

# Class 5: Data Visualization with ggplot2

Krysten Jones (A10553682)

## Using ggplot2

To use ggplot2, first we need to install it on our computers. To do this we will use the function `install.packages()`. We could put this in a code chunk, but every time we click render it will install the package again. Instead install in the console (only need to do once)

Before we use any package functions, we have to load them up with a 'library' call, like below:

```
library(ggplot2)
```

First let's examine the structure of our data frame. You can use the head or tail functions here to view the first or last 6 rows respectively

```
head(cars)
```

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

The structure function will give you even more information

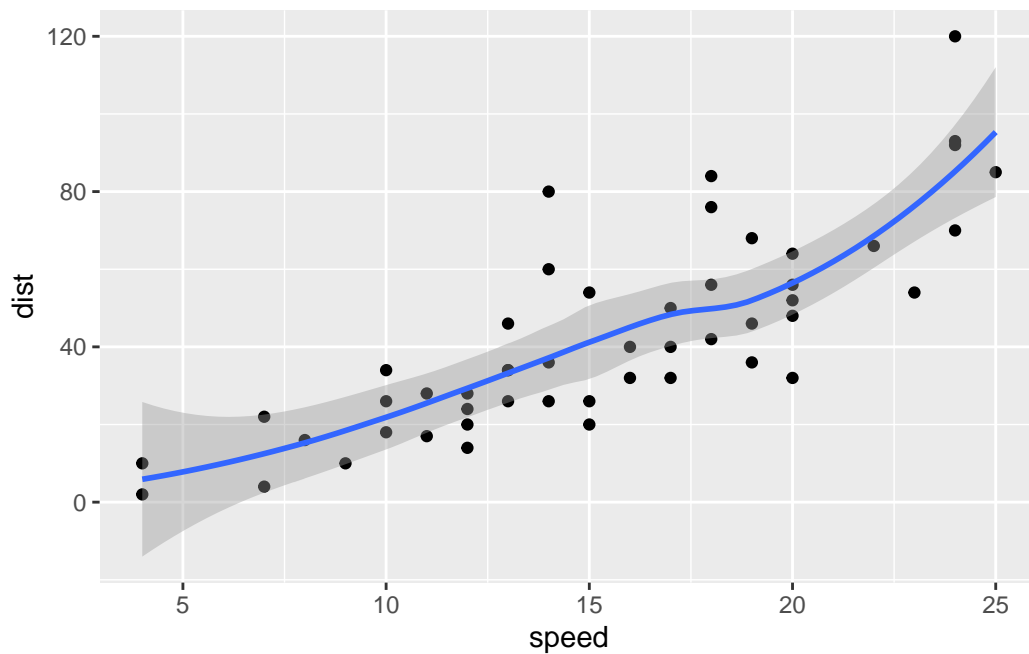
```
str(cars)
```

```
'data.frame':  50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

Plotting in ggplot: requires 3 different aspects. The first is the data, then the aesthetics `aes()` then the geometry/graph type `geom()` - data (the stuff we want to plot as a data.frame) - aesthetics (how the data map to the plot/what it looks like) - geometry (how we want something drawn)

```
ggplot(cars, aes(speed, dist)) +  
  geom_point() +  
  geom_smooth()
```

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

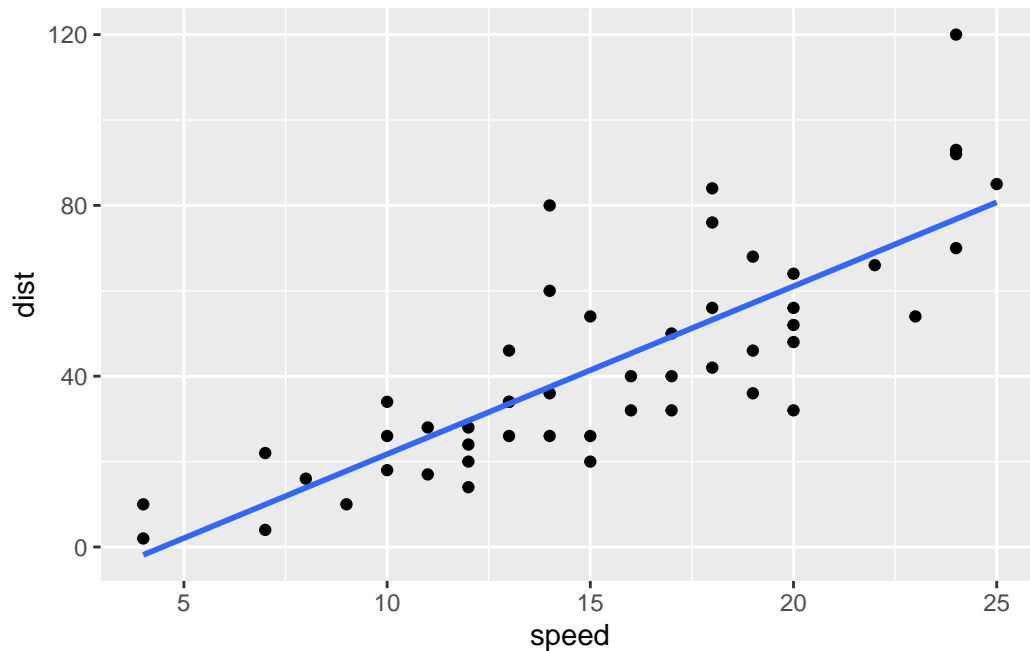


#best practice here to code on different lines, makes it easier to find errors later

method = "lm" asks for a trend line, the gray area around it is the standard error of the mean (SEM) Will add things line by line, so if want something overlaid, then add it at the end

```
ggplot(cars, aes(speed, dist)) +  
  geom_point() +  
  geom_smooth(method = "lm", se=FALSE)
```

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



`se = FALSE` will remove the standard error line

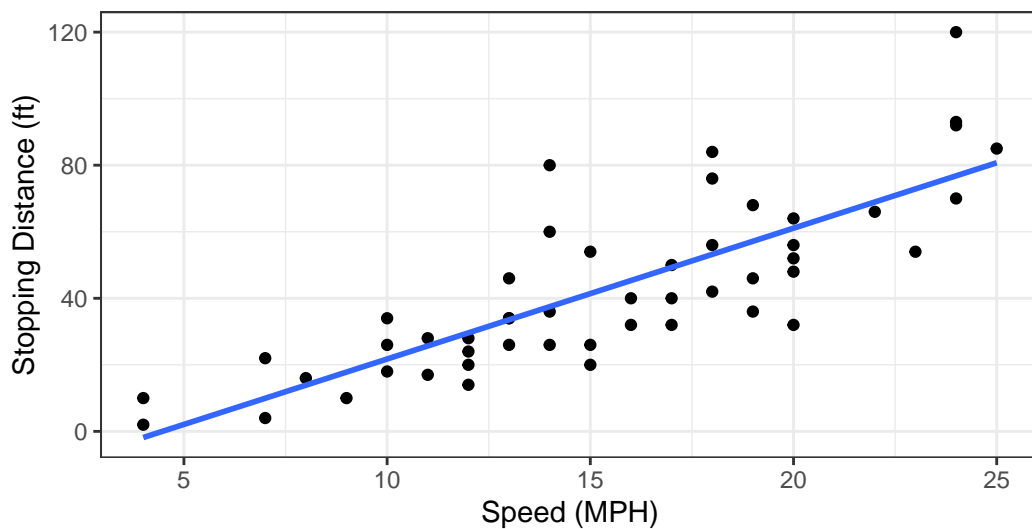
Now we're going to add more labels. The `aes( )` function will decide what DATA to put on the x and y axis. The `labs()` function is where you can customize what you want each axis to say.

```
ggplot(cars, aes(speed, dist)) +
  geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  labs(title="Speed and Stopping Distances of Cars",
       x="Speed (MPH)",
       y="Stopping Distance (ft)",
       subtitle = "Increased Stopping Distance with Speed",
       caption="Dataset: 'cars'") +
  theme_bw()
```

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

## Speed and Stopping Distances of Cars

Increased Stopping Distance with Speed



Dataset: 'cars'

```
url <- "https://bioboot.github.io/bimm143_S20/class-material/up_down_expression.txt"
genes <- read.delim(url)
head(genes)
```

	Gene	Condition1	Condition2	State
1	A4GNT	-3.6808610	-3.4401355	unchanging
2	AAAS	4.5479580	4.3864126	unchanging
3	AASDH	3.7190695	3.4787276	unchanging
4	AATF	5.0784720	5.0151916	unchanging
5	AATK	0.4711421	0.5598642	unchanging
6	AB015752.4	-3.6808610	-3.5921390	unchanging

```
nrow(genes)
```

```
[1] 5196
```

```
colnames(genes)
```

```
[1] "Gene"      "Condition1" "Condition2" "State"
```

1

[illegible]



[illegible]



[illegible]

[illegible]



[illegible]

[illegible]



[illegible]





[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]





```

[5116] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5121] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5126] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5131] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5136] "up"          "unchanging" "unchanging" "unchanging" "unchanging"
[5141] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5146] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5151] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5156] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5161] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5166] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5171] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5176] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5181] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5186] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5191] "unchanging" "unchanging" "unchanging" "unchanging" "unchanging"
[5196] "unchanging"

```

```
table(genes$State)
```

down	unchanging	up
72	4997	127

so there are 5169 rows, and 4 different columns including state. So the above is saying from the genes data.frame, choose the “State” column.

```
sum(genes$State == "up")
```

```
[1] 127
```

The above code will first go through the state column and turn everything True or False, it will then sum it for you so you know how many are upregulated vs downregulated. This is an alternative to table above

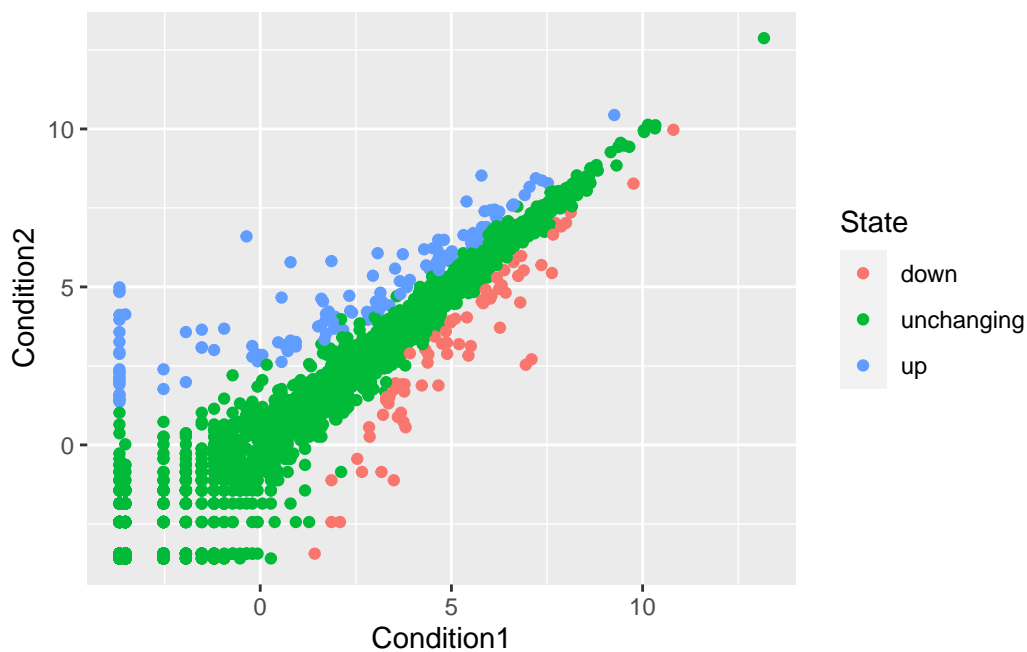
`#!/ eval:` `false` or `echo=false` in a code box will make R not run the code in the code box. This is an alternative to putting “#” in front of each line of code you do not want run, but may still want included

```
round( table(genes$State)/nrow(genes) * 100, 2)
```

down	unchanging	up
1.39	96.17	2.44

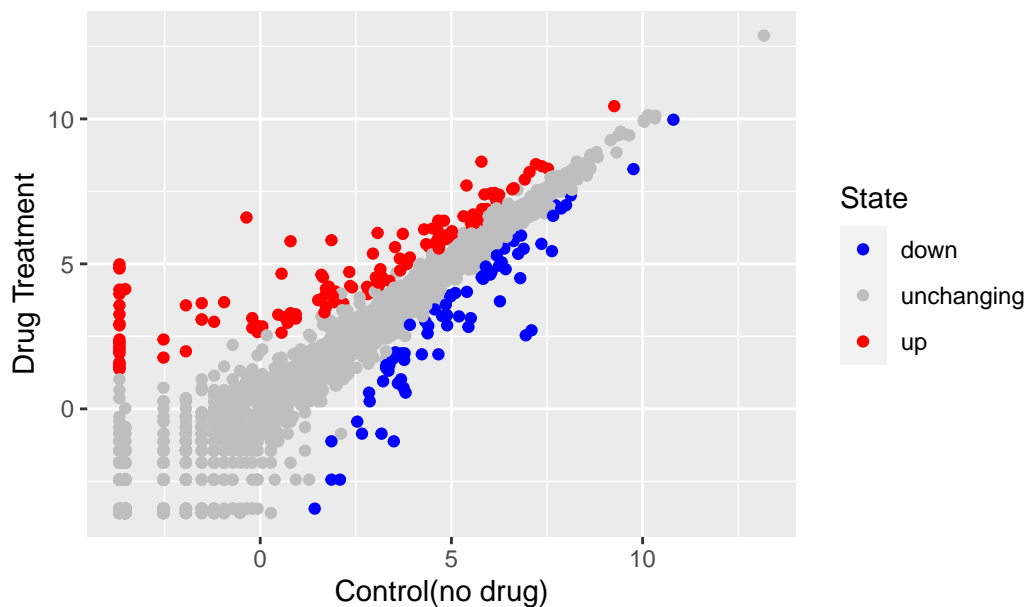
The above code is asking to use the dataframe genes, look at the state column and divide by the total numbers of rows in the genes dataframe. Then multiply by 100 and finally round it to 2 significant figures (what the 2 in the code above is referring to).

```
p <- ggplot(genes, aes(Condition1, Condition2, col=State)) +
  geom_point()
p
```



```
p + scale_colour_manual( values=c("blue","gray","red") ) +
  labs(title = "Gene Expression Changes Upon Drug Treatment",
       x= "Control(no drug)",
       y= "Drug Treatment")
```

## Gene Expression Changes Upon Drug Treatment



Order matters here, you can try different color combinations depending on what you want to highlight.

We're now going to use gapfinder dataset with the dplyr program. Just like for ggplot, you're going to want to install it from your console (and not have it run as part of this file) so you don't have to re-install it everytime you change something on this file. `install.packages("gapminder")` and `install.packages("dplyr")`

```
library(gapminder)
library(dplyr)
```

Attaching package: 'dplyr'

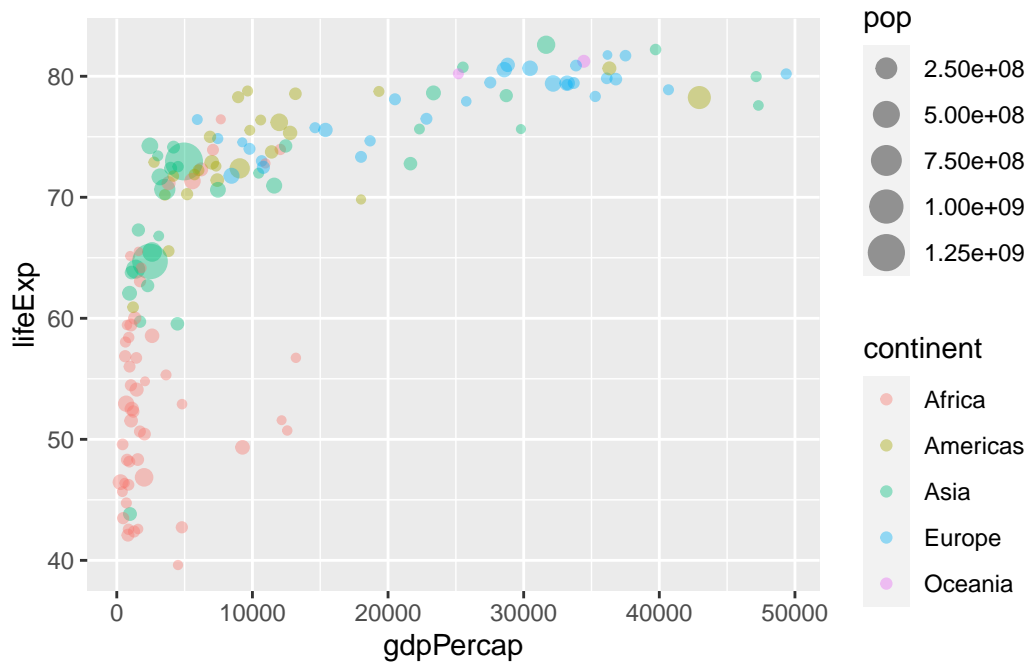
The following objects are masked from 'package:stats':

`filter`, `lag`

The following objects are masked from 'package:base':

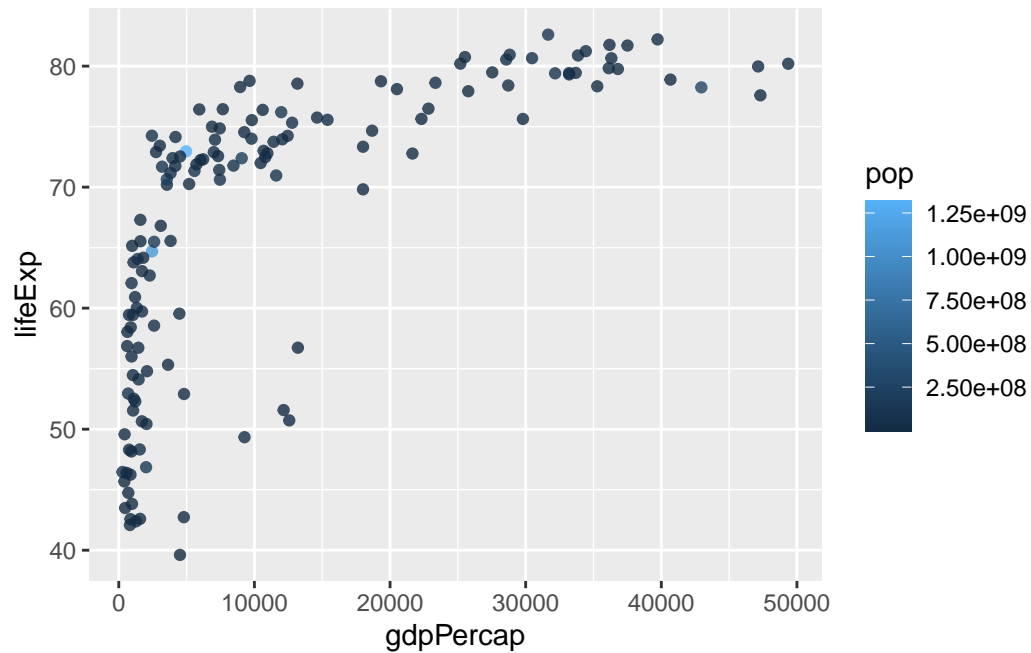
`intersect`, `setdiff`, `setequal`, `union`

```
gapminder_2007 <- gapminder %>% filter(year==2007)
ggplot(gapminder_2007,
       aes(gdpPercap, lifeExp, color = continent, size = pop)) +
  geom_point(alpha=0.4)
```



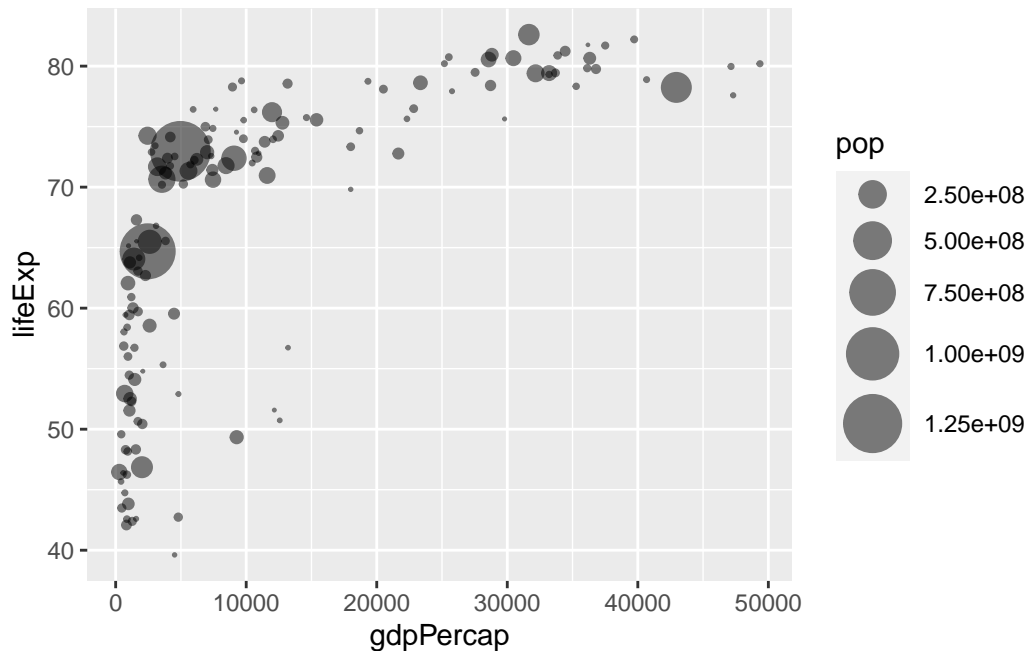
```
ggplot(gapminder_2007) +
  aes(x = gdpPercap, y = lifeExp, color = pop) +
  geom_point(alpha=0.8)
```





Now we've plotted a gradient of color based on polulation instead of using discrete categories

```
ggplot(gapminder_2007) +  
  geom_point(aes(x = gdpPercap, y = lifeExp, size = pop),  
             alpha=0.5) +  
  scale_size_area(max_size = 10)
```

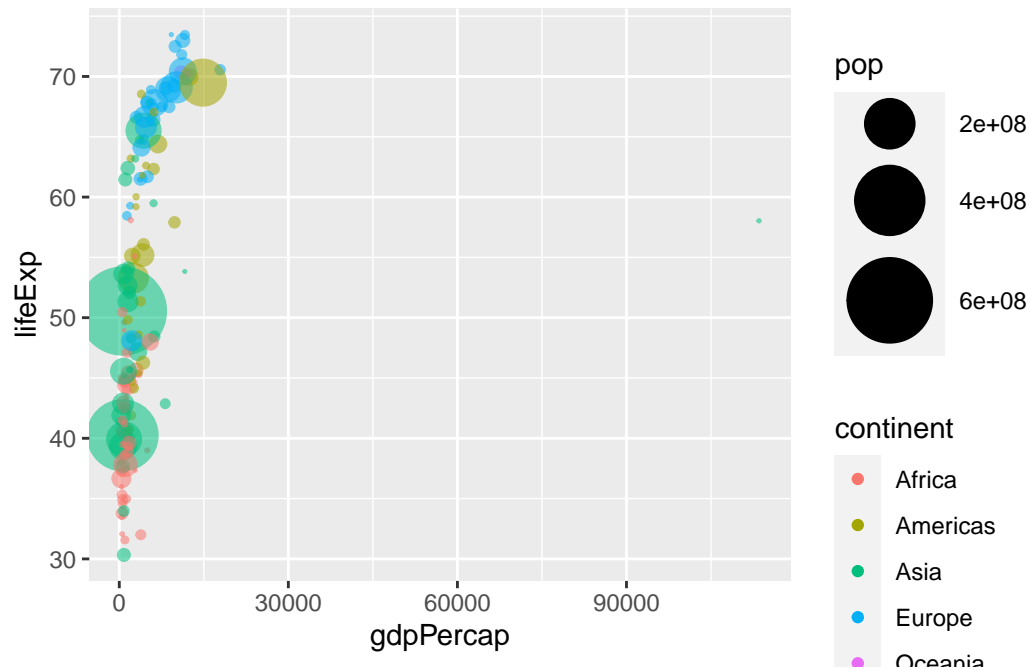


Here we are defining the size of our bubbles by population, but also making them partially transparent so we can see our data better

```
gapminder_1957 <- gapminder %>% filter(year==1957)
```

Now: Use `scale_size_area()` so that the point sizes reflect the actual population differences and set the `max_size` of each point to 15. Also, set the opacity/transparency of each point to 70% using the `alpha=0.7` parameter

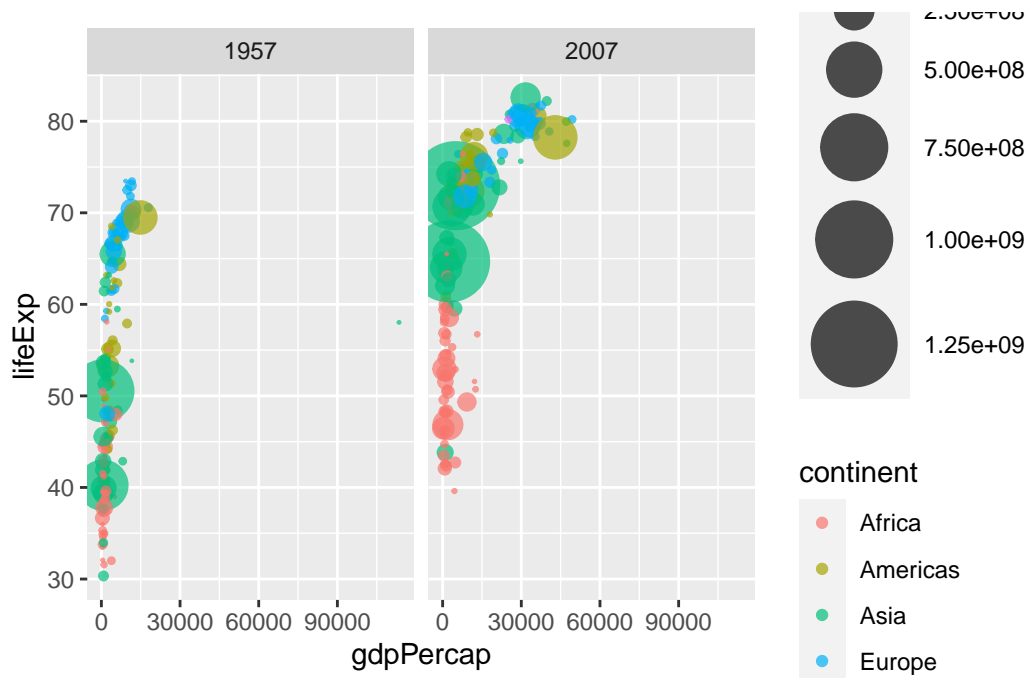
```
ggplot(gapminder_1957,
       aes(gdpPercap, lifeExp, color = continent, size = pop, alpha=0.7)) +
  geom_point() +
  scale_size_area(max_size = 15)
```



Now facet it (aka split into different graphs)

```
both <- gapminder %>% filter(year==1957 | year==2007)

ggplot(both) +
  geom_point(aes(x = gdpPercap, y = lifeExp, color=continent,
                 size = pop), alpha=0.7) +
  scale_size_area(max_size = 15) +
  facet_wrap(~year)
```



## Section 8: Bar Charts

```
gapminder_top5 <- gapminder %>%
  filter(year==2007) %>%
  arrange(desc(pop)) %>%
  top_n(5, pop)
```

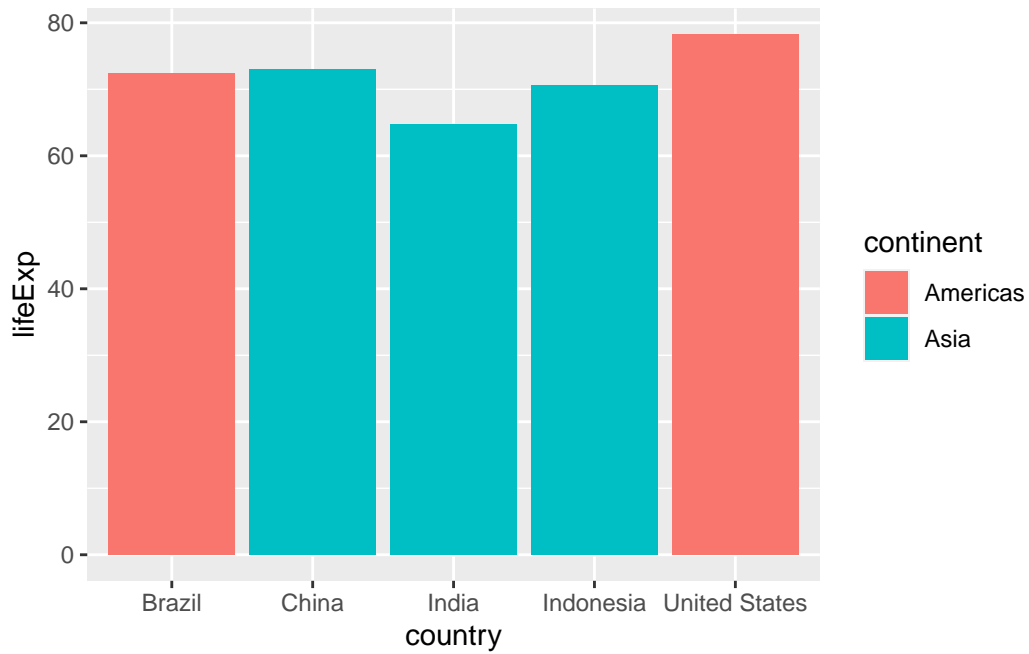
```
gapminder_top5
```

# A tibble: 5 x 6

	country	continent	year	lifeExp	pop	gdpPercap
	<fct>	<fct>	<int>	<dbl>	<int>	<dbl>
1	China	Asia	2007	73.0	1318683096	4959.
2	India	Asia	2007	64.7	1110396331	2452.
3	United States	Americas	2007	78.2	301139947	42952.
4	Indonesia	Asia	2007	70.6	223547000	3541.
5	Brazil	Americas	2007	72.4	190010647	9066.

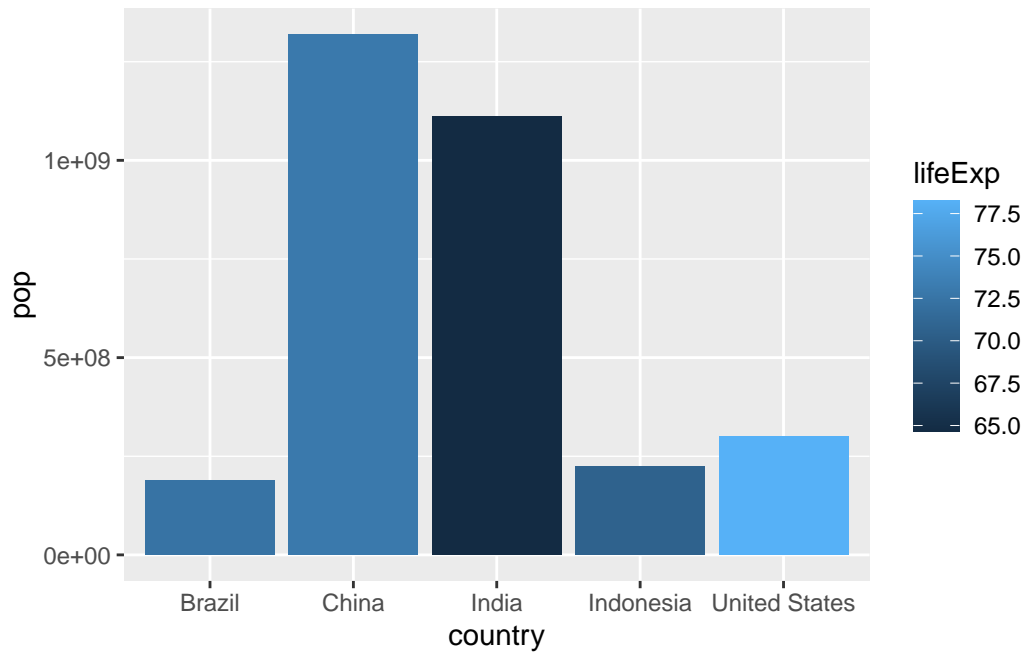
Life expectancy of the top 5 countries

```
ggplot(gapminder_top5) +  
  geom_col(aes(x = country, y = lifeExp, fill = continent))
```

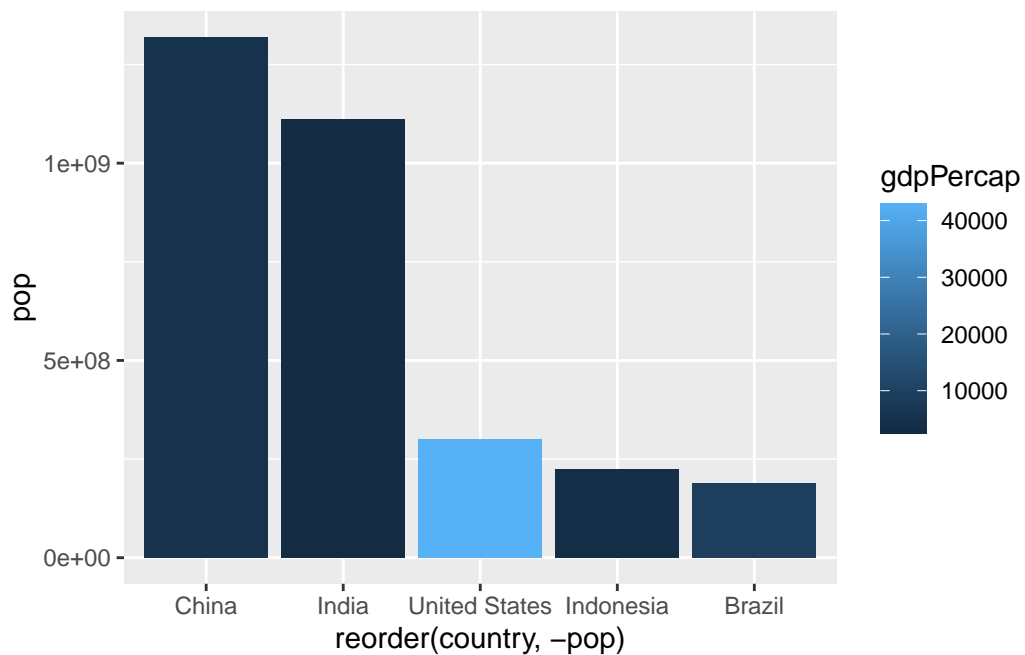


Let's try using a numerical value to color instead, so switch fill to life expectancy

```
ggplot(gapminder_top5) +  
  geom_col(aes(x = country, y = pop, fill = lifeExp))
```



```
ggplot(gapminder_top5) +  
  geom_col(aes(x=reorder(country, -pop), y = pop, fill = gdpPercap))
```



The reorder function will allow you to reorder the bars based on what you put in your arguments.