

# Programación en Julia: Herramientas para el aprendizaje

---

## Notas interactivas en Pluto

---

Héctor Medel

Benjamín Pérez

### ¿Qué es Pluto?

---

Es un ambiente de programación para Julia de tipo *notebook* (con soporte para markdown) que tiene las siguientes características:

- Reactividad
- Interactividad
- Simplicidad

Es útil para educación y exploración en proyectos más ambiciosos.

### Generalidades del entorno

---

- Los *shortcuts* del teclado podemos obtenerlos al presionar F1.
- Barra de exportación.
- Cuadro de texto para grabar.
- Documentación.
- Navegación en celdas

### Reactividad

---

¿A qué nos referimos con reactividad?

Pluto rastrea las dependencias entre celdas, y actualiza los valores de acuerdo a dicha cadena de dependencias.

```
a = 1
```

```
1 a = 1
```

```
b = 10
```

```
1 b = 10
```

```
c = 2.1
```

```
1 c = a/b + 2
```

Algunos puntos importantes:

- La reactividad nos permite hacer un acomodo *no usual* de las celdas.
- En cada celda solo hemos puesto una instrucción (hasta ahora).
- El rastreo de las dependencias de las celdas, crea un problema si volvemos a definir variables en celdas posteriores.

## Bloques de código

Para ejecutar varias líneas de código desde una celda tenemos dos opciones:

```
begin
  CODIGO
  CODIGO
  CODIGO
end
```

```
let
  CODIGO
  CODIGO
  CODIGO
end
```

¿Qué diferencia hay?

```
5
1 begin # El scope de las variables definidas es global
2   d = 2
3   e = 2+a+d
4 end
```

```
5
1 let # El scope de las variables definidas es local
2   f = 2
3   e = 2+a+f
4 end
```

```
2
1 d
```

## Interactividad

Una manera de agregar elementos interactivos (sliders, botones, etc) es usando el paquete PlutoUI.jl. Para esto, ejecutaremos el siguiente comando en una nueva celda

```
using PlutoUI
```

```
1 using PlutoUI
```

## Sliders

Tenemos la siguiente sintaxis:

```
@bind VARIABLE Slider(RANGO, VALOR POR DEFECTO, TRUE/FALSE)
```

```
5
1 @bind sliderA Slider(0:10, 5, true)
```

Para múltiples sliders usaremos el símbolo \$ dentro del entorno de md.

B  5

C  5

```
1 md"
2 B $(@bind sliderB Slider(0:10, 5, true))
3
4 C $(@bind sliderC Slider(0:10, 5, true))
5 "
```

10

```
1 sliderB+sliderC
```

## Scrubable

La sintaxis es muy similar a la de Slider. Veamos.

```
@bind VARIABLE Scrubbable(RANGO, default=VALOR POR DEFECTO)
```

5

```
1 @bind scrubD Scrubbable(0:10, default=5)
```

## Number field

La sintaxis es la siguiente:

```
@bind VARIABLE NumberField(RANGO, VALOR POR DEFECTO)
```

1

```
1 @bind numeroE NumberField(-5:5, default=1)
```

2

```
1 2*numeroE
```

## TextField

La sintaxis es:

```
@bind VARIABLE TextField(default = VALOR POR DEFECTO)
```

```
1 @bind textF TextField()
```

""

```
1 textF
```

```
1 @bind numeroG TextField()
```

""

```
1 numeroG # Esta es una variable tipo char
```

ArgumentError: cannot parse "" as Float64

```
1. _parse_failure(::Type, ::String, ::Int64, ::Int64) @ parse.jl:373
2. #tryparse_internal#478 @ parse.jl:369 [inlined]
3. tryparse_internal @ parse.jl:366 [inlined]
4. #parse#479 @ parse.jl:379 [inlined]
5. parse(::Type{Float64}, ::String) @ parse.jl:379
6. top-level scope @ [Local: 1] [inlined]
```

```
1 parse(Float64, numeroG)
```

## CheckBox

La sintaxis es:

```
@bind checkG CheckBox()
```

Palomeamos el cuadro ☐

```
1 md"
2 Palomeamos el cuadro $(@bind checkG CheckBox())
3 "
```

false

```
1 checkG
```

## Button

La sintaxis es:

```
@bind buttonH Button("TEXT0")
```

Botón

```
1 @bind buttonH Button("Botón")
```

"Botón"

```
1 buttonH
```


0.9527735870516731

```
1 begin
2     buttonH
3     rand()
4 end
```

## Clock

La sintaxis es la siguiente:

```
@bind t Clock()
```

  speed:  secs / tick

```
1 @bind t Clock()
```

deltaT = 0.1

```
1 deltaT = 1/10
```



```
1 @bind tf Clock(deltaT, true)
```

1

```
1 tf
```

# Gráficas

Queremos generar gráficas que sean manipulables vía sliders, cuadros de texto u otras entradas interactivas.

Vamos a explorar la función

$$f(x) = Ae^{-(x-x_0)^2/w^2}$$

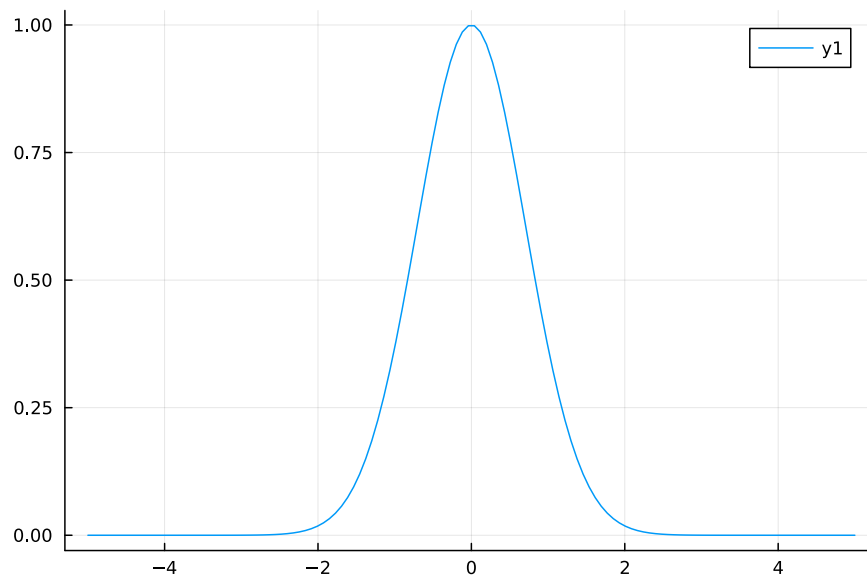
```
1 using Plots
```

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

```
1 f(x; A=1.0, w=1.0) = A * exp(-x^2/w^2)
```

```
► (1.0, 0.735759, 1.0)
```

```
1 f(0), f(1; A=2), f(0; w=2)
```



```
1 begin
2   #range(-5.0, 5.0, 128) <-- esa sintaxis funciona a partir de la v1.8
3   xs = range(-5.0, length=128, stop=5.0)
4   plot(xs, f.(xs))
5 end
```

```
► [-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
```

```
1 collect(range(-5.0, 5.0, length=11))
```

```
► [-5.0, -4.0, -3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
```

```
1 collect(range(-5.0, length=11, stop=5.0))
```

```
► [-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0]
```

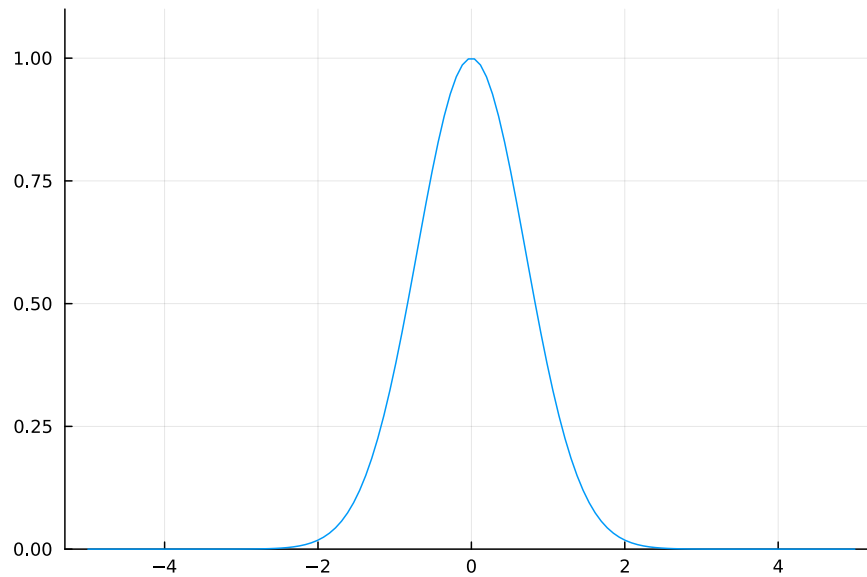
```
1 collect(-1.0:2/10:1.0)
```

Vamos a generar los sliders para manipular la gráfica.

xo

Ao

wo



```

1 begin
2   # xs = range(-5.0, 5.0, 128) # Ya esta definida en la celda anterior!
3   ys = f.(xs.-x0; A=A0, w=w0)
4   plot(xs, ys, ylims=(0,1.1), leg=:false)
5 end

```

g (generic function with 1 method)

```

1 g(x, A, w) = A * exp(-x^2/w^2)

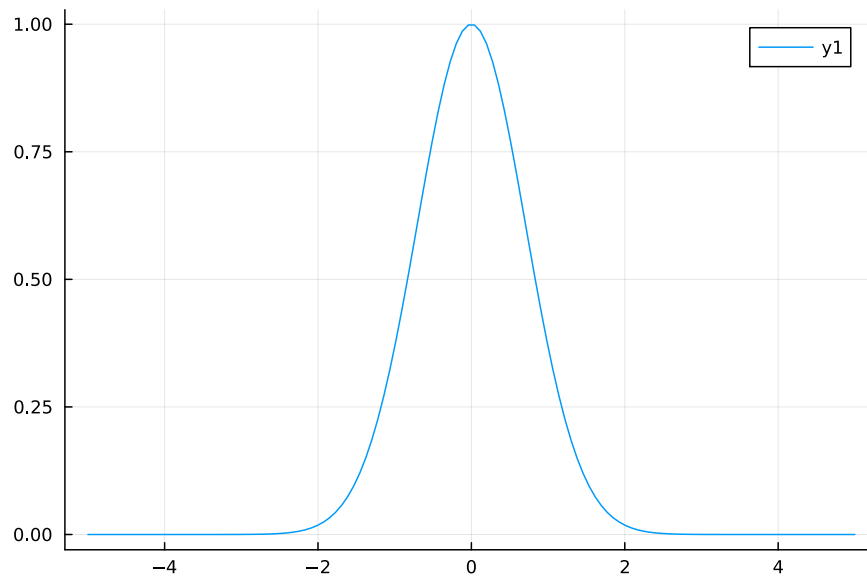
```

► (1.0, 0.735759, 1.0)

```

1 g(0,1,1), g(1, 2,1), g(0,1,1)

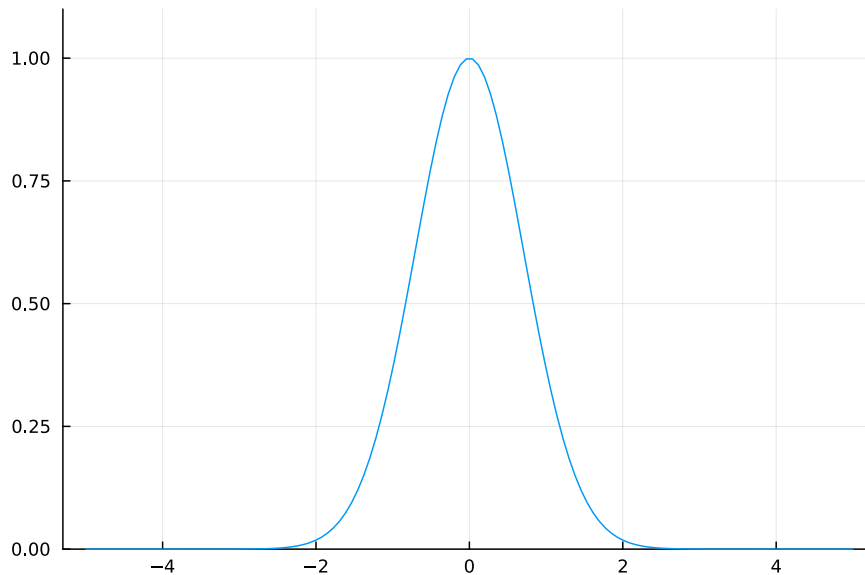
```



```

1 begin
2   # xs = range(-5.0, 5.0, 128)
3   plot(xs, g.(xs,1,1))
4 end

```



```
1 begin
2   # xs = range(-5.0, 5.0, 128)
3   zs = g.(xs.-x0,A0,w0)
4   plot(xs, zs, ylims=(0,1.1), leg=:false)
5 end
```

## Otros ajustes

### Tabla de contenidos

PlutoUI permite agregar una tabla de contenidos de acuerdo a los *headlines* del notebook.

Para agregar ejecutamos el siguiente comando en una celda

```
TableOfContents(title="Contenidos", indent=true/false, depth=3 ,aside=true/false)
```

```
1 #TableOfContents(title="Contenidos", indent=true, aside=true)
```

### Ancho de las celdas

Es posible agregar código en html o css. Esto nos puede ayudar a hacer ajustes finos de las notas que estamos generando.

Para cambiar el ancho de las celdas, usaremos la celda en modo html. Ejecutaremos la siguiente instrucción en una celda:

```
html" <style>
main {
  max-width: ANCHOpix;
}
```

```
1 html"<style>
2 main {
3   max-width: NNpx;
4 }
5 "
```

