Class 05: Data Visualization with GGPlot

Dani Weatherwax (PID:A17856408)

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

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

- So called "base" R -and add on packages like **ggplot**

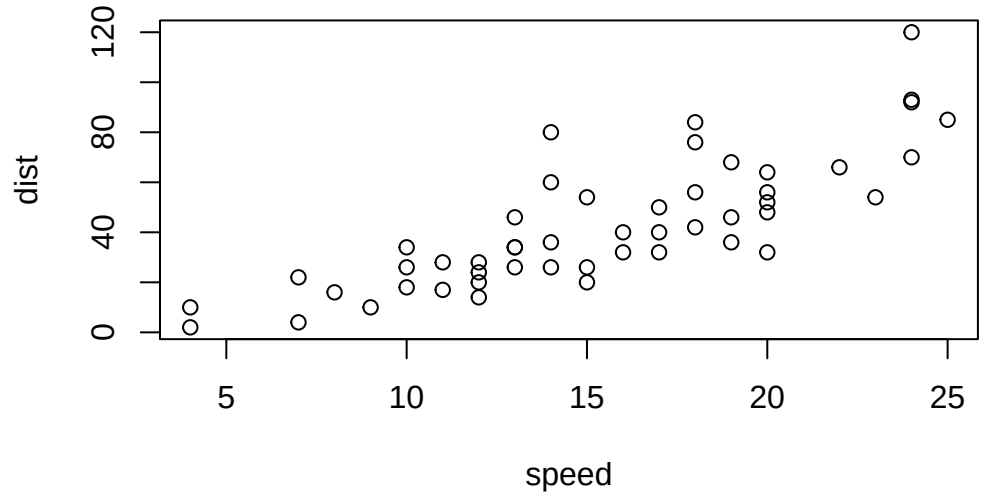
Here is a simple "base" R plot

```
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)
```

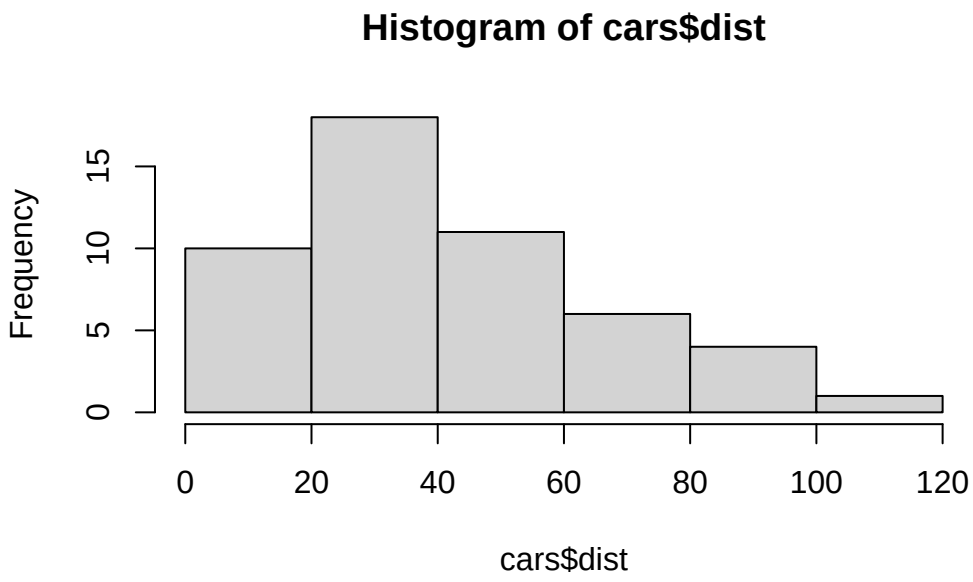


Key-point Base R is quick but not so nice looking in some folks' eyes.

```
head(cars)
```

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

```
hist(cars$dist, )
```



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

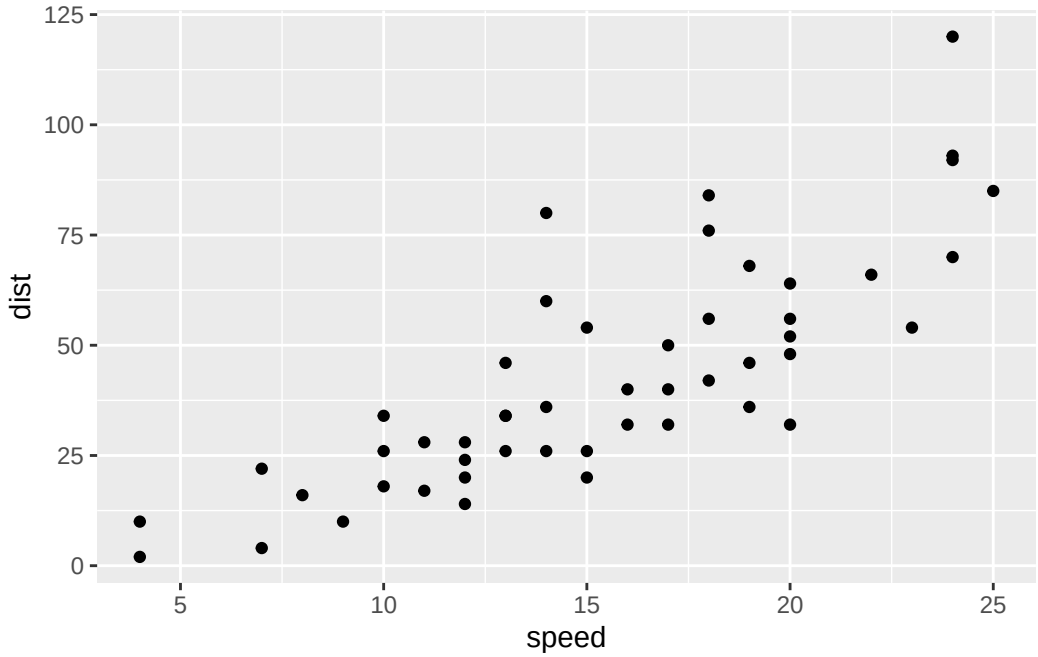
1st I need to install this add-on package. For this we use the `install.packages()` function **in the console, not our report**

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

Every ggplot is composed of at least 3 layers:

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

```
library(ggplot2)
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point()
```



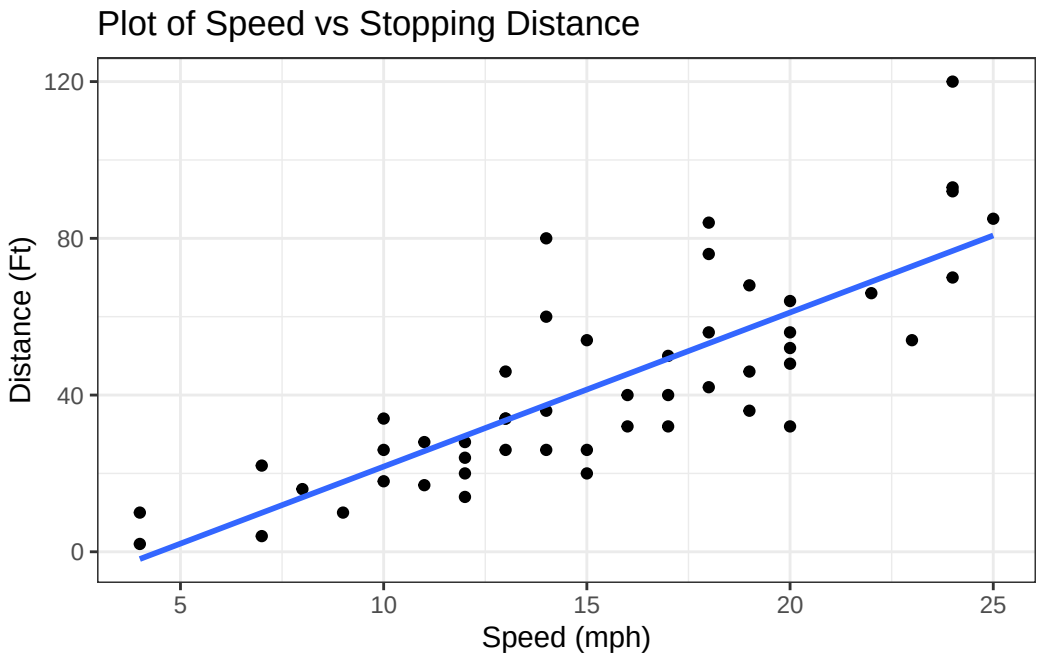
For simple "canned" graphs base R is quicker and more concise but as things get more custom and elaborate then ggplot wins out…

Let's add more layers to our ggplot (a line showing relationship between x and y)

Add a title Add custom axis labels "Speed (mph)" and "Distance (Feet)"

```
library(ggplot2)
ggplot(cars) +
  aes(x=speed, y=dist) +
  geom_point() +
geom_smooth(method="lm", se=FALSE) +
labs(title = "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 dataset?

```
nrow(genes)
```

```
[1] 5196
```

```
ncol(genes)
```

```
[1] 4
```

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

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

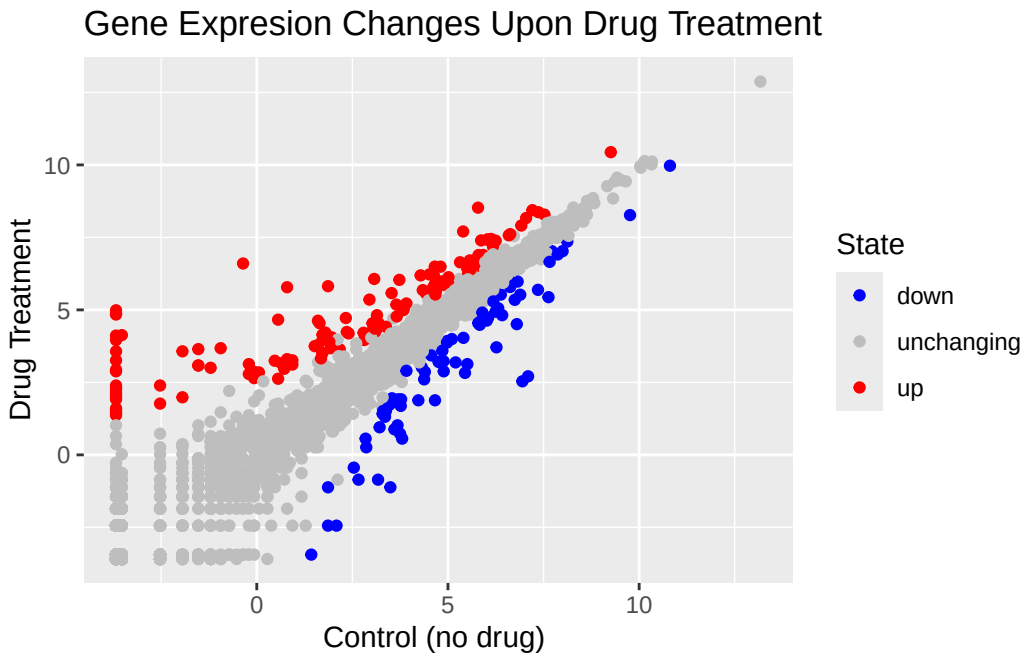
```
[1] 127
```

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

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

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

```
p <- ggplot(genes) +
  aes(x= Condition1, y= Condition2, col=State) +
  geom_point()
p + scale_colour_manual(values=c("blue","gray","red")) +
    labs(title="Gene Expresion Changes Upon Drug Treatment",
         x="Control (no drug) ",
         y="Drug Treatment")
```



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 peek at the data:

```
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
```

How many countries in this dataset?

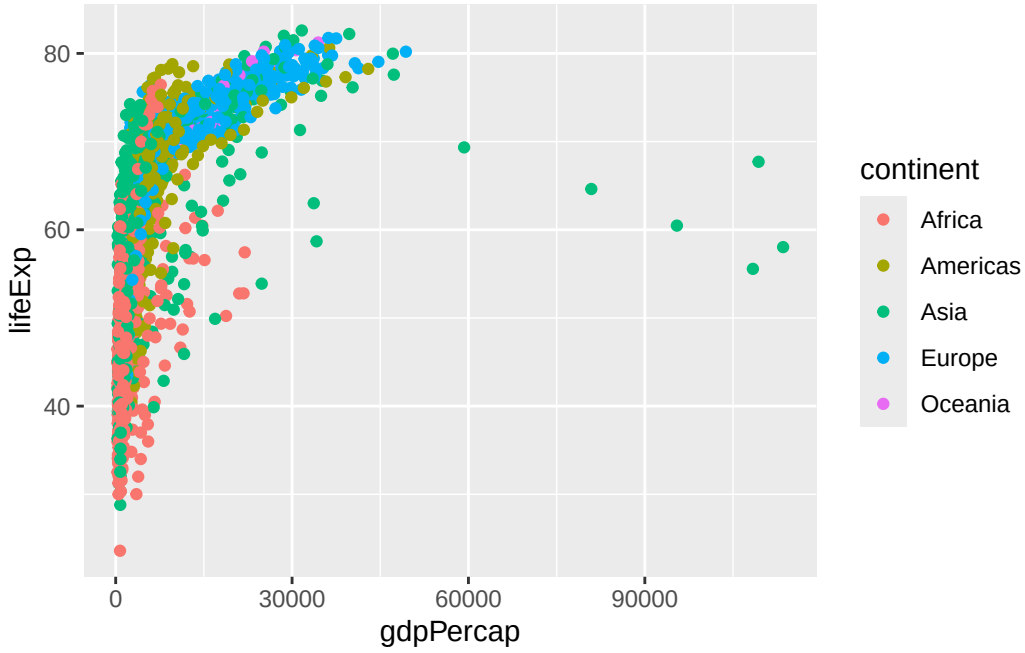
```
length(table(gapminder$country))
```

```
[1] 142
```

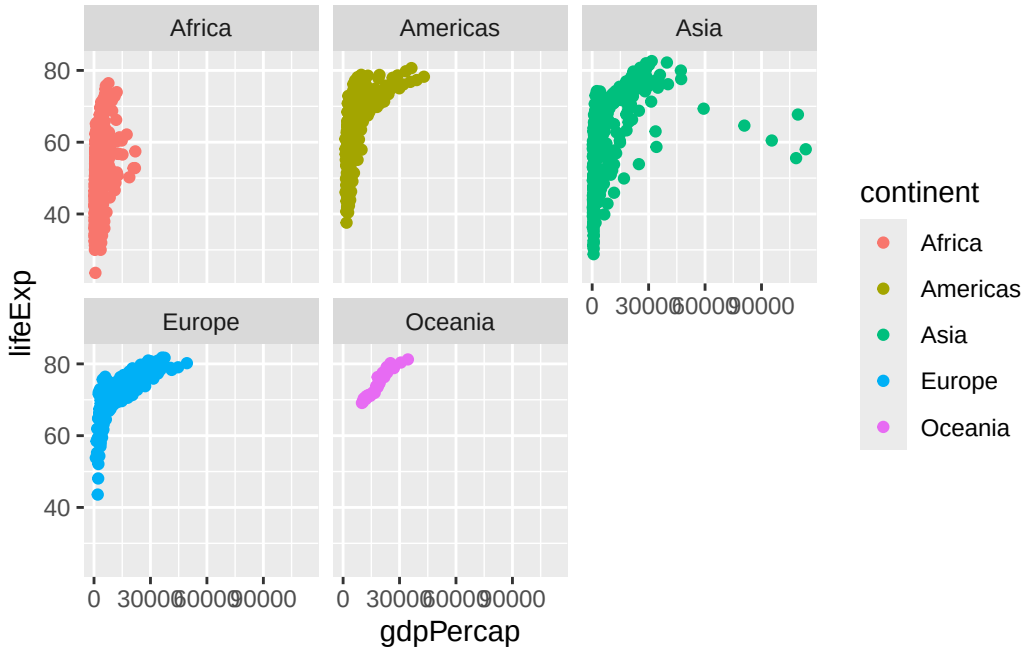
```
unique(gapminder$continent)
```

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

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




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



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

I want a separate panel per continent.

ggplot2 offers several advantages over base R plotting: Layered Grammar: ggplot2 uses a consistent, layered approach for building plots, making it easier to add or modify elements (data, aesthetics, geoms, themes) step by step

Publication Quality: ggplot2 produces attractive, publication-ready graphics with sensible defaults, reducing the need for manual tweaking

Declarative Syntax: You specify what you want to show, not how to draw it, making code more readable and maintainable

Customization: It is easier to customize and extend plots, especially for complex visualizations, compared to base R which can be fiddly and time-consuming

Data Mapping: ggplot2 directly maps data columns to visual features (aesthetics), streamlining the process for multivariate and grouped data

Reproducibility: ggplot2 code is modular and scriptable, making it simple to reproduce and update figures

Base R is faster for quick, simple plots, but ggplot2 excels for complex, beautiful, and reproducible graphics