Making plots with `ggplot2`: Barplots and Scatterplots

Emily Malcolm-White

`ggplot2` is a package built within the `tidyverse` package for creating awesome graphs!

Figure 1: Artwork by @allisonhorst

```
library(tidyverse)
```

```
#Import the can_lang dataset
can_lang <- read.csv("https://raw.githubusercontent.com/ttimbers/canlang/master/inst/extda
```

Recall our Top 10 example:

This code gave a list of 10 Aboriginal Languages which have the most number of people who speak them as their mother tongue:

```
ten_lang <- can_lang %>%  #<1>
  filter(category == "Aboriginal languages") %>% #<2>
  arrange(desc(mother_tongue)) %>% #<3>
  select(language, mother_tongue) %>%  #<4>
  slice(1:10) #<5>
```

① Start with the `can_lang` dataset
② Filter for aboriginal languages only
③ arrange the rows from highest number of people who speak the language as their mother tongue, to the lowest number of people who speak the language as their mother tongue
④ only include the language and mother_tongue columns
⑤ only include the top 10 rows

Barplots

Suppose we wanted to display this information in a barplot instead of in a table. Let's take a look at the `ggplot` syntax:

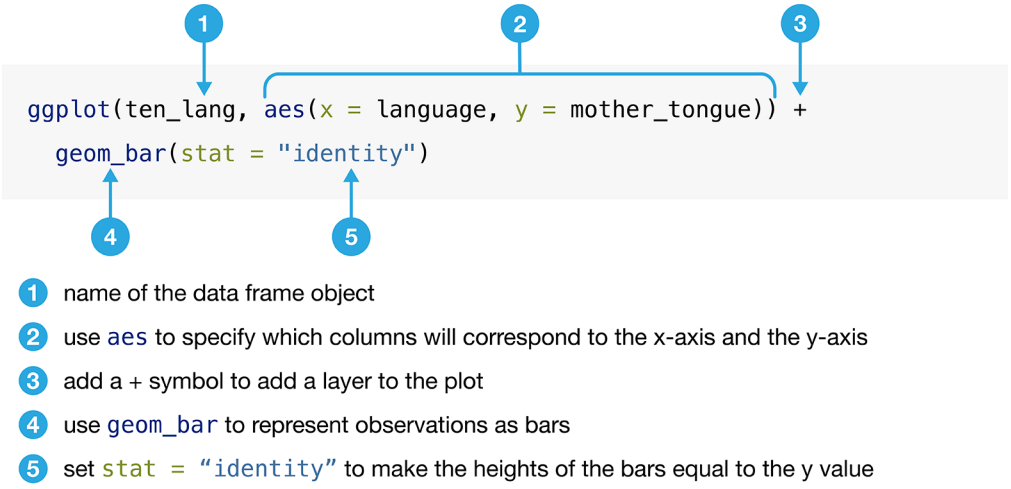


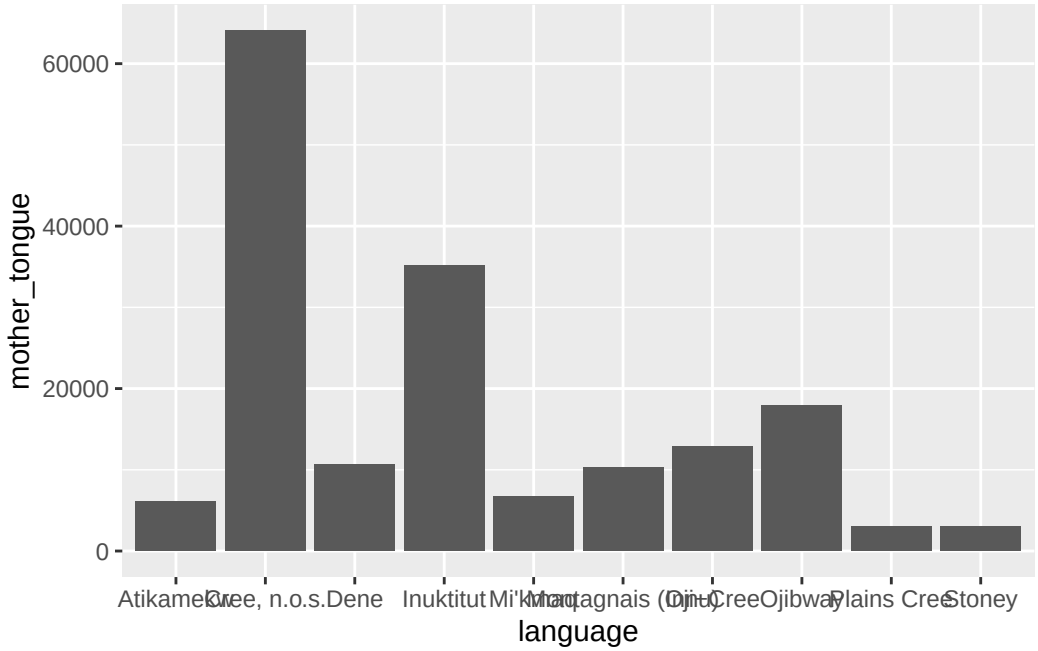
Figure 2: Credit: https://github.com/UBC-DSCI/introduction-to-datascience/

```
ten_lang %>% #<1>
  ggplot(aes(x = language, y = mother_tongue)) + #<2>
  geom_bar(stat = "identity") #<3>
```

① Begin with the `ten_lang` dataset

② Create a plot – the x-axis contains the languages and the y-axis contains the number of people who speak the language as their mother tongue. This sets up the coordinate system, but no visualization appears yet.

③ add a layer with a barplot. The height of the bars should simply be the number of people who speak the language as their mother tongue. Without `stat = "identity"`, `geom_bar()` defaults to `stat = "count"`, which means it counts rows instead of using a y-variable.



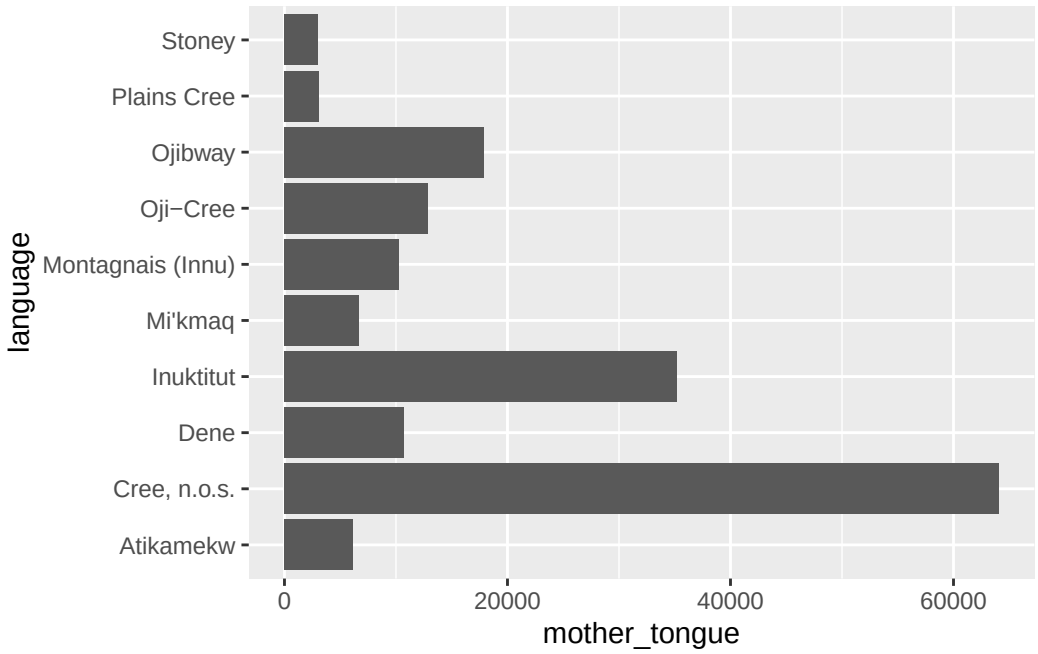
Is there any improvements we could make to this graph?

To better view text

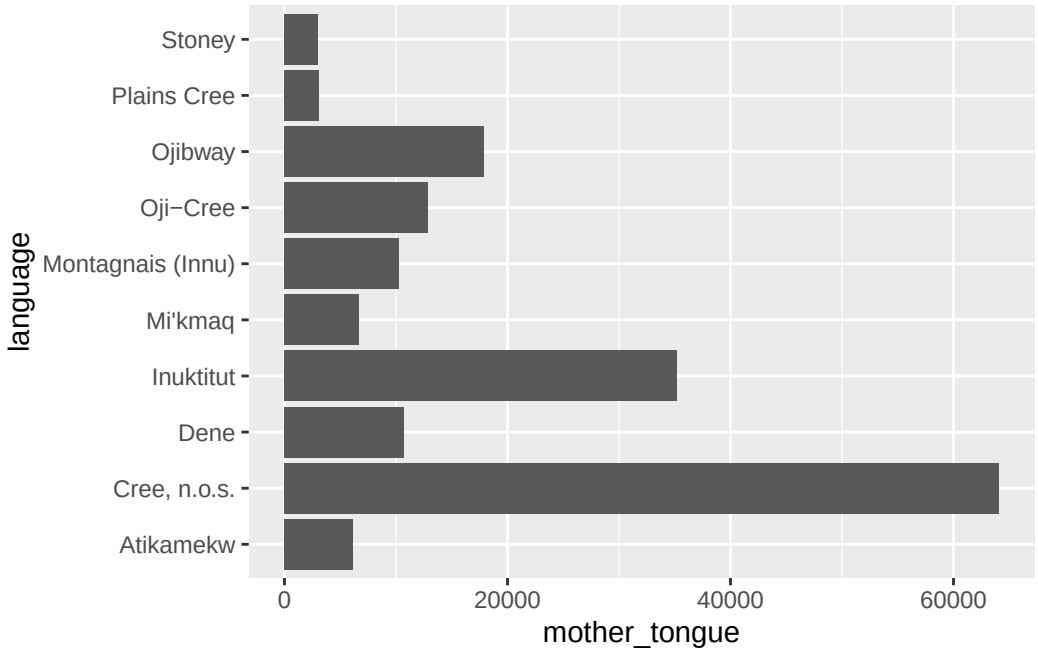
Display the bars horizontally instead of vertically!

```
ggplot(ten_lang, aes(x = language, y = mother_tongue)) +
  geom_bar(stat = "identity") +
  coord_flip() #<1>
```

① This flips the x and y axes!



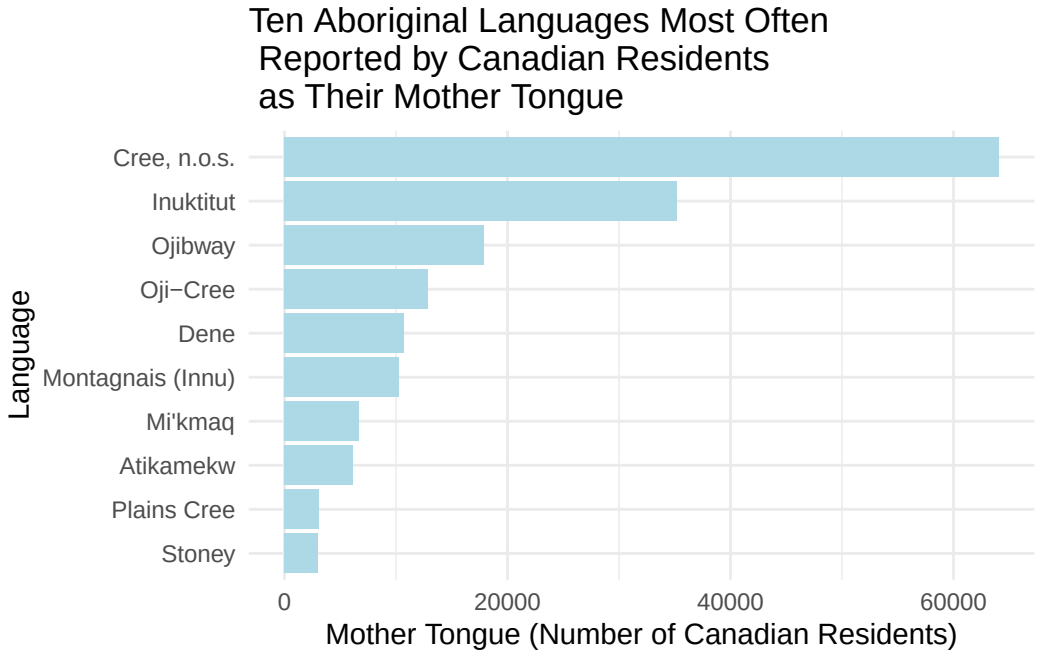

```
#OR

ggplot(ten_lang, aes(x = mother_tongue, y = language)) +
  geom_bar(stat = "identity")
```



Labels, Colors, and Themes

```
ggplot(ten_lang, aes(x = mother_tongue, y = reorder(language, mother_tongue))) + #<1>
  geom_bar(fill="lightblue", stat = "identity") + #<2>
  ylab("Language") + #<3>
  xlab("Mother Tongue (Number of Canadian Residents)") + #<4>
  ggtitle("Ten Aboriginal Languages Most Often \n Reported by Canadian Residents \n as The
  theme_minimal() #<6>
```

① The reorder function helps to reorder the languages from highest to lowest value of mother tongue.
② Changes the colors of the bars to light blue!
③ Updates x-axis label
④ Updates y-axis label
⑤ adds a title
⑥ changes the theme



> **Tip**
>
> Barplots are good for displaying one categorical variable and one numeric variable. The number variable could be counts (as above) or they could be averages or totals or maximums or minimums (or many other things!)

ggplot: scatterplot with `geom_point`

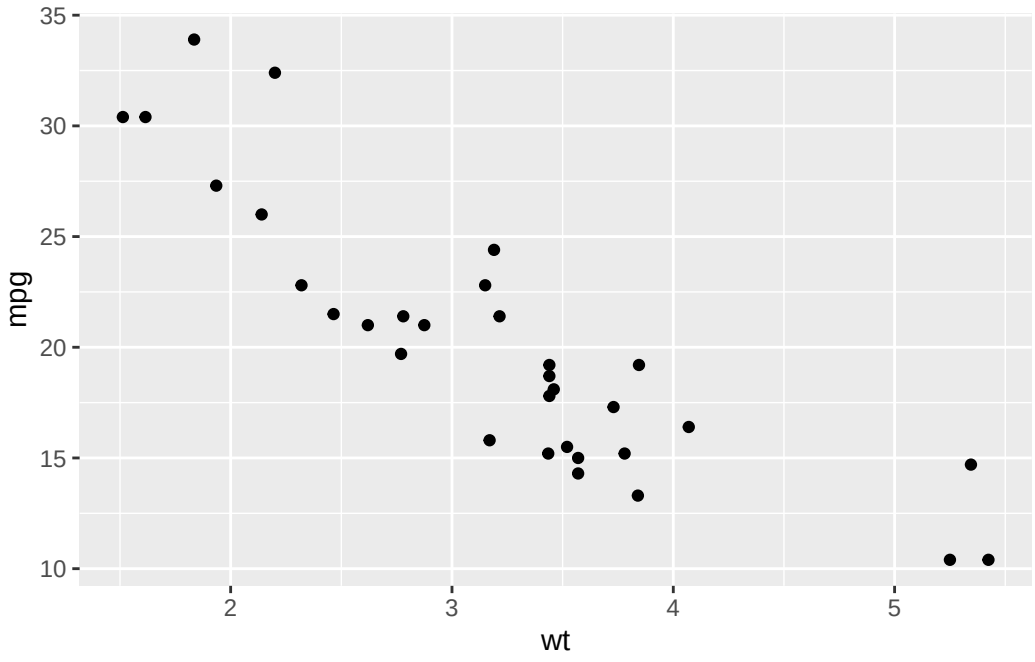
> **Tip**
>
> Scatterplots are good for displaying the relationship between two numerical variables.

The `mtcars` dataset was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). It's available inside the `ggplot` package which is already installed.

```
#load the data
data(mtcars)
```

```
mtcars %>% #<1>
ggplot(aes(x=wt, y=mpg)) + #<2>
  geom_point() #<3>
```

① Begin with the `mtcars` dataset
② set up the plot – weight on the x-axis and miles per gallon on the y-axis
③ Adds a scatter plot layer to the plot.



Note that you can change the color, shape (`pch` for plotting character) and size of these

points!

Quick update to the dataset

```
#code to update `mtcars` dataset so that `am` is treated as a factor rather than a continu
mtcars <- mtcars %>%
  mutate(am = as.factor(am))
```

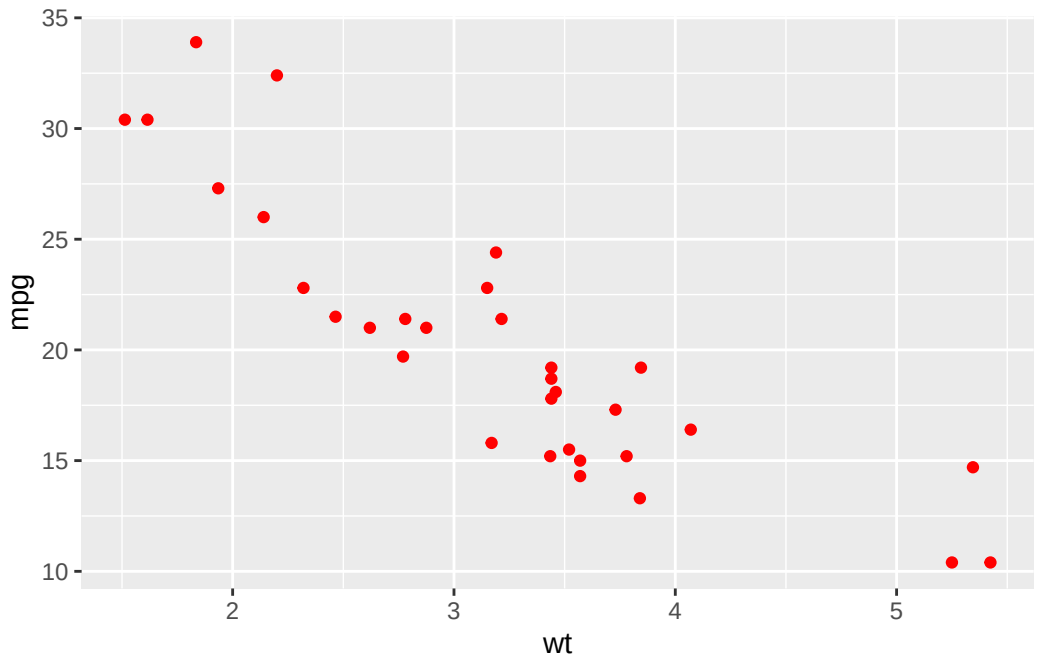
This modifies the am column, which represents the transmission type of the car (0 = automatic, 1 = manual). The as.factor(am) function converts the am variable from a numeric type (0 or 1) into a categorical factor.

Inside `aes()` or outside `aes()`?

What is the difference between these two graphs?

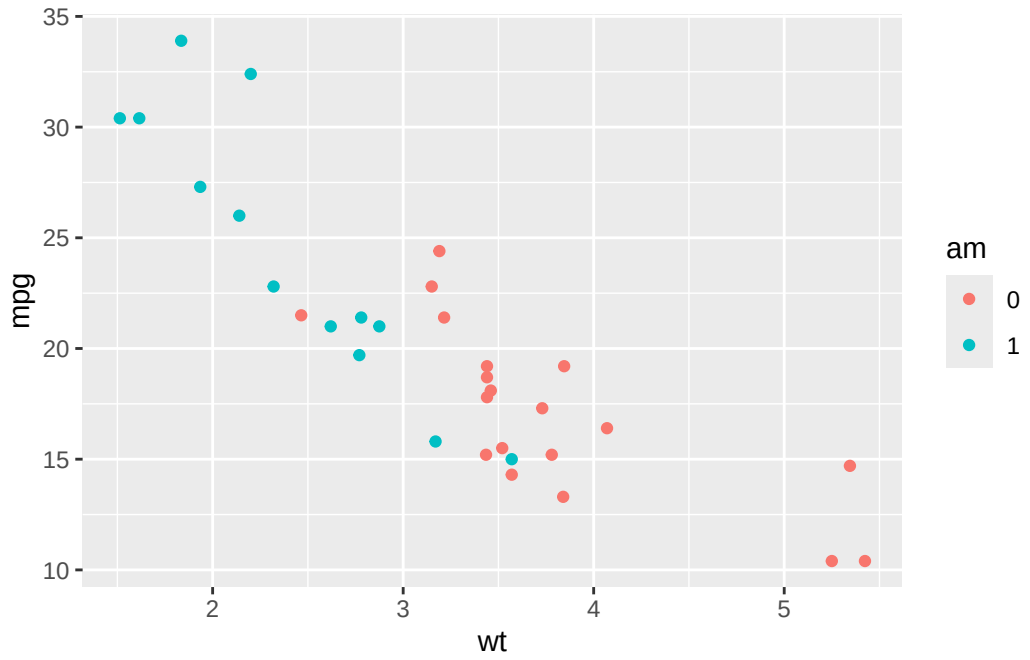
```
#color not in aesthetics
ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(color="red") #<1>
```

① color the same for all points



```
#color in aesthetics
ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=am)) #<1>
```

① color will vary based on the value of am

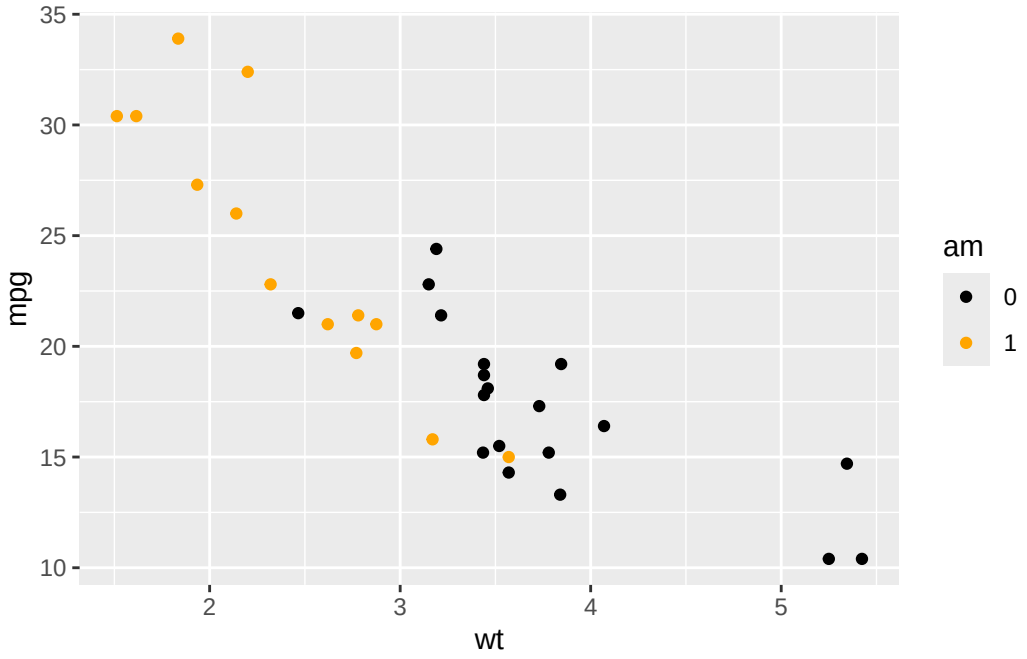


> Tip
>
> - If the thing you are trying to change (color, shape, size, etc.) depends on a variable, you should put in *inside* the aesthetics
> - If the thing you are trying to change (color, shape, size, etc.) should happen for all things, you should not put it inside the aesthetics.

Customizing Colors in Aesthetics

```
#color in aesthetics
ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=am)) +
  scale_color_manual(values=c("black", "orange")) #<1>
```

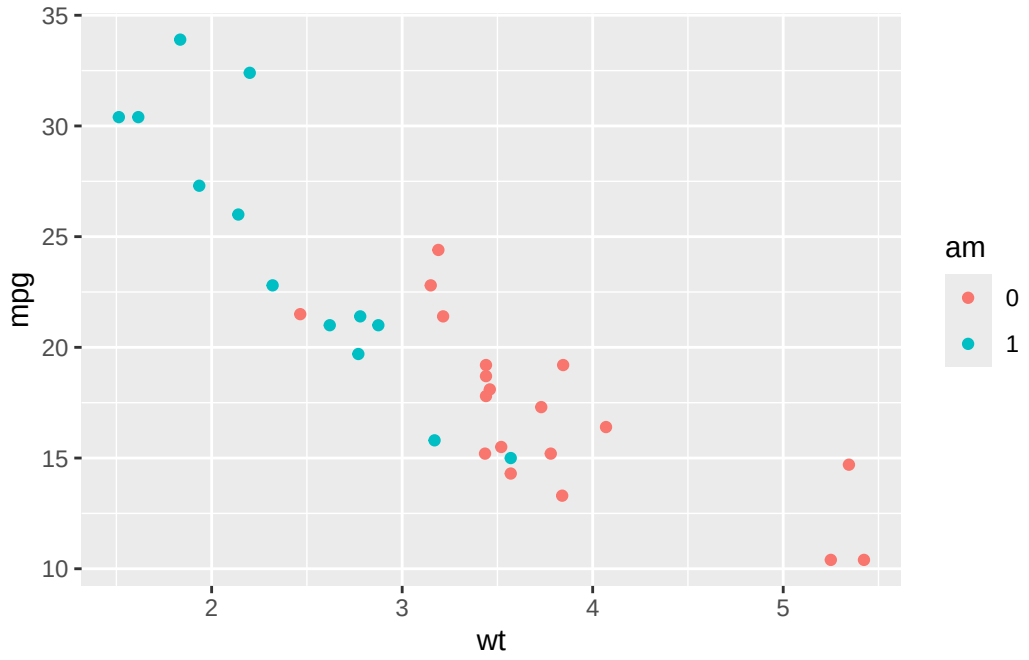
① This updates the two colors to black and orange, instead of the default colors.



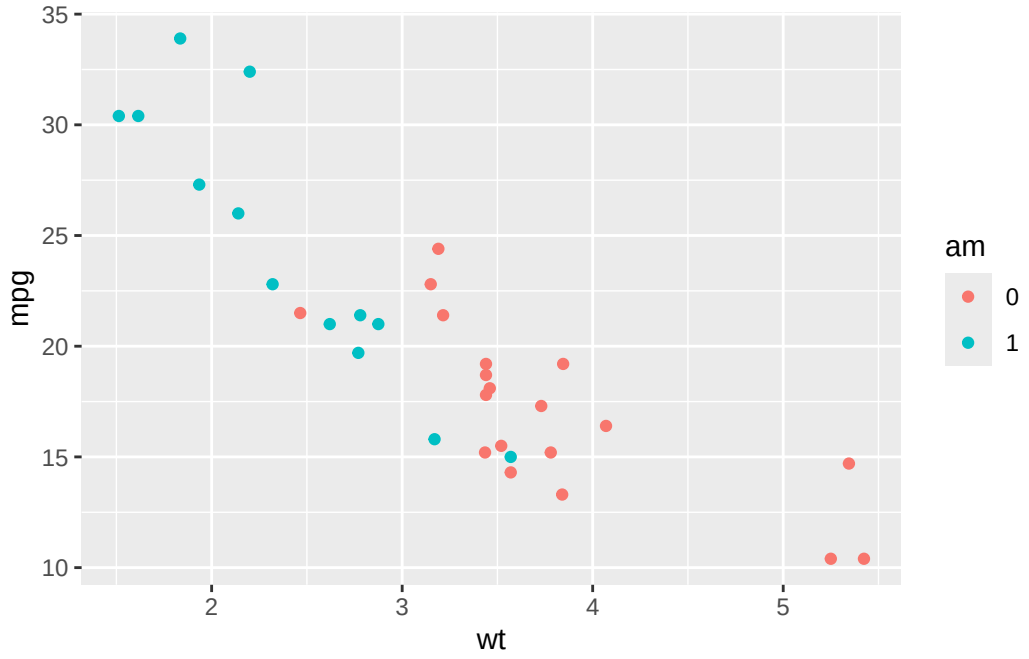
Global vs. Local Aesthetics

Global aesthetic mappings apply to all geometries and can be defined when you initially call `ggplot()`. All the geometries added as layers will default to this mapping. Local aesthetic mappings add additional information or override the default mappings.

```
#color = am as a global aethetic
ggplot(mtcars, aes(x=wt, y=mpg, color=am)) +
  geom_point()
```




```
#color = am as a local aethetic
ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(aes(color=am))
```



```r
#overwriting color = am as a global aethetic with a local aesthetic
ggplot(mtcars, aes(x=wt, y=mpg, color=am)) +
  geom_point(color="purple")
```