The Julia Programming Language

Installation

13th February, 2024. I installed julia using the following command:

curl -fsSL https://install.julialang.org — sh

At the end it also gave me a command to update my profile / path:

. /home/steve/.bashrc

At the end I had a working installation of Julia. Version 1.10.0 (2023-12-25)

Adding Packages

In the REPL tap the] key. Then type, for example:

add Plots

The package will be installed. Press backspace to go back to the REPL.

A simple plot

A simple example, which runs in the REPL, is:

```
using Plots
x = range(0, 10, length=100)
y = sin.(x)
plot(x, y)
```

This uses the default output device (called the back end), which is called GR. In general, it is much easier to use the repl to generate the plots than to use the command line for example. There is some code on the internet which allegedly generates plots from the command line but I have not tested it out.

If there is a file containing code which generates a plot it can be read in (from inside the REPL) and evaluated using the line: include(”plot01.jl”)

The plot will be displayed.

An extension of the program both displays the plot on the screen and saves it to a file.

```
using Plots;

x = range(0, 10, length=100)
y = sin.(x)
p = plot(x, y)
png(p, "sinewave")
gui(p)
```

This is the png image created:

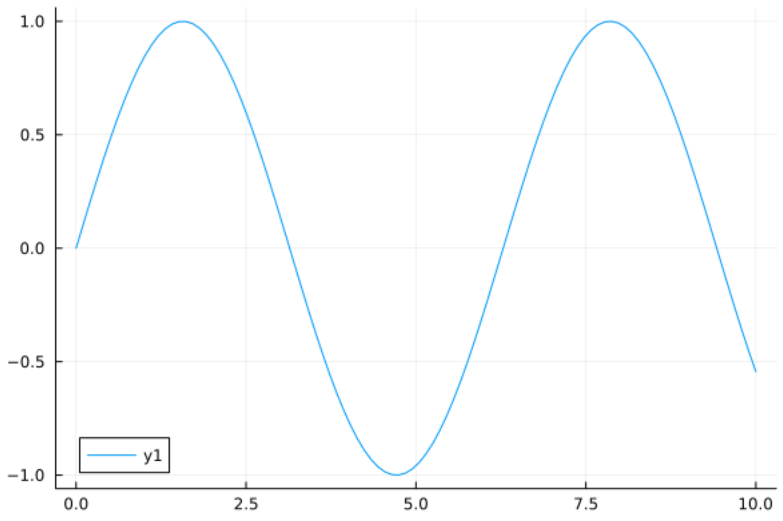

Figure 1: Sine Wave.

More complex plots can be done:

```
using Plots; gr()

x = range(-3, 3, length=30)
s = surface(
x, x, (x, y)->exp(-x^2 - y^2), c=:viridis, legend=:none,
nx=50, ny=50, display_option=Plots.GR.OPTION_SHADED_MESH,  # <-- series[:extra_kwargs]
)
png(s, "surface")
gui(s)
```

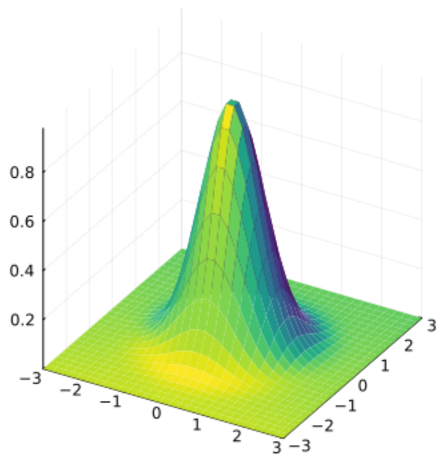

Figure 2: Surface.

A plot can have multiple functions being plotted:

```
using Plots; gr()

x = range(0, 10, length=100)
y1 = sin.(x)
y2 = cos.(x)
p2 = plot(x, [y1 y2])
png("sincos")
gui(p2)
```

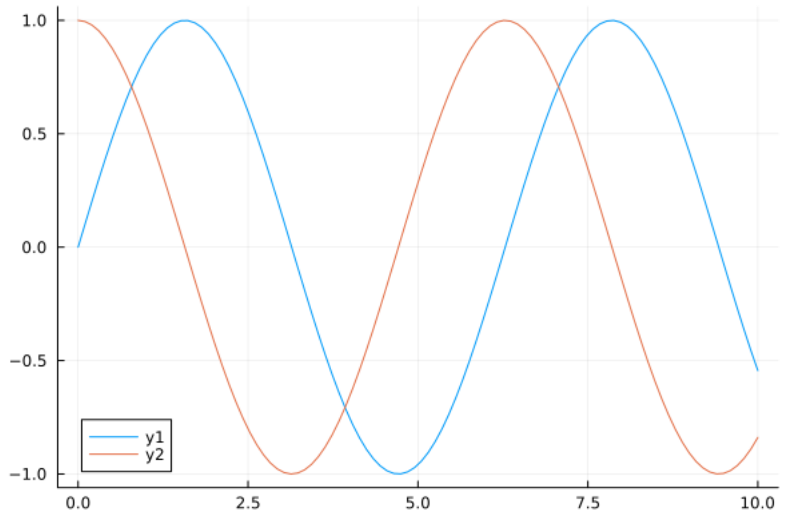

Figure 3: Sine and Cosine.

A plot can be drawn and then more elements added to it. This is done using the plot! command.

```
using Plots; gr()
x = range(0, 10, length=100)
y1 = sin.(x)
y2 = cos.(x)
p = plot(x, [y1 y2])
y3 = @. sin(x)^2 - 1/2
plot!(p, x, y3)
png(p, "sin_cos_etc")
gui(p)
```

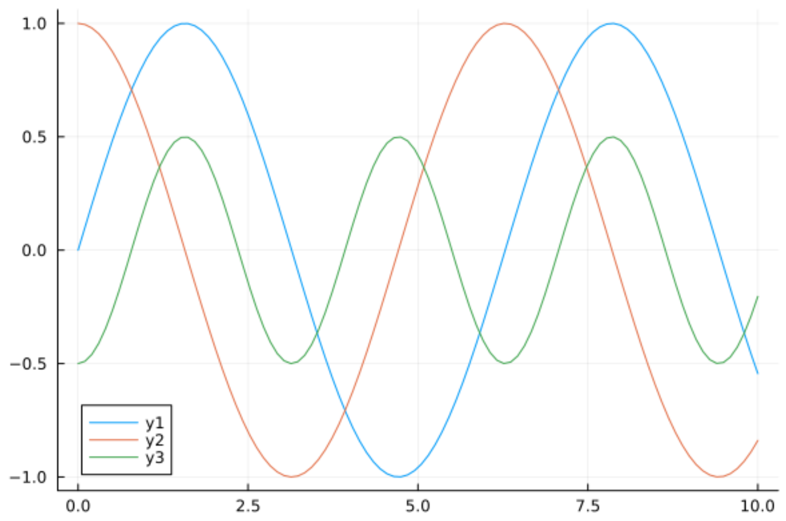

Figure 4: Sine, Cosine etc.

TODO - continue the exploration of plotting here....

List of Figures