Class5: Data Viz with ggplot

Nathan (A17395036)

Today we are exploring the **ggplot** package and how to make a nice figure 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**

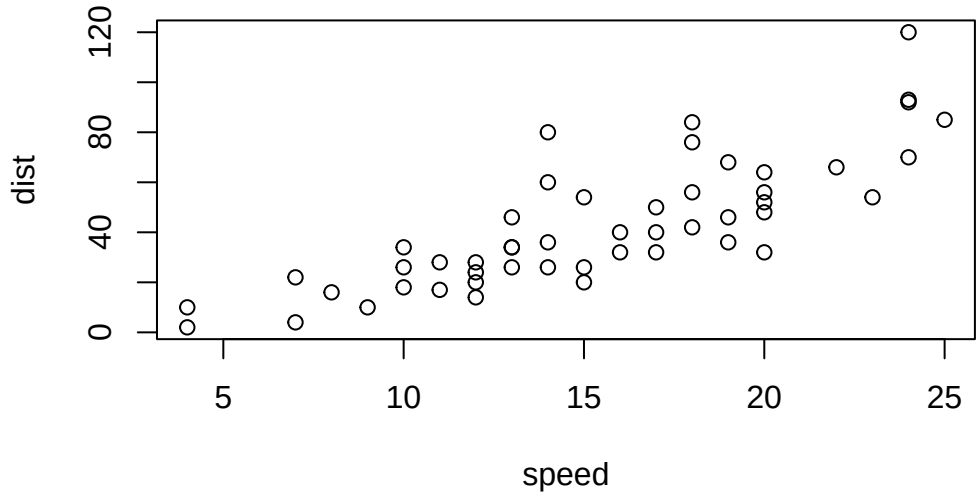
Here is a simple "base" R plot

```
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.packages()` function - **WE DO THIS IN THE CONSOLE, NOT OUR REPORT**. This is a one time only dea.

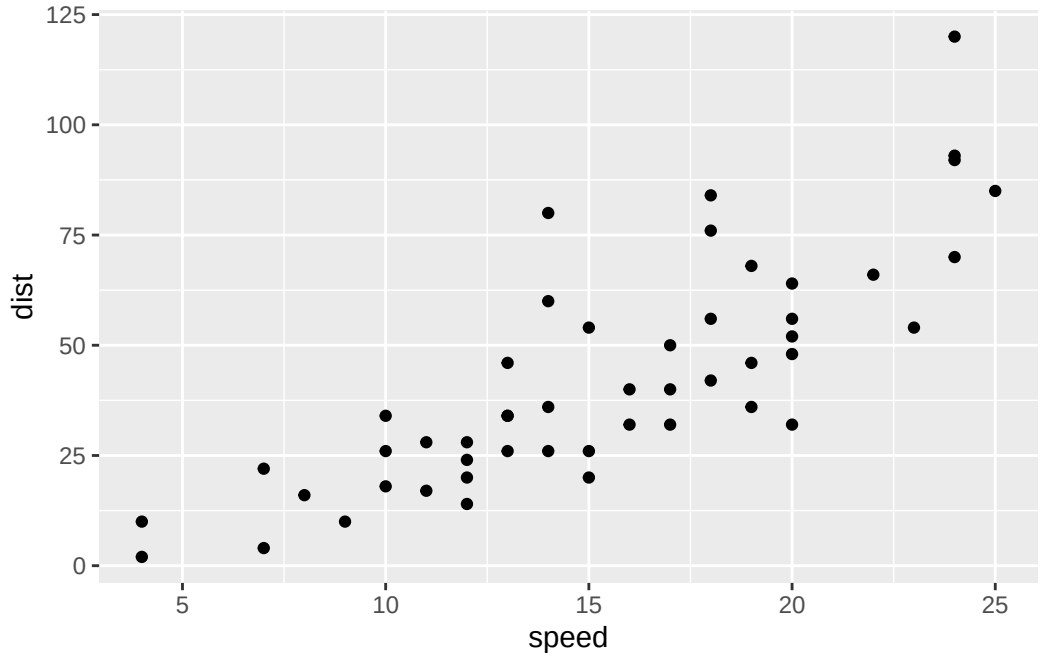
2nd We need to load the package with `library()` function every time we wan to use it

```
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),
- aesthetic **aes()** that map the columns of data to your plot features (i.e. aesthetics)
- geoms like **geom_point()** that srt how the plot appeasrs

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



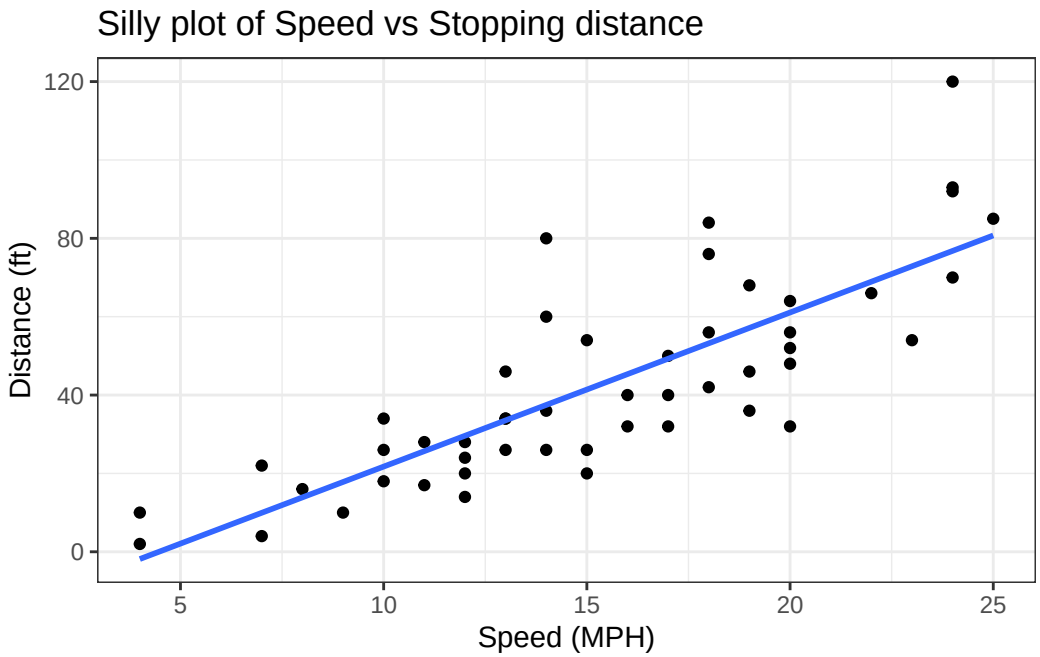
Key point: For simple "canned" graphs base R is quicker but as things get more custom and elaborate then ggplot wins out

Lets'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
```

```
ncol(genes)
```

```
[1] 4
```

Q2. How man "up" regulated genes are there

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

```
[1] 127
```

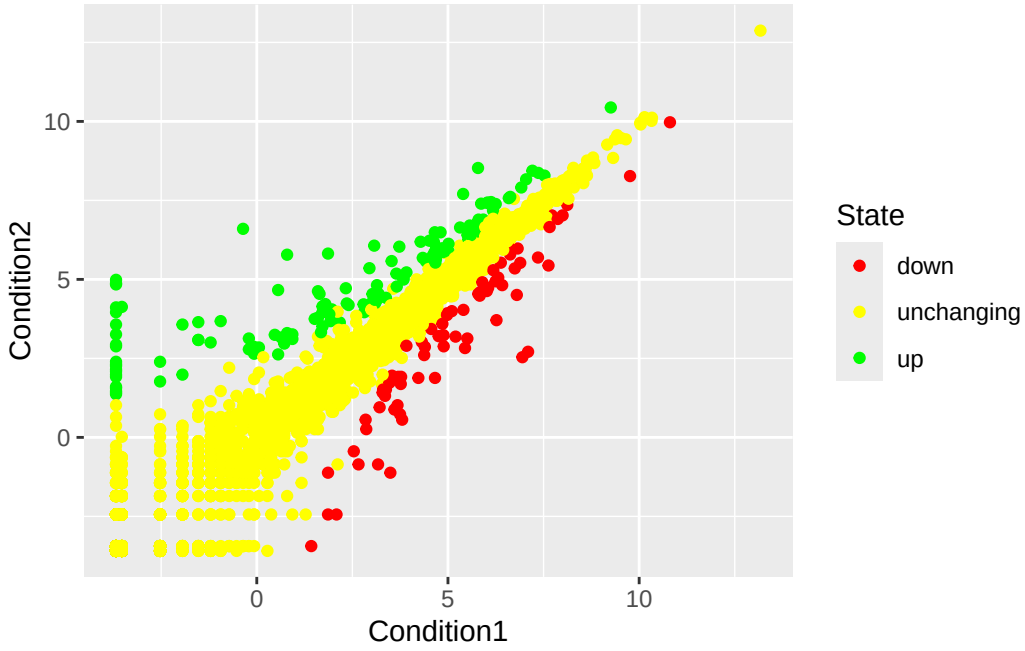
A useful function for counting up occurances of things in a vector is the `table()` function.

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

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

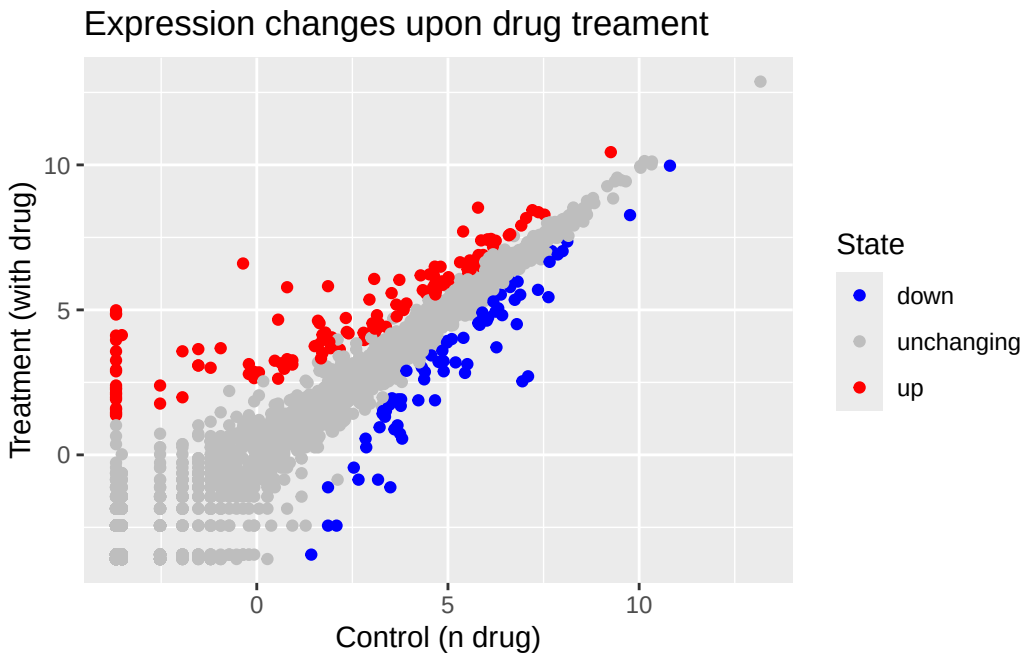
Make a v1 figure

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

p
```




```
p +
  scale_colour_manual( values=c("blue","gray","red"))+
  labs(title = "Expression changes upon drug treament",
       x = "Control (n drug)",
       y = "Treatment (with drug)")
```

```
Scale for colour is already present.
Adding another scale for colour, which will replace the existing scale.
```



More Plotting

Read in 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 cuontry values are in this dataset?

```
nrow(gapminder)
```

```
[1] 1704
```

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

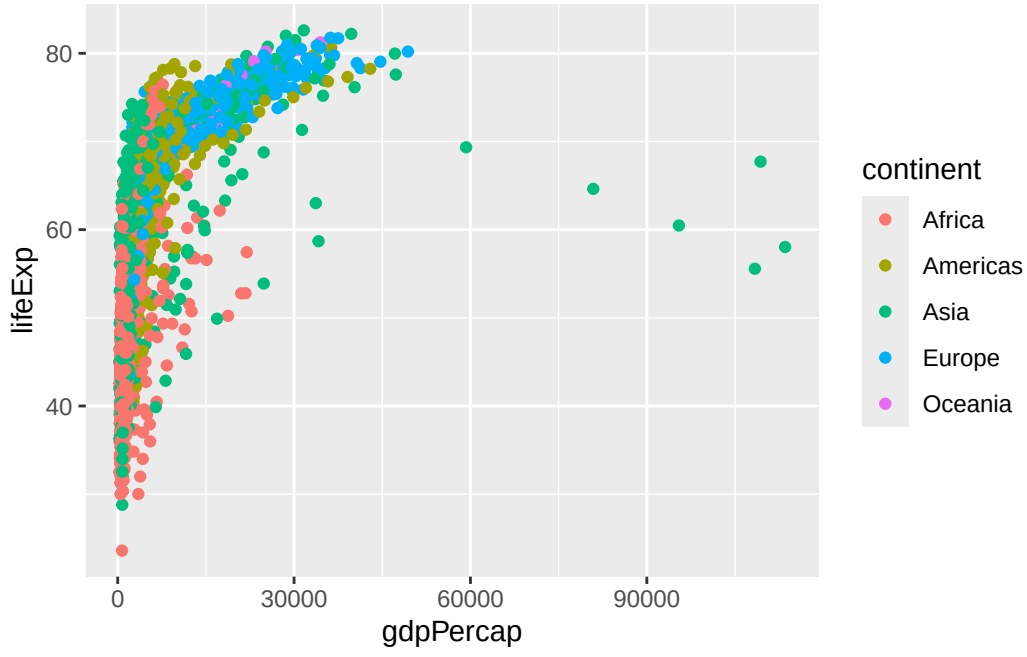
```
[1] 142
```

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

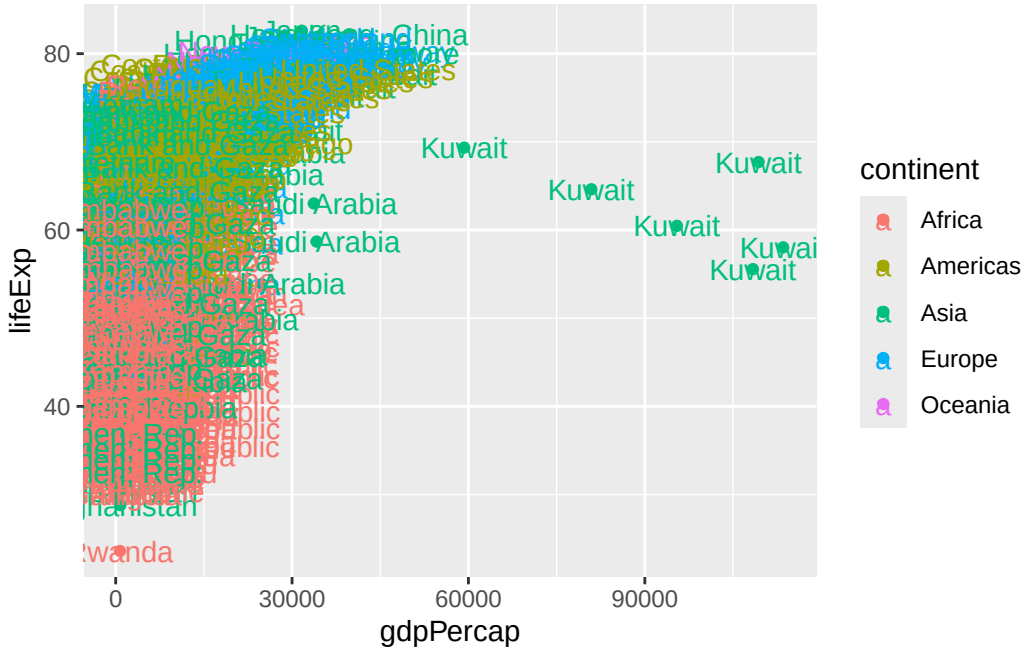
```
unique(gapminder$continent)
```

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

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



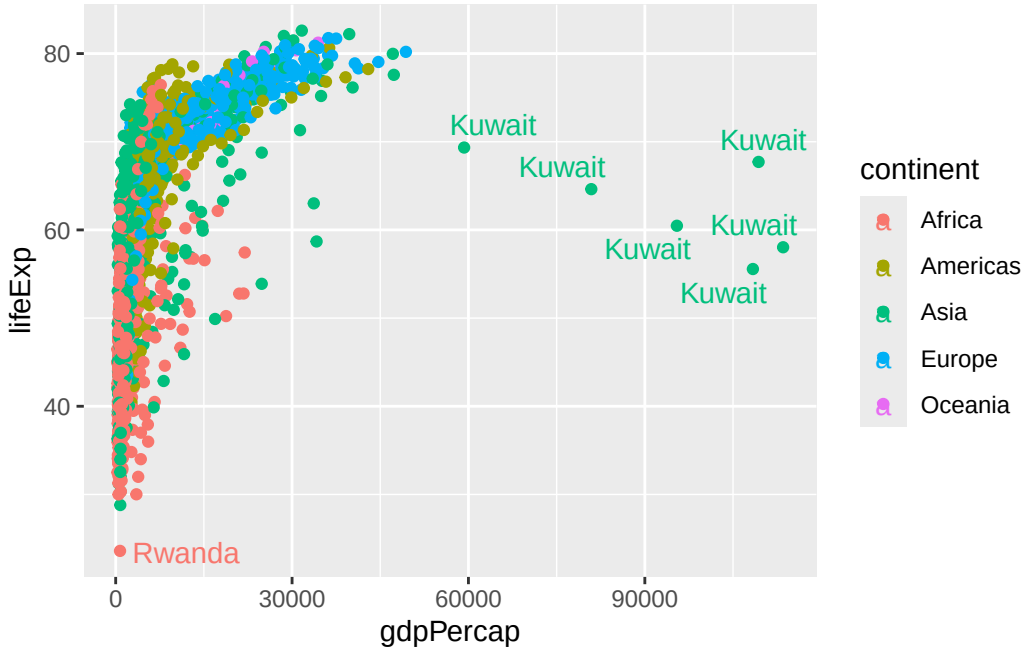
```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, col=continent, label=country) +
  geom_point() +
  geom_text()
```



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

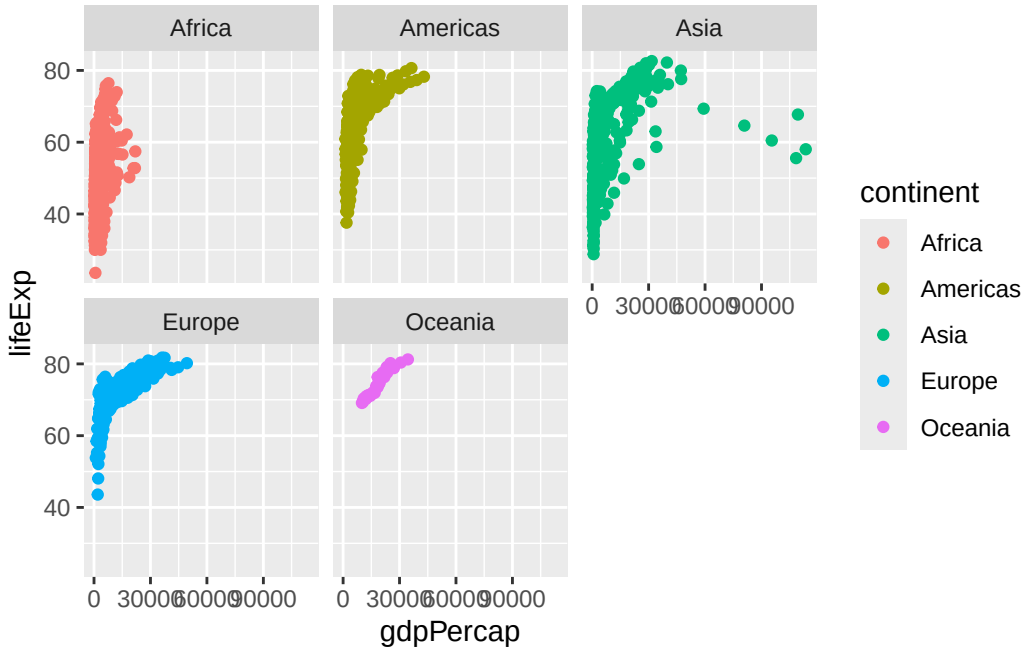
```
library(ggrepel)

ggplot(gapminder) +
  aes(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 pannel 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:

Consistent Layered Grammar: ggplot uses a consistent grammar of graphics, where you build plots by layering data, aesthetic mappings, and geometric objects. This makes it easier to create complex, publication-quality figures by adding layers step-by-step, rather than handling each plot type separately as in base R 1 , 3 , 2 , 5 . Declarative Syntax: You specify what you want to see (e.g., which variables map to axes, colors, shapes) rather than how to draw each element. This makes code more readable and easier to modify 1 , 3 , 2 , 5 . Beautiful Defaults: ggplot provides attractive default themes and legends, so your plots look good with minimal effort. Base R plots often require more manual tweaking to look polished 1 , 3 , 2 , 5 . Scalability for Complex Plots: For simple plots, base R is quick. But for complex, multi-layered figures, ggplot is more concise and manageable, while base R can become unwieldy 1 , 3 , 2 , 5 . Customization and Extensibility: ggplot makes it easy to add custom layers, annotations, and themes, and supports many plot types and extensions 1 , 3 , 2 , 5 .