Data Science for Economists

Lecture 12: Data Visualization

Alex Marsh
University of North Carolina | ECON 390

Table of contents

* Slides adapted from Grant McDermott's EC 607 at University of Oregon.

Introduction

Motivation

Visualizing data is one of the most powerful tools a data scientist can have.

R has a lot of very powerful data visualizing abilities which sets it apart from Python.

While base R plots are nice, ggplot2 is where visualizations in R really shine.

The goal of this lecture is to give y'all the basics of ggplot2 so that you can start making beautiful visualizations on your own.

Base R Plots

Base R Plots

We will not be spending much time with plots in base R, but it is worth covering the basics.

The two functions that will be the most useful are `plot()` and `hist()`.

While base R plots are rather easy to create, a lot of work is needed to make them look really nice.

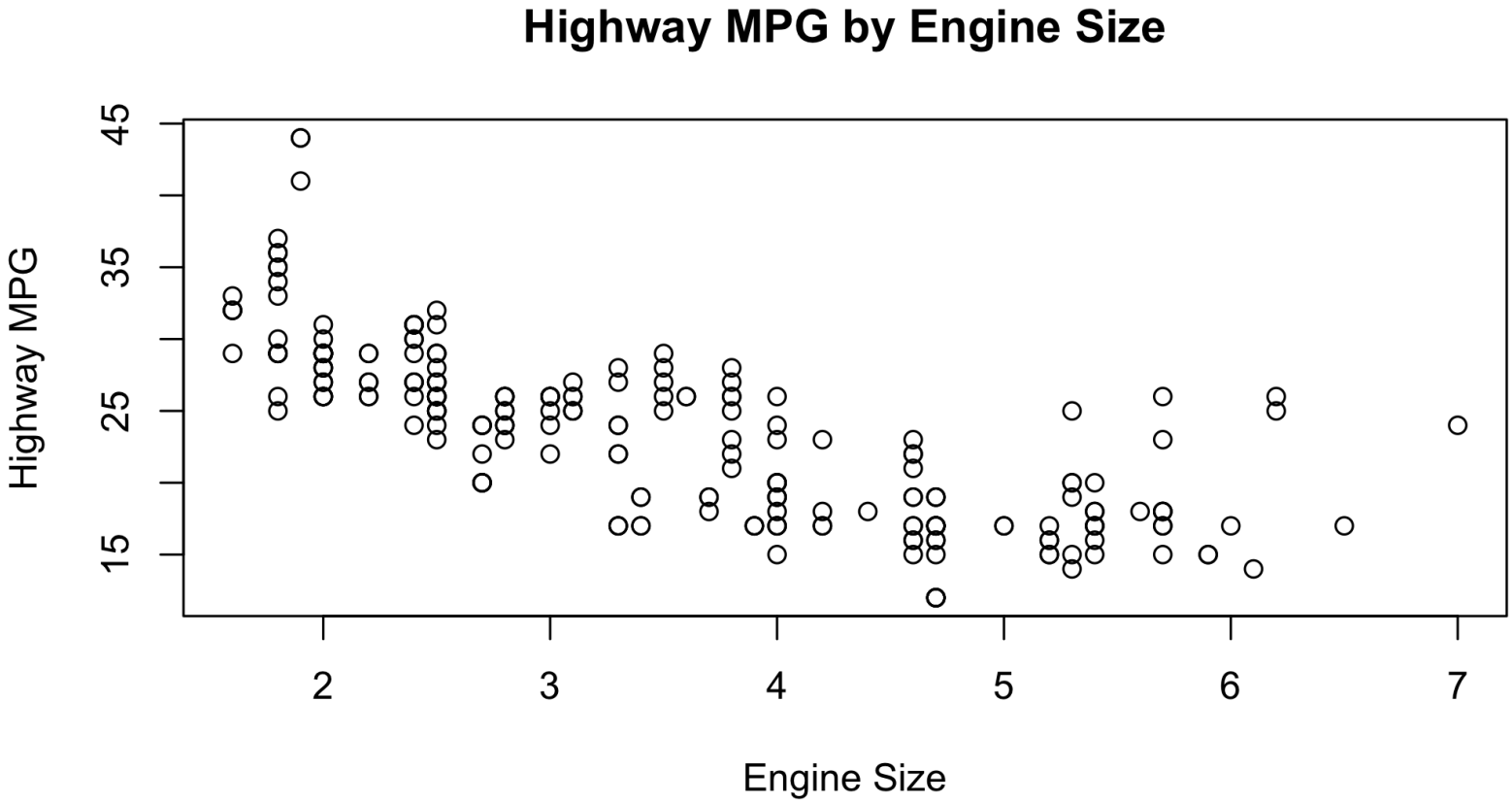
Basically everything has to be specified manually.

The idea is you make a base plot and then other things have to be added to it later.

```
par(mar = c(4, 4, 1, .1))
data(mpg)
xlab       = "Engine Size"
ylab       = "Highway MPG"
plot_title = "Highway MPG by Engine Size"
```

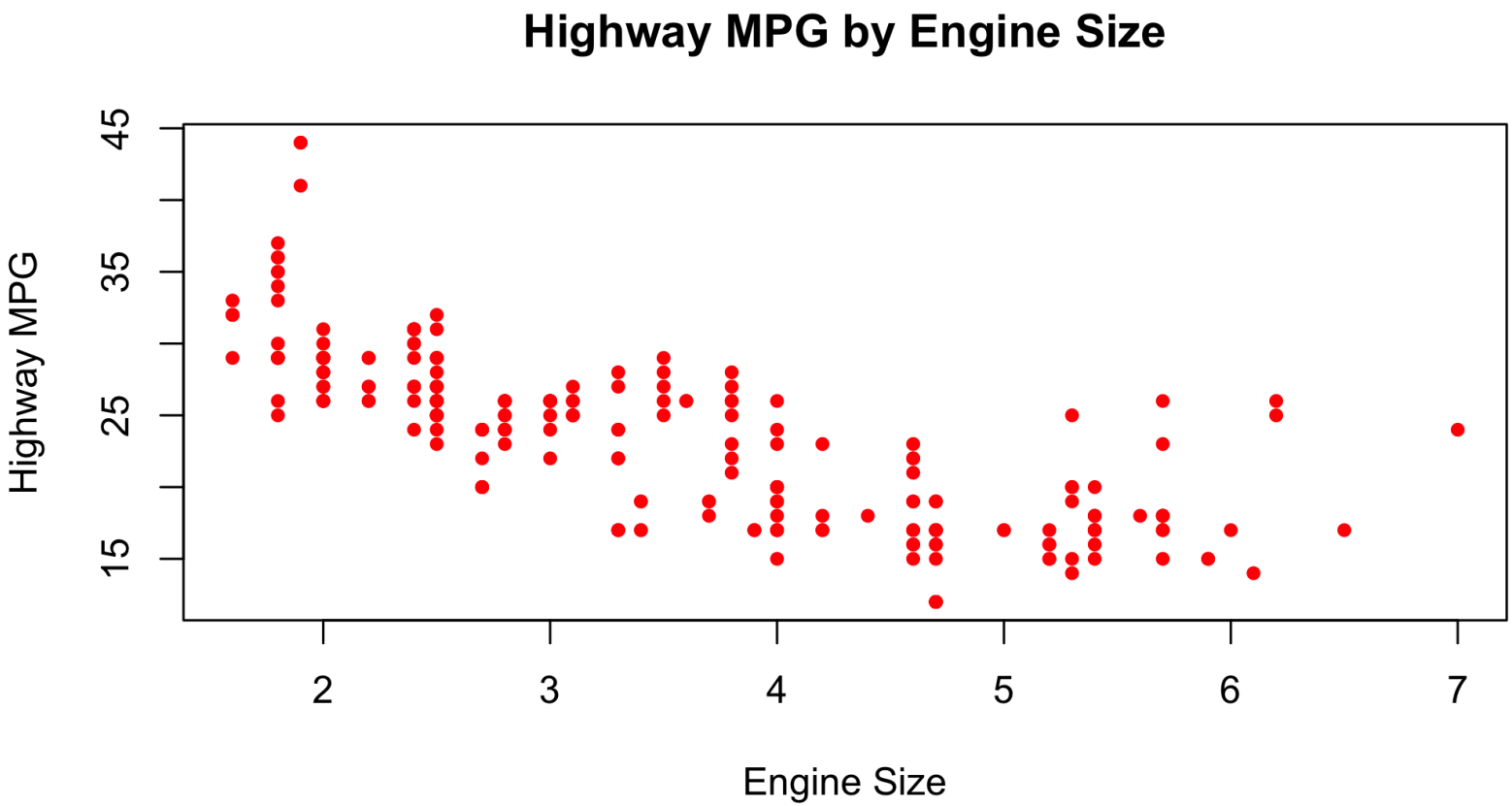
Base R Plots

```
plot(y = mpg$hwy,x=mpg$displ,main=plot_title,xlab=xlab,ylab=ylab)
```



Base R Plots

```
plot(y = mpg$hwy,x=mpg$displ,main=plot_title,xlab=xlab,ylab=ylab,
     col="red",pch = 20)
```



Base R Plot

Let's add a line of best fit.

You can do this by regressing hwy on displ and then getting the fitted (predicted) values and then adding a line to base plot.

```
plot_mod = lm(hwy ~ displ, data=mpg)
y_hat    = predict(plot_mod)
```

Base R Plots

```
plot(y = mpg$hwy,x=mpg$displ,main=plot_title,xlab=xlab,ylab=ylab,
     col="red",pch = 20)
lines(y=y_hat,x=mpg$displ)
```

Base R Plots

Histograms are relatively easy: just pass whatever data you want a histogram of into `hist()`

```
hist(mpg$hwy)
```

Base R Plots

We could spend a long time just with base R plots. However, my goal for this lecture is just to give you the basics to make them on your own.

See this cheatsheet for more.

There are much better packages for making beautiful plots in R without much effort. Which leads us to...

ggplot2

ggplot2

gglot2 is one of the best packages in R.

Technically, it is a part of the tidyverse, but I actually didn't know that until recently.

In fact, Python has it's own version of ggplot2 that is significantly more limited than what the R version can do.

While ggplot2 can be a little difficult to understand at first, once you get the hang of it, it's very straight forward and very powerful.

Elements of ggplot2

Hadley Wickham's ggplot2 is one of the most popular packages in the entire R canon.

- It also happens to be built upon some deep visualization theory: i.e. Leland Wilkinson's *The Grammar of Graphics*.

There's a lot to say about ggplot2's implementation of this "grammar of graphics" approach, but the three key elements are:

1. Your plot ("the visualization") is linked to your variables ("the data") through various **aesthetic mappings**.
2. Once the aesthetic mappings are defined, you can represent your data in different ways by choosing different **geoms** (i.e. "geometric objects" like points, lines or bars).
3. You build your plot in **layers**.

That's kind of abstract. Let's review each element in turn with some actual plots.

1. Aesthetic mappings

```
library(gapminder)
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
     geom_point()
```

1. Aesthetic mappings (cont.)

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

Focus on the top line, which contains the initialising `ggplot()` function call. This function accepts various arguments, including:

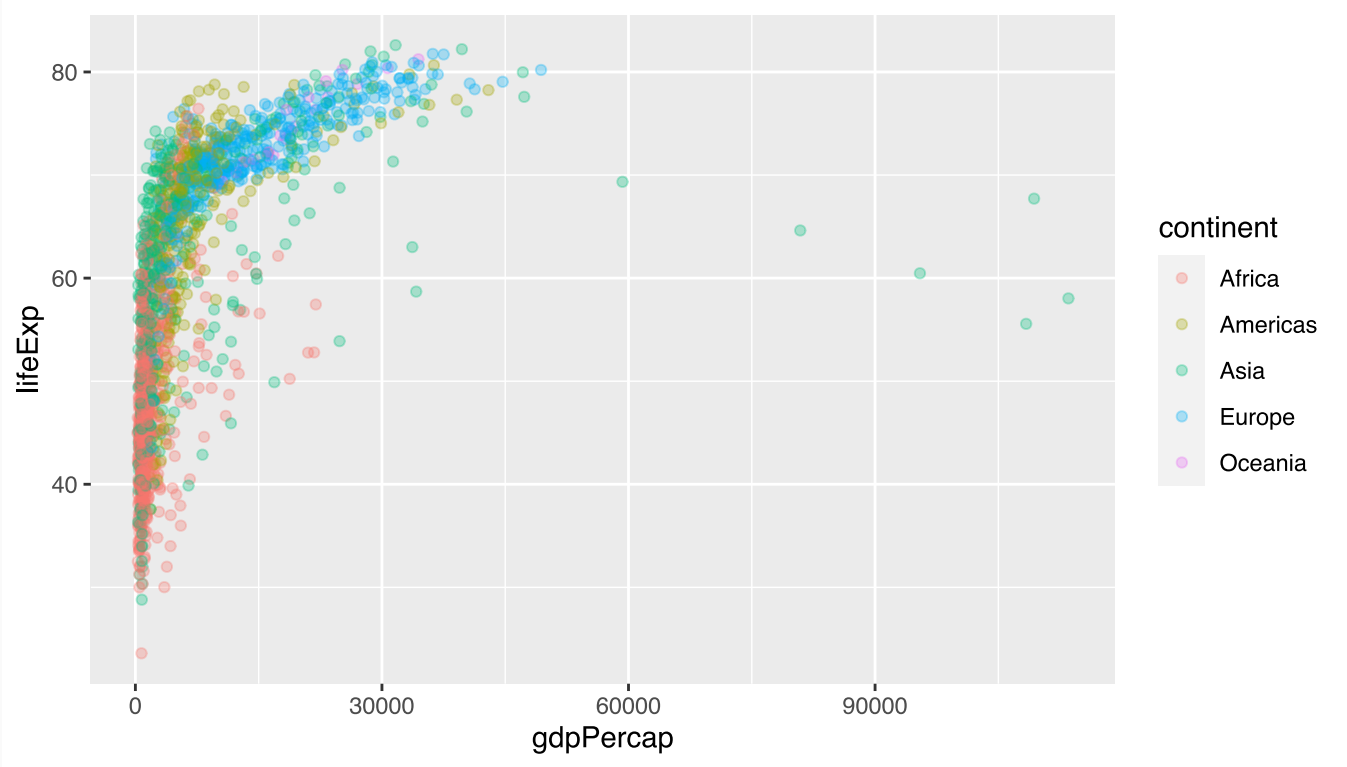
- Where the data come from (i.e. `data = gapminder`).
- What the aesthetic mappings are (i.e. `mapping = aes(x = gdpPercap, y = lifeExp)`).

The aesthetic mappings here are pretty simple: They just define an x-axis (GDP per capita) and a y-axis (life expecancy).

- To get a sense of the power and flexibility that comes with this approach, however, consider what happens if we add more aesthetics to the plot call...

1. Aesthetic mappings (cont.)

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, col = continent)) +
  geom_point(alpha = 0.3) ## "alpha" controls transparency. Takes a value between (
```



Note that I've dropped the "mapping =" part of the ggplot call. Most people just start with "aes(...)", since `ggplot2` knows the order of the arguments.

1. Aesthetic mappings (cont.)

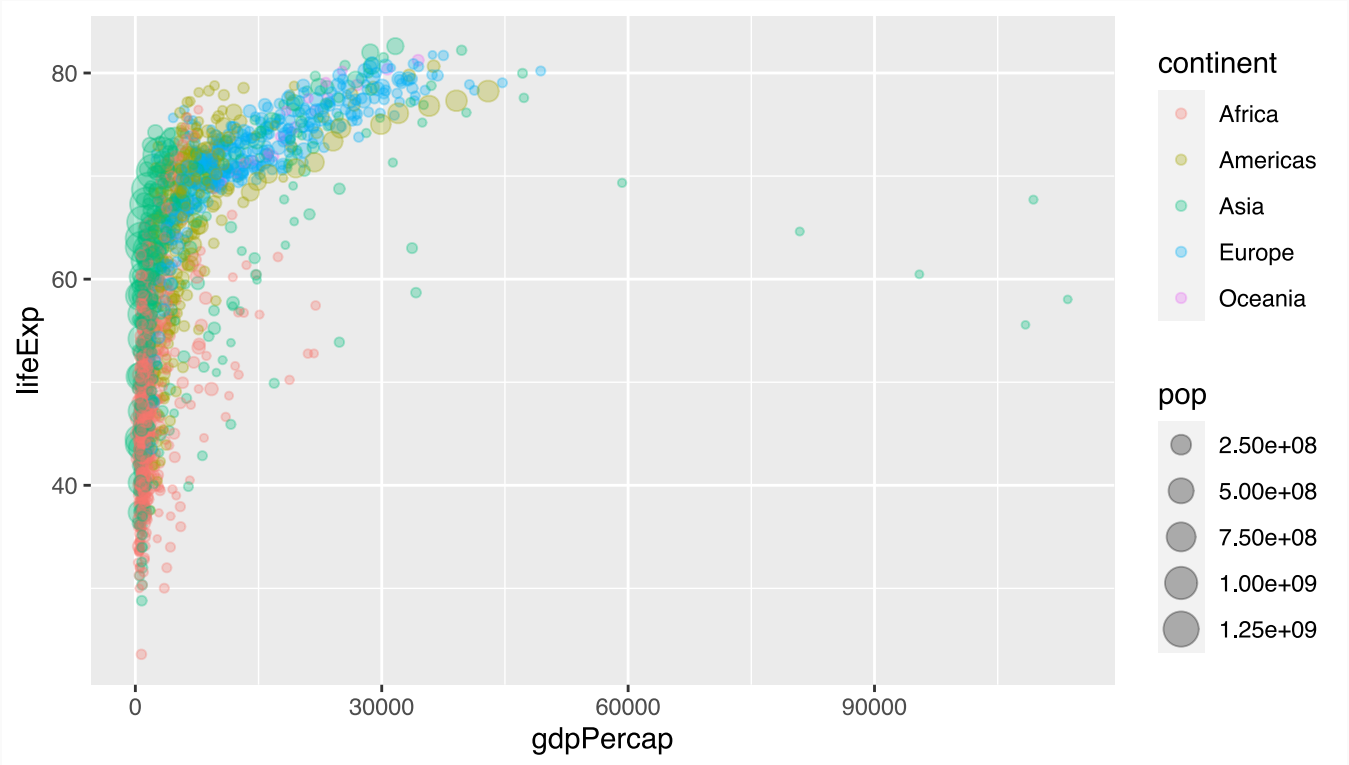
We colored the dots based on continents. What if we wanted to make the points based on population size?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size=pop, col = continent)) +
    geom_point(alpha = 0.3) ## "alpha" controls transparency. Takes a value between 0
```

1. Aesthetic mappings (cont.)

We can specify aesthetic mappings in the geom layer too.

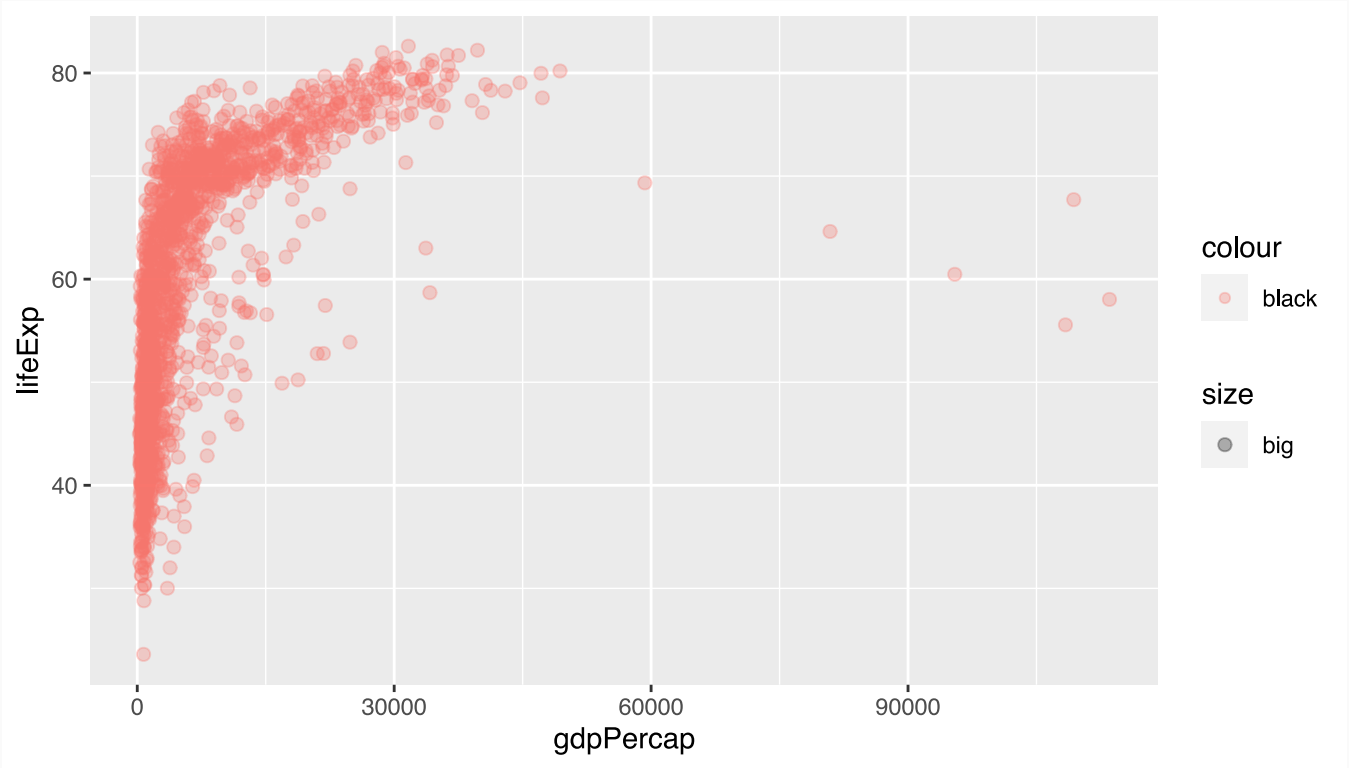
```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) + ## Applicable to all geoms
  geom_point(aes(size = pop, col = continent), alpha = 0.3) ## Applicable to this geom
```



1. Aesthetic mappings (cont.)

Oops. What went wrong here?

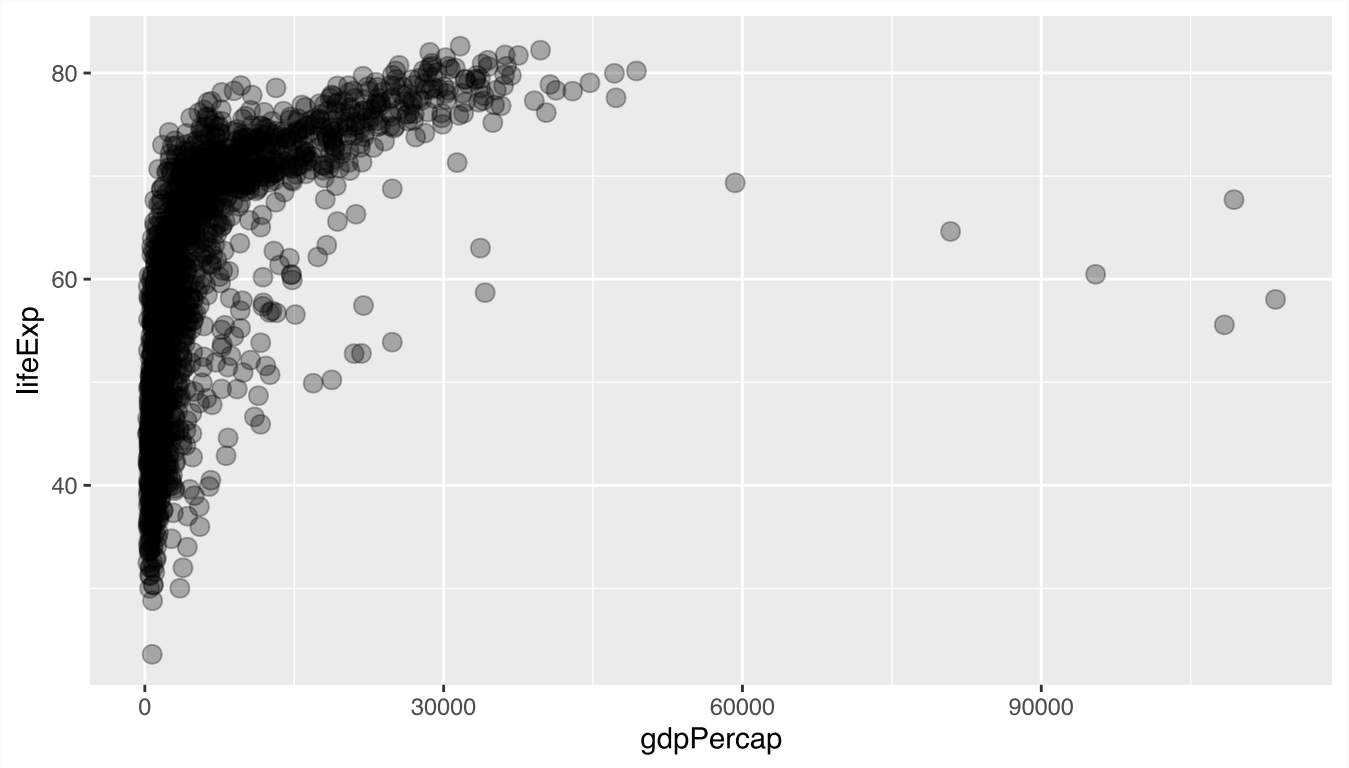
```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(size = "big", col="black"), alpha = 0.3)
```



1. Aesthetic mappings (cont.)

This is what we wanted to do!

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(size = 3, col="black", alpha = 0.3)
```

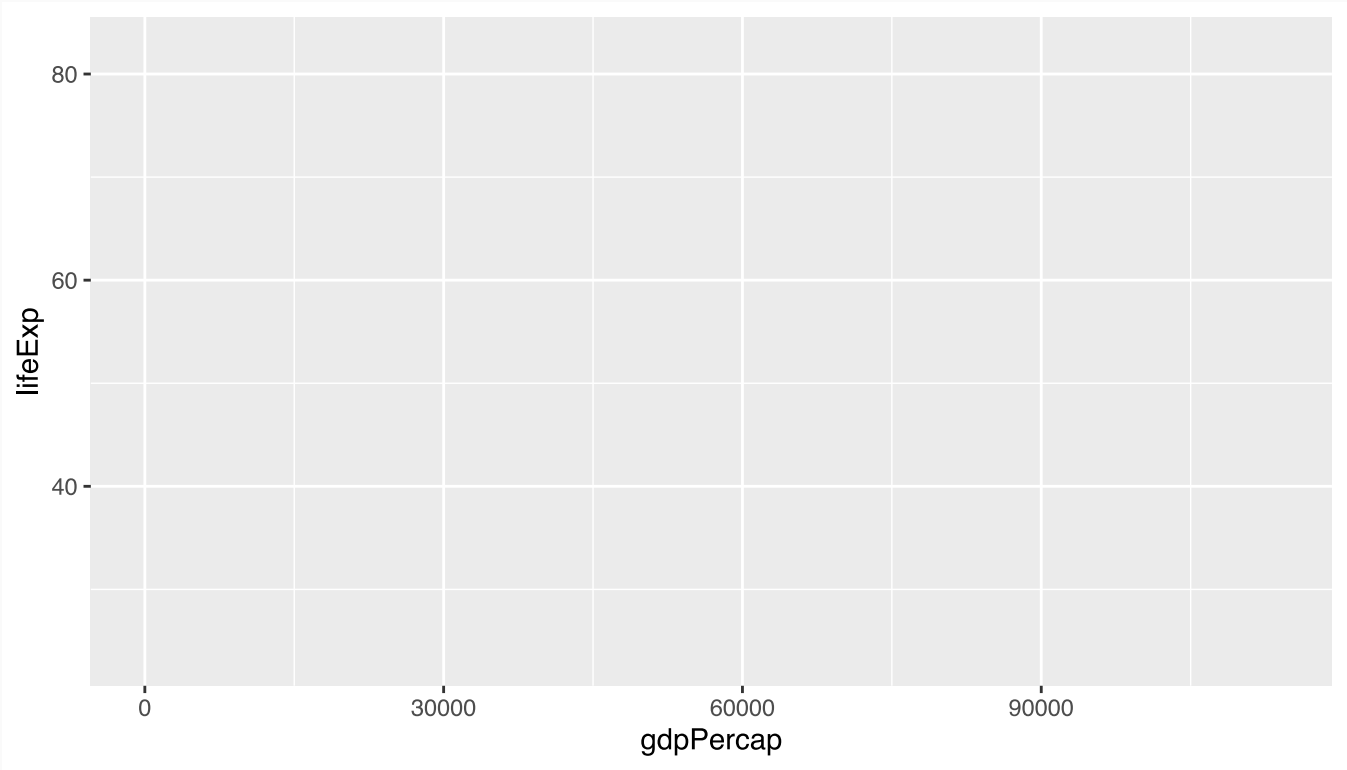


Note: `size` must take a numeric argument, not a character.

1. Aesthetic mappings (cont.)

At this point, instead of repeating the same ggplot2 call every time, it will prove convenient to define an intermediate plot object that we can re-use.

```
p = ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))
p
```

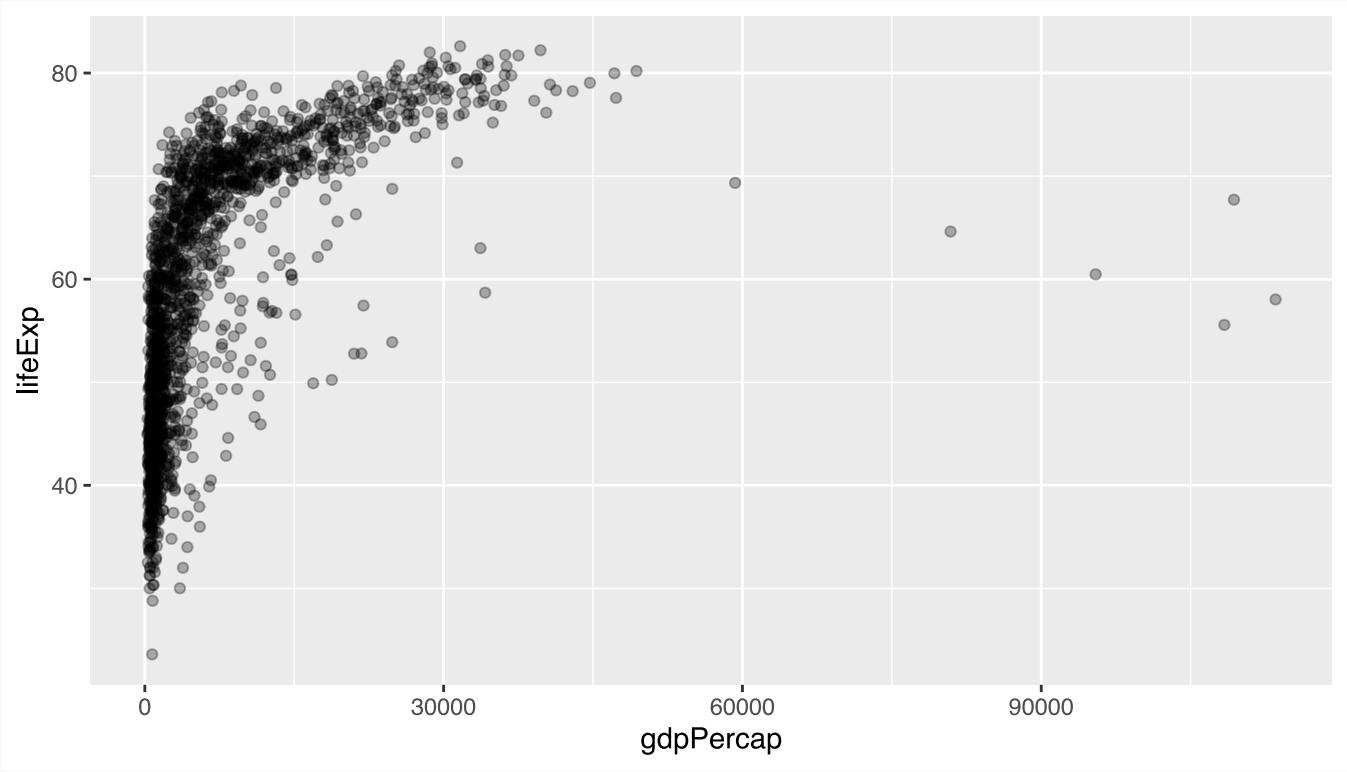


2. Geoms

Why did the base plot have no points or lines or anything?

Once your variable relationships have been defined by the aesthetic mappings, you can invoke and combine different geoms to generate different visulaizations.

```
p + geom_point(alpha = 0.3)
```

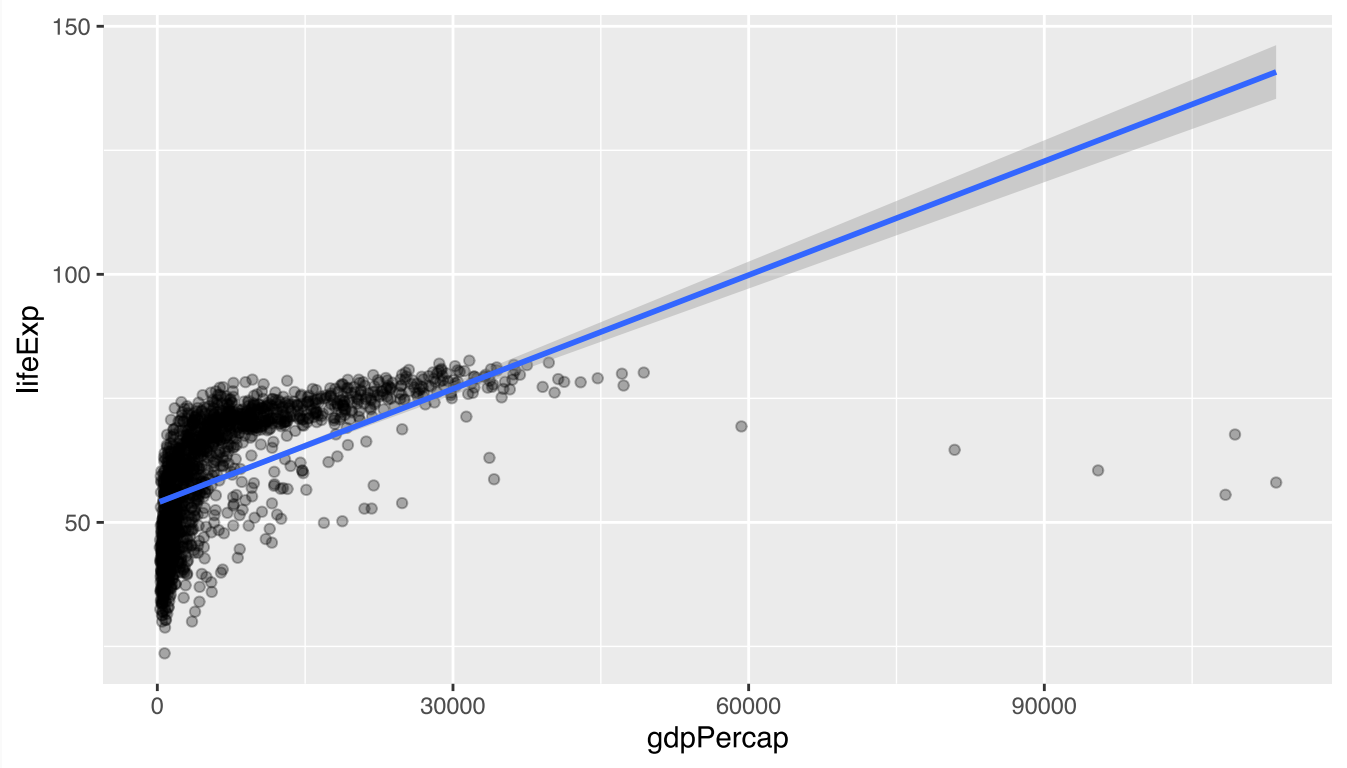


2. Geoms (cont.)

Can add line of best fit using `geom_smooth()`

```
p + geom_point(alpha = 0.3)+geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

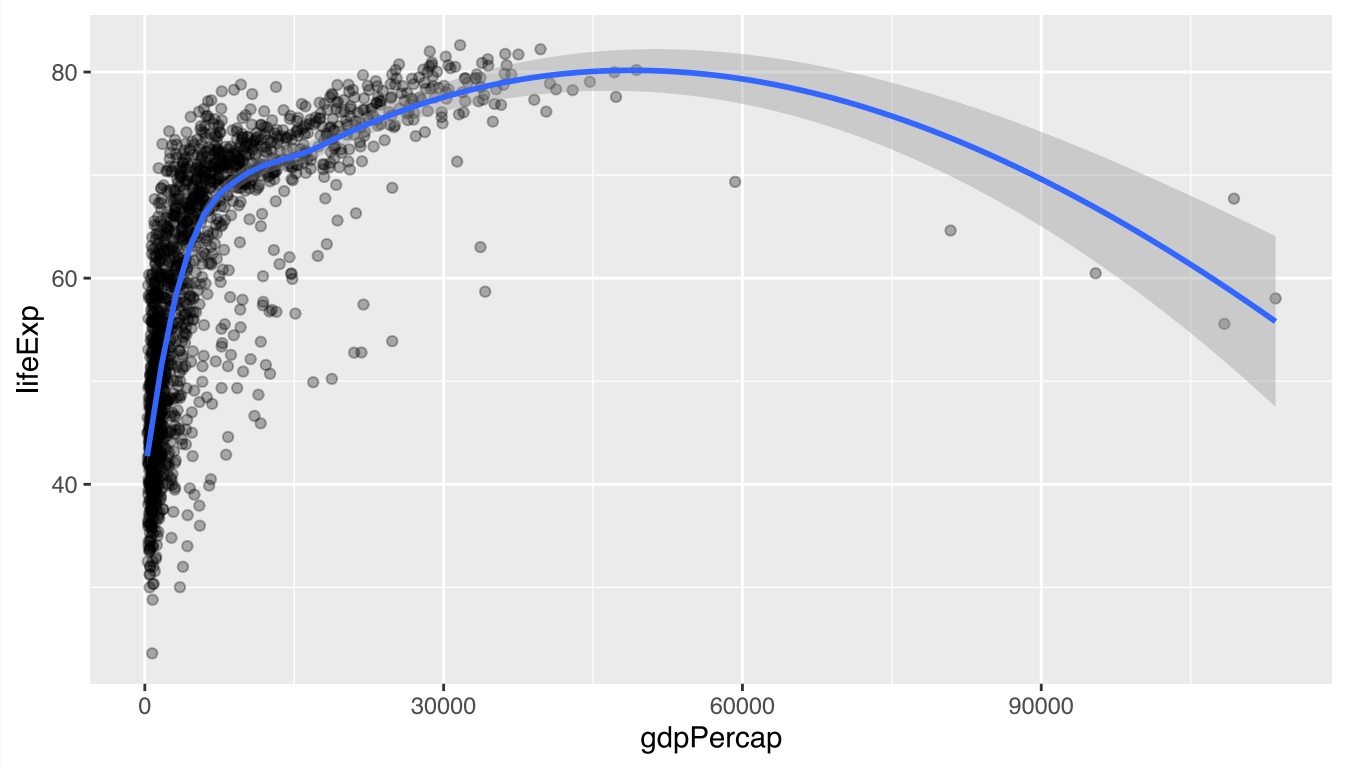


2. Geoms (cont.)

A polynomial fit might be more appropriate.

```
p + geom_point(alpha = 0.3)+geom_smooth(method="loess")
```

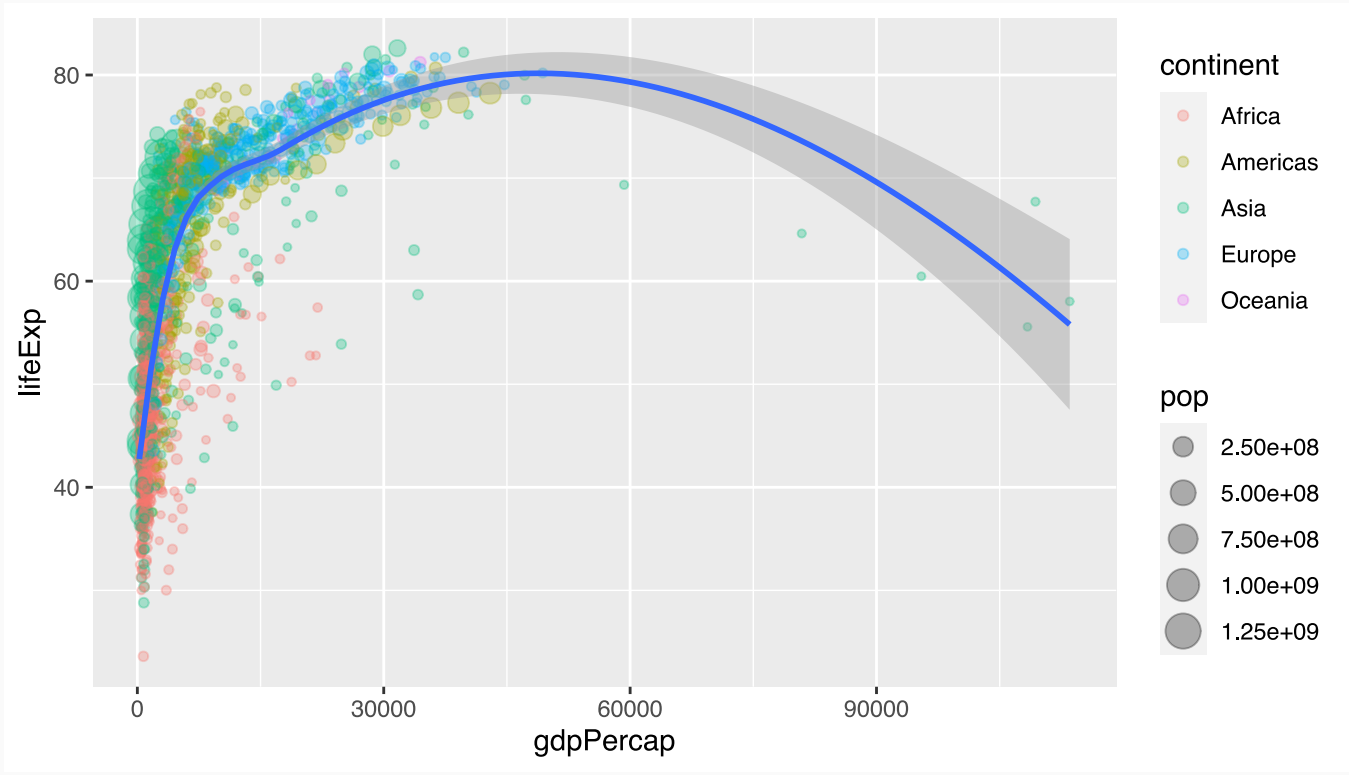
```
## `geom_smooth()` using formula 'y ~ x'
```



2. Geoms (cont.)

```
p + geom_point(aes(size = pop, col = continent),alpha = 0.3) +
  geom_smooth(method="loess")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



2. Geoms (cont.)

The previous plot provides a good illustration of the power (or effect) that comes from assigning aesthetic mappings "globally" vs in the individual geom layers.

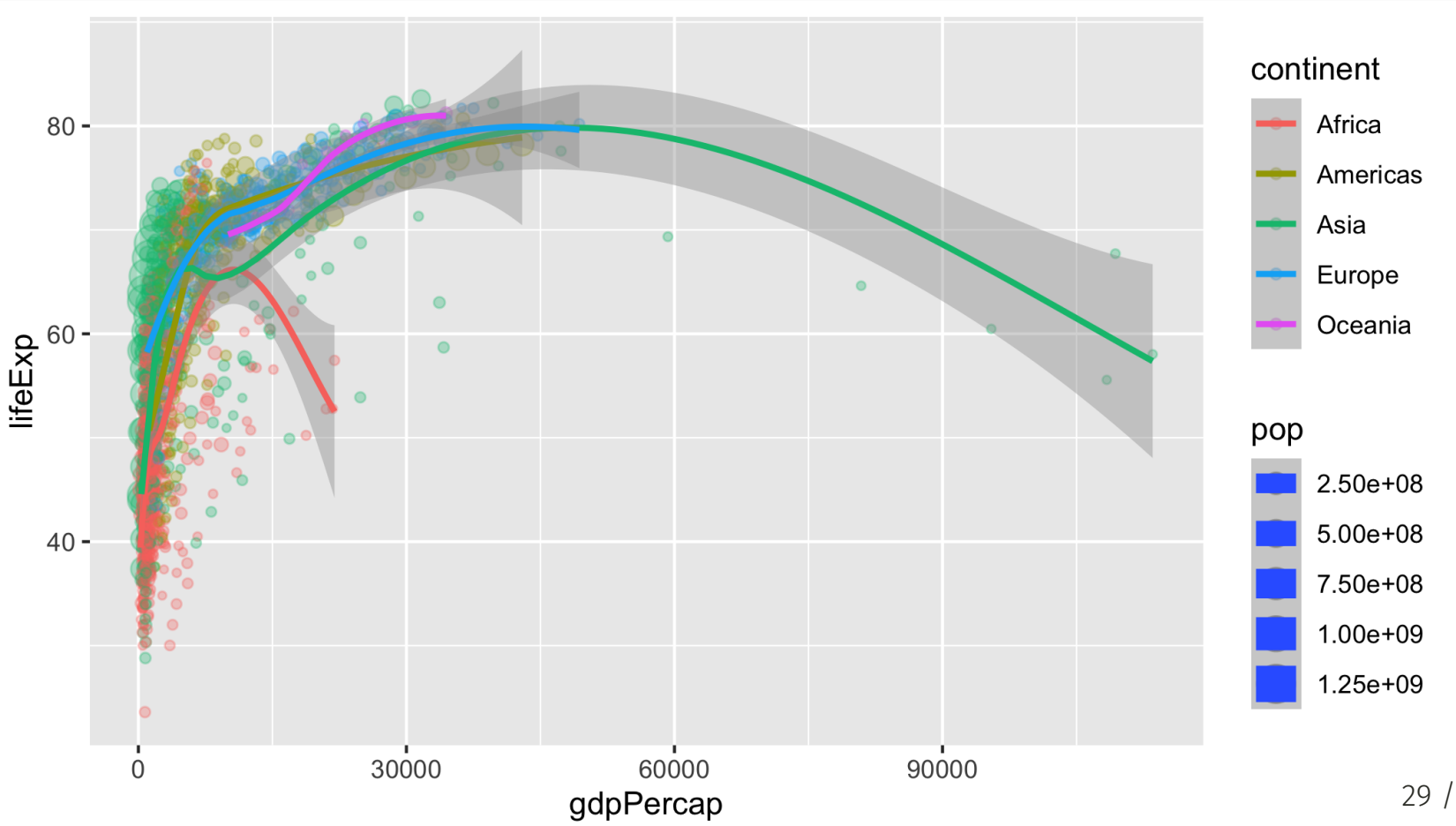
- Compare: What happens if you run the below code chunk?

```
p_bad = ggplot(data = gapminder,
               aes(x = gdpPercap, y = lifeExp, size = pop, col = continent)) +
  geom_point(alpha = 0.3)  +
  geom_smooth(method = "loess")
```

2. Geoms (cont.)

```
p_bad
```

```
## `geom_smooth()` using formula 'y ~ x'
```



2. Geoms (cont.)

Similarly, note that some geoms only accept a certain mappings. E.g. geom_density() doesn't know what to do with the "y" aesthetic mapping.

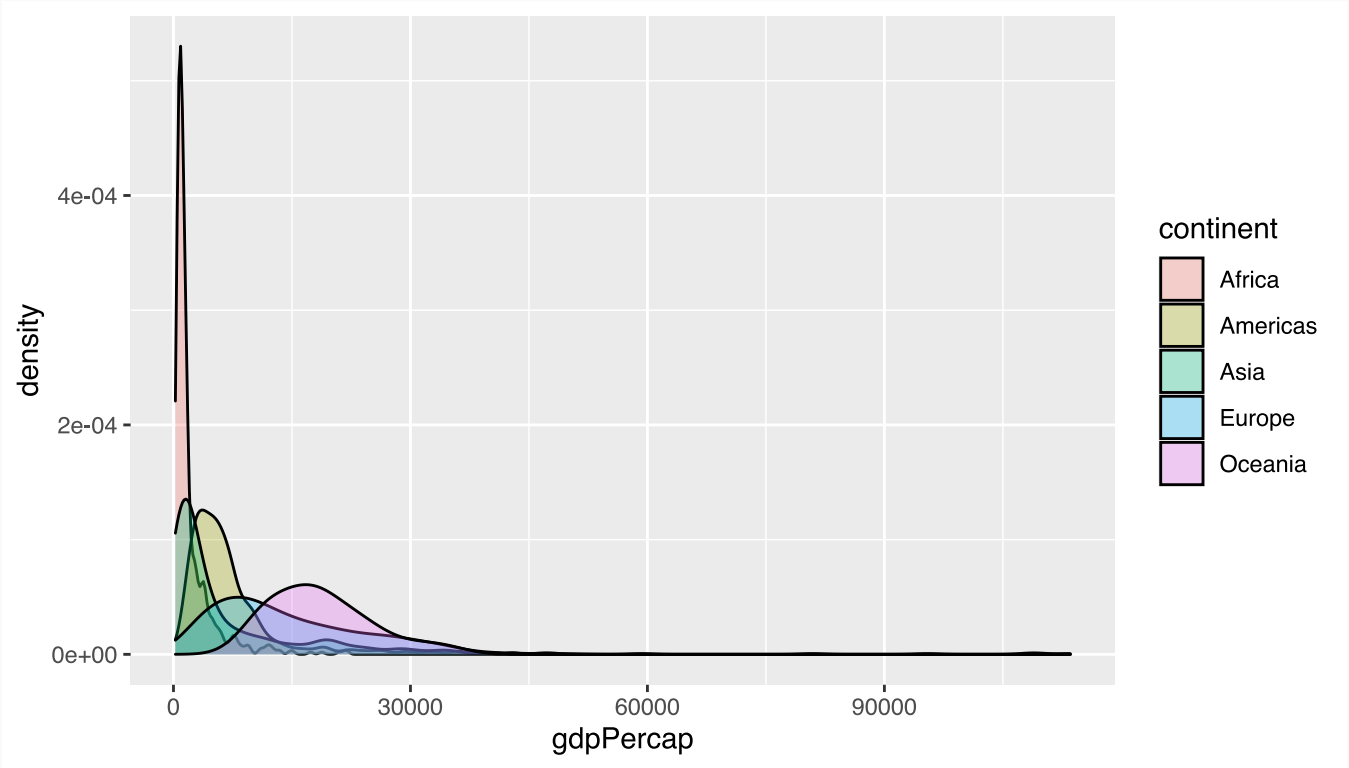
```
p + geom_density()
```

```
## Error: geom_density requires the following missing aesthetics: y
```

2. Geoms (cont.)

We can fix that by being more careful about how we build the plot.

```
ggplot(data = gapminder) + ## i.e. No "global" aesthetic mappings"
geom_density(aes(x = gdpPercap, fill = continent), alpha=0.3)
```



3. Build your plot in layers

We've already seen how we can chain (or "layer") consecutive plot elements using the `+` connector.

- The fact that we can create and then re-use an intermediate plot object (e.g. "p") is testament to this.

But it bears repeating: You can build out some truly impressive complexity and transformation of your visualization through this simple layering process.

- You don't have to transform your original data; ggplot2 takes care of all of that.
- For example (see next slide for figure).

```
p2 =
  p +
  geom_point(aes(size = pop, col = continent), alpha = 0.3) +
  scale_color_brewer(name = "Continent", palette = "Set1") + ## Different colour s
  scale_size(name = "Population", labels = scales::comma) + ## Different point (i.
  scale_x_log10(labels = scales::dollar) + ## Switch to logarithmic scale on x-axi
  labs(x = "Log (GDP per capita)", y = "Life Expectancy") + ## Better axis titles
  theme_minimal() ## Try a minimal (b&w) plot theme
```

3. Build your plot in layers (cont.)

What else?

I will be honest, a lot of the more advanced aspects of ggplot2 I do not have memorized.

- I don't know how many times I have visited this Stack Overflow page on centering a title in ggplot2.

One of the more complicated things is changing the colors of the lines/points manually.

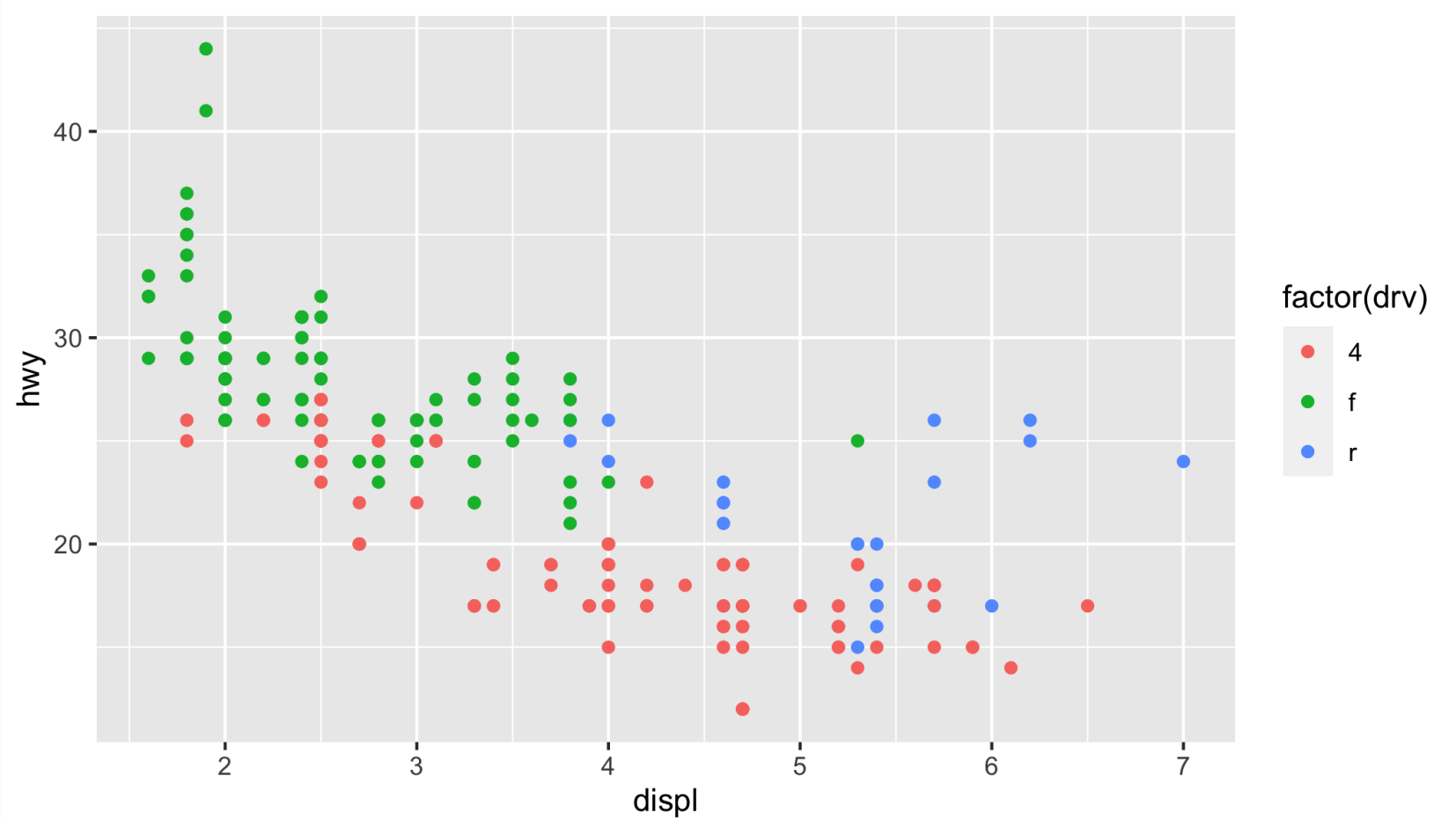
Let's go through an example of how to do this.

Scale Manually

We will use the `mpg` data set that comes with `ggplot2`. We will plot the engine size on the x-axis and the highway mpg on the y-axis. As well, let's color the points by the drive type (4-wheel, front-wheel, or rear-wheel).

```
bp = ggplot(mpg)+geom_point(aes(x=displ,y=hwy,color=factor(drv)))
```

Scale Manually (cont.)



Scale Manually (cont.)

These colors are fine, but what if we wanted other colors?

Here is a reference for the names of the different colors in ggplot2.

Let's make 4-wheel drive lightskyblue1, rear-wheel drive as springgreen1, and front-wheel drive as firebrick1.

```
man_cols = c("4" = "lightskyblue1","r" = "springgreen1","f"="firebrick1")
```

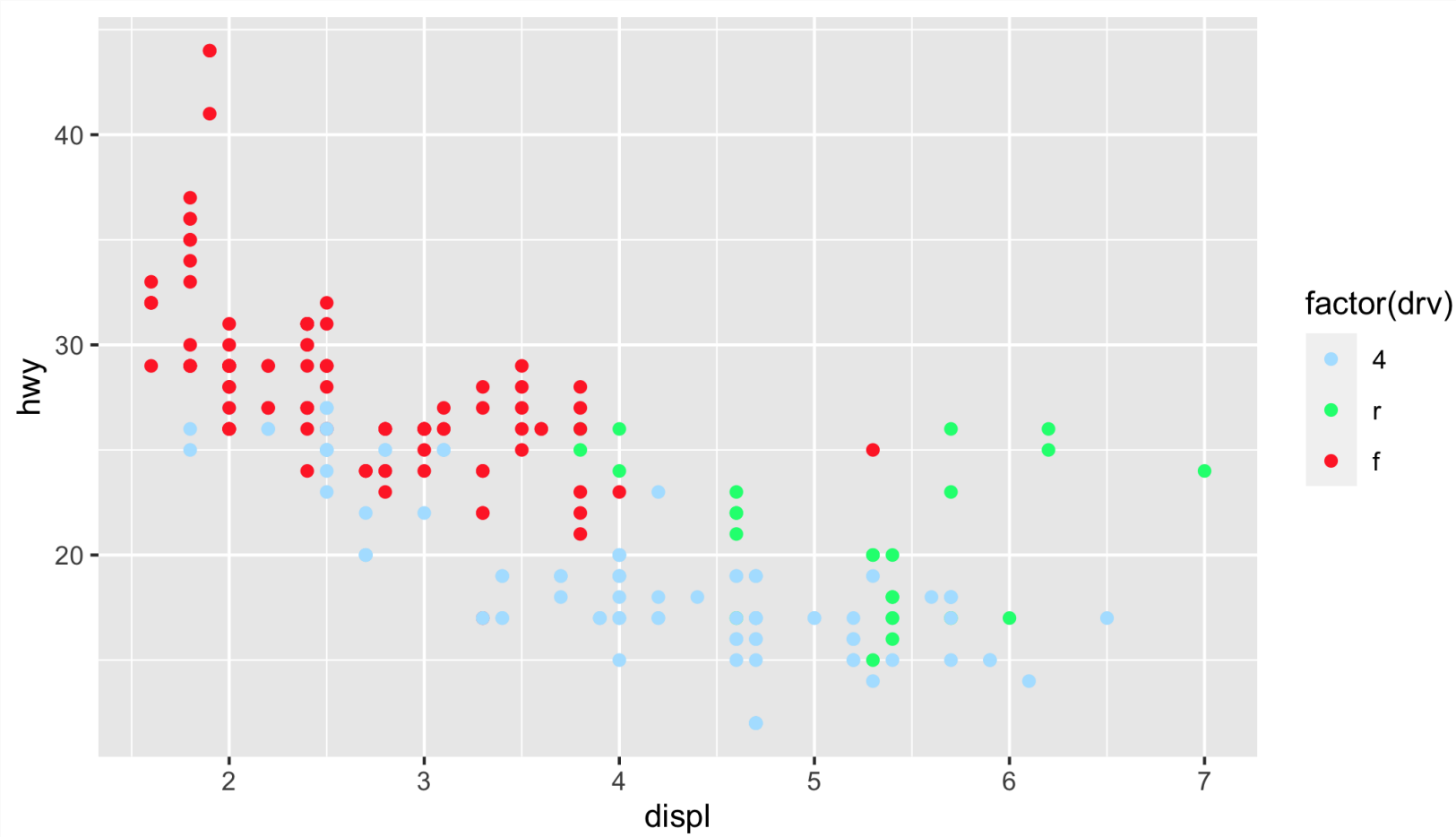
Notice, the value of the observation is on the LHS of the equal sign, and the name of the color is on the RHS.

```
sort(unique(mpg$drv))
```

```
## [1] "4" "f" "r"
```

Scale Manually (cont.)

```
bp + scale_colour_manual(values=man_cols)
```



Scale Manually (cont.)

What if we want to change the legend title?

```
bp + scale_colour_manual(name="Drive Type",values=man_cols)
```

Update Base Plot

```
bp = bp + scale_colour_manual(name="Drive Type",values=man_cols)
```

What else? Titles

Let's add titles:

```
bp + xlab("Engine Size") + ylab("Highway MPG") + ggtitle("Highway MPG by Engine Size")
```

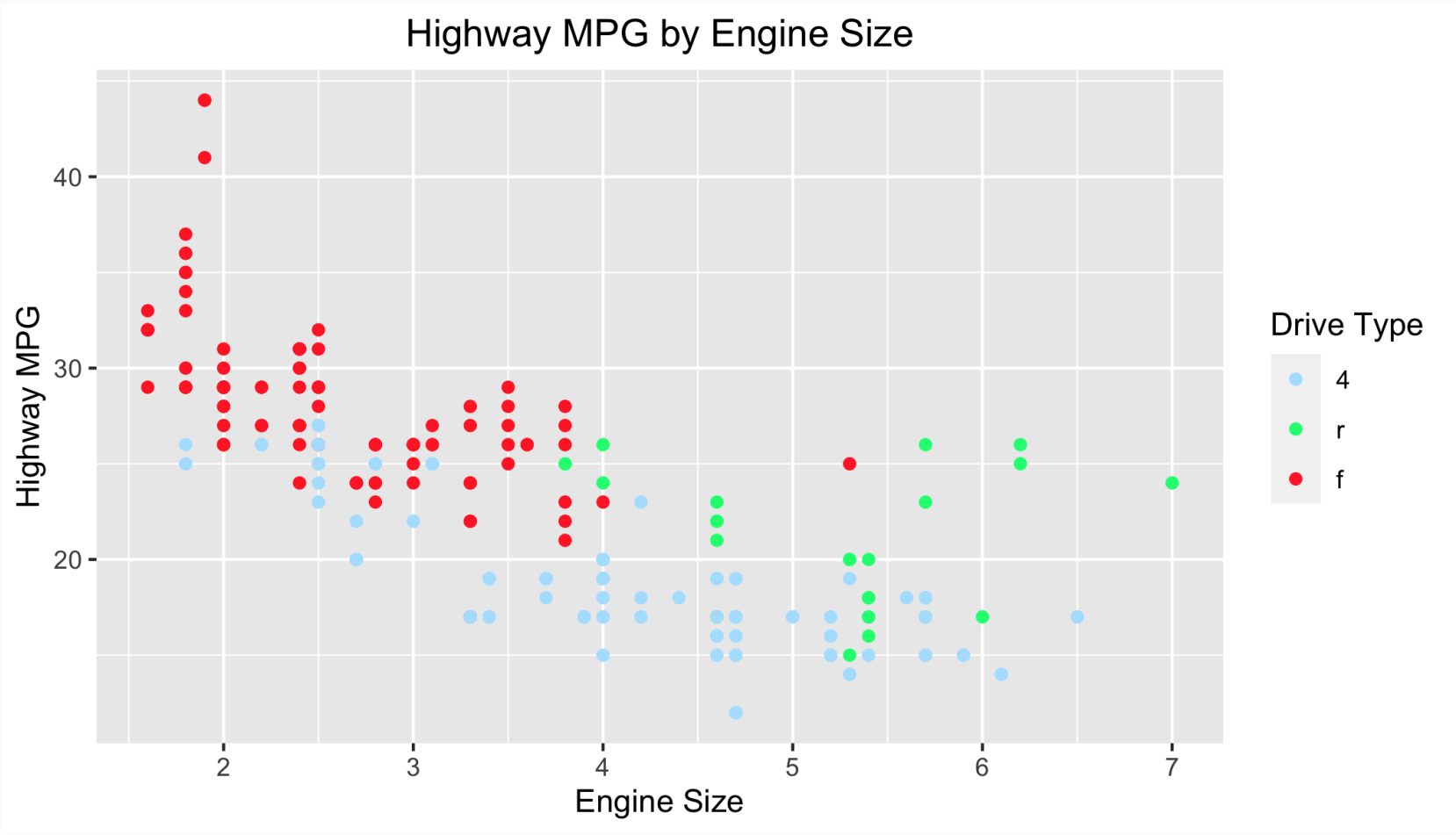
What Else? Titles

Notice above that the main title isn't centered. We can fix that. But first, let's update the base plot.

```
bp = bp + xlab("Engine Size") + ylab("Highway MPG") +
  ggtitle("Highway MPG by Engine Size")
```

What Else? Titles

```
bp + theme(plot.title = element_text(hjust = 0.5))
```



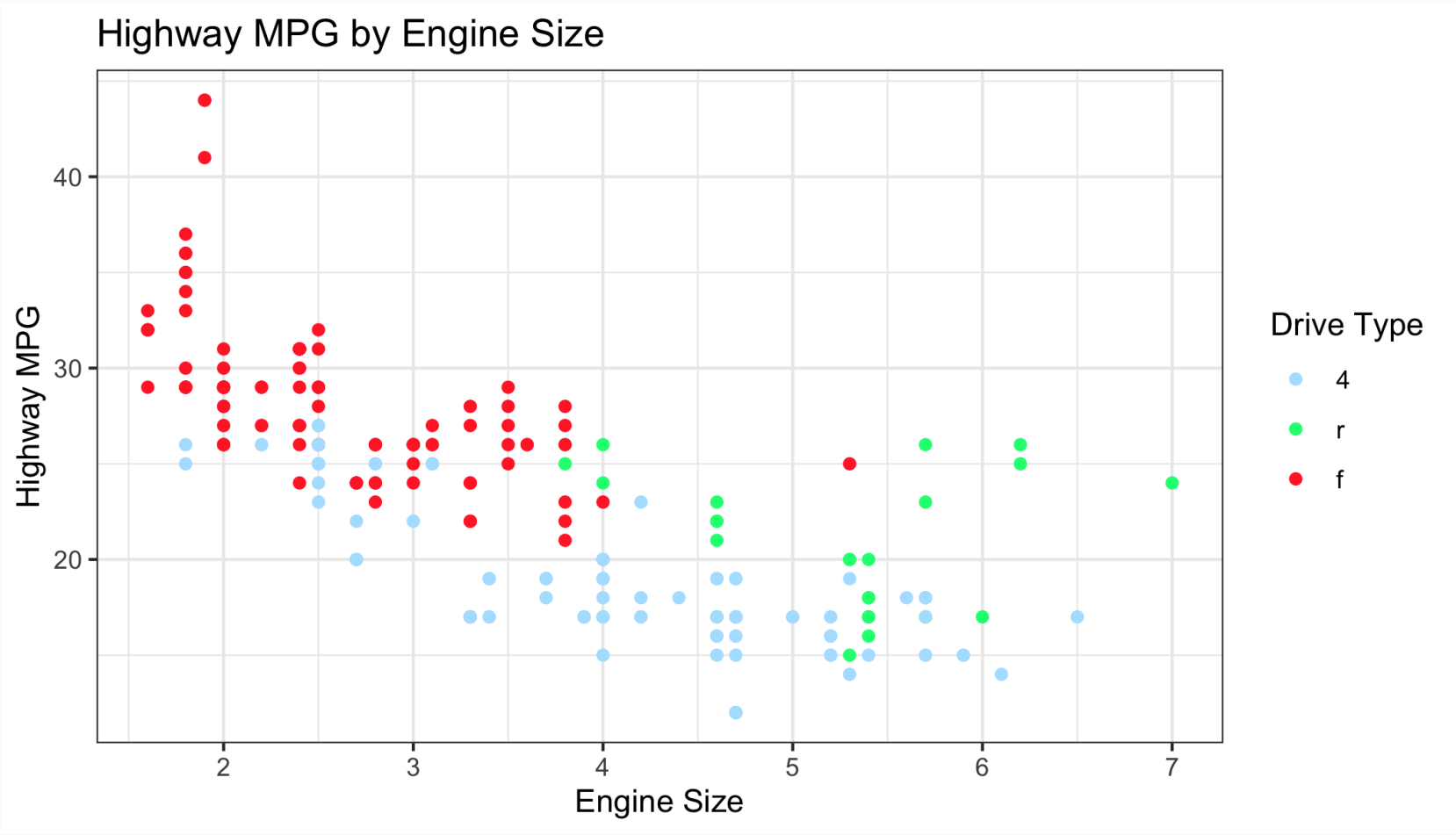
What Else? Themes

Be careful with the above code. If you change the base theme, you will need to include the centering code after that. See what happens if you don't.

Let's say we wanted to add the black and white theme. That is `theme_bw()`. If we include this after the code for centering the title, the `theme()` will be overwritten by whatever is set in `theme_bw()`.

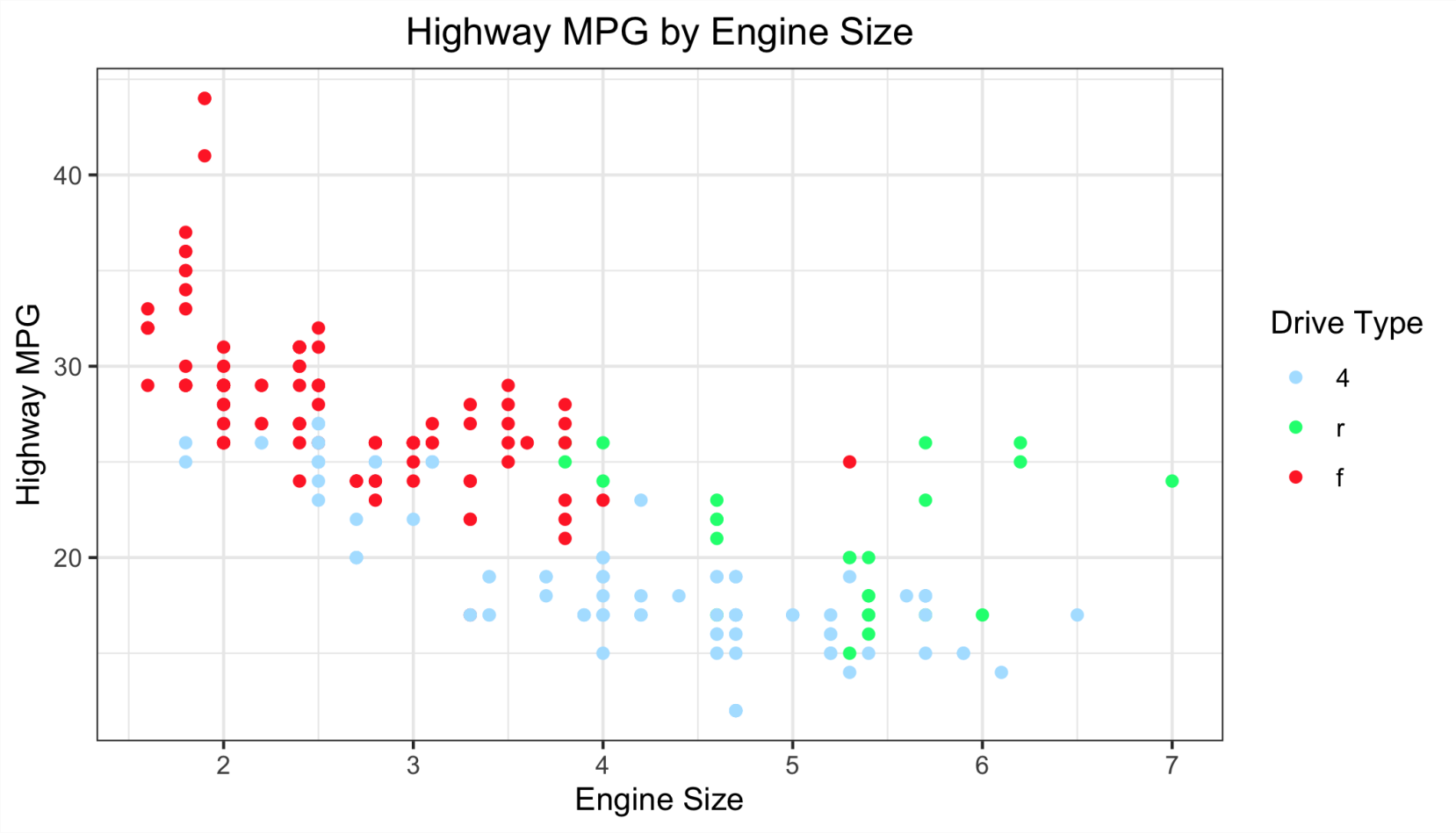
What Else? Themes (cont.)

```
bp + theme(plot.title = element_text(hjust = 0.5)) + theme_bw()
```



What Else? Themes (cont.)

```
bp + theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```



What Else? Other Geoms and Colors

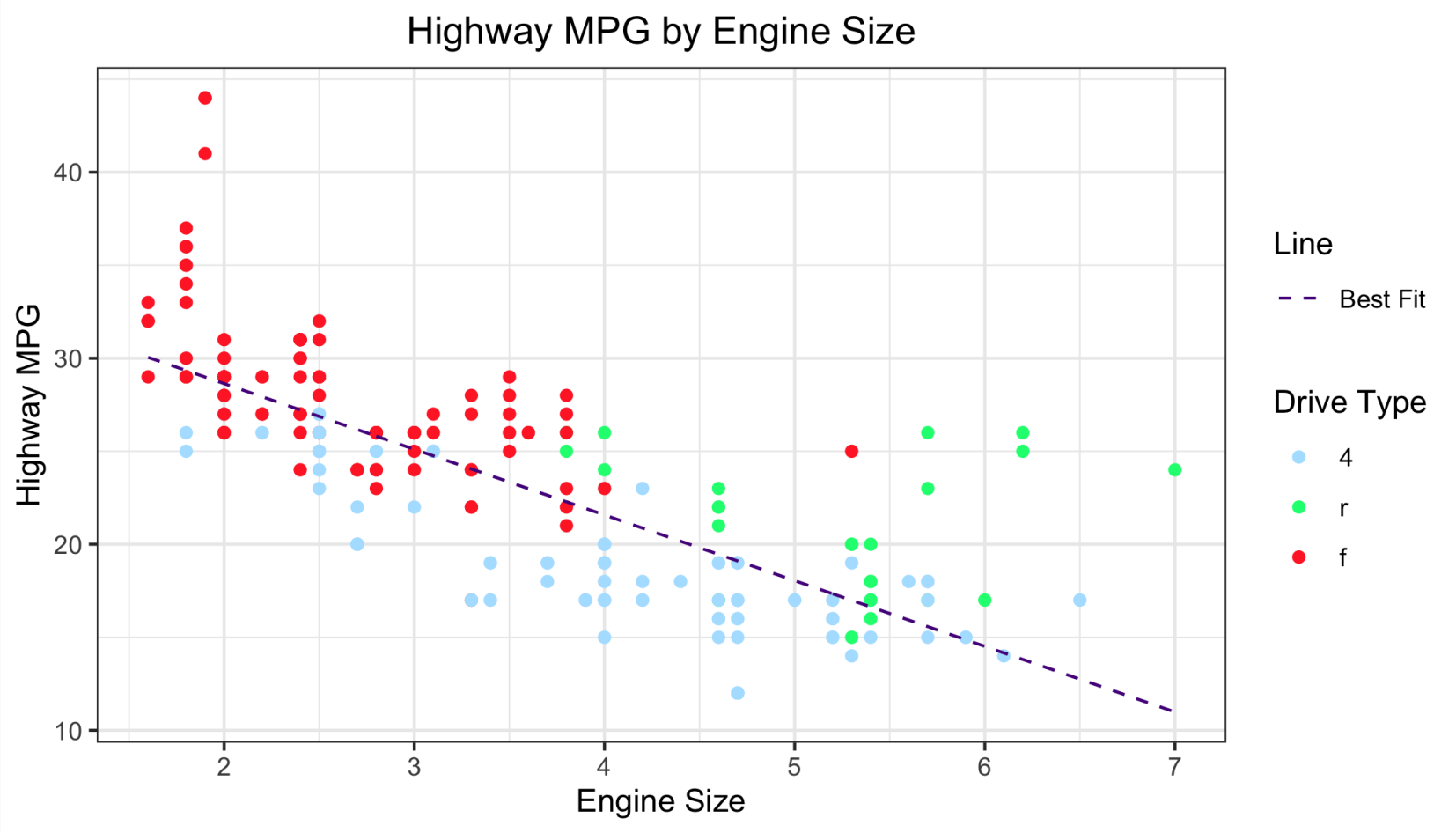
Let's do one more thing that will require a little work. Let's add a line of best fit without `geom_smooth()`. The reason I want to do this is to manually change the color which can be difficult.

```
hwy_mod = lm(hwy ~ displ,data=mpg)
mpg$hwy_hat = predict(hwy_mod)
```

What Else? Other Geoms and Colors

```
man_cols = c("4" = "lightskyblue1","r" = "springgreen1","f"="firebrick1")
bp = ggplot(mpg)+geom_point(aes(x=displ,y=hwy,color=factor(drv))) +
  geom_line(aes(x=displ,y=hwy_hat,linetype = "Best Fit"),color="purple4") +
  scale_colour_manual(name="Drive Type",values=man_cols) +
  scale_linetype_manual(name="Line",values = c("Best Fit" = "dashed")) +
  xlab("Engine Size") + ylab("Highway MPG") +
  ggtitle("Highway MPG by Engine Size") + theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

What Else? Other Geoms and Colors



What else? (cont.)

There's a lot more to say, but I think we'll stop now for today's lecture.

I want you to do some reading and practice on your own. Pick either of the following (or choose among the litany of online resources) and work through their examples:

- Chapter 3 of *R for Data Science* by Hadley Wickham and Garett Grolemund.
- *Data Visualization: A Practical Guide* by Kieran Healy.
- Designing ggplots by Malcom Barrett.

Next lecture: Intro to Data Science.