

Data Visualization with ggplot2

ETW2001 Foundations of Data Analysis and Modelling
Manjeevan Singh Seera

Accredited by:



Advanced Signatory:

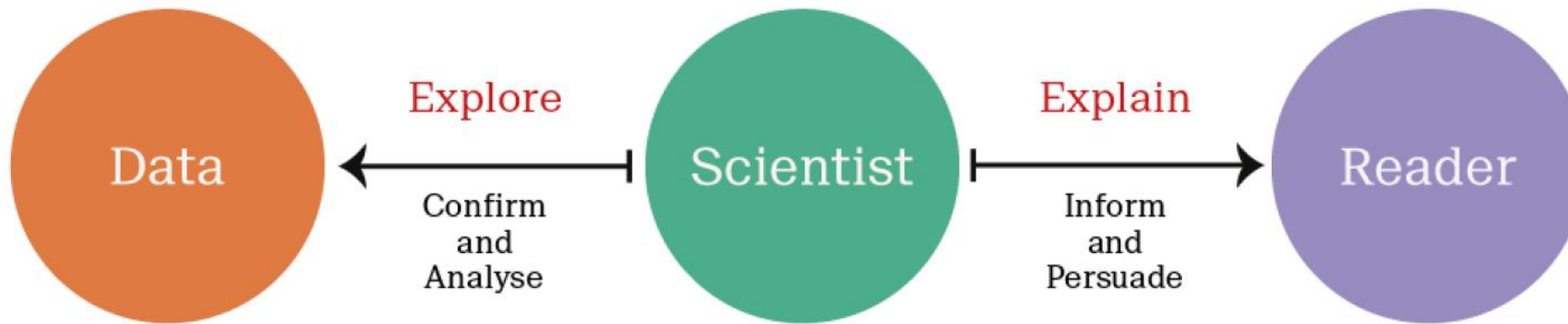


Outline

□ Data Visualization with ggplot2

- Aesthetics mappings
- Plot geometries
- Themes layers

Exploratory *vs.* explanatory



The seven grammatical elements

Element	Description
Data	The data-set being plotted.
Aesthetics	The scales onto which we map our data.
Geometries	The visual elements used for our data.
Themes	All non-data ink.
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Facets	Plotting small multiples.



Prerequisites

For the following slides, focus is on **ggplot2**, one of the core members of the **tidyverse**. We first install the **ggplot2** package.

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

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Prerequisites

In addition to installing **ggplot2**, we need to load **tidyverse** to access functions that we will be using.

```
# Load tidyverse package  
library(tidyverse)
```

— Attaching packages —

✓ tibble	3.0.4	✓ dplyr	1.0.2
✓ tidyr	1.1.2	✓ stringr	1.4.0
✓ readr	1.4.0	✓ forcats	0.5.0
✓ purrr	0.3.4		

— Conflicts —

✗ dplyr::filter()	masks	stats::filter()
✗ dplyr::lag()	masks	stats::lag()

Tidyverse

That one line of code loads the core tidyverse; packages which you will use in almost every data analysis. It also tells you which functions from the tidyverse conflicts with functions in base R.

If you run this code and get the error message: there is no package called 'tidyverse', you'll need to first install it, then run `library()` once again.

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

If we need to be explicit about where a function (or dataset) comes from, we'll use the special form `package::function()`. For example, `ggplot2::ggplot()` tells you explicitly that we're using the `ggplot()` function from the `ggplot2` package.

The mpg data frame

We can test your answer with the `mpg` data frame found in `ggplot2` (`ggplot2::mpg`). A data frame is a rectangular collection of variables (in the columns) and observations (in the rows). `mpg` contains observations collected by the US Environmental Protection Agency on 38 models of car.

`mpg`

A tibble: 234 × 11

manufacturer <chr>	model <chr>	displ <dbl>	year <int>	cyl <int>	trans <chr>	drv <chr>	cty <int>	hwy <int>	fl <chr>	class <chr>
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact

The mpg data frame

Among the variables in mpg are:

- `displ`, a car's engine size, in litres.
- `hwy`, a car's fuel efficiency on the highway, in miles per gallon (mpg). A car with a low fuel efficiency consumes more fuel than a car with a high fuel efficiency when they travel the same distance.

To learn more about mpg, open its help page by running `?mpg`.



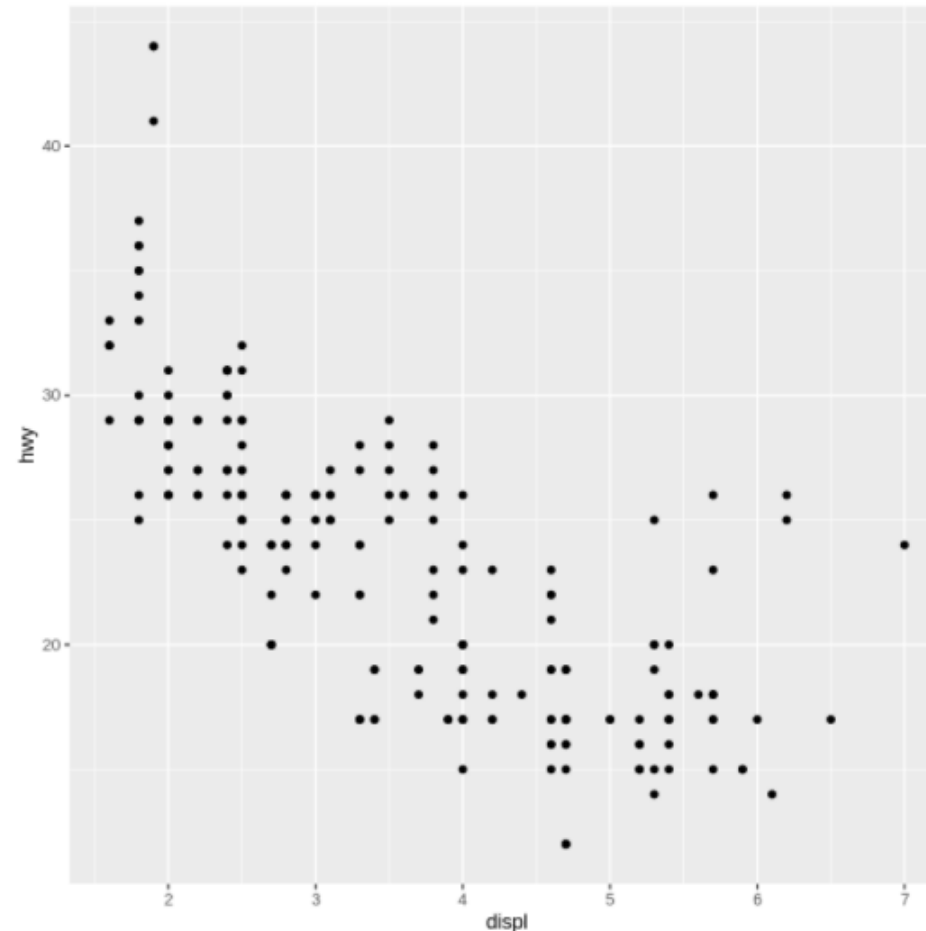
Source: <https://auto.howstuffworks.com/fuel-efficiency>

Creating a ggplot

The plot shows a negative relationship between engine size (displ) and fuel efficiency (hwy). Cars with big engines use more fuel. Is this right about your hypothesis on fuel efficiency and engine size?

With ggplot2, you begin a plot with the function `ggplot()`, which creates a coordinate system that you can add layers to.

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



Creating a ggplot

Let's complete your graph by adding one or more layers to `ggplot()`. `ggplot2` comes with many geom functions that each add a different type of layer to a plot.

Geometric objects (geoms) are the visual representations of (subsets of) observations. You'll use two common geom layer functions:

- `geom_point()` adds points (as in a scatter plot).
- `geom_smooth()` adds a smooth trend curve.

Each geom function in `ggplot2` takes a mapping argument. This defines how variables in your dataset are mapped to visual properties.

The mapping argument is always paired with `aes()`, and the x and y arguments of `aes()` specify which variables to map to the x and y axes.

A graphing template

Now, turn this code into a reusable template for making graphs with ggplot2. To make a graph, replace the bracketed sections in the code below with a dataset, a geom function, or a collection of mappings.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

The remaining slides will show how to complete and extend this template to make different types of graphs. We will begin with the `<MAPPINGS>` component.

Outline

✓ Data Visualization with ggplot2

☐ **Aesthetics mappings**

☐ Plot geometries

☐ Themes layers

All about aesthetics: color, shape and size

We will apply the visible aesthetics to categorical variable — the cylinders in `mtcars`, `cyl`.

These are the aesthetics you can consider within `aes()`: `x`, `y`, `color`, `fill`, `size`, `alpha`, `labels` and `shape`.

One common convention is that you don't name the `x` and `y` arguments to `aes()`, since they almost always come first, but you do name other arguments.

Column `fcyl` is categorical, which will be transformed to a factor.

All about aesthetics: color *vs.* fill

Color aesthetic changes the outline of a geom and the fill aesthetic changes the inside. `geom_point()` is an exception: you use color (not fill) for the point color.

Some shapes however, have special behavior.

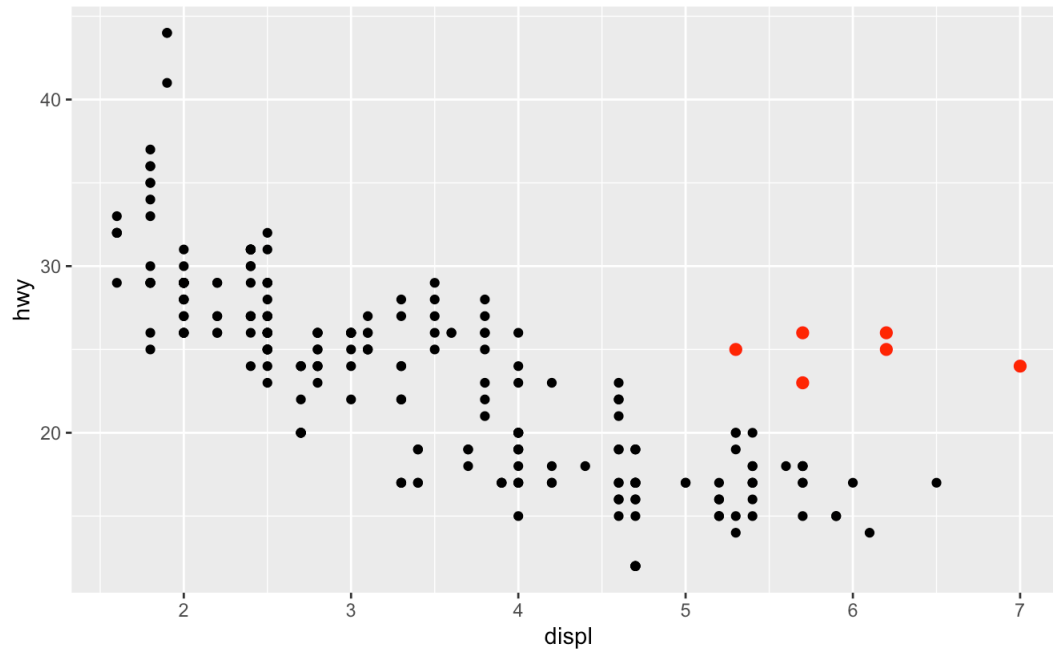
- Default `geom_point()` uses `shape = 19`: a solid circle.
- Alternative `shape = 21`: a circle that allow you to use both fill for the inside and color for the outline.

This is lets you to map two aesthetics to each point.

All shape values are described on the `points()` help page.

Aesthetic mappings

In the plot below, one group of points (highlighted in red) seems to fall outside of the linear trend. These cars have a higher mileage than you might expect. How can you explain these cars?



Source: <https://r4ds.had.co.nz/>

Aesthetic mappings

Let's hypothesize that the cars are **hybrids**.

One way to test this hypothesis is to look at the class value for each car. The class variable of the mpg dataset classifies cars into groups such as:

- Compact,
- Midsize,
- SUV.

If the outlying points are hybrids, they should be classified as one of the above categories.

Variable

Adding a third variable to a 2-dimensional scatterplot can be done by mapping it to an aesthetic.

An aesthetic is a visual property of the objects in your plot. Aesthetics include things like the size, the shape, or the color of your points. You can display a point in different ways by changing the values of its aesthetic properties. Since we already use the word “value” to describe data, let’s use the word “level” to describe aesthetic properties.

Here we change the levels of a point’s size, shape, and color to make the point small, triangular, or blue:

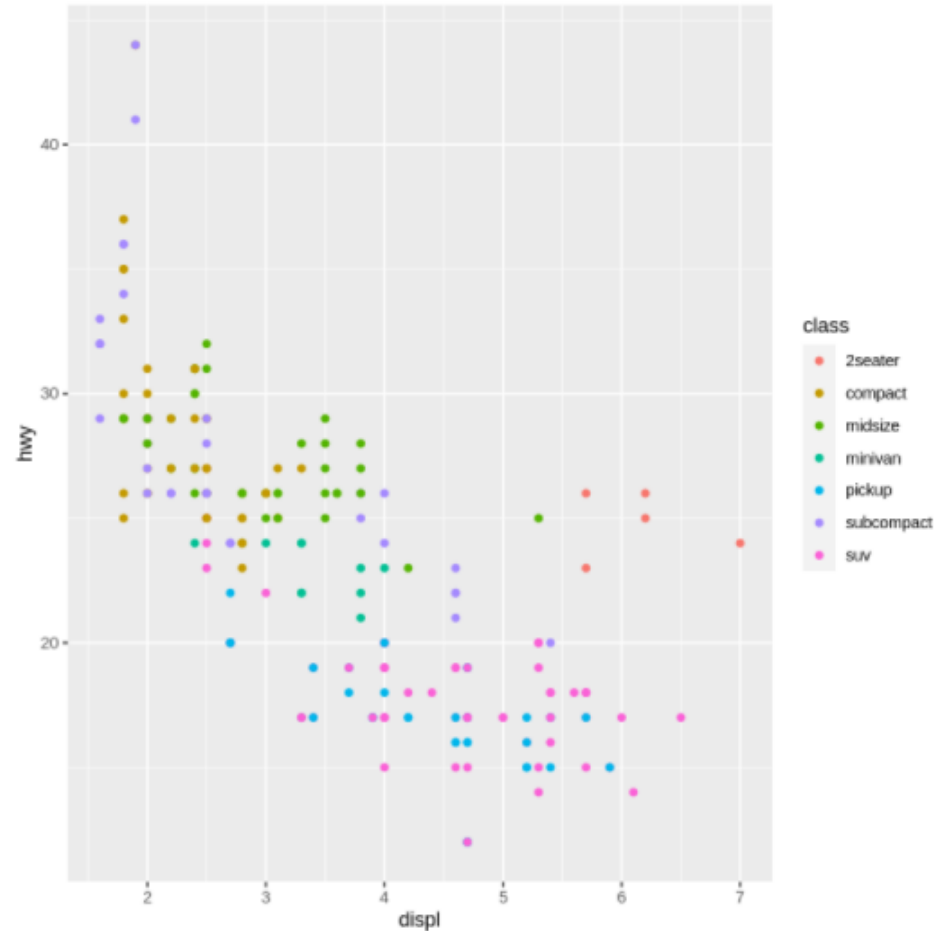


Variable

We can convey information about your data by mapping the aesthetics in your plot to the variables in your dataset.

For example, we can map the colors of your points to the class variable to reveal the class of each car.

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



Aesthetic

To map an aesthetic to a variable, associate the name of the aesthetic to the name of the variable inside `aes()`. `ggplot2` will:

- Automatically assign a unique level of the aesthetic (here a unique color) to each unique value of the variable, a process known as scaling,
- Add a legend that explains which levels correspond to which values.

In previous example, we mapped class to the color aesthetic. The exact size of each point would reveal its class affiliation. We get a warning here, because mapping an unordered variable (class) to an ordered shape is not a good idea.

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

Warning message:
“Using alpha for a discrete variable is not advised.”

Aesthetic

We could have mapped class:

- To the alpha aesthetic (example on left), which controls the transparency of the points, or
- To the shape aesthetic (example on right), which controls the shape of the points.

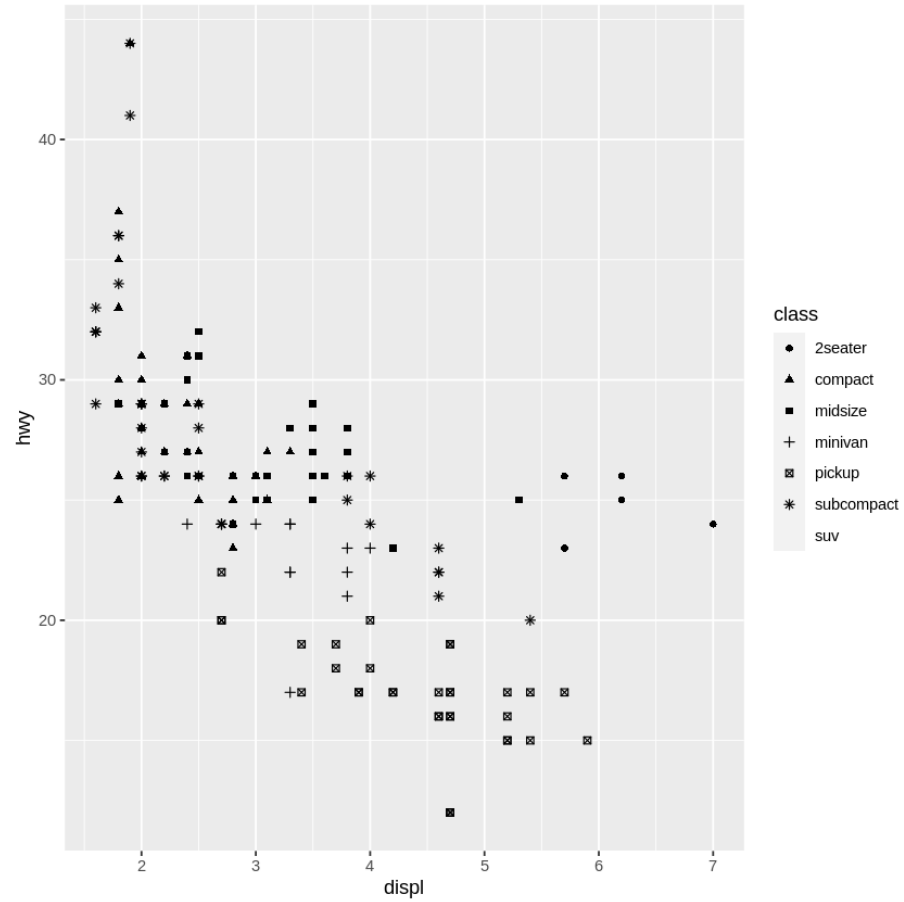
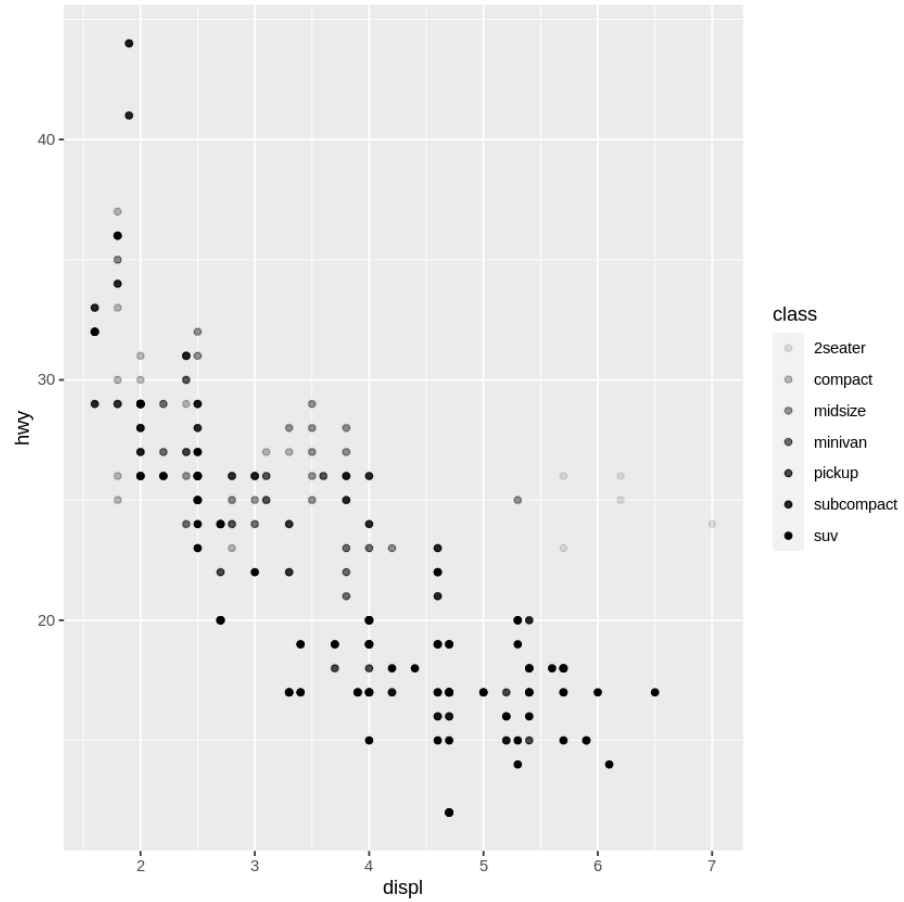
```
# Left
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))

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

Aesthetic



MONASH
University
MALAYSIA



MONASH
BUSINESS

Aesthetic

For each aesthetic, you use `aes()` to associate the name of the aesthetic with a variable to display.

- It gathers together each of the aesthetic mappings used by a layer and passes them to the layer's mapping argument.
- It highlights a useful insight about x and y: the x and y locations of a point are themselves aesthetics, visual properties that you can map to variables to display information about the data.

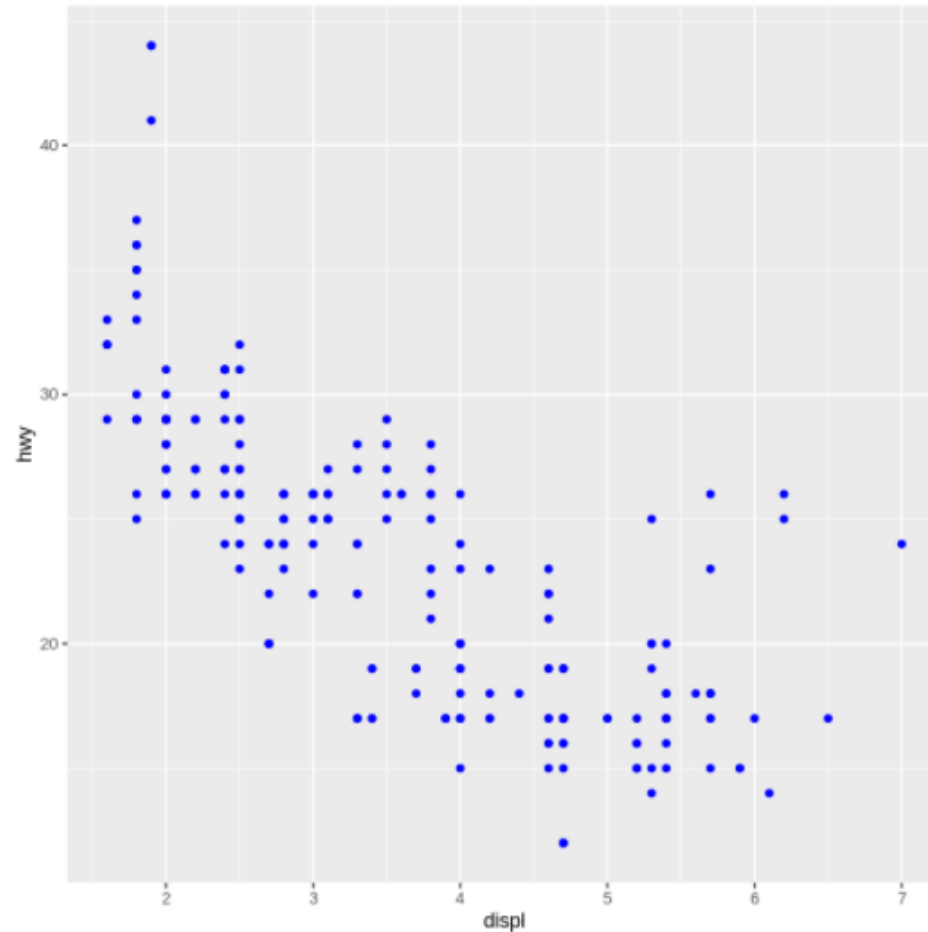
Once you map an aesthetic, ggplot2 takes care of the rest.

- It selects a reasonable scale to use with the aesthetic, and it constructs a legend that explains the mapping between levels and values.
- For x and y aesthetics, ggplot2 does not create a legend, but it creates an axis line with tick marks and a label. The axis line acts as a legend; it explains the mapping between locations and values.

Aesthetic

We can make all of the points in the plot to be blue.

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



MONASH
University
MALAYSIA

MONASH
BUSINESS

Aesthetic

Here, the color **doesn't convey information** about a variable, but only changes the **appearance** of the plot. To set an aesthetic manually, set the aesthetic by name as an argument of your geom function; i.e. it goes outside of `aes()`.


























You'll need to pick a level that makes sense for that aesthetic:

- The name of a color as a character string,
- The size of a point in mm,
- The shape of a point as a number.

Aesthetic

R has 25 built in shapes that are identified by numbers. There are some seeming duplicates: for example, 0, 15, and 22 are all squares. Main difference comes from `colour` and `fill` aesthetics.

The hollow shapes (0 – 14) have a border determined by `colour`; the solid shapes (15 – 20) are filled with `colour`; the filled shapes (21 – 24) have a border of `colour` and are filled with `fill`.

 0	 4	 10	 15	 22
 1	 6	 11	 16	 21
 2	 7	 12	 17	 24
 5	 8	 13	 18	 23
 3	 9	 14	 19	 20

Outline

- ✓ Data Visualization with ggplot2
 - ✓ Aesthetics mappings
 - ☒ **Plot geometries**
 - ☐ Themes layers



MONASH
University
MALAYSIA

MONASH
BUSINESS

Overplotting 1: Large datasets

Scatter plots (using `geom_point()`) are intuitive, easily understood, and very common, but we must always consider overplotting, particularly in the following four situations:

- Large datasets
- Aligned values on a single axis
- Low-precision data
- Integer data

Typically, alpha blending (i.e. adding transparency) is recommended when using solid shapes. Alternatively, you can use opaque, hollow shapes.

Small points are suitable for large datasets with regions of high density (lots of overlapping).

Overplotting 2: Aligned values

Let's take a look at another case where we should be aware of overplotting:
Aligning values on a single axis.

This occurs when one axis is continuous and the other is categorical, which can be overcome with some form of jittering.

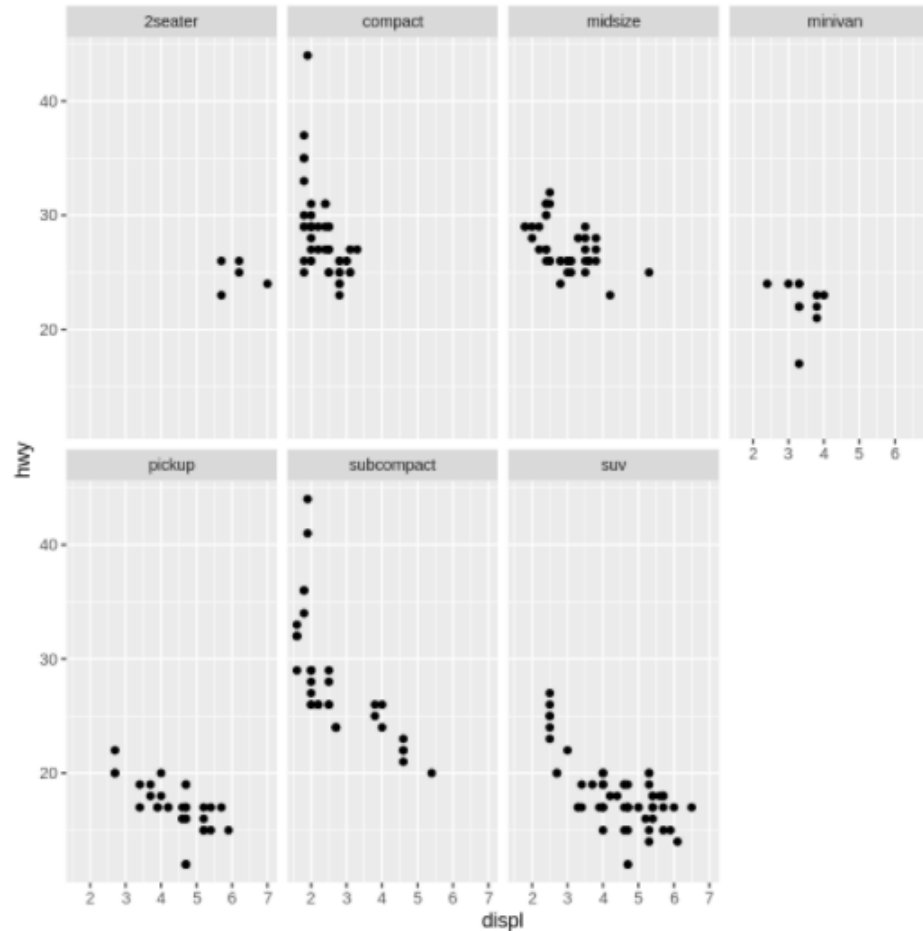
In the mtcars data set, fam and fcyl are categorical variants of cyl and am.

Facets

For categorical variables, it is useful to split your plot into facets, subplots that each display one subset of the data.

To facet your plot by a single variable, use `facet_wrap()`. The first argument of `facet_wrap()` should be a formula, which you create with `~` followed by a variable name. The variable that you pass to `facet_wrap()` should be discrete.

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  facet_wrap(~ class, nrow = 2)
```



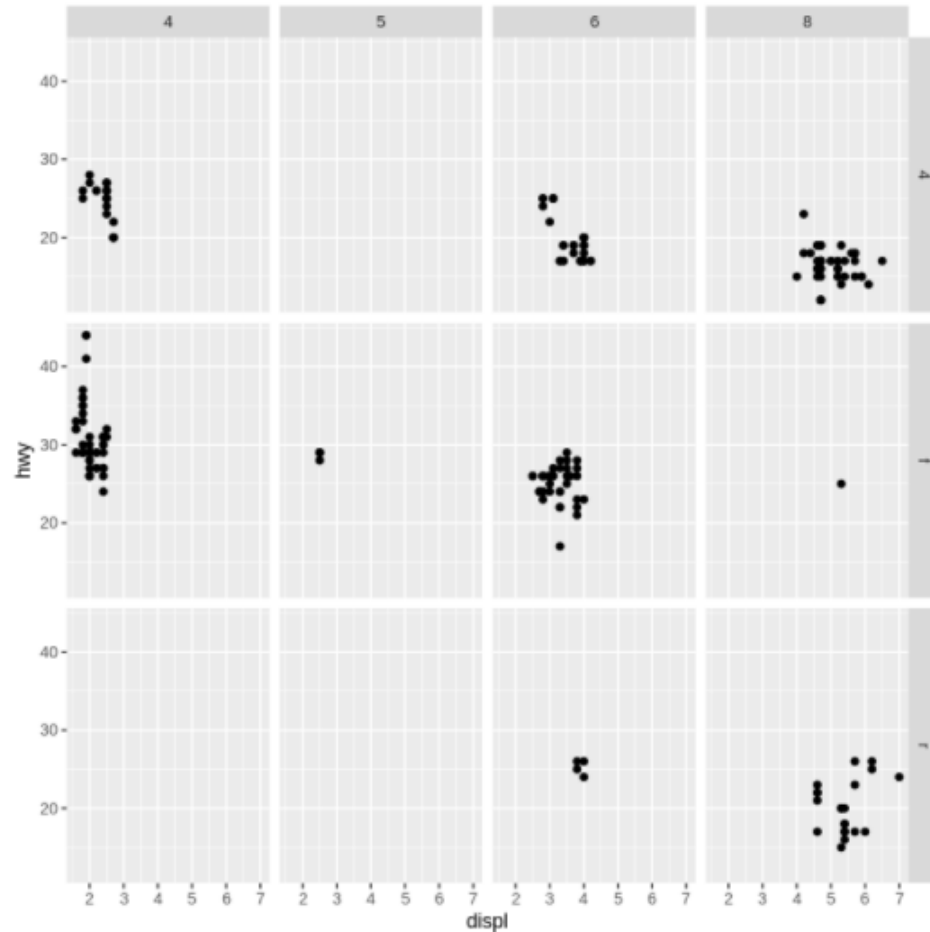
Facets



To facet your plot on the combination of two variables, add `facet_grid()` to your plot call. The first argument of `facet_grid()` is also a formula. This time the formula should contain two variable names separated by a `~`.

If you prefer to not facet in the rows or columns dimension, use a `.` instead of a variable name, e.g. `+ facet_grid(. ~ cyl)`.

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  facet_grid(drv ~ cyl)
```



Geometrical object

People often describe plots by the type of geom that the plot uses. As an example, bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms. Scatterplots break the trend; they use the point geom.

Plot on the left uses the point geom, and the plot on the right uses the smooth geom, a smooth line fitted to the data.

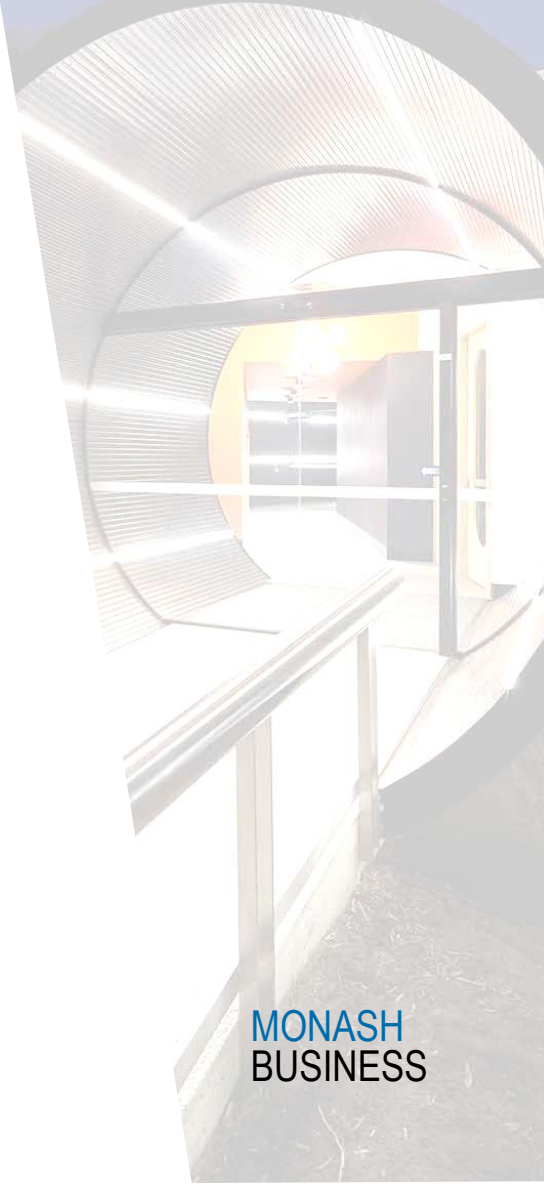
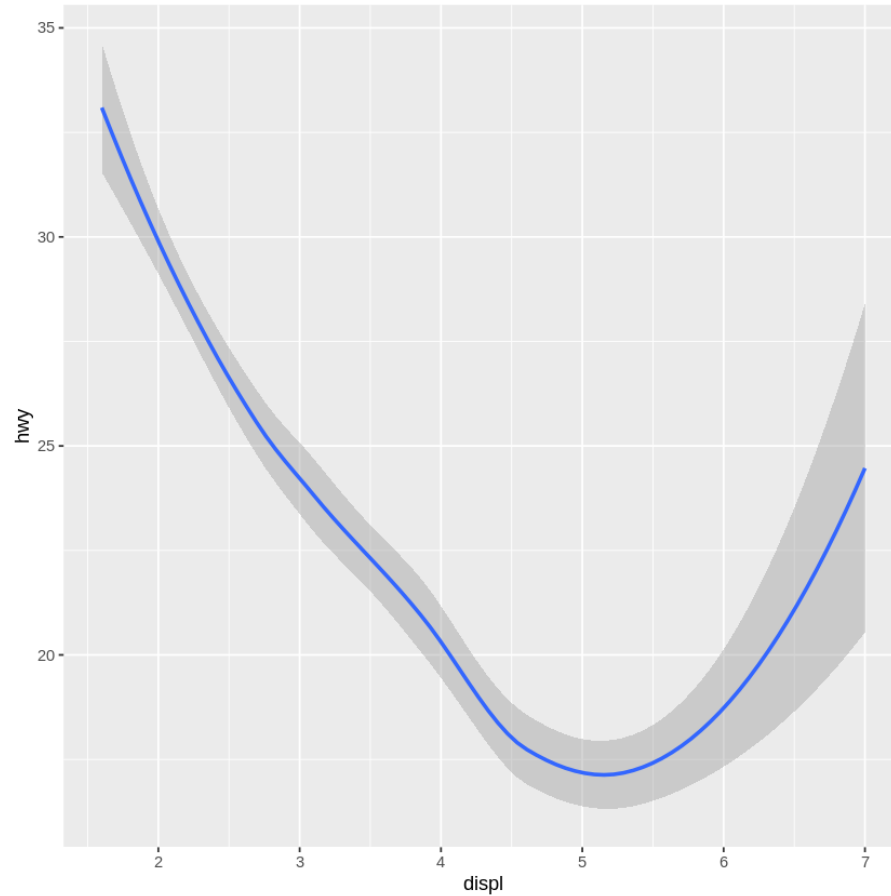
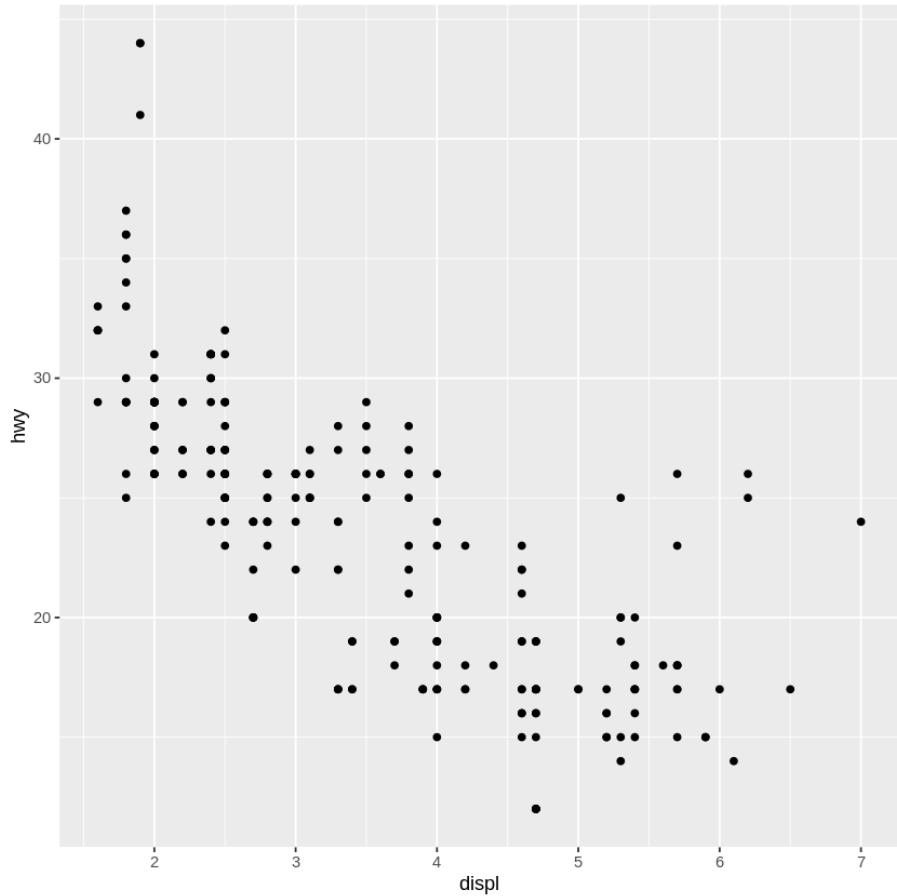
```
# left
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))

# right
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

Geometrical object



MONASH
University
MALAYSIA



MONASH
BUSINESS

Geometrical object

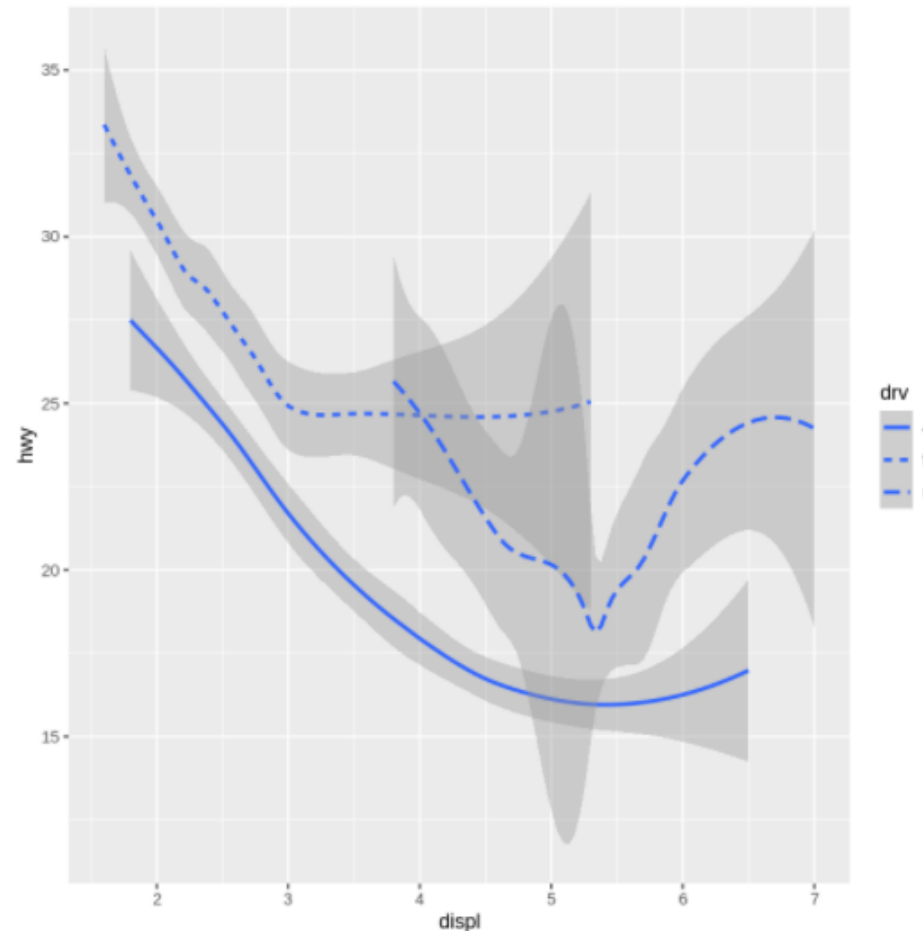
Every geom function in ggplot2 takes a mapping argument.

Not every aesthetic works with every geom. You could set the shape of a point, but you couldn't set the "shape" of a line.

`geom_smooth()` will draw a different line, with a different linetype, for each unique value of the variable that you map to linetype.

```
ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```

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



Smoothed conditional means

Here `geom_smooth()` separates the cars into three lines based on their `drv` value, which describes a car's drivetrain.

One line describes all of the points with a 4 value, one line describes all of the points with an f value, and one line describes all of the points with an r value.

- 4 stands for four-wheel drive,
- f for front-wheel drive,
- r for rear-wheel drive.

ggplot2 provides over 40 geoms. To learn more about any single geom, use help: `?geom_smooth`.

Smoothed conditional means

`geom_smooth()` use a single geometric object to display multiple rows of data. For these geoms, you can set the group aesthetic to a categorical variable to draw multiple objects.

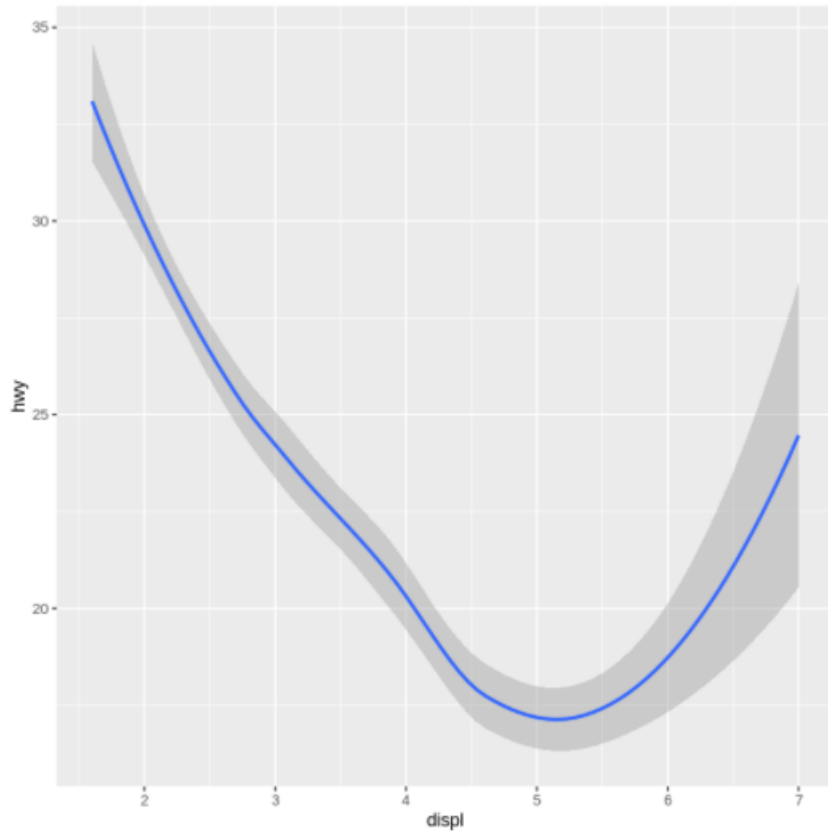
ggplot2 will draw a separate object for each unique value of the grouping variable. In practice, ggplot2 will automatically group the data for these geoms whenever you map an aesthetic to a discrete variable (as in the linetype example).

It is convenient to rely on this feature because the group aesthetic by itself does not add a legend or distinguishing features to the geoms.

Display multiple rows of data

```
# geom_smooth uses single geometric object to display multiple rows of data  
ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

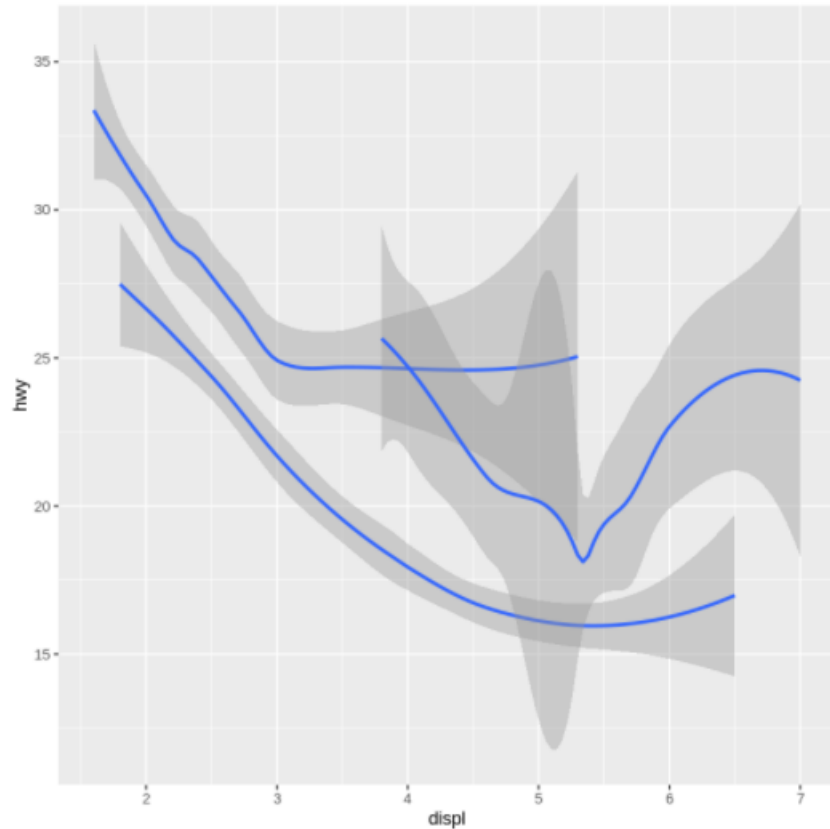
`geom_smooth()` using method = 'loess' and formula 'y ~ x'



Draw multiple objects

```
# Group aesthetic set to a categorical variable (drive) to draw multiple objects  
# ggplot2 draws separate object for each unique value of grouping variable  
ggplot(data = mpg) +  
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))
```

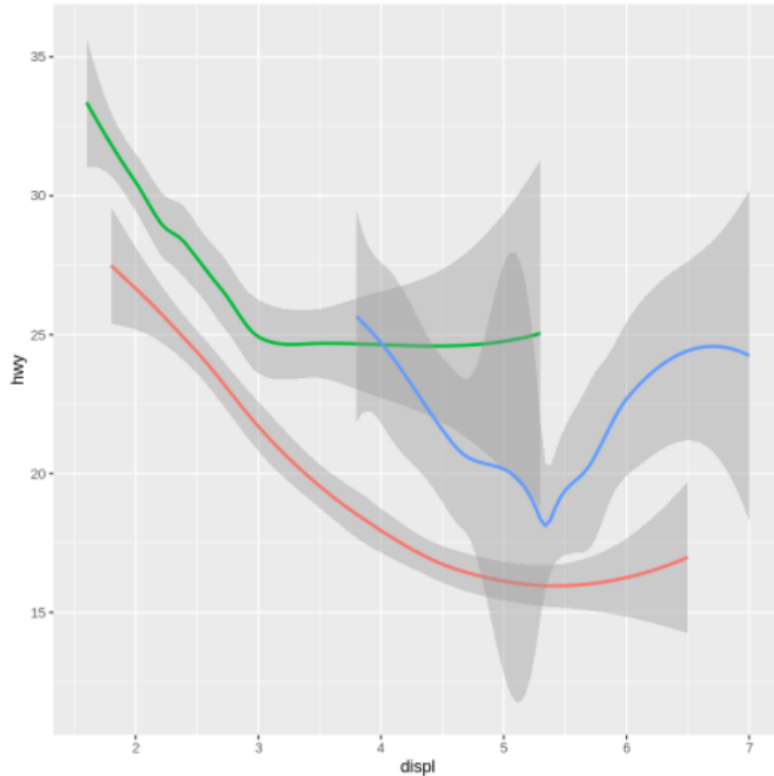
`geom_smooth()` using method = 'loess' and formula 'y ~ x'



Different colours for grouping variables

```
# Different color for each value of grouping variable
ggplot(data = mpg) +
  geom_smooth(
    mapping = aes(x = displ, y = hwy, color = drv),
    show.legend = FALSE # no legend
  )
```

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



Smoothed conditional means

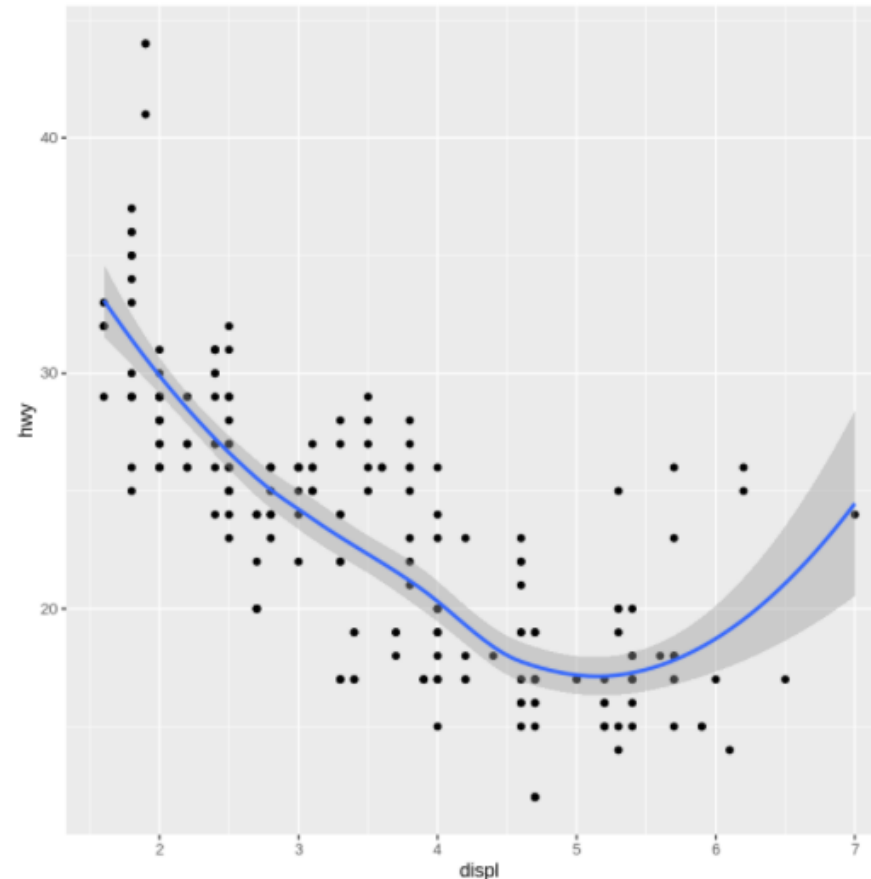


To display multiple geoms in the same plot, add multiple geom functions to `ggplot()`.

Notice you that you need to type the same thing twice? There is some duplication in our code.

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

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



Smoothed conditional means

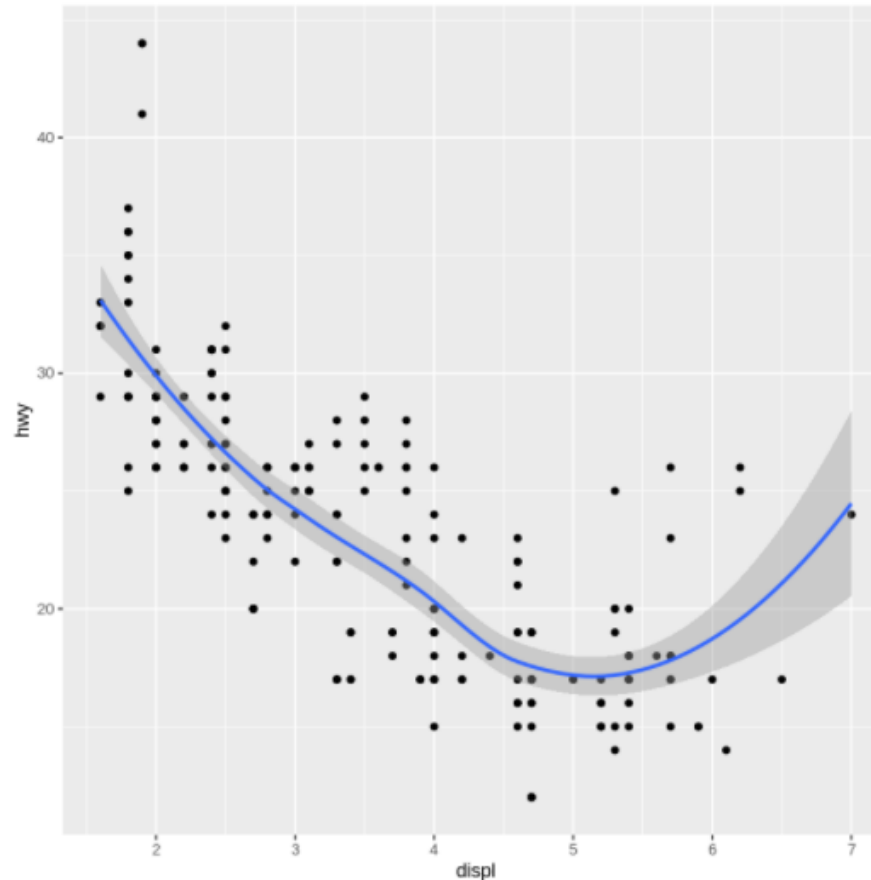


Imagine if you wanted to change the y-axis to display `cty` instead of `hwy`. You'd need to change the variable in two places, and you might forget to update one.

You can avoid this type of repetition by passing a set of mappings to `ggplot()`. `ggplot2` will treat these mappings as global mappings that apply to each geom in the graph. In other words, this code will produce the same plot as the previous code.

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

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



Smoothed conditional means



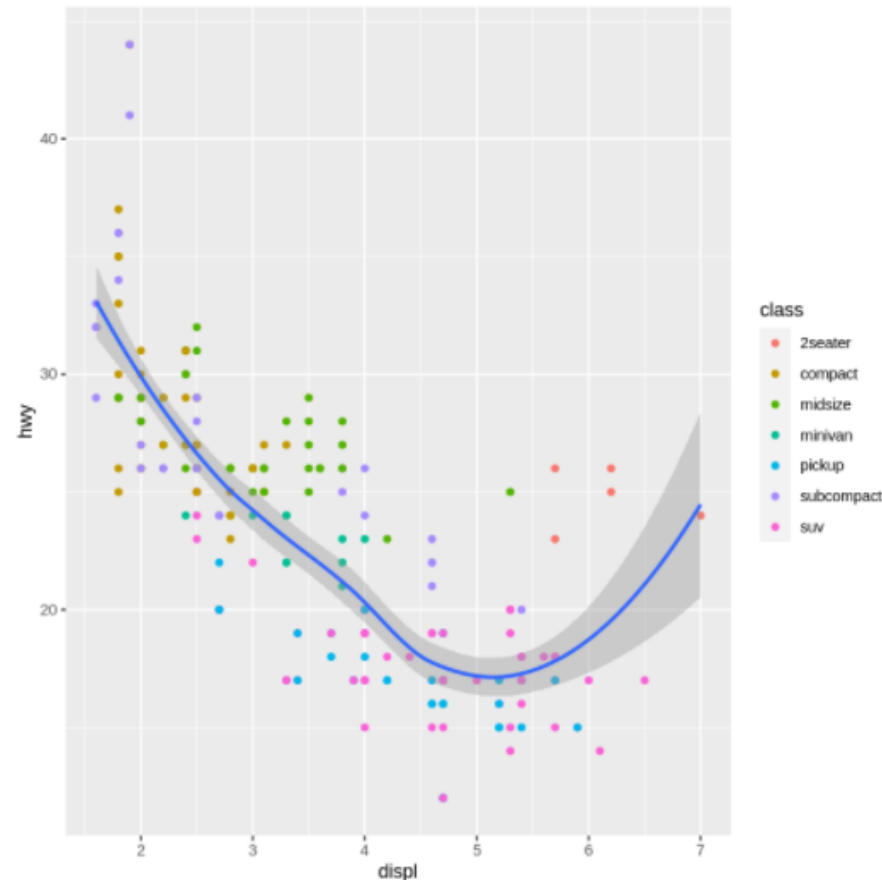
If you place mappings in a geom function, ggplot2 will treat them as local mappings for the layer.

It will use these mappings to extend or overwrite the global mappings for that layer only.

This makes it possible to display different aesthetics in different layers.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = class)) +  
  geom_smooth()
```

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



Smoothed conditional means

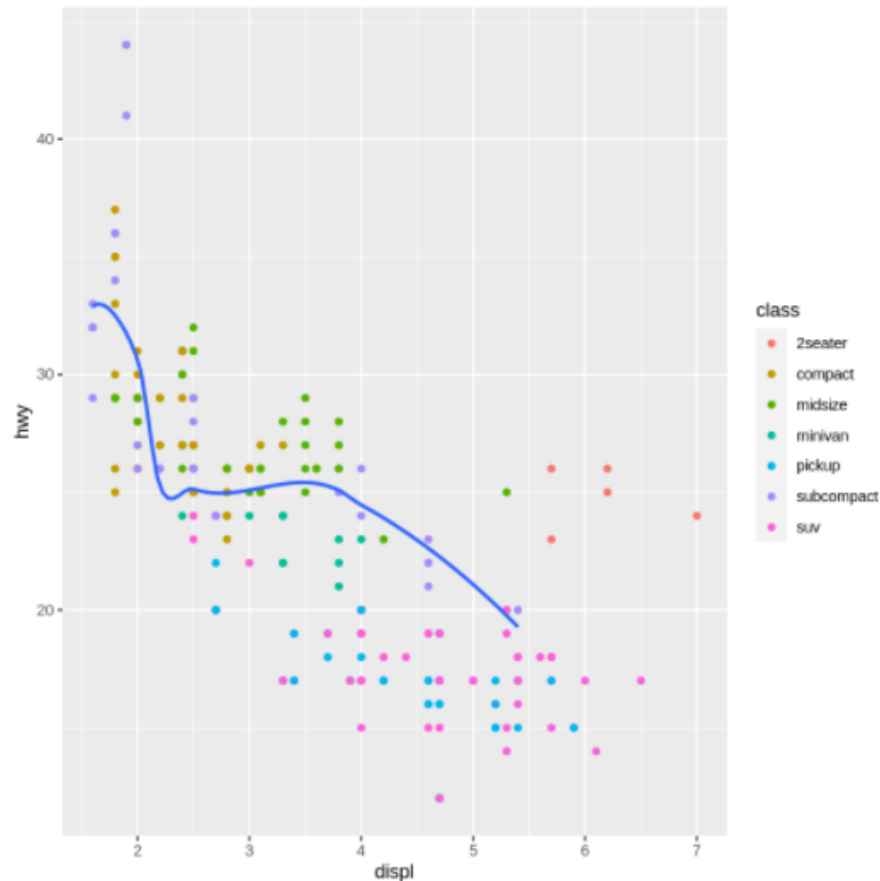
You can use the same idea to specify different data for each layer.

Here, our smooth line displays just a subset of the mpg dataset, the subcompact cars.

The local data argument in `geom_smooth()` overrides the global data argument in `ggplot()` for that layer only.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = class)) +  
  geom_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)
```

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



Outline

- ✓ Data Visualization with ggplot2
 - ✓ Aesthetics mappings
 - ✓ Plot geometries
 - ❑ **Themes layers**

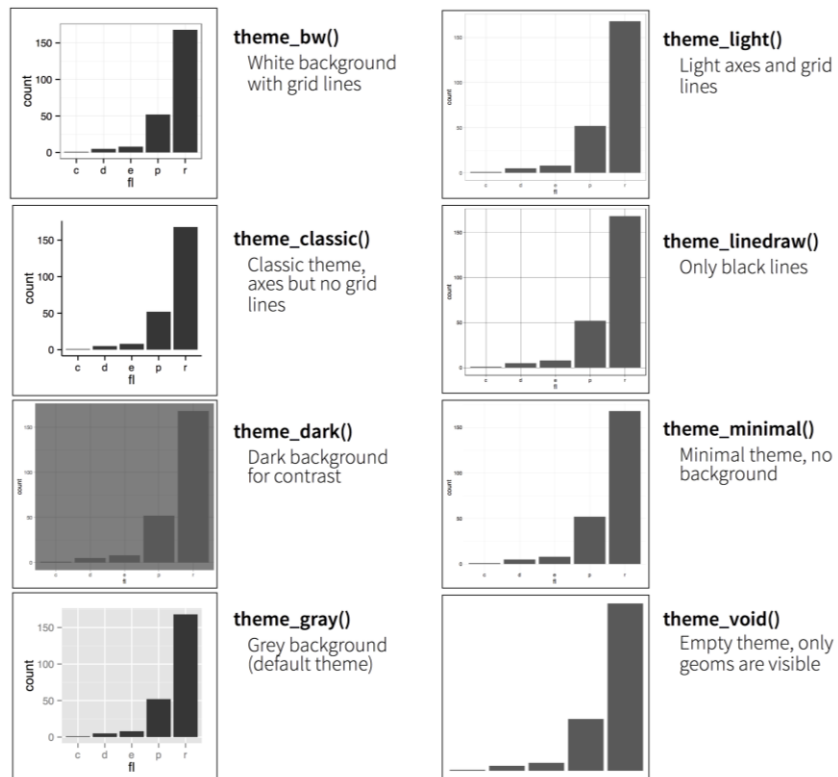


MONASH
University
MALAYSIA

MONASH
BUSINESS

Themes

ggplot2 includes eight themes by default. Many more are included in add-on packages like [ggthemes](#).



Default themes

Why the default theme has a grey background?

- This was a deliberate choice because it puts the data forward while still making the grid lines visible.
- The white grid lines are visible (which is important because they significantly aid position judgements), but they have little visual impact and we can easily tune them out.
- The grey background gives the plot a similar typographic colour to the text, ensuring that the graphics fit in with the flow of a document without jumping out with a bright white background. It creates a continuous field of colour which ensures that the plot is perceived as a single visual entity.

It's also possible to control individual components of each theme, like the size and colour of the font used for the y axis.

Default themes

Let's work on a sample. We first need to install the package.

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

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

For the following plots, the code below is used.

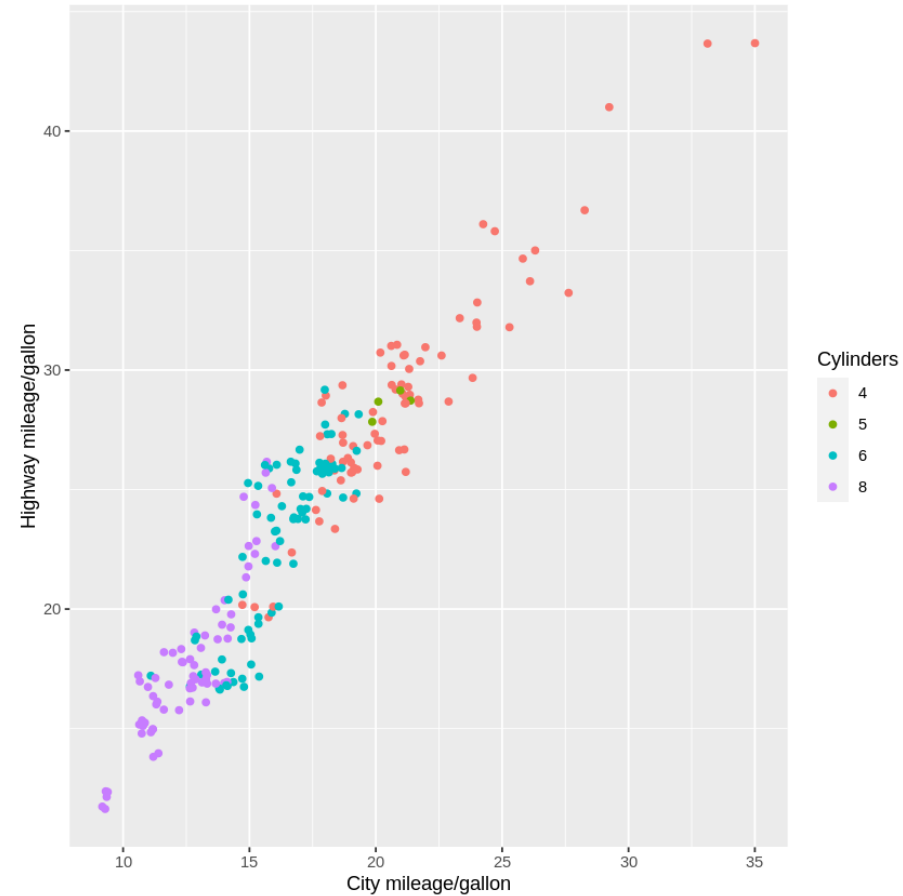
```
p = ggplot(mpg, aes(x = cty, y = hwy, color = factor(cyl))) +  
  geom_jitter() +  
  labs(  
    x = "City mileage/gallon",  
    y = "Highway mileage/gallon",  
    color = "Cylinders"  
  )
```

Default theme

Output of the default theme.

p

Standard grey background.

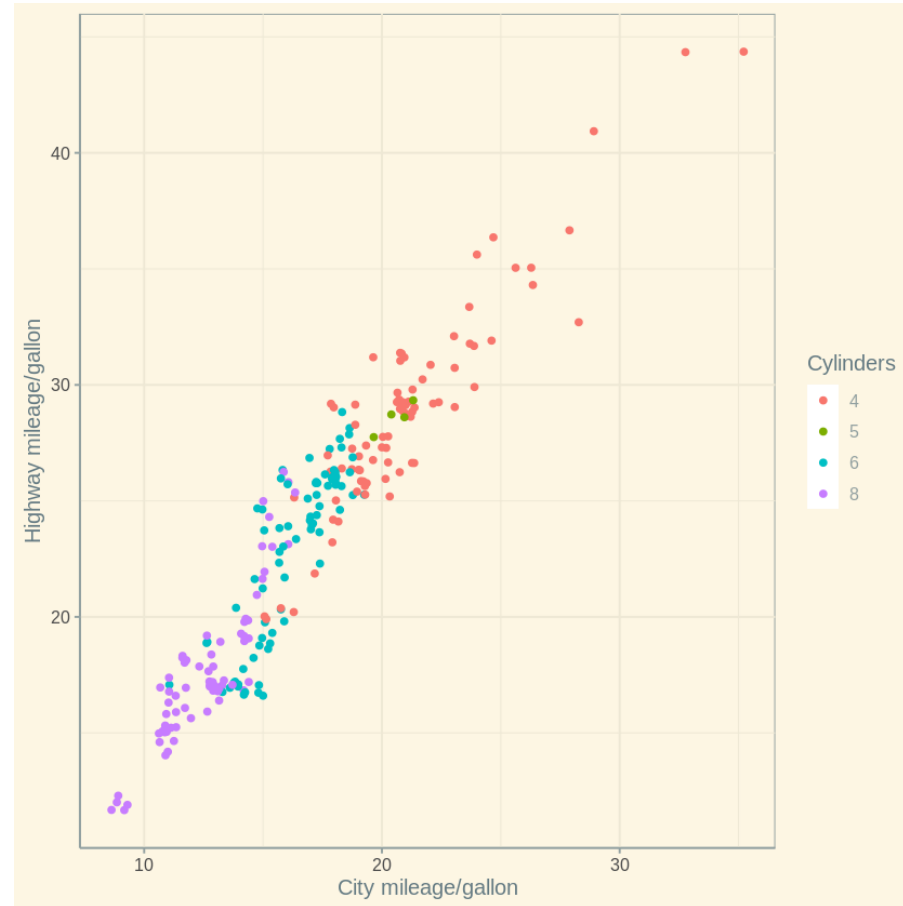


MONASH
University
MALAYSIA

MONASH
BUSINESS

ggthemes: Solarized

Output of solarized theme.
`p + theme_solarized()`

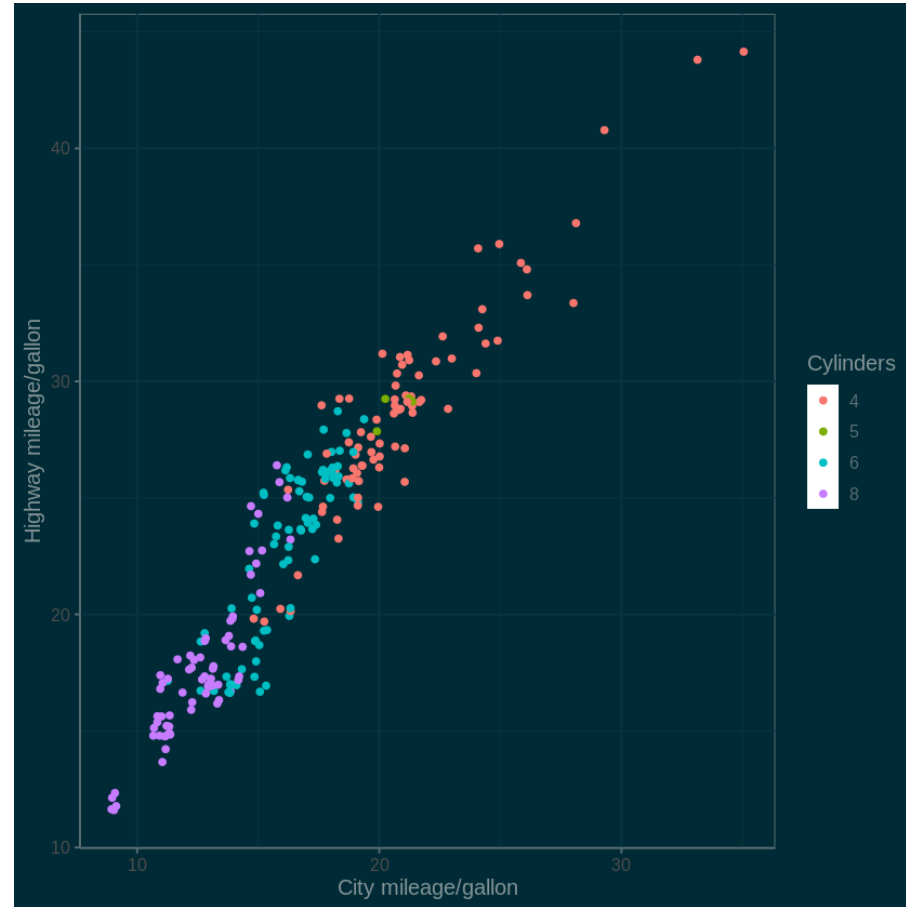


ggthemes: Solarized Dark

Output of solarized dark.

```
p + theme_solarized(light=FALSE)
```

Too dark maybe?

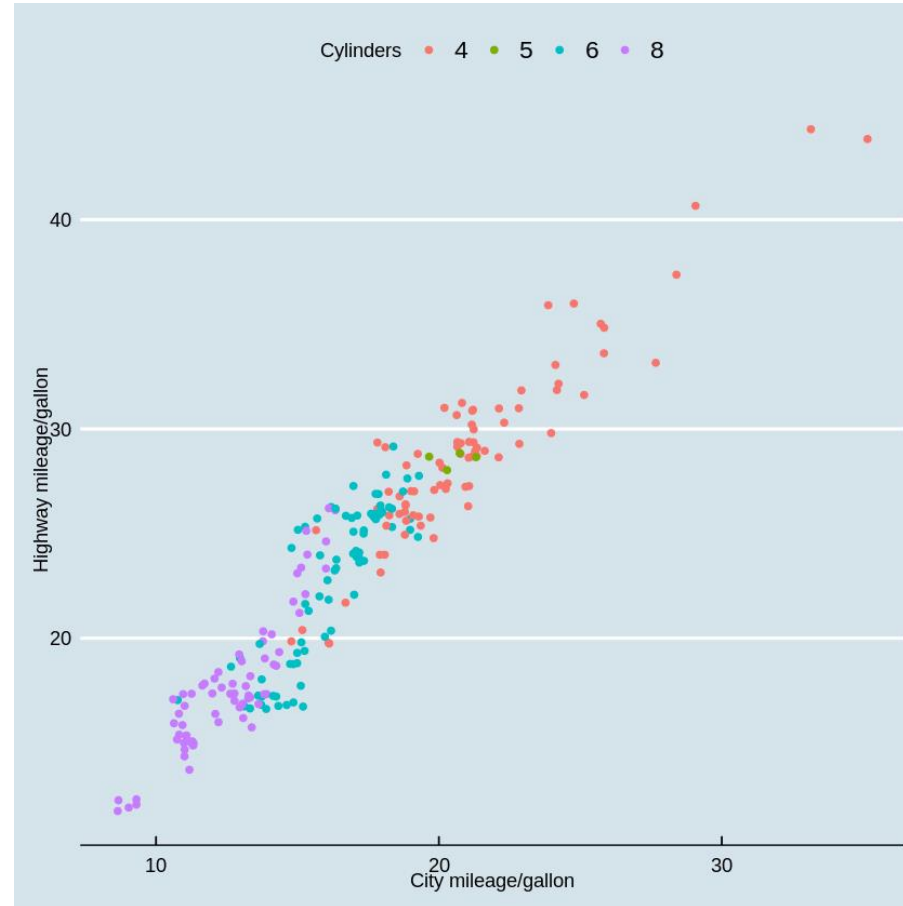


ggthemes: The Economist

Output of The Economist theme.

```
p + theme_economist()
```

Looks neat!



Saving your plots

There are two main ways to get your plots out of R and into your document:
`ggsave()` will save the most recent plot to disk.

```
ggsave("my-plot.pdf")
```

Saving 6.67 x 6.67 in image

This will save the using the default dimensions. To know more, you can find out using `?ggsave()` in the documentation.

THANK YOU

FIND OUT MORE AT [MONASH.EDU.MY](https://monash.edu.my)
LIKE [@MONASH UNIVERSITY MALAYSIA](https://www.facebook.com/MONASHMALAYSIA) ON FACEBOOK
FOLLOW [@MONASHMALAYSIA](https://twitter.com/MONASHMALAYSIA) ON TWITTER

