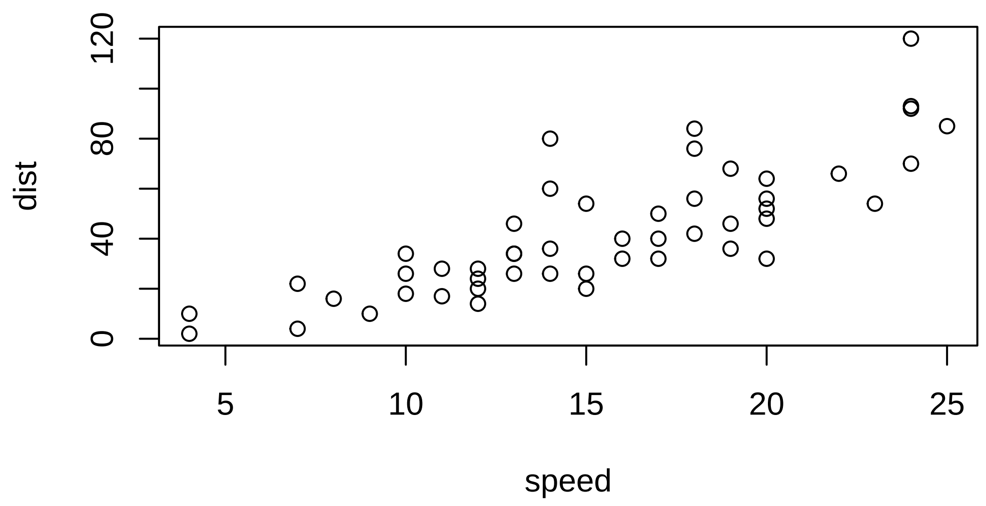
Class5_BGGN213: Using Ggplot2

10-12-22

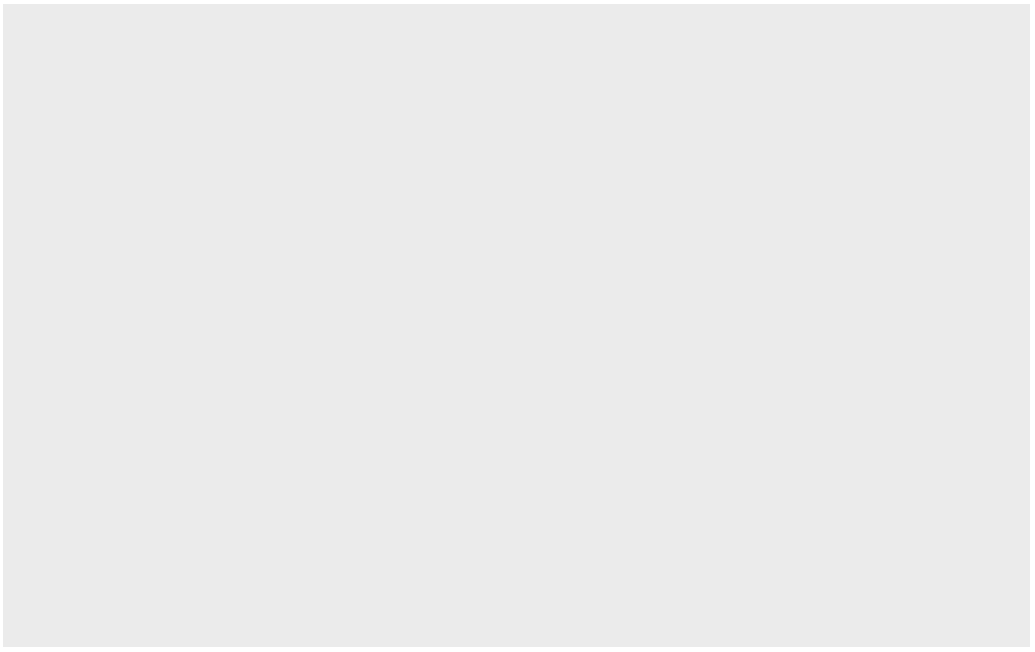
first plot R has base graphics we can use.

```
plot(cars)
```



Plot the same graph with ggplot2 Remember to install ggplot2 before using it. Use: 'install.packages()' function. Do it in the console and not in the report. To create plots with ggplot2 you first need to load the package using library(ggplot2).

```r
library(ggplot2)

ggplot(cars)
```

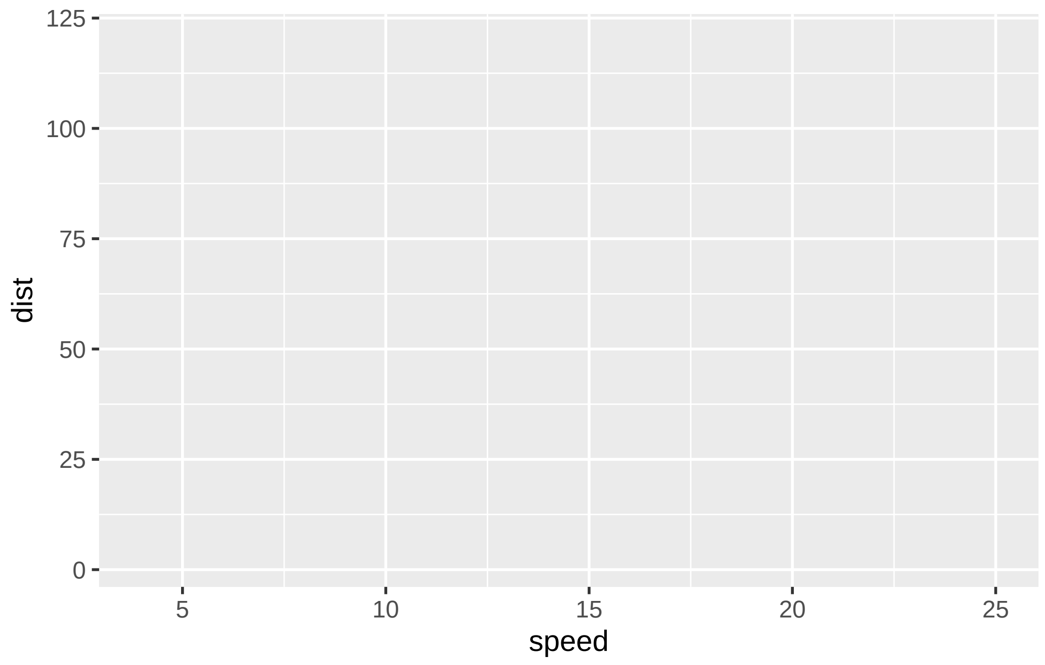


Note that this command does not plot anything but a blank gray canvas yet. The ggplot() function alone just defines the dataset for the plot and creates an empty base on top of which we will add additional layers to build up our plot.

ggplot needs three things: 1. the 'data=' defined to map to a certain data.frame 2. the 'aes()' function with the parameters for the plot 3. the 'geom()' function with the type of plot (i.e. points or lines or box)

Specifing aesthetic mappings with aes(). We will use the columns labeled speed and distance from the cars dataset to set the x and y aesthetics of our plot. Critically, we combine our call to the aes() function with our previous specification of the input dataset with the ggplot(cars) function call from above.

```r
ggplot(cars) +
  aes(x=speed, y=dist)
```

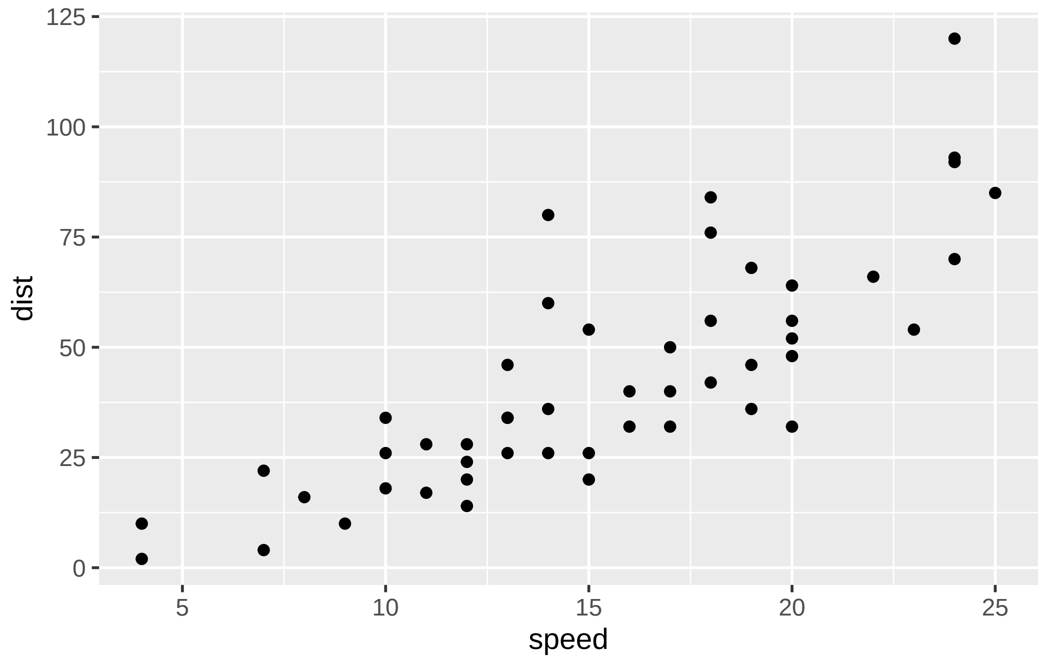


Now that we have our axes, we now need to add one of ggplot's geometric layers (or geoms) to define how we want to visualize our dataset.

Specifing a geom layer with geom_point() geom_line() produces a line plot, geom_bar() produces a bar plot, geom_boxplot() a box plot, geom_point() adds a scatter plot.

In this example, we use geom_line()

```
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point()
```

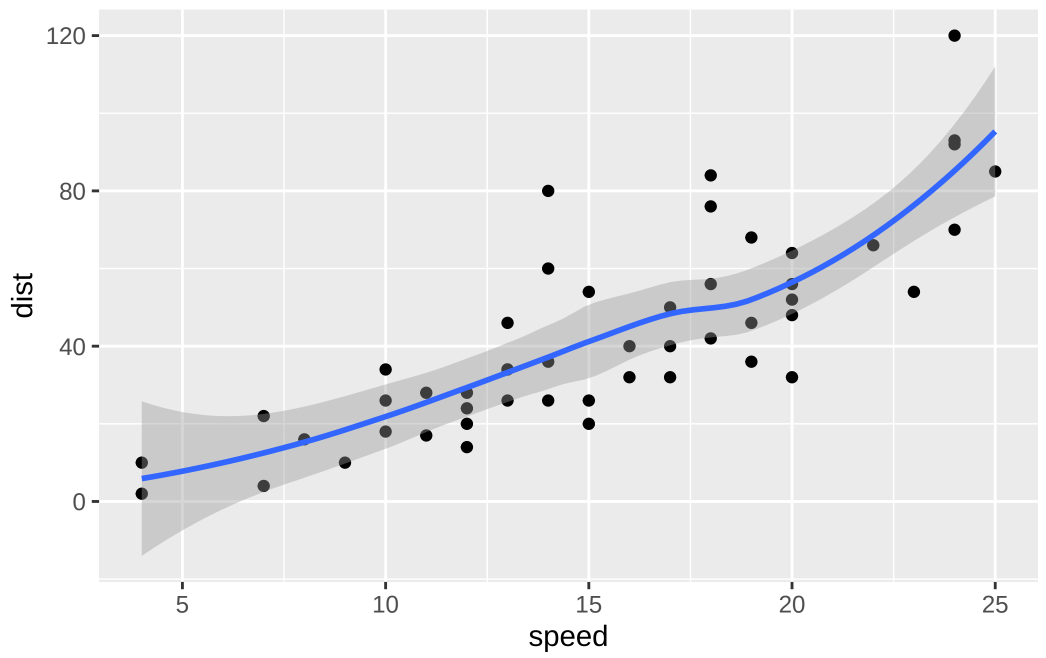


Now we have data being plotted!

To add a trend line to the data, add " `geom_smooth()` "

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

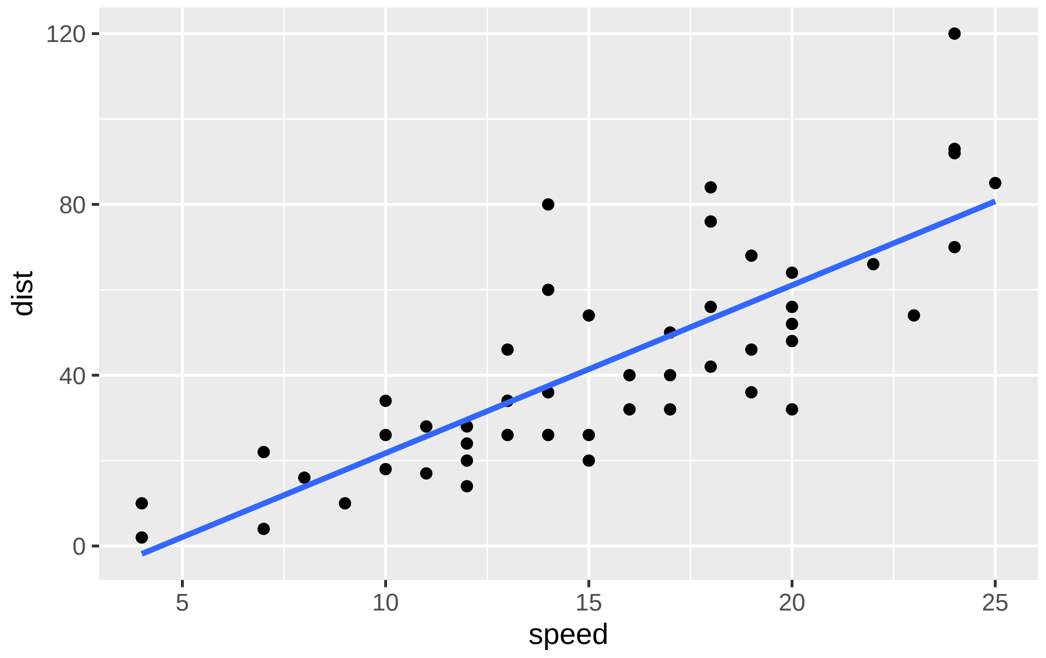
```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



using formula 'y ~ x' and se = FALSE we can remove the shaded region and make the line straight.

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

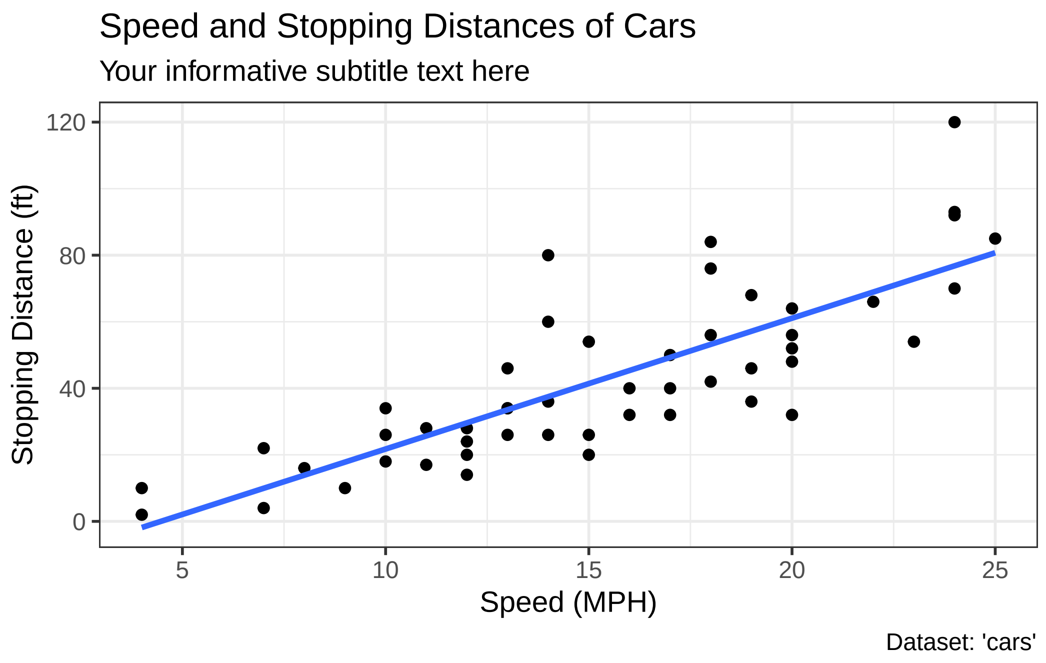
```
`geom_smooth()` using formula 'y ~ x'
```



Now finish this plot by adding various label annotations with the labs() function and changing the plot look to a more conservative "black & white" theme by adding the theme_bw() function.

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

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



Next, we will cover how to:

1. Adjust the point size of a scatter plot using the **size** parameter.
2. Change the point color of a scatter plot using the **color** parameter.
3. Set a parameter **alpha** to change the transparency of all points.

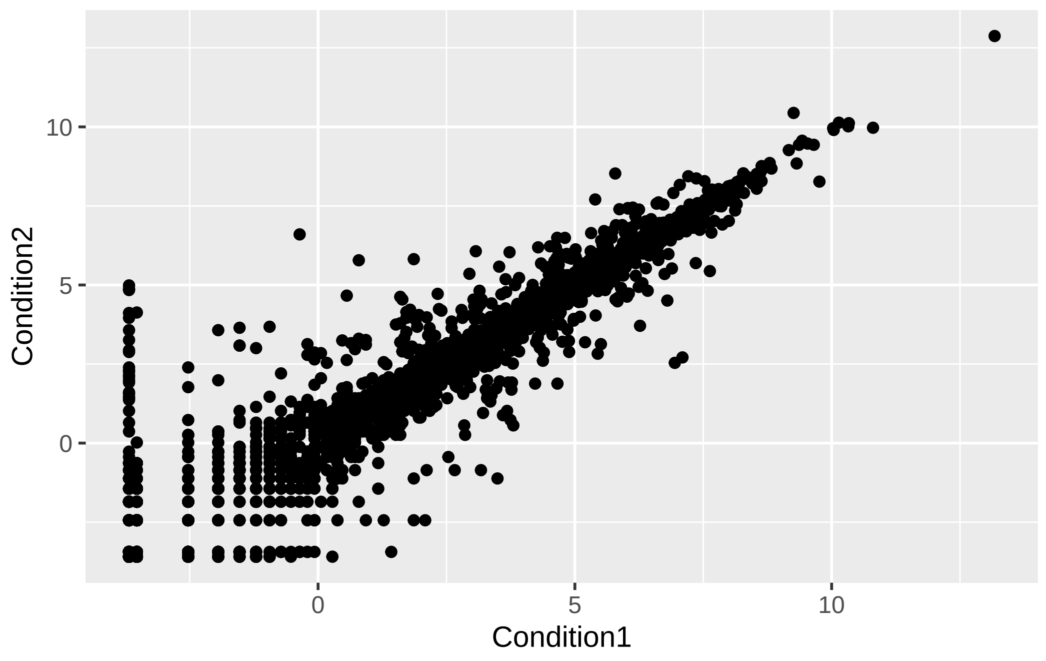
The code below reads the results of a differential expression analysis where a new anti-viral drug is being tested.

```
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
```

Check out the dataset.

```r
ggplot(data=genes) + aes(x=Condition1, y=Condition2) +
 geom_point()
```



Q. Use the nrow() function to find out how many genes are in this dataset. What is your answer?

```r
nrow(genes)
```

```
[1] 5196
```

There are 5196 genes in this dataset.

Q. Use the colnames() function and the ncol() function on the genes data frame to find out what the column names are (we will need these later) and how many columns there are. How many columns did you find?

```r
colnames(genes)
```

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

```r
ncol(genes)
```

```
[1] 4
```

There are 4 columns in this dataset and their names are, "Gene" "Condition1" "Condition2" "State".

Q.Use the table() function on the State column of this data.frame to find out how many 'up' regulated genes there are. What is your answer?

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

```

      down unchanging         up 
        72       4997        127 
```

There are 127 genes upregulated.

Q. Using your values above and 2 significant figures. What fraction of total genes is up-regulated in this dataset?

```r
 fraction <- 100*table(genes[,4])[3]/nrow(genes)
  signif(fraction, digits=3)
```

```
  up 
2.44 
```

The fraction of total genes is up-regulated in this dataset is 2.44.

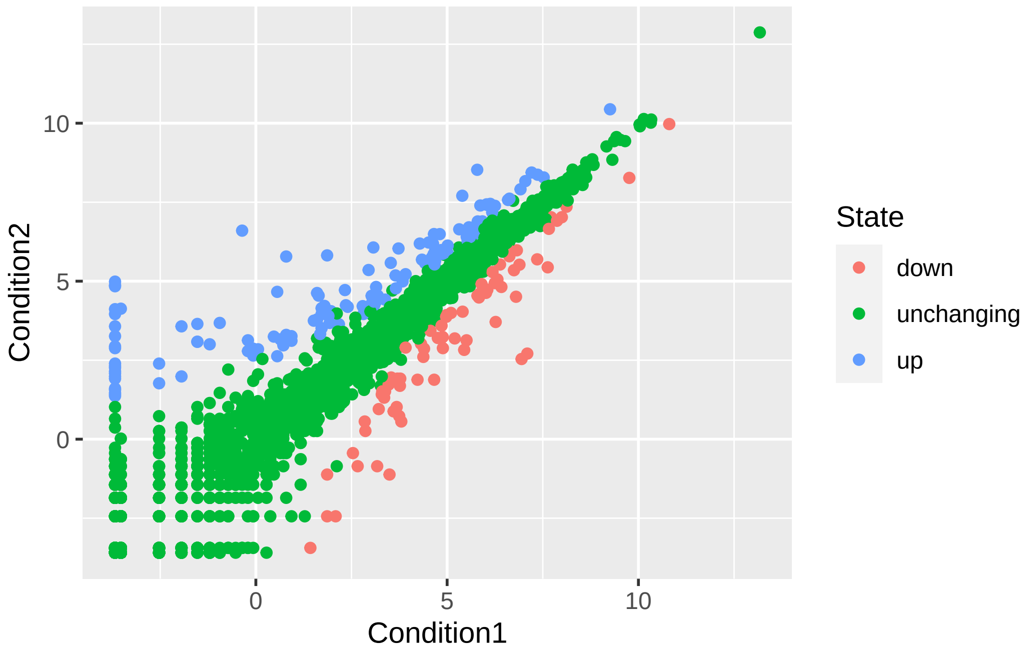
Could also have done: round(table(genes$State)/nrow(genes) * 100, 2) to view fraction of all states.

Now make a first basic scatter plot of this dataset.

1. Pass the genes data.frame as input to the ggplot() function.
2. Then use the aes() function to set the x and y aesthetic mappings to the Condition1 and Condition2 columns.
3. Finally add a geom_point() layer to add points to the plot. Don't forget to add layers step-wise with the + operator at the end of each line.

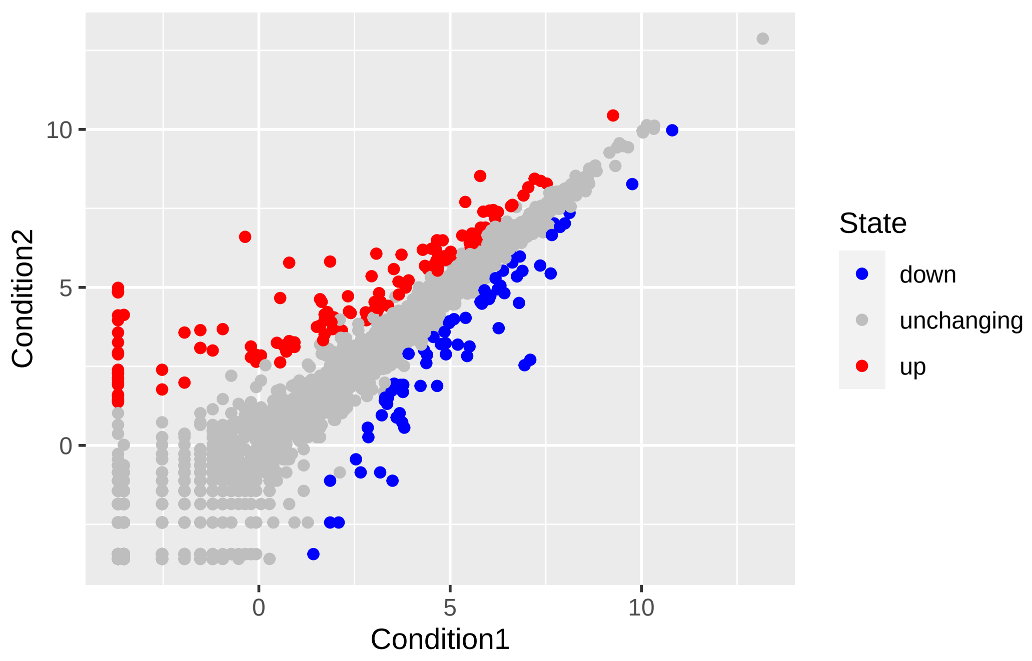
Color by the State of gene modification (up, down or unchanging). Assign the plot to object p.

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



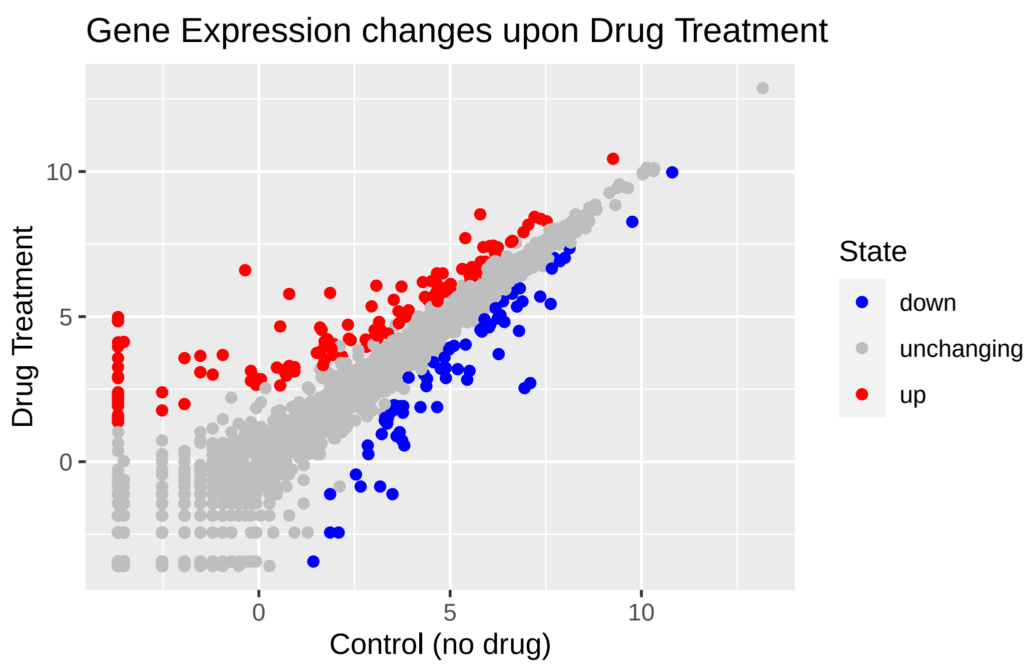
You can also change the colors of the categorical variable of state. Build onto p.

```
p2 <- p + scale_color_manual(values = c("blue", "gray", "red"))

p2
```



Now add labels to the plot to add context.This is done using the labs() function.

```
p2 +
   labs(title = "Gene Expression changes upon Drug Treatment",
        x = "Control (no drug)", y = "Drug Treatment")
```



Gap minder dataset Acquire the data first

```
# File location online
url <- "https://raw.githubusercontent.com/jennybc/gapminder/master/inst/extdata/gapminder.

gapminder <- read.delim(url)
```

Use dplyr to view dataset at a filtered level containing only the rows with a year value of 2007.

```
# install.packages("dplyr")  ## un-comment to install if needed
library(dplyr)
```

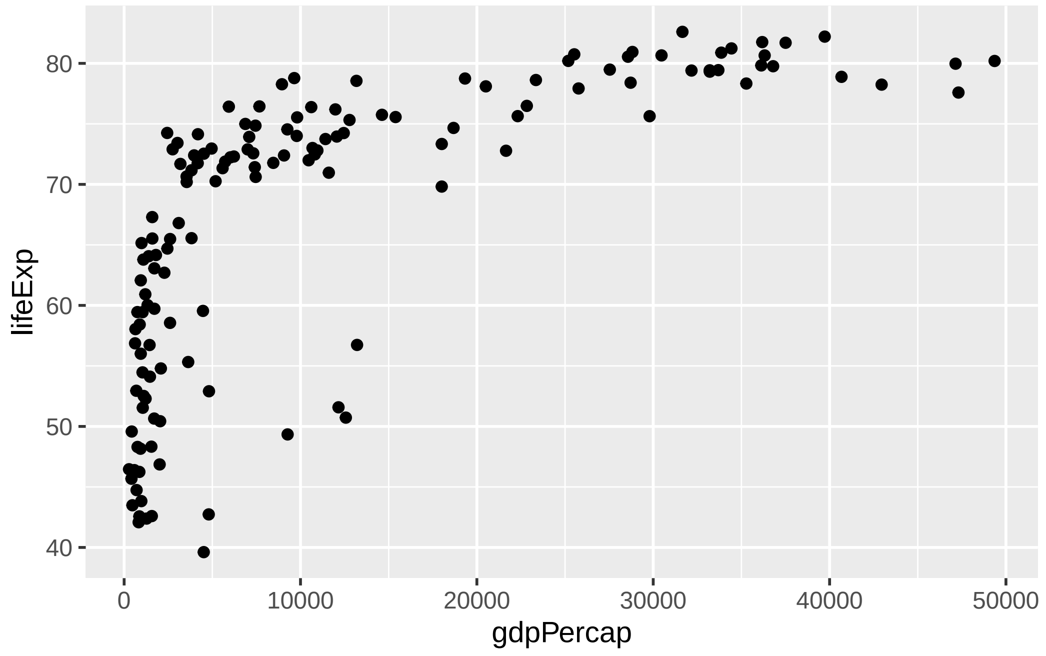
```

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)
```

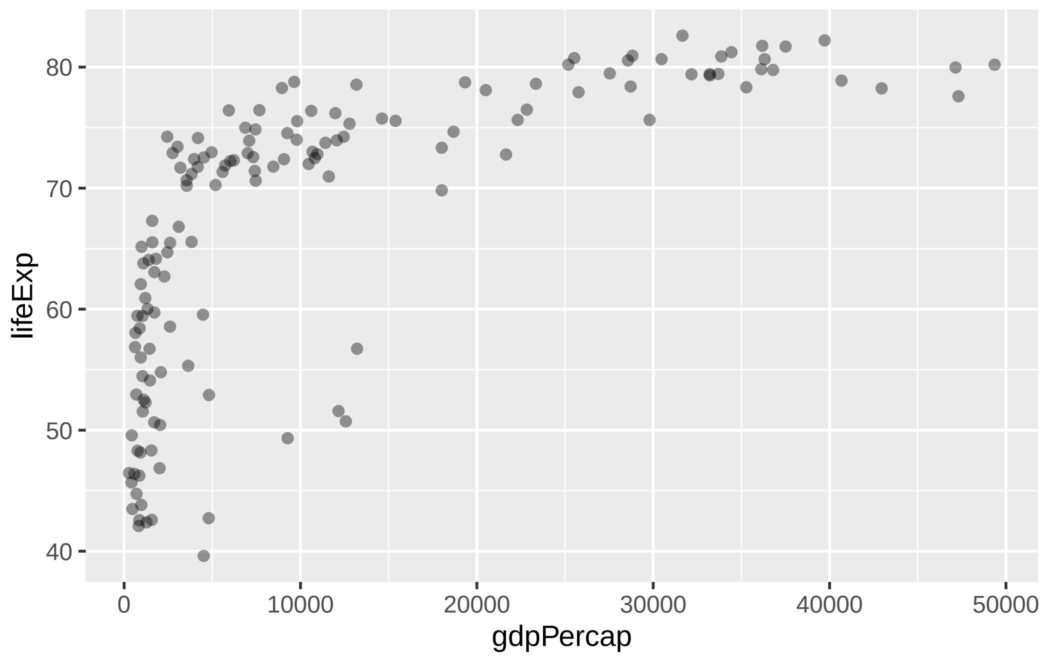
Make a scatter plot of this data with x axis being the GDP per capita and y axis is life expectancy.

```
ggplot(data=gapminder_2007) + aes(x=gdpPercap, y=lifeExp) +
  geom_point()
```



Use the alpha parameter to change the transparency of the points. This can help with seeing data points that overlap too.

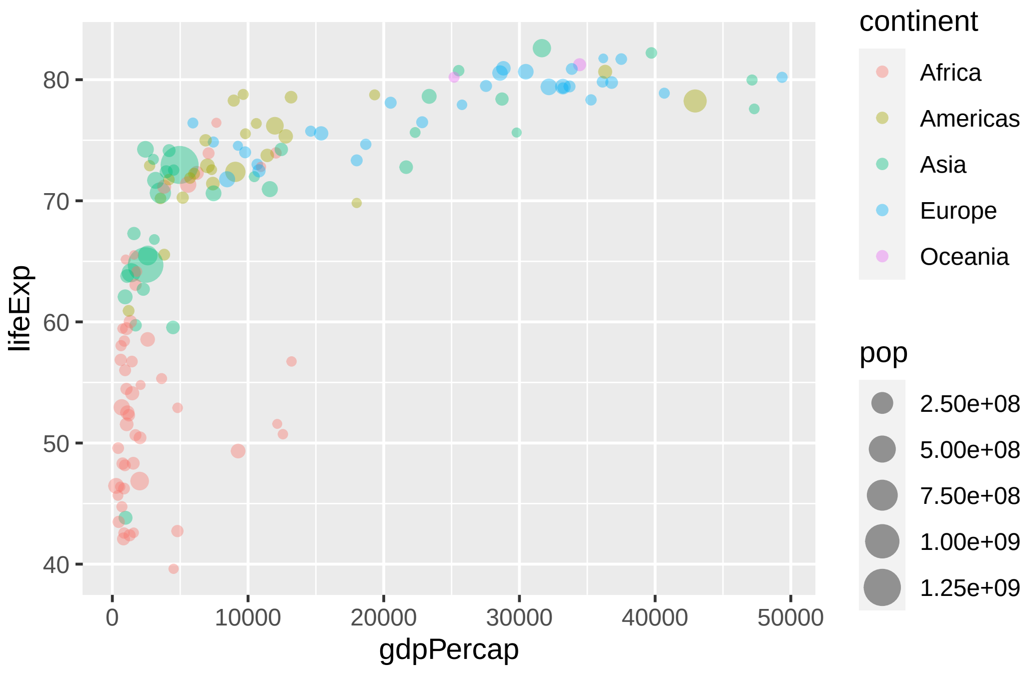
```
ggplot(data=gapminder_2007) + aes(x=gdpPercap, y=lifeExp) +
  geom_point(alpha=0.4)
```



Now add more varibales:

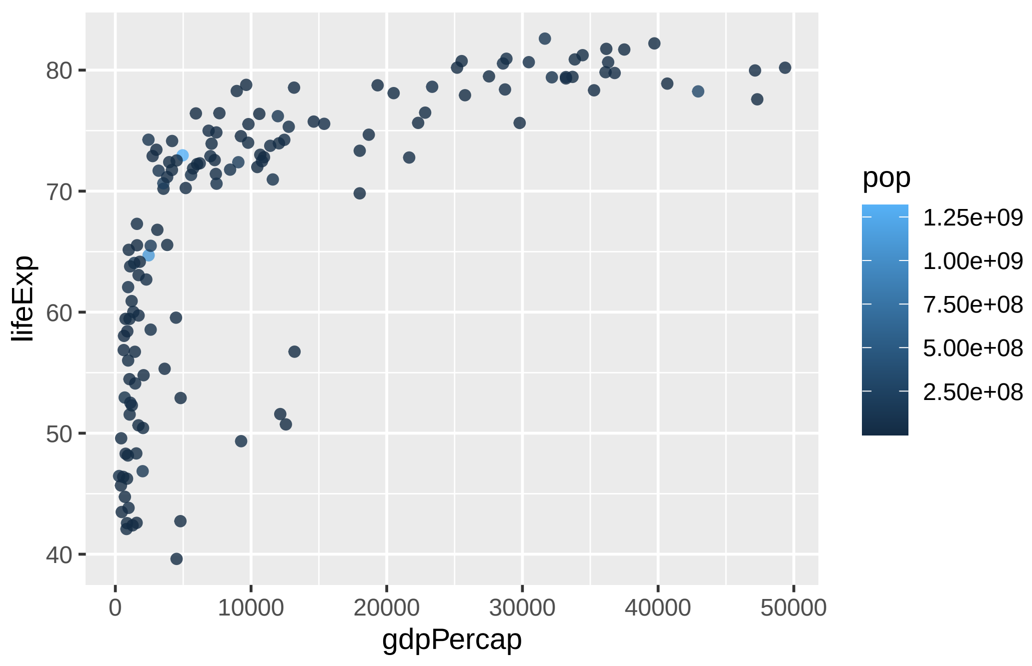
By mapping the continent variable to the point color aesthetic and the population pop (in millions) through the point size argument to aes() we can obtain a much richer plot that now includes 4 different variables from the data set.

```
ggplot(data=gapminder_2007) + aes(x=gdpPercap, y=lifeExp,
                                   color=continent, size=pop) +
geom_point(alpha=0.4)
```



By contrast, let's see how the plot looks like if we color the points by the numeric variable population pop:

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

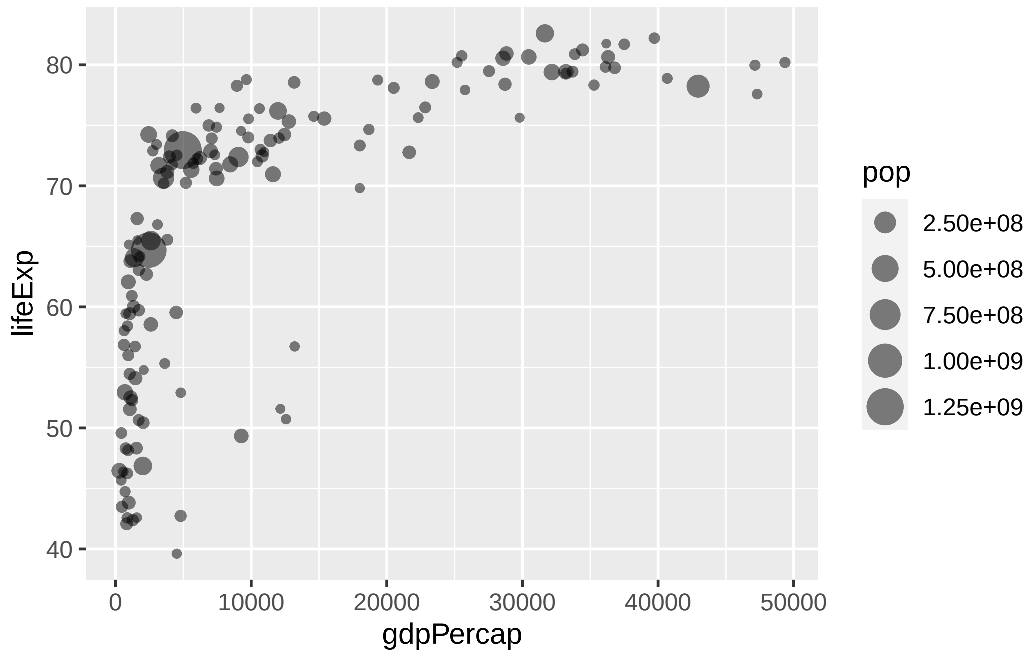


The scale changes to continuous as can be seen in the legend and the light-blue points are now the countries with the highest population number (China and India). This helps with seing gradient of magnitude of given variable value.

Next we can adjust the point size.

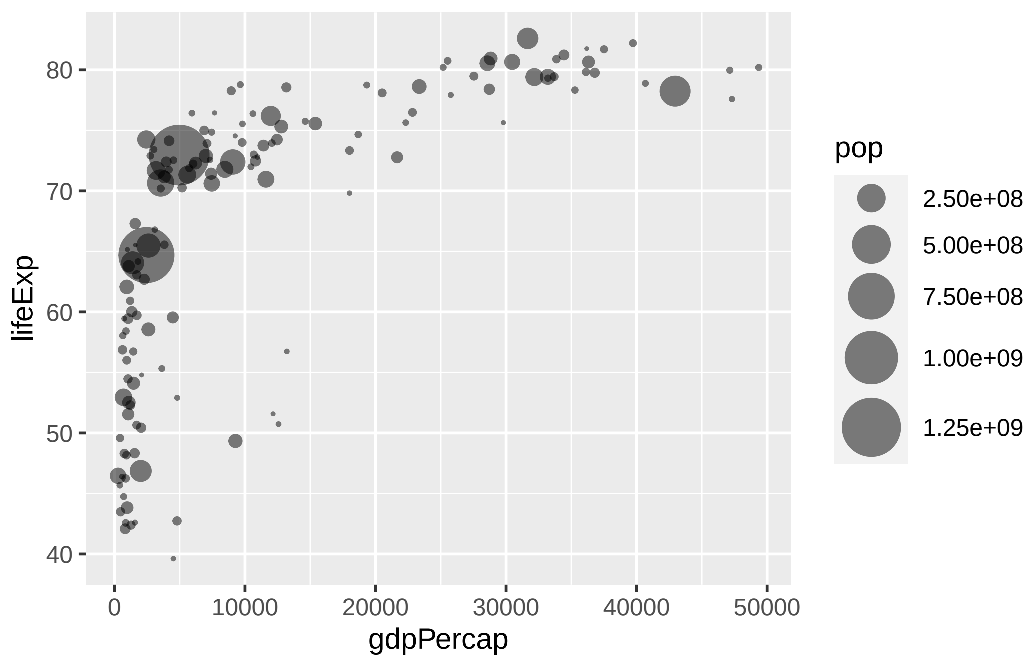
For the gapminder_2007 dataset we can plot the GDP per capita (x=gdpPercap) vs. the life expectancy (y=lifeExp) and set the point size based on the population (size=pop) of each country we can use:

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



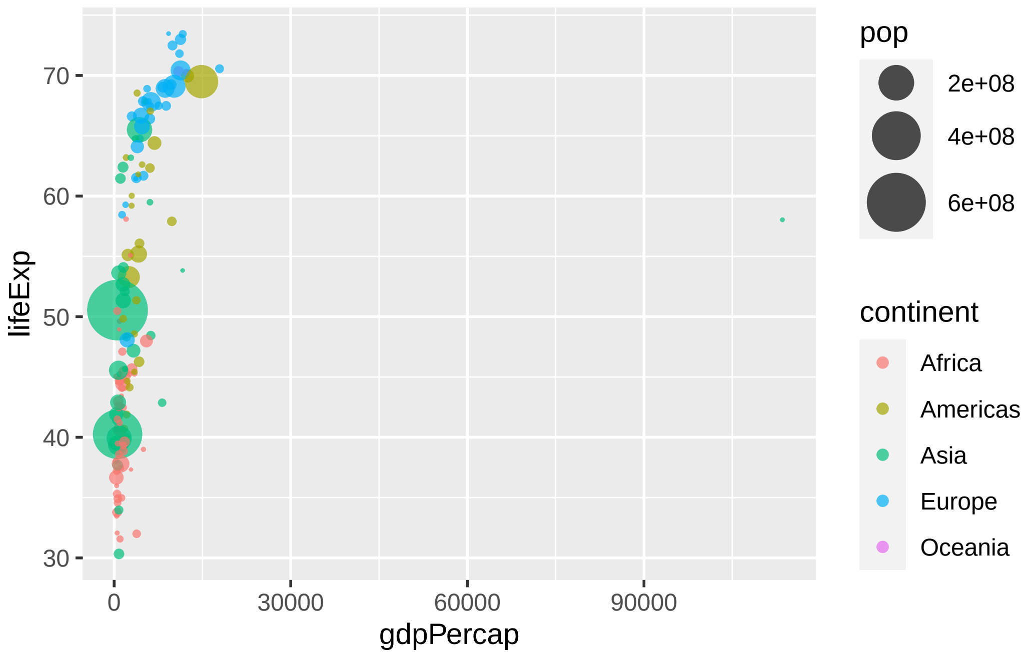
To reflect the actual population differences by the point size we can use the scale_size_area() function instead. The scaling information can be added like any other ggplot object with the + operator:

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



Q. Can you adapt the code you have learned thus far to reproduce our gapminder scatter plot for the year 1957? What do you notice about this plot is it easy to compare with the one for 2007?

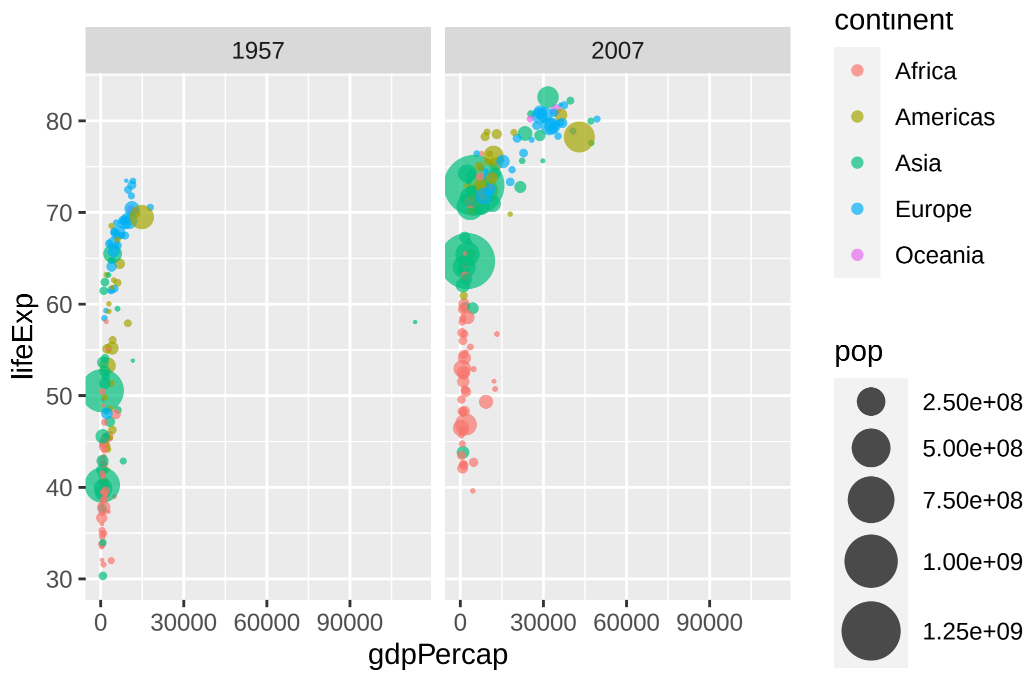
```r
gapminder_1957 <- gapminder %>% filter(year==1957)

ggplot(gapminder_1957) +
  aes(x = gdpPercap, y = lifeExp, color=continent,
                 size = pop) +
  geom_point(alpha=0.7) +
  scale_size_area(max_size = 10)
```



Use dplyr to filter the gapmider dataset to include only the year 1957 (check above for how we did this for 2007). Save your result as gapminder_1957. Use the ggplot() function and specify the gapminder_1957 dataset as input Add a geom_point() layer to the plot and create a scatter plot showing the GDP per capita gdpPercap on the x-axis and the life expectancy lifeExp on the y-axis Use the color aesthetic to indicate each continent by a different color Use the size aesthetic to adjust the point size by the population pop Use scale_size_area() so that the point sizes reflect the actual population differences and set the max_size of each point to 15 -Set the opacity/transparency of each point to 70% using the alpha=0.7 parameter

Q. Do the same steps above but include 1957 and 2007 in your input dataset for ggplot(). You should now include the layer facet_wrap(~year) to produce the following plot:

```
gapminder_1957 <- gapminder %>% filter(year==1957 | year==2007)

ggplot(gapminder_1957) +
  geom_point(aes(x = gdpPercap, y = lifeExp, color=continent,
                 size = pop), alpha=0.7) +
  scale_size_area(max_size = 10) +
  facet_wrap(~year)
```



Bar Charts Bar charts visualize numeric values grouped by categories. Each category is represented by one bar with a height defined by each numeric value.

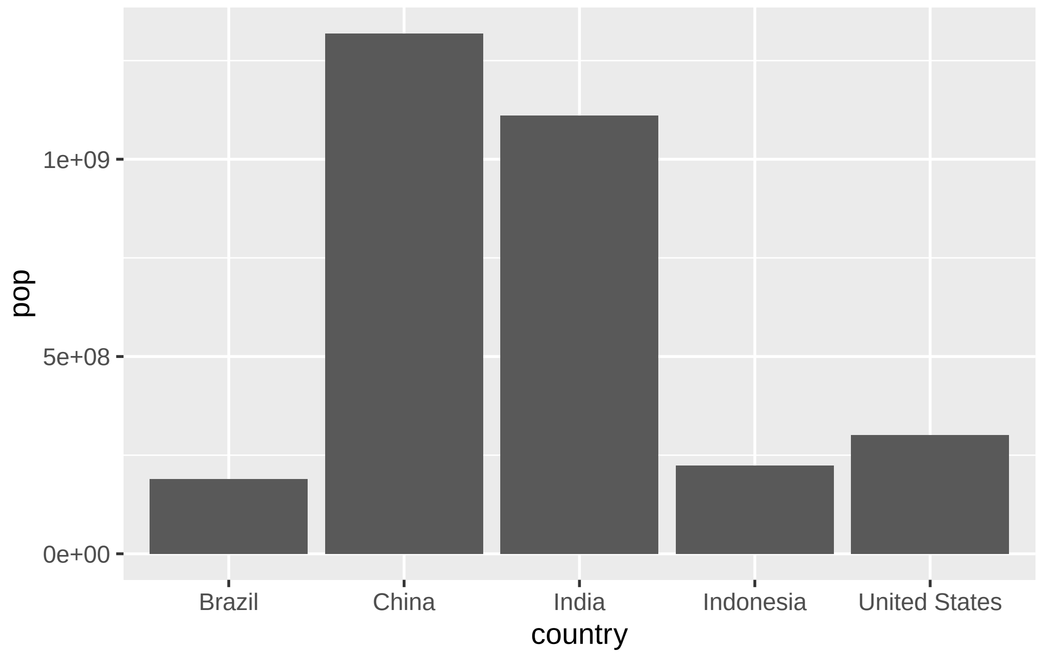
Below you can find an example showing the number of people (in millions) in the five biggest countries by population in 2007:

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

gapminder_top5
```

```
        country continent year lifeExp        pop gdpPercap
1         China      Asia 2007  72.961 1318683096  4959.115
2         India      Asia 2007  64.698 1110396331  2452.210
3 United States  Americas 2007  78.242  301139947 42951.653
4     Indonesia      Asia 2007  70.650  223547000  3540.652
5        Brazil  Americas 2007  72.390  190010647  9065.801
```

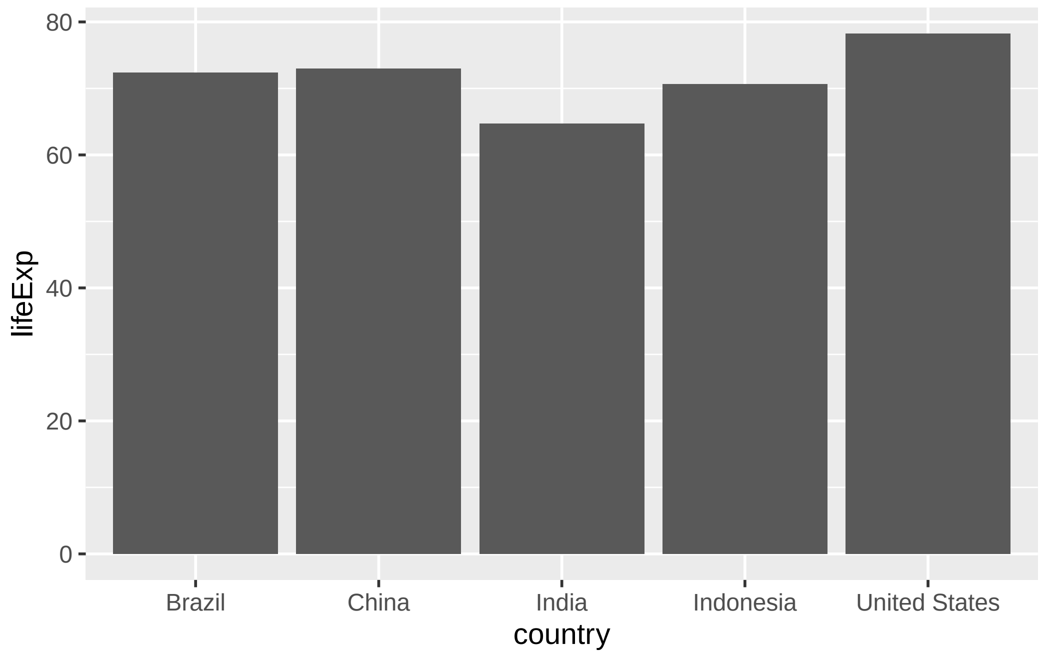
In ggplot2, bar charts are created using the geom_col() geometric layer. The geom_col() layer requires the x aesthetic mapping which defines the different bars to be plotted. The height of each bar is defined by the variable specified in the y aesthetic mapping. Both mappings, x and y are required for geom_col(). Let's create our first bar chart with the gapminder_top5 dataset. It contains population (in millions) and life expectancy data for the biggest countries by population in 2007.

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



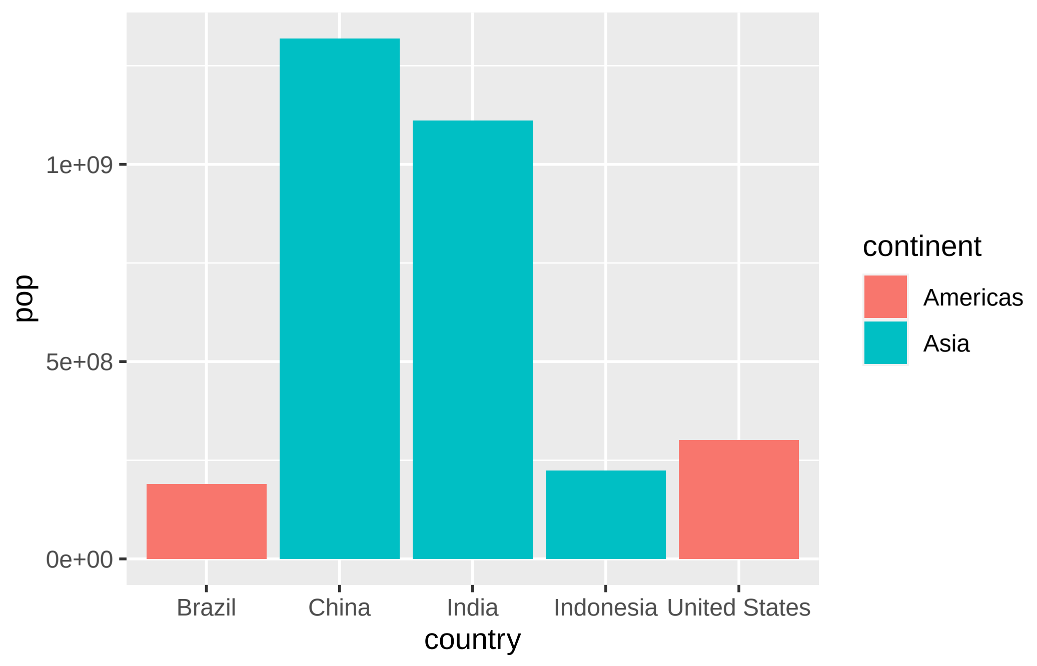
Q Create a bar chart showing the life expectancy of the five biggest countries by population in 2007.

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



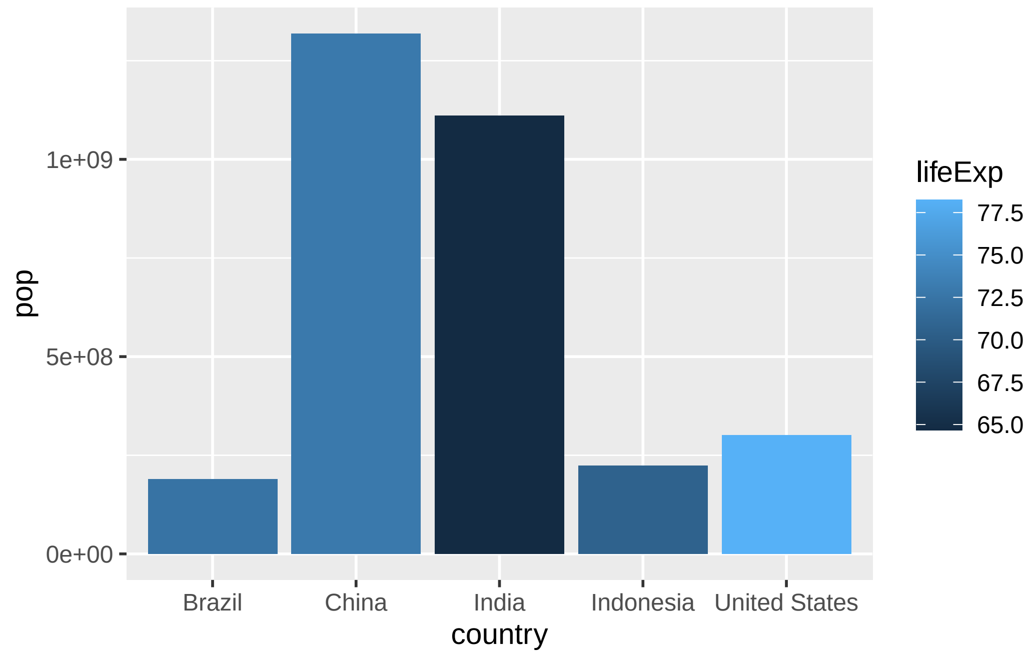
Now fill the bars with colors using continent variable as meaning.

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



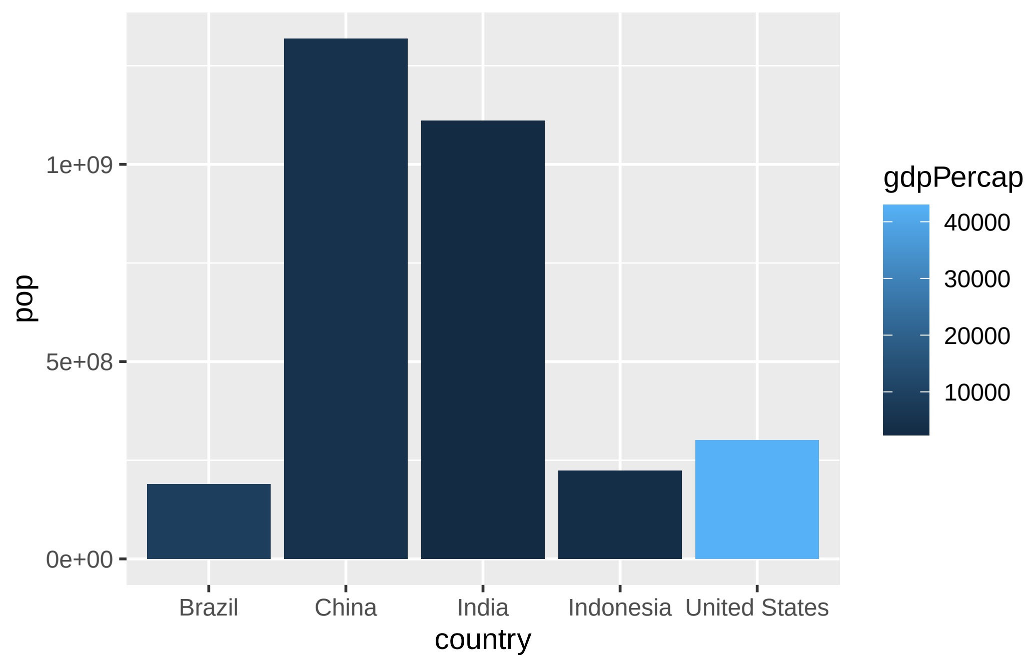
Or use the life expectancy variable as a numeric variable.

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



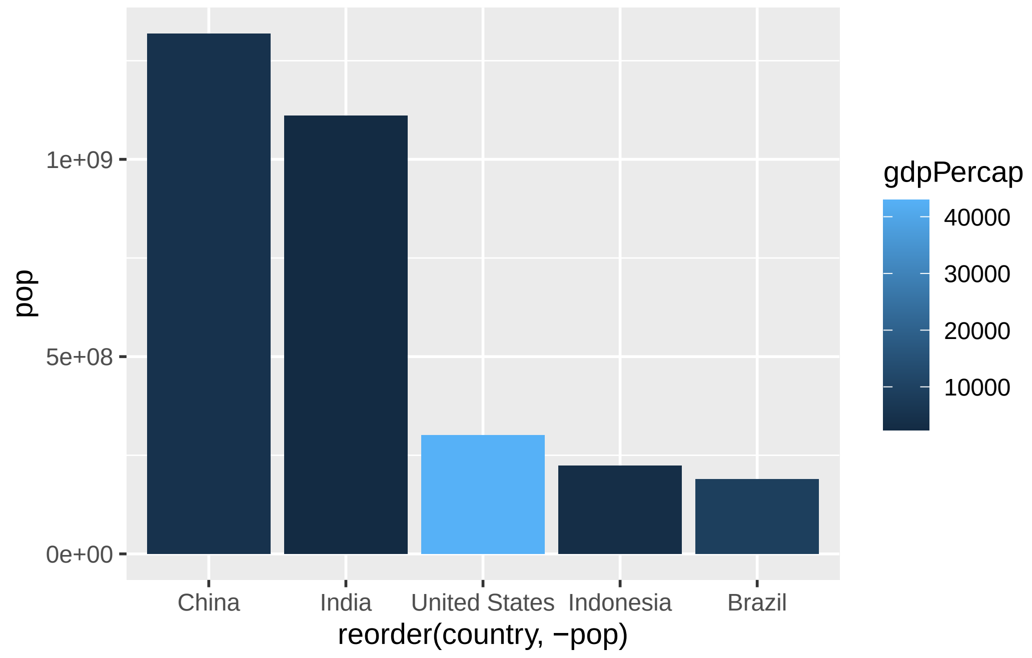
Q. Plot population size by country. Create a bar chart showing the population (in millions) of the five biggest countries by population in 2007.

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



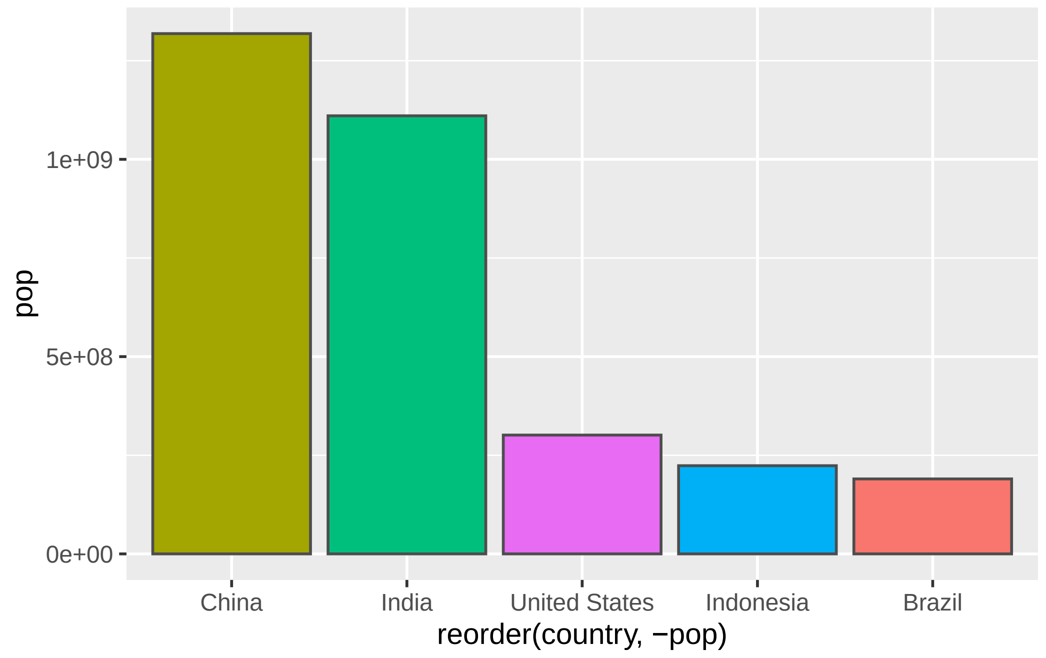
Now chnage the order of the bars

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



Now fill by country

```
ggplot(gapminder_top5) +
  aes(x=reorder(country, -pop), y=pop, fill=country) +
  geom_col(col="gray30") +
  guides(fill="none")
```

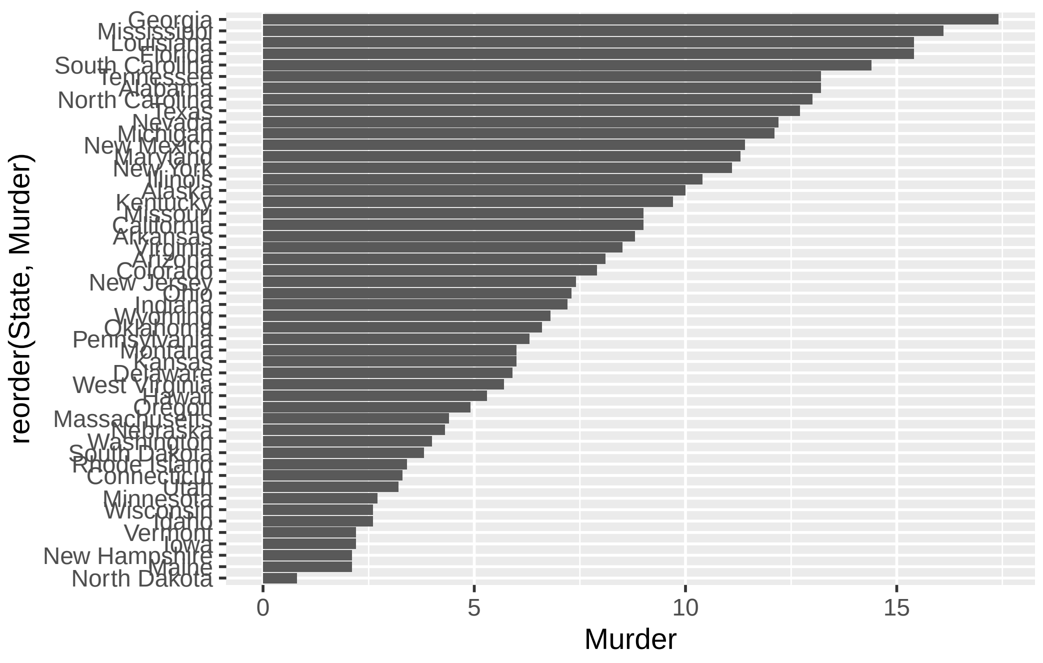


Sometimes it might be useful to rotate (or “flip”) your plots to enable a more clear visualization. For this we can use the coord_flip() function. Lets look at an example considering arrest data in US states. This is another inbult dataset called USArrests.

```
head(USArrests)
```

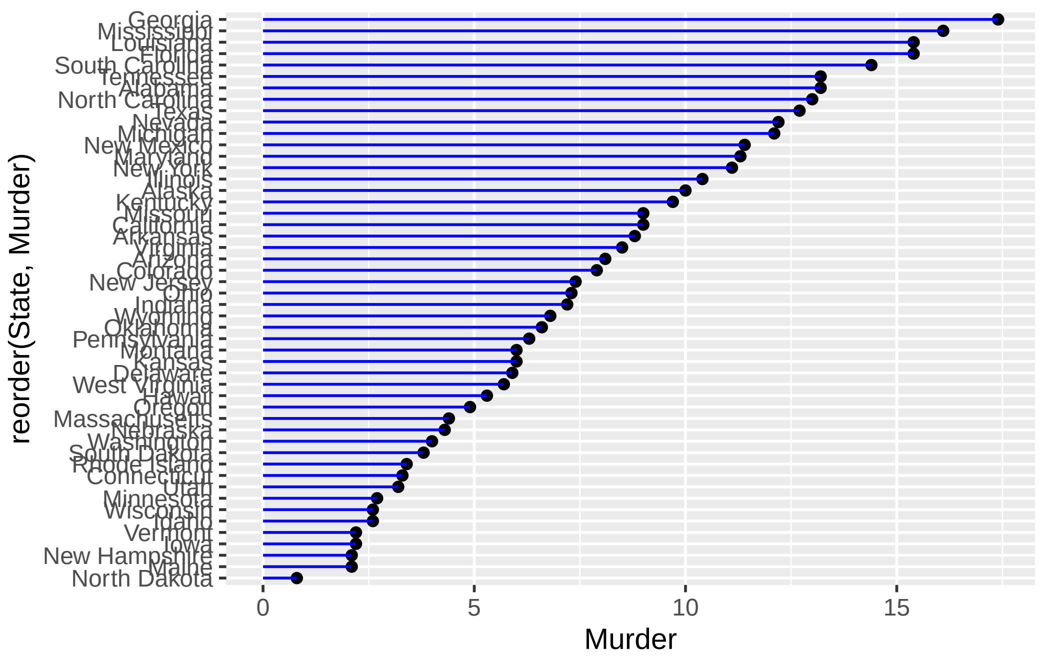
```
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

```
USArrests$State <- rownames(USArrests)
ggplot(USArrests) +
  aes(x=reorder(State,Murder), y=Murder) +
  geom_col() +
  coord_flip()
```



Combine two different plotting methods by combining geom_point() and geom_segment().

```
ggplot(USArrests) +
  aes(x=reorder(State,Murder), y=Murder) +
  geom_point() +
  geom_segment(aes(x=State,
                  xend=State,
                  y=0,
                  yend=Murder), color="blue") +
  coord_flip()
```


Georgia
Mississippi
Louisiana
Florida
South Carolina
Tennessee
Alabama
North Carolina
Texas
Nevada
Michigan
New Mexico
Maryland
New York
Illinois
Alaska
Kentucky
Missouri
California
Arkansas
Virginia
Arizona
Colorado
New Jersey
Ohio
Indiana
Wyoming
Oklahoma
Pennsylvania
Montana
Kansas
Delaware
West Virginia
Hawaii
Oregon
Massachusetts
Nebraska
Washington
South Dakota
Rhode Island
Connecticut
Utah
Minnesota
Wisconsin
Idaho
Vermont
Iowa
New Hampshire
Maine
North Dakota
reorder(State, Murder)
0
5
10
15
Murder