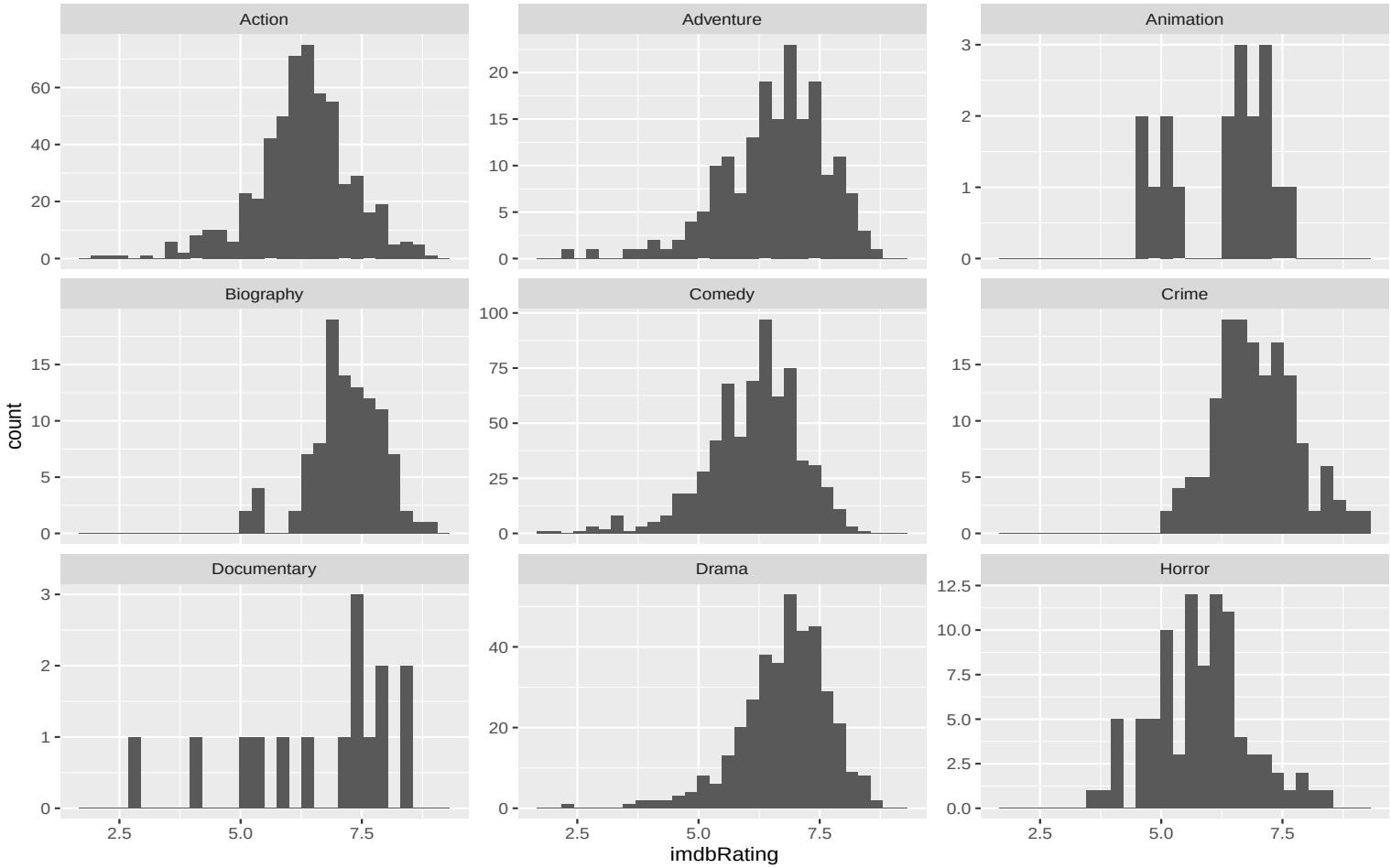
```
movies_small <- read.csv('Data/movies_small.csv')
ggplot(movies_small, aes(x = imdbRating)) + geom_histogram() +
  facet_wrap(~genre_first)
```



ggplot2: faceting

Free y axis to vary

```
ggplot(movies_small, aes(x = imdbRating)) + geom_histogram() +
  facet_wrap(~genre_first, scales = 'free_y')
```

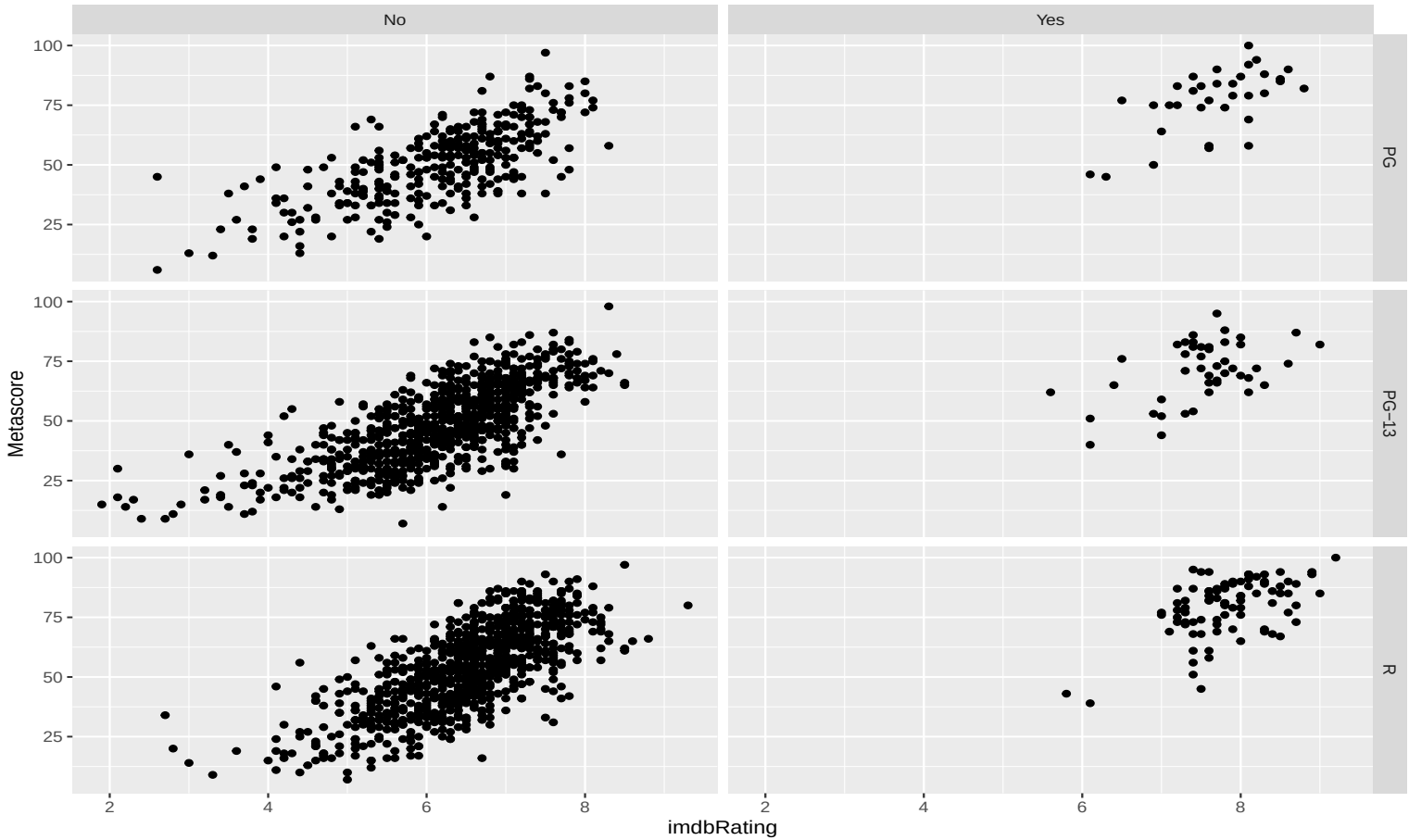


ggplot2: faceting

- facet_grid() allows to create 2d grid of panels defined by variables which form the rows and columns.
- Look at the formula: Rated is in rows, has_oscar is in columns

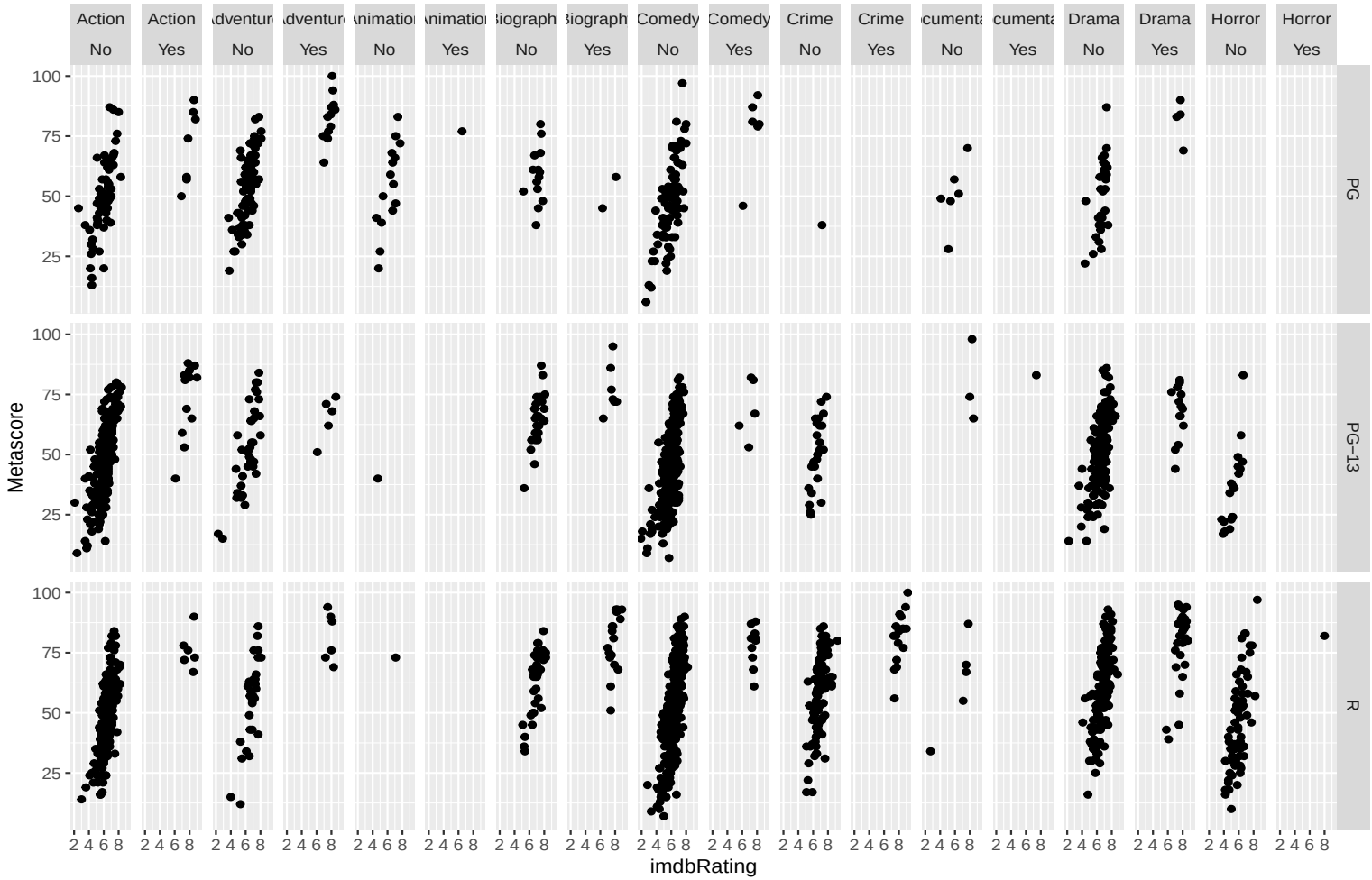
ggplot2: faceting

```
ggplot(movies_small, aes(x = imdbRating, y = Metascore)) + geom_point() +
  facet_grid(Rated~has_oscar)
```



ggplot2: faceting - more than 2 variables

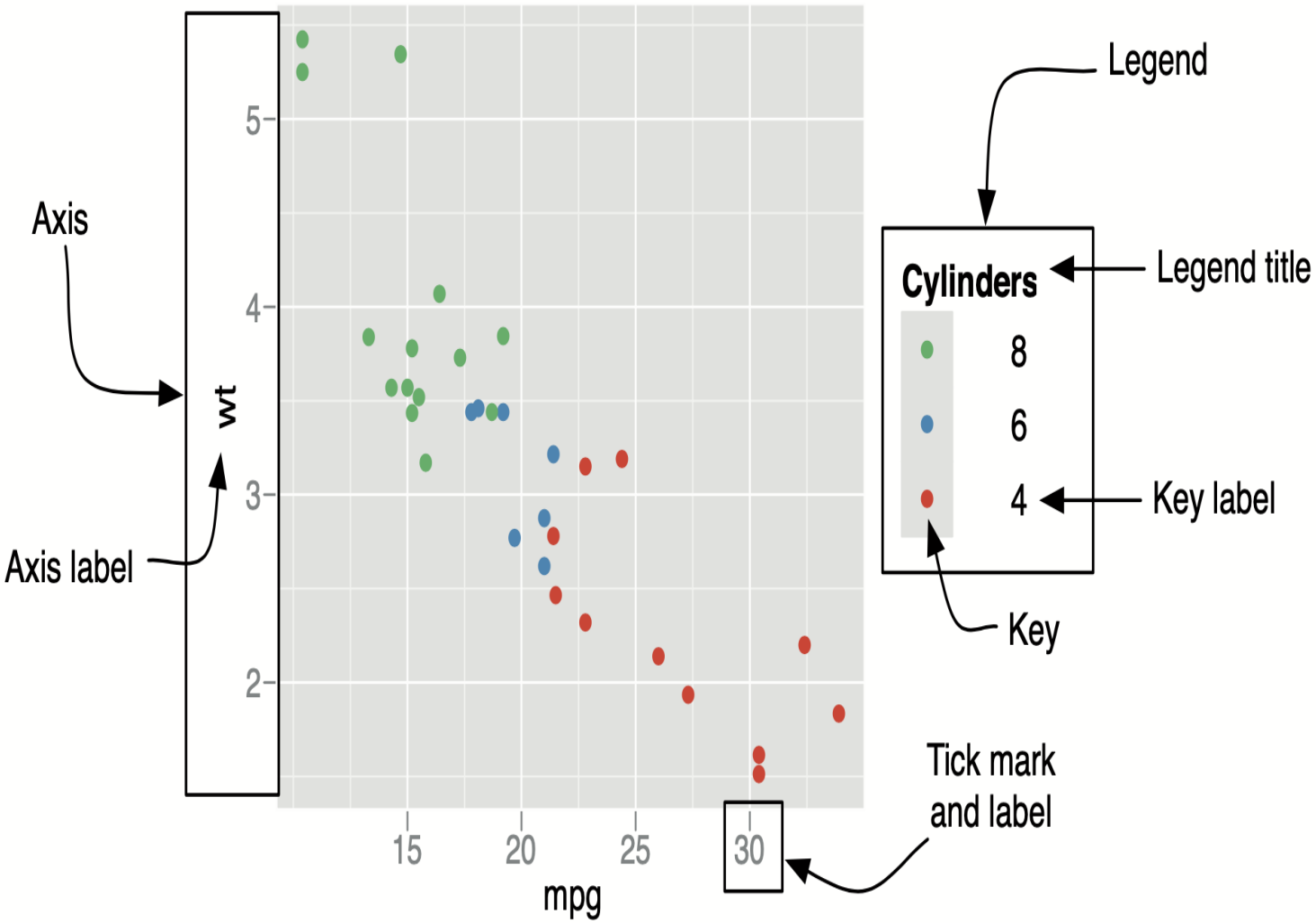
```
ggplot(movies_small, aes(x = imdbRating, y = Metascore)) + geom_point() +
  facet_grid(Rated~genre_first+has_oscar)
```



ggplot2: guides and themes

- Collectively, axes and legends are called guides
- They allow you to read observations from the plot and map them back to their original values.

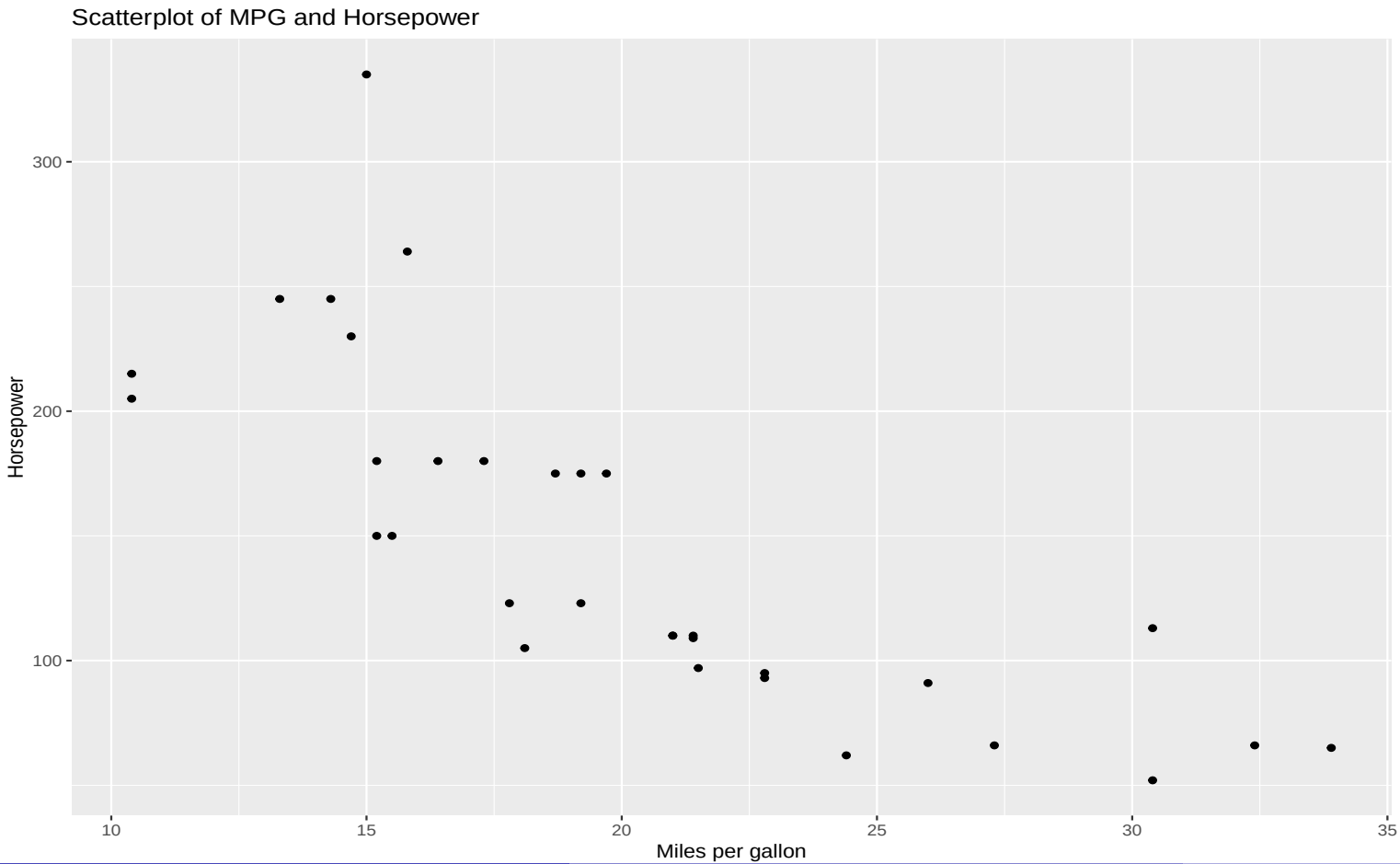
ggplot2: guides and themes



ggplot2: guides and themes

Axis labels and title

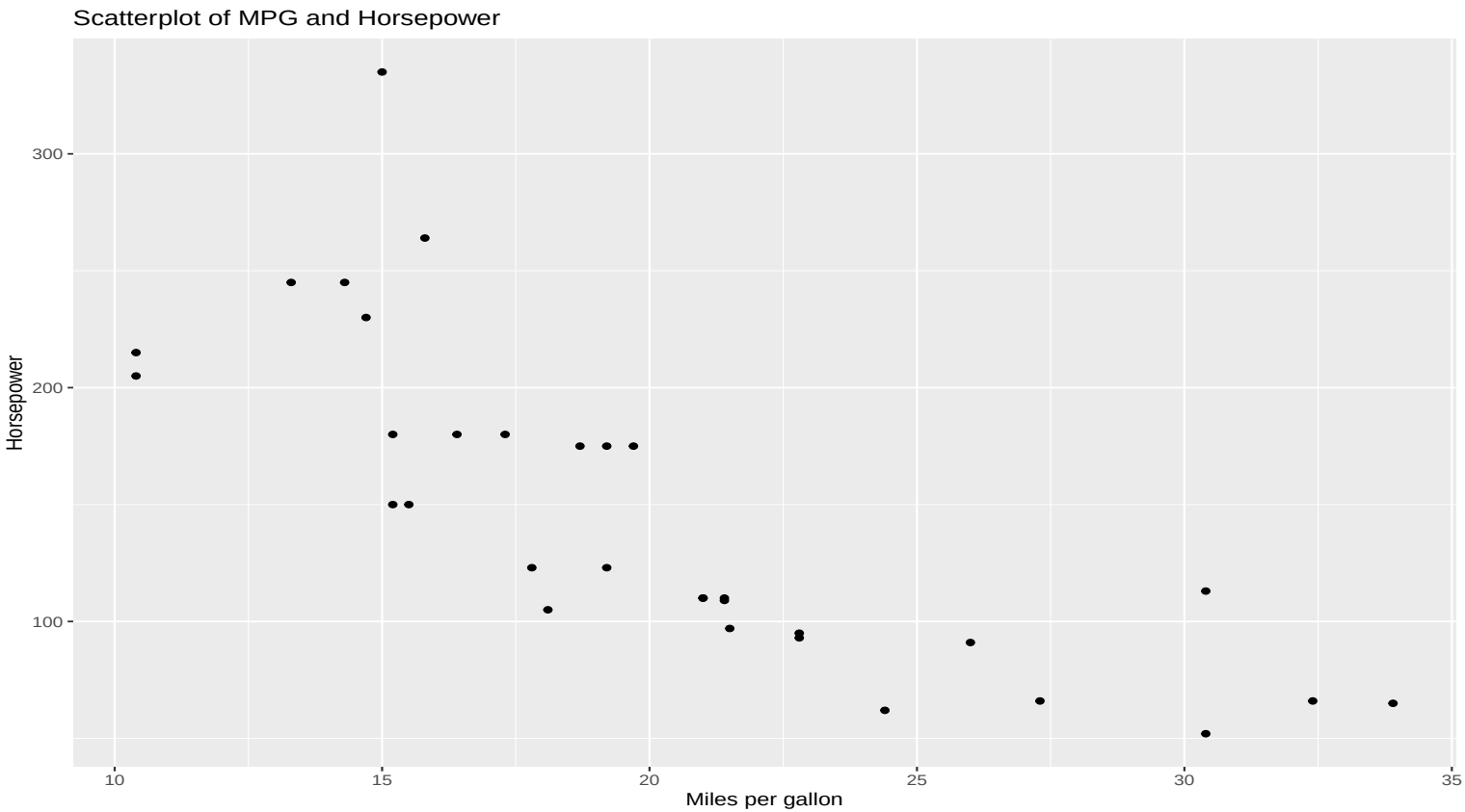
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  xlab('Miles per gallon') + ylab('Horsepower') +
  ggtitle('Scatterplot of MPG and Horsepower')
```



ggplot2: guides and themes

The same using labs() layer

```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  labs(x = 'Miles per gallon', y = 'Horsepower',
       title = 'Scatterplot of MPG and Horsepower')
```



ggplot2: guides and themes

- The appearance of non-data elements of the plot is controlled by the theme system.
- The theme system does not affect how the data is rendered by geoms, or how it is transformed by scales. - They don't change the perceptual properties of the plot, but help to make the plot aesthetically pleasing or match existing style guides.
- Themes give control over things like the fonts in all parts of the plot: the title, axis labels, axis tick labels, legend labels and legend key labels; and the colour of ticks, grid lines and backgrounds (panel, plot, strip and legend).

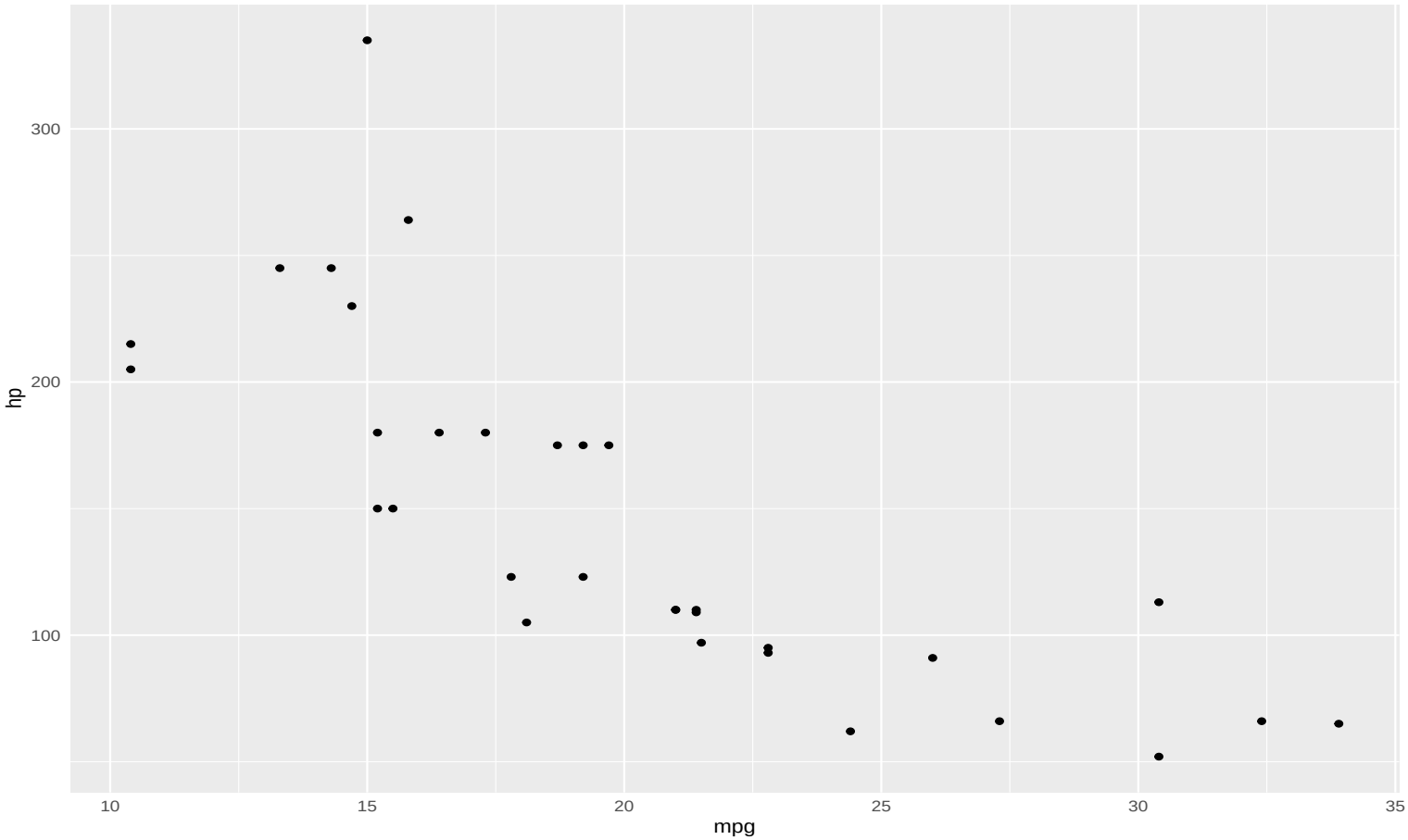
ggplot2: guides and themes

- A theme is made up of multiple elements which control the appearance of a single item on the plot.
- There are three elements that have individual x and y settings: **axis.text, axis.title and strip.text**. - Having a different setting for the horizontal and vertical elements allows you to control how text should appear in different orientations.
- The appearance of each element is controlled by an element function. There are four basic types of built-in element functions: **text, lines and segments, rectangles and blank**.
- Each element function has a set of parameters that control the appearance.

ggplot2: guides and themes

If you want to remove an element, use element_blank() - remove axis ticks

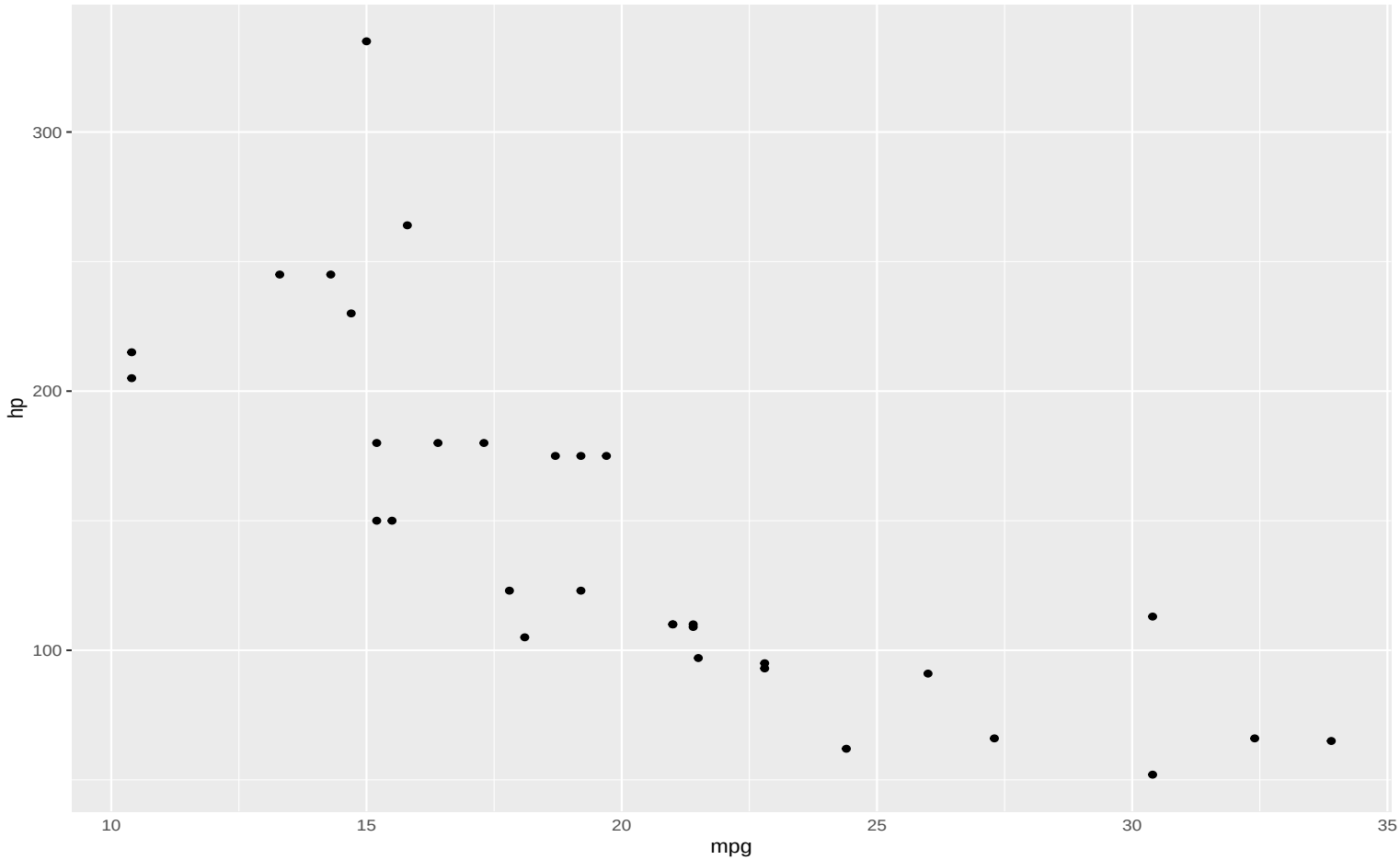
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme(axis.ticks = element_blank())
```



ggplot2: guides and themes

Control elements of the axis

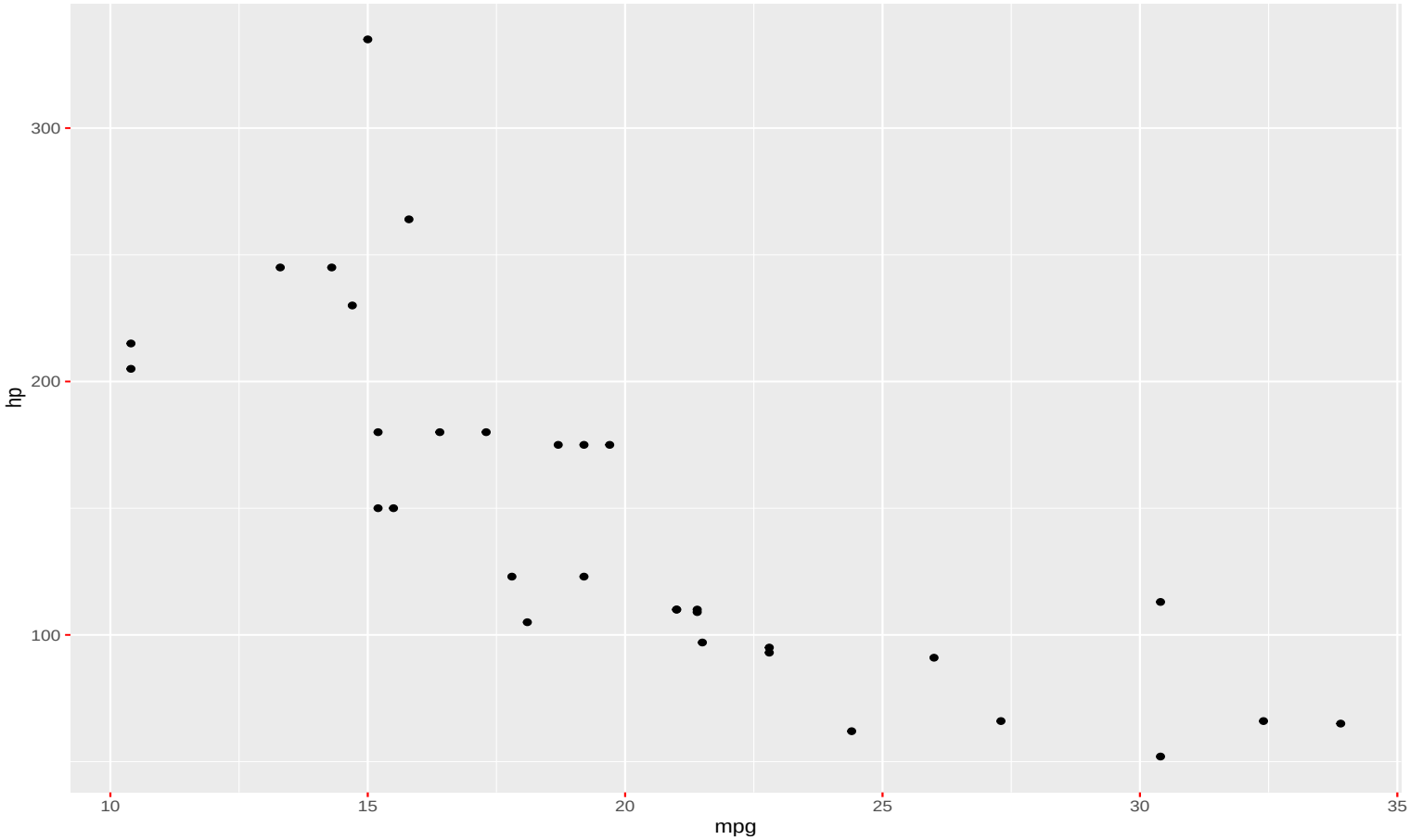
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme(axis.ticks.x = element_blank())
```



ggplot2: guides and themes

Controlling line elements

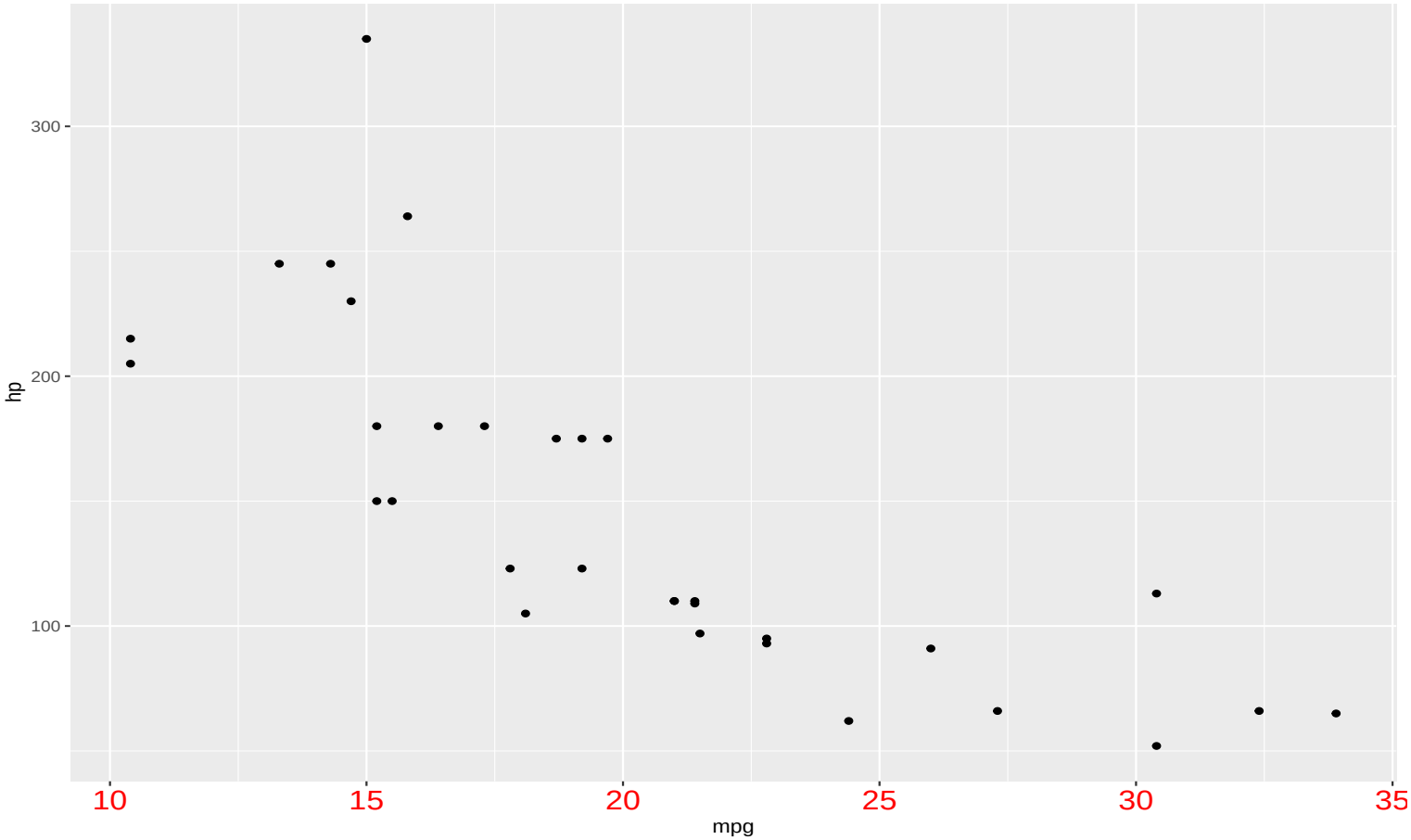
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme(axis.ticks = element_line(colour = 'red'))
```



ggplot2: guides and themes

controlling element_text()

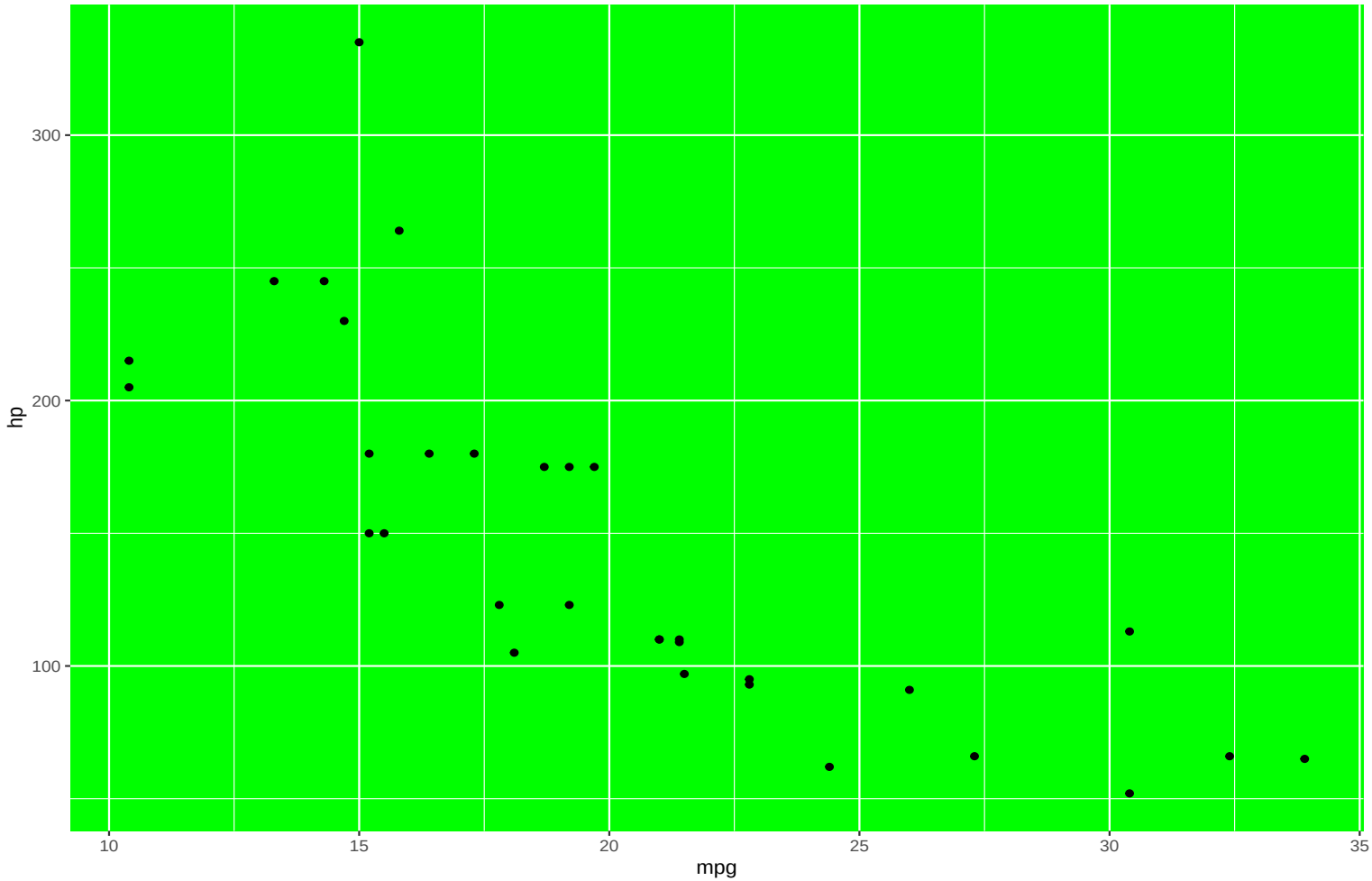
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme(axis.text.x = element_text(colour = 'red', size = 16))
```



ggplot2: guides and themes

Change background: element_rect()

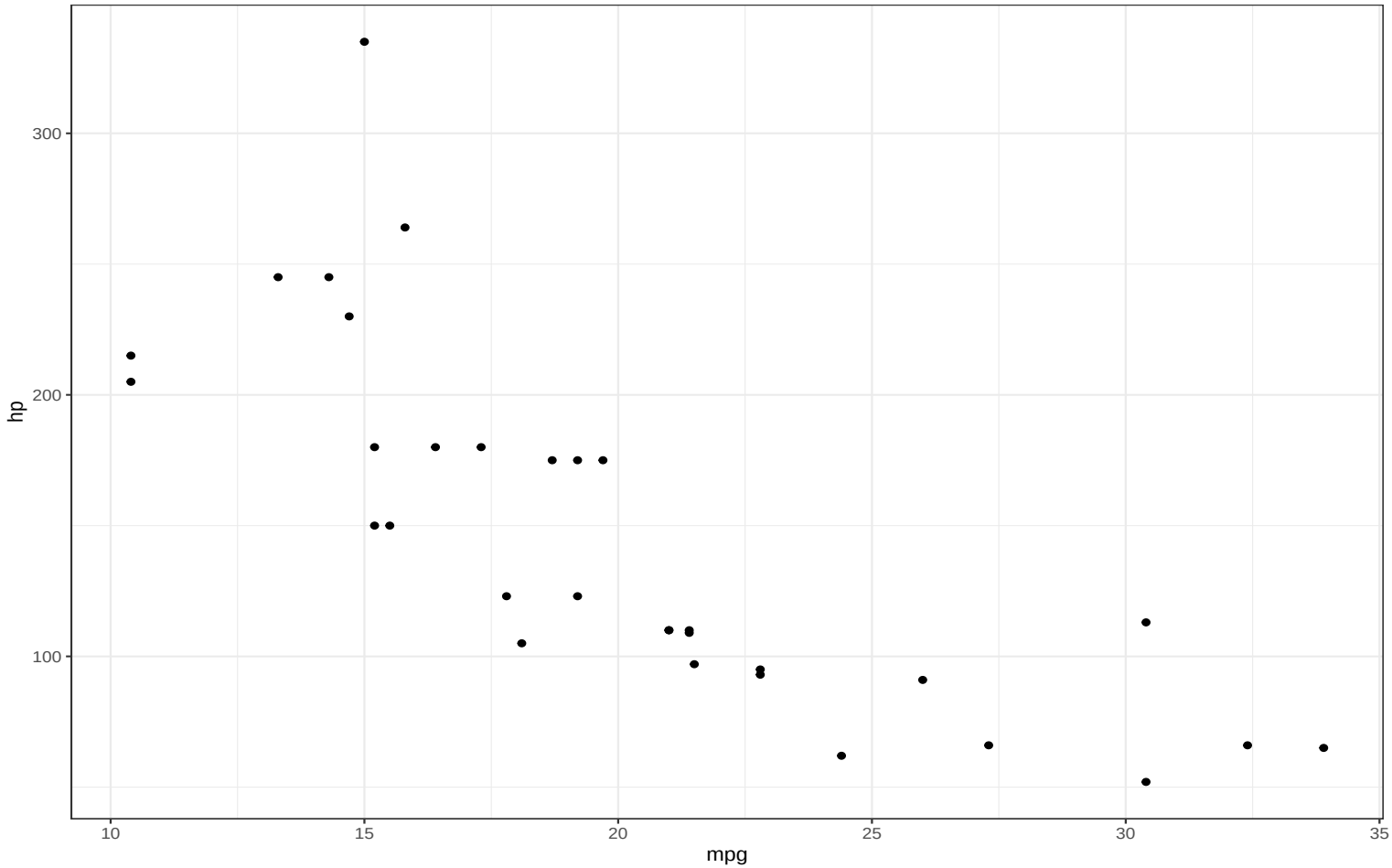
```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme(panel.background  = element_rect(fill = 'green'))
```



ggplot2: guides and themes

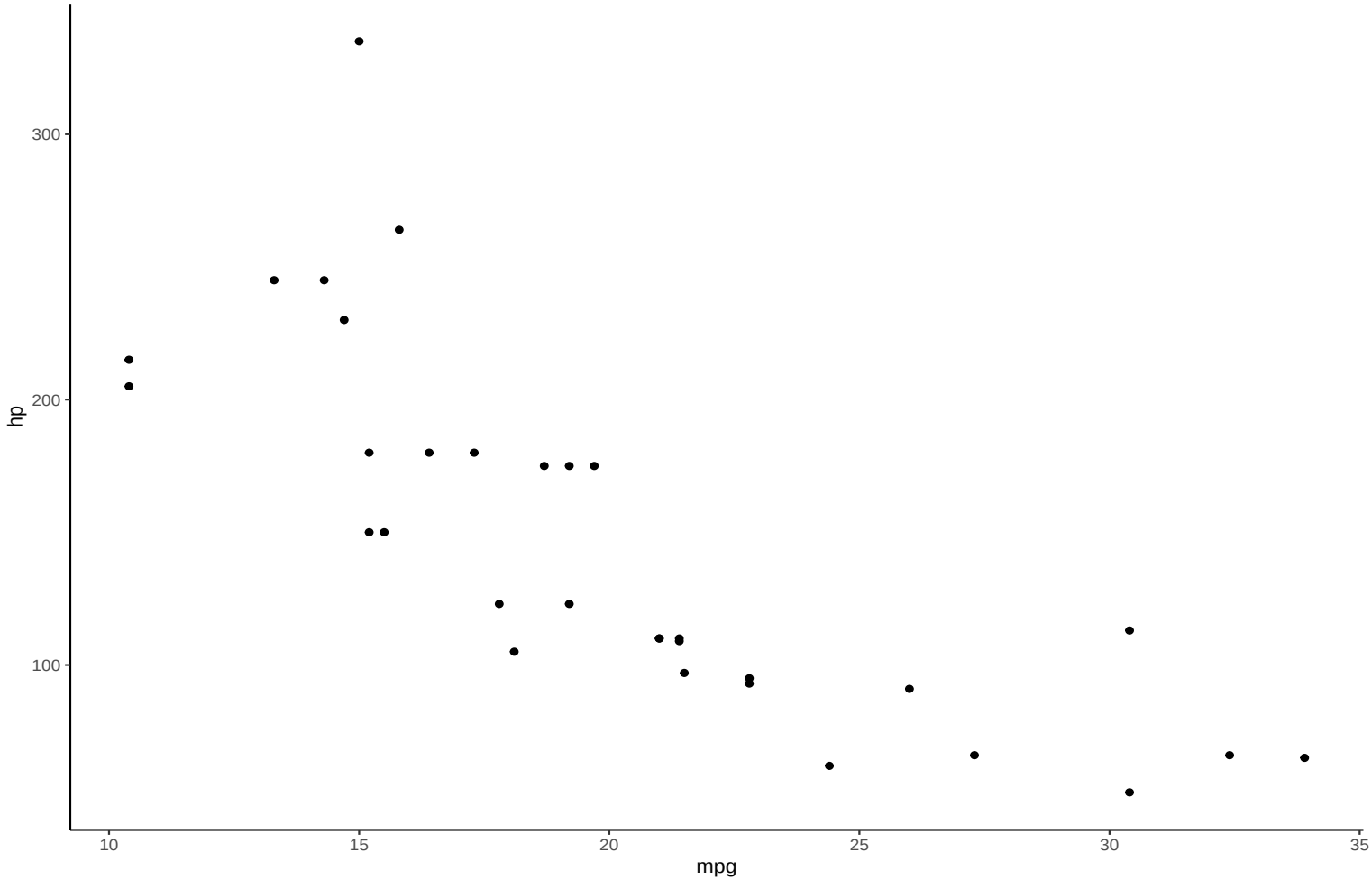
You can use predefined themes

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



ggplot2: guides and themes

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



ggplot2: guides and themes

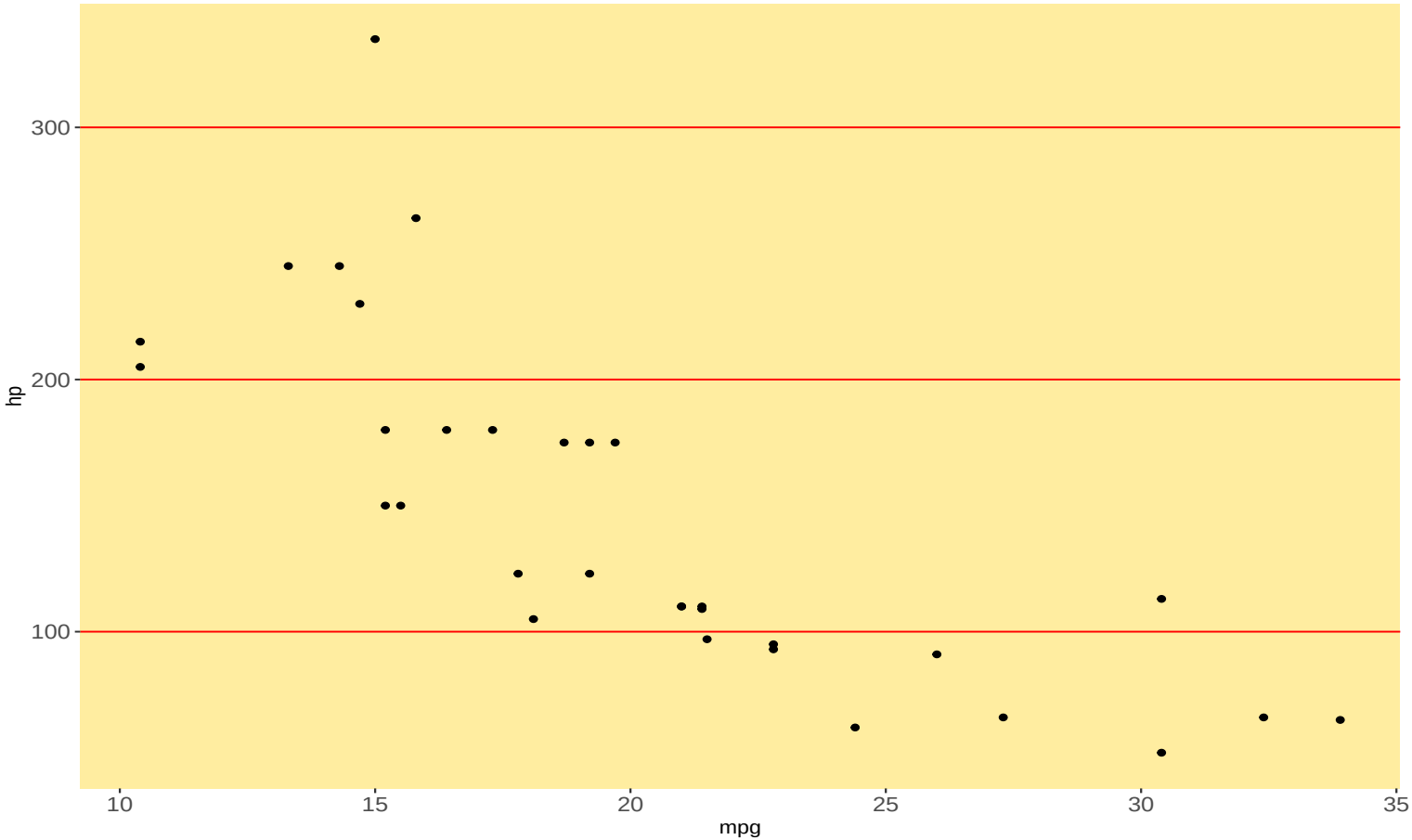
You can also define your own theme, than add it to the plot.

```
theme_my <- theme(
  panel.background = element_rect(fill = '#ffeda0'),
  panel.grid.major.x  = element_blank(),
  panel.grid.major.y = element_line(color = 'red'),
  panel.grid.minor = element_blank(),
  axis.text = element_text(size = 12)
  )
```

ggplot2: guides and themes

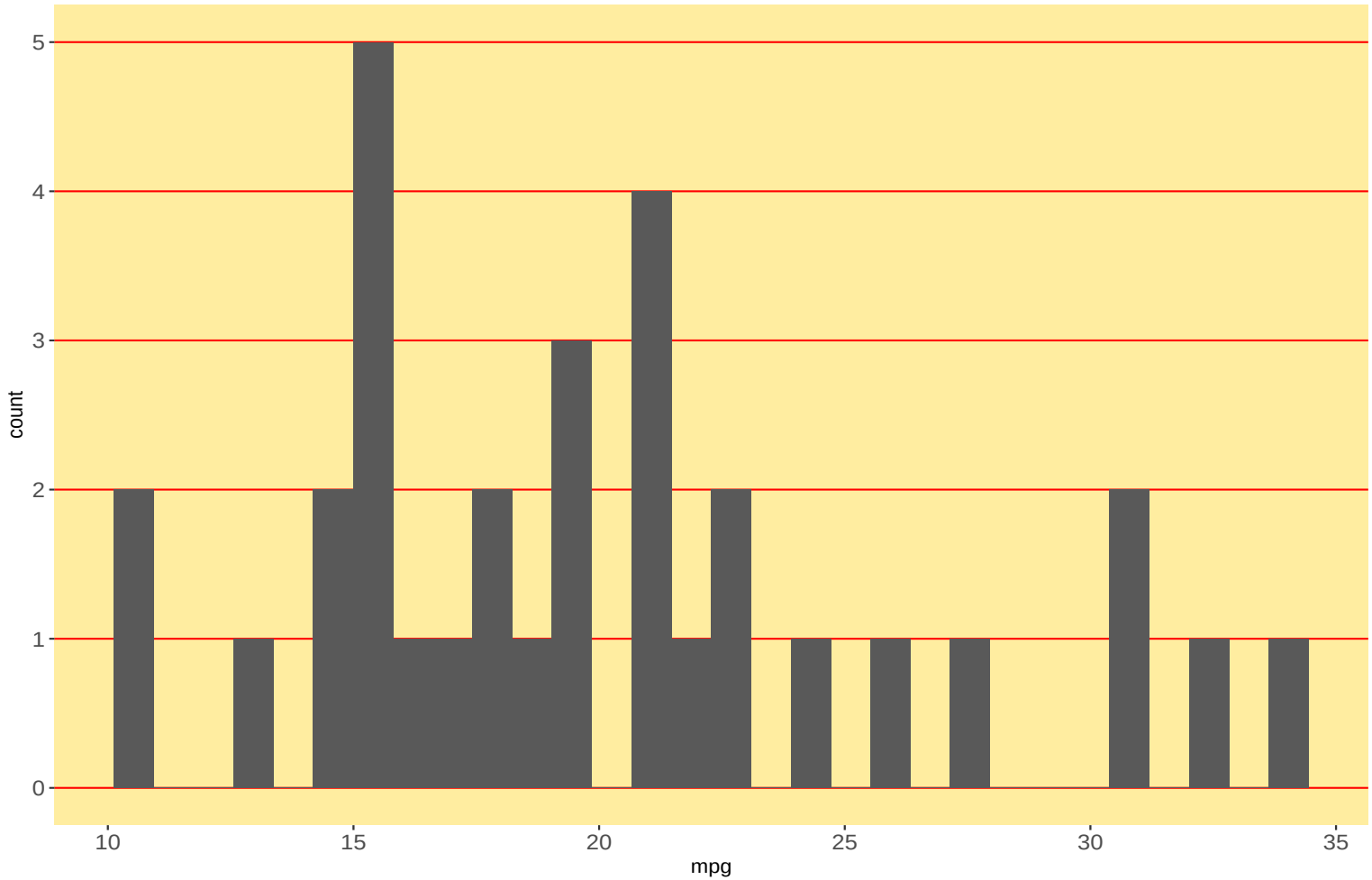
Than add it to the plot

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



ggplot2: guides and themes

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram() +
  theme_my
```



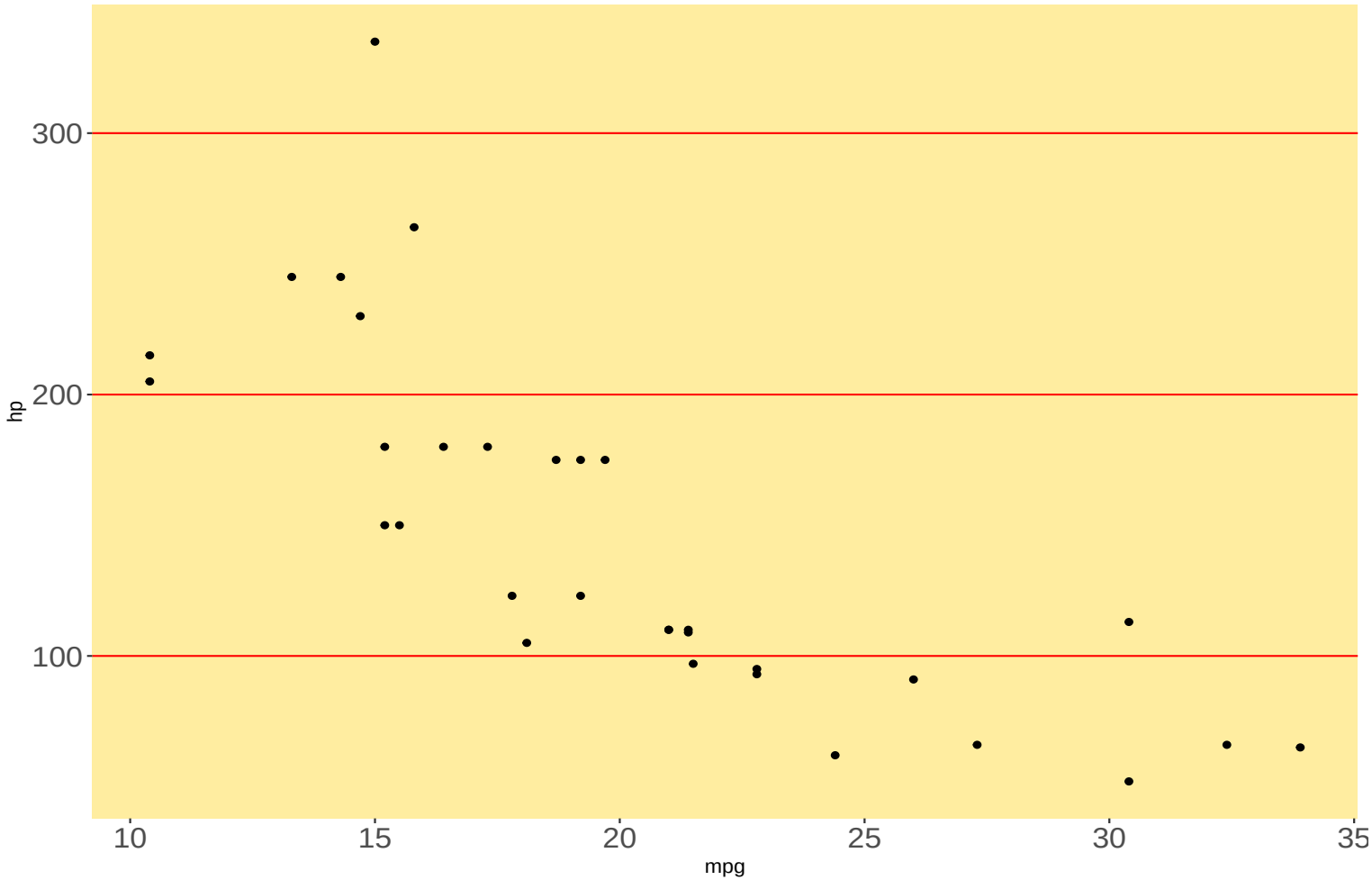
ggplot2: guides and themes

Or you can make it as a function as well

```
theme_my <- function(base_size = 12)
  {
  theme(panel.background = element_rect(fill = '#ffeda0'),
        panel.grid.major.x  = element_blank(),
        panel.grid.major.y = element_line(color = 'red'),
        panel.grid.minor = element_blank(),
        axis.text = element_text(size = base_size))
  }
```

ggplot2: guides and themes

```
ggplot(mtcars, aes(x = mpg, y = hp)) + geom_point() +
  theme_my(base_size = 16)
```



ggplot2: arrange in grid

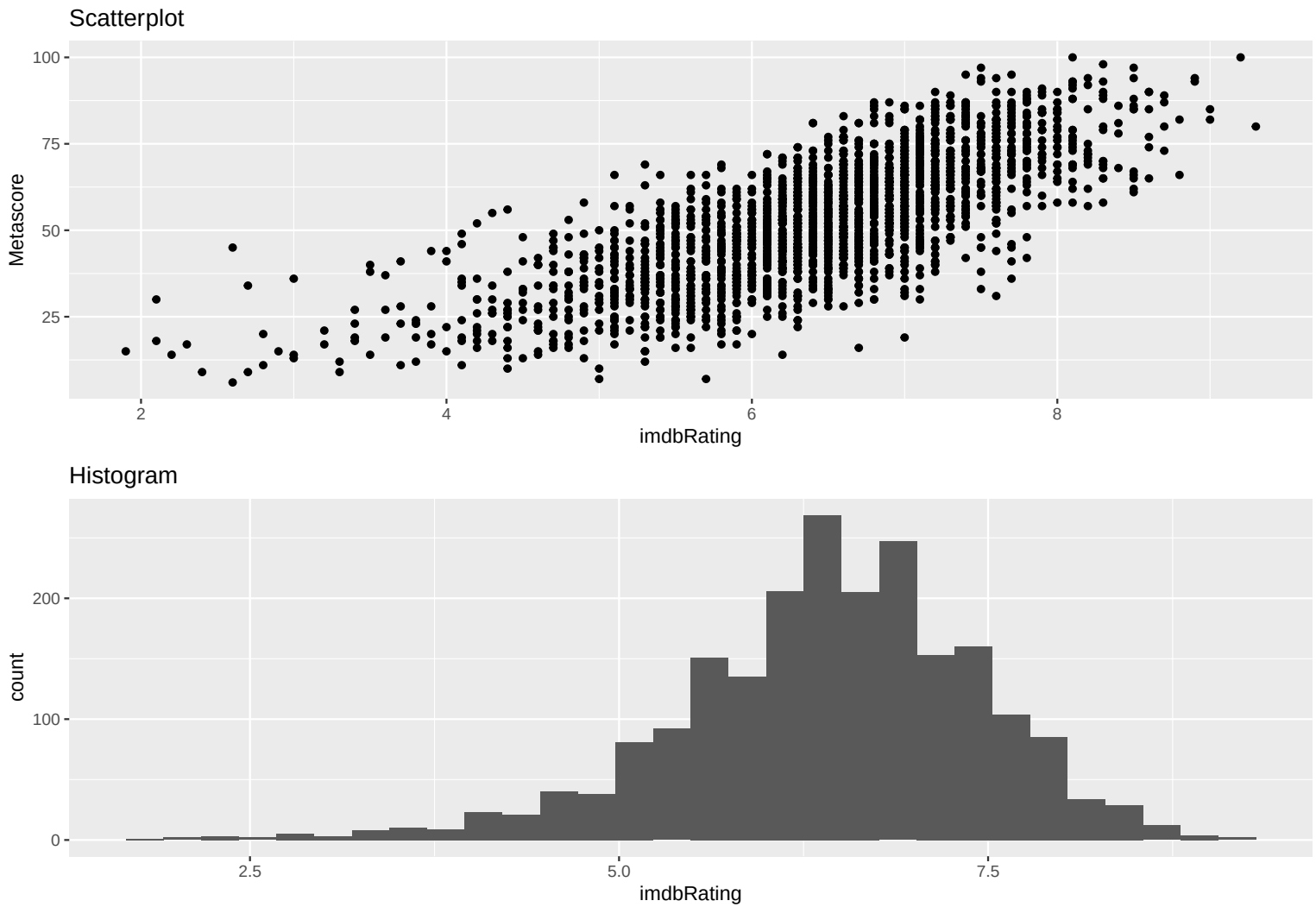
Sometimes you have multiple plots on the same page. You do it using the library gridExtra

```
library(gridExtra)
```

```
p1 <- ggplot(movies_small, aes(x = imdbRating, y = Metascore)) +
      geom_point() + ggtitle('Scatterplot')
p2 <- ggplot(movies_small, aes(x = imdbRating)) +
      geom_histogram() + ggtitle('Histogram')
```

ggplot2: arrange in grid

```
grid.arrange(p1, p2)
```



ggplot2: arrange in grid

Arrange by columns

```
grid.arrange(p1, p2, nrow=1)
```