



Actividad | # | Método secante y newton

Métodos numéricos
Ingeniería en Desarrollo de Software



TUTOR: MIGUEL ANGEL RODRIGUEZ VEGA

ALUMNO: GUSTAVO ALONSO ESPINOZA ROMERO_A2

FECHA: 5/03/2024

INDICE	
INTRODUCCION	3
DESCRIPCION	4
JUSTIFICACION.....	5
DESARROLLO	6
Método de newton rapson	6
Método de la secante	9
Interpretación de resultados.....	11
CONCLUSION	12

INTRODUCCION

Para esta segunda actividad vamos a hacer el uso de rstudio, el cual nos ayudará a resolver unas ecuaciones que se solicitan, estos son el método de newton rapson y de la secante, estos métodos nos ayudan a encontrar aproximaciones a la solución que estemos buscando, cada uno de ellos tiene sus propios procedimientos, pero ambos tienen sus ventajas, y a su vez uno puede llegar a ser mejor que el otro, todo esto podremos visualizarlo en la parte de desarrollo, los cuales serán explicados mediante una descripción y una imagen de cada proceso.

Mostraremos como se resolvieron cada uno de ellos y explicaremos los pasos que seguimos para poder realizar estos métodos con la herramienta rstudio.

se explicará cuáles son las ventajas de utilizar rstudio, sus veneficios, ya que este te facilita mucho tiempo a la hora de trabajar, teniendo muchas ventas, y una opción de uso para trabajar en ella

DESCRIPCION

Las ejecuciones que se solicitan en la actividad serán mostradas mediante imágenes, se tomaran capturas de todos los pasos que necesitamos para poder resolver las ecuaciones, cada paso a resolver será explicado, y así poder tener una comprensión de cómo es que se resuelve y como es que estos métodos pueden ayudarnos de una manera fácil y rápida.

Uno de los métodos que usaremos el cual es el de newton rapson, es un método iterativo lo cual quiere decir que se repite un determinado número de veces, hasta que se llegue a la solución, para realizarlo se basa en una formula, con la cual permite que se realicen las iteraciones.

El método de la secante es parecido al anterior mencionado, ya que basa en la fórmula de newton rapson, evitando el cálculo de la derivada, este tiene un poco de desventaja ya que tarda un poco más en encontrar la aproximación a la solución.

JUSTIFICACION

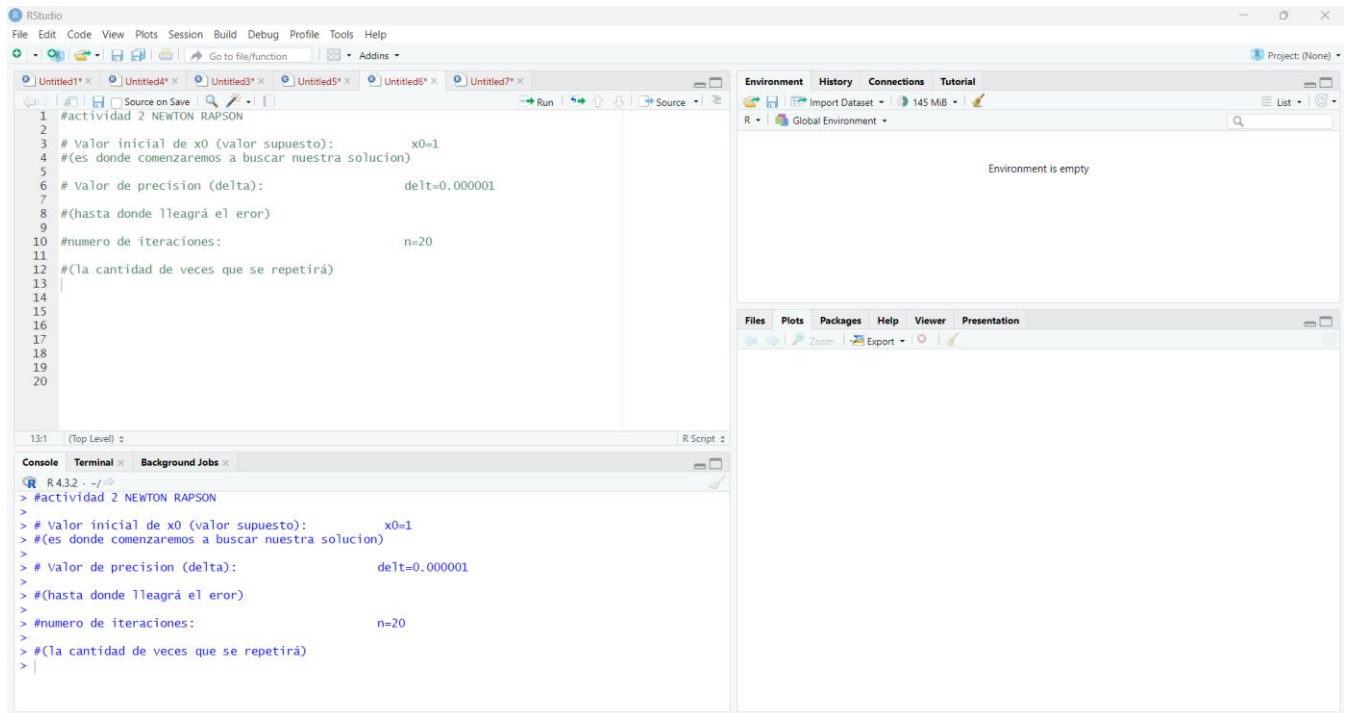
El hacer uso de rstudio para la ejecución de las ecuaciones que se nos solicitan en la actividad, nos permitirá comprender como es el uso de esta herramienta, como funciona y en que se puede usar, sirviendo como practica para que podamos adquirir los conocimientos necesarios para que al momento de que los necesitemos podamos hacer uso de ellos de una manera correcta. rstudio tiene más ventajas que desventajas, siendo así que sea una elección para ser utilizada sobre otras, la más grande ventaja que tiene es que es gratuito, siendo accesible para cualquiera, también teniendo facilidad de uso, todas sus funciones que necesitamos, etc.

Este tipo de herramientas, ayuda a que resolvamos rápidamente los métodos que se nos solicitan, mostrando la gran ventaja que tenemos al utilizarlo, siendo más fácil llegar a la solución sin que tengamos que esforzarnos tanto, como cuando se hace de otras maneras o con otras herramientas.

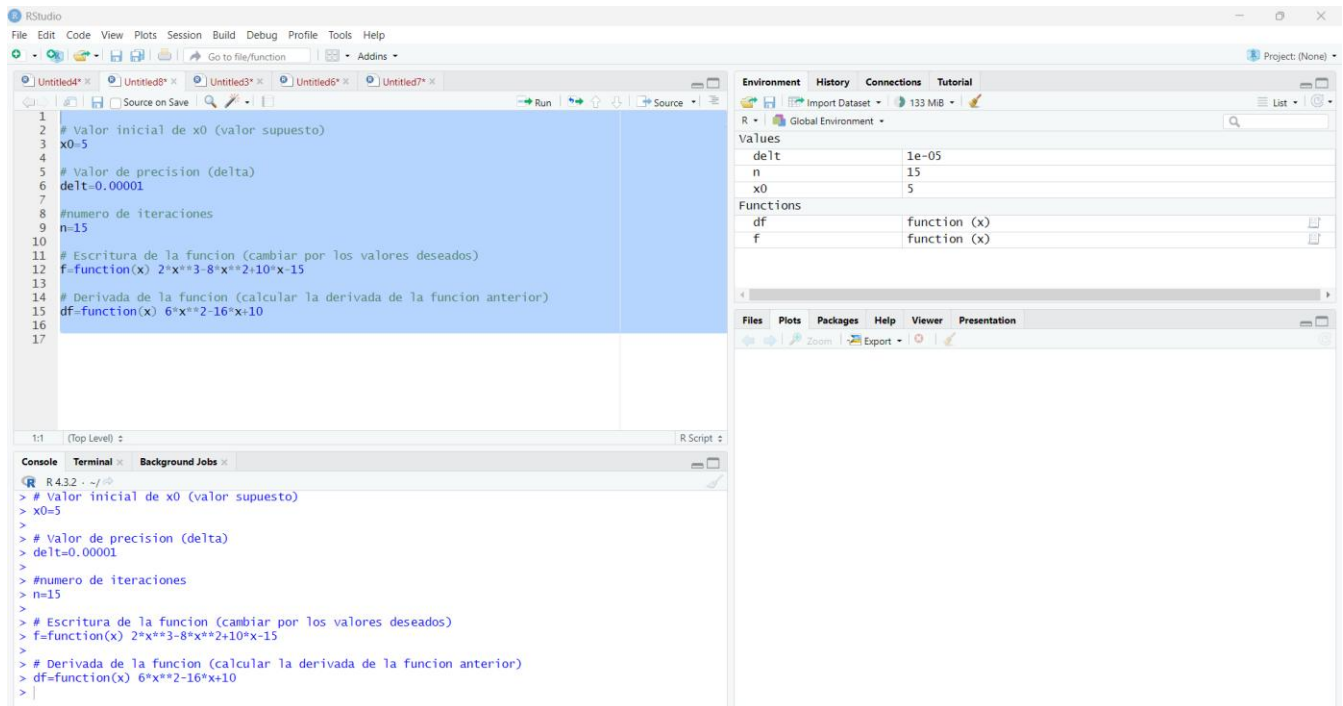
DESARROLLO

Método de newton rapson

En la siguiente imagen se muestran los valores iniciales que necesitamos para realizar el método de newton, los valores son: $x_0=1$, la precisión= 0.000001 y el máximo de iteraciones es 20.



Posteriormente se coloca la función y la derivada de la misma, tal como se muestra.



The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains an R script with the following code:

```
1  
2 # Valor inicial de x0 (valor supuesto)  
3 x0=5  
4  
5 # valor de precision (delta)  
6 delt=0.00001  
7  
8 #numero de iteraciones  
9 n=15  
10  
11 # Escritura de la funcion (cambiar por los valores deseados)  
12 f=function(x) 2*x**3-8*x**2+10*x-15  
13  
14 # Derivada de la funcion (calcular la derivada de la funcion anterior)  
15 df=function(x) 6*x**2-16*x+10  
16  
17
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
delt	1e-05
n	15
x0	5

Under the 'Functions' section, it lists:

Function Name	Definition
df	function (x)
f	function (x)
- Console:** Shows the execution of the script, with the following output:

```
R 4.3.2 ~ /  
> # Valor inicial de x0 (valor supuesto)  
> x0=5  
>  
> # Valor de precision (delta)  
> delt=0.00001  
>  
> #numero de iteraciones  
> n=15  
>  
> # Escritura de la funcion (cambiar por los valores deseados)  
> f=function(x) 2*x**3-8*x**2+10*x-15  
>  
> # Derivada de la funcion (calcular la derivada de la funcion anterior)  
> df=function(x) 6*x**2-16*x+10  
>
```

Una vez colocada la función, se pasa a realizar un ciclo el cual es for, con el que nos permitirá hacer las iteraciones que necesitamos, ponemos la condicional y lo que hará si esta es verdadera, le indicamos lo que mostrara en pantalla con print y nos muestra el número de iteraciones que se hicieron para llegar a la solución, el resultado de x1 y las aproximaciones que se obtuvieron, para llegar a la solución.

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for a function `f` and its derivative `df`, and a `for` loop that iterates until the solution converges within a specified precision (`delta`).
- Console:** Shows the execution output, including the iterative values of `x0`, `x1`, and the error, followed by a message indicating convergence after 6 iterations.
- Environment Pane:** Lists the variables created during execution, including `delt`, `error`, `i`, `n`, `x0`, and `x1`, along with the functions `df` and `f`.

```
# Valor de precision (delta)
delt=0.00001
# numero de iteraciones
n=15
# Escritura de la funcion (cambiar por los valores deseados)
f=function(x) 2*x^3-8*x^2+10*x-15
# Derivada de la funcion (calcular la derivada de la funcion anterior)
df=function(x) 6*x^2-16*x+10
# Ciclo de iteraciones y resultados
for (i in 1:n) {
  x1=x0-f(x0)/df(x0)
  print(c(i,x0,x1)); error=abs(x1-x0)
  if (error<delt){
    cat("La solucion converge en ",i , " iteraciones. raiz= ", x1);
    break()
  }
  x0=x1
}
print("maximo numero de iteraciones alcanzada !!!")
```

Console Output:

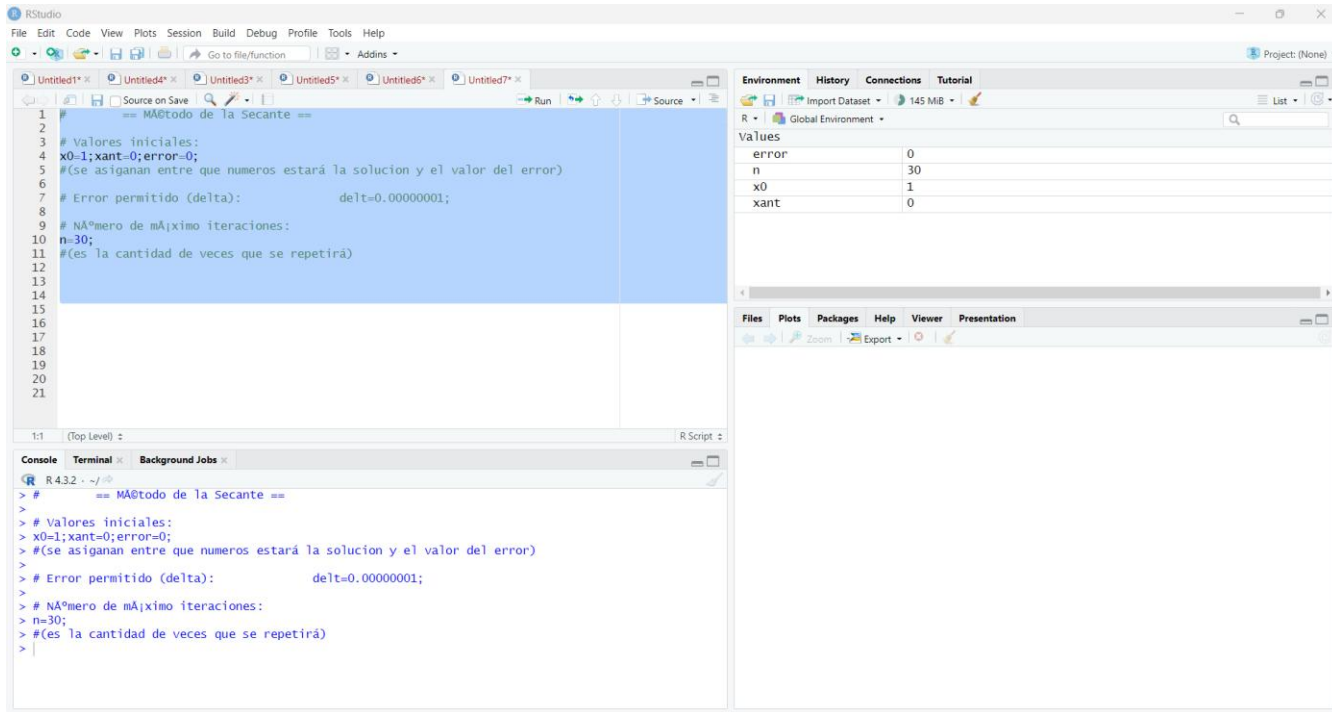
```
[1] 1.0000 5.0000 3.9375
[1] 2.000000 3.937500 3.376903
[1] 3.000000 3.376903 3.190023
[1] 4.000000 3.190023 3.169282
[1] 5.000000 3.169282 3.169038
[1] 6.000000 3.169038 3.169038
La solucion converge en 6 iteraciones. raiz= 3.169038> print("maximo numero de iteraciones alcanzada !!!")
[1] "maximo numero de iteraciones alcanzada !!!"
>
```

Environment Variables:

Variable	Value
delt	1e-05
error	3.35739618329001e-08
i	6L
n	15
x0	3.16903769674166
x1	3.1690376631677
df	function (x)
f	function (x)

Método de la secante

En esta parte asignamos los valores, estos fueron: x_0 con un valor de 1, x_{ant} con valor de 0, la precisión(error) 0.0000001, y el número máximo de iteraciones 30.



```
1 == MA@todo de la secante ==
2
3 # Valores iniciales:
4 x0=1;xant=0;error=0;
5 #(se asignan entre que numeros estará la solución y el valor del error)
6
7 # Error permitido (delta):      delt=0.00000001;
8
9 # NAúmero de máximo iteraciones:
10 n=30;
11 #(es la cantidad de veces que se repetirá)
12
13
14
15
16
17
18
19
20
21
```

Environment History Connections Tutorial

R Global Environment

Values

error	0
n	30
x0	1
xant	0

Files Plots Packages Help Viewer Presentation

Console Terminal Background Jobs

```
R 4.3.2 ~ />
> == MA@todo de la Secante ==
>
> # Valores iniciales:
> x0=1;xant=0;error=0;
> #(se asignan entre que numeros estará la solución y el valor del error)
>
> # Error permitido (delta):      delt=0.00000001;
>
> # NAúmero de máximo iteraciones:
> n=30;
> #(es la cantidad de veces que se repetirá)
>
```

Colocamos la función la cual es: $f(\text{teta}) = \sin(\text{teta}) + \cos(1 - \text{teta}^2) - 1$

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains an R script for the Secant method. The code includes comments in Spanish and defines a function `f` to find the root of the equation $f(\text{teta}) = \sin(\text{teta}) + \cos(1 - \text{teta}^2) - 1$.
- Environment:** Shows the current environment with variables `error` (0), `n` (30), `x0` (1), and `xant` (0). The function `f` is also listed.
- Console:** Shows the execution of the script, with the function `f` being defined.

```
1 # == MA@todo de la Secante ==  
2  
3 # Valores iniciales:  
4 x0=1;xant=0;error=0;  
5 #(se asignan entre que numeros estará la solución y el valor del error)  
6  
7 # Error permitido (delta):      delt=0.00000001;  
8  
9 # numero de máximo iteraciones:  
10 n=30;  
11 #(es la cantidad de veces que se repetirá)  
12  
13 # Funcion para encontrar su raíz:  
14 f<-function(teta) sin(teta) + cos(1-teta^2)-1;  
15  
16  
17  
18  
19  
20  
21  
22  
23
```

Environment:

Variable	Value
error	0
n	30
x0	1
xant	0

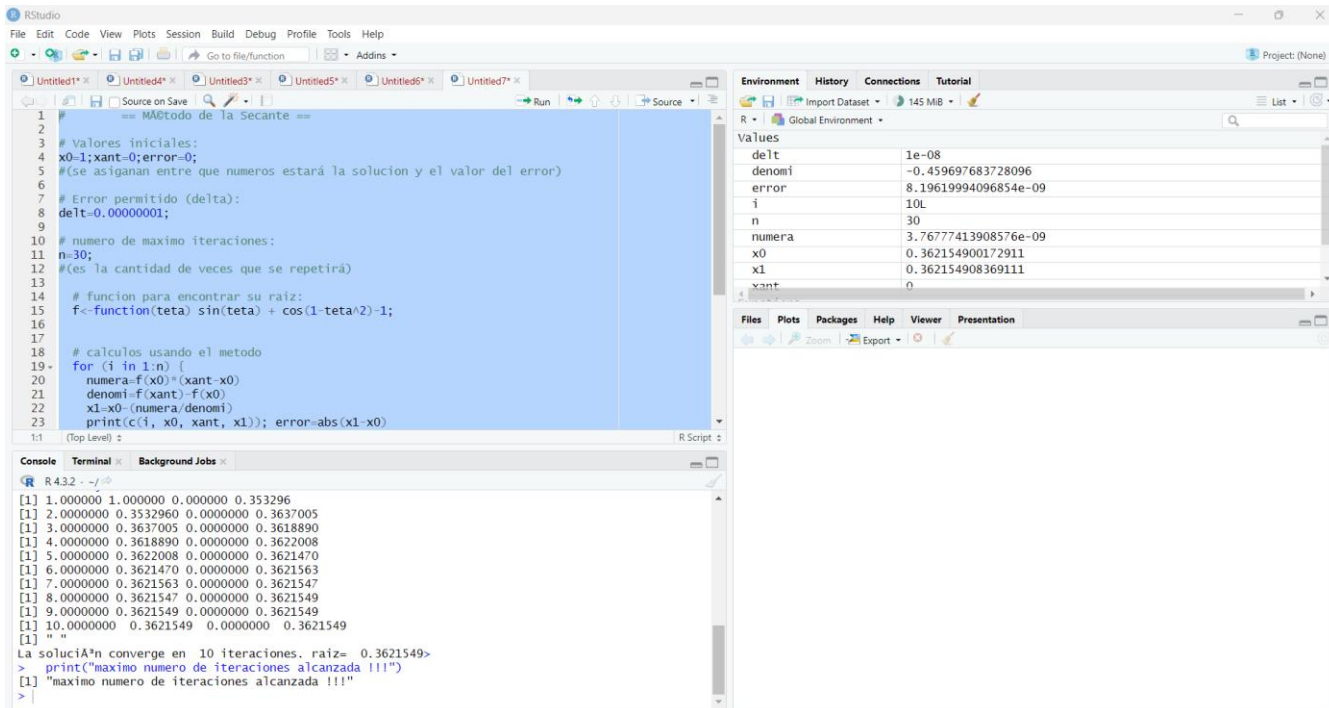
Functions:

Function	Definition
f	function (teta)

Console:

```
> # == MA@todo de la Secante ==  
>  
> # Valores iniciales:  
> x0=1;xant=0;error=0;  
> #(se asignan entre que numeros estará la solución y el valor del error)  
>  
> # Error permitido (delta):      delt=0.00000001;  
>  
> # numero de máximo iteraciones:  
> n=30;  
> #(es la cantidad de veces que se repetirá)  
>  
> # Funcion para encontrar su raíz:  
> f<-function(teta) sin(teta) + cos(1-teta^2)-1;  
>
```

Posteriormente realizamos el ciclo para las iteraciones que necesitamos, ponemos la fórmula para resolver la secante y le damos la orden de que muestre los valores en cada iteración, hasta llegar al resultado buscado, en este caso el resultado fue 0.3621549.



```
1 #== Método de la secante ==
2
3 # Valores iniciales:
4 x0=1;xant=0;error=0;
5 #Se asignan entre que numeros estará la solución y el valor del error)
6
7 # Error permitido (delta):
8 delt=0.00000001;
9
10 # numero de maximo iteraciones:
11 n=30;
12 #es la cantidad de veces que se repetirá)
13
14 # funcion para encontrar su raiz:
15 f<-function(teta) sin(teta) + cos(1-teta^2)-1;
16
17 # calculos usando el metodo
18 for (i in 1:n) {
19   numera=f(x0)*(xant-x0)
20   denom=f(xant)-f(x0)
21   x1=x0-(numera/denom)
22   print(c(i, x0, xant, x1)); error=abs(x1-x0)
23 }
24
25 La solución converge en 10 iteraciones. raiz= 0.3621549>
26 print("maximo numero de iteraciones alcanzada !!!")
27 [1] "maximo numero de iteraciones alcanzada !!!"
```

Variable	Value
delt	1e-08
denom	-0.459697683728096
error	8.19619994096854e-09
i	10L
n	30
numera	3.76777413908576e-09
x0	0.362154900172911
x1	0.362154908369111
xant	0

Interpretación de resultados

Newton rapson. Nos dimos cuenta que con este método nos tomó solo cuatro iteraciones para llegar a la solución, demostrando que es el método más rápido, con lo cual tiene la ventaja sobre los otros métodos que hay.

Secante. para este método nos tomó 18 iteraciones para llegar a la solución, siendo también un método muy rápido y eficiente, demostrando lo parecido que es al método de newton rapson en ciertas partes, solo cambiando su fórmula para encontrar la solución.

CONCLUSION

Con el término de la actividad aprendimos a realizar los métodos de la secante y newton rapson en RSTUDIO, dándonos cuenta lo mucho que nos facilita el uso de esta herramienta, también nos queda más claro cómo es que se manejan este tipo de programas, permitiéndonos tener una comprensión y una idea de cómo pueden ser otro tipo de herramientas parecidas a esta.

Al ingresar los métodos, se obtuvieron ciertos conocimientos de cómo es que se usa RSTUDIO, como se realizan ciertas operaciones del mismo y para qué sirven ciertas funciones, teniendo un gran beneficio en cuestiono de conocimiento al usar esta gran herramienta. De esta manera en el futuro no será problema hacer uso de estos conocimientos, ya que se aprendió lo que se necesita para reanalizar la actividad y posteriormente aplicarlo en el mundo real o mundo laboral, utilizándolo en el momento que nosotros lo vayamos a necesitar.

