

Class 5. Data Visualization

A16138722

Base R graphics vs ggplot2

There are many graphics systems available in R, including so-called “base” R graphics and the very popular **ggplot2** package.

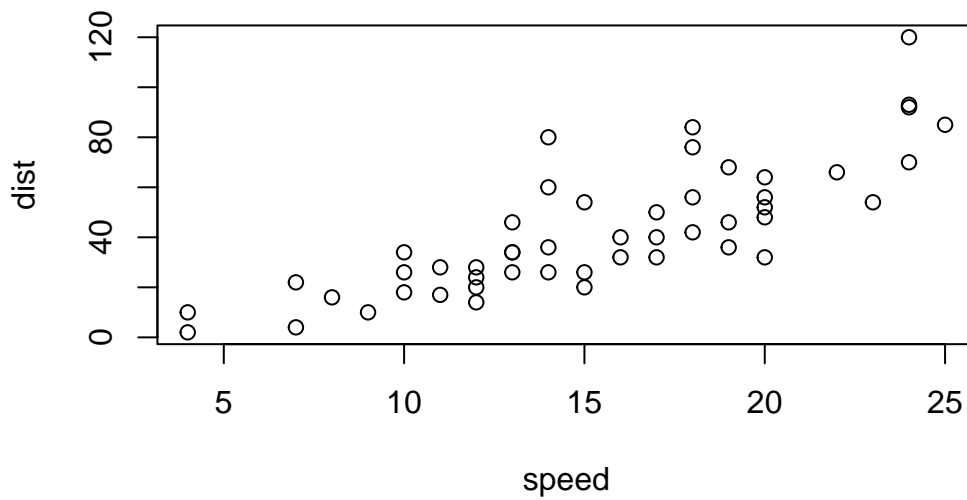
To compare these let's play with the in-built `cars` dataset.

```
head(cars, 3) #returns the top 3 rows of cars dataset
```

	speed	dist
1	4	2
2	4	10
3	7	4

To use ‘base’ R, I can simply call the ‘plot()’ function:

```
plot(cars)
```



To use `ggplot2` package, first I need to install it with the function `install.packages("ggplot2")`.

I will run this in my R console (i.e. the R brain) as I do not want to re-install it every time I render my report.

Any package in R that isn't part of 'base' R, I need to call it everytime I want to use it

The main function in this package is called `ggplot()`. Can I just call it?

```
# No, it would show "Error in ggplot() : could not find function \"ggplot\""

# install.packages("ggplot2")

library(ggplot2)
ggplot()
```



To make a figure with ggplotI need always at least 3 things:

- **the data** (i.e. what I want to plot)
- **aes** (the aesthetic mapping of the data to the plot I want)
- the **geoms** (i.e. how I want to plot the data)

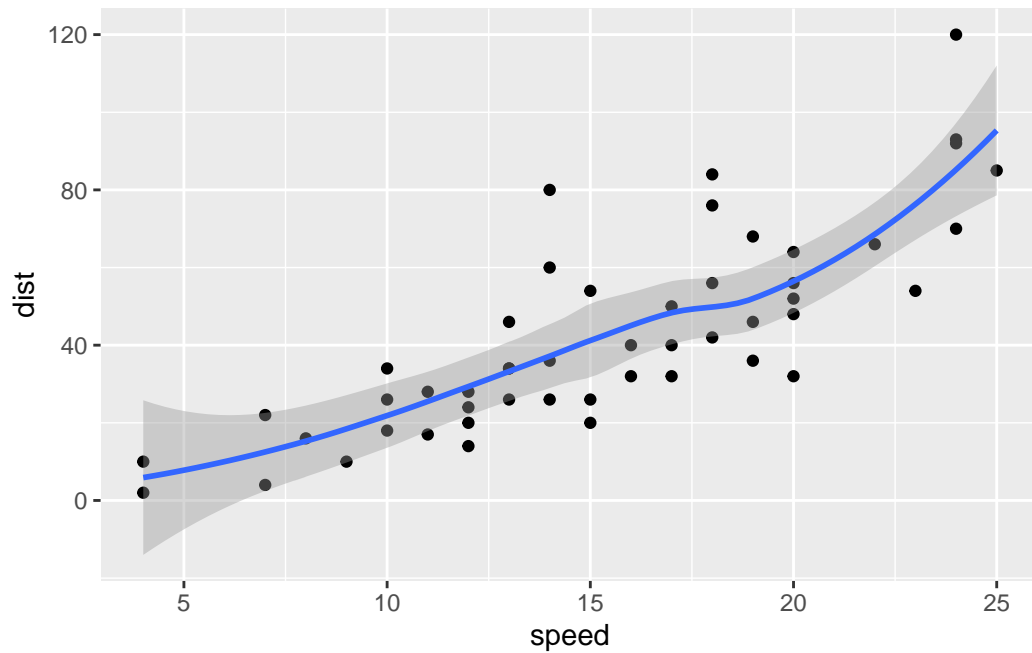
```
ggplot(data = cars) +  
  aes(x=speed, y=dist) +  
  geom_point()
```



If I want to add more stuff, I can just keep adding layers, e.g.

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

`geom_smooth()` using method = 'loess' and formula = 'y ~ x'

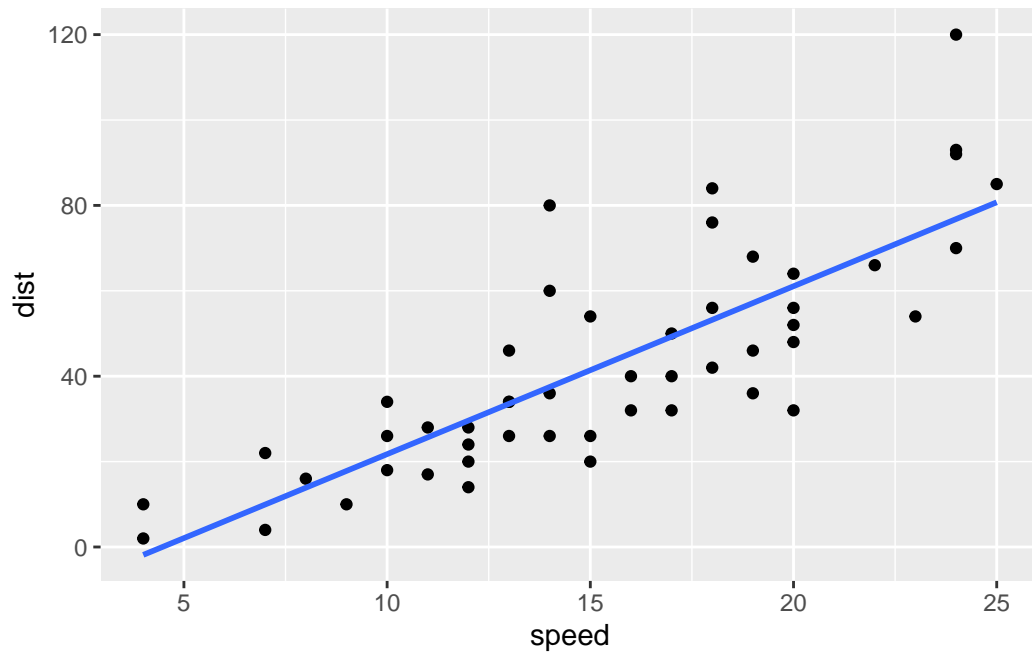


Ggplot is much more verbose than base R plots for standard plots but it has a consistent layer system that I can use to make just about any plot.

Let's make a plot with a straight line fit - i.e. a linear model and no standard error

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

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



```
# help(ggeom_smooth)
```

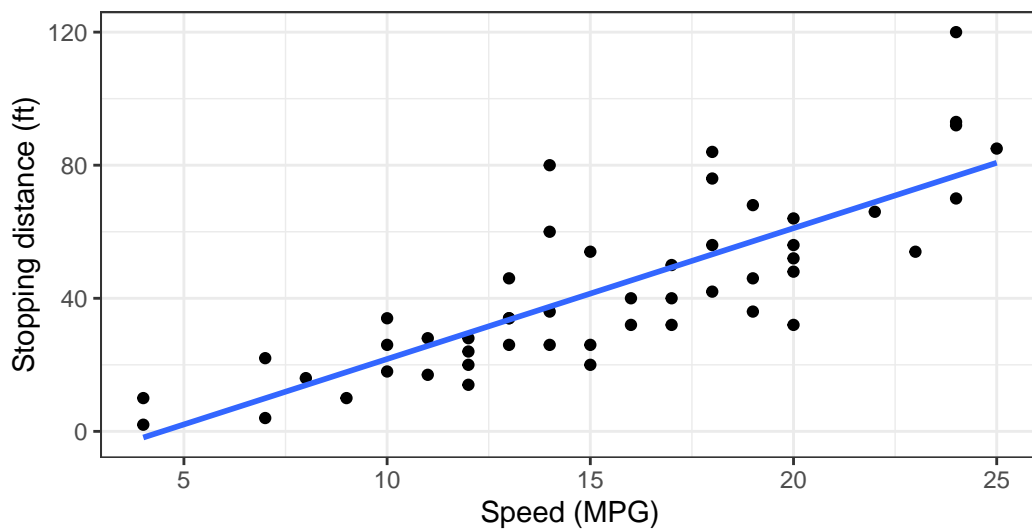
Let's add a title and subtitle

```
ggplot(data = cars) +
  aes(x=speed, y=dist) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE) +
  labs(title="Stopping distance for old cars",
        subtitle="From the inbuilt car dataset",
        caption="BIMM143",
        x="Speed (MPG)", y="Stopping distance (ft)") +
  theme_bw()
```

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

Stopping distance for old cars

From the inbuilt car dataset



BIMM143

A more complicated plot

Let's plot some gene expression data.

The code below reads the results of a differential expression analysis where a new anti-viral drug is being tested.

```
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

Q. How many genes are in this dataset?

```
nrow(genes)
```

```
[1] 5196
```

```
ncol(genes)
```

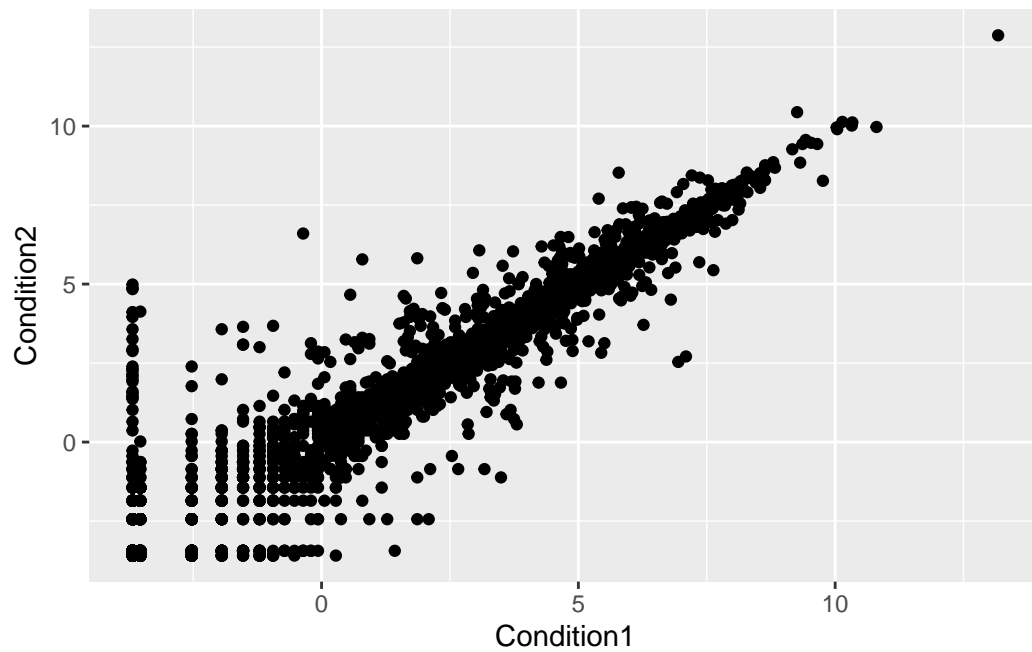
```
[1] 4
```

Q. How can we summarize that last column - the “State” column?

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

down	unchanging	up
72	4997	127

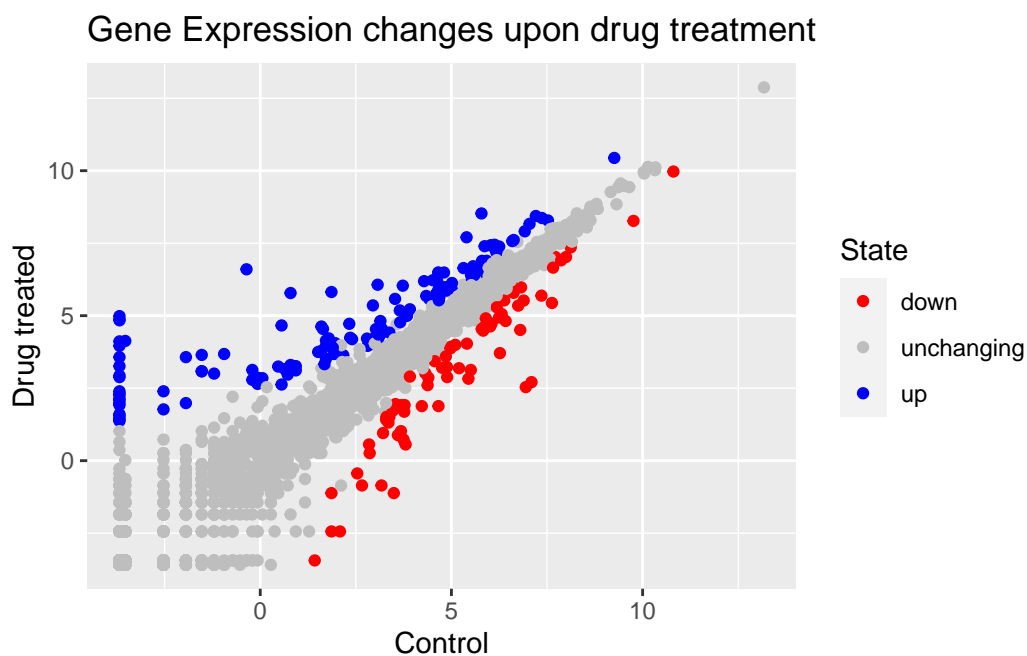
```
ggplot(genes) + #sets up the canvas  
aes(x=Condition1, y=Condition2) +  
geom_point()
```




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

I can now just call p when I want to plot or add to it.

```
p + labs(title="Gene Expression changes upon drug treatment",
         x="Control", y="Drug treated") +
  scale_colour_manual( values=c("red","gray","blue") )
```



Going Further

Where I read a slightly larger dataset

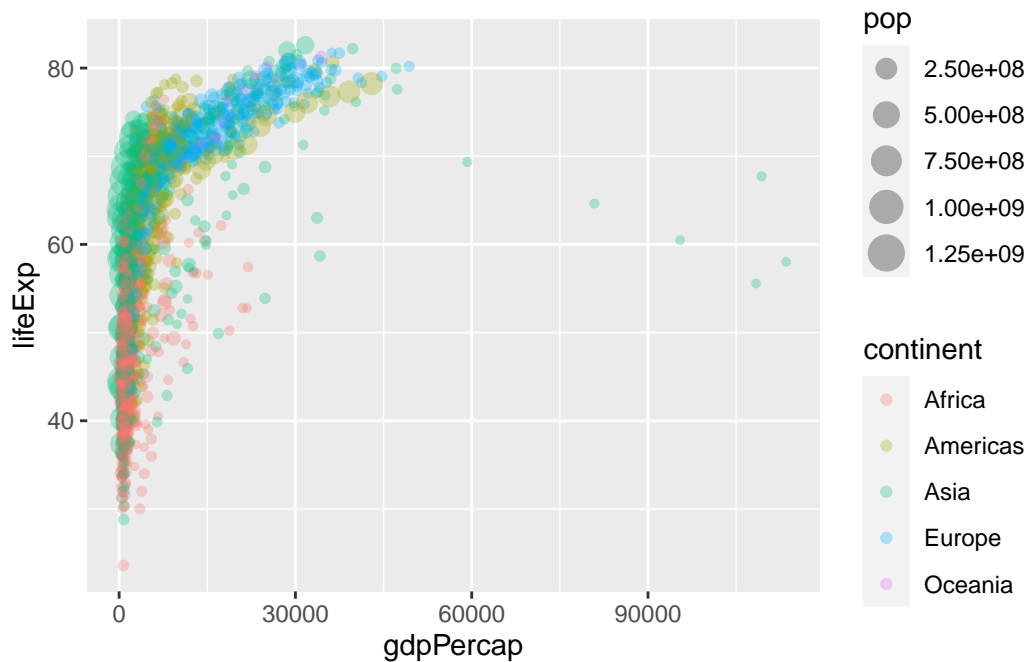
```
# file location online
url <- "https://raw.githubusercontent.com/jennybc/gapminder/master/inst/extdata/gapminder."

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

```
country continent year lifeExp      pop gdpPercap
```

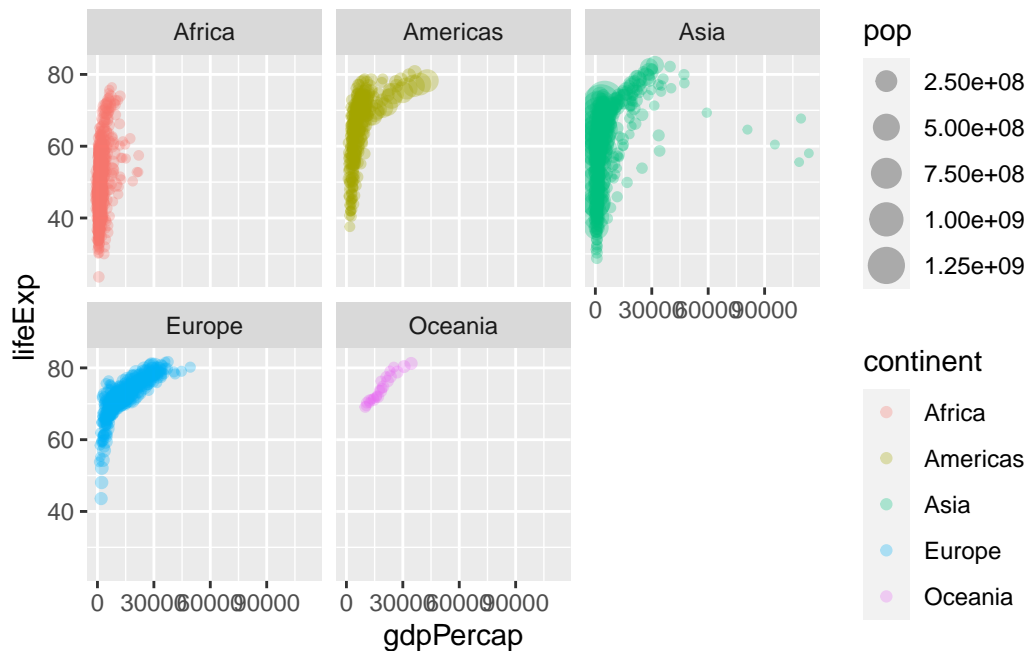
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	Asia	1957	30.332	9240934	820.8530
3	Afghanistan	Asia	1962	31.997	10267083	853.1007
4	Afghanistan	Asia	1967	34.020	11537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811
6	Afghanistan	Asia	1977	38.438	14880372	786.1134

```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, color=continent, size=pop) +
  geom_point(alpha=0.3) #'alpha' will make the points more transparent
```



A very useful layer to add sometimes is for “facetting”

```
ggplot(gapminder) +
  aes(x=gdpPercap, y=lifeExp, color=continent, size=pop) +
  geom_point(alpha=0.3) +
  facet_wrap(~continent) #separate the data by continent
```



Bar plot

```
#install.packages("dplyr")
library(dplyr)
```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

filter, lag

The following objects are masked from 'package:base':

intersect, setdiff, setequal, union

```
gapminder_top5 <- gapminder %>%
  filter(year==2007) %>%
  arrange(desc(pop)) %>%
  top_n(5, pop)
```

```
gapminder_top5
```

	country	continent	year	lifeExp	pop	gdpPercap
1	China	Asia	2007	72.961	1318683096	4959.115
2	India	Asia	2007	64.698	1110396331	2452.210
3	United States	Americas	2007	78.242	301139947	42951.653
4	Indonesia	Asia	2007	70.650	223547000	3540.652
5	Brazil	Americas	2007	72.390	190010647	9065.801

```
ggplot(gapminder_top5) +  
  geom_col(aes(x = country, y = pop))
```

