Class 5: Data Viz with ggplot

Berne Chu (A18608434)

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 **ggplots2**

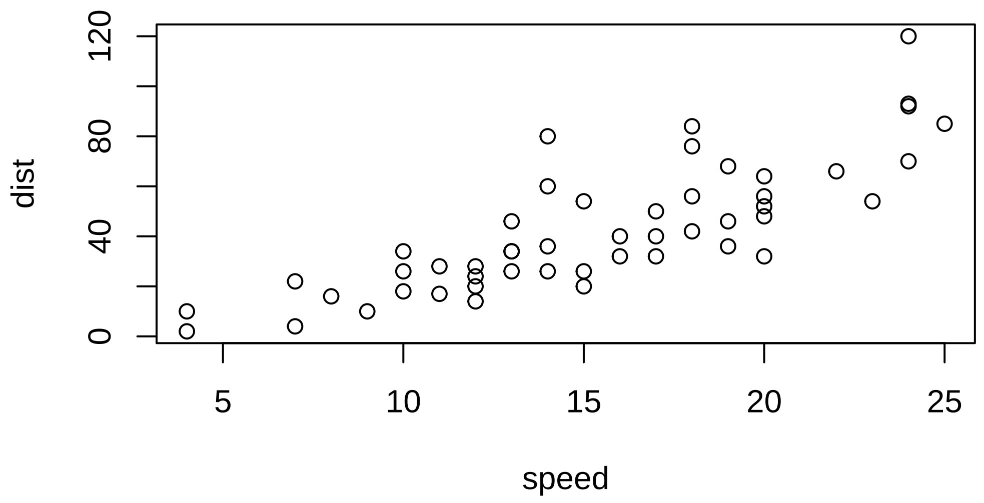
Here is a simple "base" R plots.

```r
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.

```r
plot(cars)
```



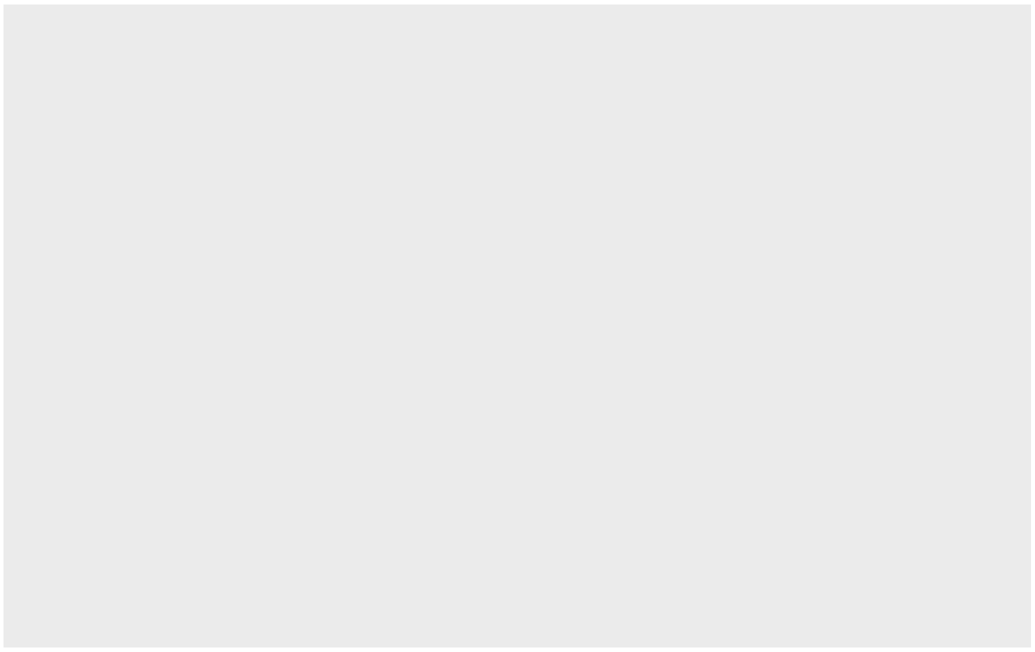
Key-point: Base R is quick but not so nice looking.

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 on time only deal.

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

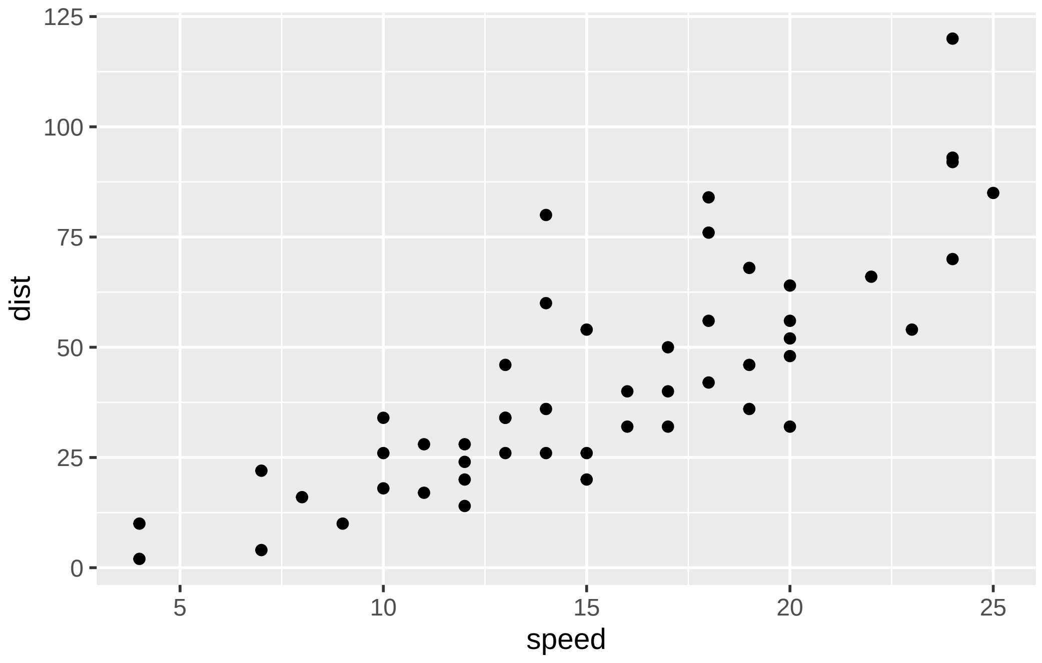
```
library(ggplot2)
ggplot(cars)
```



Every ggplot is composed of at least 2 layers:

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

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



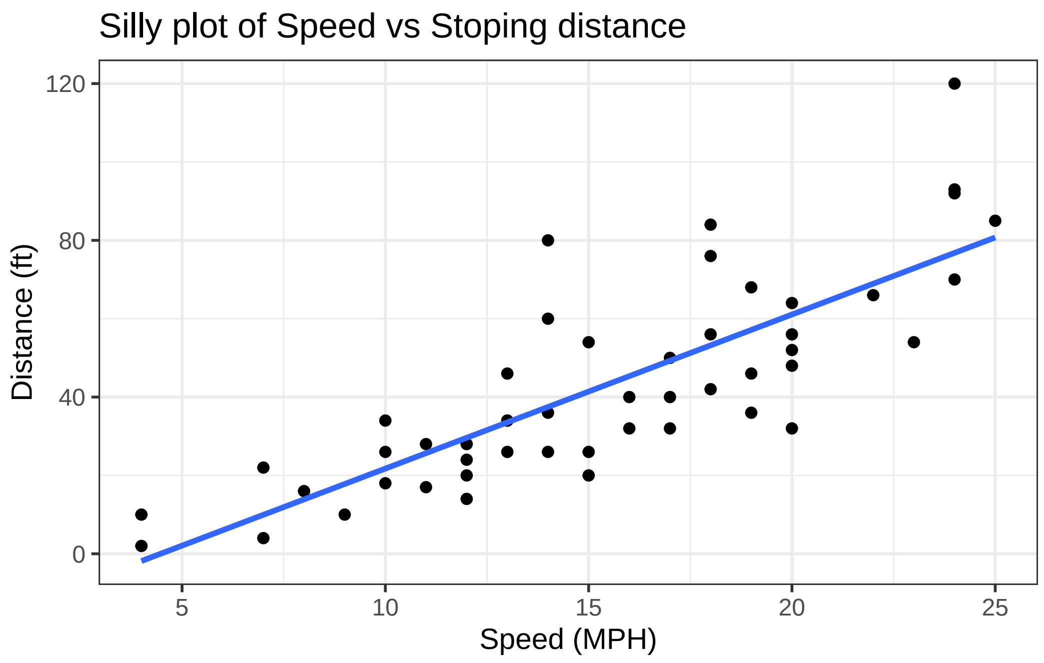
> Key point: For simple "canned" graphs base R is quicker but as things get more custom and elobrate then ggplot wins out…

Let's add more layers to our ggplot

Add a line showing the relationship between x and y Add a title Add custom axis labs "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
```

```r
ncol(genes)
```

```
[1] 4
```

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

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

```
[1] 127
```

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

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

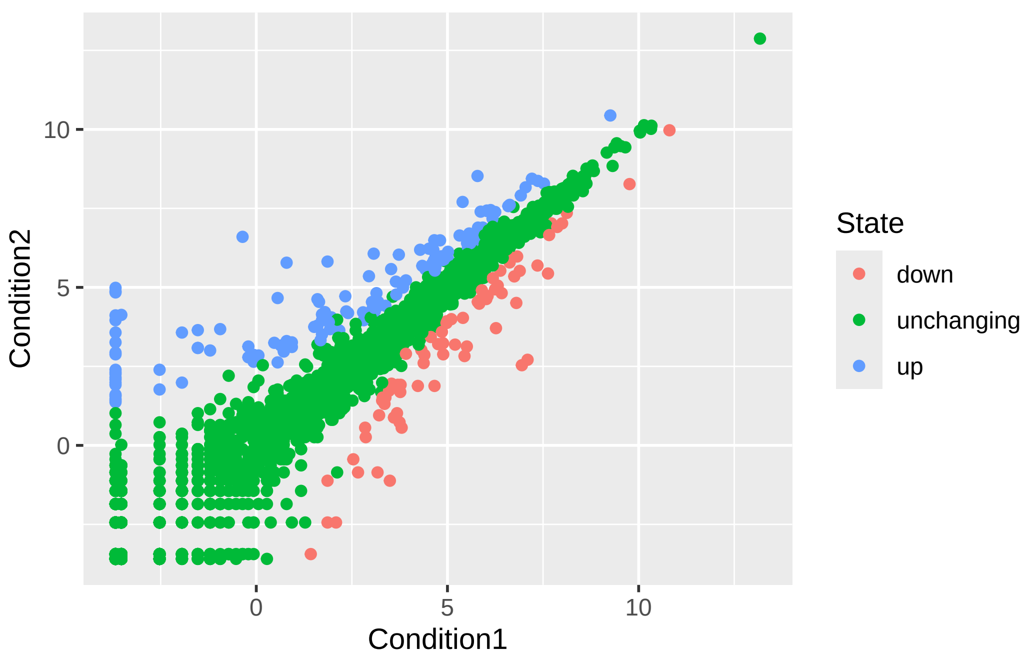
```

      down unchanging         up 
        72       4997        127 
```

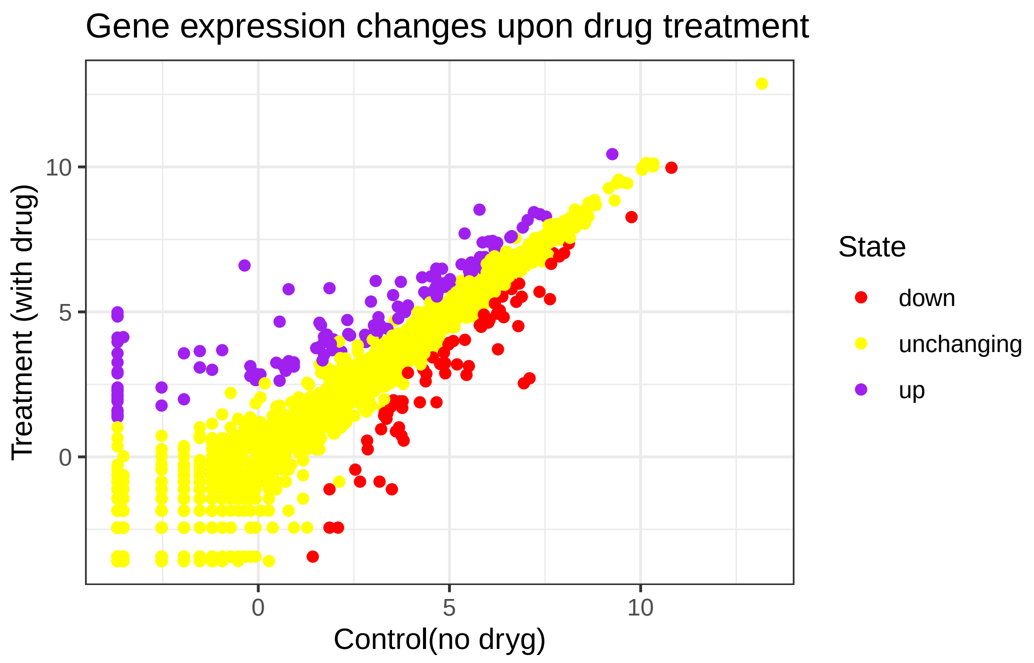
Make a v1 figure

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

p
```



```
p +
  scale_colour_manual(values=c("red", "yellow", "purple"))+
  labs(title="Gene expression changes upon drug treatment",x="Control(no dryg)",
       y="Treatment (with drug)")+
  theme_bw()
```



More Plotting

Reading in the gapmider dataset

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

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

Lets 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
```

```r
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 are in the dataset?

```r
nrow(gapminder)
```

```
[1] 1704
```

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

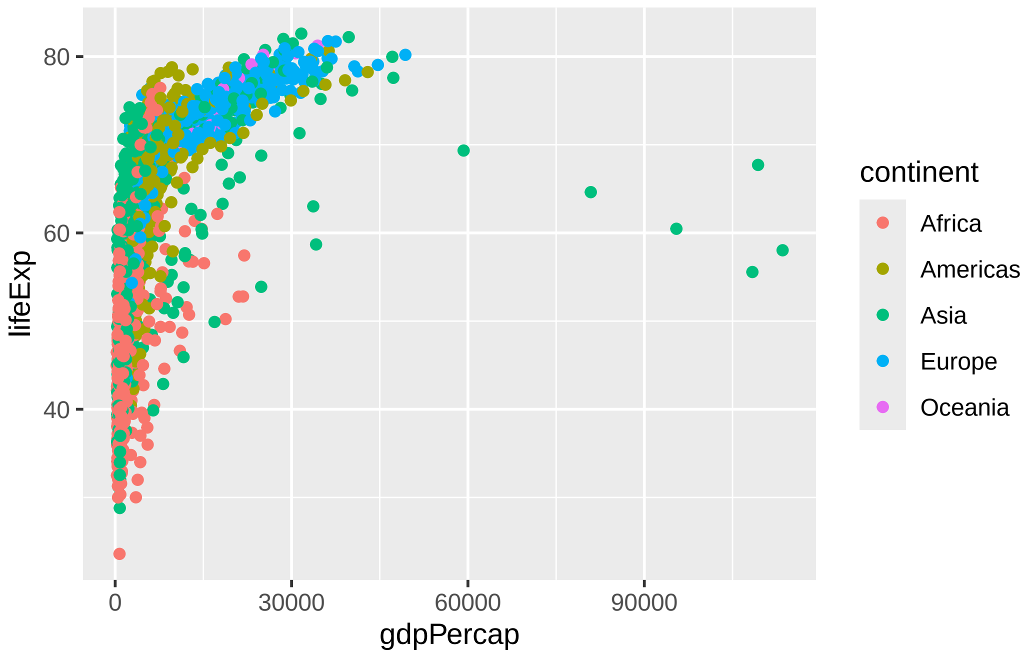
```
[1] 142
```

Q5. How many different continent calues are in this dataset

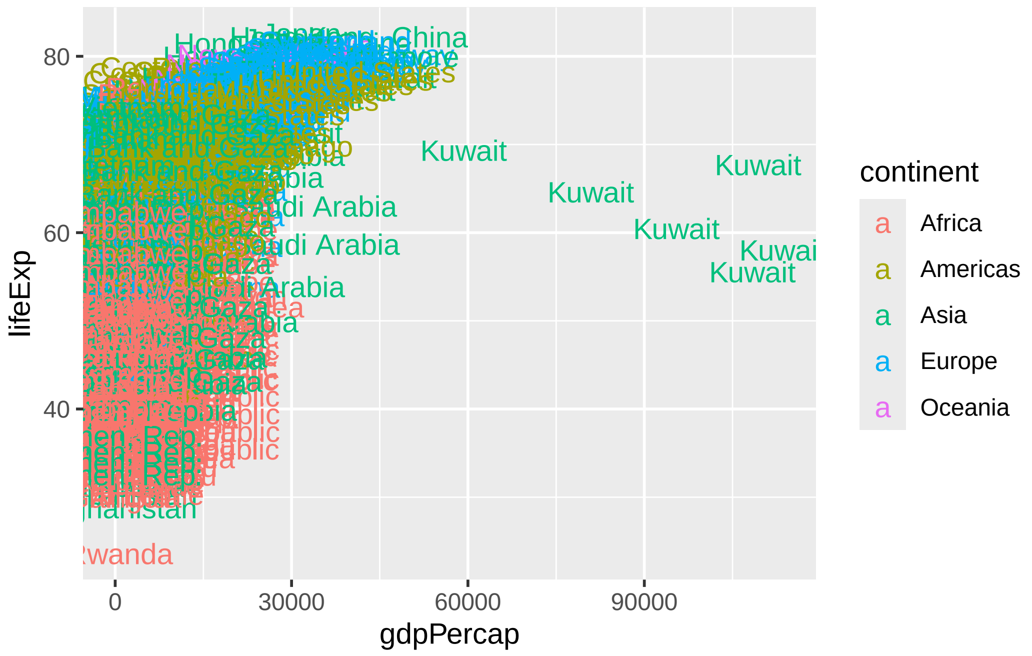
```r
unique(gapminder$continent)
```

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

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



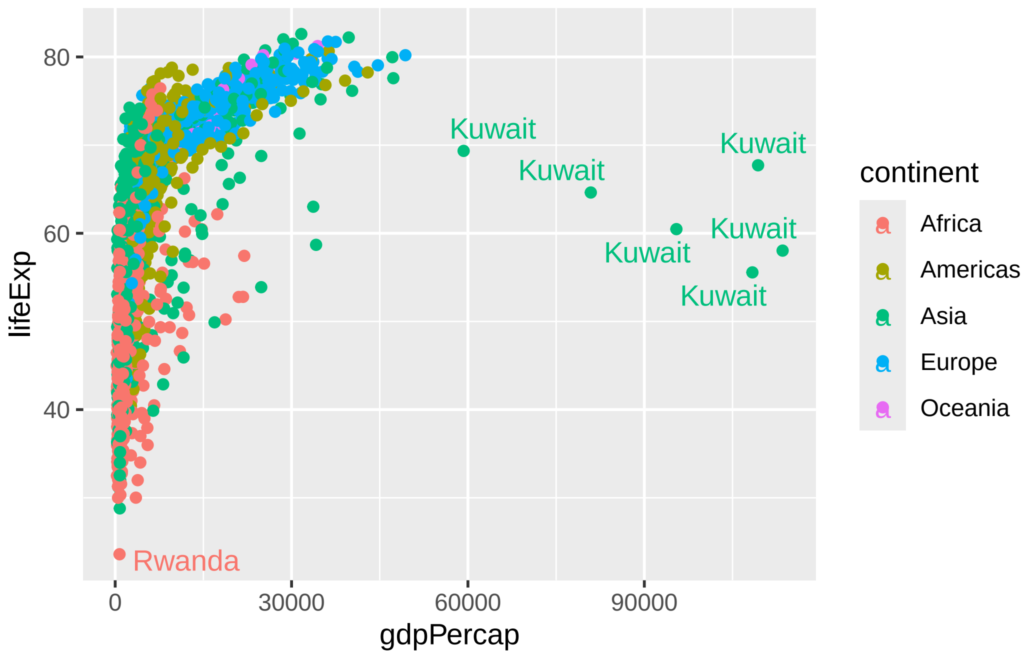
```
ggplot(gapminder) +
  aes(gdpPercap, lifeExp, col=continent, label=country) +
  geom_text()
```



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

```r
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 to seperate pannel per continent

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

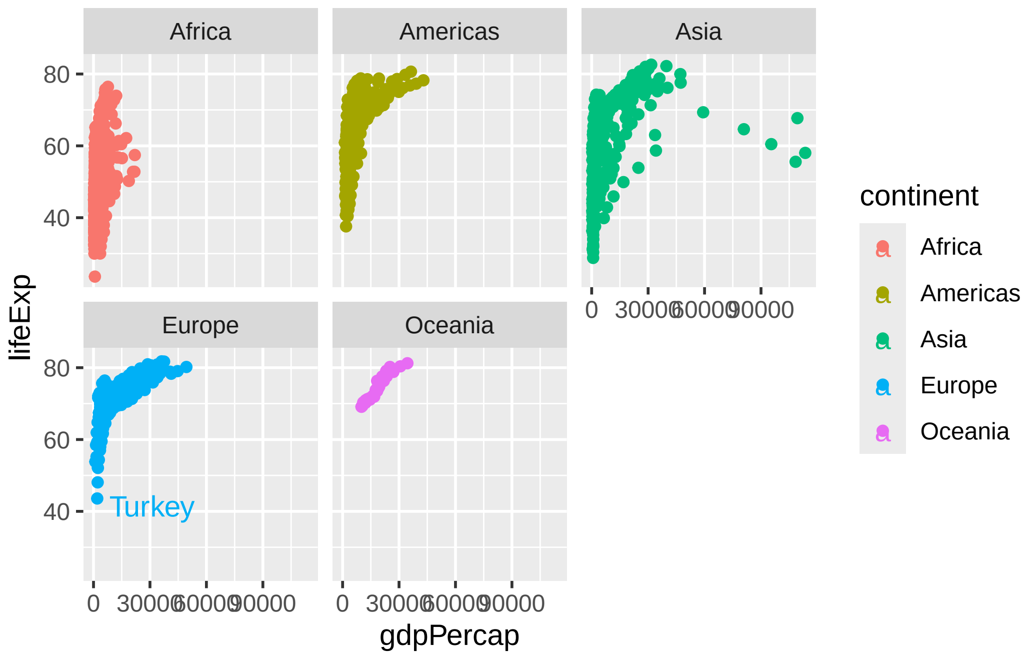
```
Warning: ggrepel: 624 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```

```
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

ggplot2 offers several main advantages over base R plotting:

1. Layered grammar: ggplot2 builds plots by adding layers (data, aesthetics, geoms, themes) in a consistent, stepwise fashion, making complex plots easier to construct and modify[1], [2], [3], [5], [4].
2. Declarative syntax: You specify what you want to see (mapping data to aesthetics), rather than how to draw each element, which simplifies code and improves readability[1], [2], [3], [5], [4].

3. Publication-quality output: ggplot2 produces attractive, professional figures by default, with sensible color schemes, legends, and layouts[1], [2], [3].
4. Customization: It is easier to customize and extend plots (e.g., adding color, labels, themes) compared to base R, which often requires more manual tweaking[1], [2], [3], [5].
5. Consistency: The same grammar applies to all plot types, reducing the need to learn different functions for each plot style[1], [2], [3], [5].