Tutorial 8: ggplot2

Anh Le (anh.le@duke.edu)

Oct 16, 2015

Agenda

ggplot2

Introduction

```
f_install_and_load <- function(packagename) {
  if (!packagename %in% rownames(installed.packages())) {
    install.packages(packagename)
  }
  library(packagename, character.only = TRUE)
}
packs <- c("ggplot2", "gridExtra")
lapply(packs, f_install_and_load)
```

```
## Loading required package: grid
```

```
## [[1]]
## [1] "ggplot2"   "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
##  [1] "gridExtra" "grid"      "ggplot2"   "stats"     "graphics"
##  [6] "grDevices" "utils"     "datasets"  "methods"   "base"
```

This tutorial aims to teach you not only about the `ggplot2` syntax but also a new way of thinking about graphics. What's revolutionary about `ggplot2` is really this "Grammar of Graphics" thinking (i.e. what `gg-` stands for).

Deconstructing a Scatterplot (learning about Aesthetics)

Below is a simple scatterplot. How exactly is data represented in a scatterplot? Each observation is represented by a point, whose x- and y- coordinate represent the values of the two variables. Along with the coordinates, each point can also have size, color, and shape.

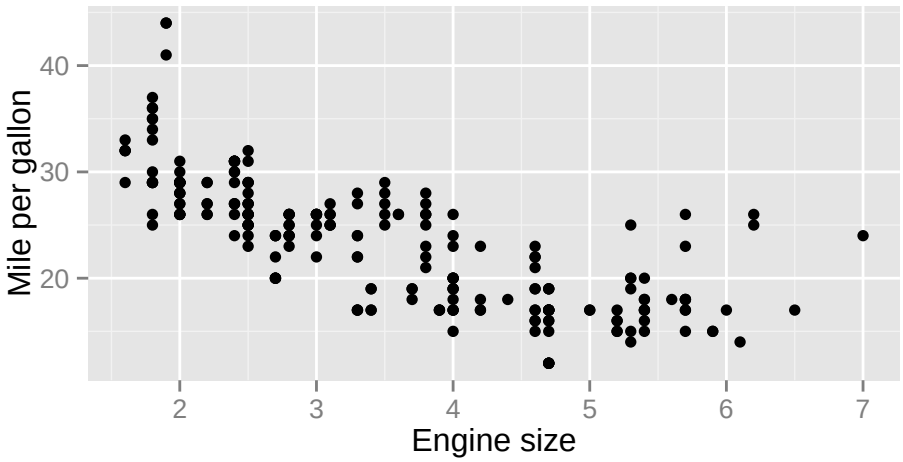
These attribute are called **aesthetics**, and are the properties that can be perceived on the graphic. Each aesthetic can be mapped to a variable, or set to a constant value.

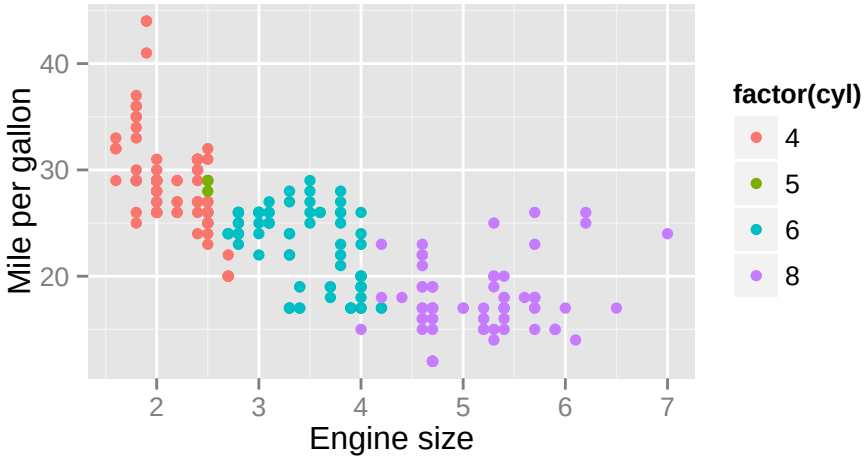
Q (conceptual): Without looking at the code, in this plot, what are the aesthetics, and what variables are they mapped to?

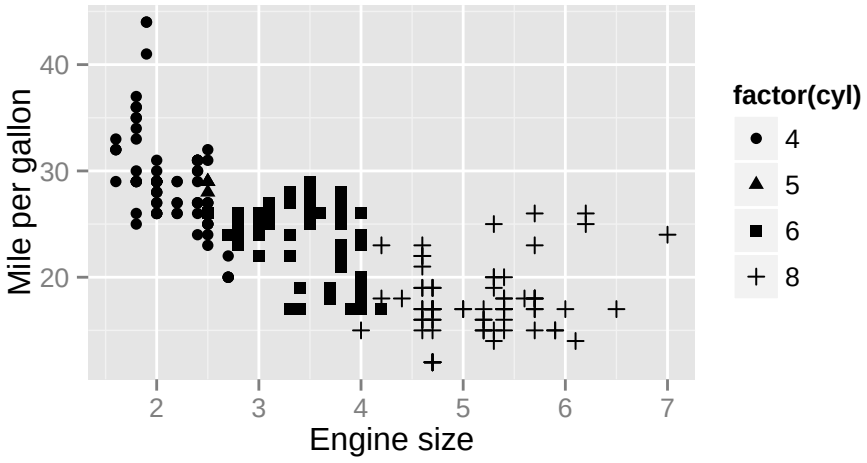
Q (ggplot syntax): Now look at the code. How did I specify the mapping?

Q (comparison with base R): How would you create the color plot in base R? Do you see how `ggplot2` is less ad-hoc and more expressive?

```
data("mpg")

p1 <-  ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  labs(y = "Mile per gallon", x = "Engine size")

p2 <- ggplot(data = mpg, aes(x = displ, y = hwy, color = factor(cyl))) +
  geom_point() +
  labs(y = "Mile per gallon", x = "Engine size")

p3 <- ggplot(data = mpg, aes(x = displ, y = hwy, shape = factor(cyl))) +
  geom_point() +
  labs(y = "Mile per gallon", x = "Engine size")

grid.arrange(p1, p2, p3, nrow = 3)
```



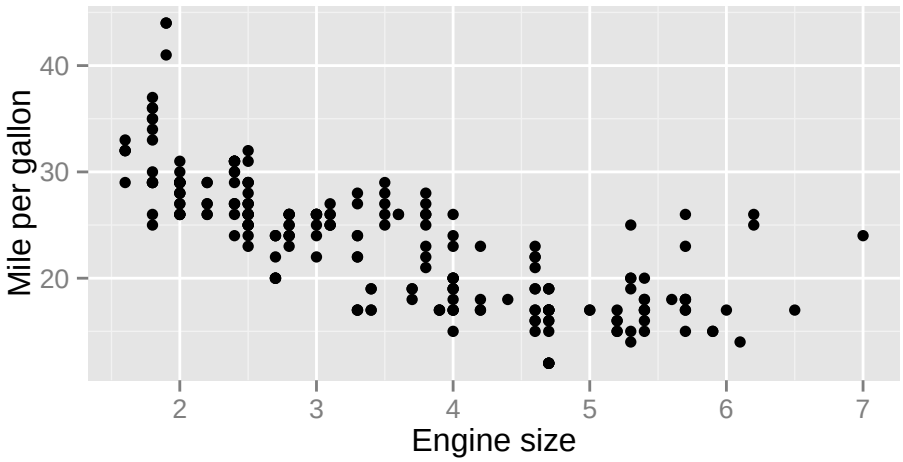


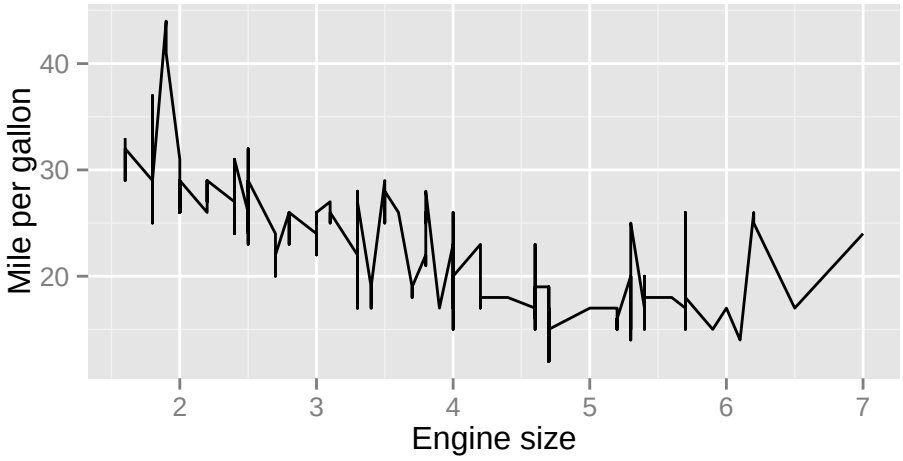


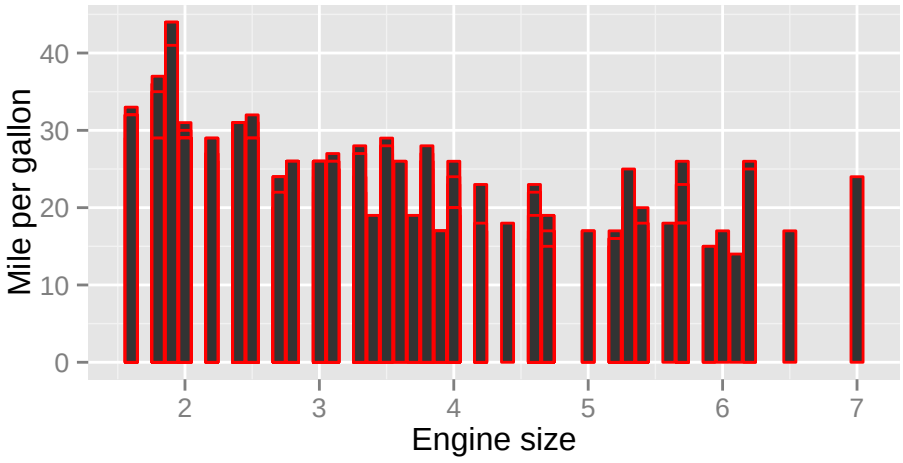
Note that the **aesthetics** is all we need to map the data to the graphics. In the plot above, I use **points** as the **geometric** form (hence `geom_point`), but it doesn't have to be. I can also use other **geometric** form, such as **line**, or **bar** (`geom_line`, `geom_bar`). Notice how the geomtric form changes, but the fundamental mapping of data to graphics does not change.

This is what we means by "grammar of graphics". The line plot and bar plot are grammatically correct, even though it makes little sense.

```r
p_point <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  labs(y = "Mile per gallon", x = "Engine size")

p_line <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_line() +
  labs(y = "Mile per gallon", x = "Engine size")

p_bar <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_bar(stat = 'identity', position = "identity", color = 'red') +
  labs(y = "Mile per gallon", x = "Engine size")

grid.arrange(p_point, p_line, p_bar, nrow=3)
```



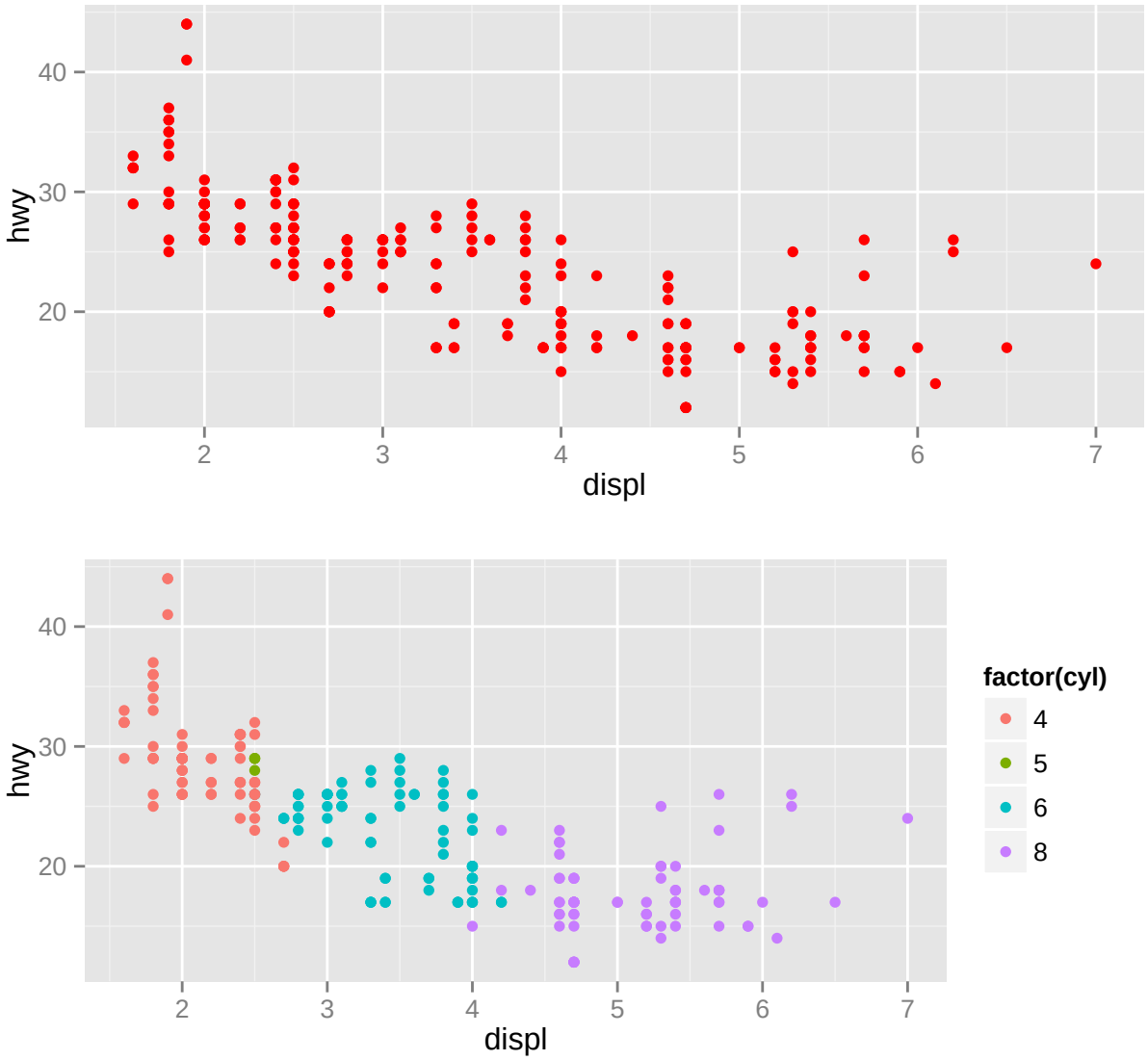




Setting aesthetics to a constant

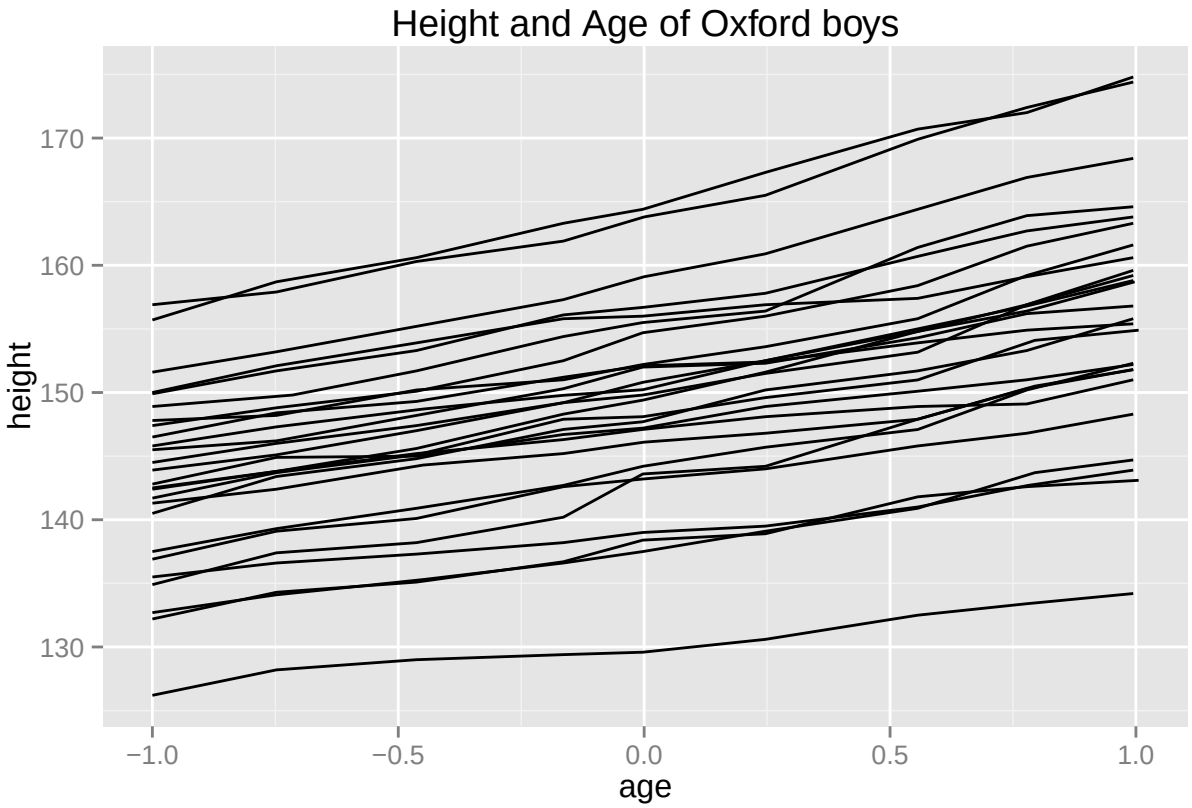
Sometimes I want to simply use a different color for my geoms, which is different from mapping the color to a variable. See:

```
p1 <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(color = 'red')

p2 <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(color = factor(cyl)))

grid.arrange(p1, p2, nrow = 2)
```



Grouping

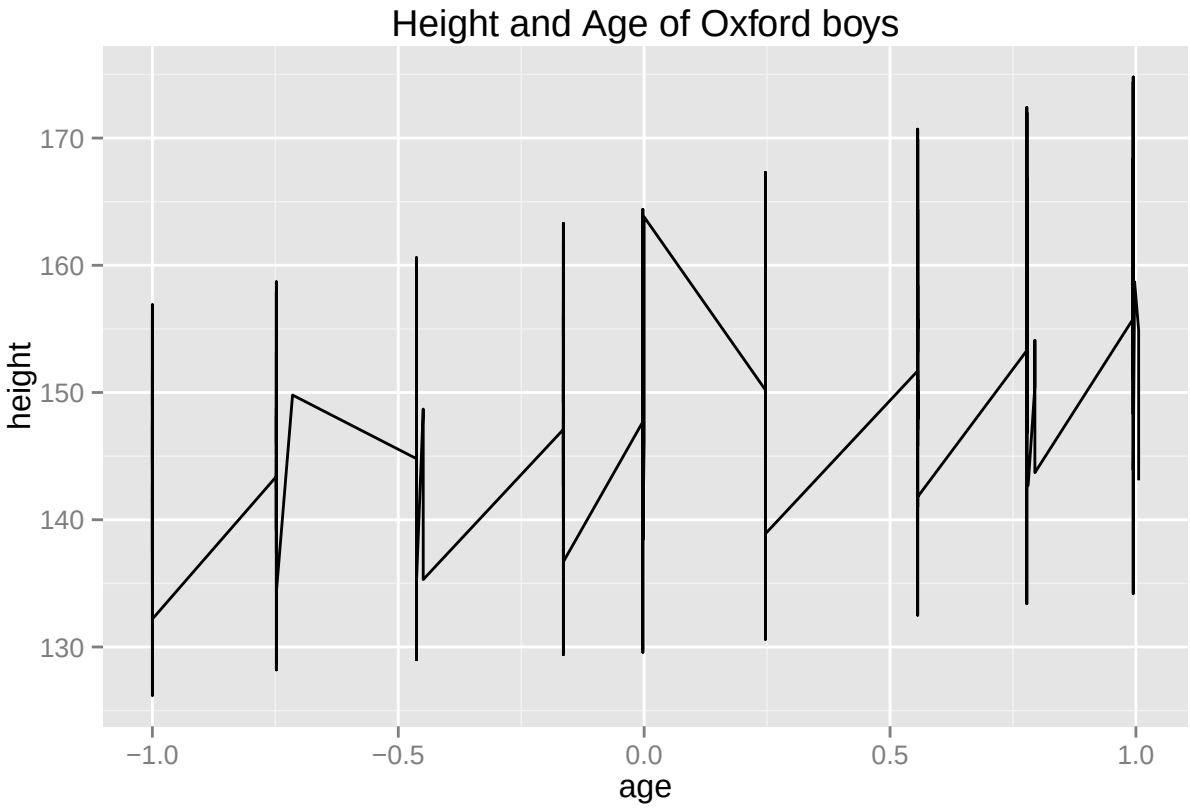
Q: Without looking at the code, what **aes** are specified in this plot?

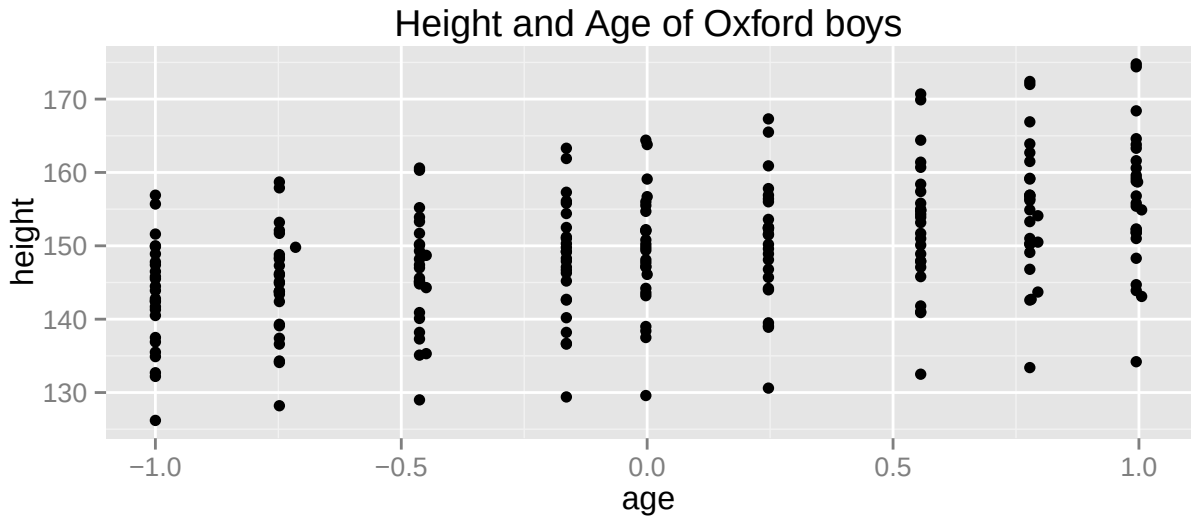


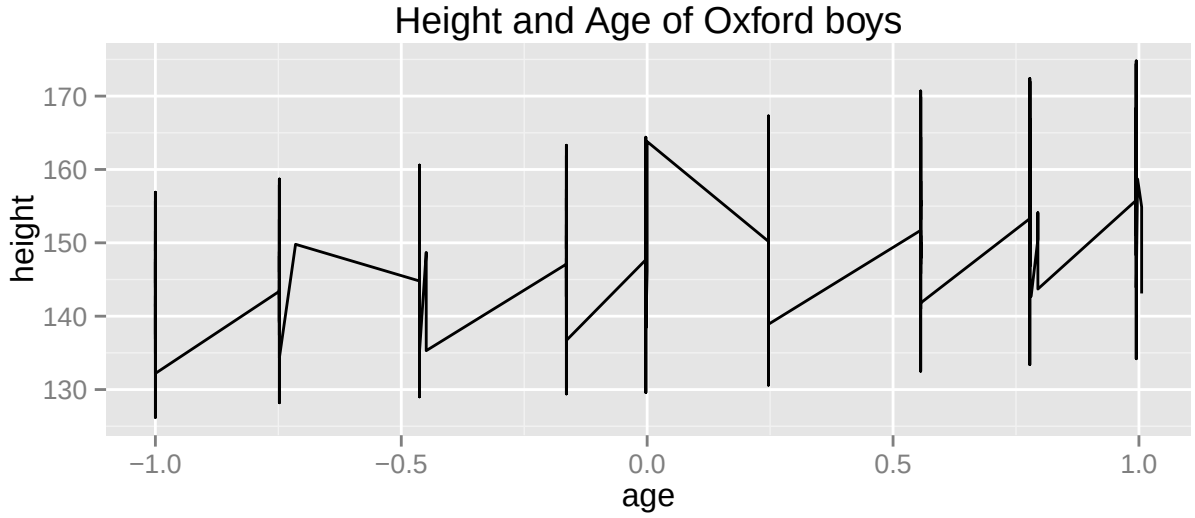
The code is (notice the **group** aesthetics). What would be consequence of not specifying this aesthetics?

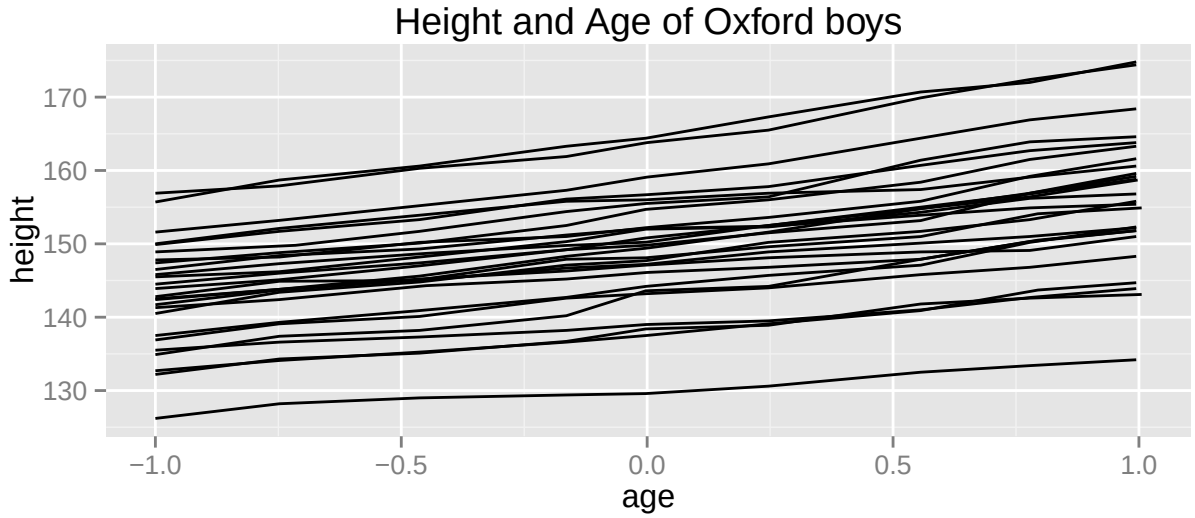
```
ggplot(Oxboys, aes(age, height, group = Subject)) +
  geom_line() +
  labs(title = "Height and Age of Oxford boys")
```

```
ggplot(Oxboys, aes(age, height)) +
  geom_line() +
  labs(title = "Height and Age of Oxford boys")
```


Height and Age of Oxford boys
height
170
160
150
140
130
−1.0
−0.5
0.0
0.5
1.0
age


Height and Age of Oxford boys
height
170
160
150
140
130
−1.0
−0.5
0.0
0.5
1.0
age


Height and Age of Oxford boys
height
170
160
150
140
130
−1.0
−0.5
0.0
0.5
1.0
age


Height and Age of Oxford boys
height
170
160
150
140
130
−1.0
−0.5
0.0
0.5
1.0
age

More about geoms

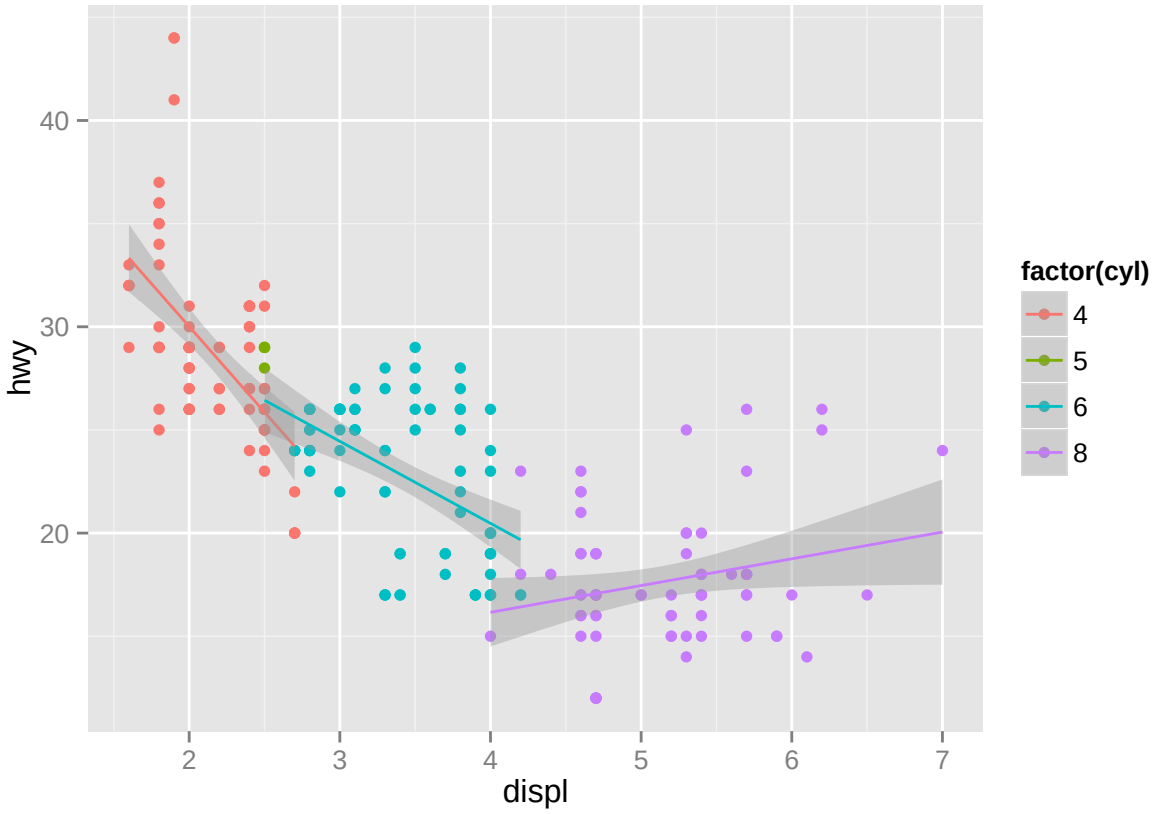
There are many **geom** built into ggplot2 see this doc. With this wide variety of selection, you can build all the common plot types.

Q (common plot): Think about how different **geom** can be useful:

- `geom_text` example
- `geom_vline` example
- `geom_errorbar` example
- `geom_polygon` example

But the magic is really in building any custom plot that you want, mixing and matching different **geoms**.

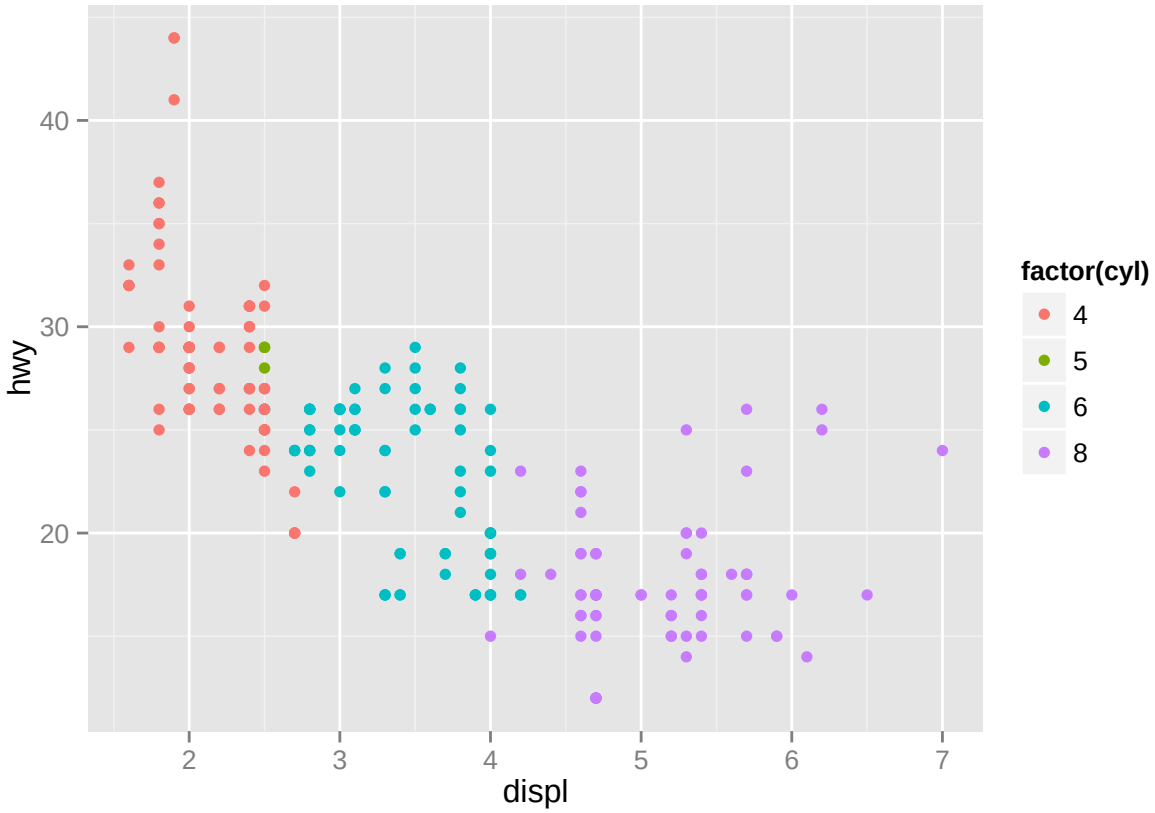
```
ggplot(data = mpg, aes(displ, hwy, color = factor(cyl))) +
  geom_point() +
  geom_smooth(data= subset(mpg, cyl != 5), method="lm")
```



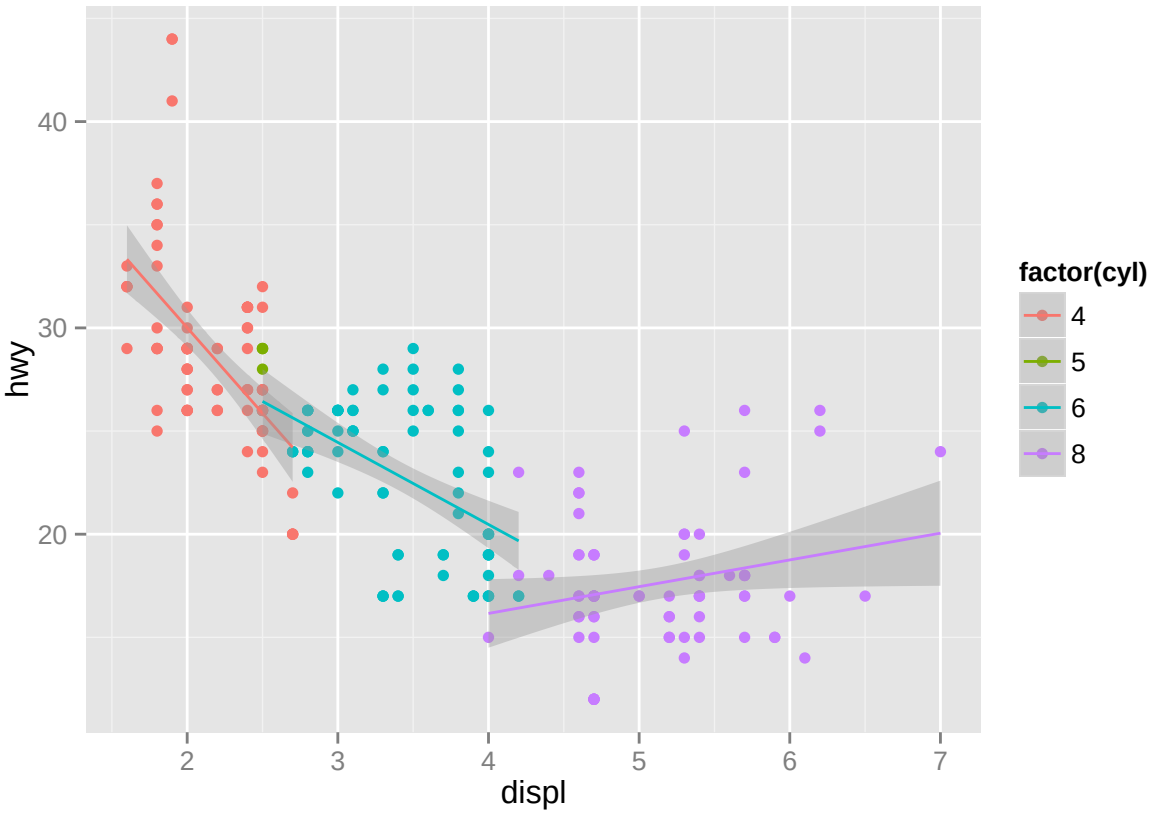
Building a plot layers by layers

As you may have noticed, the last plot has two geoms. In fact, we can add as many geoms as we want. This is thought of as building the plot layers by layers.

```
# First I add geom_point as the first layer
ggplot(data = mpg, aes(displ, hwy, color = factor(cyl))) +
  geom_point()
```

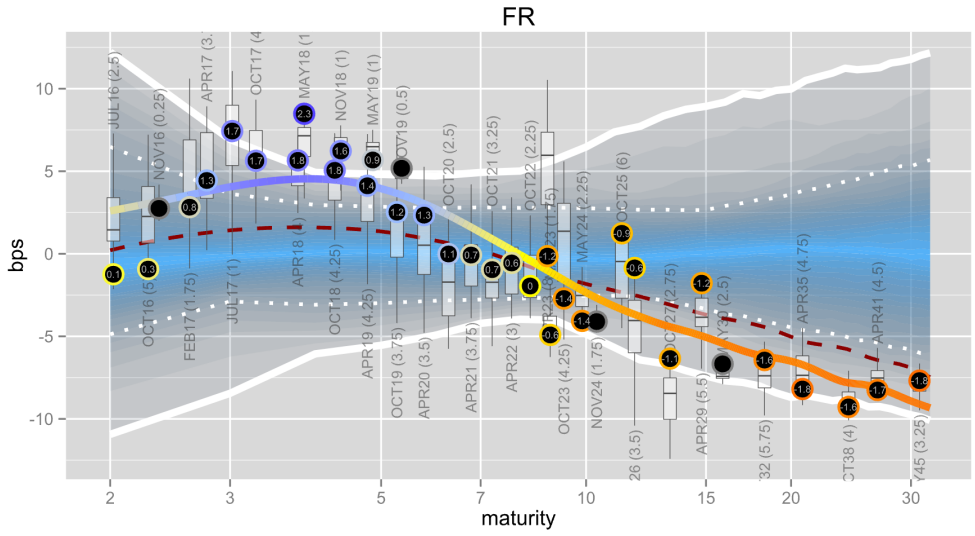



```
# Then I add geom_smooth as the second layer
ggplot(data = mpg, aes(displ, hwy, color = factor(cyl))) +
  geom_point() +
  geom_smooth(data= subset(mpg, cyl != 5), method="lm")
```



A rather extreme example of layering:

```
## Scale for 'colour' is already present. Adding another scale for 'colour', which will replace the existing scale.
```

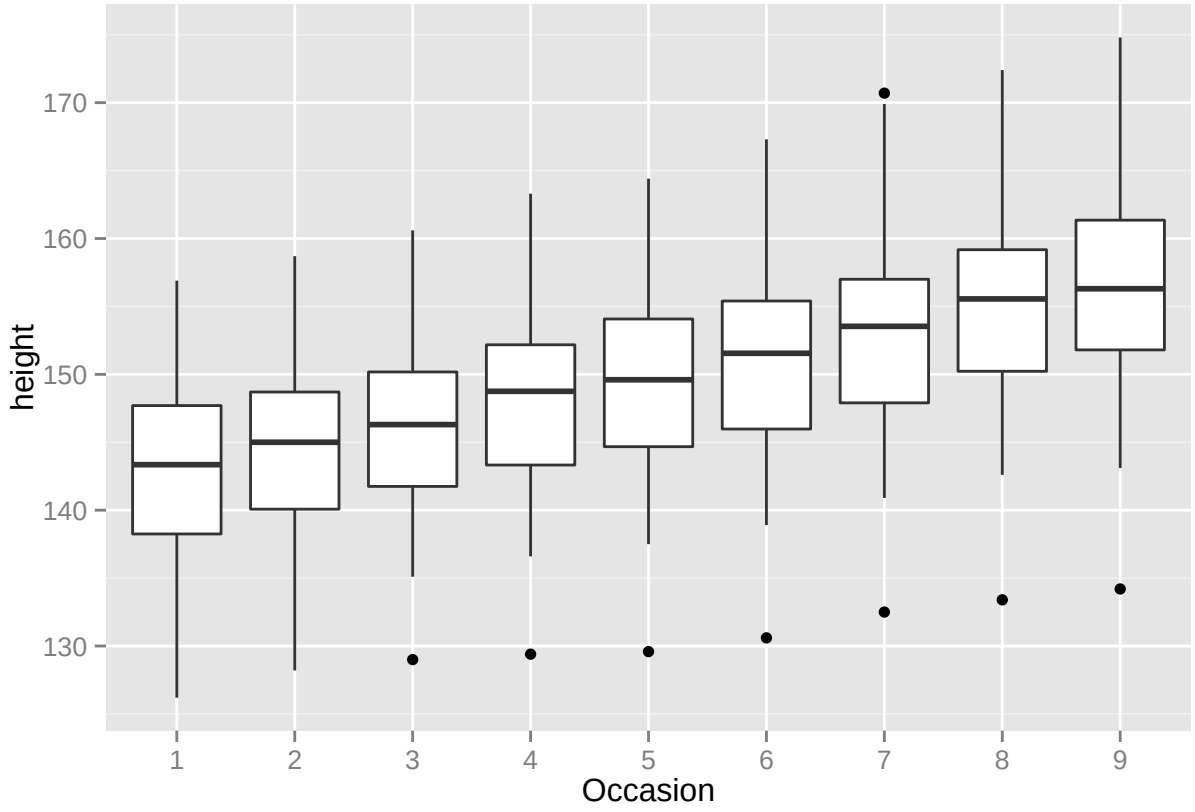


Overriding default aesthetics mapping

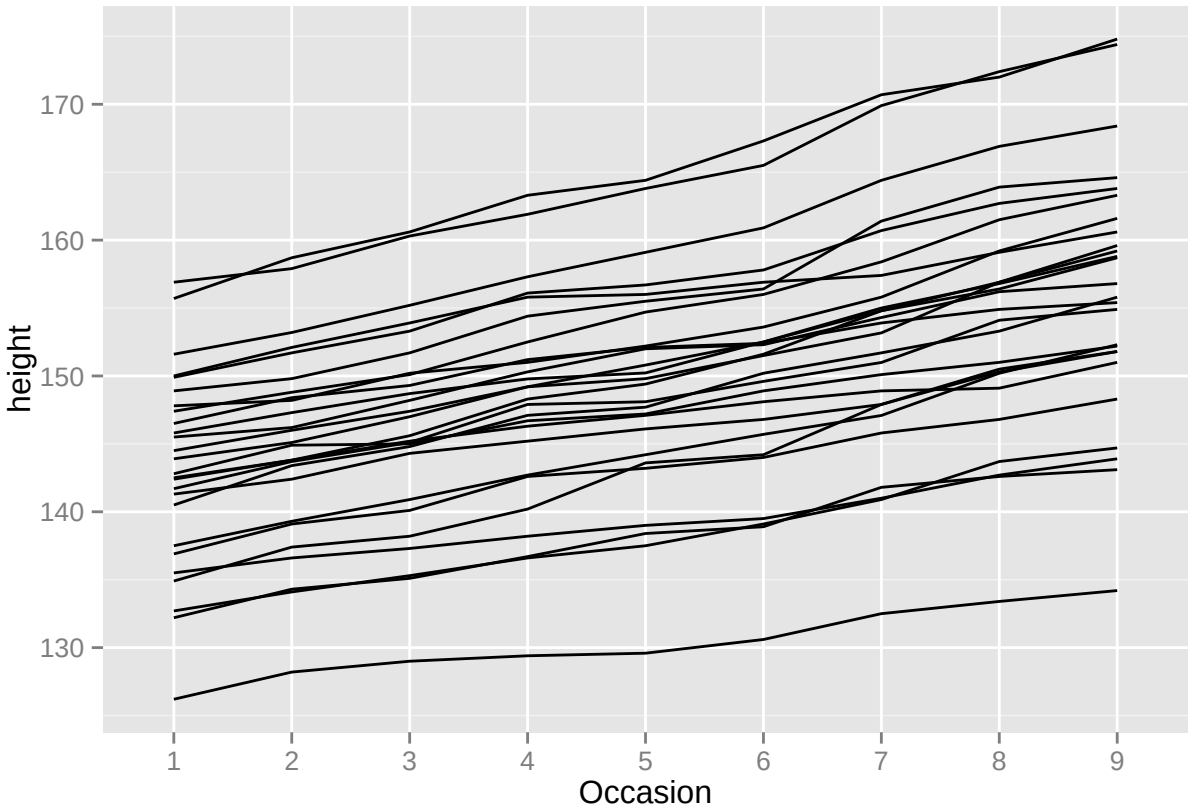
Now that we know how to build the plot layers by layers, sometimes the situation demands that we have different aesthetics mapping for each layer.

For example, consider the Oxford boy dataset. This is panel dataset, and I want to build 2 layers: a boxplot layer for each occasion (i.e. variation across boys at each time point), and a line layer for each boy (i.e. variation over time of each boy).

```
ggplot(data = Oxboys, aes(Occasion, height)) +
  geom_boxplot()
```

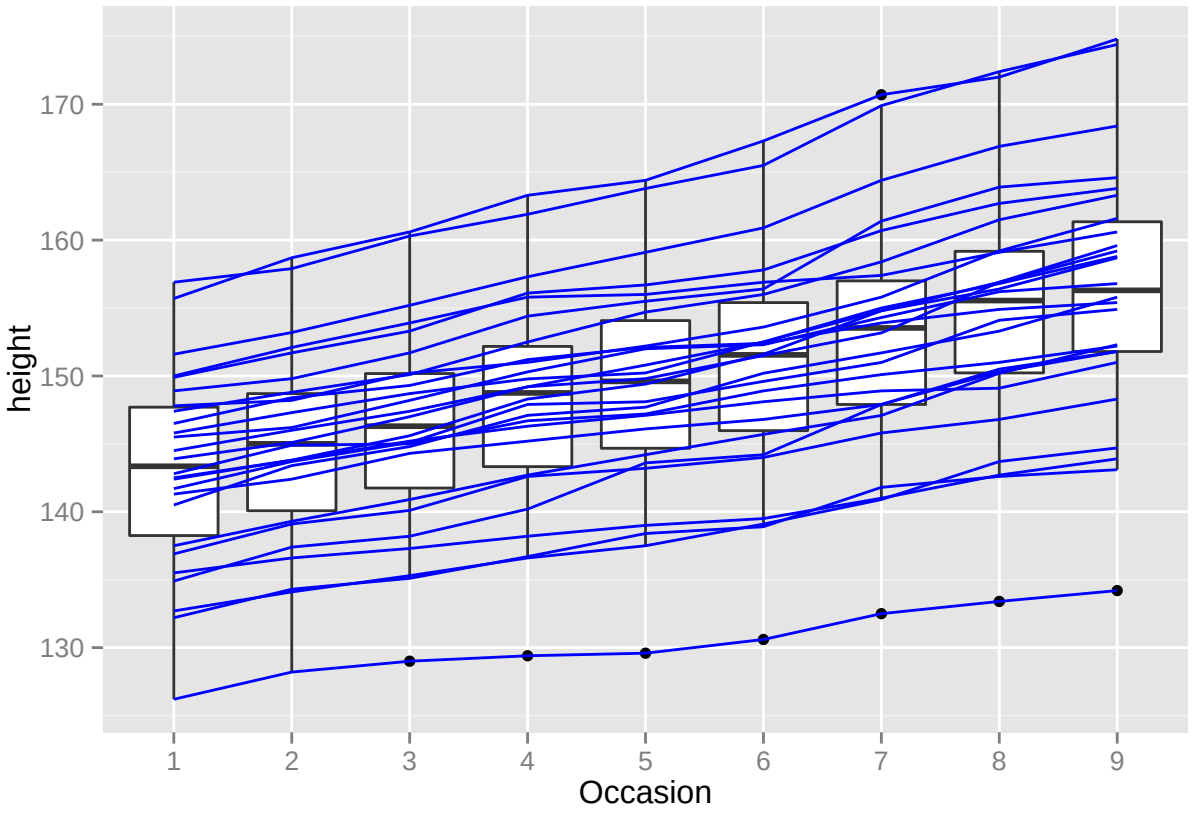



```
ggplot(data = Oxboys, aes(Occasion, height, group = Subject)) +
  geom_line()
```



I can specify an aesthetic mapping specific to each layer as follows

```
ggplot(data = Oxboys, mapping = aes(Occasion, height)) +
  geom_boxplot() +
  geom_line(aes(group = Subject), color = 'blue')
```


height
170
160
150
140
130
1
2
3
4
5
6
7
8
9
Occasion