Class 5: Data Viz with ggplot

Mai Tamura (PID: A18594079)

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**

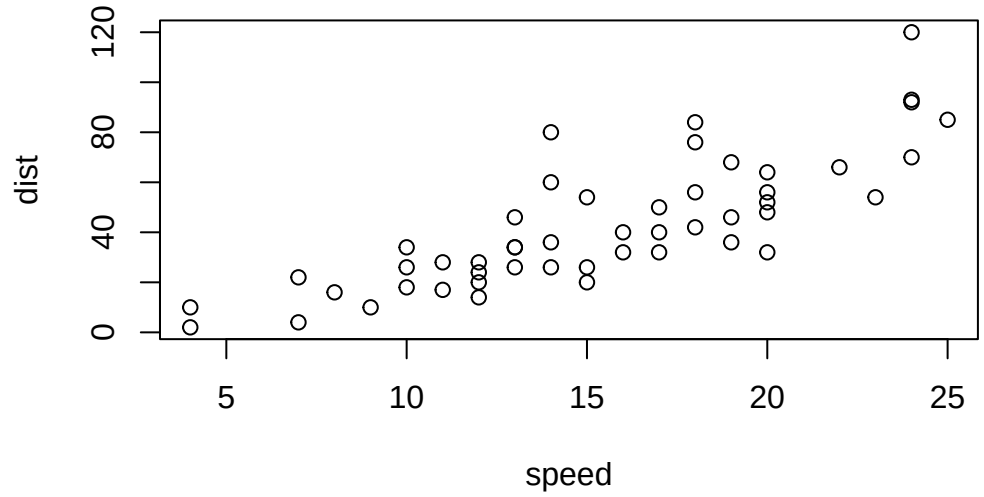
Here is a simple "base" R plot.

```
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 to the `plot()` function.

```
plot(cars)
```



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

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

1st I need to install this add-on package. For this we use the `install.package()` function - **WE DO THIS IN THE CONSOLE, NOT our report**. This is a one time only deal.

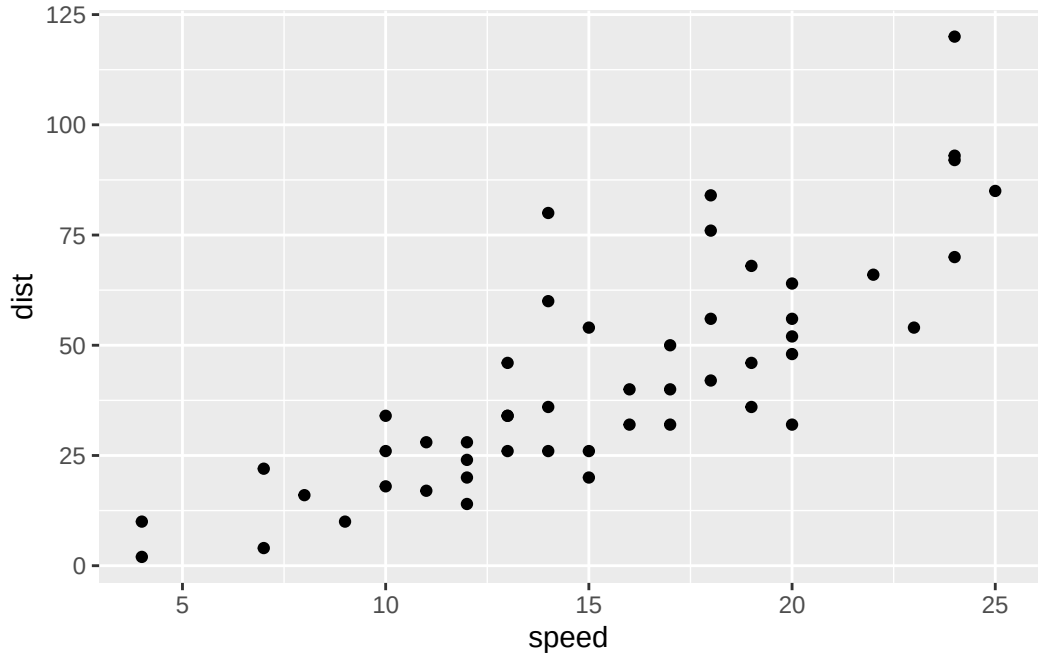
2nd we need to loas the package with the `library()` function every time we want to use it.

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

Every ggplot is composed of at least 3 layers:

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

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



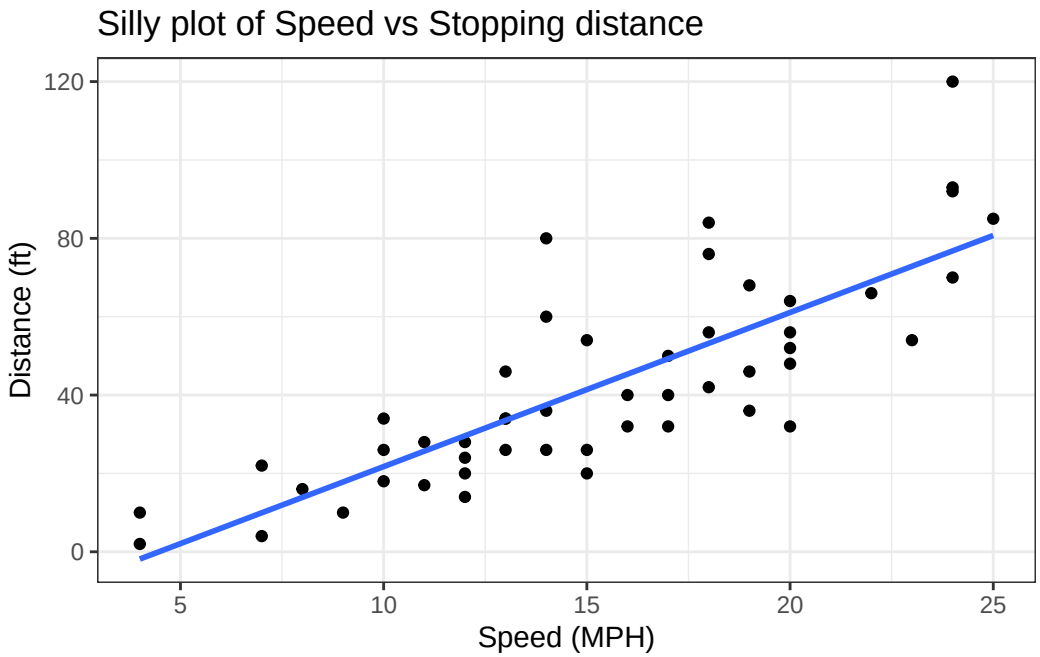
For simple "canned" graphs base R is quicker but as things get more custum and elobrate than ggplot wins out…

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 Stopping 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?

table(State$up)

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

```
[1] 127
```

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

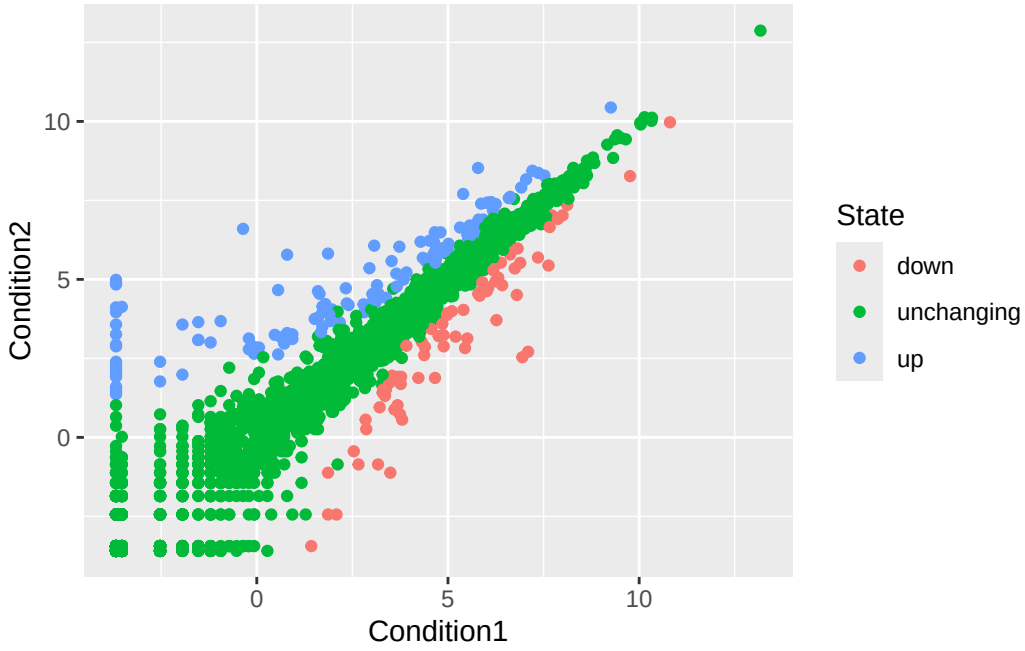
```
table(genes$State)
```

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

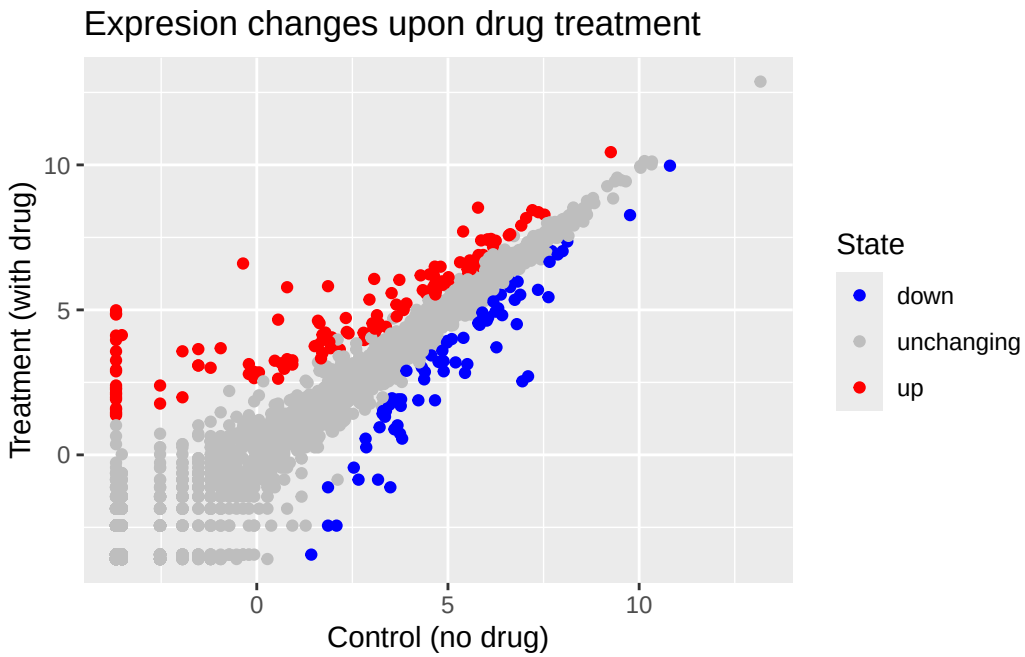
Make a v1 figure

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

p
```




```
p +
  scale_colour_manual( values=c("blue","gray","red") ) +
  labs(title="Expresion changes upon drug treatment",
       x="Control (no drug)",
       y="Treatment (with drug)")
```



More plotting

Read the gapminder dataset

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

```
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
```

Q4. How many different country values in this dataset?

```
nrow(gapminder)
```

```
[1] 1704
```

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

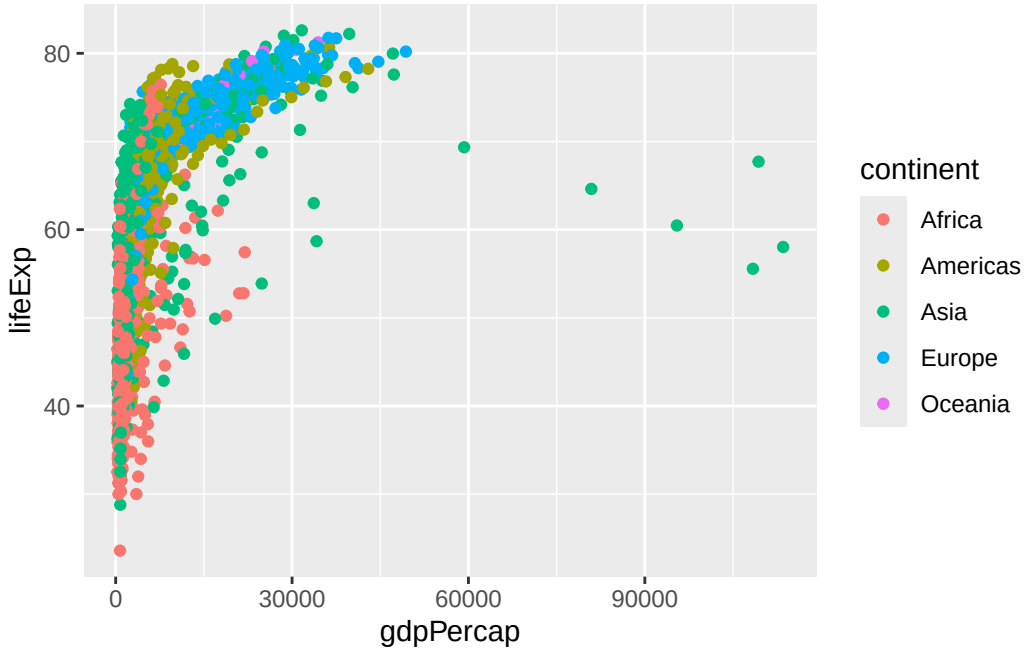
```
[1] 142
```

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

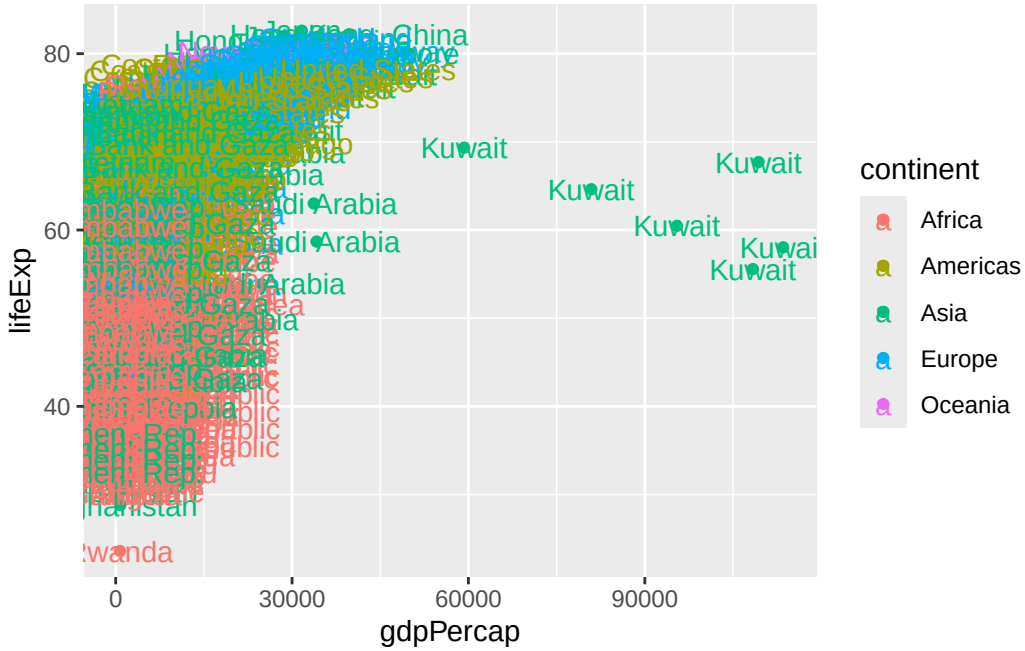
```
unique(gapminder$continent)
```

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

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



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

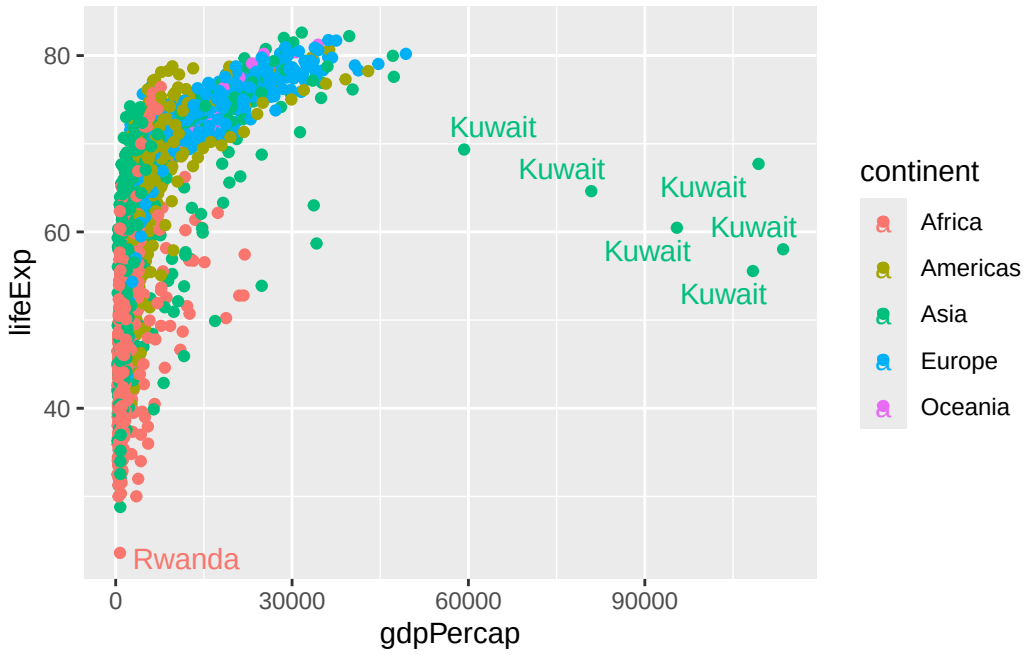


I can use **ggrepel** package to make more sensible labels here.

```
library(ggrepel)

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

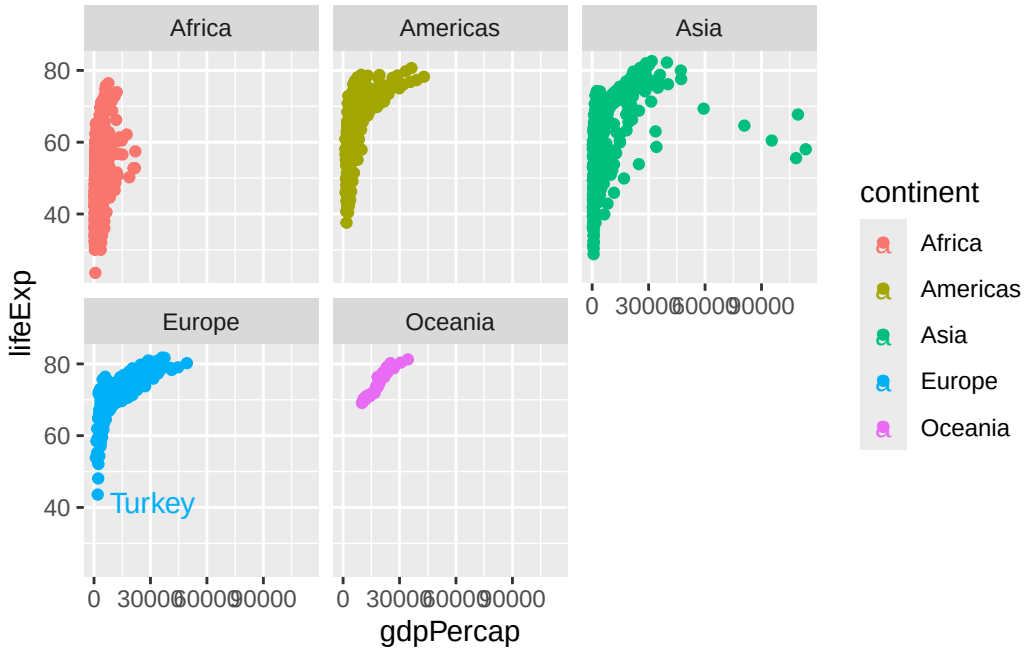
```
Warning: ggrepel: 624 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

```
Warning: ggrepel: 359 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

```
Warning: ggrepel: 300 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

```
Warning: ggrepel: 24 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

```
Warning: ggrepel: 396 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



Summary

The main advantages of ggplot over base R plot are:

1. **Layered Grammar of Graphics**: ggplot uses a consistent, layered approach (data, aesthetics, geometry) to build plots, making it easier to create complex visualizations by adding layers step-by-step [1], [3], [2], [5], [6].
2. **Declarative Syntax**: You specify *what* you want to show (mapping data to aesthetics), not *how* to draw it. This makes code more readable and easier to modify [1], [3], [2], [5], [6].
3. **Beautiful Defaults**: ggplot produces publication-quality figures with attractive default themes, reducing the need for manual tweaking [1], [3], [2], [5].
4. **Customization and Extensibility**: It is easier to customize colors, labels, themes, and add new layers (like trend lines, annotations) compared to base R, which often requires more code and manual adjustments [1], [3], [2], [5].
5. **Consistency Across Plot Types**: The same syntax and logic applies to many plot types, unlike base R where each plot type may require a different function and arguments [1], [3], [2], [5].
6. **Faceting and Grouping**: ggplot makes it simple to split data into subplots (facets) and color/group by variables, which is more cumbersome in base R [3], [2], [5].