```
using PlutoUI
```

Presentación

Programación en Julia: Primeros pasos

Gráficas

Héctor Medel

Benjamín Pérez

Existen varias opciones para elaborar gráficos en Julia

- Plots
- GR
- Plotly
- PyPlot (matplotlib)
- UnicodePlots
- Makie

La documentación la podemos consultar en **https://docs.juliaplots.org/**

Plots es una interfaz de visualización que utiliza otros motores para graficar, como: GR, PyPlot, Plotly, entre otros. Por defecto utiliza GR. Para utilizarla usaremos el comando **using Plots**

```
using Plots
```

```
GRBackend()
```

```
gr()
```

Una vez cargado el paquete **Plots**, podemos elegir el motor que usaremos para graficar. Lo anterior lo logramos ejecutando alguna de las siguientes opciones:

- gr()
- pyplot()
- plotlyjs()

Hagamos una gráfica simple

Nos interesa ahora visualizar la siguiente función

$$f(x) = x^2$$

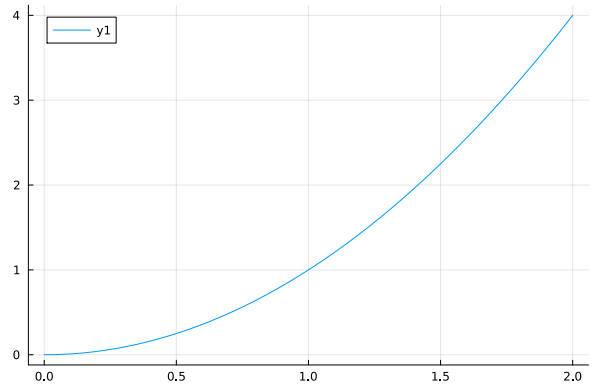
```
x = 0.0:0.01:2.0
```

```
x = 0.0:0.01:2.0  # notemos que podemos hacer x = collect(0.0:0.01:2.0)
```

```
y =
```

```
[0.0, 0.0001, 0.0004, 0.0009, 0.0016, 0.0025, 0.0036, 0.0049, 0.0064, 0.0081, 0.01, 0.0121, 0.0144, 0.0169, 0.0196, 0.0225, 0.025
```

```
y = x.^2
```



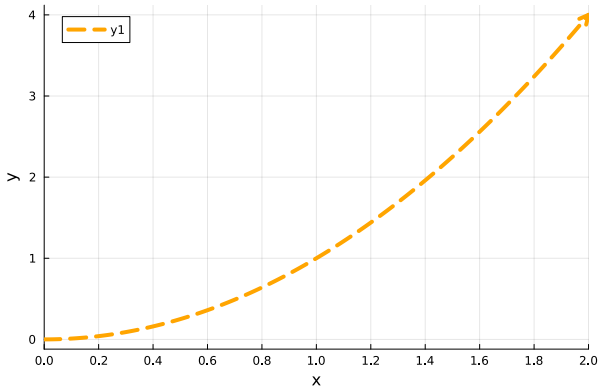

```
plot(x,y)
```

Atributos básicos de gráficas

Dependiendo del tipo de gráfica y el motor utilizado, existen atributos que podemos modificar, por ejemplo el color, estilo o grosor.

```
plotattr(:Series); # Podemos ver los atributos de :Plot, :Subplot, :Axis;
```

```
Defined Series attributes are:
arrow, bar_edges, bar_position, bar_width, bins, colorbar_entry, connections, contour_labels, contours, extra_kwargs,
fill_z, fillalpha, fillcolor, fillrange, fillstyle, group, hover, label, levels, line_z, linealpha, linecolor, linesty
le, linewidth, marker_z, markeralpha, markercolor, markershape, markersize, markerstrokealpha, markerstrokecolor, mark
erstrokestyle, markerstrokewidth, normalize, orientation, permute, primary, quiver, ribbon, series_annotations, series
alpha, seriescolor, seriestype, show_empty_bins, smooth, stride, subplot, weights, x, xerror, y, yerror, z, z_order, z
error
```



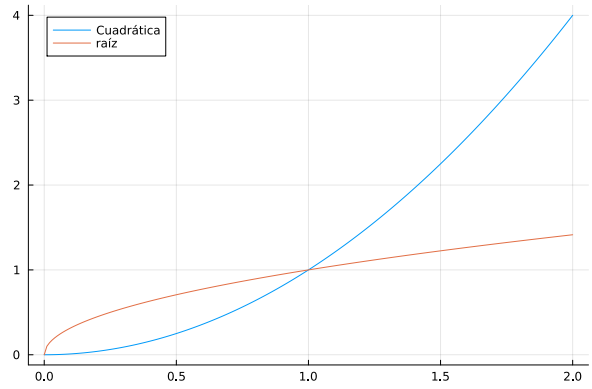

```
plot(x, y,
    color=:orange,
    linewidth=4,
    linestyle=:dash,
    arrow=:arrow,
    xlabel="x",
    ylabel="y",
    xticks = 0:0.2:2,
    xlims = (0,2)
    )  # Entre otras propiedades!
```

Varias gráficas en una misma figura

Para graficar varias funciones en una misma figura usaremos el símbolo **!** de la siguiente manera

```
y2 = sqrt.(x);  # Recordemos que y sigue en memoria con la función cuadrática
```

```
plot(x,y,
    label="Cuadrática");
```




```
plot!(x,y2,
      label="raíz")
```

```
x3 = -2.0:0.01:2.0
```
```
x3 = -2:0.01:2
```

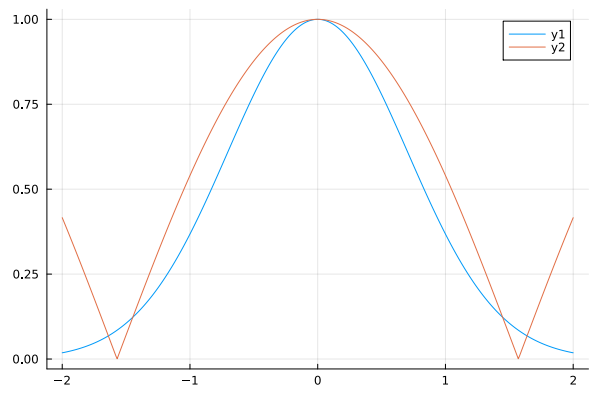
```
y3 =
```
▶ [0.0183156, 0.0190612, 0.0198332, 0.0206322, 0.0214592, 0.0223149, 0.0232001, 0.0241155, 0.0250621, 0.0260406, 0.0270518, 0.0280

```
y3 = exp.(-x3.^2)
```

```
y4 =
```
▶ [0.416147, 0.407033, 0.397879, 0.388685, 0.379452, 0.370181, 0.360873, 0.351529, 0.34215, 0.332736, 0.32329, 0.313811, 0.3043, 0.

```
y4 = abs.(cos.(x3))
```

```
begin
    p = plot();
    p = plot(x3,y3);
    p = plot!(x3,y4);
end;
```




```
plot(p)
```

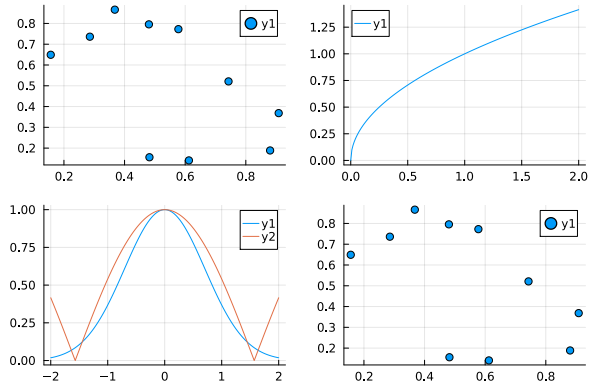
Gráficas tipo scatter y subplots

```
x5 = ▶ [0.61225, 0.285559, 0.908456, 0.480311, 0.577586, 0.367845, 0.482099, 0.742955, 0.880149, 0.156467]
```
```
x5 = rand(10)
```

```
y5 = ▶ [0.140753, 0.736192, 0.368343, 0.795609, 0.772453, 0.866277, 0.155995, 0.521032, 0.188637, 0.649165]
```
```
y5 = rand(10)
```

```
p1 = scatter(x5,y5); # Poner ;  para ocultar la grafica
```

```
p2 = plot(x,y2);
```




```
plot(p1,p2,p,p1)
```

Visualización de funciones de 2 variables

Para funciones bivariadas, tenemos varias opciones de visualización. Usemos como ejemplo la función

$$g(x,y) = e^{-(x^2+y^2)}$$

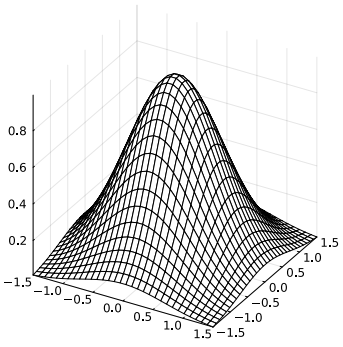
```
begin
    # Size of numerical window
    w0 = 1.0
    xmax = 1.5*w0
    ymax = 1.5*w0

    xmin = -1.5*w0
    ymin = -1.5*w0

    pointsx = 32
    pointsy = 32

    # Generates ranges for xs and ys
    xs = collect(range(xmin,length=pointsx,stop=xmax))
    ys = collect(range(ymin,length=pointsy,stop=ymax))

    # Generates matrices Matlab-style
    mx, ny = length(xs), length(ys)
    Xs = reshape(xs, 1, mx)
    Ys = reshape(ys, ny, 1);
end;
```

Tenemos las coordenadas (x,y) en memoria. Visualicemos la función.

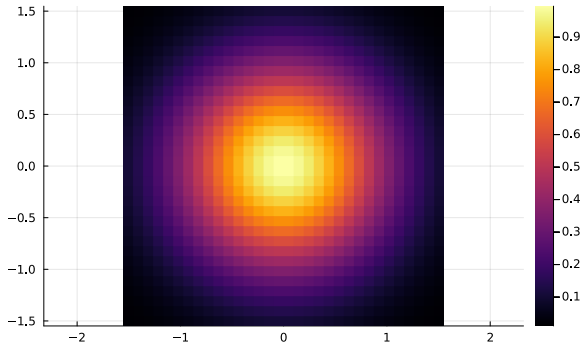
```
gg (generic function with 1 method)
```

```
gg(x,y) = exp(-x^2 - y^2)  # Esto define la funcion de interes
```

```
GG = gg.(Xs,Ys);
```



```
wireframe(xs, ys, GG)  # Puede ser tambien surface()
```

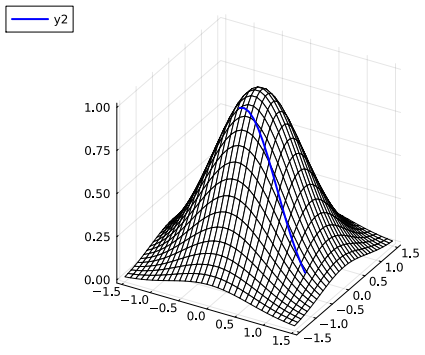


```
heatmap(xs, ys, GG, aspect_ratio=1)
```

Podemos visualizar curvas sobre superficies

[0.907012, 0.920443, 0.913931, 0.887897, 0.844002, 0.784978, 0.714337, 0.636036, 0.554106, 0.472319, 0.393923, 0.321454, 0.25666

```
begin
    tt = -0.2:0.1:1.2
    rx1 = tt
    ry1 = -0.3tt .- 0.3
    rz1 = gg.(rx1,ry1)
end
```




```
begin
    p7 = wireframe(xs, ys, GG)  # Puede ser tambien surface()
    p7 = plot3d!(rx1, ry1, rz1, linewidth=2, color=:blue)
end
```

Podemos grabar las visualizaciones

Usaremos el comando **savefig()**

```
"/home/ben/Dropbox/Documents/Numericos/JuliaLinux/JuliaEnvironments/Dell/pluto/figura.png"
savefig(p7,"figura.png")
```

```
"/home/ben/Dropbox/Documents/Numericos/JuliaLinux/JuliaEnvironments/Dell/pluto/figura.pdf"
savefig(p7,"figura.pdf")
```

```
"/home/ben/Dropbox/Documents/Numericos/JuliaLinux/JuliaEnvironments/Dell/pluto/figura.svg"
savefig(p7,"figura.svg")
```

- Es importante comentar que los formatos soportados dependen del motor para graficar que estemos usando.

Uso de otros motores para graficar

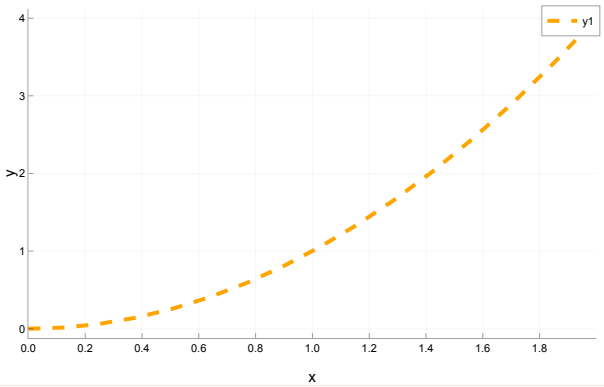
Antes de llamar alguno de los otros motores, es necesario instalarlos. Usamos la función **using** como lo hemos hecho para otros paquetes. Por ejemplo: **using Plotly** (o PlotlyJS)

Posteriormente, para invocar el motor hacemos **plotly()**

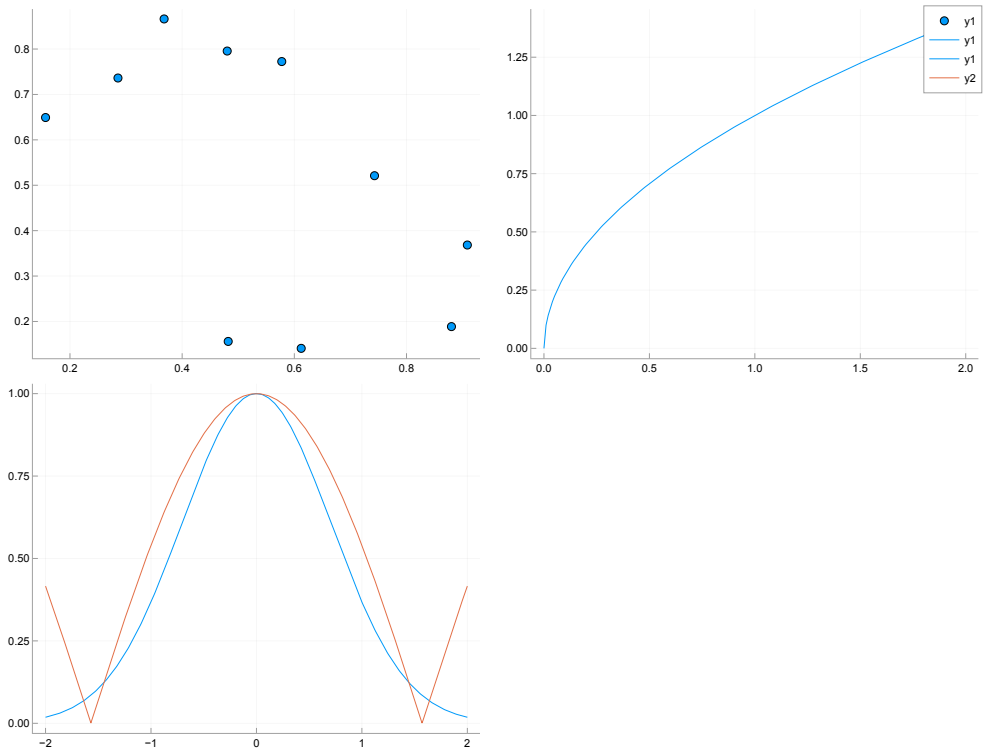
```
▸ PlotlyBackend()
```

```
plotly()
```

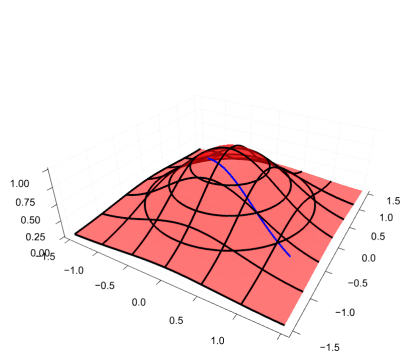
```
⚠ For saving to png with the `Plotly` backend `PlotlyBase` and `PlotlyKaleido` need to be installed.
  err: ▸ ArgumentError("Package PlotlyBase not found in cur" … 52 bytes … " to install the PlotlyBase package.")
```



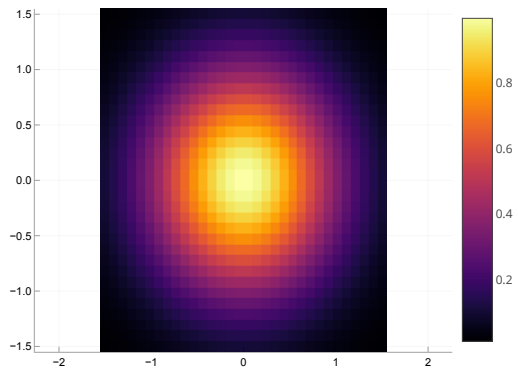

```
plot(x, y,
    color=:orange,
    linewidth=4,
    linestyle=:dash,
    arrow=:arrow,
    xlabel="x",
    ylabel="y",
    xticks = 0:0.2:2,
    xlims = (0,2)
    )  # Entre otras propiedades!
```




```
begin
    p9 = scatter(x5,y5) # Poner ;  para ocultar la grafica
    p10 = plot(x,y2)
    plot(p9, p10, p, size=(1000,800))
end
```




```
begin
    p8 = surface(xs, ys, GG, alpha=0.75, color=:red, colorbar=:false)  # Puede ser tambien surface()
    p8 = wireframe!(xs, ys, GG)  # Puede ser tambien surface()
    p8 = plot3d!(rx1, ry1, rz1, linewidth=4, color=:blue)
end
```




```
heatmap(xs, ys, GG, aspect_ratio=1)
```