Graphs and Data Visualisation in R

Advanced Psychological Research Methods

Dr Christopher Wilson

Questions from last week’s session?

Submit your attendance

Attendance code: 26004

http://bit.ly/APRM22

By the end of this section, you will be able to:

- Describe the ggplot “grammar of visualisation”: coordinates and geoms
- Write a graph function to display multiple variables on a plot
- Amend the titles and legends of a plot
- Save plots in PDF or image formats

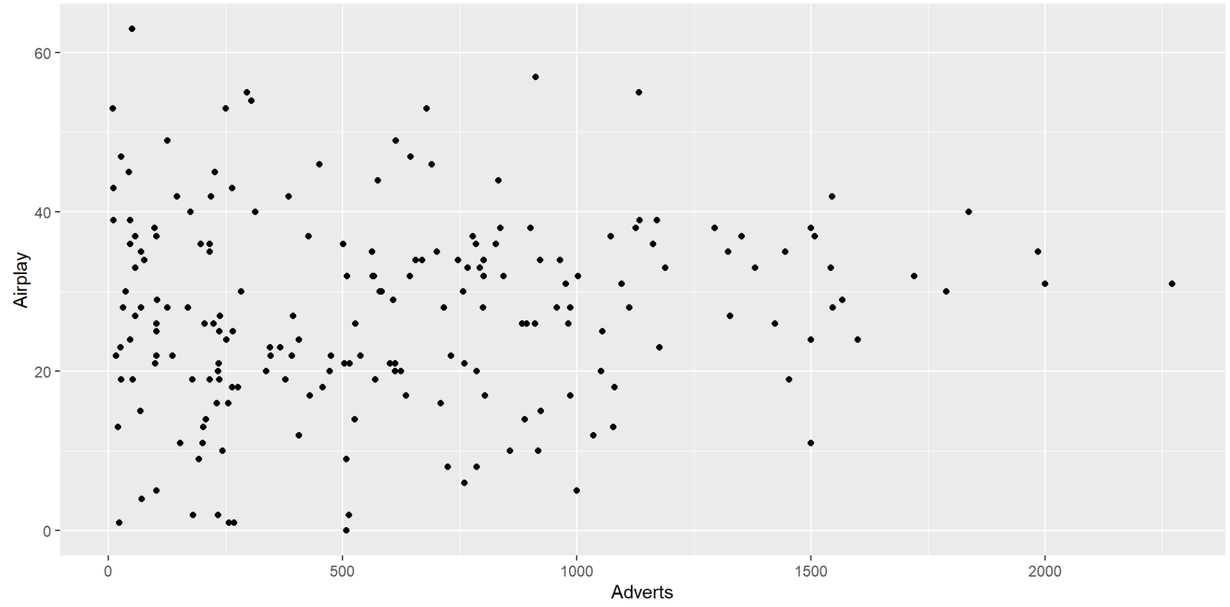
We will use the ggplot package for making graphs

The “grammar of visualisation”

- Graphs are made up of 3 components:
  - A dataset
  - A coordinate system
  - Visual marks to represent data (**geoms**)

The “grammar of visualisation” #2

	Adverts	Sales	Airplay	Attract	Genre
1	10.256	330	43	10	Country
2	985.685	120	28	7	Pop
3	1445.563	360	35	7	HipHop
4	1188.193	270	33	7	HipHop
5	574.513	220	44	5	Metal
6	568.954	170	19	5	Country



- In the above example, the dataset is the **album_sales** that we used previously.
- The *adverts* variable is mapped to the X axis
- The *airplay* variable is mapped to the Y axis

How to code a graph

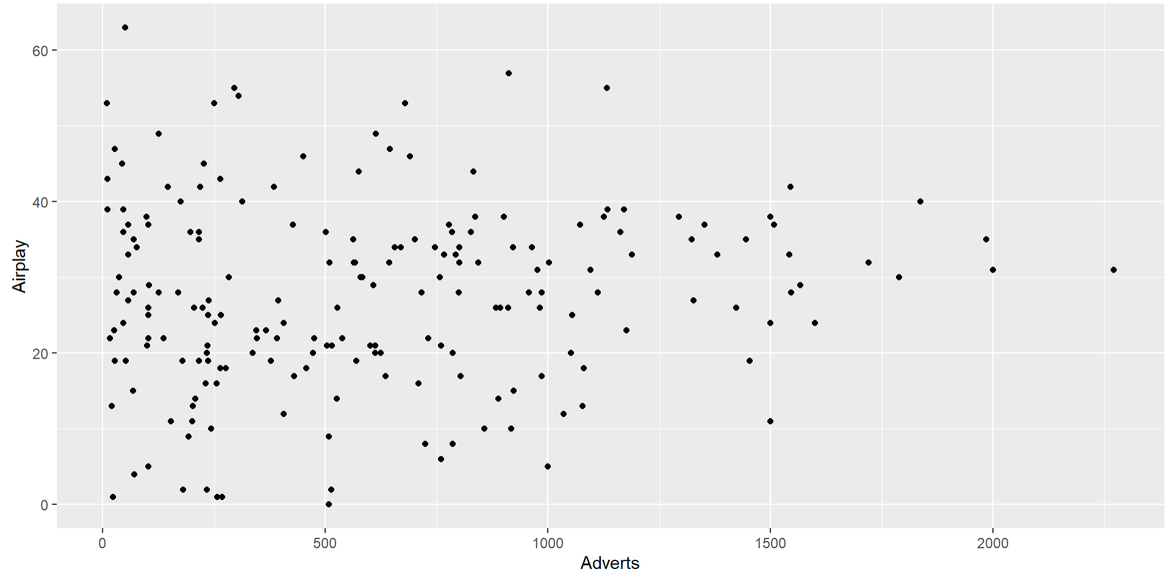
- The graph is created using the following code:

```
1  ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point()
```

- In this code, we specify the dataset, the variables for the X and Y axes and the **geom** that will represent the data points visually (in this case, each datum is a point)

The graph output

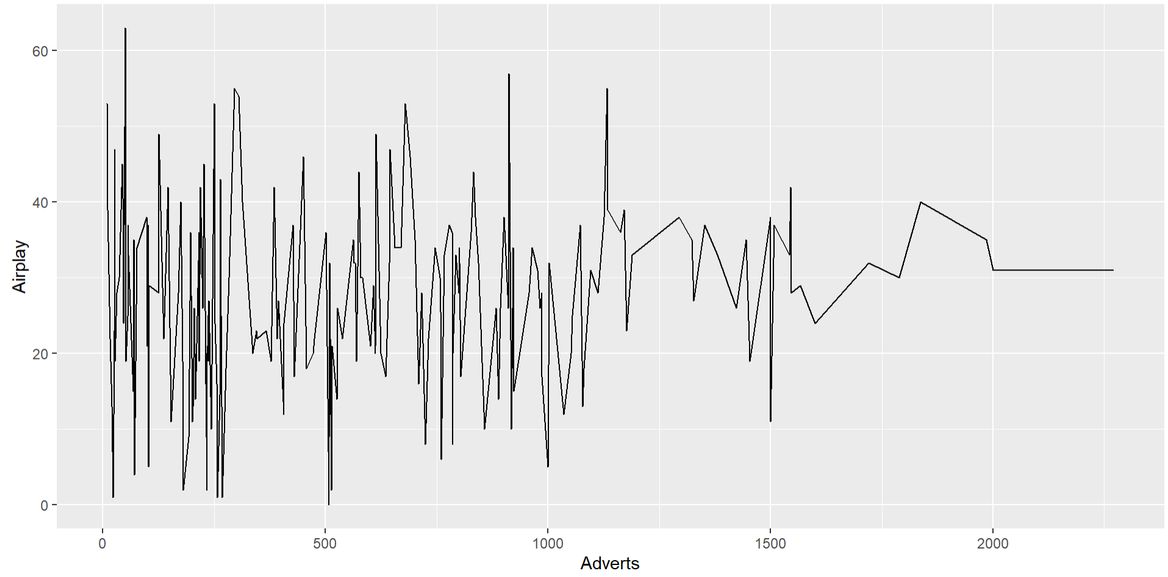
```
1 library(ggplot2)
2
3 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point()
```



Changing the geoms leads to different visualisations

- If we change from points to lines, for example we get a different plot:

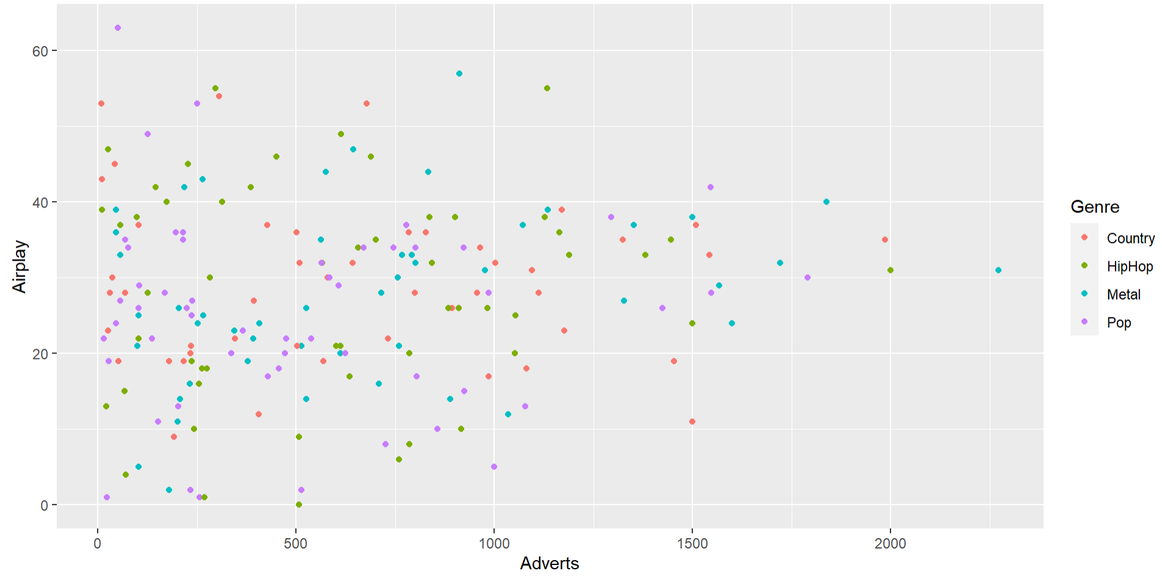
```
1 library(ggplot2)
2
3 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_line()
```



It is possible to represent more variables on the plot

- By specifying that colours of our points should be attached to the **Genre** variable, the data is now colour-coded

```
1 library(ggplot2)
2
3 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point(aes(color = Genre))
```



It is possible to represent more variables on the plot #2

- By specifying that size of our points should be attached to the **Attract** variable, the size of the points adjusts

```
1 library(ggplot2)
2
3 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point(aes(color = Genre, size = Att
```

It is possible to represent more variables on the plot #3

- By specifying that shape of our points should be attached to the **Genre** variable, the shape of the points changes accordingly

```
1 library(ggplot2)
2
3 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point(aes(color = Genre, size = Att
```

Something a little more useful…

- What if we wanted to make a bar chart of the average sales of different genres?

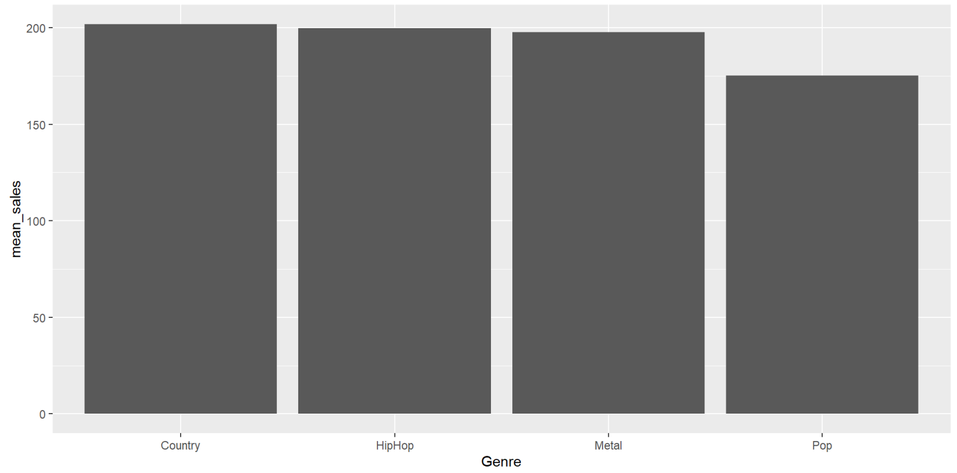
Let's view the data:

```
1 head(album_sales)
```

```
   Adverts Sales Airplay Attract   Genre
1   10.256   330      43      10 Country
2  985.685   120      28       7     Pop
3 1445.563   360      35       7  HipHop
4 1188.193   270      33       7  HipHop
5  574.513   220      44       5   Metal
6  568.954   170      19       5 Country
```

Bar chart of means

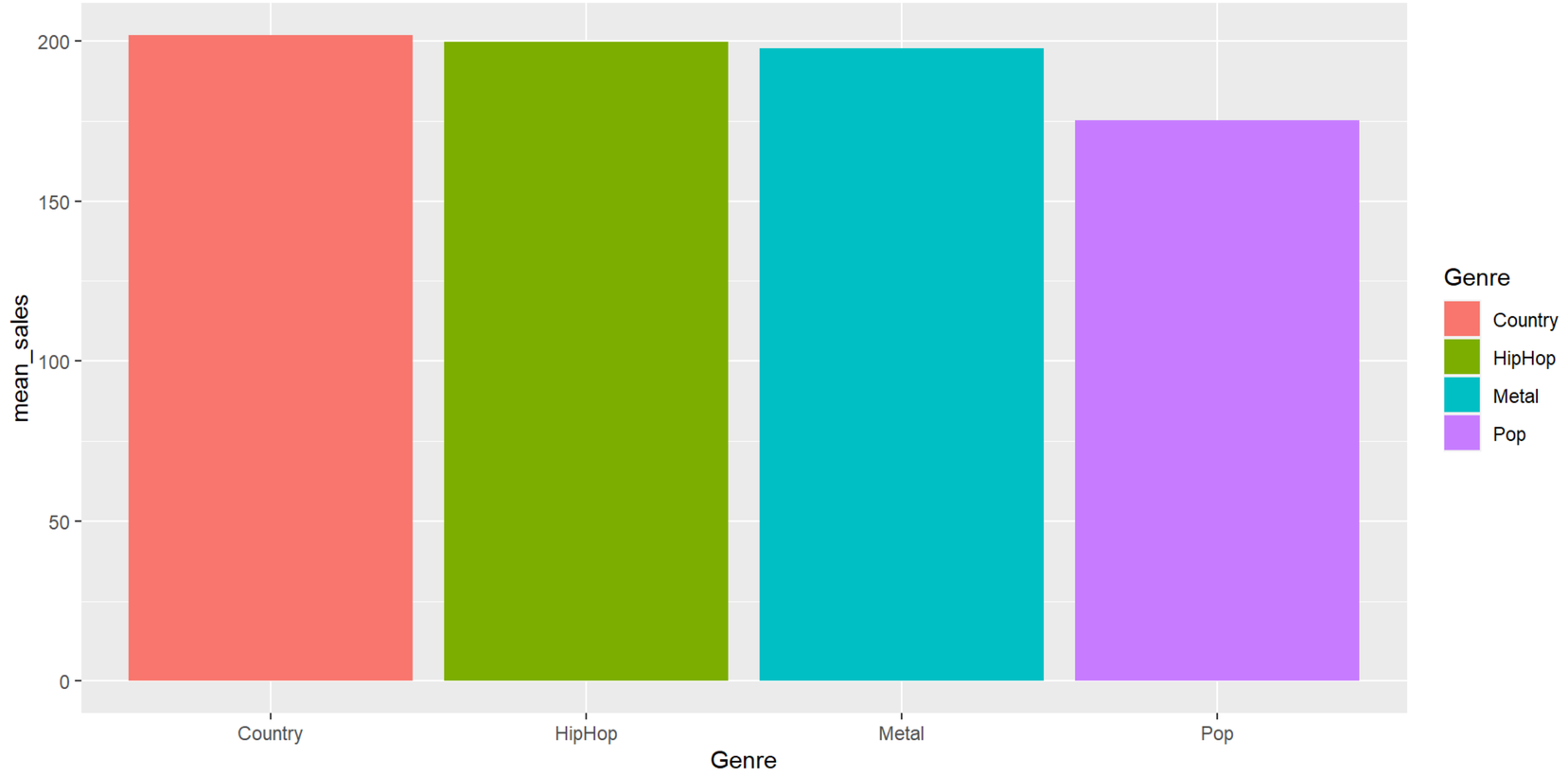
```r
library(tidyverse) # contains ggplot and the summarise function used below

# Changing Genre to a factor
album_sales$Genre <- as.factor(album_sales$Genre)

# create a summary of the mean sales for each genre

meanSales <- album_sales %>%
  group_by(Genre) %>%
  summarise(mean_sales = mean(Sales))

# use the summary data for the graph
```



Changing colours

- We can change colours by modifying the geom
- Some objects use “fill”, some use “colour”

```
1  ggplot(data = meanSales, aes(x = Genre, y = mean_sales)) + geom_col(aes(fill = Genre))
```

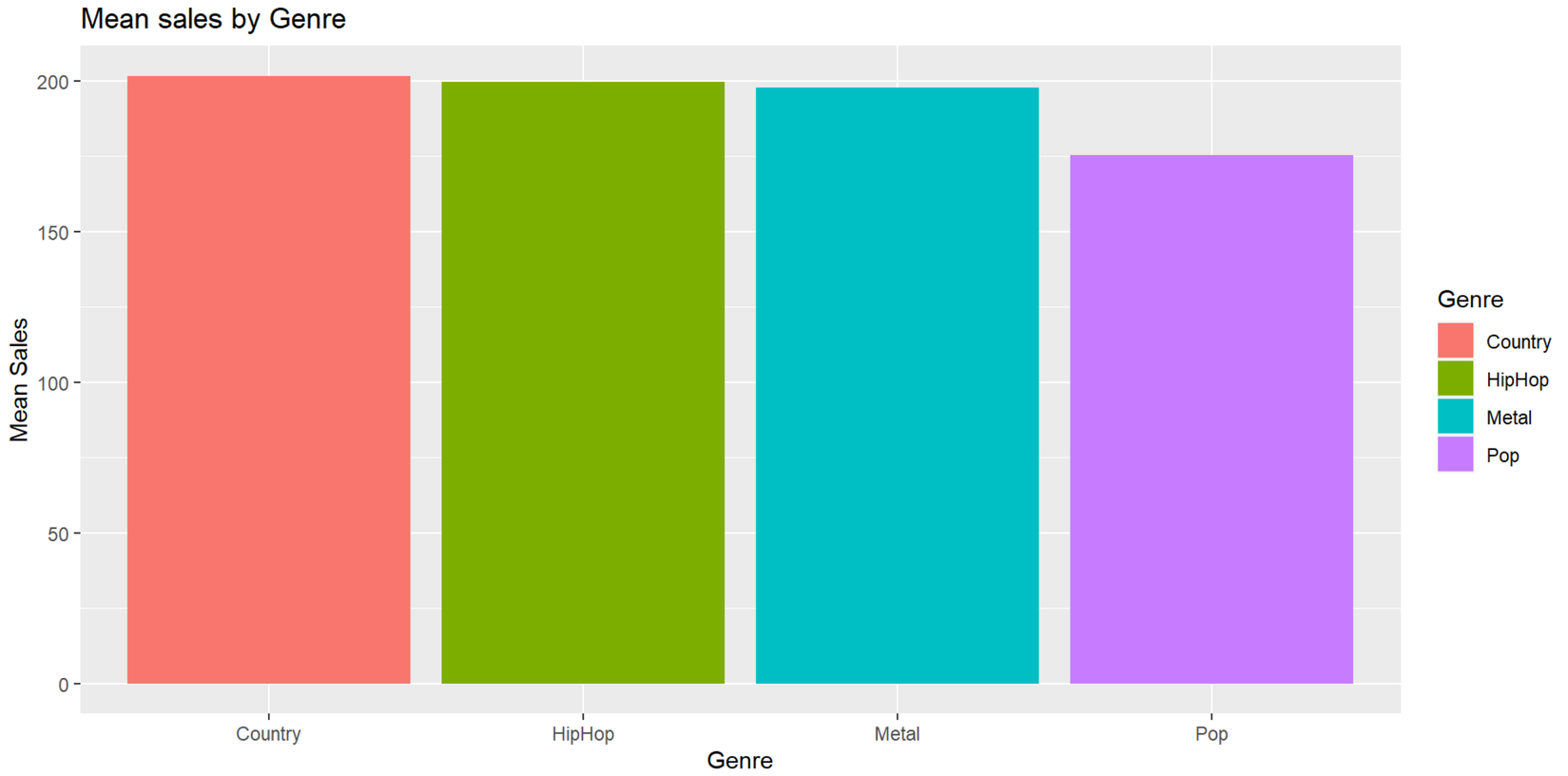


Changing the axis labels and title on a plot

We can change the axis labels and title using the **labs()** command:

labs(x="Genre", y="Mean Sales", title = "Mean sales by Genre")

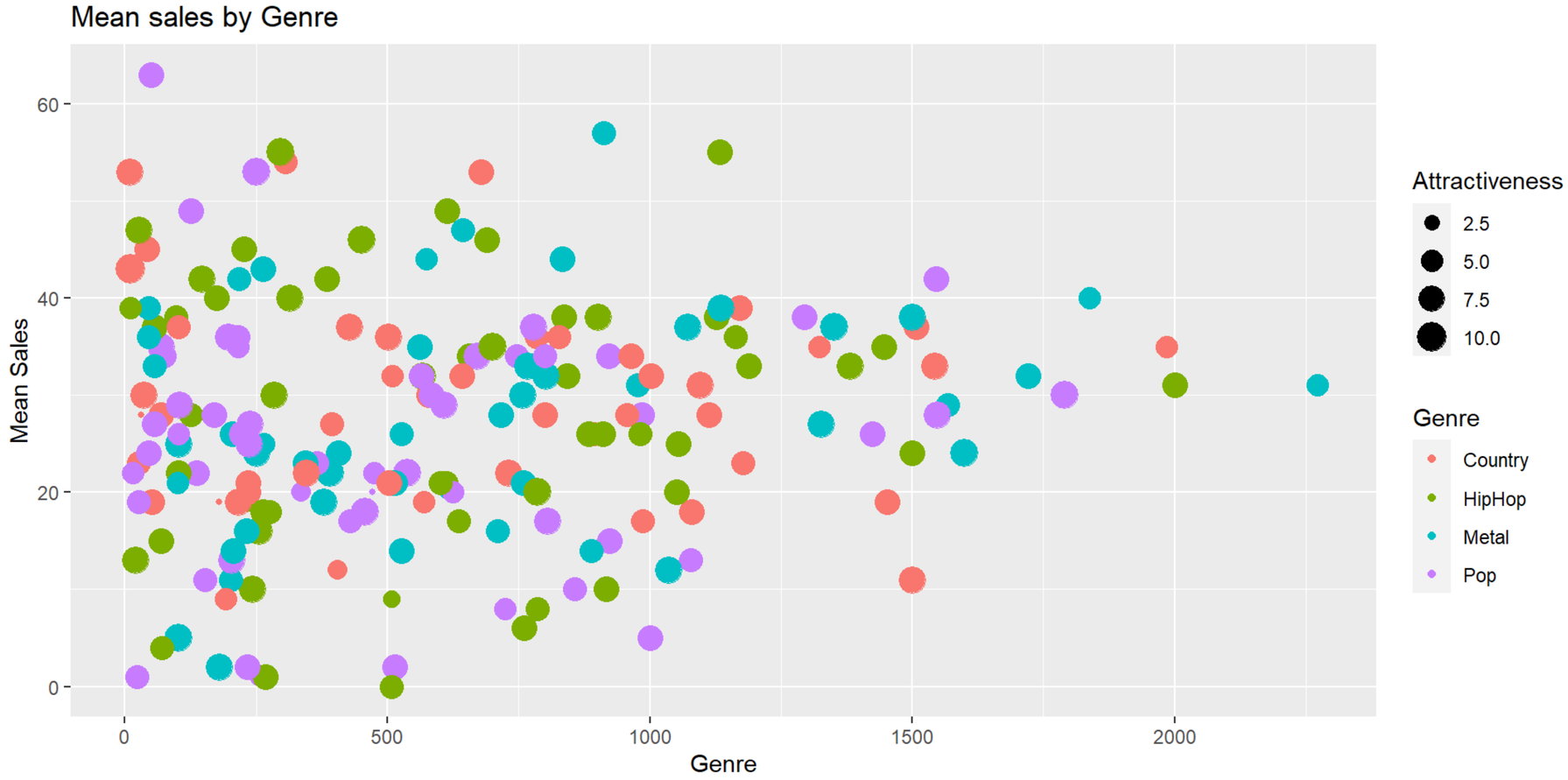
```
1  ggplot(data = meanSales, aes(x = Genre, y = mean_sales)) + geom_col(aes(fill = Genre)) +
2    labs(x="Genre", y="Mean Sales", title = "Mean sales by Genre")
```



Changing the legend on a plot

To change the legend, we use the **labs()** command too, and reference the relevant property (e.g. size, shape, colour)

```
1 ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point(aes(color = Genre, size = Att
2 labs(x="Genre", y="Mean Sales", title = "Mean sales by Genre", size="Attractiveness")
```



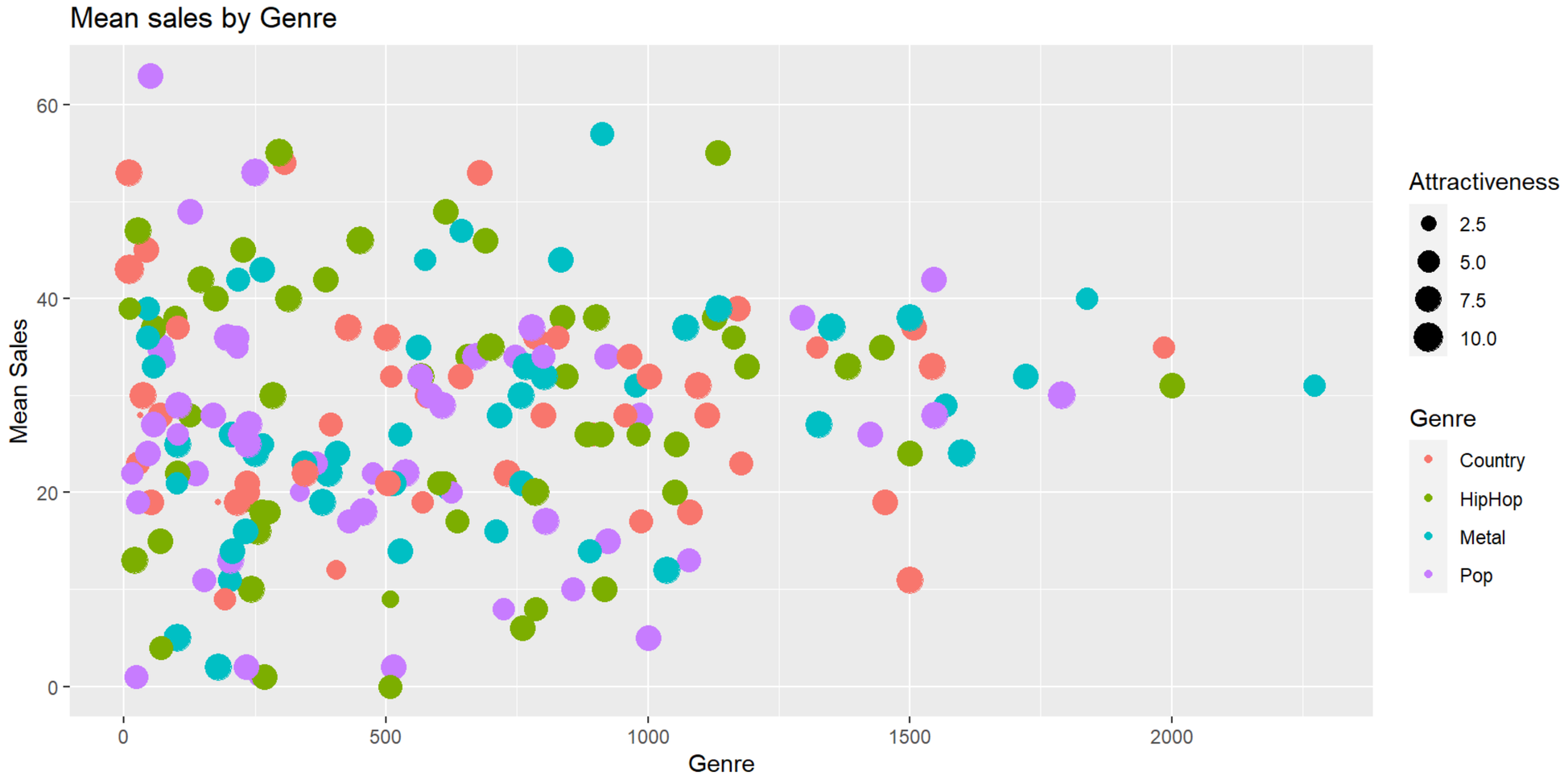
Storing plots to be recalled later

- Plots can be assigned to objects in R and recalled later, just like any other piece of data

```
library(ggplot2)

## Create plot and store it as "myPlot" object

myPlot <- ggplot(data=album_sales, aes(x=Adverts,y=Airplay)) + geom_point(aes(color = Genre,
labs(x="Genre", y="Mean Sales", title = "Mean sales by Genre", size="Attractiveness")
```

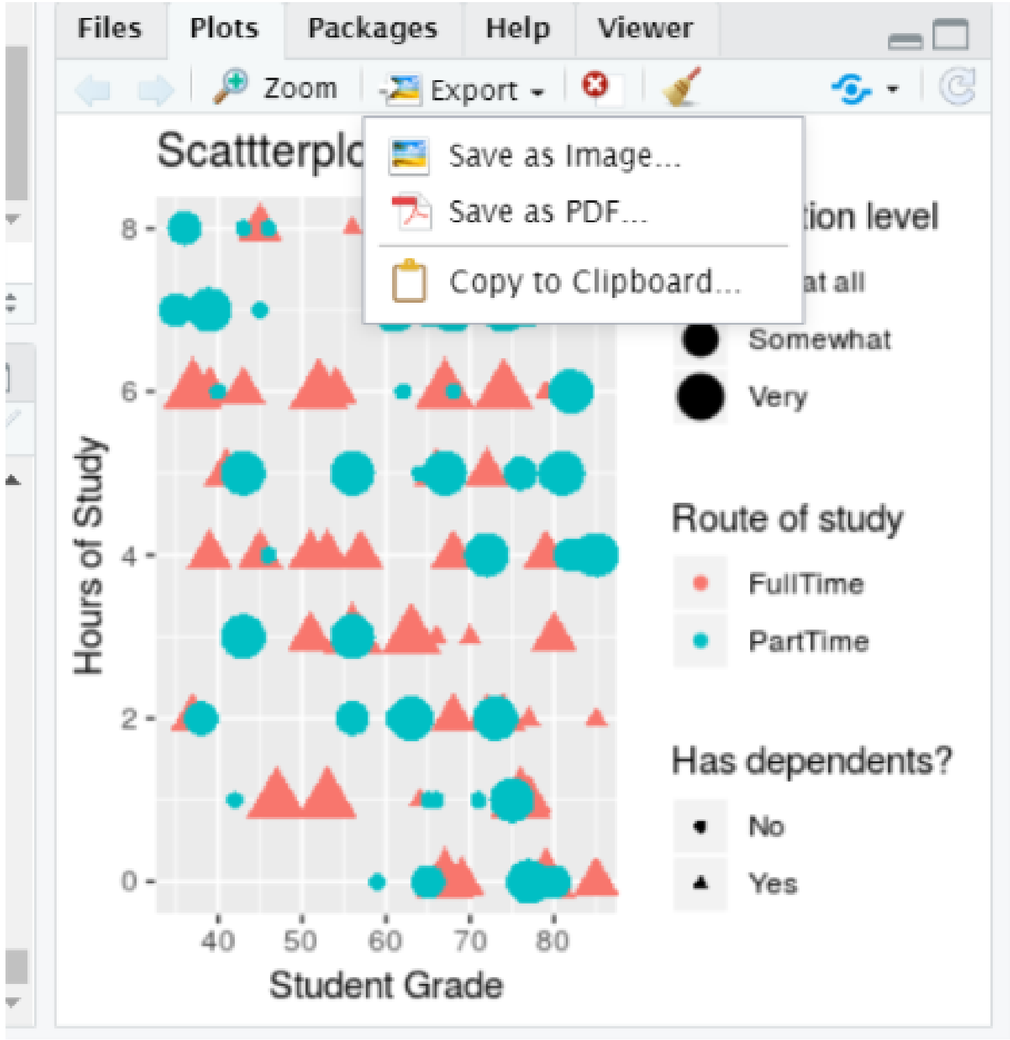
Recalling a stored plot

```
1   #Recall myPlot
2 myPlot
```



Saving plots # 1

- Plots can be save using the **export** button in the plots tab



Plots can also be saved using code

- You might want to include code to save your plot in a script, for example
- This can allow greater control over the output file and plot dimensions:

```
ggsave(plot= myPlot, file="myPlot.pdf", width = 4, height = 4)
ggsave(plot= myPlot, file="myPlot.png", width = 4, height = 4, units="cm", dpi=320)
```

Questions?