Class 05: Data Visualization with GGPLOT

Le, Sarah (PID: A18518276)

Today we are exploring the **ggplot** package and how to make nice figures in R.

There are lots of ways to make figures and plot in R. These include:

- so called "base" R
- and add on packages like **ggplot2**

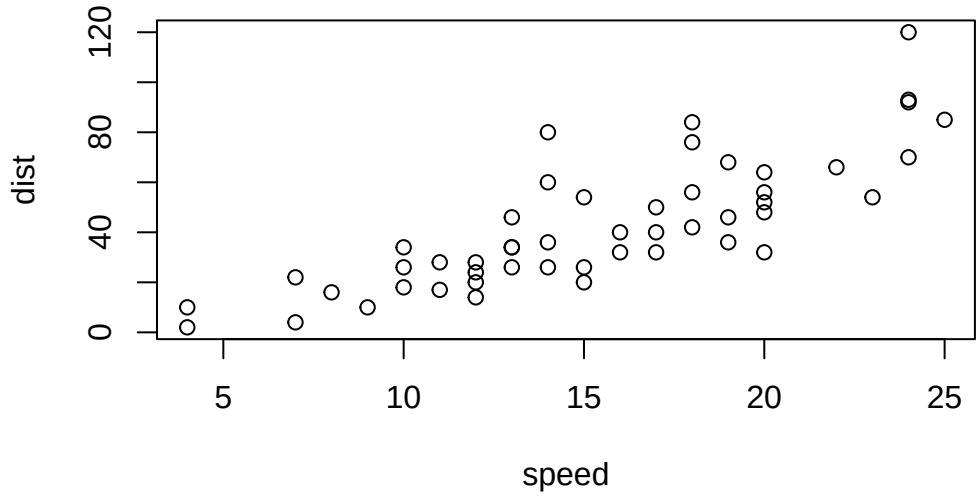
Here is a simple "base" R plot.

```
head(cars)
```

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

We can simply pass this to the `plot()` function.

```
plot(cars)
```



Key-point: Base R is quick but not so nice and simple looking in some folks eyes.

Let's see how we can plot this with **ggplot2**...

First, I need to install this add-on package. For this we use the `install.packages()` function - **WE DO THIS IN THE CONSOLE, NOT our report**. This is a one time only deal.

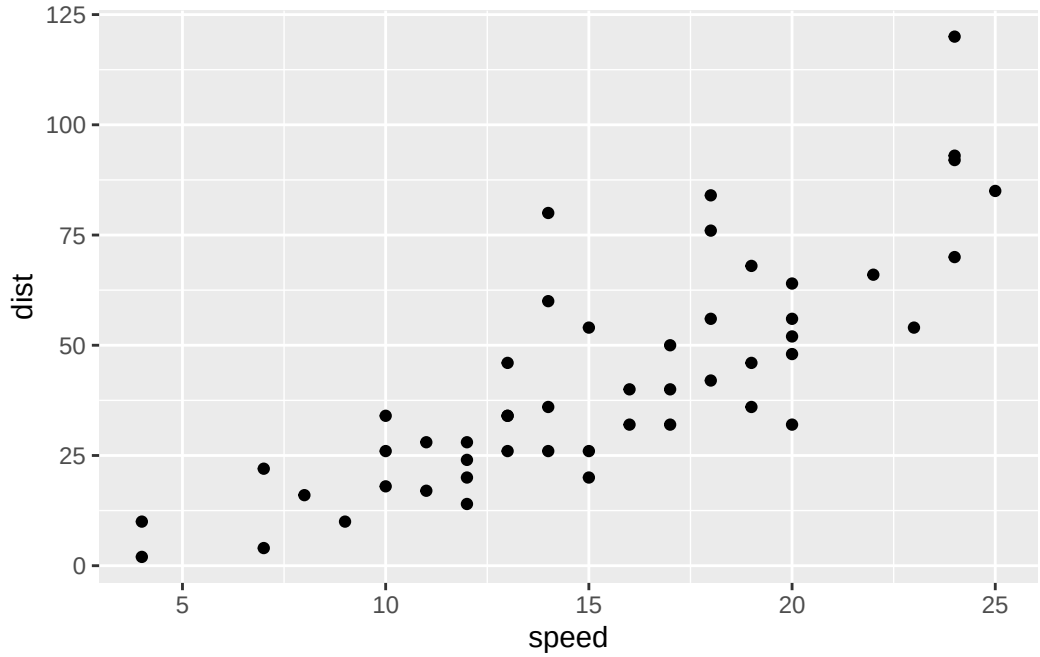
Second, we need to load the package with `library()` function every time we want to use it.

```
library(ggplot2)
ggplot(cars)
```

Every ggplot is composed of at least 3 layers:

- **data** (i.e. a data.frame with the things you want to plot)
- aesthitics **aes()** that map the columns of data to your plot features (i.e. aesthitics)
- geoms like **geom_point()** that srt how the plot appears.

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



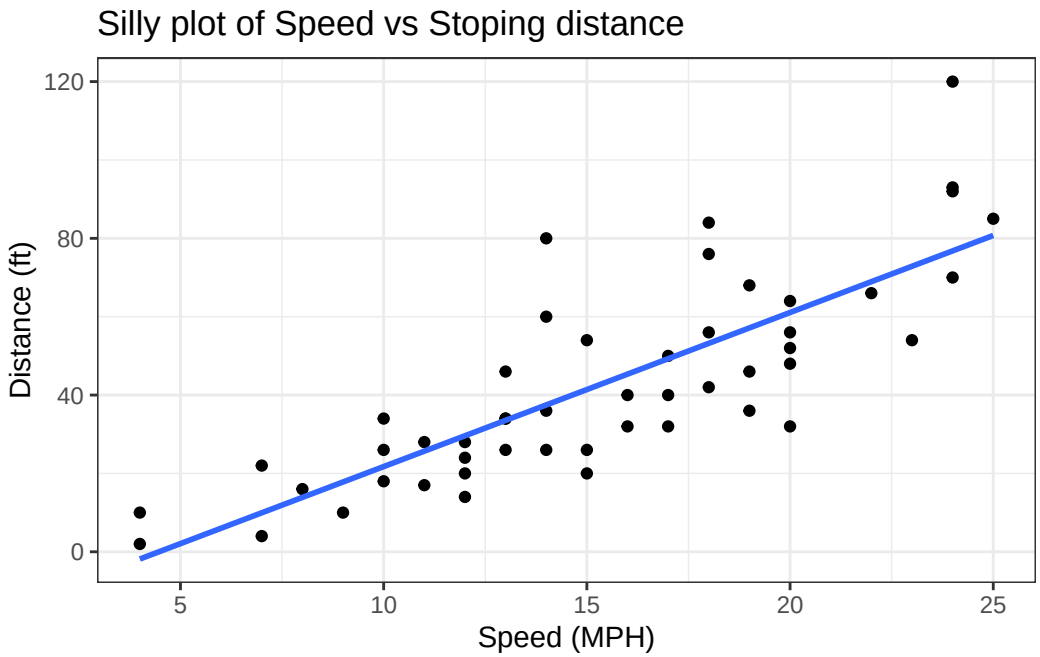
Key point: For simple "canned" graphs base R is quicker but as things get more custom and elaborate then ggplot wins out…

Let's add more layers to our ggplot.

Add a line showing the relationship between x and y. Add a title. Add custom axis labels "Speed (MPH)" and "Distance (ft)" Change the theme…

```
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point() +
  geom_smooth(method= "lm", se=FALSE) +
  labs(title= "Silly plot of Speed vs Stoping distance",
       x= "Speed (MPH)",
       y= "Distance (ft)") +
  theme_bw()
```

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



Going further

Read some gene expression data

```
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
```

Q1. How many genes are in this wee dataset?

```
nrow(genes)
```

```
[1] 5196
```

Q2. How many "up" regulated genes are there?

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

```
[1] 127
```

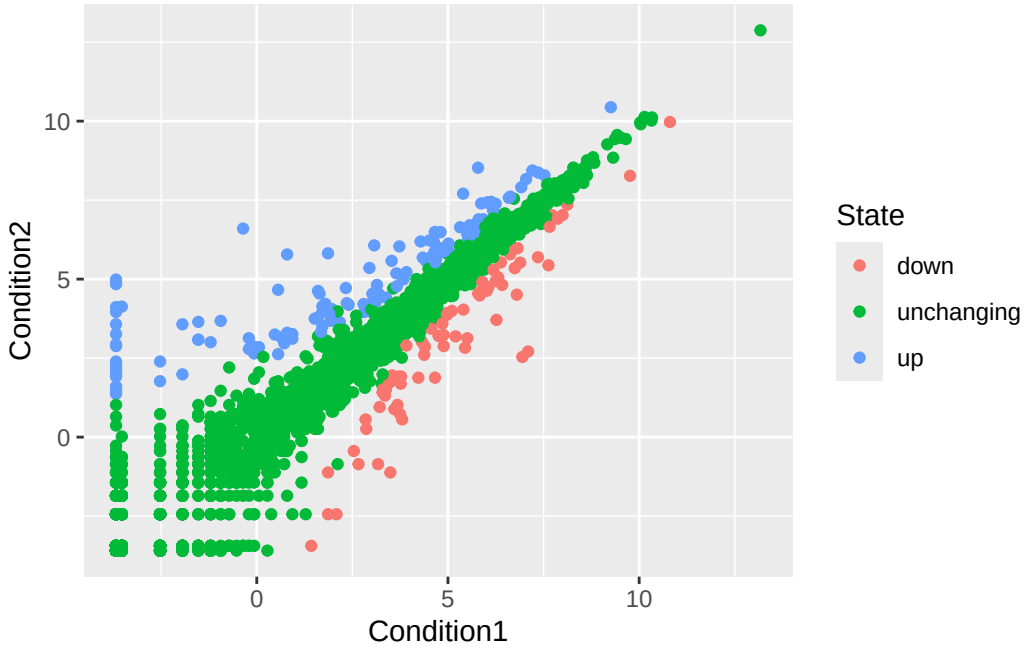
A useful function for counting up occurances of things in a vector is the `table()` function.

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

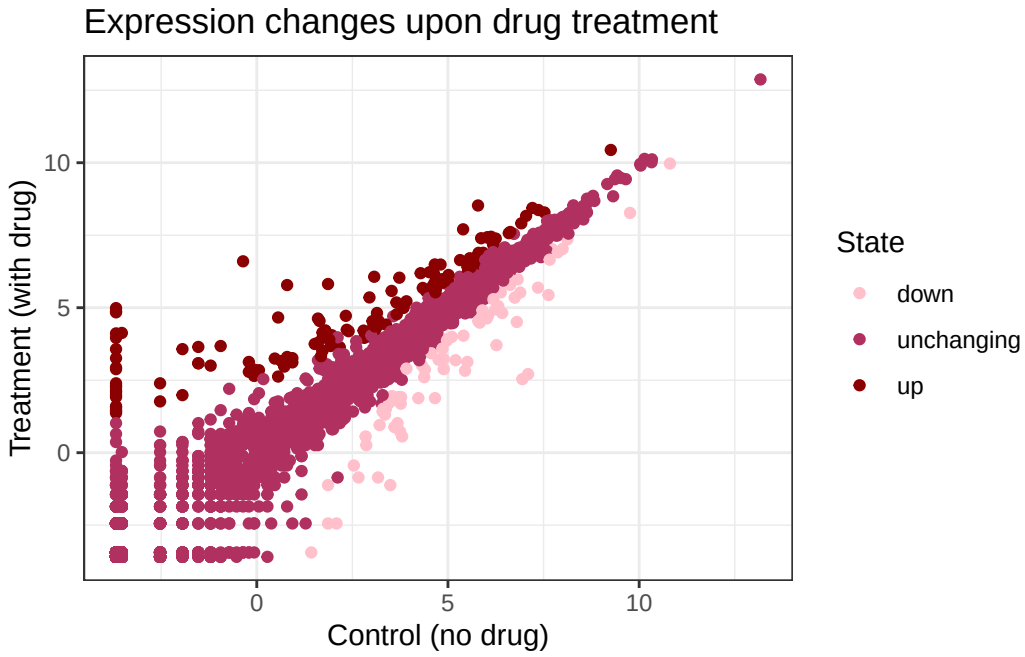
```
      down unchanging         up
        72       4997        127
```

Make a v1 figure.

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



```
p +
  scale_color_manual(values= c("pink", "maroon", "darkred")) +
  labs(title= "Expression changes upon drug treatment",
       x= "Control (no drug)",
       y= "Treatment (with drug)") +
  theme_bw()
```



More Plotting

Read in the gapminder dataset.

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

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

Let's have a wee peak

```
head(gapminder,3)
```

```
      country continent year lifeExp      pop gdpPercap
1 Afghanistan      Asia 1952  28.801  8425333  779.4453
2 Afghanistan      Asia 1957  30.332  9240934  820.8530
3 Afghanistan      Asia 1962  31.997 10267083  853.1007
```

Q4. How many different countries in this dataset?

```
length(table(gapminder$country))
```

```
[1] 142
```

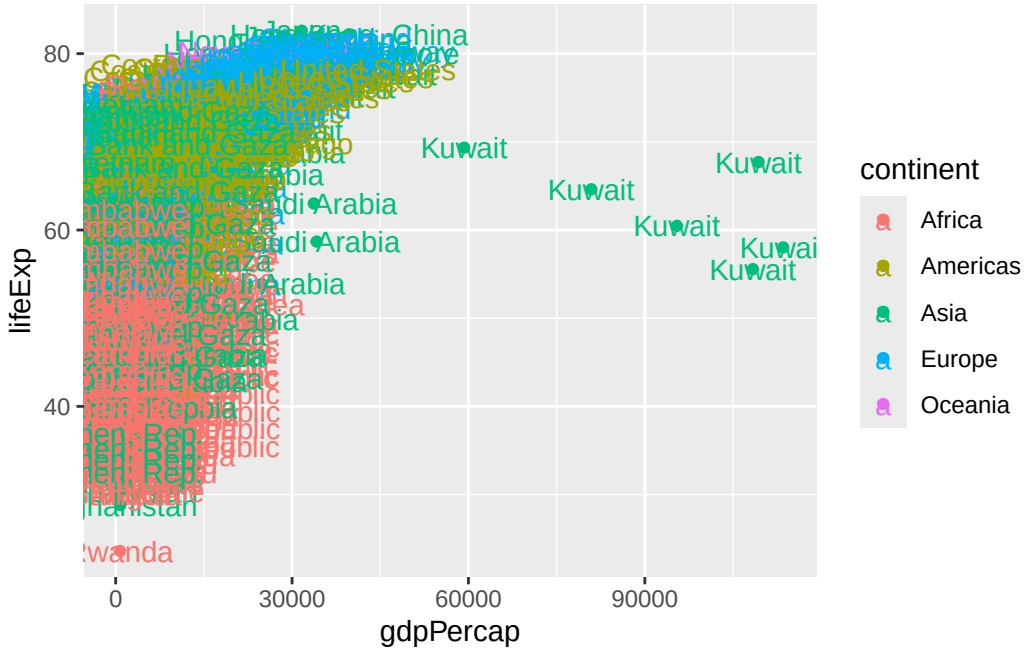
Q5. How many different continent values are in this dataset?

```
unique(gapminder$continent)
```

```
[1] "Asia"     "Europe"   "Africa"   "Americas" "Oceania"
```

```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col= continent) +
  geom_point()
```

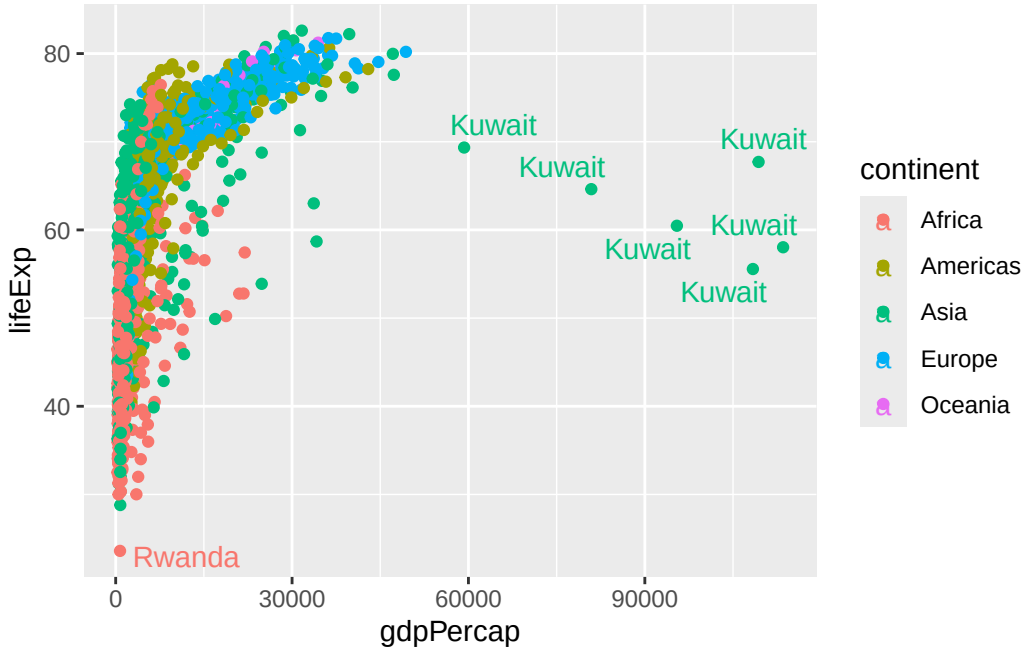
```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col= continent, label= country) +
  geom_point()+
  geom_text()
```



I can use the **ggrepel** package to make more sensible labels here.

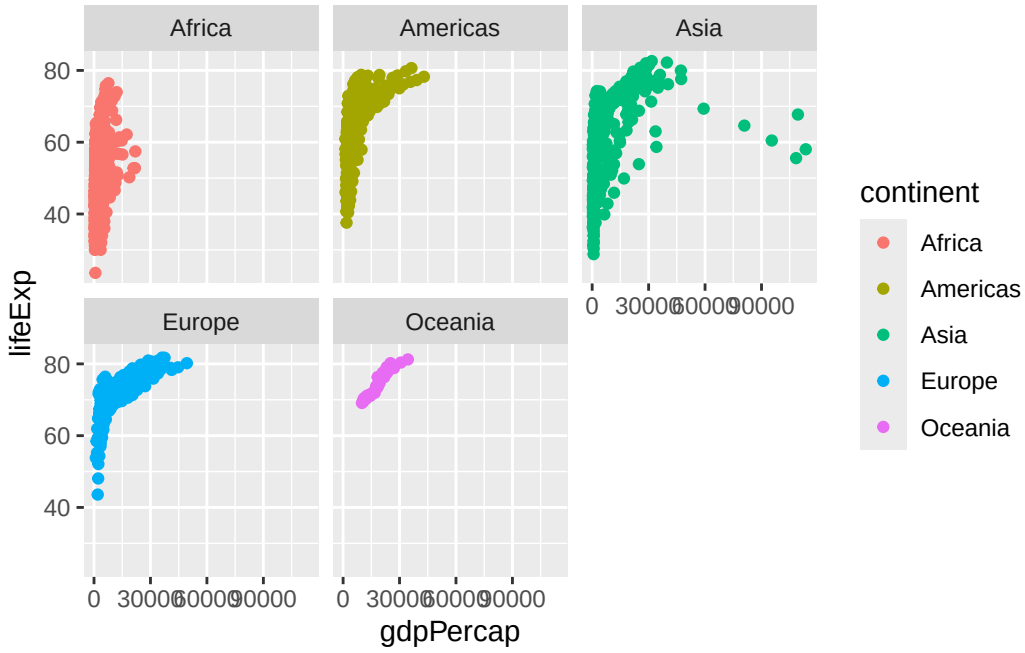
```
library(ggrepel)
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col= continent, label= country) +
  geom_point()+
  geom_text_repel()
```

```
Warning: ggrepel: 1697 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



I want a separate panel per continent.

```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col= continent, label= country) +
  geom_point()+
  facet_wrap(~continent)
```



Summary

The main advantages of ggplot over base R plot are:

1. Consistency: ggplot uses a unified grammar for all plot types, so you build every plot using the same sequence—specifying data, mapping aesthetics, and adding geometric layers. This makes it easier to learn and apply to different visualizations [1], [2], [3], [5], [4].
2. Layering: You can add layers (points, lines, themes, etc.) step-by-step, making complex plots easier to construct and modify [1], [2], [3], [5], [4].
3. Beautiful Defaults: ggplot produces attractive, publication-quality graphics with sensible defaults, reducing the need for manual tweaking [1], [2], [3], [5], [4].
4. Customization: It's easier to customize legends, colors, themes, and combine multiple plots compared to base R, which can be fiddly and time-consuming for newcomers [1], [2], [3], [5], [4].
5. Reproducibility: ggplot code is more reproducible and scriptable, making it simple to automate and regenerate figures with new data [1], [2], [3], [5], [4].