Data Visualization with R

Armand Tossou

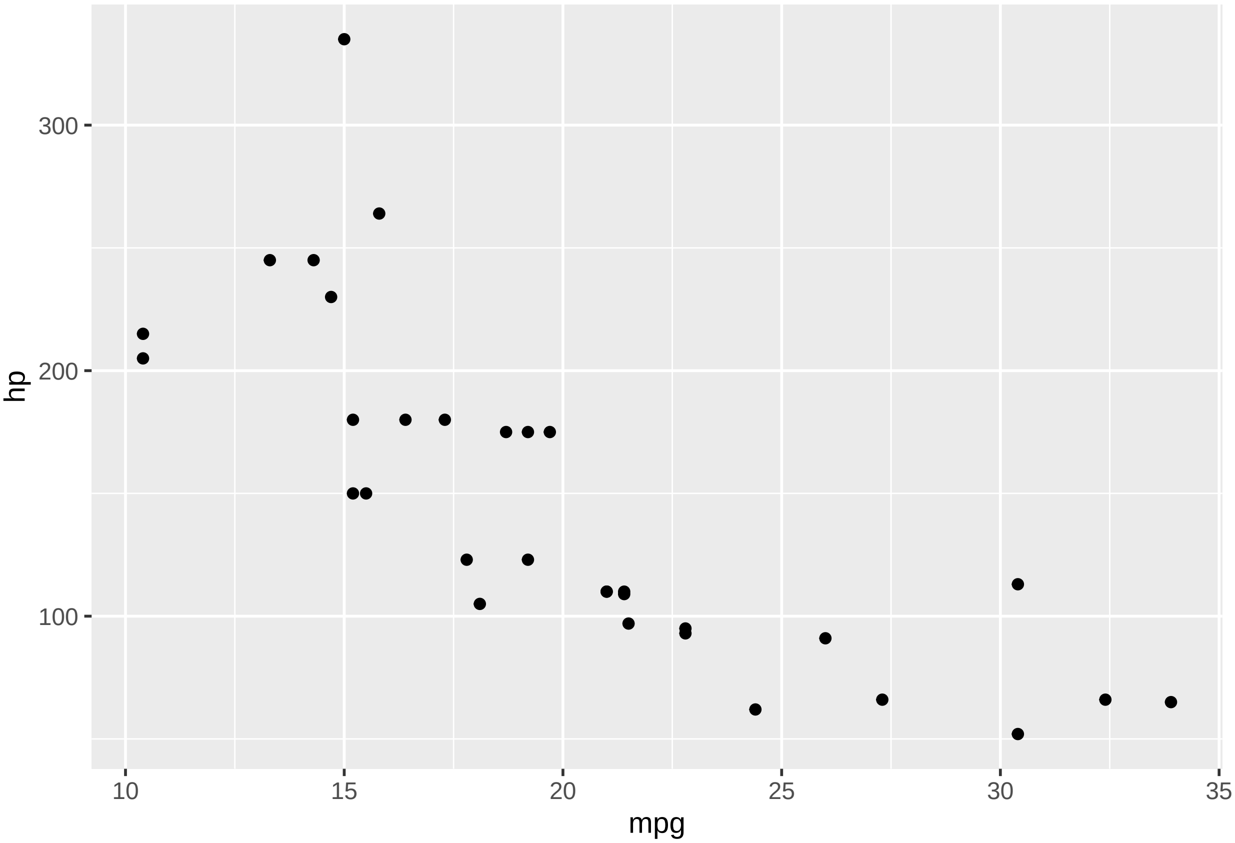
9/6/2021

Grammar of Graphics with ggplot2

Each `ggplot2` plot has 3 basic layers: - a data layer - an aesthetics layer - and a geometries layer

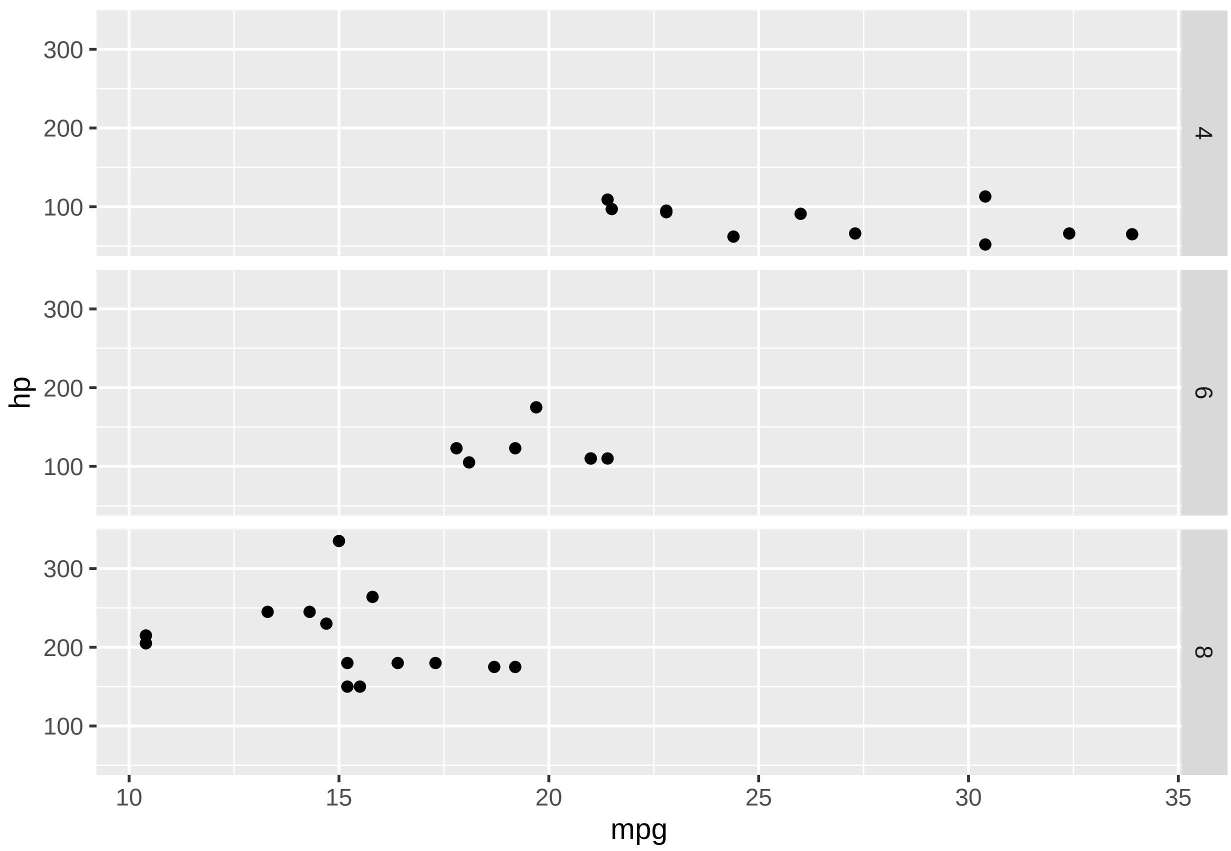
Additional layers include: - facets: to put multiple plots on the same canvas - statistics: - coordinates - and a theme

```
#load the package
library(ggplot2)

# create a scatterplot
pl <- ggplot(data=mtcars,aes(x=mpg,y=hp))
pl + geom_point()
```



Adding facets:

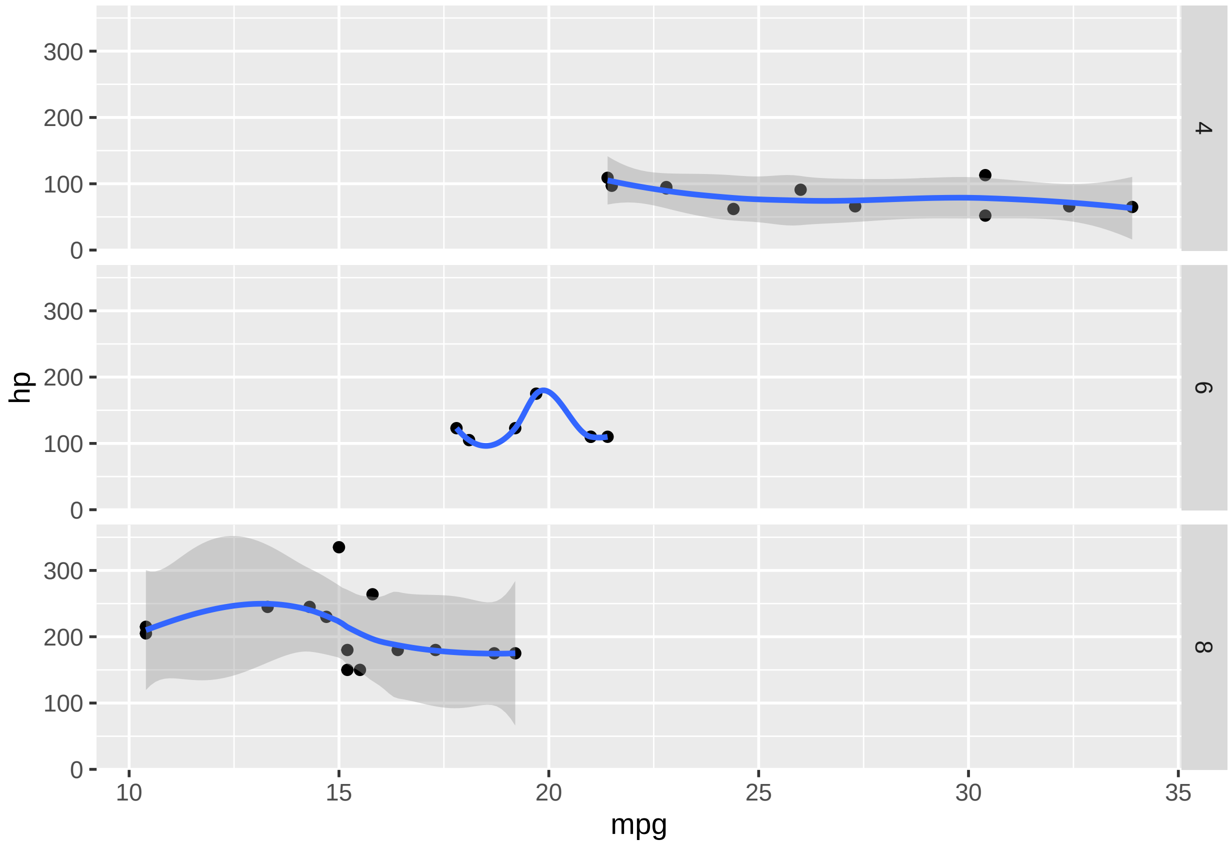
```
# create a scatterplot for each cylendar type
pl <- ggplot(data=mtcars,aes(x=mpg,y=hp)) + geom_point()
pl + facet_grid(cyl ~ .)
```



Let’s add a statistics layer:

```
# create a scatterplot for each cylendar type, adding a smoothed line of fit
pl <- ggplot(data=mtcars,aes(x=mpg,y=hp)) + geom_point()
pl + facet_grid(cyl ~ .) + stat_smooth()
```

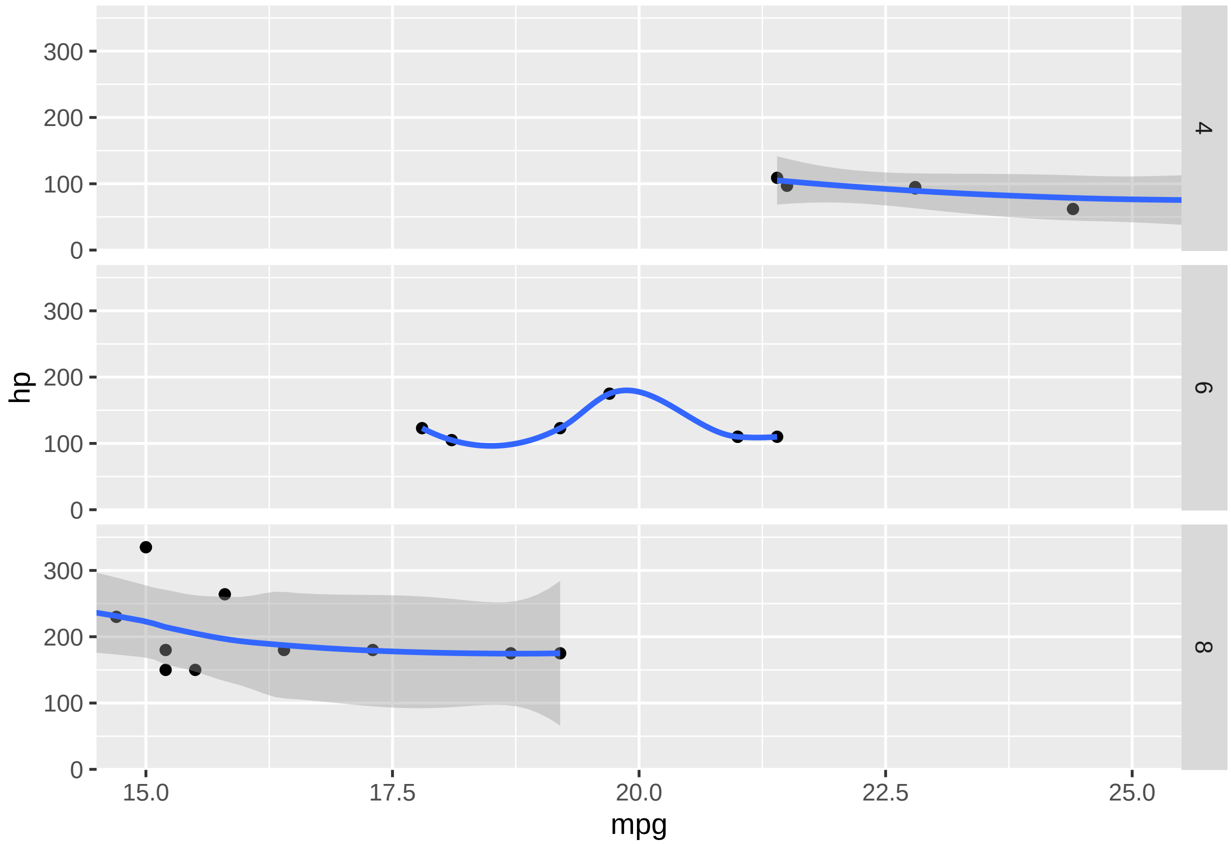
```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



Next, let’s add coordinates to our plot.

```
# create a scatterplot for each cylendar type, adding a smoothed line of fit, and adding coordinates
pl <- ggplot(data=mtcars,aes(x=mpg,y=hp)) + geom_point()
pl2 <- pl + facet_grid(cyl ~ .) + stat_smooth()
pl2 + coord_cartesian(xlim = c(15,25))
```

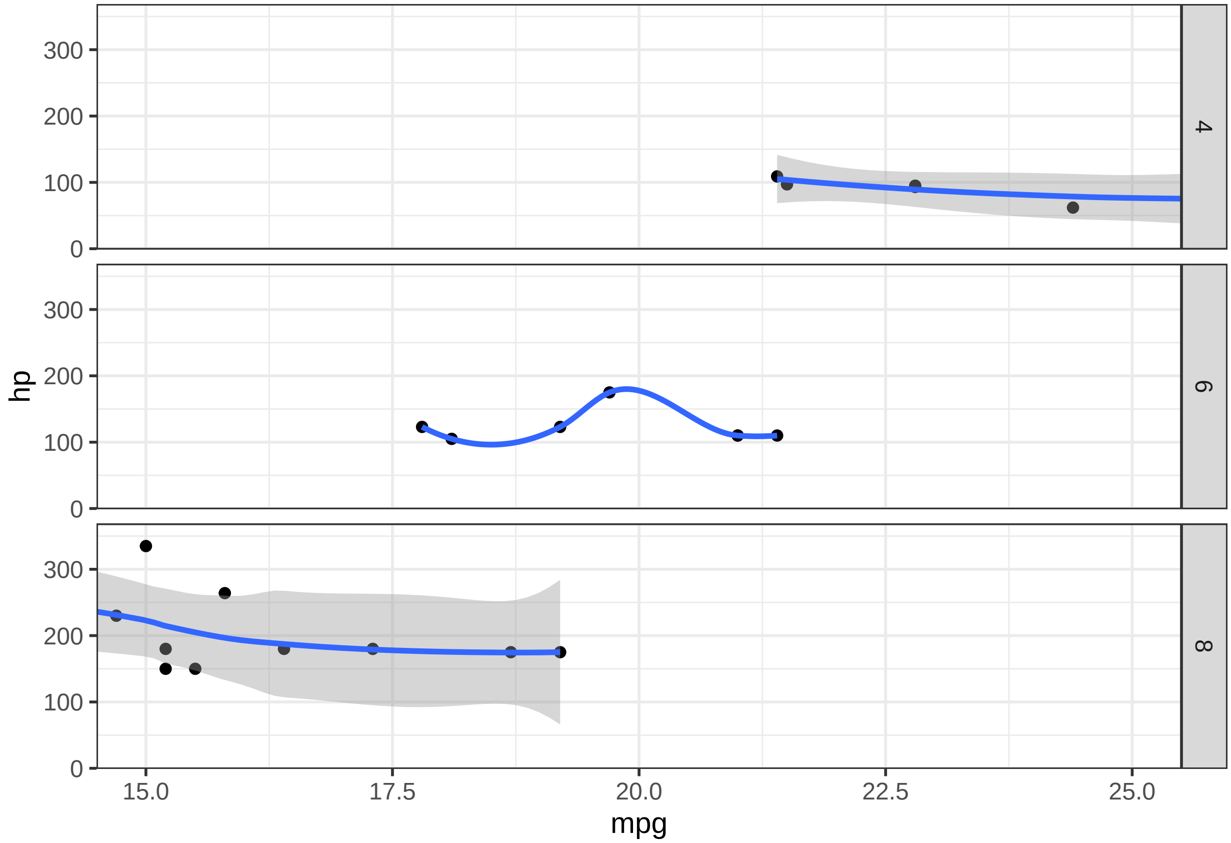
```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



Finally, let's add a theme to our plot.

```
# create a scatterplot for each cylendar type, adding a smoothed line of fit,
# and adding coordinates, and adding a theme
pl <- ggplot(data=mtcars,aes(x=mpg,y=hp)) + geom_point()
pl2 <- pl + facet_grid(cyl ~ .) + stat_smooth()
pl2 + coord_cartesian(xlim = c(15,25)) + theme_bw()
```

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



## Histograms

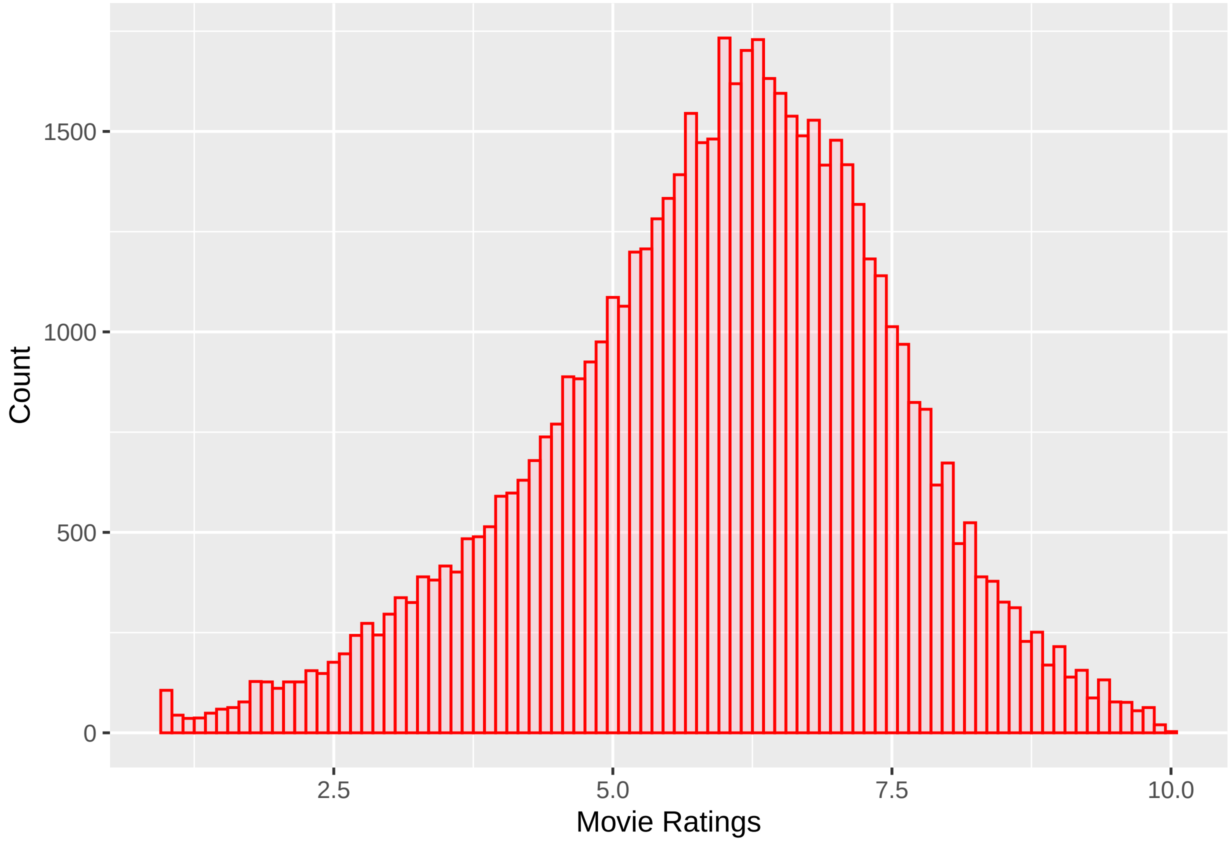
RStudio ggplot cheat sheet: https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf

```
# install.packages("ggplot2")
library(ggplot2)

# install the dataset we'll be working with
#install.packages("ggplot2movies")
library(ggplot2movies)

# show the columns of the 'movies' dataset
colnames(movies)
```

```
##  [1] "title"       "year"        "length"      "budget"      "rating"
##  [6] "votes"       "r1"          "r2"          "r3"          "r4"
## [11] "r5"          "r6"          "r7"          "r8"          "r9"
## [16] "r10"         "mpaa"        "Action"      "Animation"   "Comedy"
## [21] "Drama"       "Documentary" "Romance"     "Short"
```

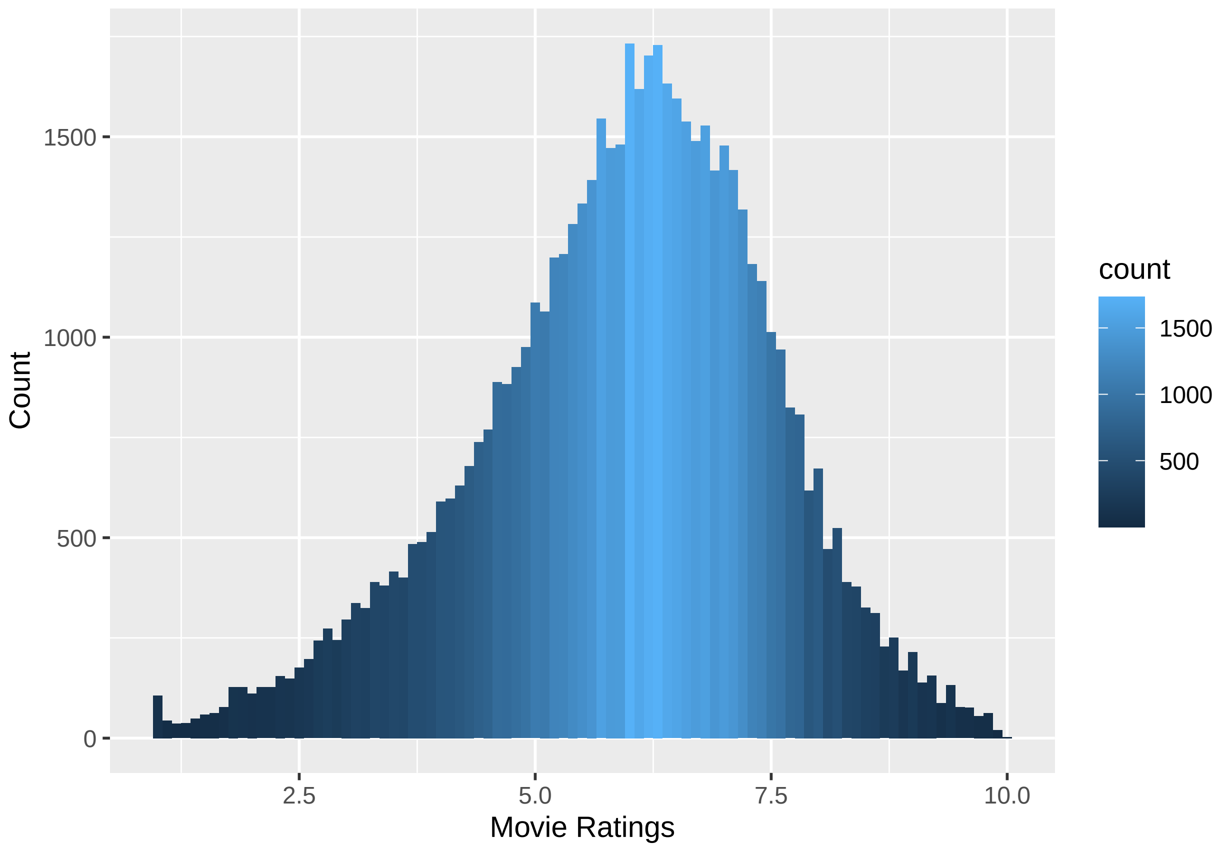
Start out by plotting a basic histogram:

```
# data & aesthetics
pl <- ggplot(movies,aes(x=rating))

# geometry
pl2 <- pl + geom_histogram(binwidth = 0.1, color='red',fill='pink',alpha=0.4)

# add labels
pl3 <- pl2 + xlab('Movie Ratings') + ylab('Count')

print(pl3)
```



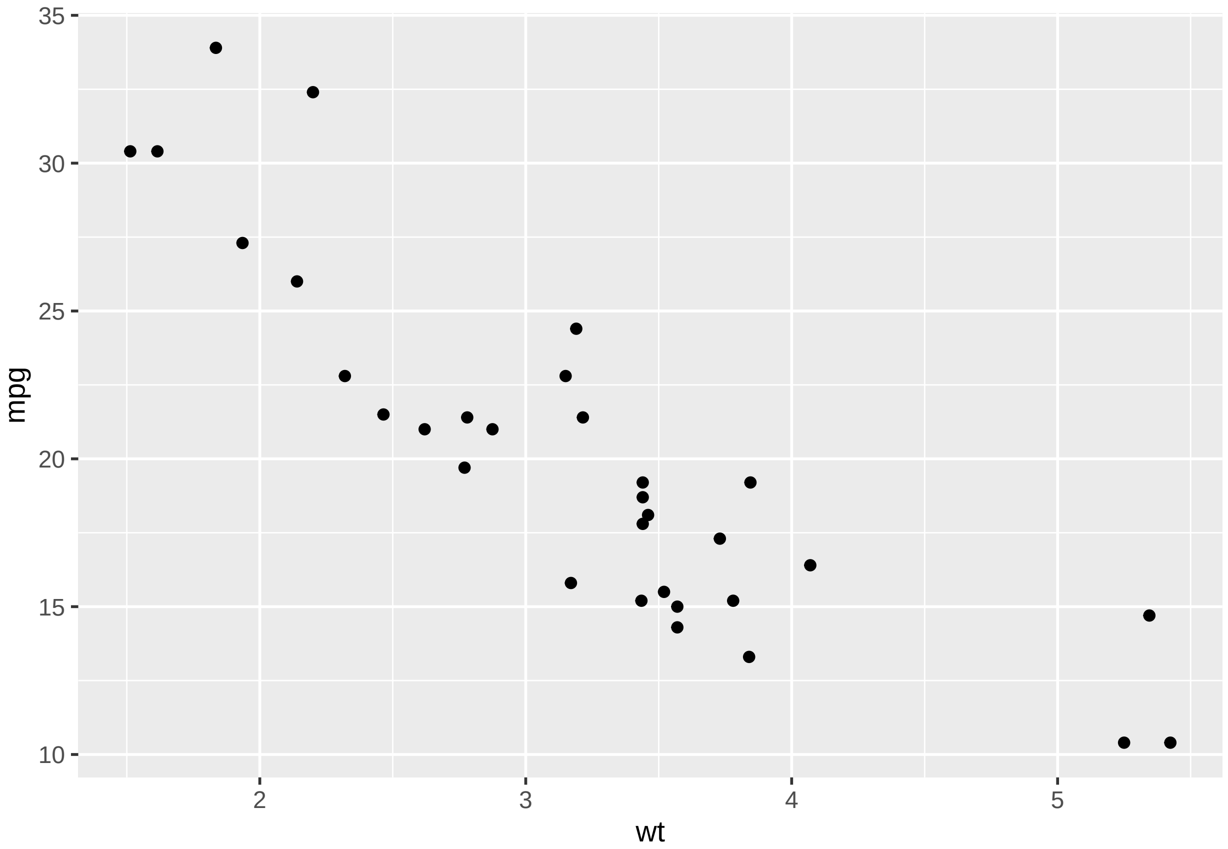
There are also some advanced options that we can apply to the geometry layer.

```
pl <- ggplot(movies,aes(x=rating))

pl2 <- pl + geom_histogram(binwidth = 0.1,aes(fill=..count..))

pl3 <- pl2 + xlab('Movie Ratings') + ylab('Count')

print(pl3)
```



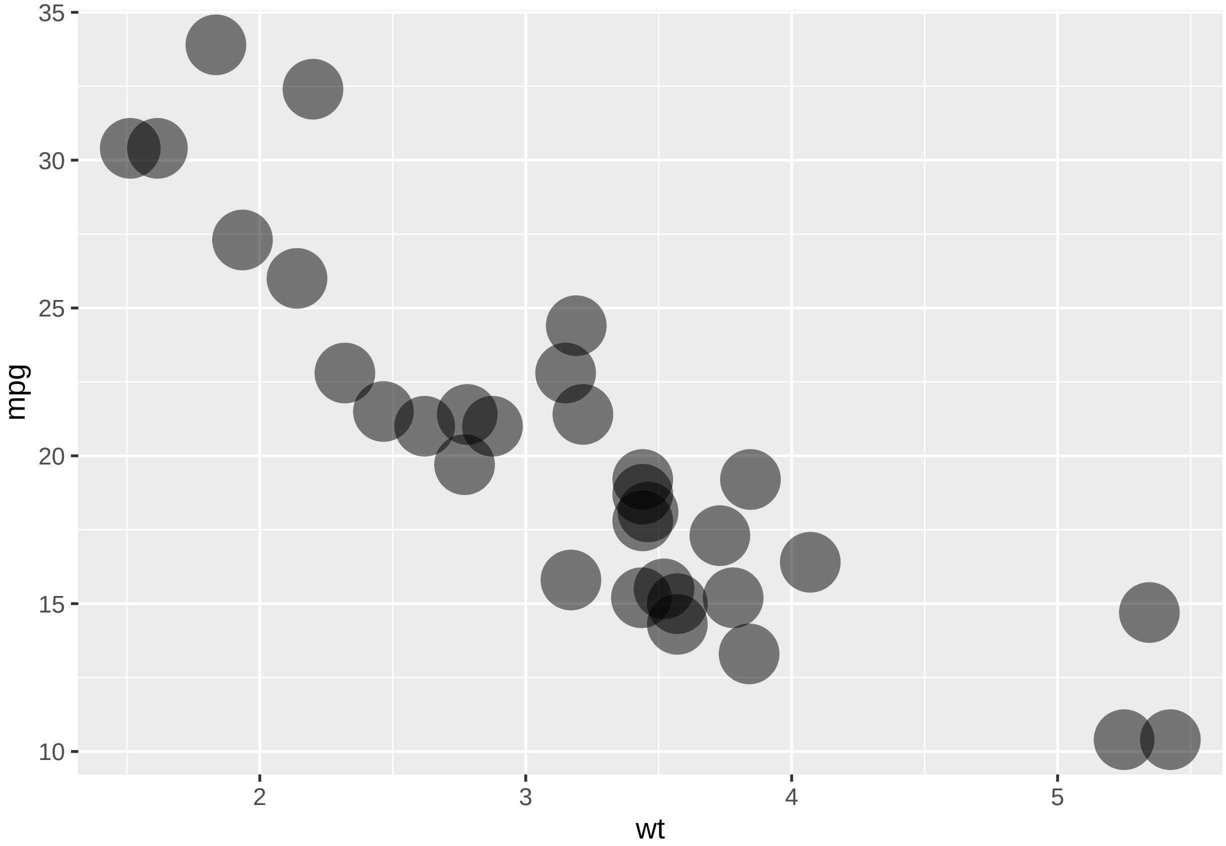
## Scatterplots

```
# we'll use the built-in 'mtcars' dataframe to plot mile per galon by car weight
df <- mtcars

# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
print(pl + geom_point())
```



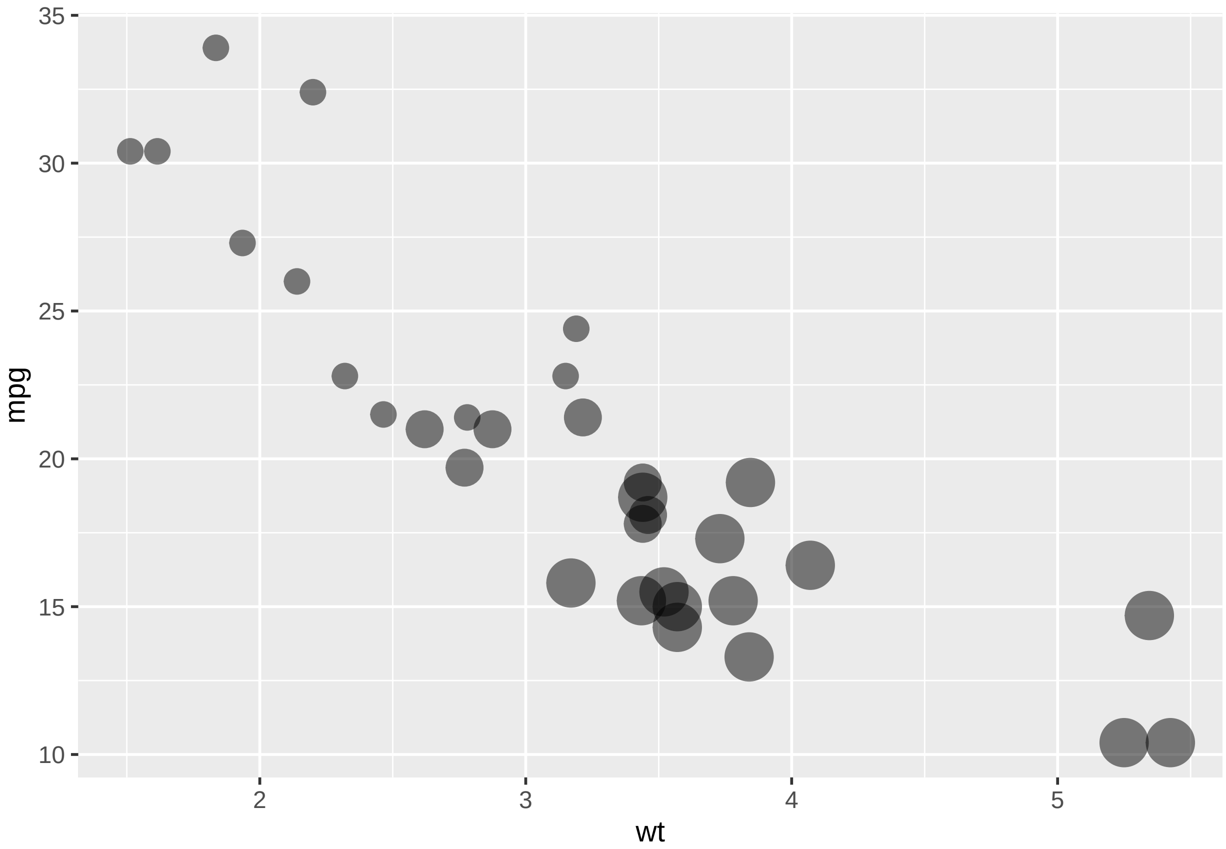
Now let’s improve upon our scatterplot by playing around with: - the size of points - transparency

```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(size=10,alpha=0.5)

print(pl2)
```



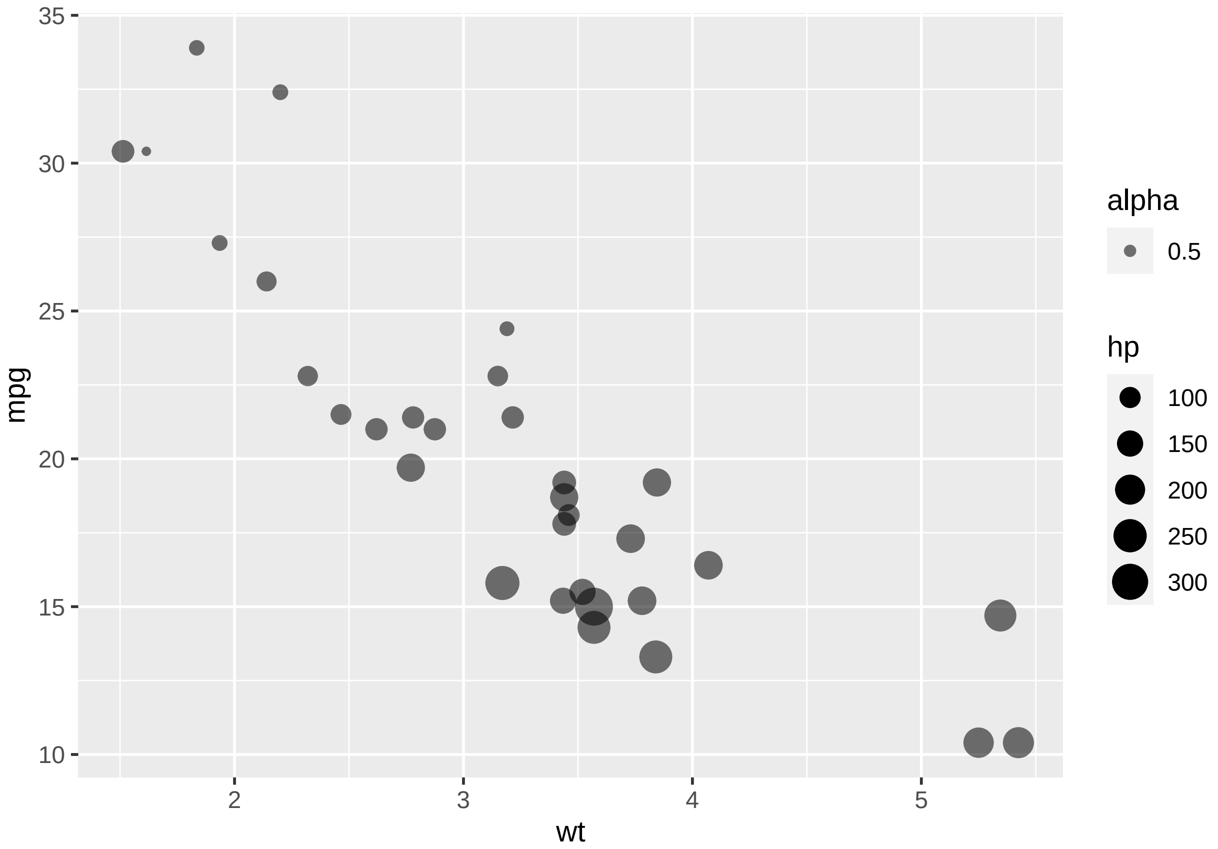
We can size points in our scartterplot with other features in the dataset:

```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(size=mtcars$cyl,alpha=0.5) # sizing points based on the number of car 'cylinders

print(pl2)
```



Pass the aesthetics parameters directly inside the geometries:

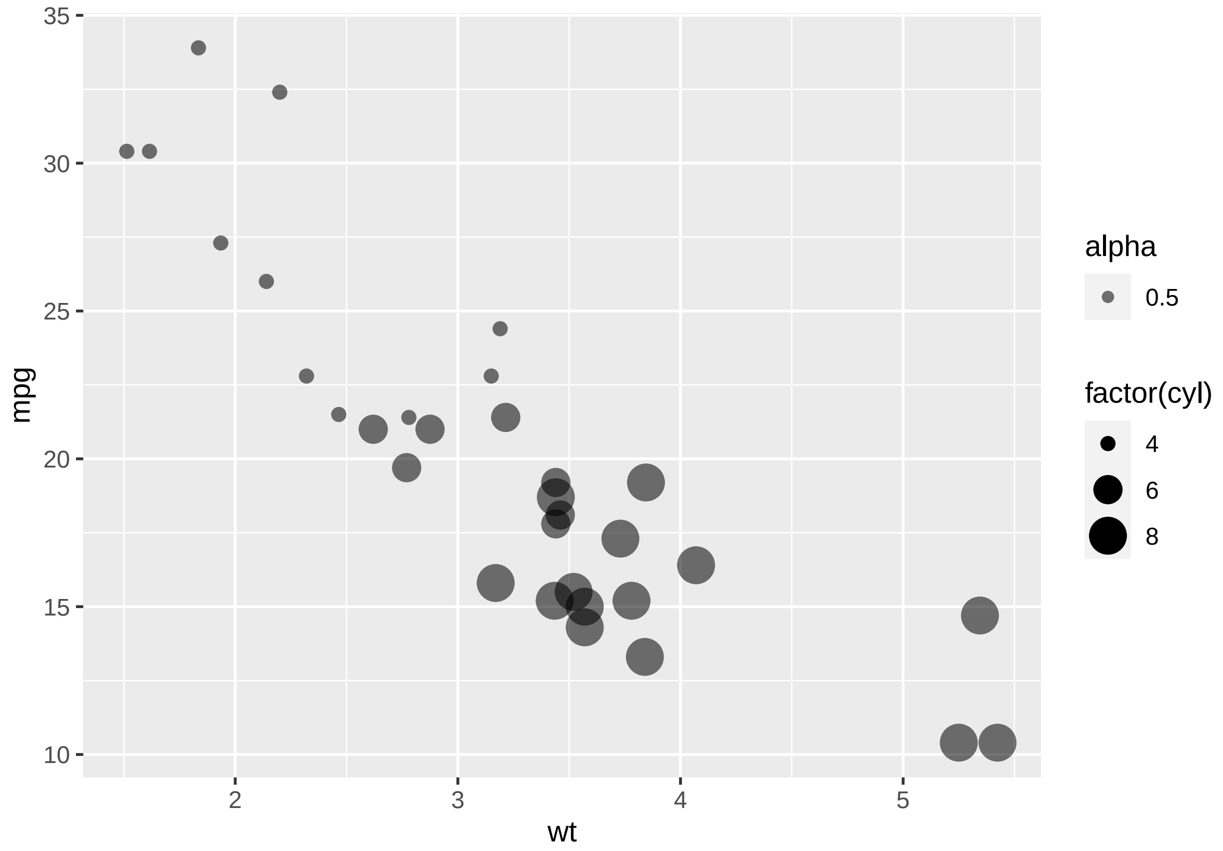
```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(aes(size=hp,alpha=0.5)) # sizing points based on the number of car 'horsepower'

print(pl2)
```



Let's size up points based on the number of car 'cylinders'. Note how we tell R that the 'cyl' variable is categorical (rather than continuous).

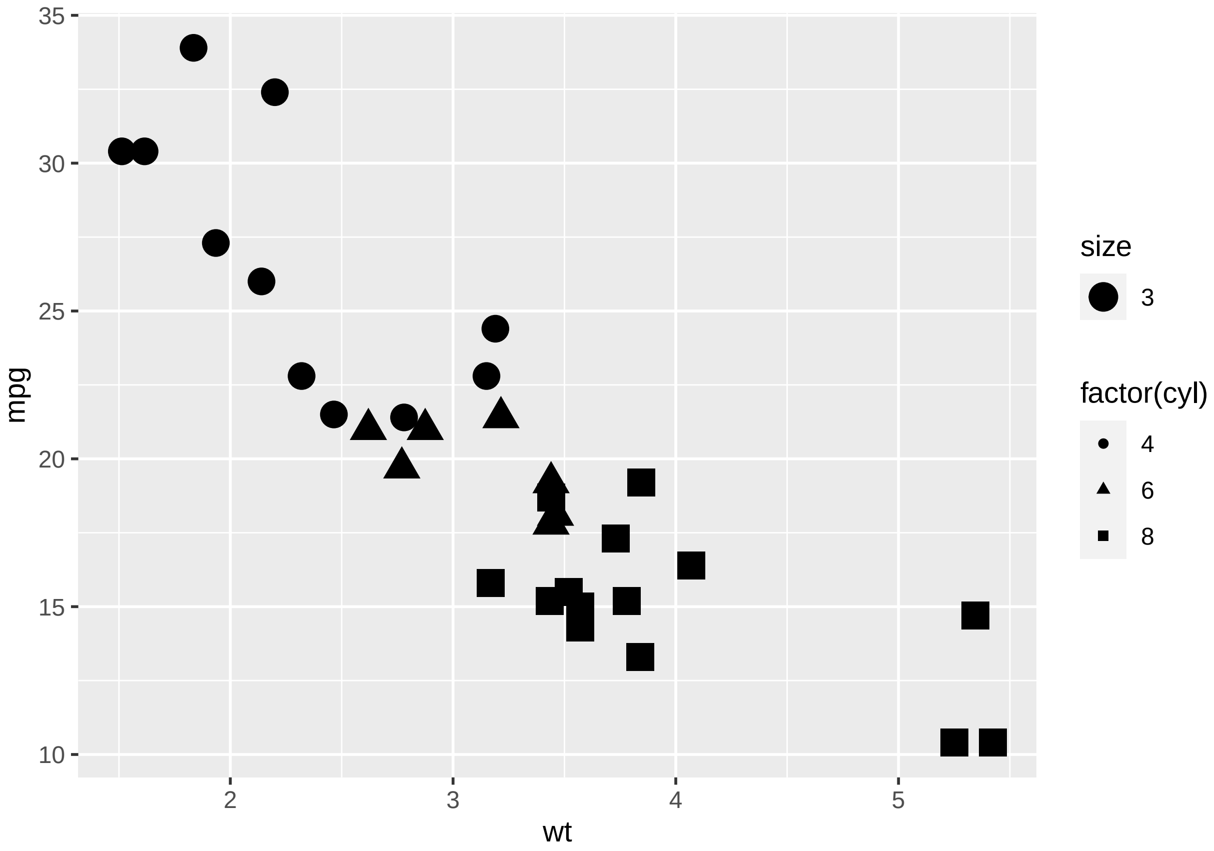
```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(aes(size=factor(cyl),alpha=0.5))

print(pl2)
```

```
## Warning: Using size for a discrete variable is not advised.
```



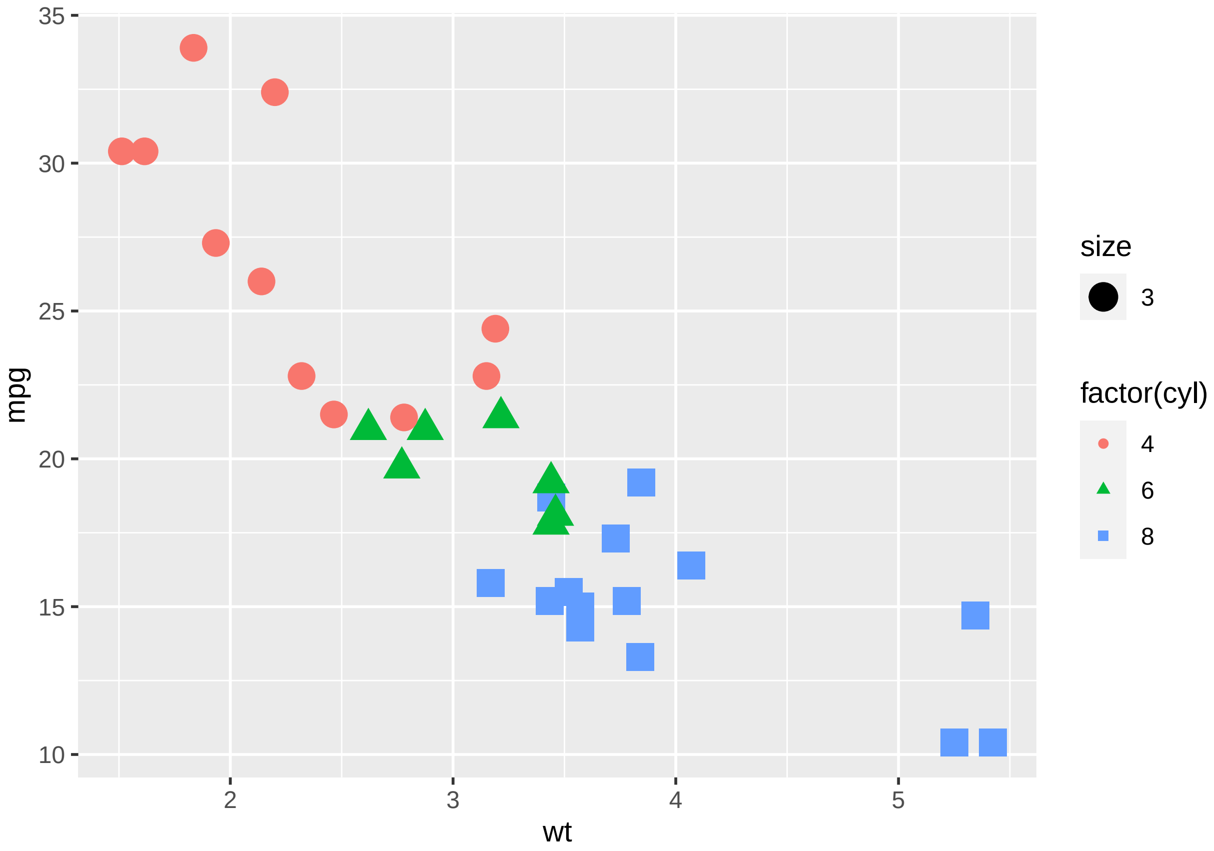
However, it's typically recommended to rather define shapes for categorical variables. In the scatterplot below, we get a different geometric shape for each cylinder type.

```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(aes(shape=factor(cyl),size=3))

print(pl2)
```



Now let's try coloring by cylinder. So we get both shapes and colors based on the levels in the 'cylinder' variable.

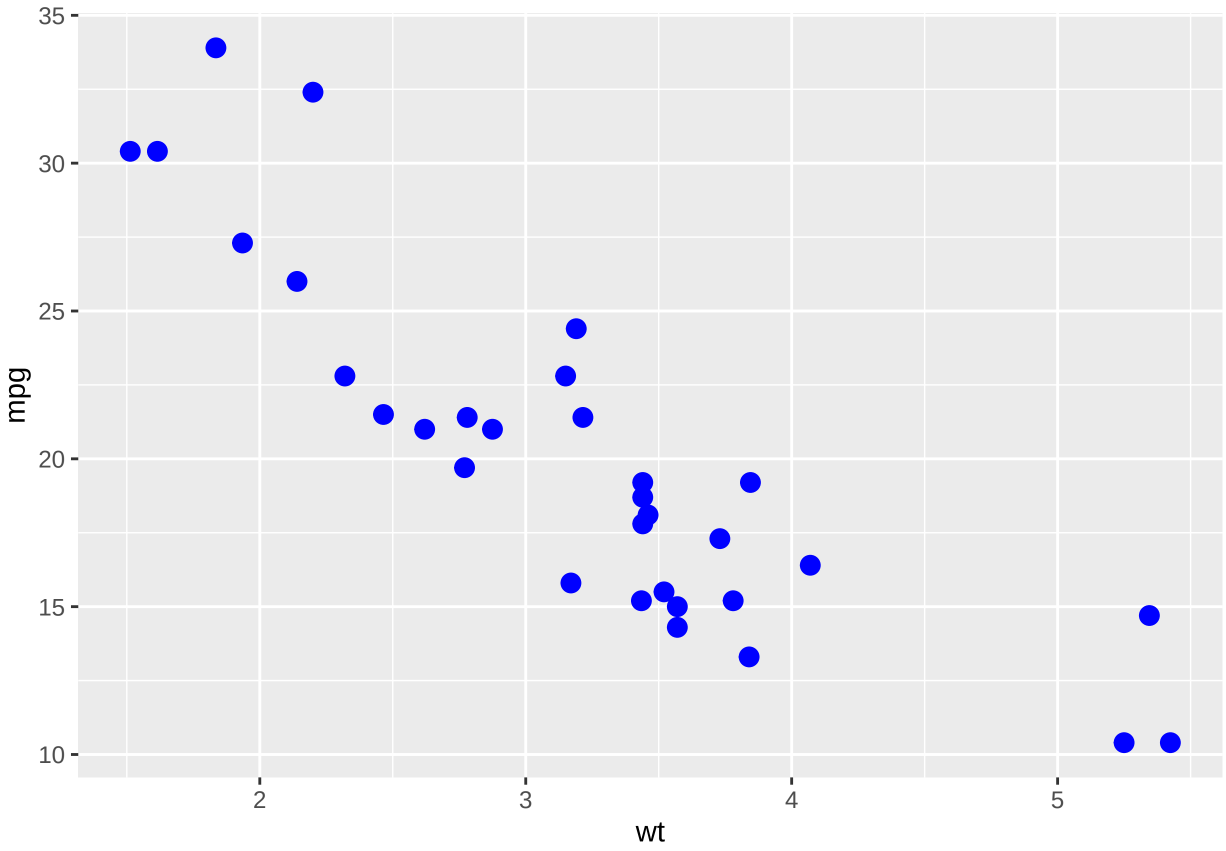
```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(aes(shape=factor(cyl),color=factor(cyl),size=3))

print(pl2)
```



Now let's play around with colors.

First, let's color all points blue.

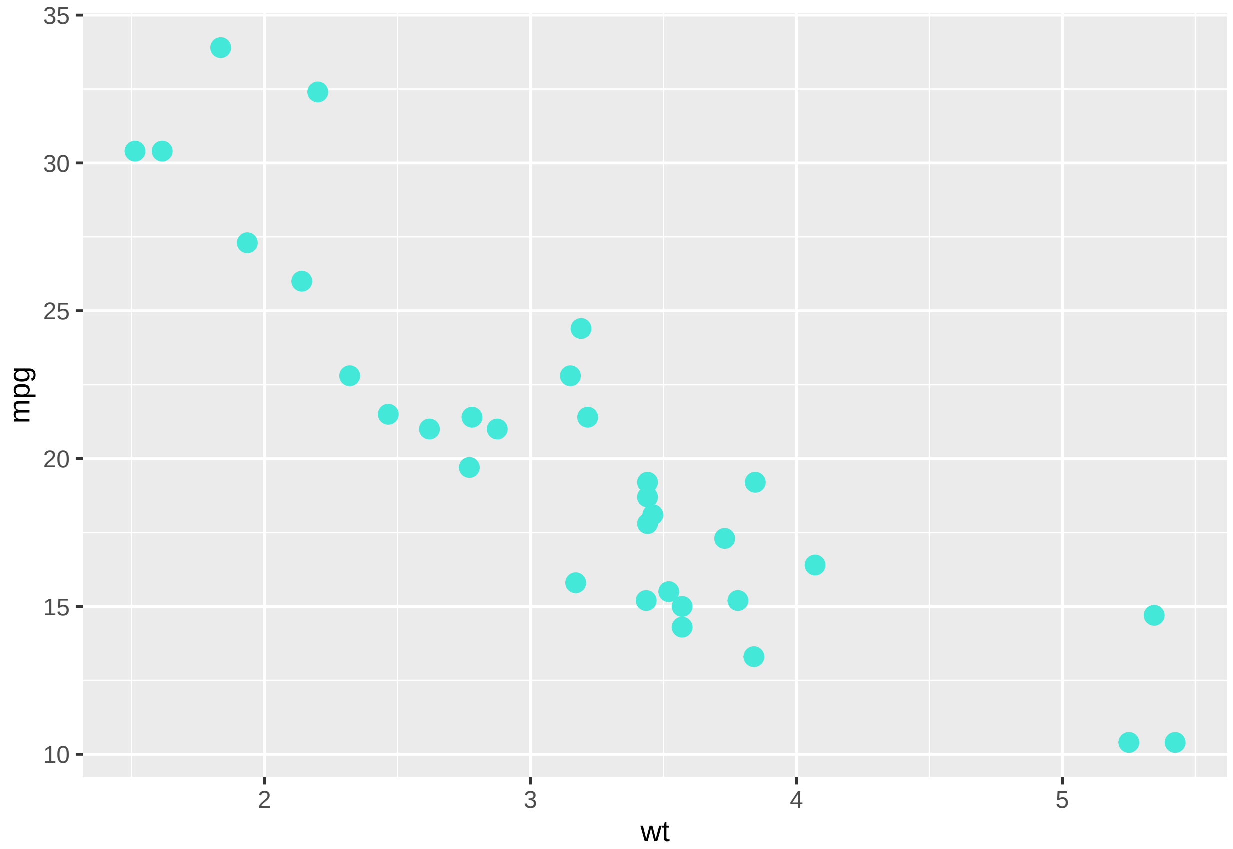
```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(color='blue',size=3)

print(pl2)
```



Let's define colors with HEX color values. This makes it possible to get more specific colors.

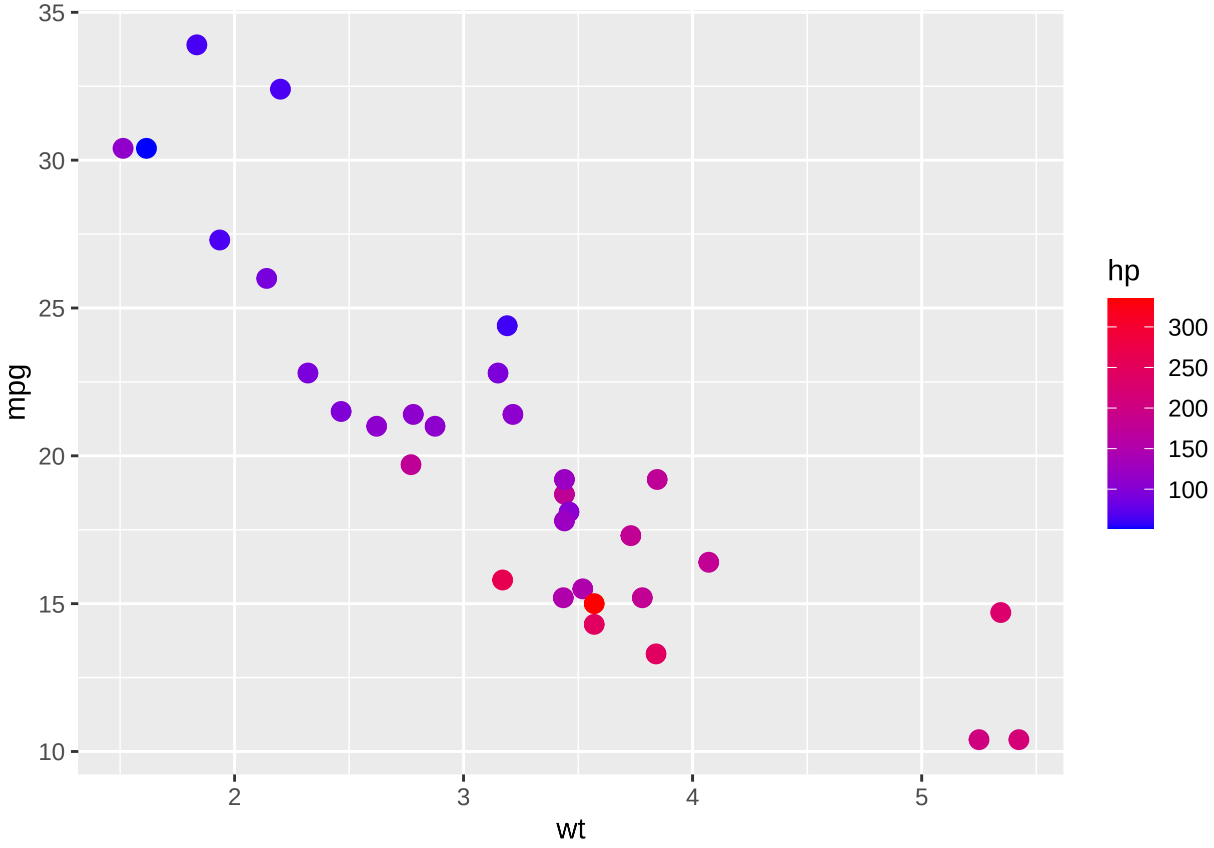
Color picker websites like this one https://www.color-hex.com/ can come in handy.

```
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(color='#43e8d8',size=3) # teal color

print(pl2)
```



Color points by gradient. Here for instance, we can color the scatterplot by 'horsepower', a continuous variable.

```r
# data & aesthetics
pl <- ggplot(df,aes(x=wt,y=mpg))

# geometry
pl2 <- pl + geom_point(aes(color=hp),size=3)

# we can specify the color range
pl3 <- pl2 + scale_color_gradient(low='blue',high='red')

print(pl3)
```

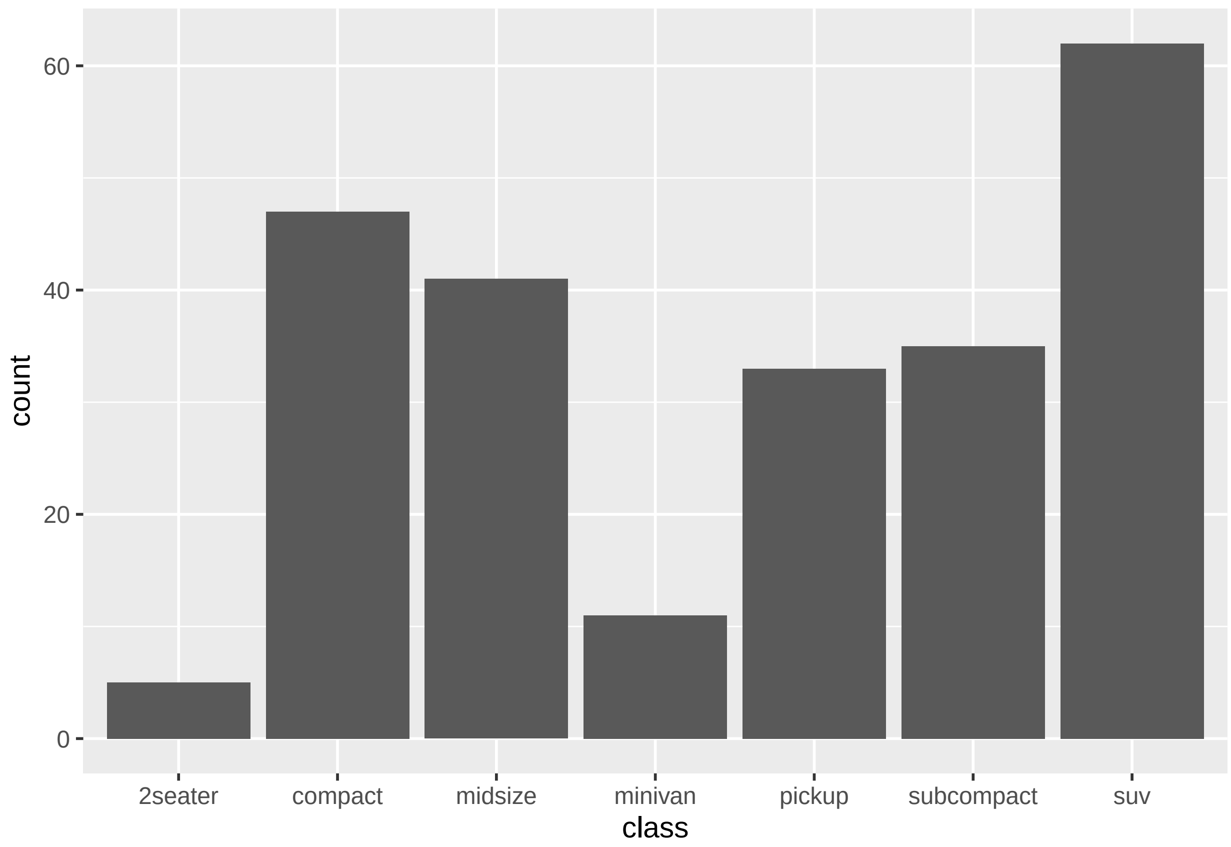


## Barplots

Show a count when we're dealing with categorical data.

Let's categorize the class of vehicles vurses the count of how many times they occur in the dataset.

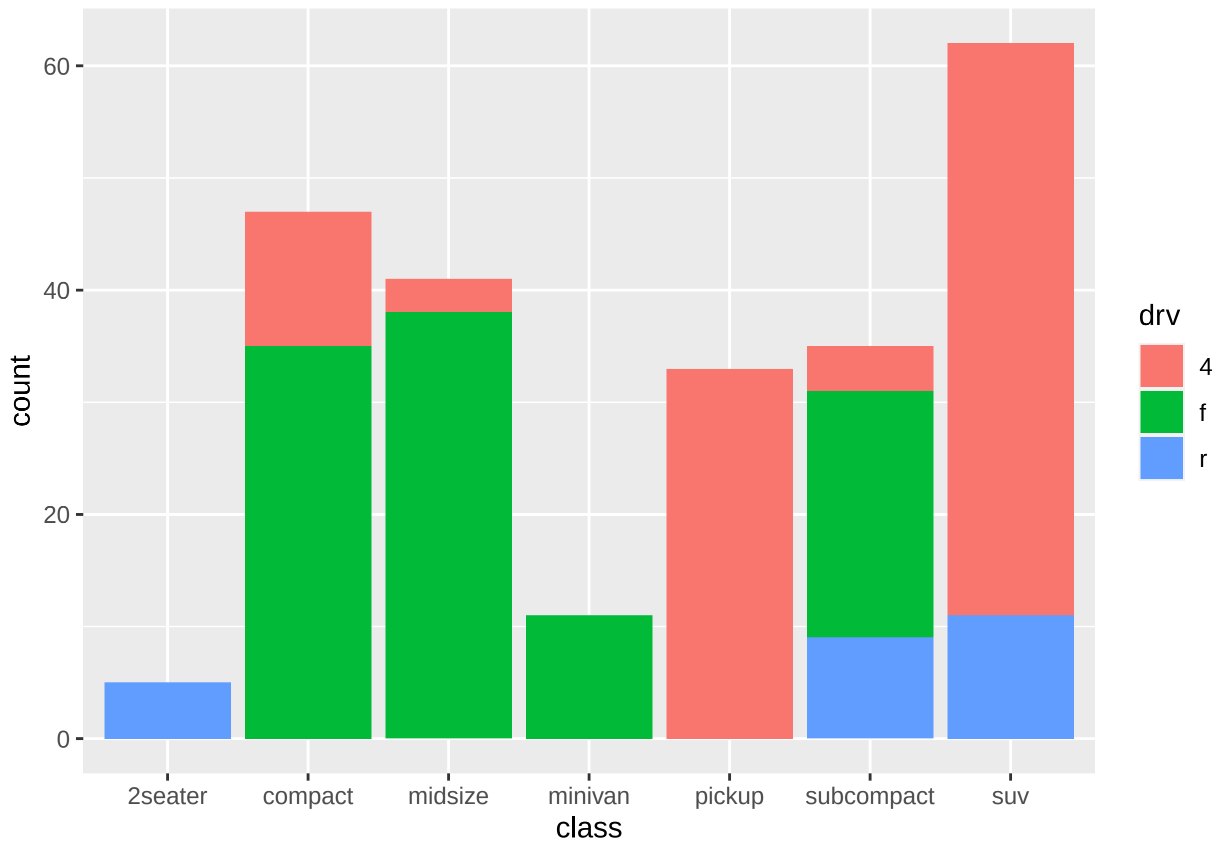
```
# use the 'mpg' data frame that's built-in ggplot2
df <- mpg

# data & aesthetics
pl <- ggplot(df,aes(x=class))

# geometry
pl2 <- pl + geom_bar()

print(pl2)
```



Use the bar value to add a 3rd layer of data.

Let's color the bars based off the 'drive' column. This creates a stacked barplot.

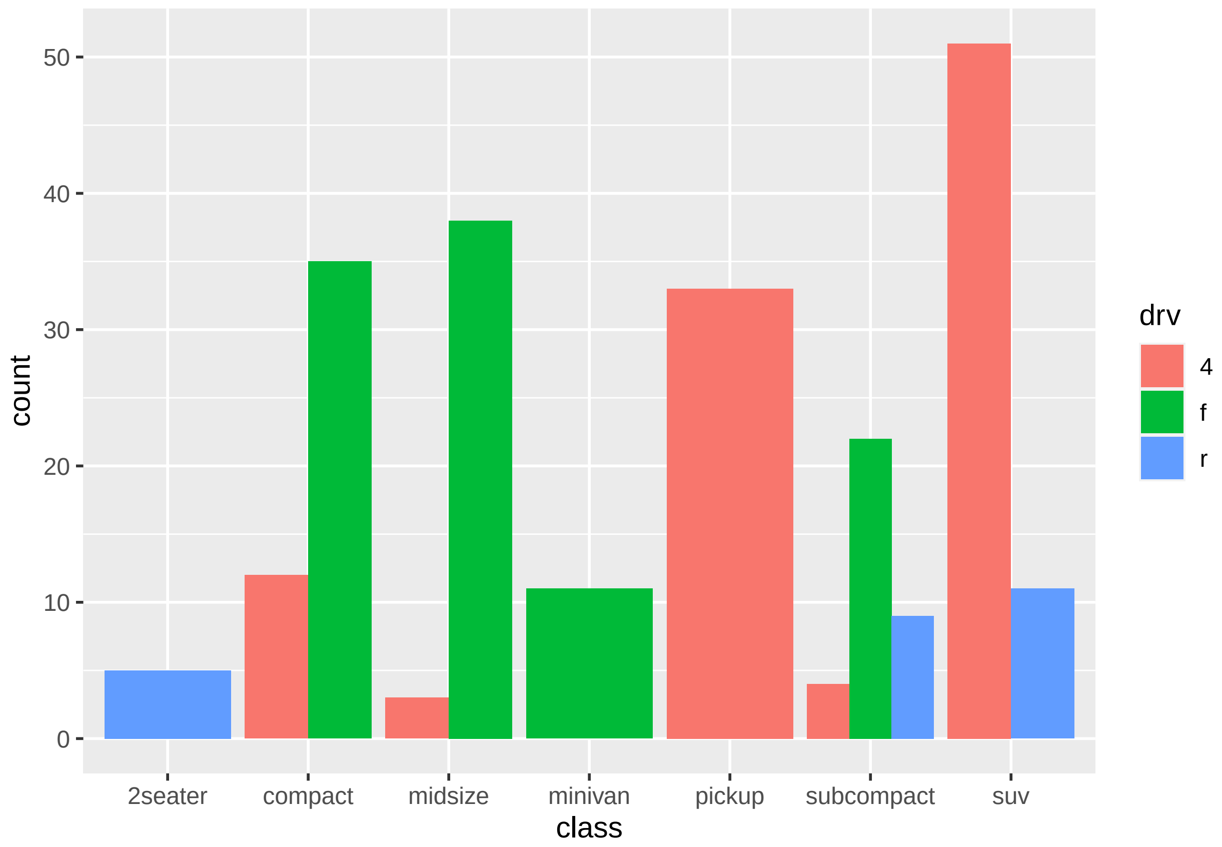
```
# data & aesthetics
pl <- ggplot(df,aes(x=class))

# geometry
pl2 <- pl + geom_bar(aes(fill=drv))

print(pl2)
```



Let’s adjust the position adjustment inside the barplot.

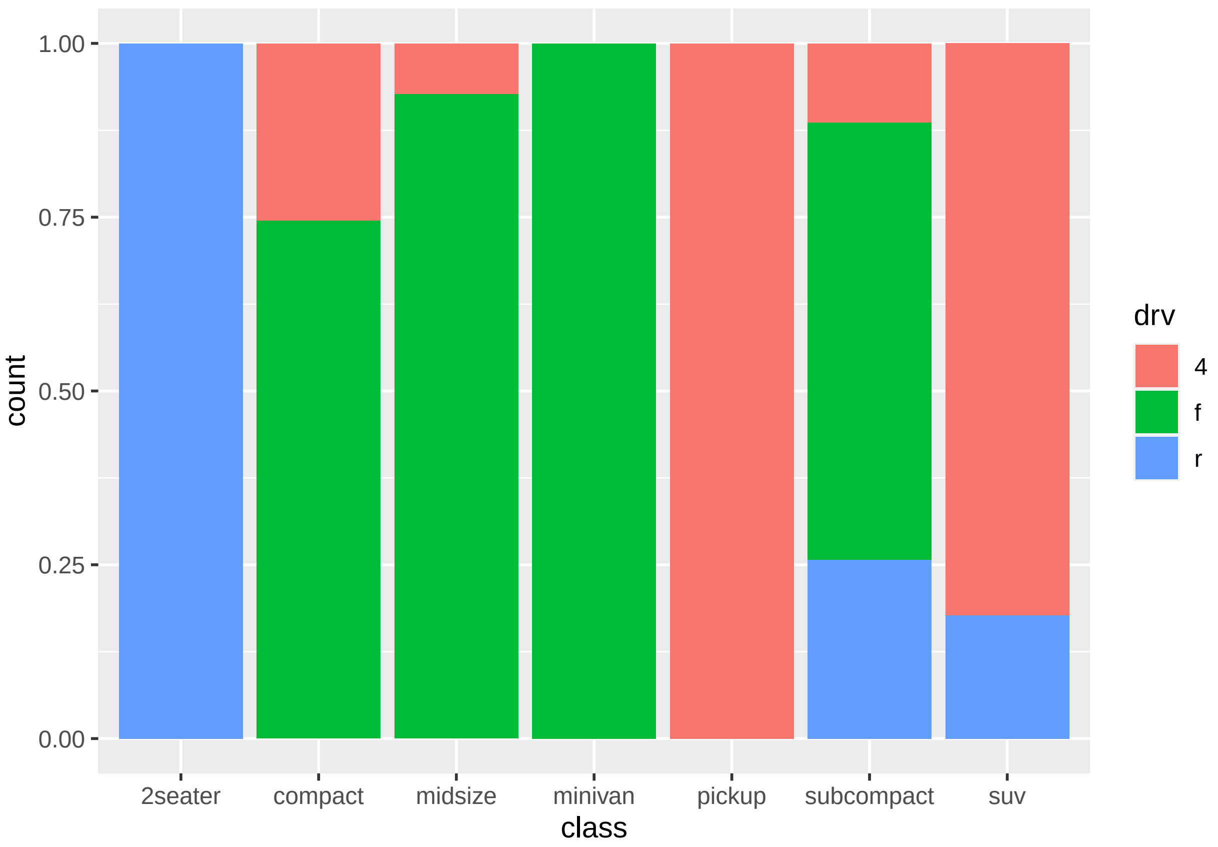
Here we create side-by-side bars instead of stacked bars.

```
# data & aesthetics
pl <- ggplot(df,aes(x=class))

# geometry
pl2 <- pl + geom_bar(aes(fill=drv),position = "dodge")

print(pl2)
```



A different type of positioning within the barplot. Let's show percentages on the bars, rather than counts of instances.

```
# data & aesthetics
pl <- ggplot(df,aes(x=class))

# geometry
pl2 <- pl + geom_bar(aes(fill=drv),position = "fill")

print(pl2)
```

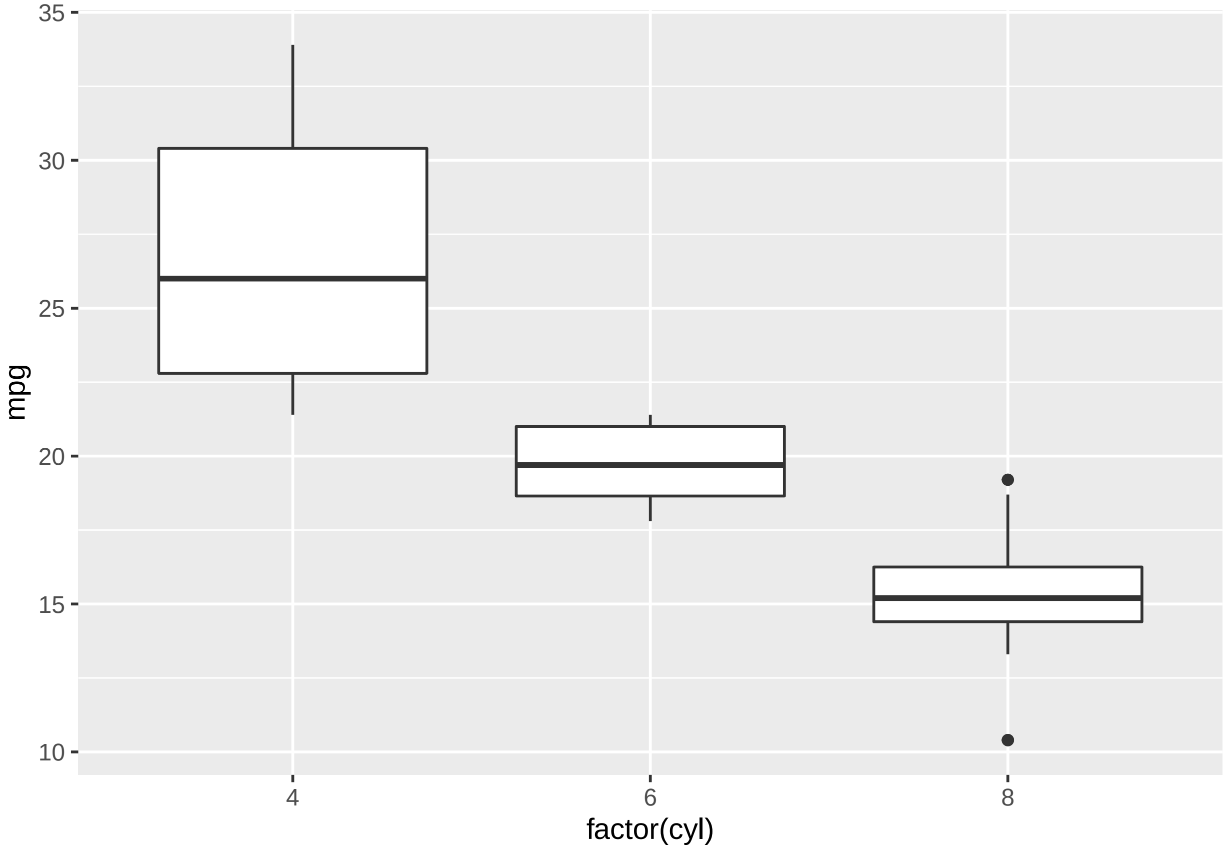


## Boxplots

We graphically depict groups of data through their quartiles.

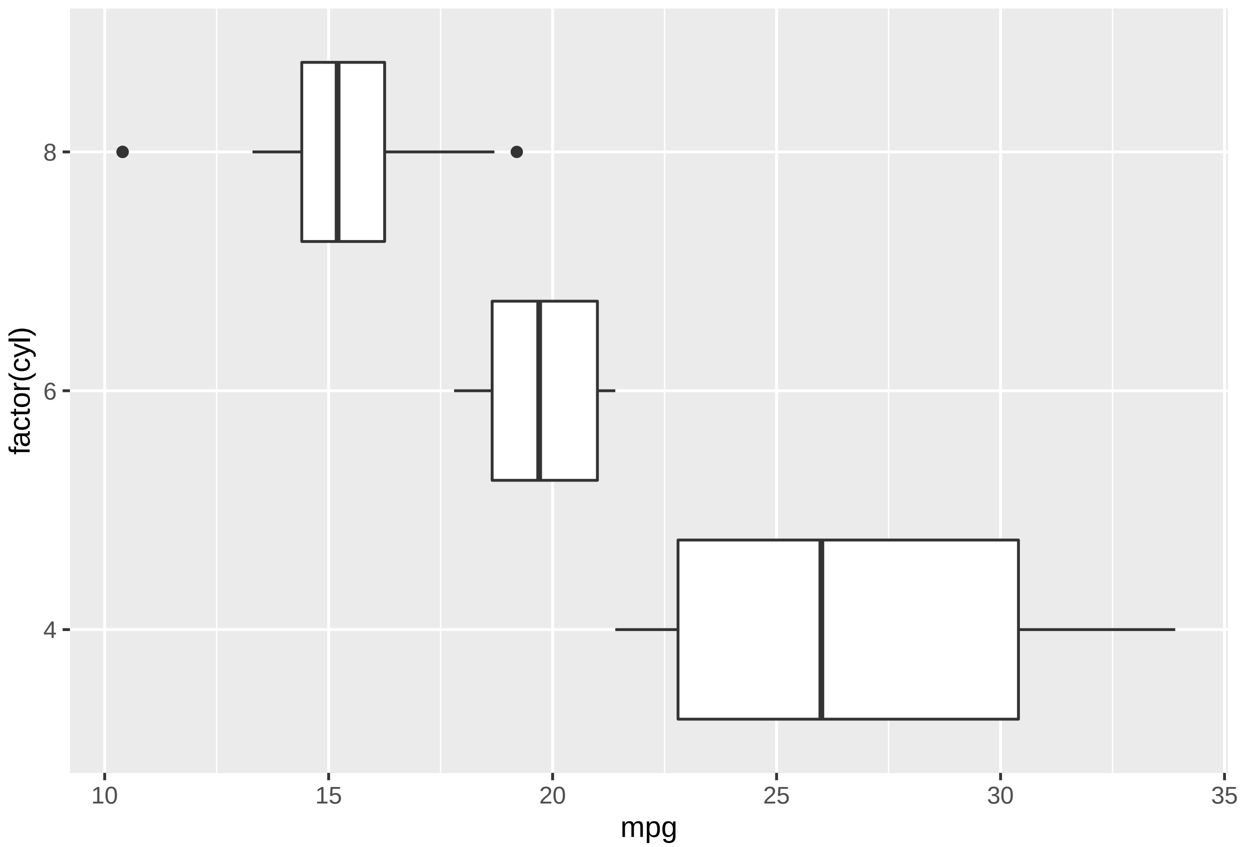
Let’s create box plots of ‘miles per galon’ by ‘cylinder’ type.

```
# we'll use the 'mtcars' data frame again
df <- mtcars

# data & aesthetics
pl <- ggplot(df,aes(x=factor(cyl),y=mpg))

# geometry
pl2 <- pl + geom_boxplot()

print(pl2)
```



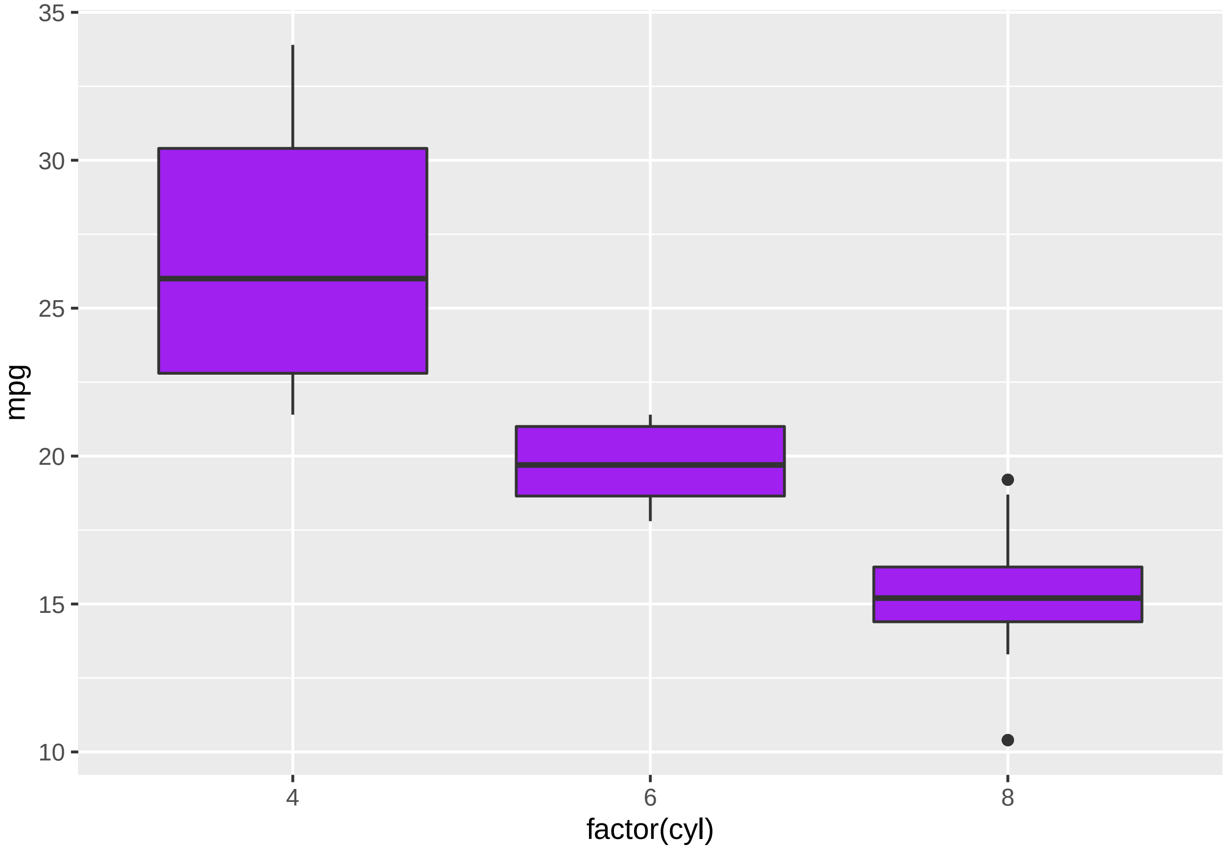
Flip the coordinates of the boxplot by showing it vertically instead of horizontally.

```
# data & aesthetics
pl <- ggplot(df,aes(x=factor(cyl),y=mpg))

# geometry
pl2 <- pl + geom_boxplot() + coord_flip()

print(pl2)
```



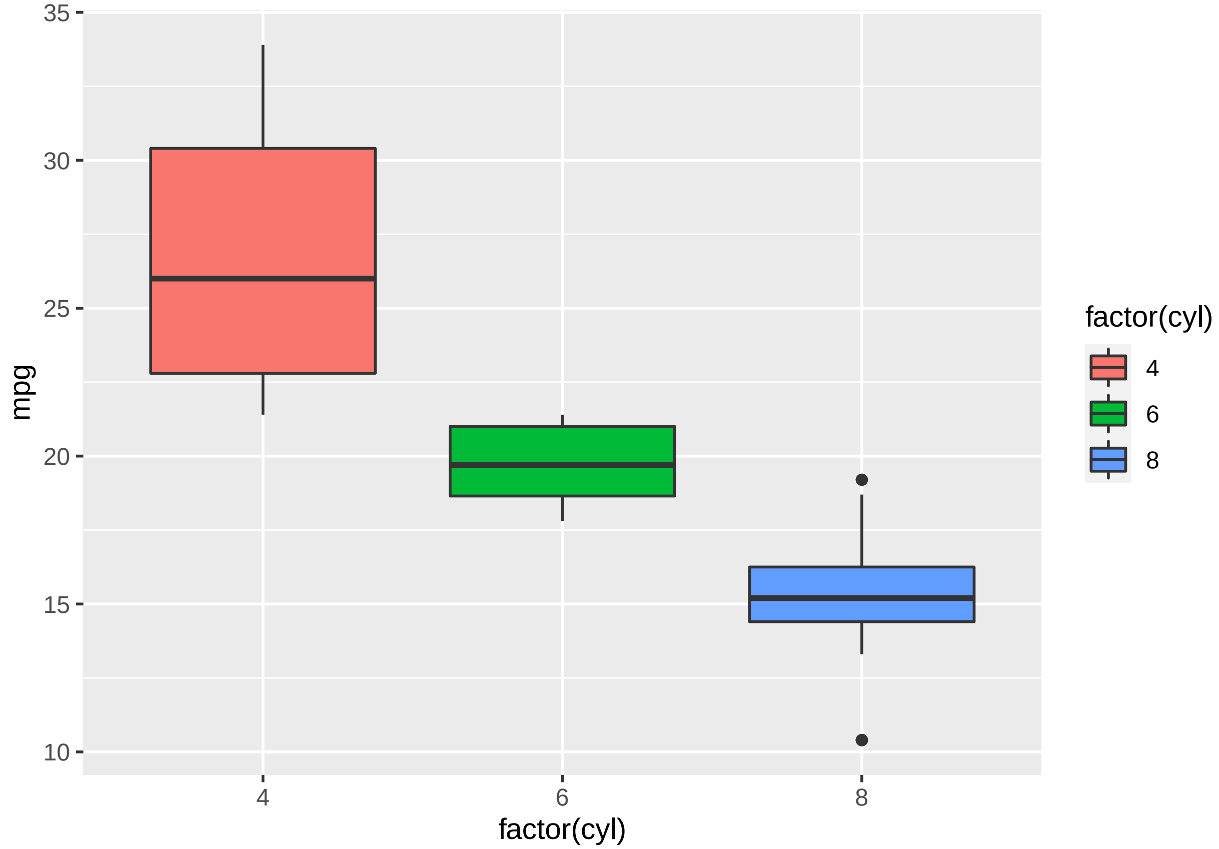
Color the boxes.

```
# data & aesthetics
pl <- ggplot(df,aes(x=factor(cyl),y=mpg))

# geometry
pl2 <- pl + geom_boxplot(fill='purple')

print(pl2)
```



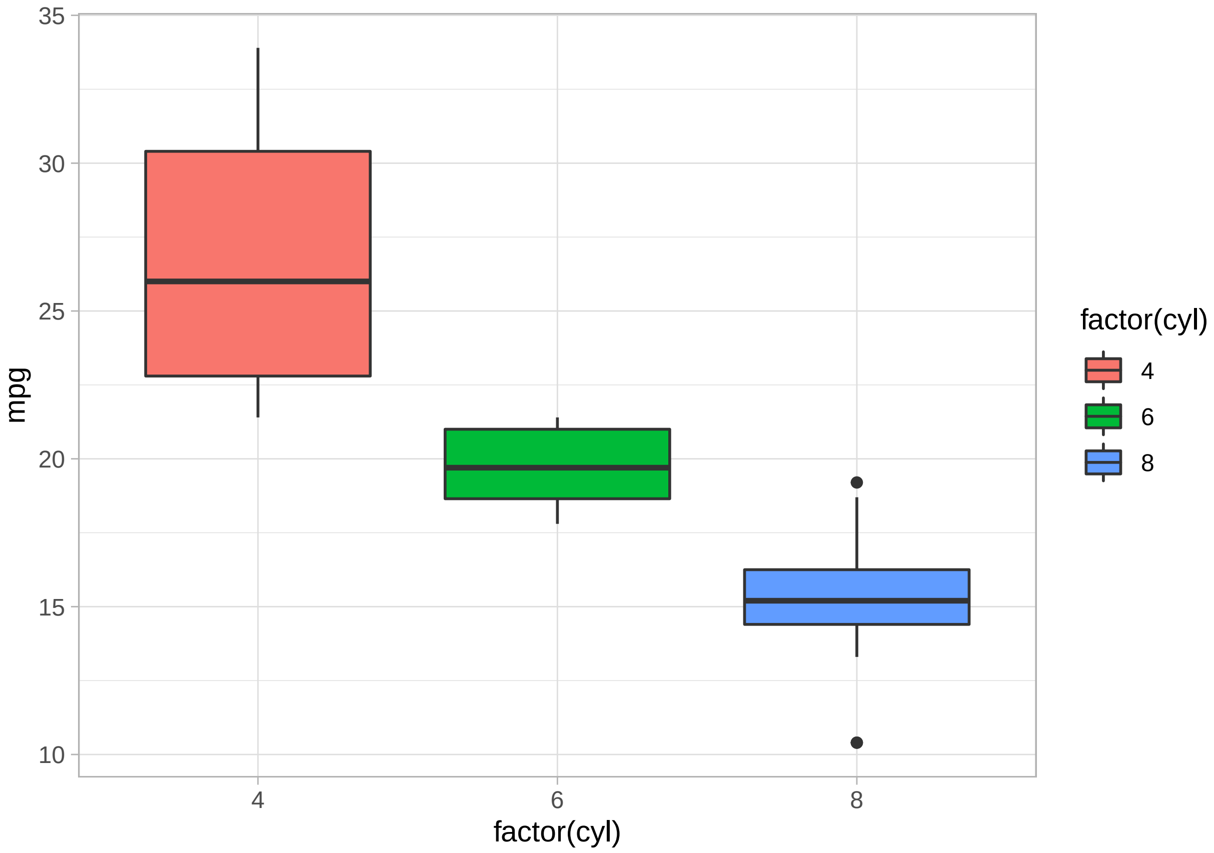
Color box plots based on factors of the cylinder variable.

```
# data & aesthetics
pl <- ggplot(df,aes(x=factor(cyl),y=mpg))

# geometry
pl2 <- pl + geom_boxplot(aes(fill=factor(cyl)))

print(pl2)
```



Now, let's add a theme layer.

```
# data & aesthetics
pl <- ggplot(df,aes(x=factor(cyl),y=mpg))

# geometry
pl2 <- pl + geom_boxplot(aes(fill=factor(cyl))) + theme_light()

print(pl2)
```

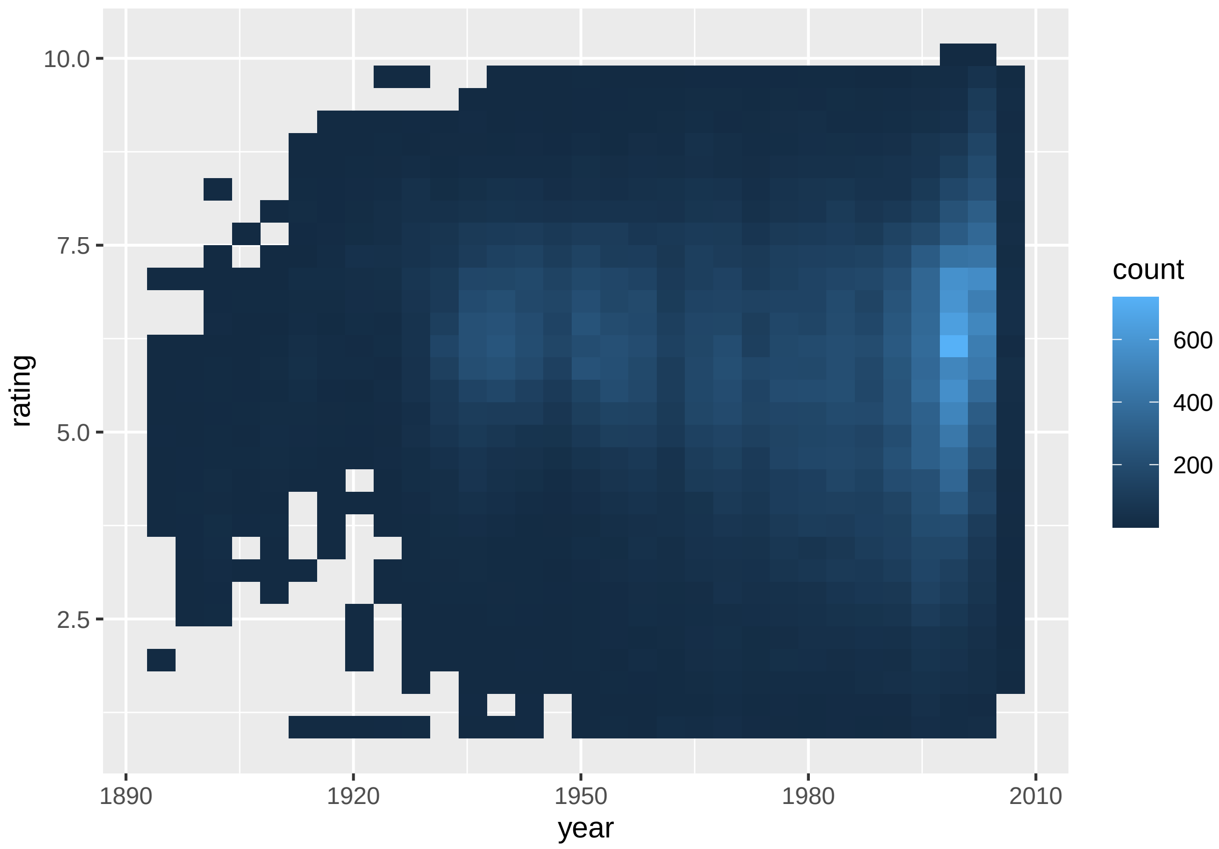


## Variable Plotting

We'll compare two variables from the same dataset.

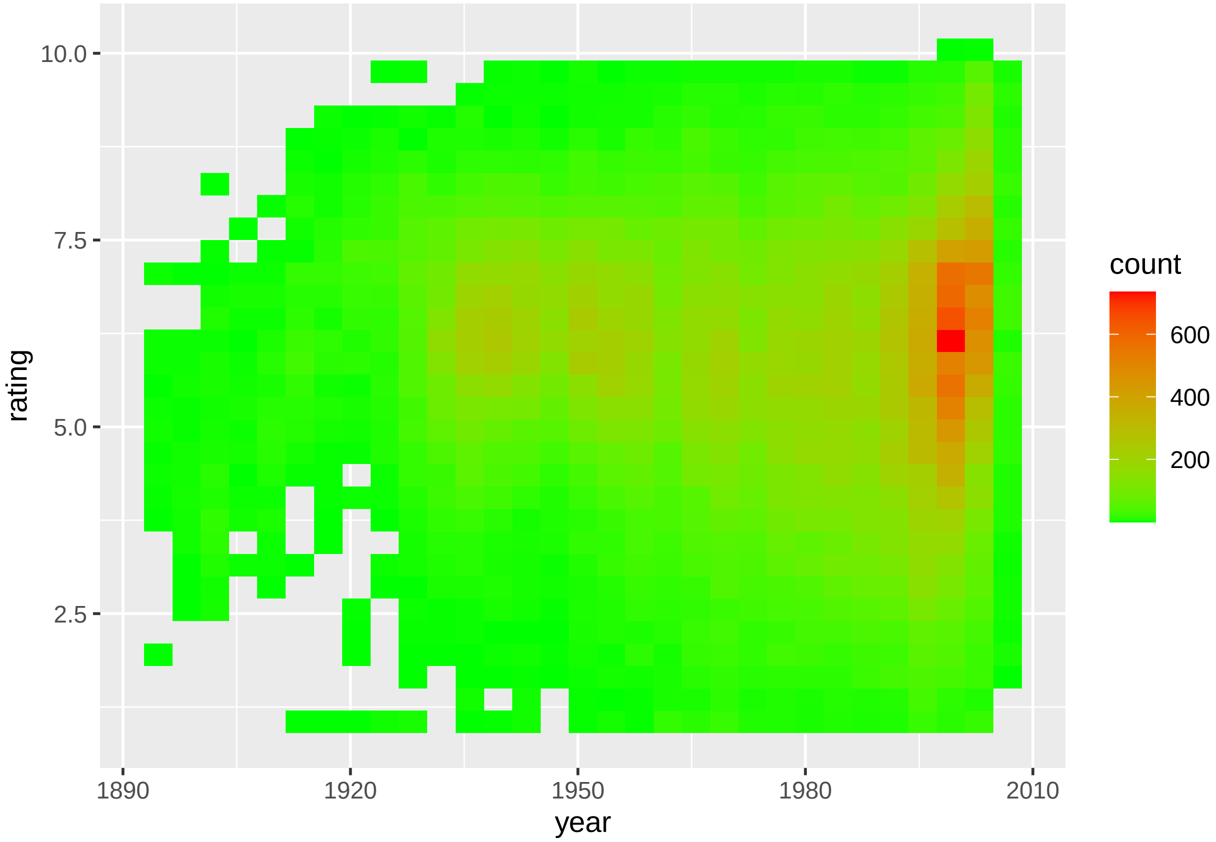
Let's use a '2D bin' chart to plot a heatmap of movie ratings by year.

```
# use the movies dataset
library(ggplot2movies)

# data & aesthetics
pl <- ggplot(movies,aes(x=year,y=rating))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_bin2d()

print(pl2)
```



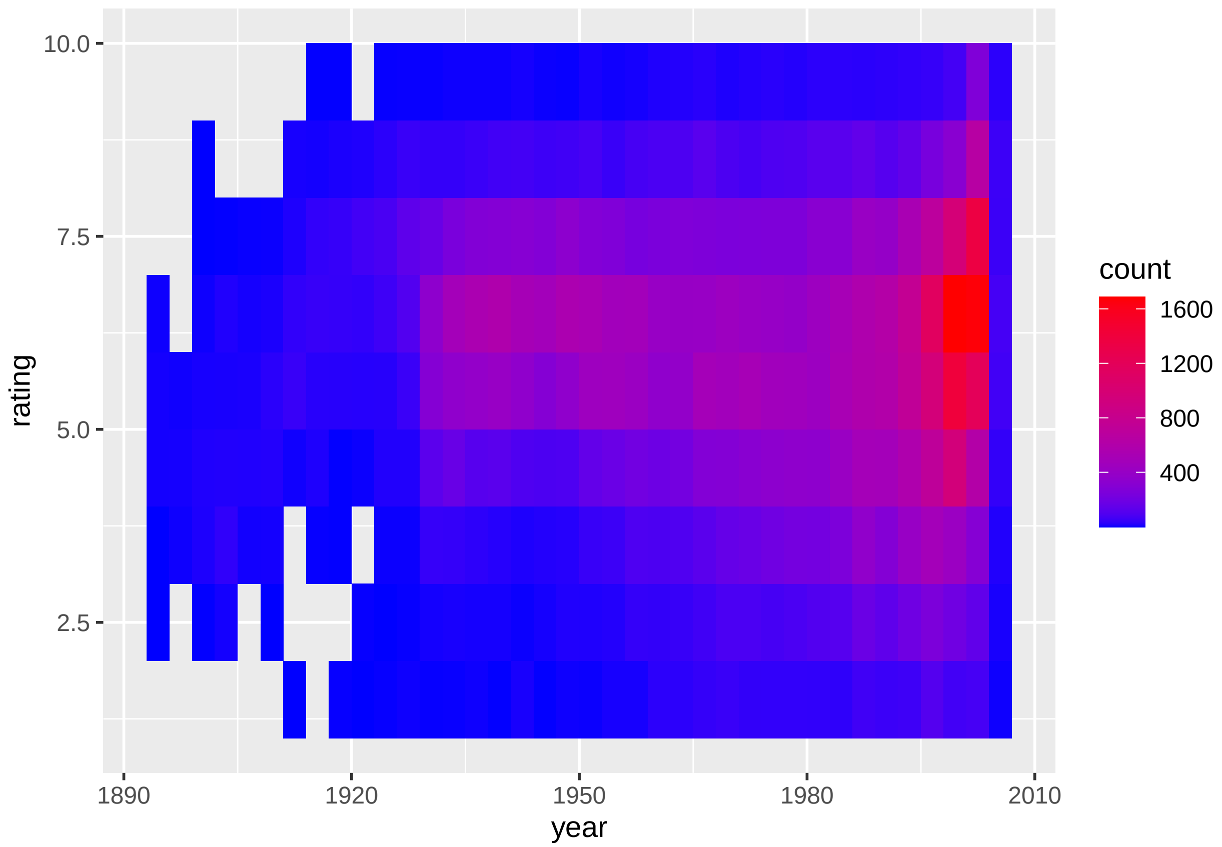
Let's change the default color map.

```
# data & aesthetics
pl <- ggplot(movies,aes(x=year,y=rating))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_bin2d()

# we can specify the color range
pl3 <- pl2 + scale_fill_gradient(low='green',high='red')

print(pl3)
```



By default, the bin sizes are just c(1,1). We can Control the bin sizes.

```
# data & aesthetics
pl <- ggplot(movies,aes(x=year,y=rating))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_bin2d(binwidth=c(3,1))

# we can specify the color range
pl3 <- pl2 + scale_fill_gradient(low='blue',high='red')

print(pl3)
```

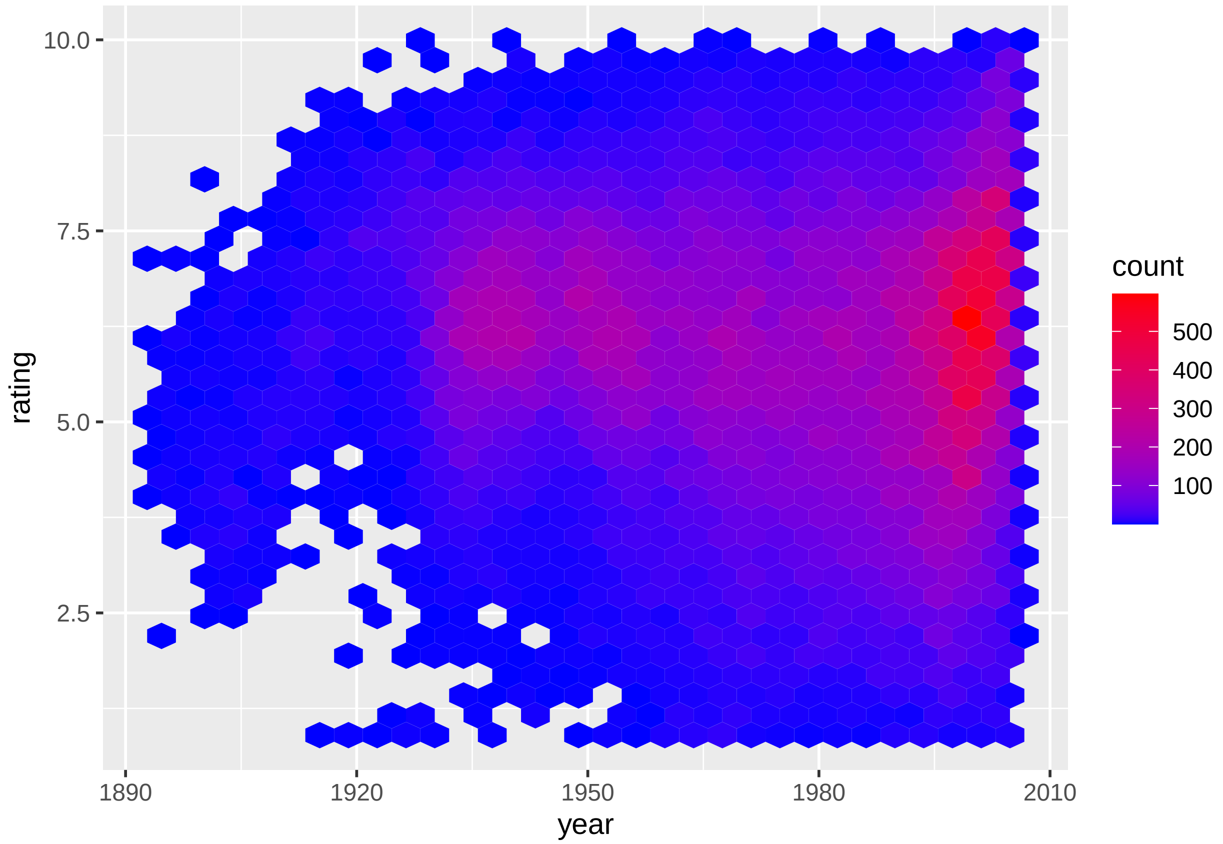


### HEX plot

We can also create a HEX plot instead of a rectangle '2d bin' chart.

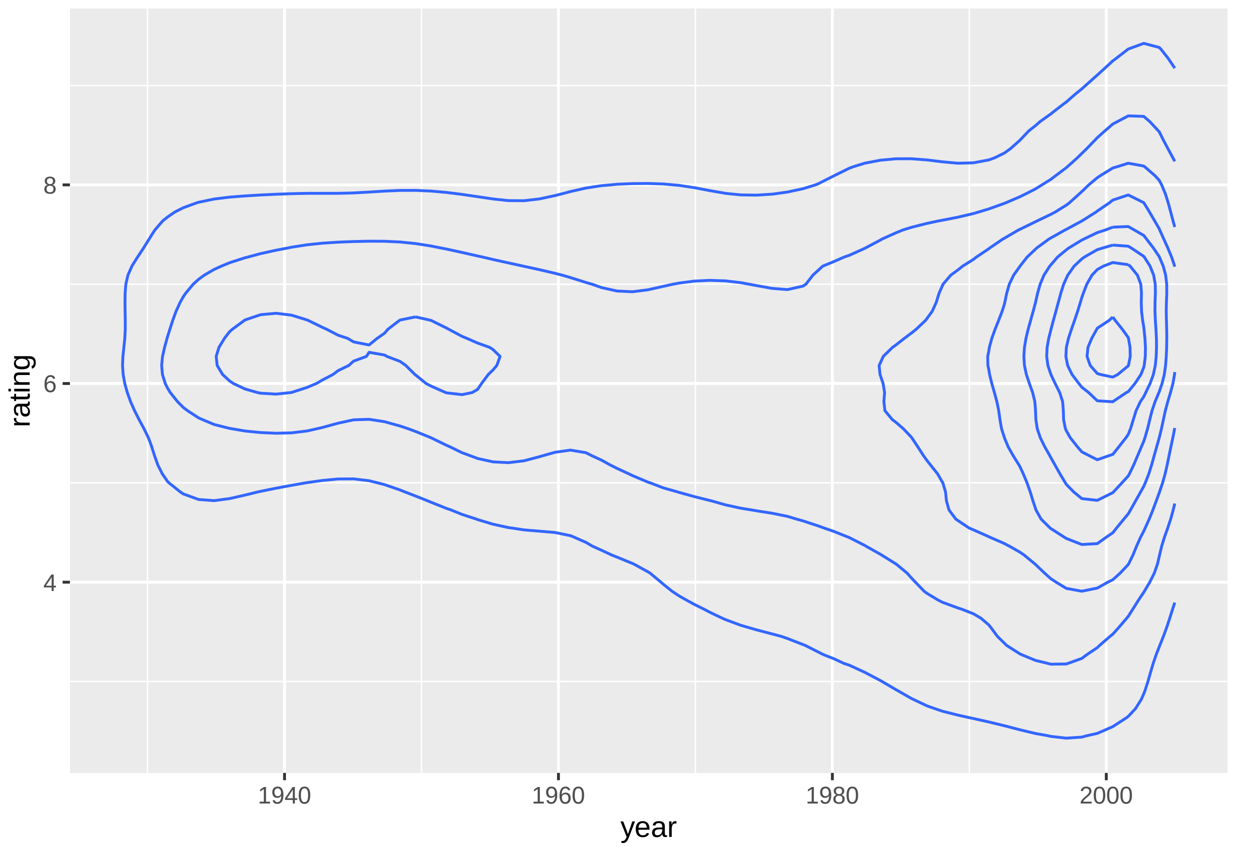
Note that the `geom_hex()` option requires the `hexbin` package to be installed.

```
# install the `hexbin` package
#install.packages("hexbin")

# data & aesthetics
pl <- ggplot(movies,aes(x=year,y=rating))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_hex()

# we can specify the color range
pl3 <- pl2 + scale_fill_gradient(low='blue',high='red')

print(pl3)
```



2D Density plot

```
# data & aesthetics
pl <- ggplot(movies,aes(x=year,y=rating))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_density_2d()

print(pl2)
```



## Coordinates and Faceting

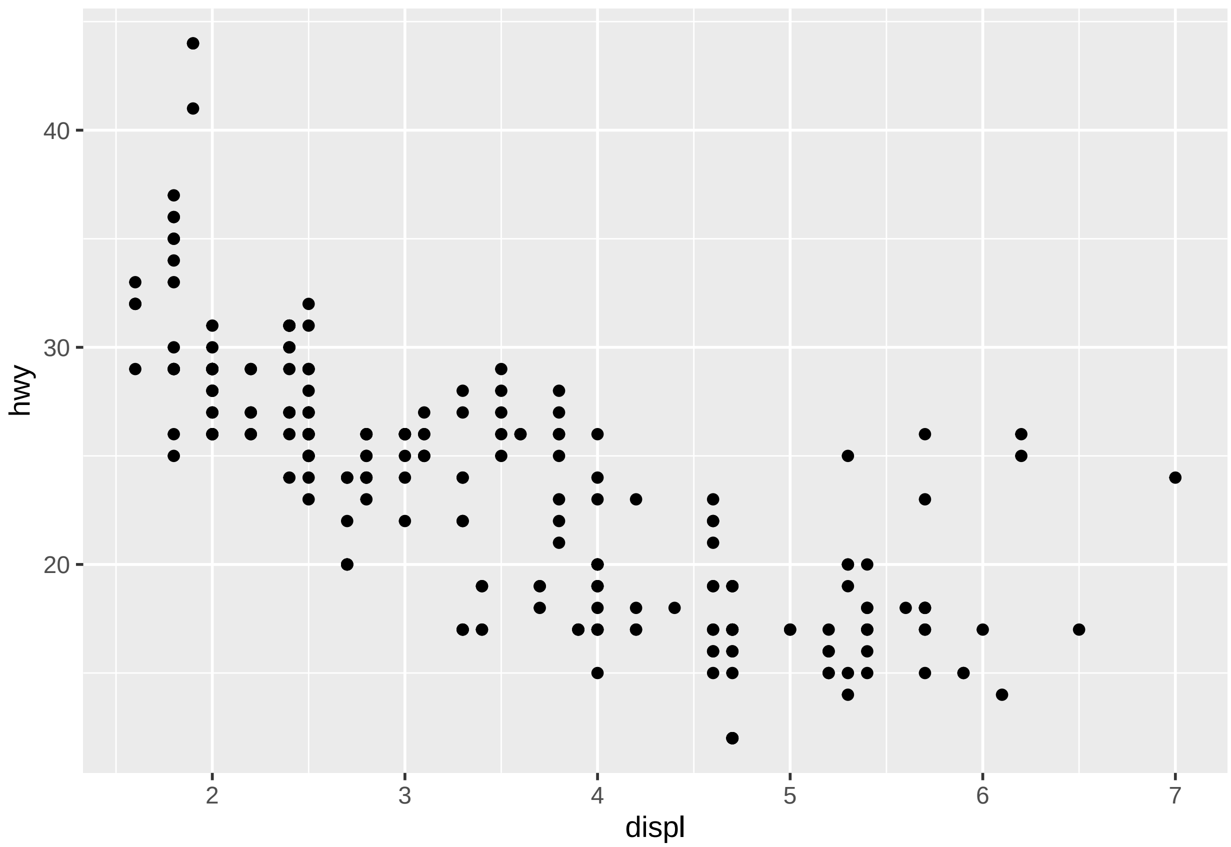
Learning to deal with coordinates will allow us to resize our plots correctly.

Faceting will allow us to place multiple plots next to one another.

Coordinates

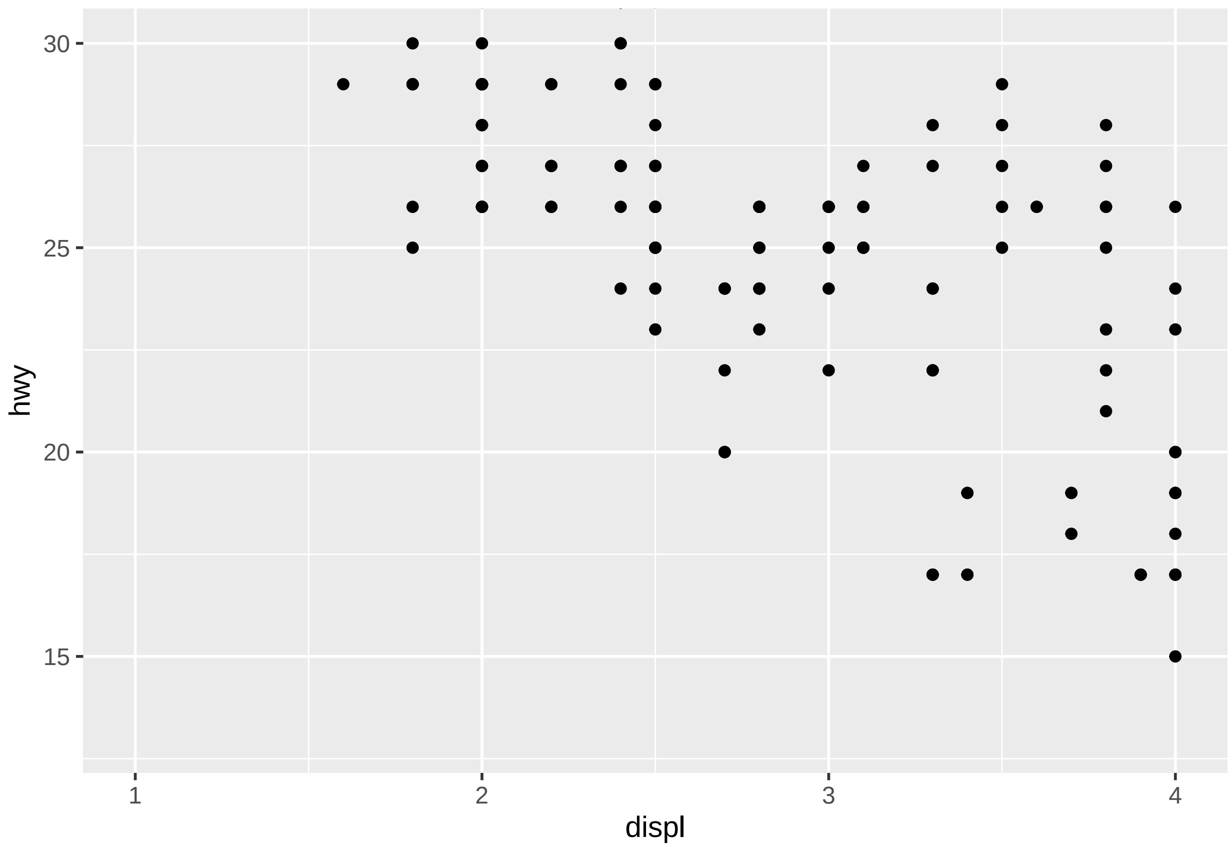
We'll use the 'mpg' bulit-in data set to create a scatterplot.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

print(pl2)
```



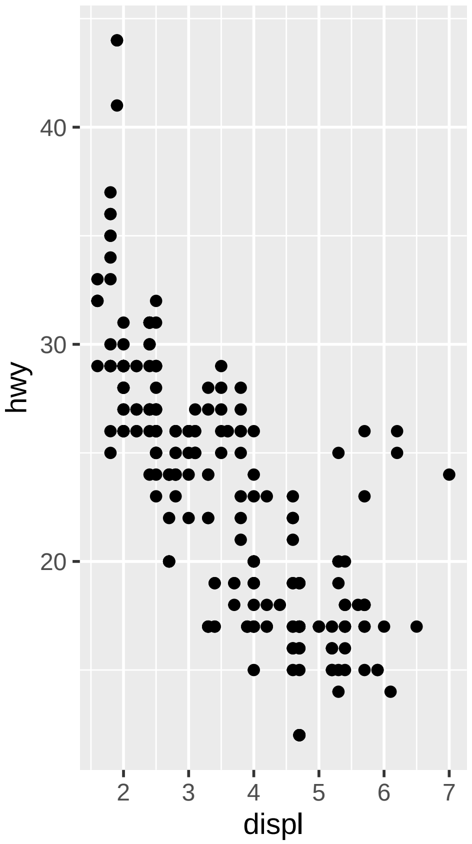
Play around with coordinates in order to set x and y limts.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set x-y limits
pl3 <- pl2 + coord_cartesian(xlim = c(1,4),ylim = c(13,30))

print(pl3)
```



Set aspect ratio. This ratio is 1:1 by default.

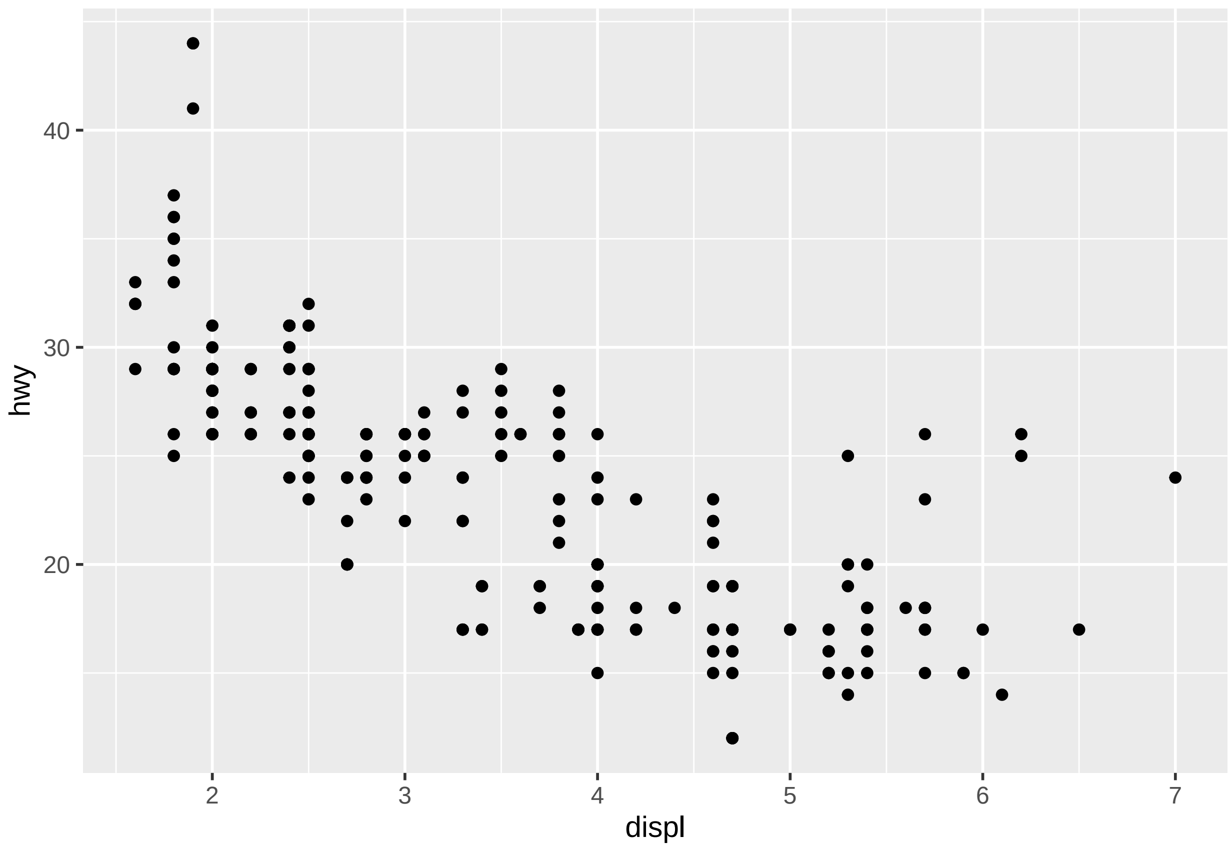
```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set the 'x to y' aspect ratio to 1/3
pl3 <- pl2 + coord_fixed(ratio= 1/3)

print(pl3)
```



Faceting

First create a normal scatterplot.

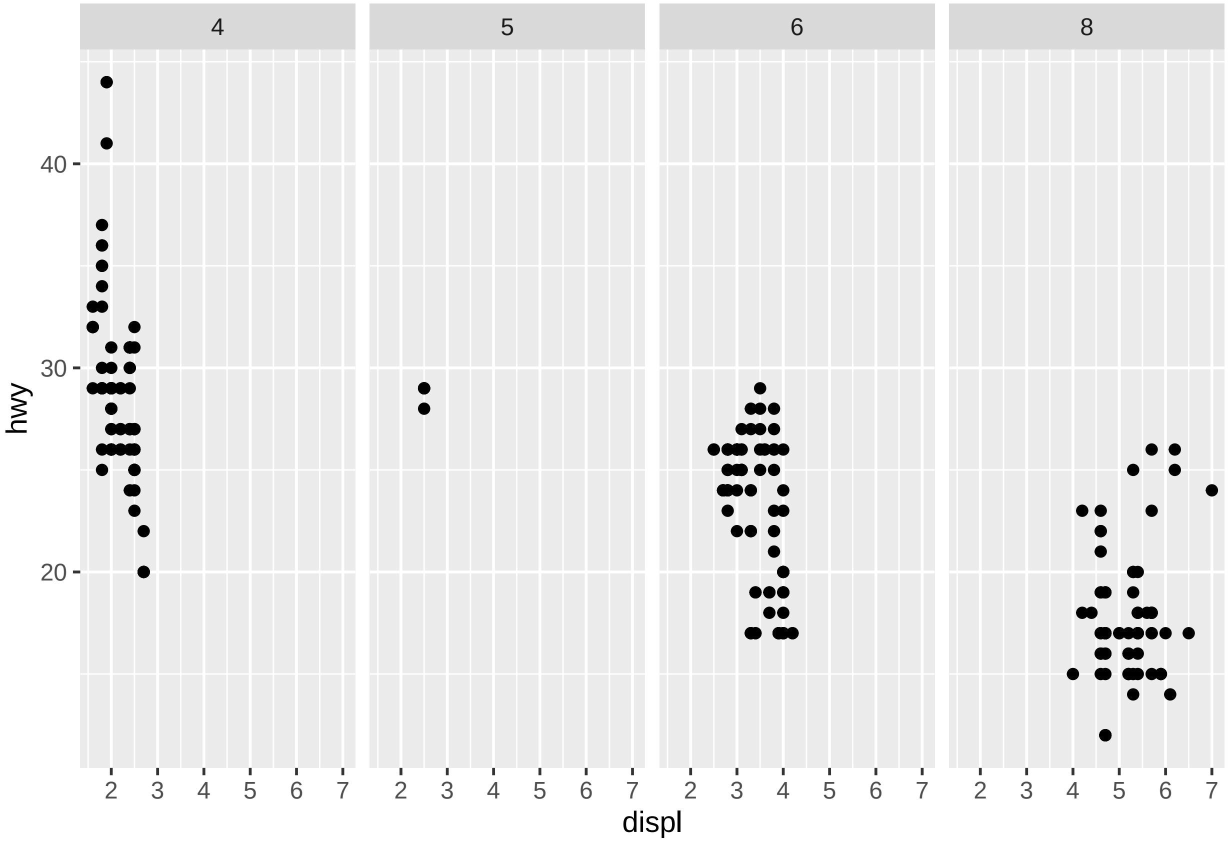
```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

print(pl2)
```



Now let's separate the normal scatterplot into facets based on the number of cylinders.

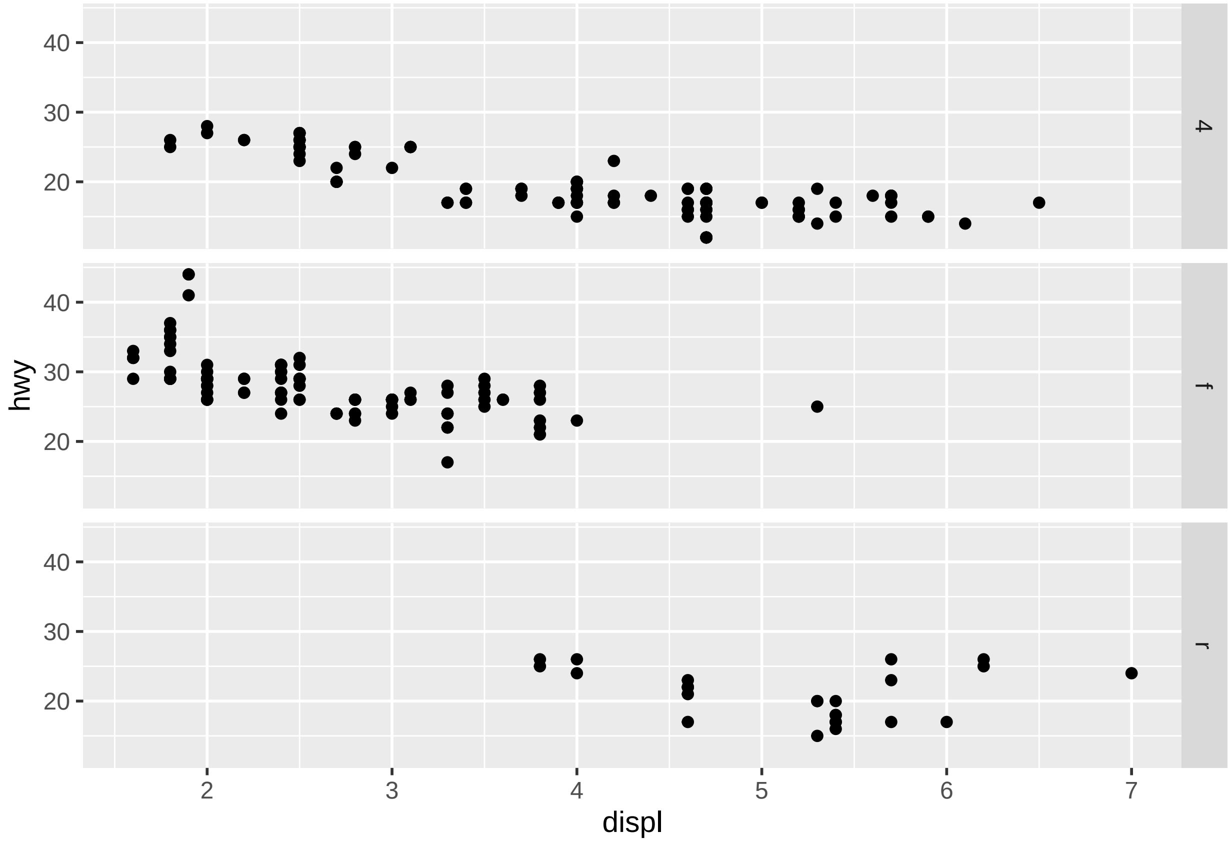
The . symbol in `facet_grid()` represents everything.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# separate the normal scatterplot into facets based on the number of cylinders
pl3 <- pl2 + facet_grid(. ~ cyl)

print(pl3)
```



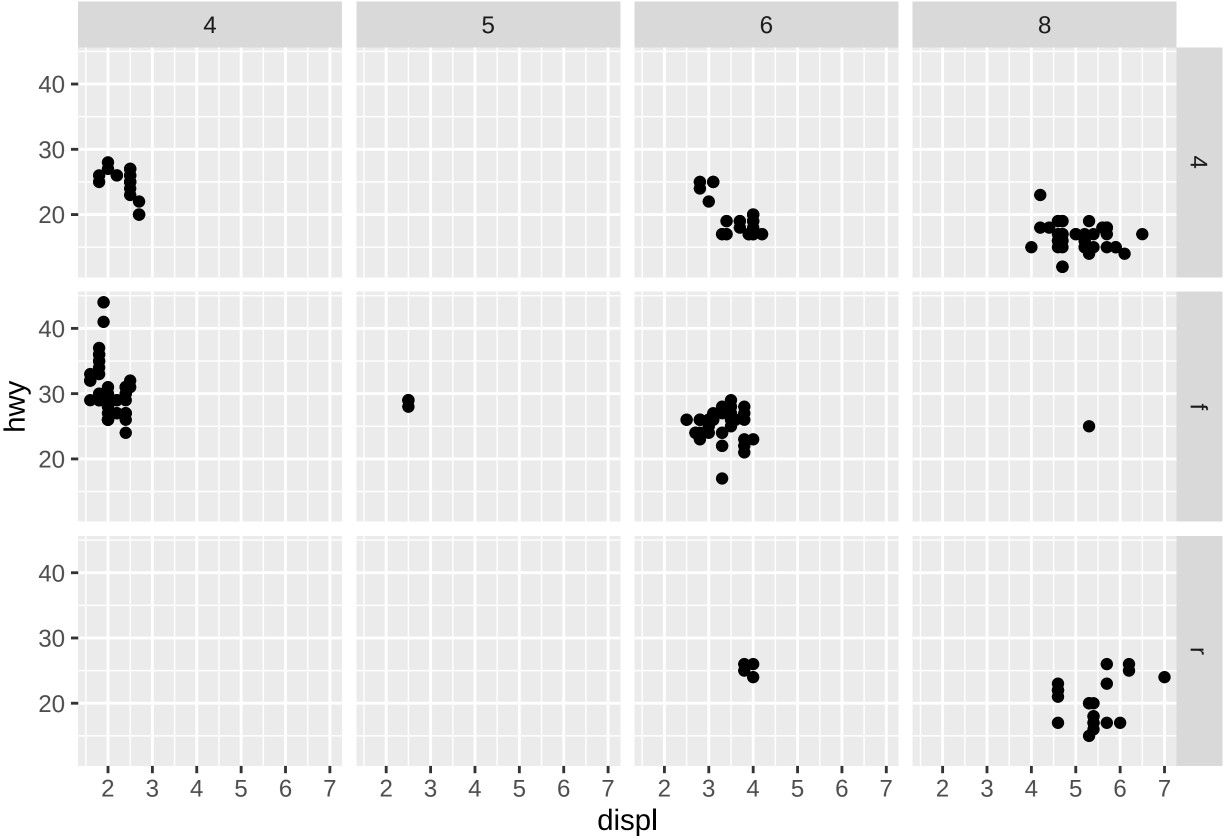
Another example: facet by drive type.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# separate the normal scatterplot into facets based on the number of cylinders
pl3 <- pl2 + facet_grid(drv ~ .)

print(pl3)
```



Facet by drive type and the number of cylinders.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=displ,y=hwy))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# separate the normal scatterplot into facets based on the number of cylinders
pl3 <- pl2 + facet_grid(drv ~ cyl)

print(pl3)
```



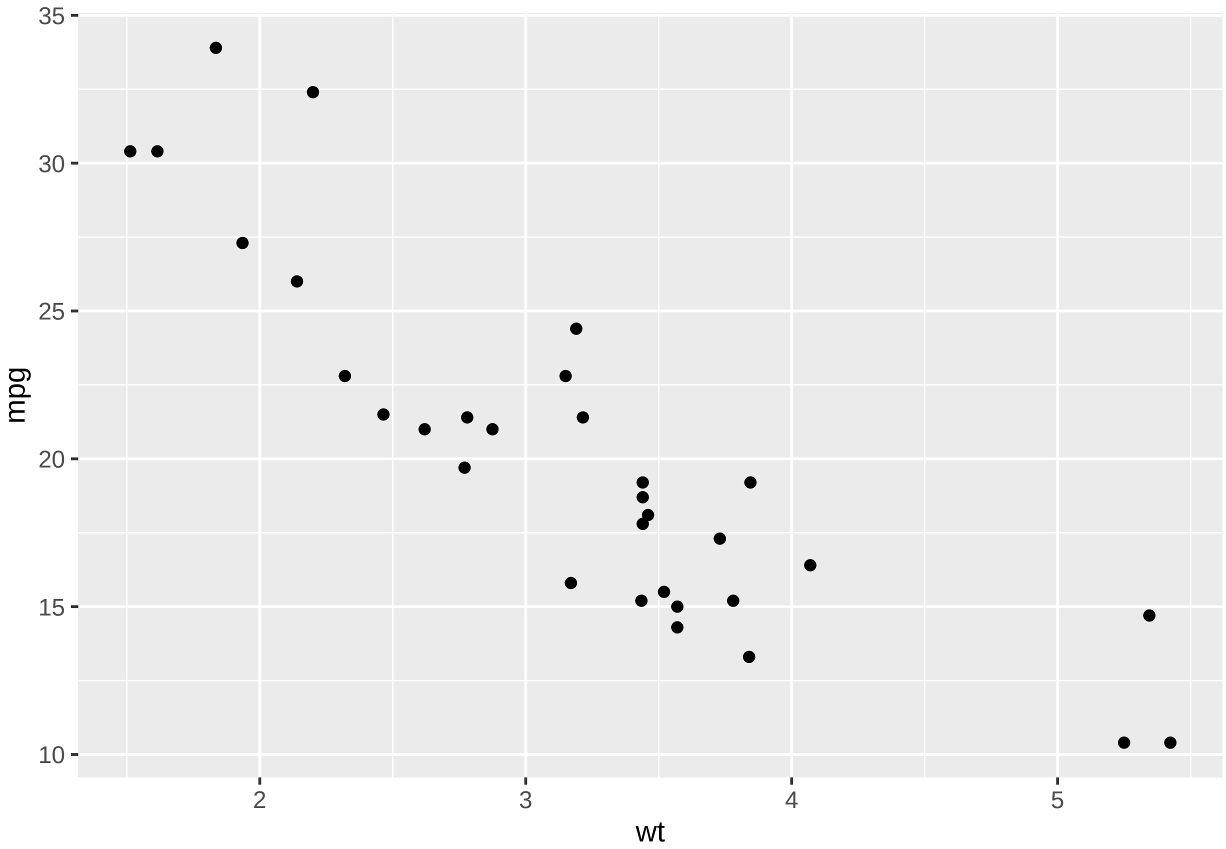
## Themes

Themes allow us to quickly create beautiful looking graphs.

Using built-in themes

Create a standard scatterplot.

```
# use the built-in 'mtcars' dataset
df <- mtcars

# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

print(pl2)
```



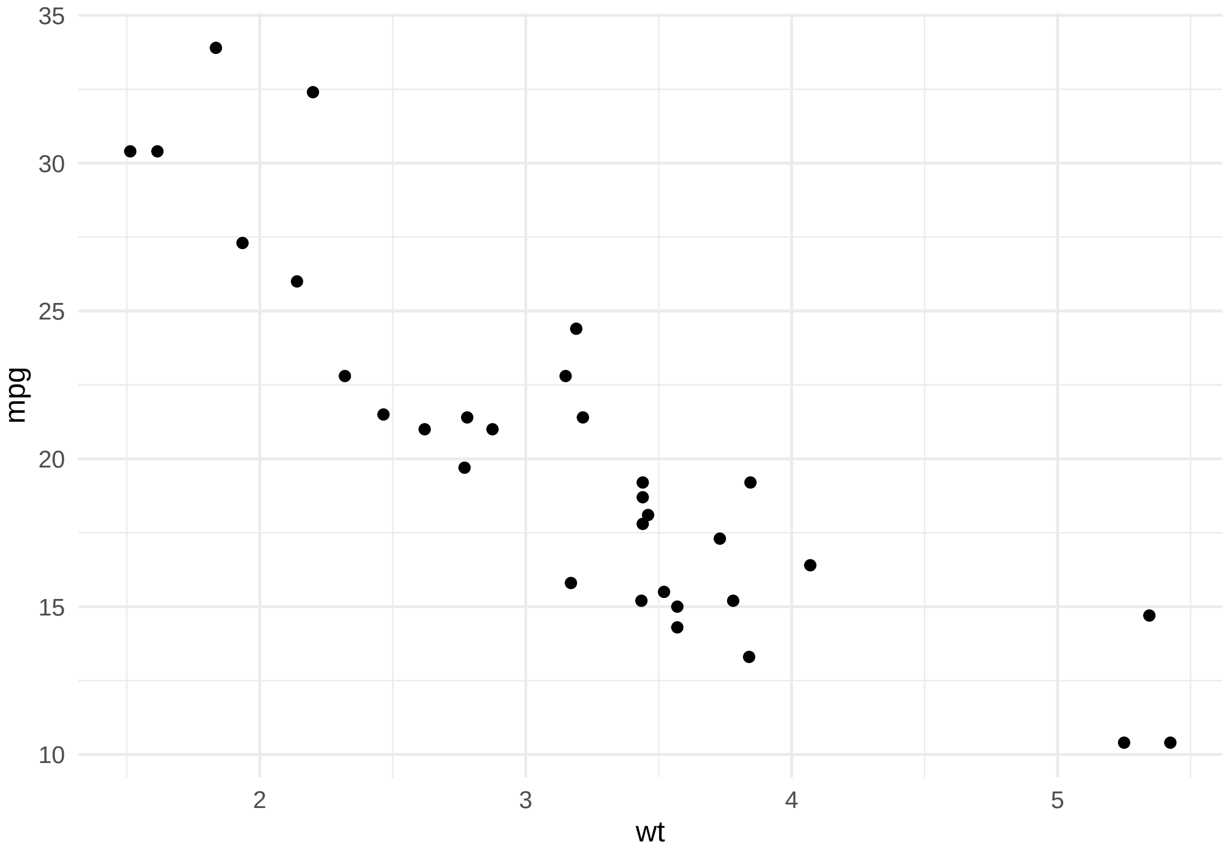
Let's apply a theme layer.

We can set a theme that applies to all our plots like this:

```
theme_set(theme_minimal())
```

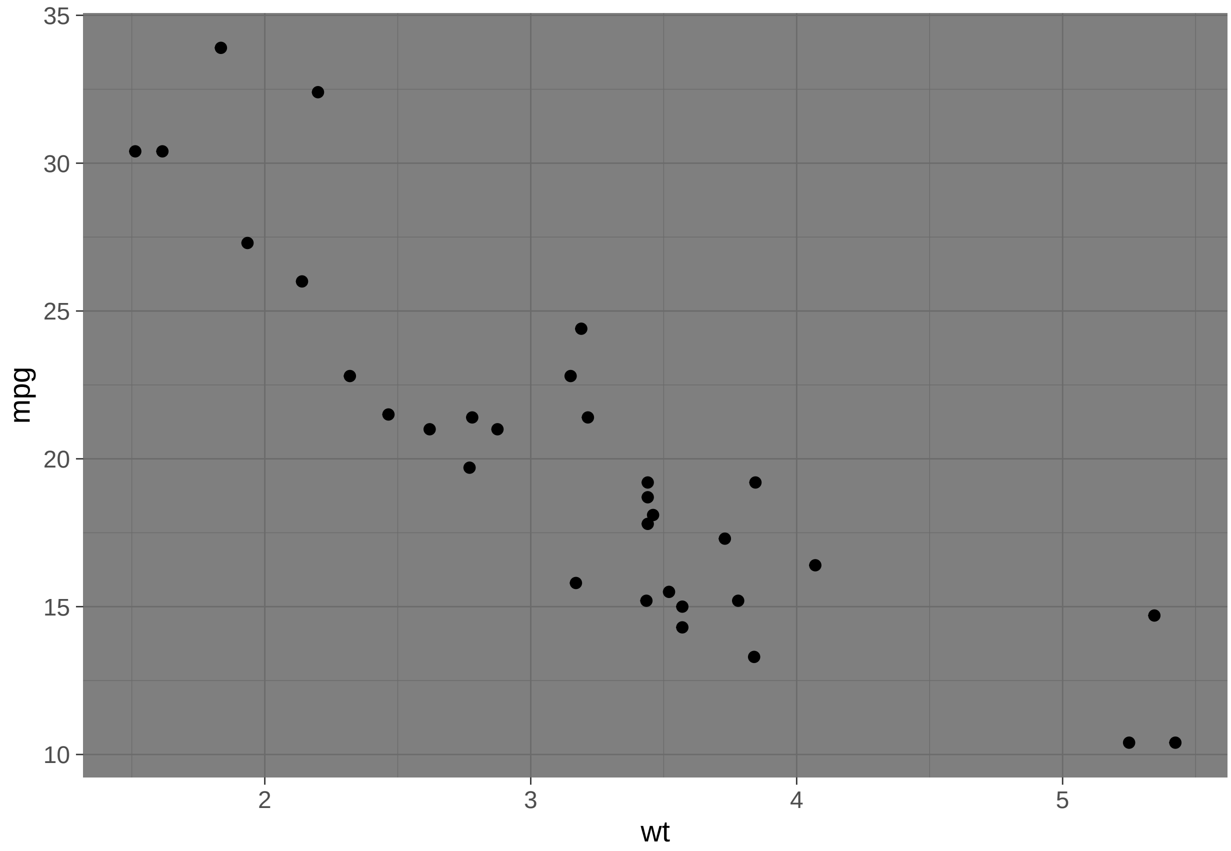
Now let's re-plot our basic scatter plot.

```
# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

print(pl2)
```



Instead, we can manually set a theme for each plot.

```
# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set theme manually
pl3 <- pl2 + theme_dark()

print(pl3)
```



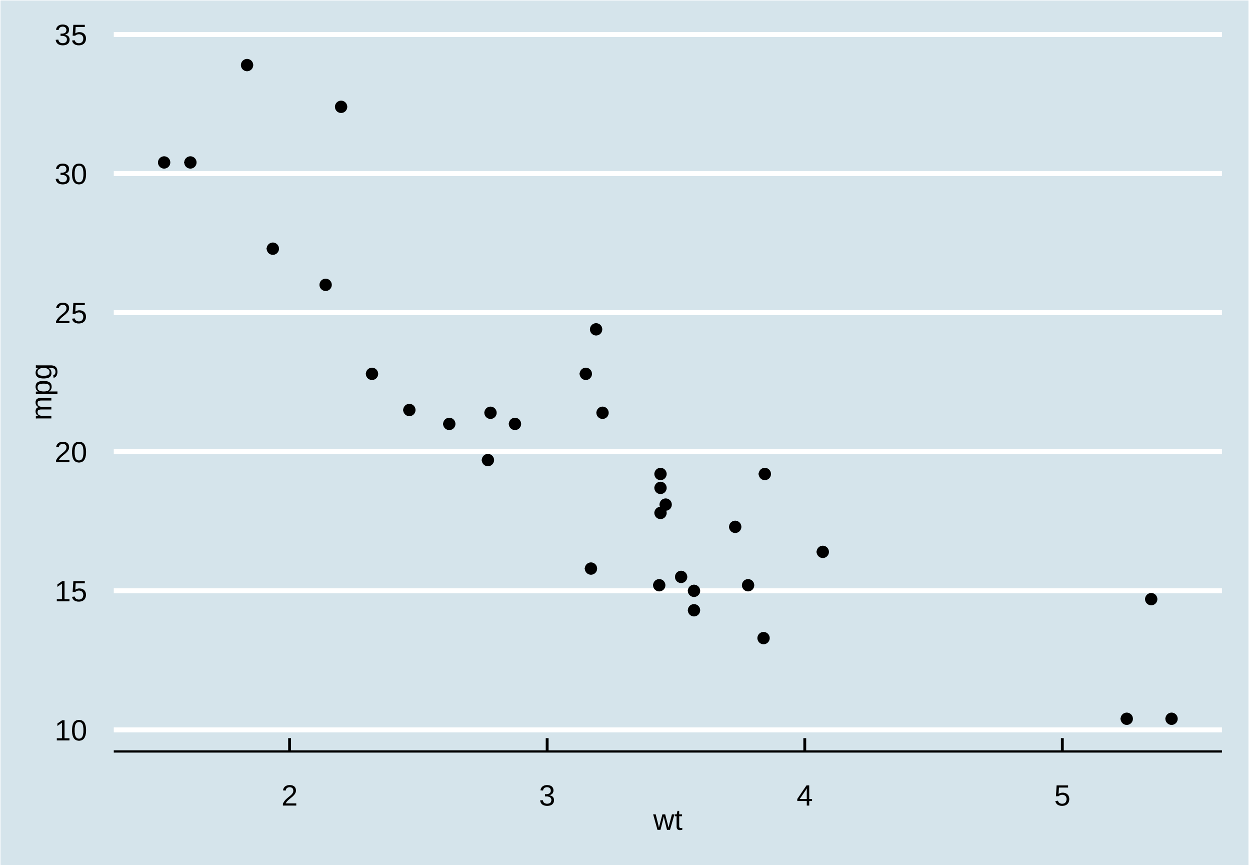
Accessing more theme options

We can install more themes by installing and calling the `ggthemes` package.

```
#install.packages("ggthemes")

library(ggthemes)
```

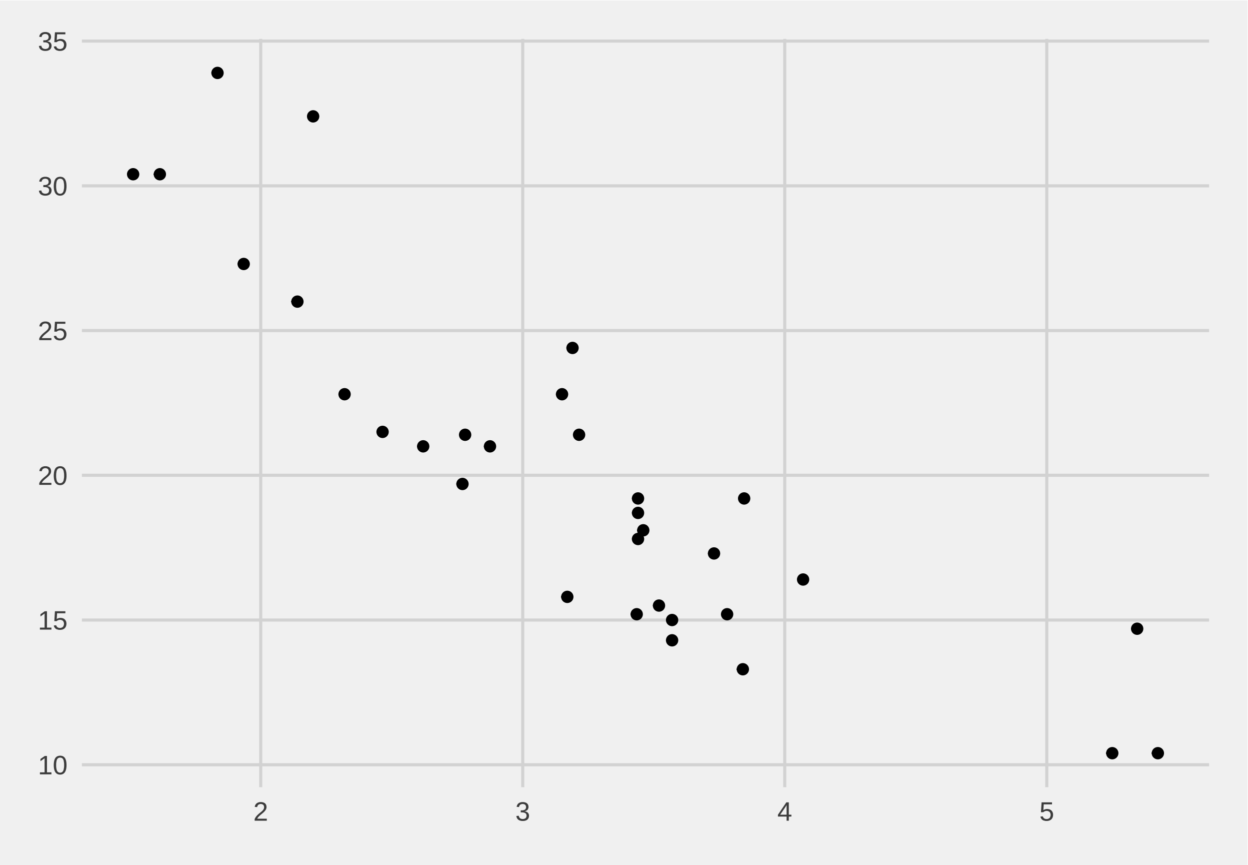
Now we can access additional themes, like the Economist's publications.

```
# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set theme manually
pl3 <- pl2 + theme_economist()

print(pl3)
```



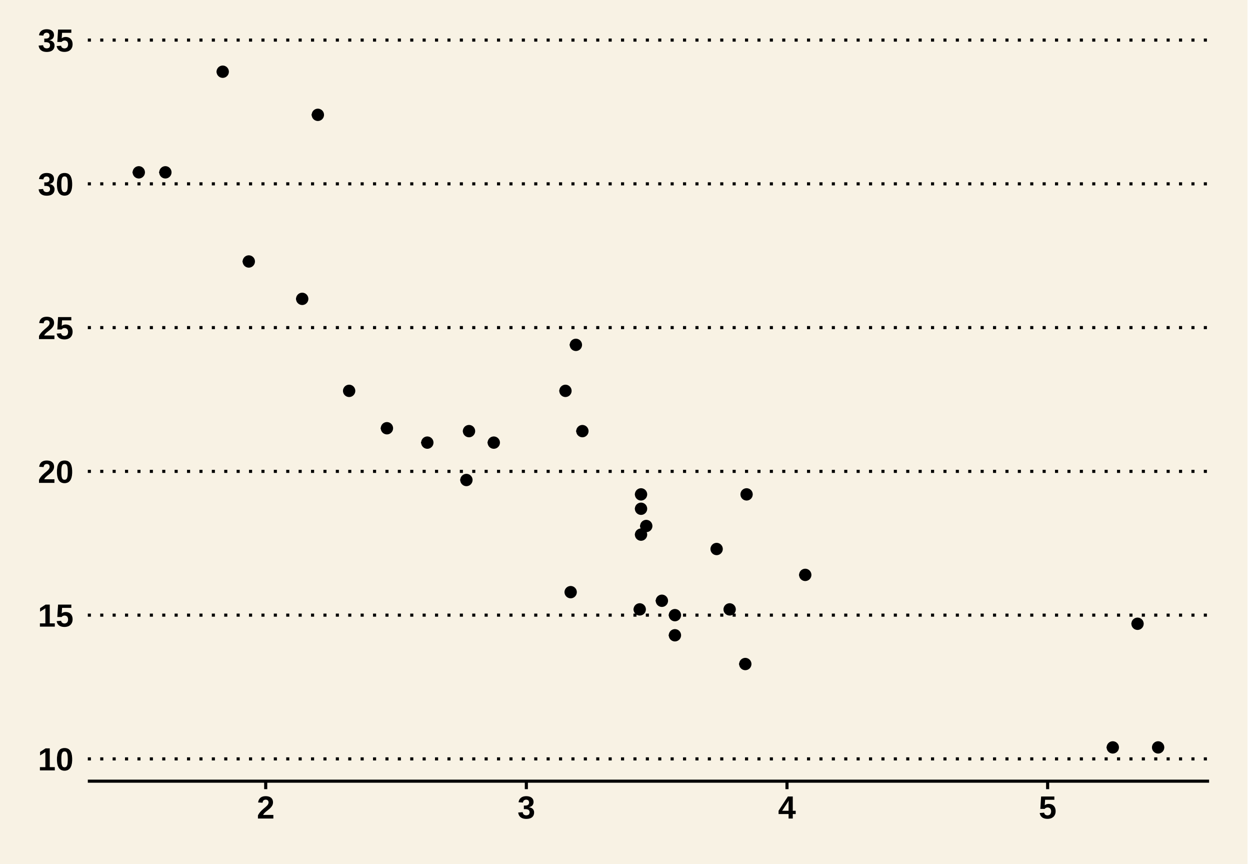
Imitate the style of the 'fivethirtyeight' blog.

```
# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set theme manually
pl3 <- pl2 + theme_fivethirtyeight()

print(pl3)
```



Try the Wall Street Journal theme.

```
# data & aesthetics
pl <- ggplot(mtcars,aes(x=wt,y=mpg))

# geometry: check out the '2D bin' chart
pl2 <- pl + geom_point()

# set theme manually
pl3 <- pl2 + theme_wsj()

print(pl3)
```



## ggplot2 Exercises

For the first few plots, use the mpg dataset.

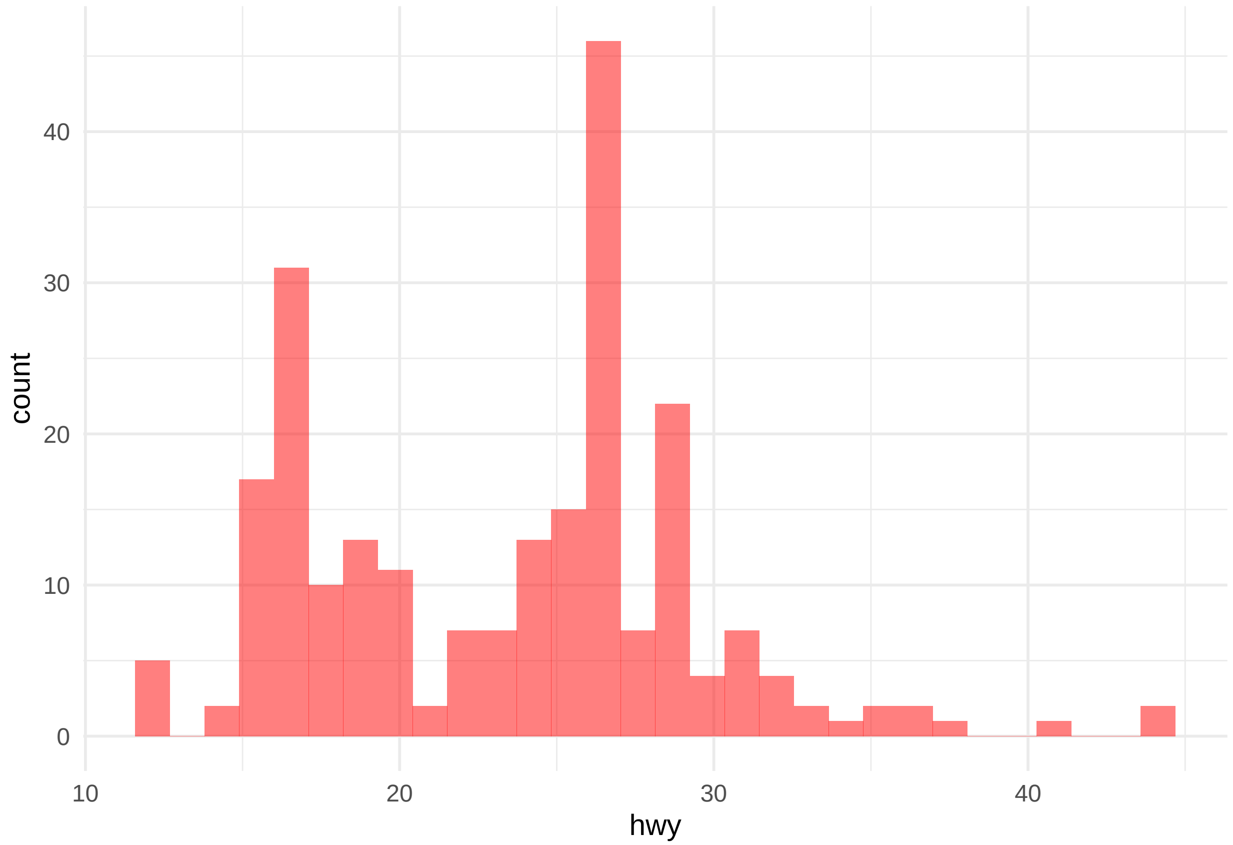
```
# call necessary libraries
# library(ggplot2)
# library(ggthemes)

# preview the 'mpg' dataset
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

Ex1: Histogram of hwy mpg values:

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=hwy))

# geometry
#pl2 <- pl + geom_histogram(binwidth = 0.1, color='red',fill='pink',alpha=0.4)
pl2 <- pl + geom_histogram(fill='red',alpha=0.5)

# add labels
#pl3 <- pl2 + xlab('Highway') + ylab('Count')

print(pl2)
```

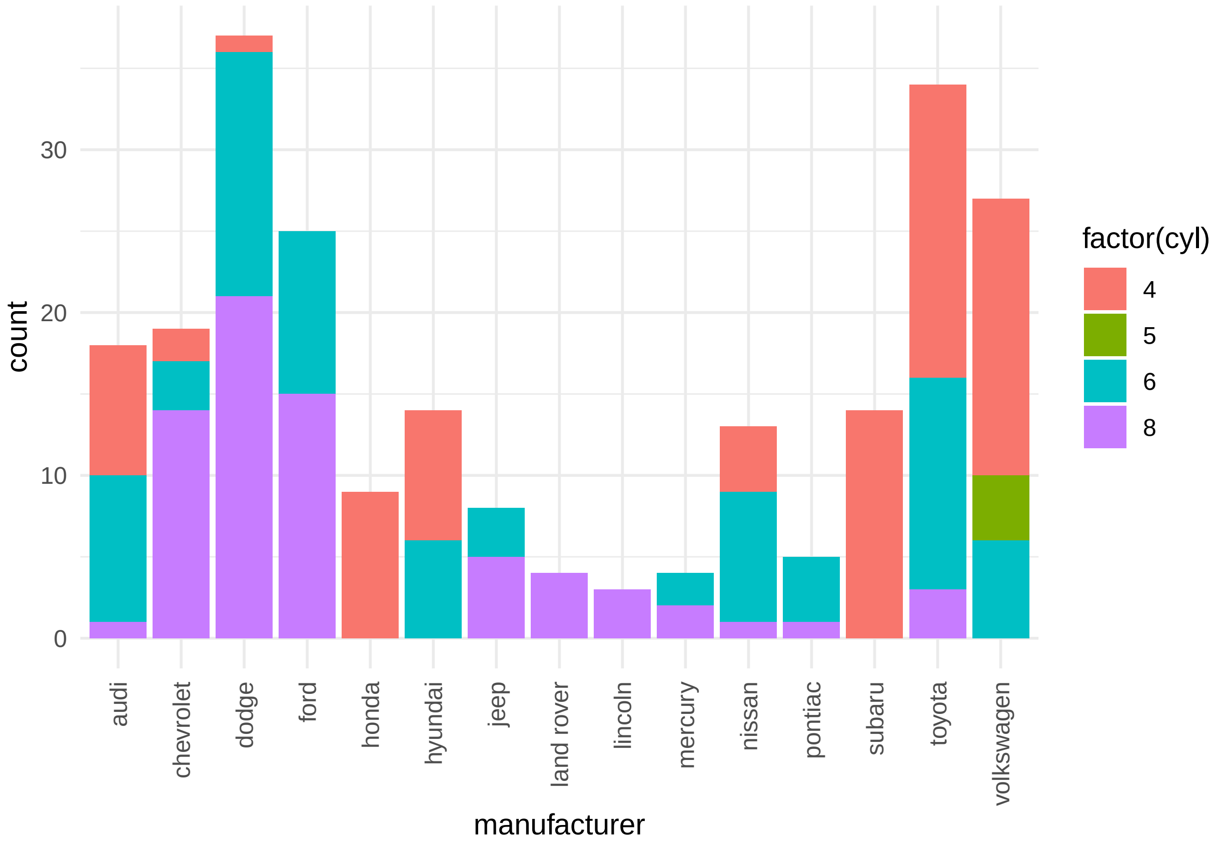
```
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



Ex2: Barplot of car counts per manufacturer with color fill defined by cyl count.

```
# data & aesthetics
pl <- ggplot(mpg,aes(x=manufacturer))

# geometry
#pl2 <- pl + geom_bar(aes(fill=factor(cyl)),position = "fill")
pl2 <- pl + geom_bar(aes(fill=factor(cyl)))

# rotate labels by 90 degrees on x-asis for readability
pl3 <- pl2 + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
print(pl3)
```



Switch now to use the txhousing dataset that comes with ggplot2

```
head(txhousing)
```

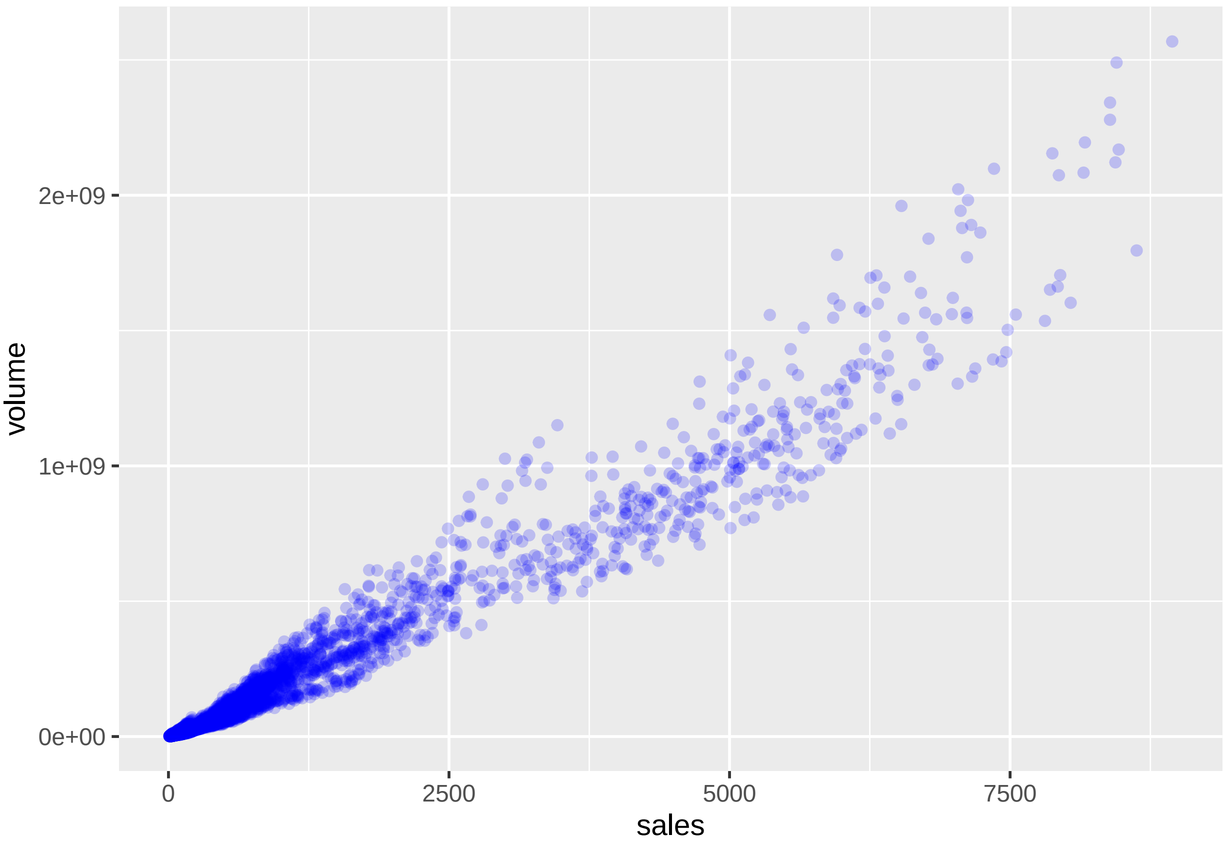
```
## # A tibble: 6 x 9
##   city      year month sales   volume median listings inventory  date
##   <chr>    <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.
## 6 Abilene  2000     6   156 13910000  66900      780       6.6 2000.
```

Ex3: Create a scatterplot of volume versus sales. Afterwards, play around with alpha and color arguments to clarify information.

```
# data & aesthetics
pl <- ggplot(txhousing,aes(x=sales,y=volume))

# geometry
#pl2 <- pl + geom_point(aes(shape=factor(cyl),color=factor(cyl),size=3))
```

```
pl2 <- pl + geom_point(color="blue",alpha=0.2) + theme_gray()

print(pl2)
```

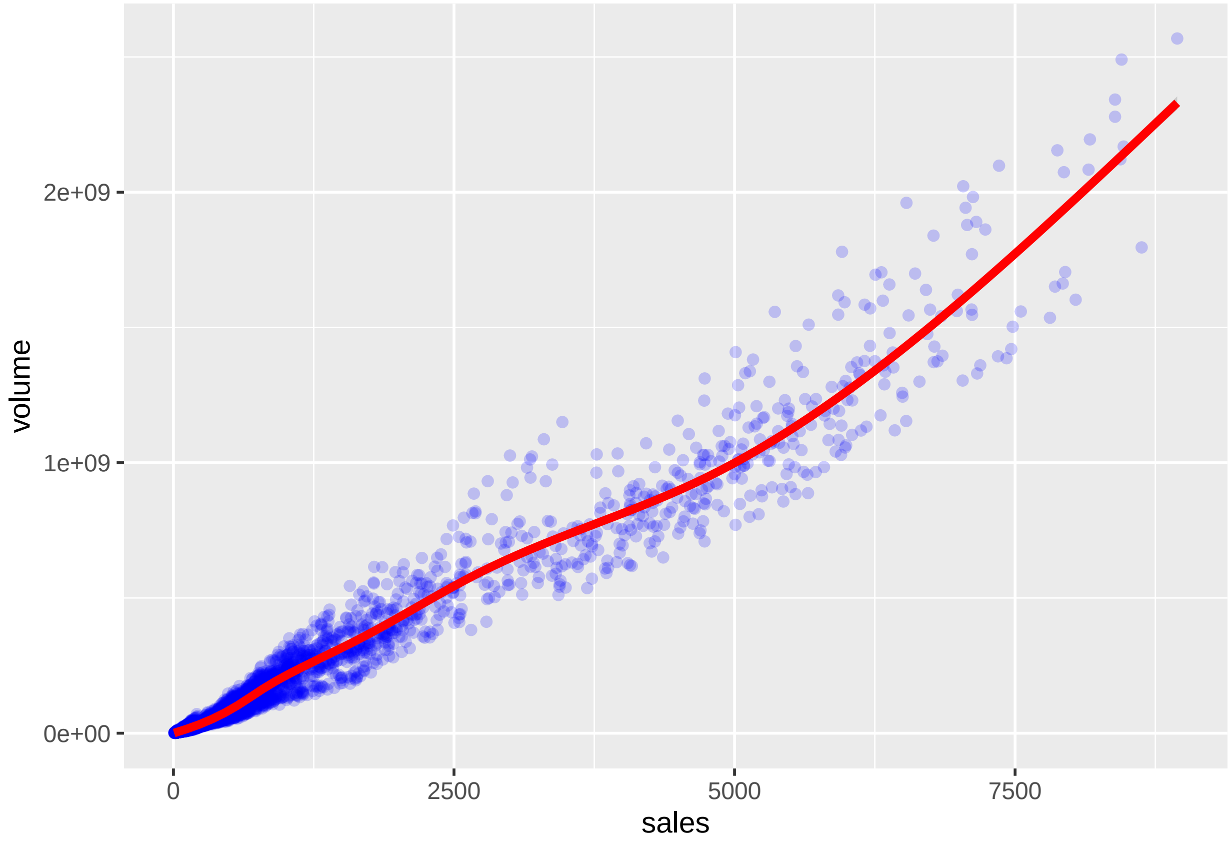
```
## Warning: Removed 568 rows containing missing values (geom_point).
```



Ex4.: Add a smooth fit line to the scatterplot from above. Hint: You may need to look up `geom_smooth()`

```
# data & aesthetics
pl <- ggplot(txhousing,aes(x=sales,y=volume))

# geometry
#pl2 <- pl + geom_point(aes(shape=factor(cyl),color=factor(cyl),size=3))
pl2 <- pl + geom_point(color="blue",alpha=0.2) + theme_gray()

# add a smooth fit line
#pl3 <- pl2 + geom_smooth(method = 'loess',color="red",size=1.5)
pl3 <- pl2 + geom_smooth(color="red",size=1.5,se=TRUE)

print(pl3)
```

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

## Warning: Removed 568 rows containing non-finite values (stat_smooth).
## Warning: Removed 568 rows containing missing values (geom_point).
```


volume
2e+09
1e+09
0e+00
sales