LAB100

Week 10: Creating Plots

Your aim for this workshop is to learn how to create and customise plots. Upon completing this workshop you should be able to:

- Use the `plot` command
- Understand the main plotting arguments
- Add points and lines to an existing plot
- Export images from RStudio

Remember that you need to change the working directory and open a new R script at the start of each workshop.

1 Creating a Plot

To illustrate how to create a plot in RStudio we will first be considering the following set of points.

$$(0.00, 1.00),\quad (0.59, -0.81),\quad (-0.95, 0.31),\quad (0.95, 0.31),\quad (-0.59, -0.81)\ \text{ and }\ (0.00, 1.00)$$

The first step is to create two vectors, one containing all of the x direction co-ordinates and another containing the y direction co-ordinates. Note that the n^{th} element in each vector must describe the same point. To create a plot we use the `plot` command, which requires the two co-ordinate vectors as input. Type the following into your editor and execute them to the console.

```
x <- c(0.00, 0.59, -0.95, 0.95, -0.59, 0.00)
y <- c(1.00, -0.81, 0.31, 0.31, -0.81, 1.00)
plot(x, y)
```

In the bottom right panel, click on the 'Plots' tab to see the plot you have just made. This should be a image of five points arranged in a circle, with a sixth point overlayed on the first point because it has the same co-ordinates. The `plot` command accepts many input arguments such that you can change how the plot appears. One input option is `type`, which controls how RStudio plots the points. For example, the next command redraws the plot and joins consecutive points (according to the order in the vectors `x` and `y`) with a straight solid line.

```
plot(x, y, type="l")
```

Other values for `type` is presented in the table below.

type=	Name	Meaning
"p"	Points	Plots the points as points (default if unspecified)
"l"	Lines	Joins consecutive points with a straight line
"o"	Overplotted	Does points and lines
"n"	None	Creates a blank plotting frame

2 Plotting Functions

We can also use the `plot` command to create a plot of any function by setting `type="l"`. Similarly as before, the two x and y vectors must first be created. However, instead of defining specific points, we can ask RStudio to create a sequence of points for the x co-ordinates and evaluate the y co-ordinates according to the function of interest. Type the following into your editor and execute to plot the quadratic function $f(x) = x^2 - x - 2$ for the range $-5 \leq x \leq 5$.

```
x <- seq(-5, 5, by=0.1)
y <- x^2 - x - 2
plot(x, y, type="l")
grid()
```

Be careful when using the `seq` command to create the `x` vector for plotting because you need to make sure that you have specified enough points in order to create a smooth curve. Recall from the previous section that the `type="l"` argument in `plot` connects successive points within the x and y vectors with a *straight line*. To give the appearance of a curved function we require many points to be evaluated. To illustrate, consider the next block of code that plots $f(x) = \cos(x)$ for the range $-2\pi \leq x \leq 2\pi$.

```
x <- seq(-2*pi, 2*pi, by=1)
y <- cos(x)
plot(x, y, type="l")
grid()
```

This plot appears very disjoint because the function has only been evaluated at 13 points, which is too few to get a smooth image. To improve the appearance of the plot we increase the length of `x` by decreasing the spacing between successive points:

```
x <- seq(-2*pi, 2*pi, by=0.1)
y <- cos(x)
plot(x, y, type="l")
grid()
```

3 Plot Formatting

The `plot` command can accept many of arguments such that you can customise your plots. The table below describes some of the main plotting options that will be useful for you.

Argument	Meaning	Example
`main`	Adds a title to the plot.	`main="My Plot"`
`xlab`, `ylab`	Changes the x- and y-axes labels.	`ylab="f(x)"`
`xlim`, `ylim`	Alters the x and y co-ordinates ranges and requires the upper and lower bounds.	`xlim=c(-5, 5)`
`pch`	Changes the point symbol.	`pch=4`
`col`	Changes the colour of the points or lines.	`col=4` or `col="blue"`
`lty`	Changes the type of line to be plotted.	`lty=3`

The plot title and axis labels must be entered into the `plot` command using the quotation marks `" "` to prevent RStudio looking into its memory for an object with the name `My Plot`. The text within the quotation marks is called a string of characters and can be a mixture of letters, numbers and punctuation.

Type the following example into your editor and execute to create a plot that illustrates some of the colours and point types that are available.

```
x <- rep(1:8, times=8)
y <- rep(1:8, each=8)
plot(x, y, pch=x, col=y, xlim=c(0, 9), ylim=c(0, 9), main="Plot Formatting",
  xlab="pch Number", ylab="col Number")
```

4 Adding Points and Lines

A single curve or a single set of points may not be sufficient for the image that you wish to create. For example, create a plot of $f(x) = x^5 + 3x^3 - 4x$ for the range $-2 \leq x \leq 2$ and indicate the where the turning points are and also draw a tangent to the curve at $x = 0$. The first step is to create a plot of the function of interest. Copy and execute the following set of commands from your editor.

```
x <- seq(-2, 2, by=0.1)
y <- x^5 + 3*x^3 - 4*x
plot(x, y, type="l", main="x^5 + 3*x^3 - 4*x", ylim=c(-6,6), xlim=c(-1.5,1.5))
grid()
```

The turning points for this function are at $(-0.61, 1.67)$ and $(0.61, -1.67)$, correct to 2 decimal places. To add these points to the plot we use the `points` command, which requires a vector of x co-ordinates, a vector of y co-ordinates and any other formatting arguments. The next set of code illustrates how to use this command to add the turning points to the plot.

```
turnx <- c(-0.61, 0.61)
turny <- c( 1.67,-1.67)
points(turnx, turny, pch=5, col=2)
```

The final step is to add the tangent to the curve at $x = 0$, which is defined by the line $y = -4x$. To add this line to the plot we can use the `lines` command, which also accepts the x and y co-ordinate vectors and the formatting augments. Type the next two lines into your editor and whilst executing make sure you understand what each argument does.

```
tangent <- -4*x
lines(x, tangent, lty=2, col=3)
```

The `lines` command can also be used to add non-linear function to an existing plot; just ensure that the co-ordinate vectors contain enough points to make the curved line appear smooth.

Adding straightlines to an existing plot is something that might be done quite often. As a consequence there is a built-in function in **R** to add a straightline of the form

$$y = bx + a$$

called `abline`. Its main inputs are `a` corresponding to the intercept and `b` corresponding to the slope. The same additional options for `lines` such as `lty` and `col` are also available. As a consequence, the following code will also add the tangent to the plot

```
abline(a=0, b= -4, col=3, lty=2)
```

Note that when you use the `plot` command a new image is created every time, whereas the `points`, `lines` and `abline` commands add to an existing plot.

5 Exporting Plots

Once you have created a plot, you may need to export it out of RStudio such that you can import it into another document or to print it. There are two methods of exporting images: saving or copying.

To save your plot, click on 'Export' button in the bottom right panel and then on 'Save Plot as Image...'. A new window will appear containing your plot. In the space next to 'File name:'

type "MYPLOT" or any other name you which to call the file. Finally, click on the 'Save' button. This will be saved into your directory that you specified at the start of the workshop.

Alternatively to saving your plot, you can copy it into a separated document. To do this, click on the 'Export' button in the bottom right panel again, but then click on 'Copy Plot to Clipboard...' instead. A new window will appear with your plot where you need to click on the 'Copy Plot' button. Your plot is now ready to be pasted into any document.

Quiz 1: Plotting a Shape

Copy the following vectors into your console and create a plot that joins the points. What shape does the plot create?

```
x <- c(0.0, 0.8, -0.8, -0.5, 0.0)
y <- c(1.0, -0.5, -0.5, 1.0, 1.0)
```

(**A**) Circle (**B**) Triangle (**C**) Trapezium (**D**) Square (**E**) Parallelogram

---

Quiz 2: Point Number

What is the `pch` number for the plotting symbol $\triangledown$?

Quiz 3: Intersection

Plot the two functions below on the same axes for $-5 \leq x \leq 10$.

$$f(x) = x^4 - 2x^2 - 4 \qquad h(x) = \sinh(x) - 5\sin(x)$$

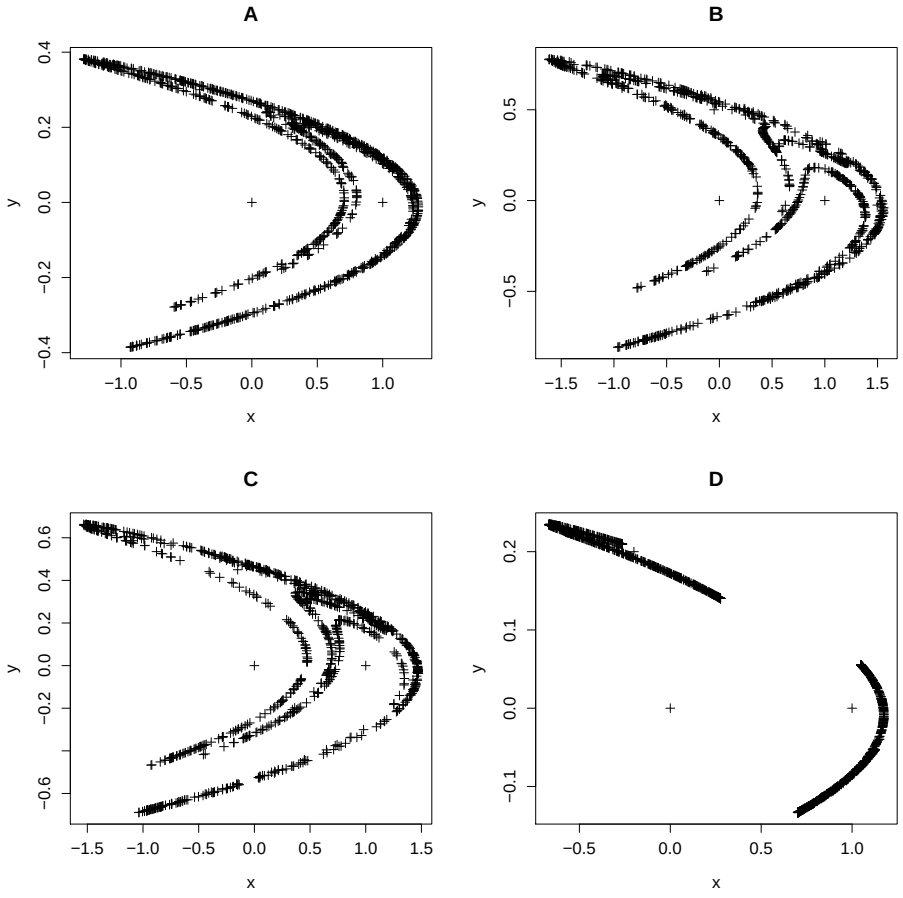
How many times do these functions intersect?

Quiz 4: Hénon Map

The Hénon map is defined by the recursion equations

$$x_{n+1} = y_n + 1 - ax_n^2$$
$$y_{n+1} = bx_n$$

The following plots were produced by starting at $x_0 = y_0 = 0$ and generating a large sequence of (x_n, y_n) (e.g. up to $n = 1000$) for different parameters a, b.
Match the plots to the set of pairs of parameters.



(i) $a = 1.15, b = 0.45$ (iii) $a = 1.4, b = 0.3$
(ii) $a = 1.05, b = 0.5$ (iv) $a = 1.2, b = 0.2$