Class_05___012424

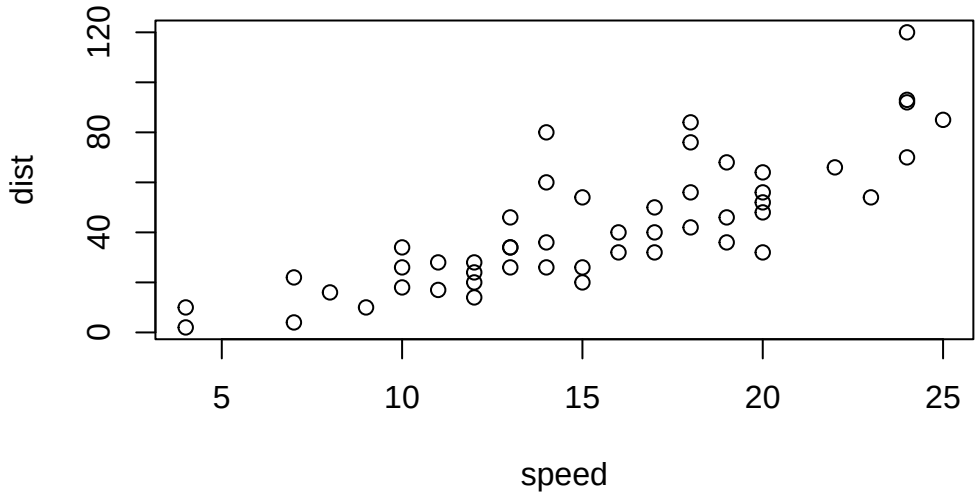
Johann Tailor

2024-01-24

Graphics System in R

There are many so called graphics system for R. Some of these include: **"base-R"** and other add-ons like **"ggplot2"**

```
plot(cars)
```



How can we make this in the `ggplot2` It doesn't work as soon as you run it. It needs to be installed.

To install any package,

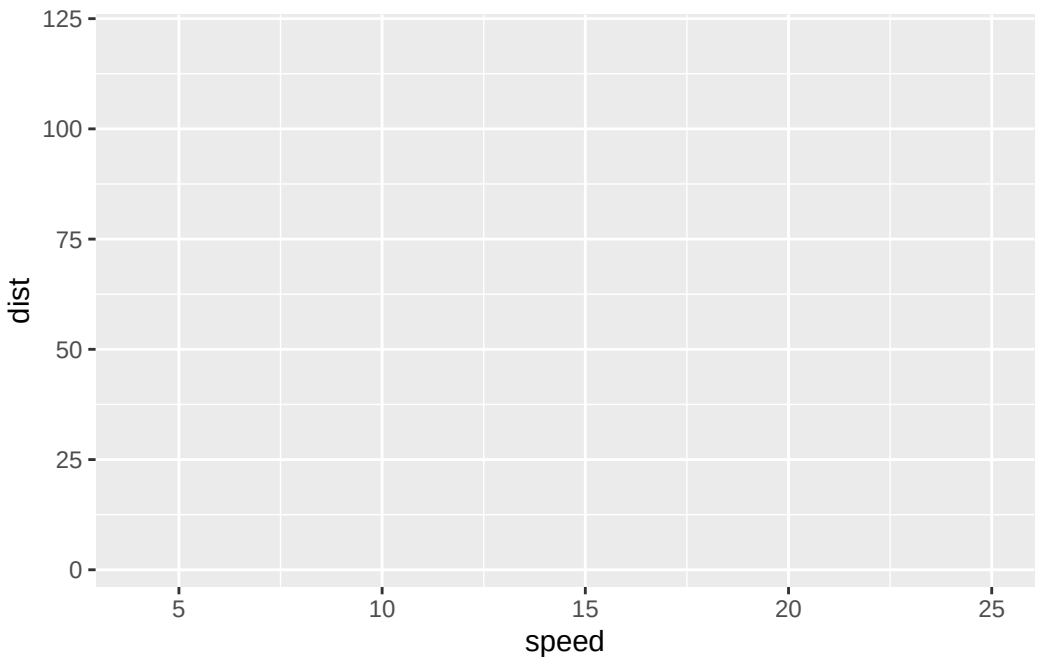
- we use the `install.packages()` function
- we load the package from our library of installed packages: `library(ggplot2)`

```
library(ggplot2)
```

ggplot2 is a complex functions and requires to following arguments

- data (data.frame)
- aesthetic (how the data map to the plot)
- geoms (type of plots: points, lines, etc)

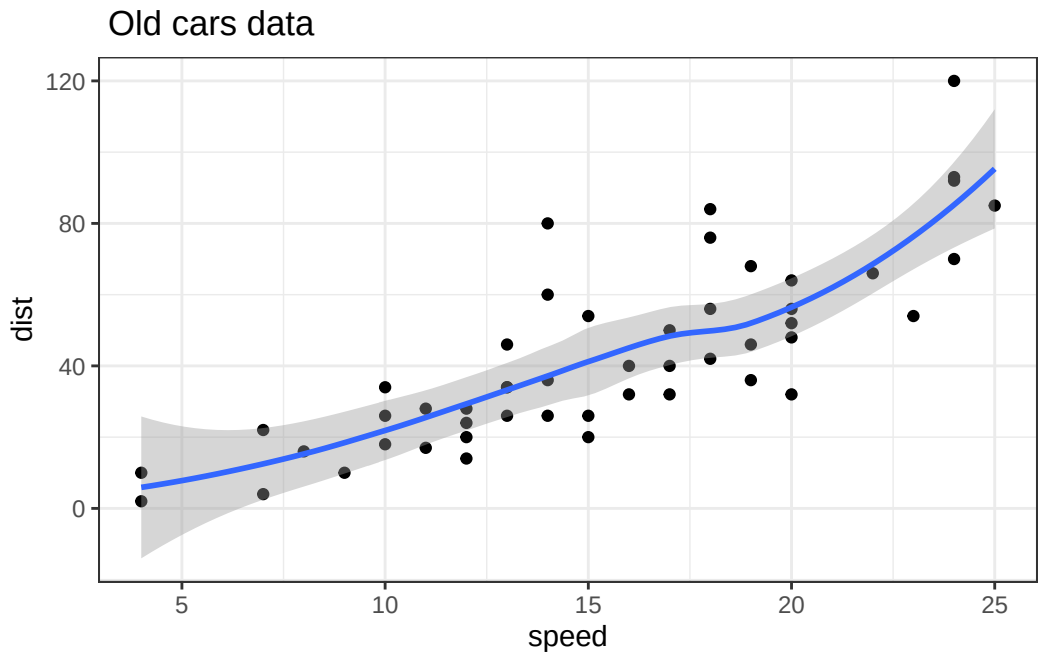
```
ggplot(cars) +
  aes(speed, dist)
```



Here ggplot was more work/typing-than base R. However, I can add more layers like this:

```
ggplot(cars) +
  aes(speed, dist) +
  geom_point() +
  geom_smooth() +
  labs(title = " Old cars data") +
  theme_bw()
```

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

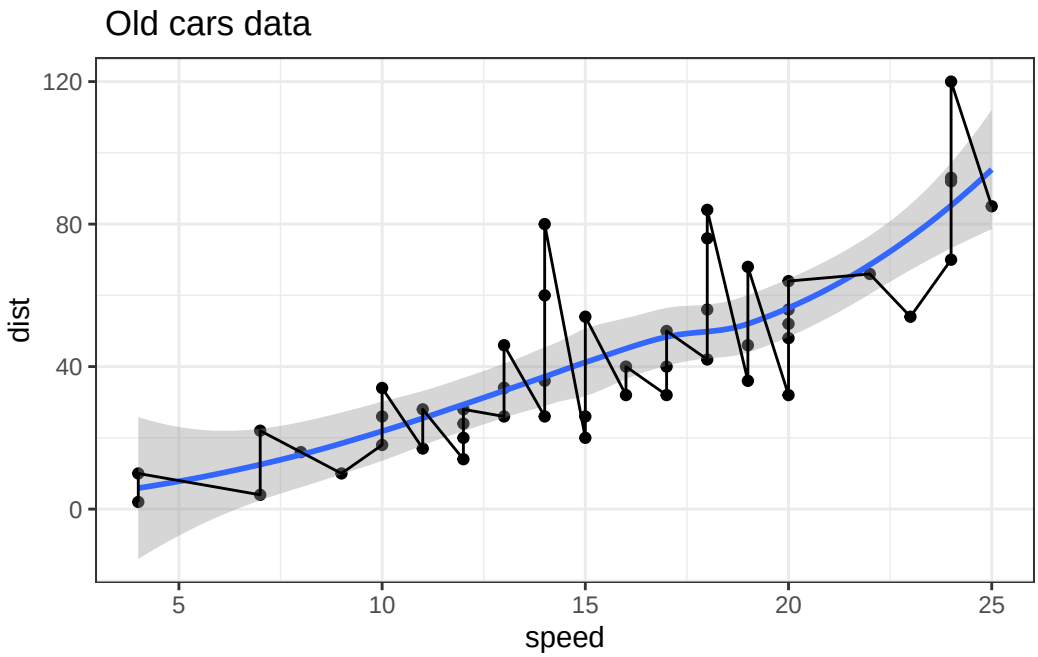


To make life easier, you can assign the whole plot to a variable and add layers to the variable such as:

```
pl <- ggplot(cars) +
  aes(speed, dist) +
  geom_point() +
  geom_smooth() +
  labs(title = " Old cars data") +
  theme_bw()
```

```
pl + geom_path()
```

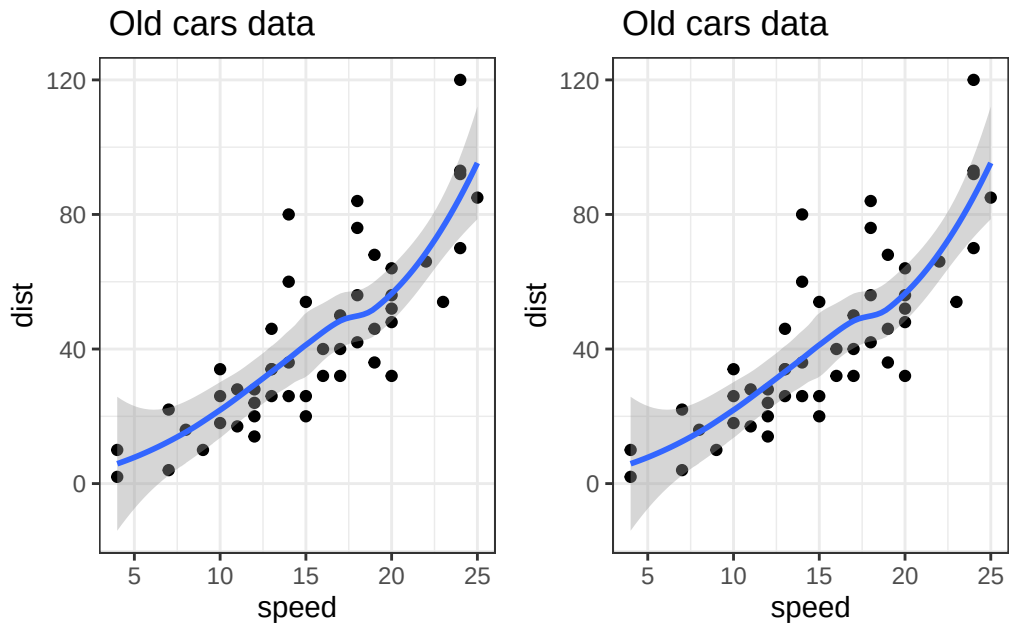
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



To arrange data for publications, you can use `library("patchwork")`

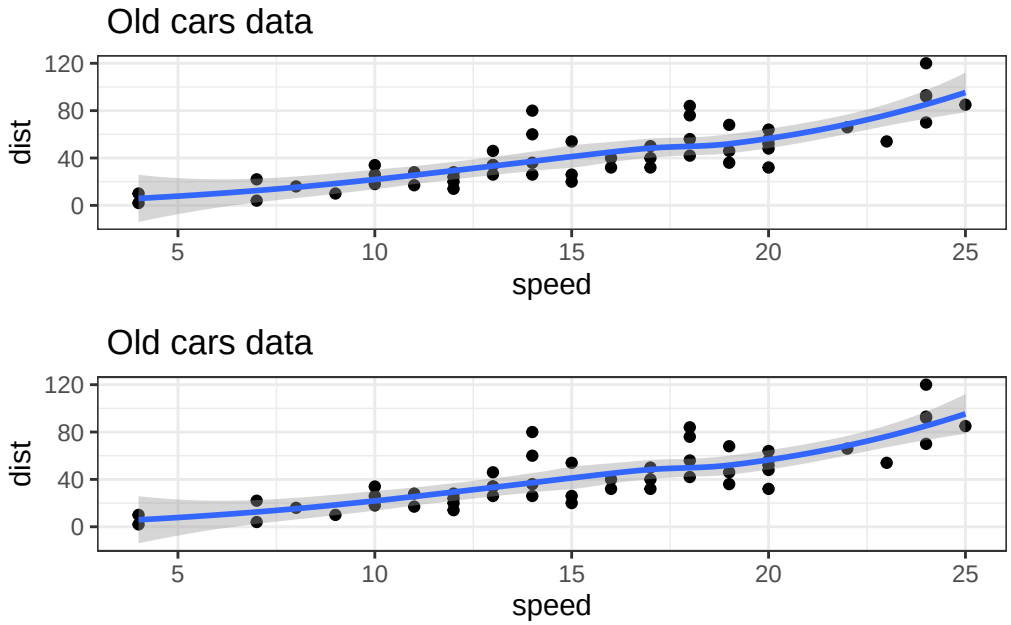
```
library("patchwork")
pl | pl
```

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




```
(pl/pl)
```

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



Lab sheet section 6

```
url <- "https://bioboot.github.io/bimm143_S20/class-material/up_down_expression.txt"
genes <- read.delim(url)
head(genes, 20)
```

```
         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
7       ABCA7  3.4484220  3.8266509 unchanging
8   ABCA9-AS1 -3.6808610 -3.5921390 unchanging
9      ABCC11 -3.5288580 -1.8551732 unchanging
10      ABCC3  0.9305738  3.2603040         up
11      ABCC5  4.6004252  5.4994435         up
12  ABCC5-AS1 -3.6808610 -3.4401355 unchanging
13    ABCC6P1 -0.7215031 -0.2702107 unchanging
14      ABCD1  2.6805956  3.3800430 unchanging
```

```
15     ABHD11  4.4136560  3.9521816 unchanging
16     ABI3BP -1.2069298 -3.5921390 unchanging
17       ABL1  6.3583620  6.0814650 unchanging
18     ABLIM2 -1.9438953 -1.1182077 unchanging
19        ABO -3.6808610 -3.5921390 unchanging
20       ABP1 -3.6808610 -3.5921390 unchanging
```

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

```
nrow(genes)
```

```
[1] 5196
```

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?

```
ncol(genes)
```

```
[1] 4
```

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?

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

```
      down unchanging         up
        72       4997        127
```

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

```
table(genes$State) / nrow(genes)*100
```

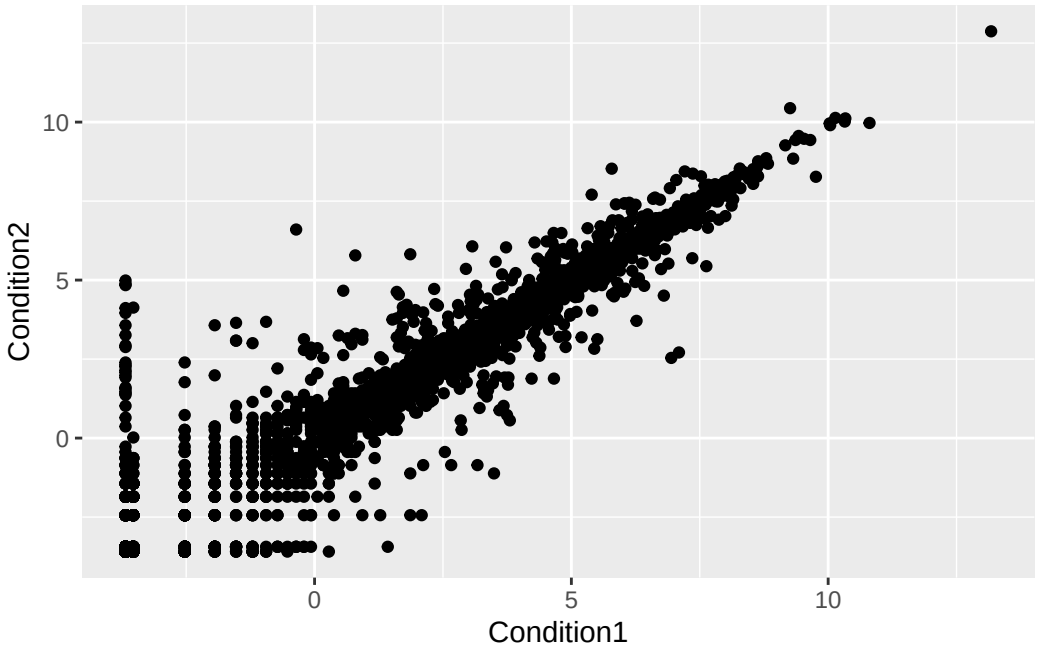
```
      down unchanging         up
  1.385681  96.170131   2.444188
```

```
round(table(genes$State) / nrow(genes)*100, 3)
```

```
    down unchanging          up
   1.386     96.170      2.444
```

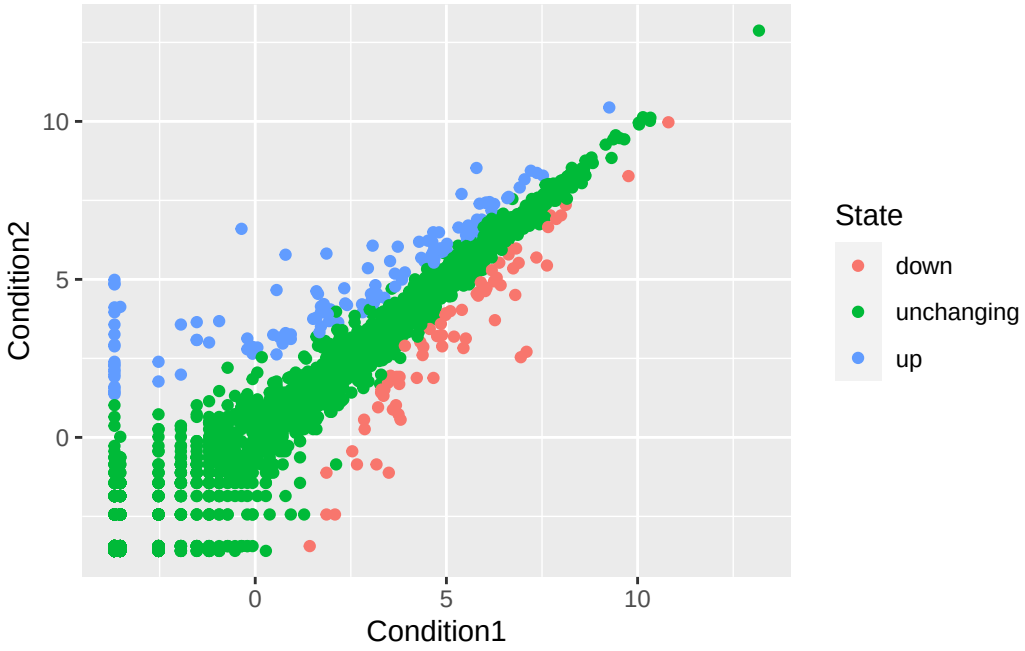
Make a graph of the above data:

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



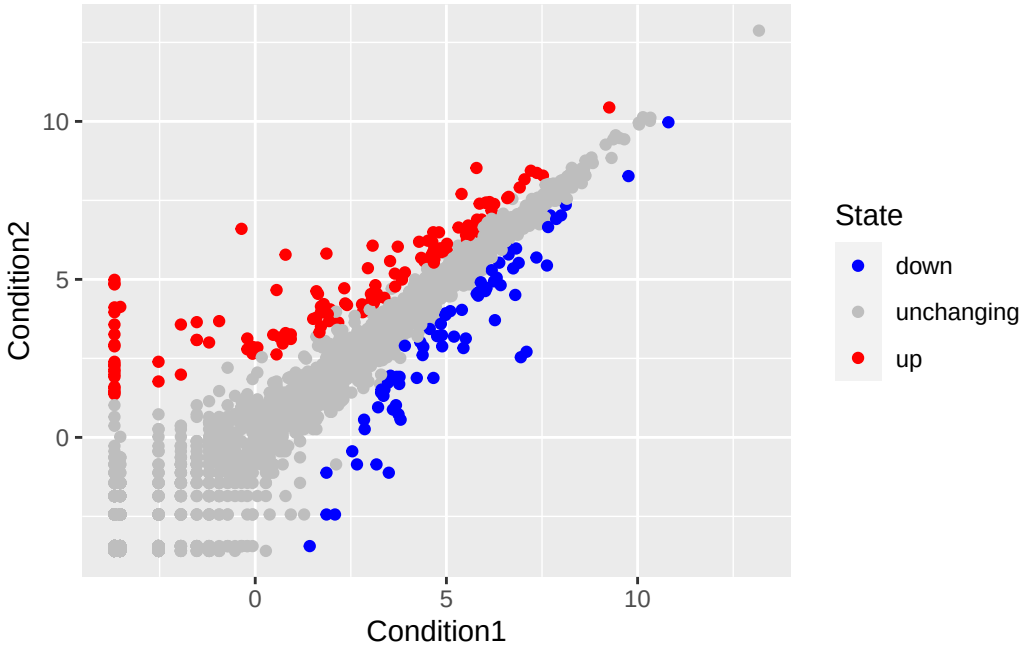
Lets add colors:

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



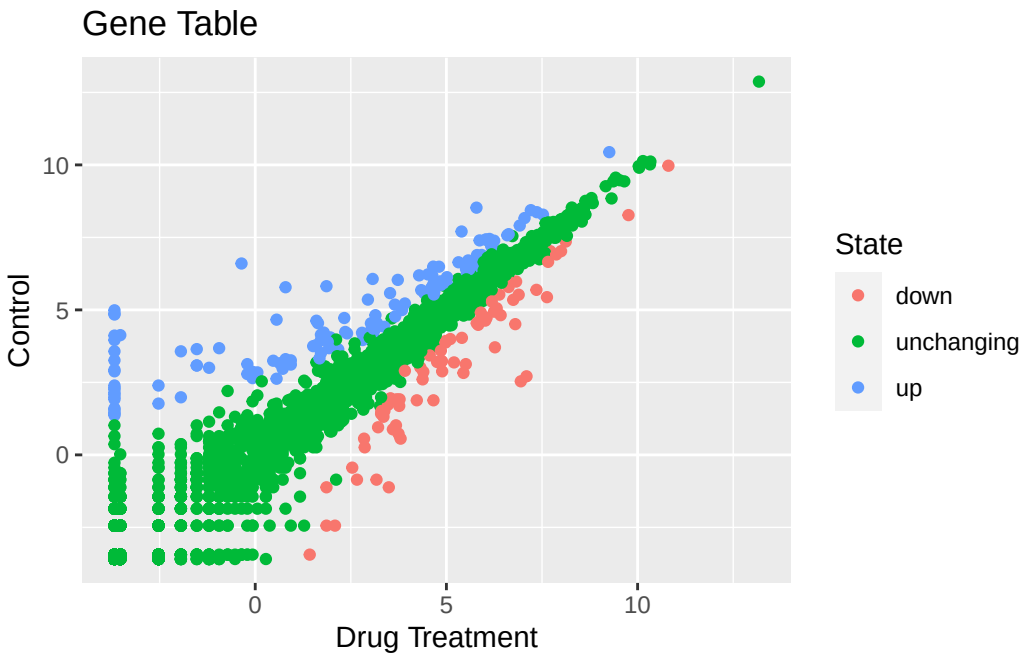
Lets change it up:

```
p <- ggplot(genes) +
    aes(x=Condition1, y=Condition2, col=State) +
    geom_point()
p + scale_colour_manual( values=c("blue","gray","red") )
```



Adding labels to the axis:

```
ggplot(genes) +
    aes(x=Condition1, y=Condition2, col=State, name=Gene) +
    geom_point() +
    labs(title = "Gene Table", x= "Drug Treatment", y="Control")
```




```
Interactive_plot_genes <- ggplot(genes) +
    aes(x=Condition1, y=Condition2, col=State, name=Gene) +
    geom_point() +
    labs(title = "Gene Table", x= "Drug Treatment", y="Control")
```

Make it interactive!

NOTE: The plotly feature will not work for pdf format.

```
library("plotly")
##ggplotly(Interactive_plot_genes)
```

How to remove the library verbage:

```
library("plotly")
```

Section 7: Going Further with Gapminder feature

Making different types of plots:

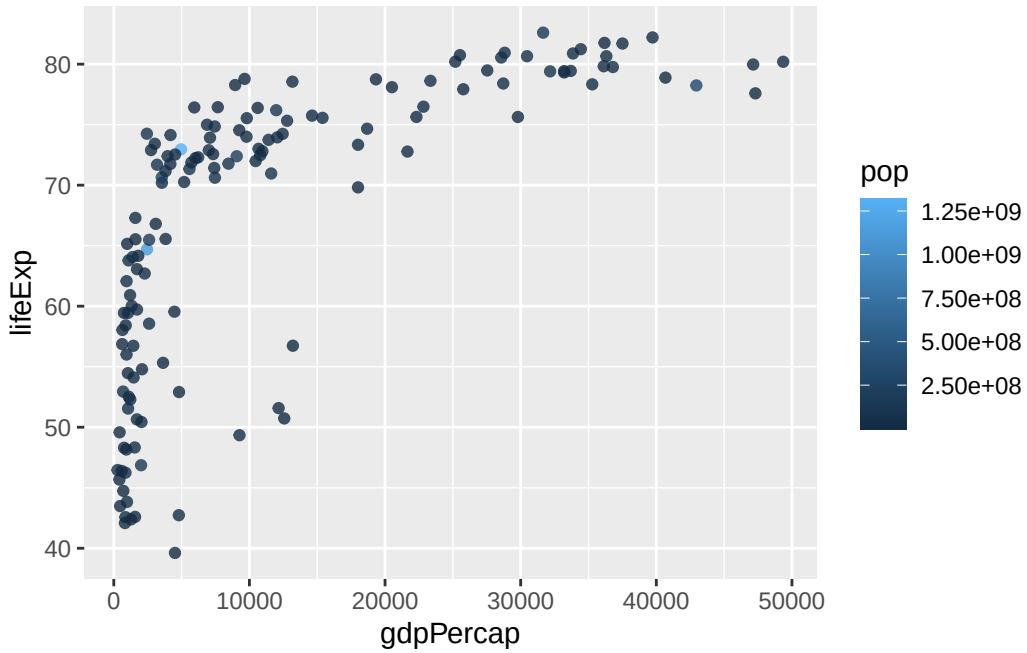
1 - A Scatter Plot

```
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
```

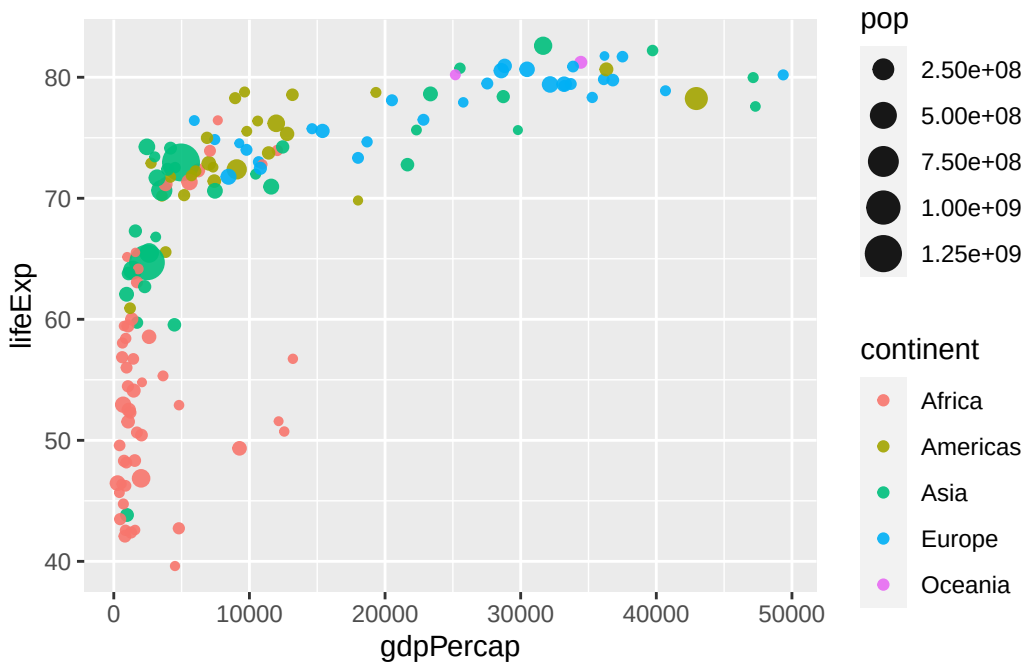
```
gapminder_2007 <- gapminder %>% filter(year==2007)
```

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



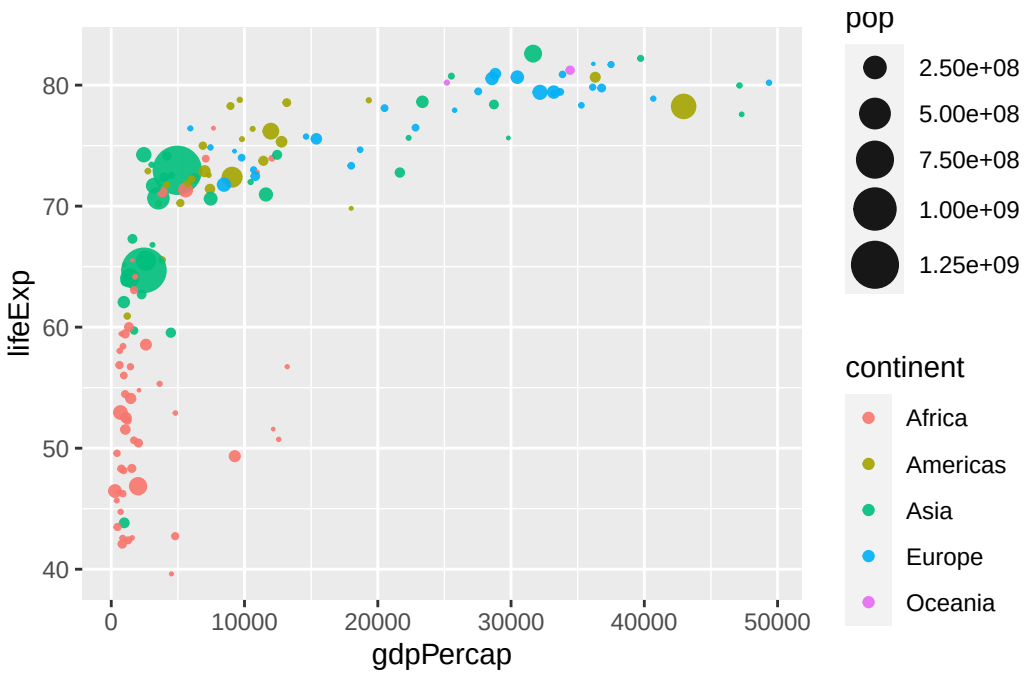
Lets add colors to the above graph:

```
ggplot(gapminder_2007) +
  aes(x=gdpPercap, y=lifeExp, color=continent, size=pop) +
  geom_point(alpha=0.9)
```



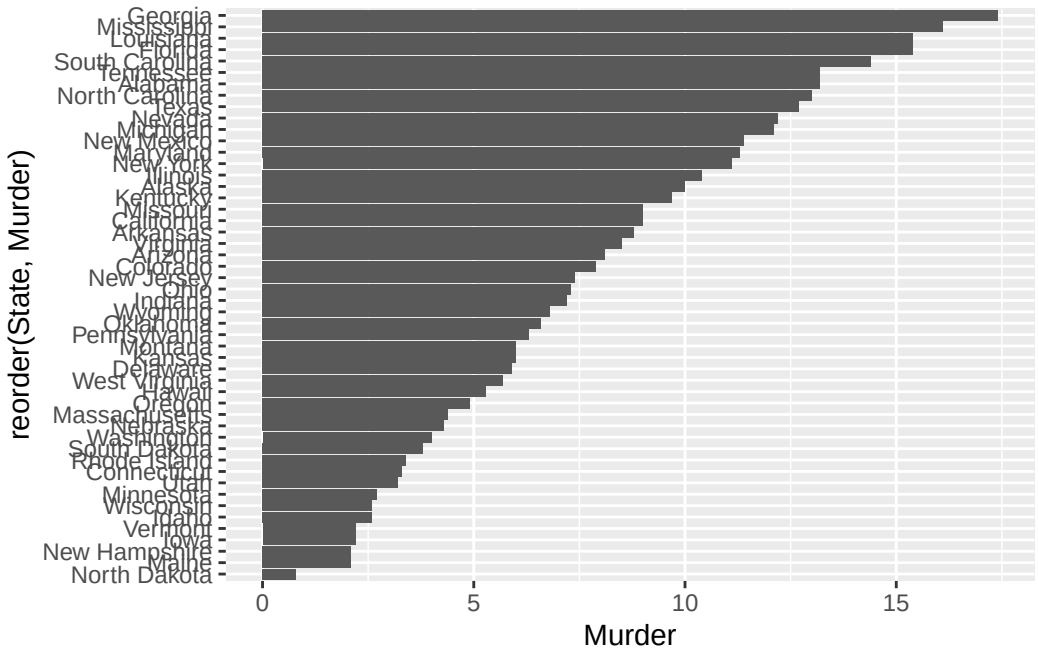
Changing the size of the points:

```
ggplot(gapminder_2007) +
  aes(x=gdpPercap, y=lifeExp, color=continent, size=pop) +
  geom_point(alpha=0.9)+
  scale_size_area(max_size = 8)
```



2 - Using a bar graph to plot USAarrest data:

```
#way to add a new named column
USArrests$State <- rownames(USArrests)

#Now plot:
ggplot(USArrests) +
  aes(x=reorder(State,Murder), y=Murder) +
  geom_col() +
  coord_flip()
```




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

Now, lets make this graph interactive using `ggplotly`

```
#ggplotly(US_arrest)
```

##Section 8: Lets animate!

Let's use gganimate to make cool graphs:

```
library(gapminder)
library(gganimate)

# Setup nice regular ggplot of the gapminder data
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
```

```
    # Facet by continent
    facet_wrap(~continent) +
    # Here comes the gganimate specific bits
    labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'life expectancy') +
    transition_time(year) +
    shadow_wake(wake_length = 0.1, alpha = FALSE)
```

##Section 10: Combining all plots

```
library("patchwork")
(pl/pl)
```

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