Class05: Data Viz w/ GGplot

Suraj Sidhu (A18512793)

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**

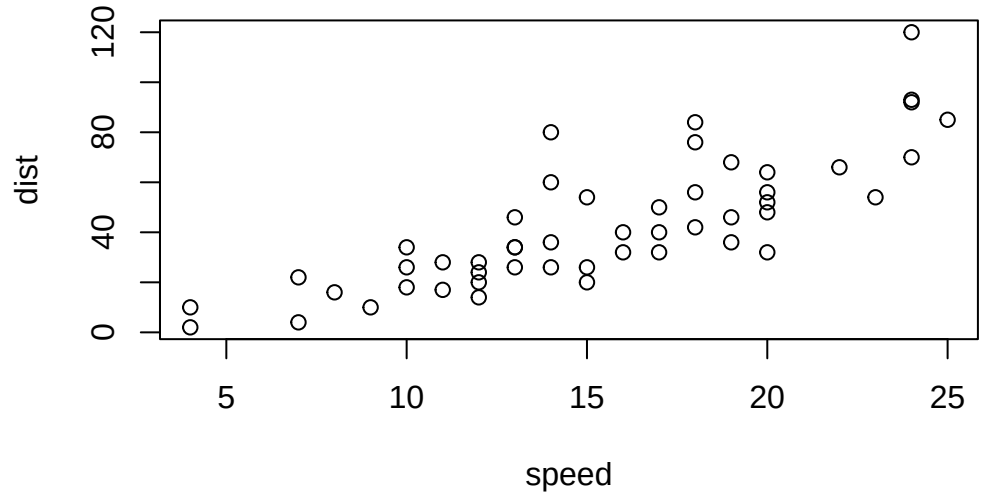
Here is a simple "base" R plot.

```
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()` fucntion.

```
plot(cars)
```



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

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

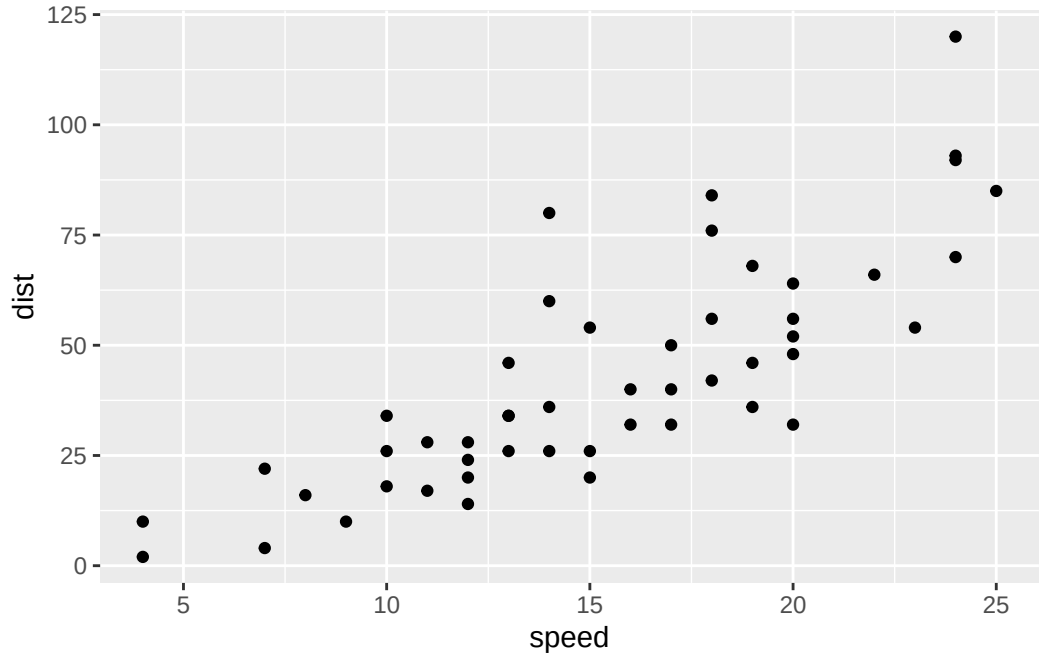
First, I need to install this add-on package. For this we use the `install.packages()` function - **WE DO THIS IN THE CONSOLE, NOT OUR REPORT**. This is a one time deal.

Second, we need to load the package with the `library()` function every tie we ant to use it

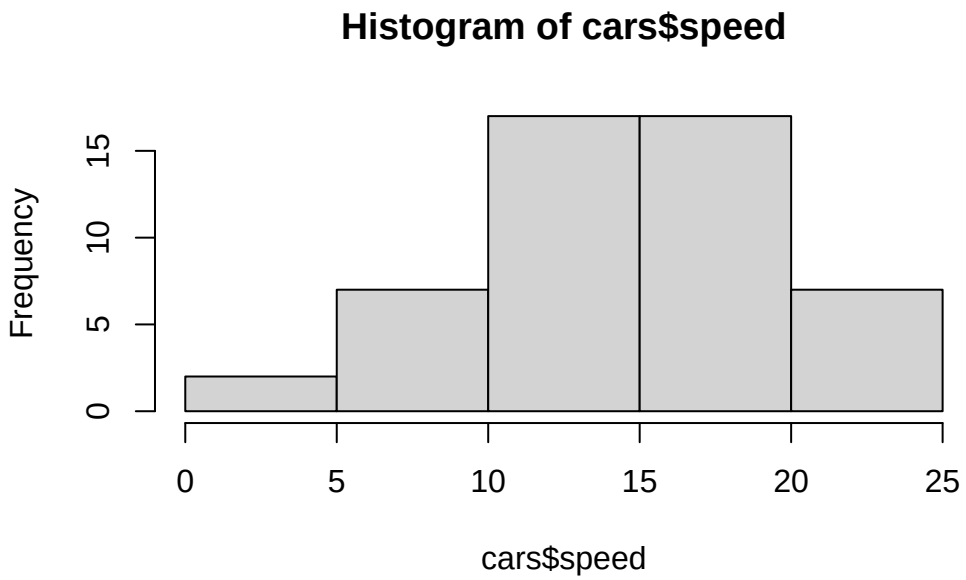
```
library(ggplot2)
ggplot(cars)
```

Every ggplot is composed of at least 3 layers: 1. **Data** (i.e a data.frame with the things you want to plot) 2. Aesthetics **aes()** that map the columns of data to your plot features (i.e aesthetics) 3. Geometry like **geom_point()** that sort how the plot appears

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




```
hist(cars$speed)
```



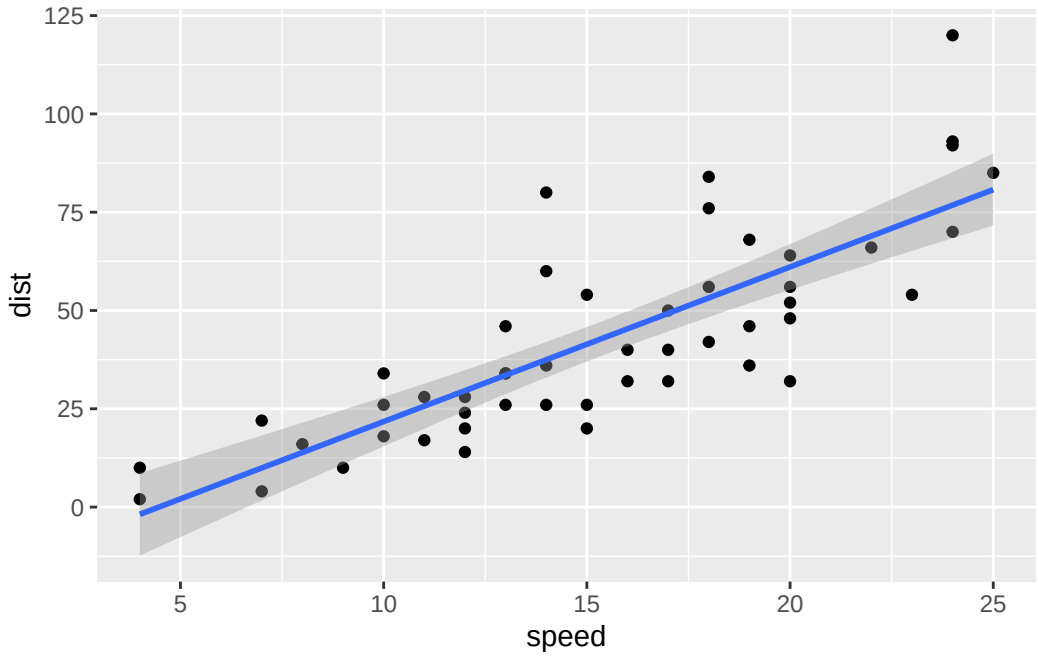
Key-Points: For simple "canned" graphs, base R is quicker but as things get more custom and elaborate then ggplot wins out…

Let's add some more layers to our ggplot

Add a line showing the relationship between x and y

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

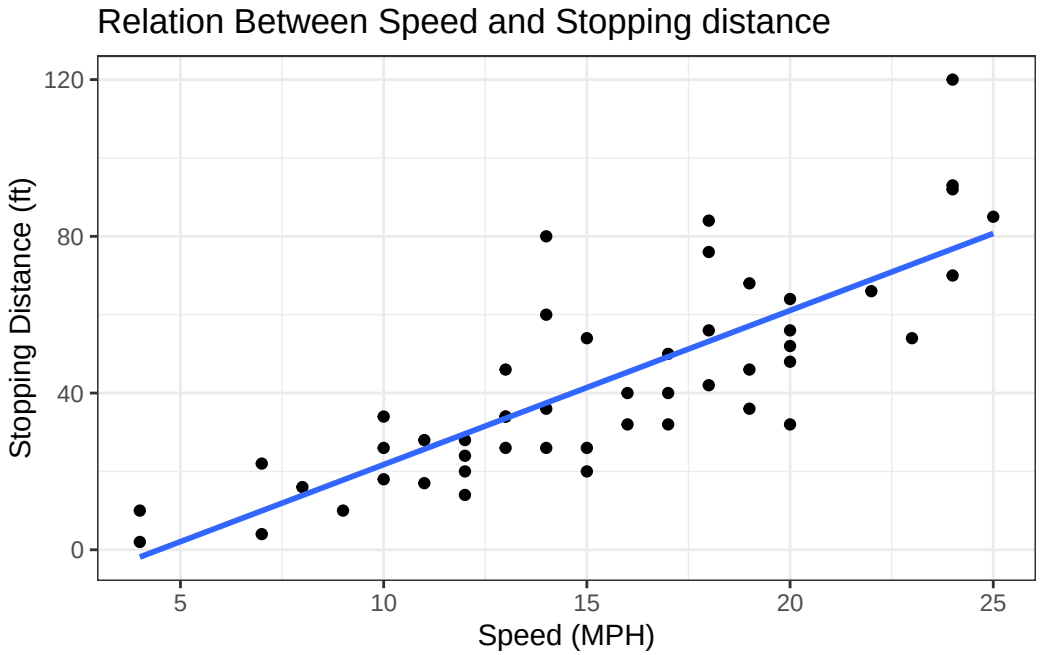
```
`geom_smooth()` using formula = 'y ~ x'
```



Now let's customize a bit more. Add labels for the axis Add a title

```
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE) +
  labs(title="Relation Between Speed and Stopping distance",
       x = "Speed (MPH)",
       y = "Stopping 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 dataset ?

```
nrow(genes)
```

```
[1] 5196
```

Q2. How many “up” regulated genes are there ?

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

```
[1] 127
```

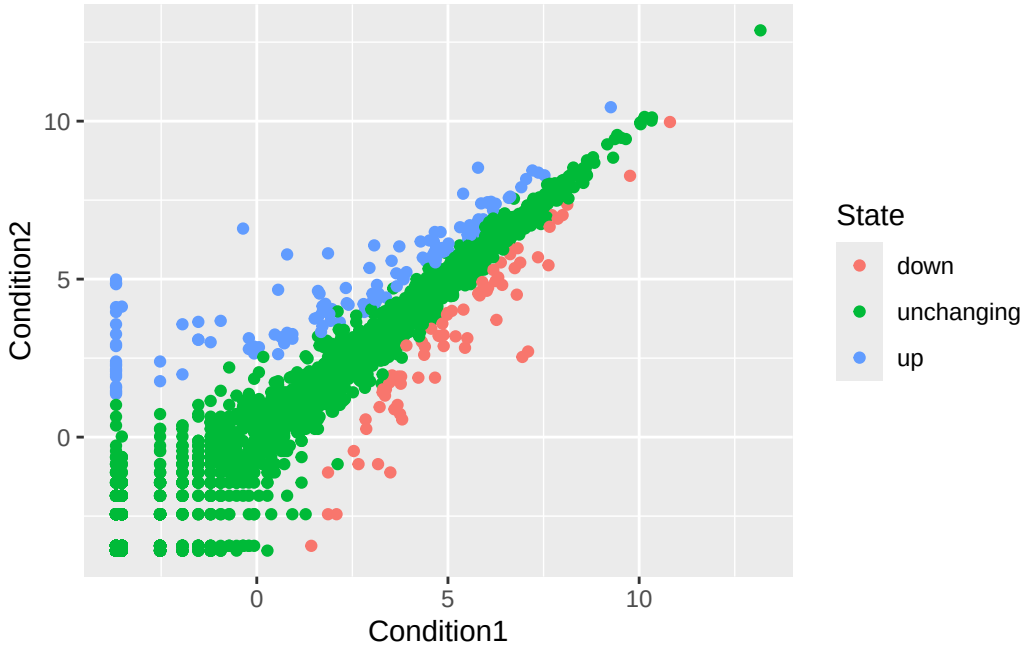
A useful function for counting up occurrences of things in a a vector is the `table()` function

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

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

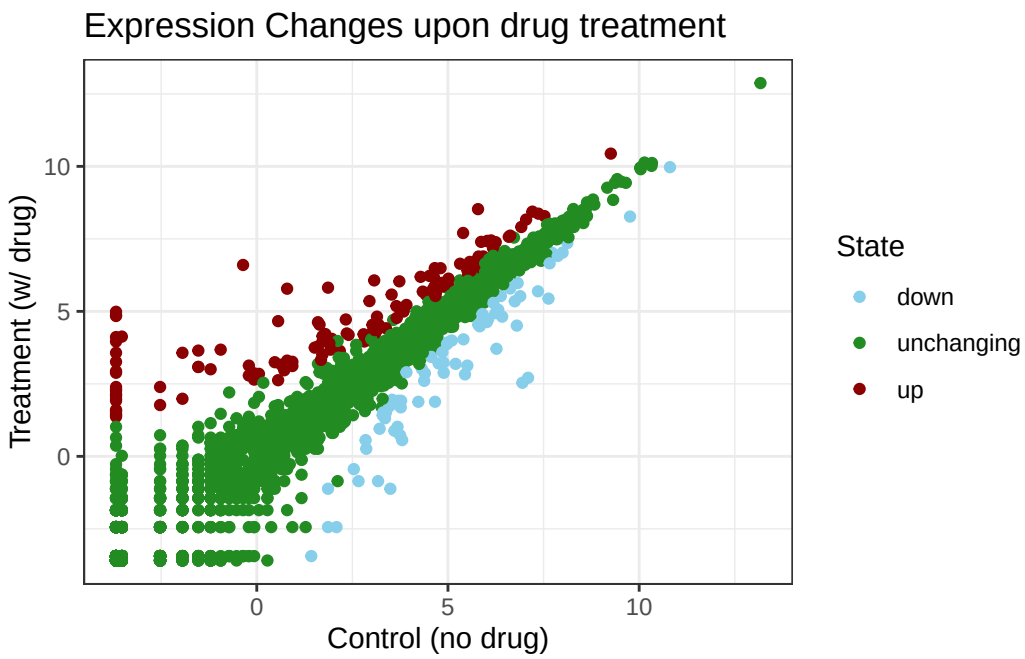
Make a V1 Figure

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



Now lets add color, a title and label the axis

```
p + scale_colour_manual(values = c("skyblue","forestgreen","darkred"))+
  labs(title = "Expression Changes upon drug treatment",
       x = "Control (no drug)",
       y = "Treatment (w/ 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)
```

Lets have a wee peek

```
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
```

```
tail(gapminder,3)
```

```
      country continent year lifeExp      pop gdpPercap
1702 Zimbabwe    Africa 1997  46.809 11404948  792.4500
1703 Zimbabwe    Africa 2002  39.989 11926563  672.0386
1704 Zimbabwe    Africa 2007  43.487 12311143  469.7093
```

Q3. How many different country values are in this dataset?

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

```
[1] 142
```

Q4. How many different continents are in the dataset ?

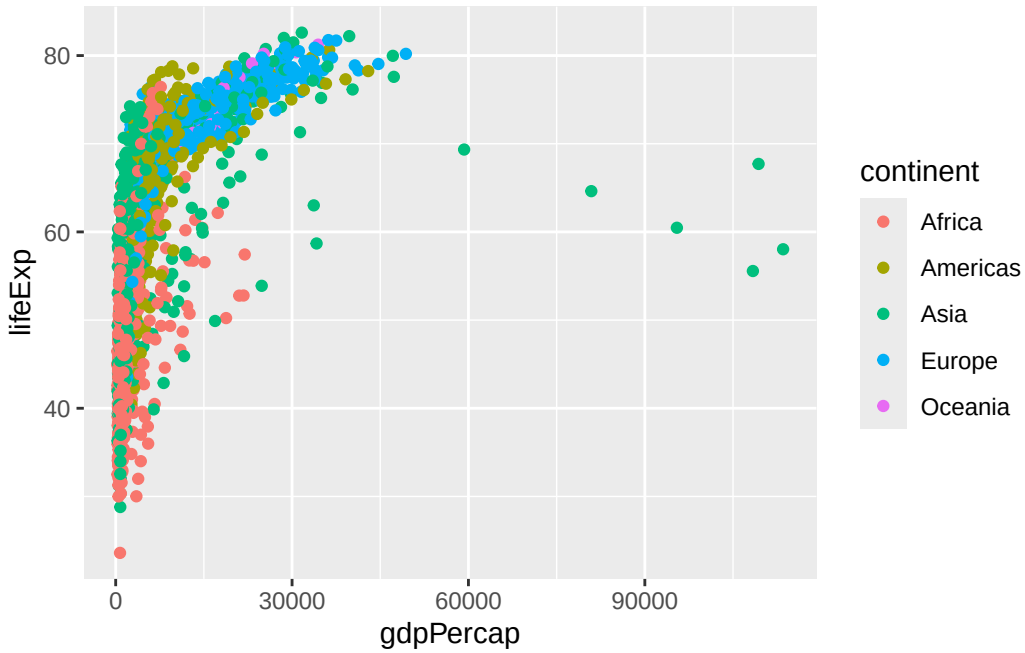
```
length(table(gapminder$continent))
```

```
[1] 5
```

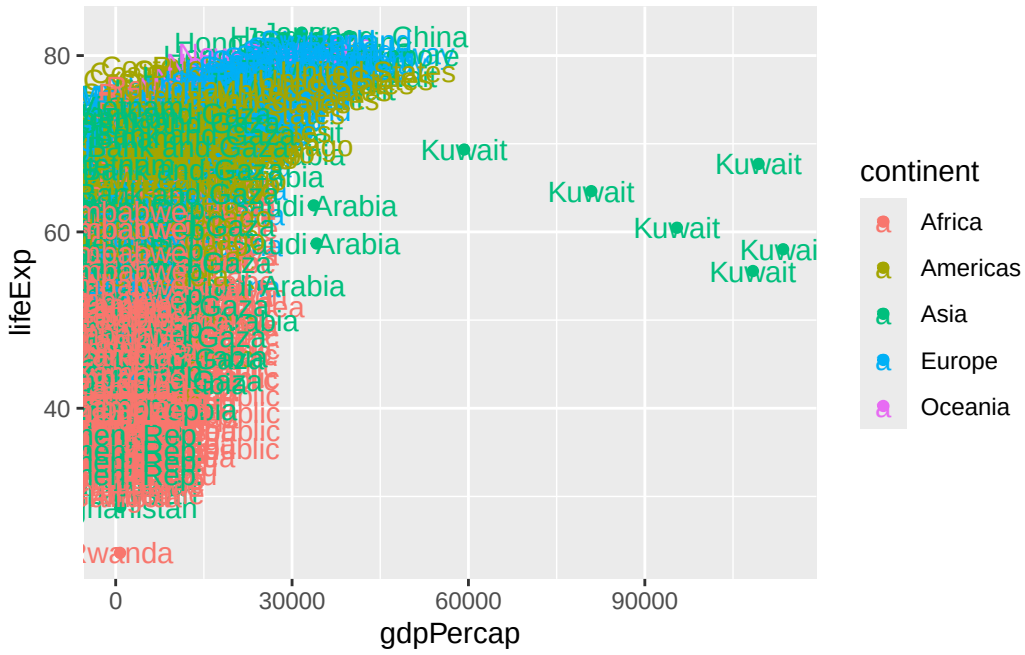
```
unique(gapminder$continent)
```

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

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



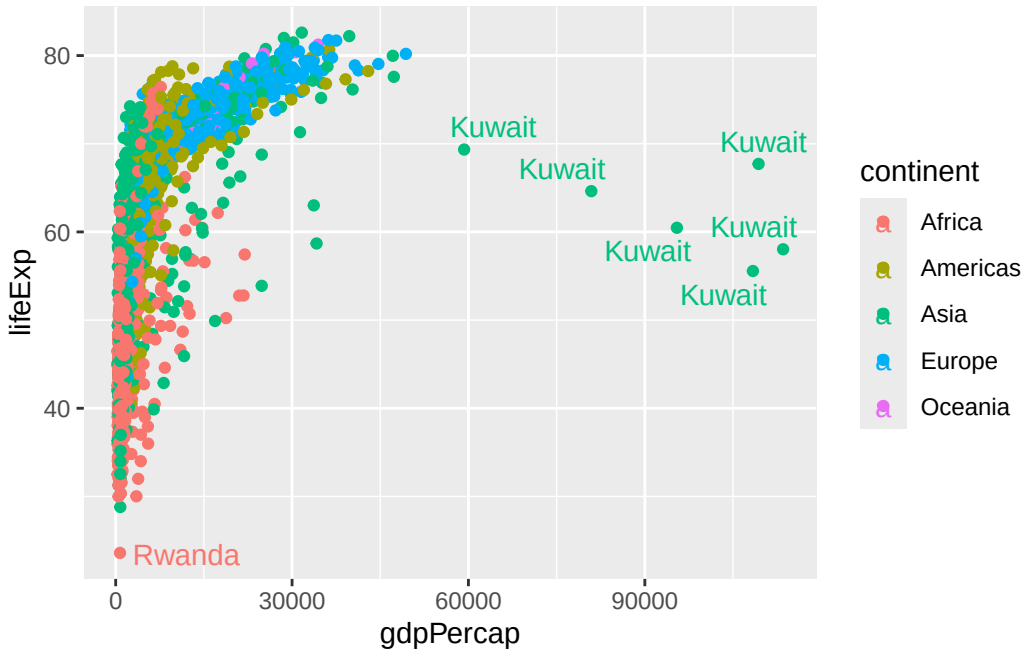

```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col=continent, label=country) +
  geom_point() +
  geom_text()
```



I can use the **ggrepel** package to make more sensible label here (always install packages in the console!)

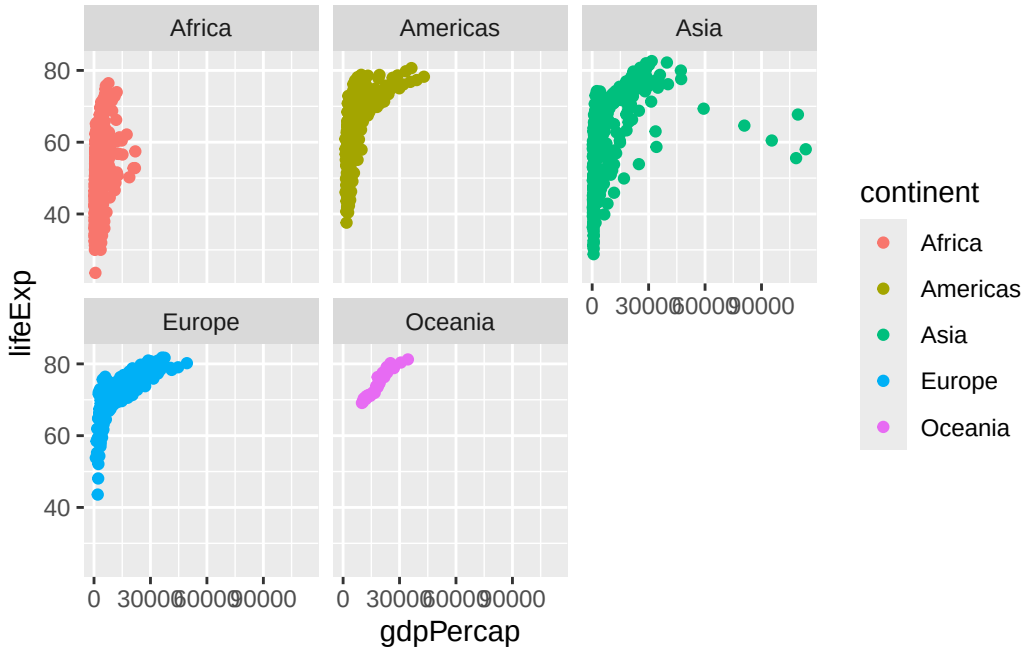
```
library(ggrepel)
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col=continent, label=country) +
  geom_point() +
  geom_text_repel()
```

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



I want a seperate panel per continent

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



what are the main advantages of ggplot over base R?

Layered Grammar of Graphics: ggplot2 uses a consistent, layered approach. You build plots by adding layers (data, aesthetics, geoms, etc.), making complex plots easier to construct and modify. Base R requires different functions and arguments for each plot type, which can be less intuitive and harder to customize for complex figures

. Aesthetic Mapping: ggplot2 makes it straightforward to map data variables to visual properties (color, size, shape, etc.) using the aes() function. This mapping is less direct and more manual in base R

. Publication-Quality Defaults: ggplot2 produces attractive, publication-ready plots with sensible defaults. Base R plots often require extensive tweaking to look polished

. Code Reproducibility and Modularity: ggplot2 code is modular and reproducible. You can easily add, remove, or modify layers. Base R code can become messy and hard to maintain for complex plots

. Customization and Extensions: ggplot2 supports extensive customization and has many extensions for specialized plots. Base R is powerful but less flexible for advanced visualizations