



Actividad | #1 | Análisis de conceptos

Métodos Numéricos

Ingeniería en Desarrollo de Software



TUTOR: MIGUEL ANGEL RODRIGUEZ VEGA

ALUMNO: ANWAR DAVID CARAVANTES PERAZA

FECHA: 15 DE JULIO DE 2025

ÍNDICE

ÍNDICE.....	2
INTRODUCCIÓN.....	3
DESCRIPCIÓN.....	4
JUSTIFICACIÓN.....	5
DESARROLLO.....	9
Descarga de RStudio.....	9
Carga de Valores_numéricos.R.....	13
Ejecución de Valores_numéricos.R.....	15
Dígitos Enteros y Fracciones.....	15
Redondeo.....	18
Dígitos Significativos.....	19
Definir Variables y Asignar Valores.....	20
Definir Funciones y Pasar Parámetros.....	24
Graficación de Funciones.....	25
CONCLUSIÓN.....	27
REFERENCIAS.....	28

INTRODUCCIÓN

Cuando hablamos de Ingeniería vamos a encontrarnos con una amplia variedad de problemas, desde problemas muy sencillos, hasta problemas muy complejos. Para este tipo de problemas complejos es de gran dificultad su resolución por métodos tradicionales, por lo que tenemos que emplear un análisis más complejo encontrar sus soluciones, aquí es donde entra en práctica los métodos numéricos, herramientas indispensables para poder completar estos desafíos complejos y laboriosos.

En esta actividad estaremos conociendo los conceptos de los métodos numéricos como lo son los distintos tipos de errores, los valores iniciales, las iteraciones, las variables, las funciones, interpretación de graficas de funciones, etc., y no solo eso, sino que lo estaremos trabajando aplicándolo a la programación por medio de lenguaje R apoyandonos del ambiente RStudio, donde aprenderemos los comandos y funciones básicas de este lenguaje, que nos ayudará a formar las bases para poder trabajar con los métodos numéricos de una forma más práctica.

DESCRIPCIÓN

En esta actividad se nos solicita trabajar con el software RStudio para poder trabajar una serie de comandos básicos del lenguaje R que se estarán trabajando para la aplicación de distintos métodos numéricos a lo largo de las actividades.

Primero debemos instalar tanto el Lenguaje R como el RStudio para nuestro sistema operativo, una vez que lo tenemos instalado, se nos proporciona en la actividad el archivo con los scripts que debemos cargar al RStudio y posterior debemos ir ejecutando y explicando cada uno de los pasos que nos indican en el archivo.

Vamos a definir valores a variables y asignar valores, trabajaremos en definir funciones y pasar parámetros, trabajaremos con dígitos enteros y fracciones, debemos de redondear valores de dos maneras diferentes, vamos a descubrir cómo mostrar valores, de qué manera cambiar los dígitos significativos de números, y por último debemos poder graficar funciones para que nos podamos apoyar por medio de la gráfica en la aplicación de los métodos numéricos.

JUSTIFICACIÓN

“Los métodos numéricos son una sucesión de operaciones matemáticas utilizadas para encontrar una solución numérica aproximada a un problema determinado. Es decir, se trata de una serie de cálculos para acercarnos lo más posible a una solución numérica con una precisión razonablemente buena. Los métodos numéricos son utilizados en ingeniería para facilitar la resolución de problemas que conllevan una enorme cantidad de cálculos, lo que permite ahorrar tiempo. Entendiéndose como cálculos aritméticos a las operaciones aritméticas básicas, cálculos de funciones, consultas de tablas de datos, etc, podemos decir entonces, que los métodos numéricos son una lista finita de instrucciones precisas, las cuales requieren una secuencia de operaciones algebraicas y lógicas (algoritmo). Dichas secuencias dan como resultado una solución aproximada a un problema determinado.” (Noguera & Noguera, 2025)

Estos métodos numéricos nos facilitarán la resolución de problemas complejos, además los trabajaremos a través de un lenguaje de programación, en este caso R, pero ¿cómo vamos a trabajar con él?. Para conocer y abordar sobre el lenguaje R, vamos a citar a Blasco (2024) que nos explica qué es R y cómo podemos trabajar con él:

“R es un lenguaje de programación y un entorno de software libre y de código abierto diseñado específicamente para el análisis estadístico y la visualización de datos. Es ampliamente utilizado en diversas disciplinas, incluyendo la ciencia de datos, la bioinformática, la economía, la epidemiología y muchas otras áreas donde se requiere análisis cuantitativo.

R ofrece diversas estructuras de datos que permiten almacenar y manipular información

de manera eficiente. Algunas de las estructuras de datos más comunes en R incluyen:

Vector: Un vector es una secuencia ordenada de elementos del mismo tipo. Puede ser un vector numérico, de caracteres, lógico, entre otros.

```
vector_numerico <- c(1, 2, 3, 4, 5)
```

```
vector_caracteres <- c("a", "b", "c", "d", "e")
```

Matriz: Una matriz es una estructura bidimensional que contiene elementos del mismo tipo organizados en filas y columnas.

```
matriz <- matrix(1:9, nrow = 3, ncol = 3)
```

Lista: Una lista es una colección ordenada de objetos que pueden ser de diferentes tipos, como vectores, matrices, data frames, entre otros.

```
lista <- list(numeros = c(1, 2, 3), letras = c("a", "b", "c"))
```

Data frame: Un data frame es una estructura tabular similar a una matriz, donde cada columna puede ser de un tipo diferente y se pueden aplicar operaciones de data frame específicas.

```
data <- data.frame(ID = 1:3, Nombre = c("Juan", "María", "Pedro"), Edad = c(25, 30, 28))
```

En R, se pueden realizar diversas operaciones básicas para manipular datos, como:

Asignación de valores: Se utiliza el operador `<-` o `=` para asignar valores a variables. Por

ejemplo: `x <- 10` asigna el valor 10 a la variable x.

Operaciones aritméticas: R permite realizar operaciones aritméticas básicas como suma, resta, multiplicación y división utilizando los operadores `+`, `-`, `*` y `/`, respectivamente.

Indexación: Para acceder a elementos específicos de una estructura de datos, se utiliza la indexación. Por ejemplo, `vector[3]` accede al tercer elemento de un vector.

```
mi_vector <- c(10, 20, 30, 40, 50)
```

```
elemento <- mi_vector[3]
```

Funciones de resumen: R proporciona funciones para calcular estadísticas resumidas, como la media (`mean()`), la mediana (`median()`), la desviación estándar (`sd()`), entre otras.

```
mean_resultado <- mean(mi_vector)
```

```
median_resultado <- median(mi_vector)
```

```
sd_resultado <- sd(mi_vector)
```

Operaciones lógicas: Se pueden realizar operaciones lógicas como AND (`&`), OR (`|`) y NOT (`!`) para comparar valores.

```
resultado_logico <- (x > 5) & (x < 20)
```

Las funciones en R son bloques de código que realizan una tarea específica y pueden

aceptar argumentos como entrada. R incluye una amplia variedad de funciones predefinidas para realizar tareas comunes, como cálculos estadísticos, manipulación de datos y visualización.

Los usuarios pueden definir sus propias funciones utilizando la palabra clave `function`.

Por ejemplo,

```
mi_funcion <- function(x) { return(x^2) }
```

Define una función llamada `mi_funcion` que devuelve el cuadrado de su argumento `x`.

Para llamar a una función, simplemente se escribe el nombre de la función seguido de paréntesis que pueden contener argumentos si es necesario. Por ejemplo, `resultado <-`

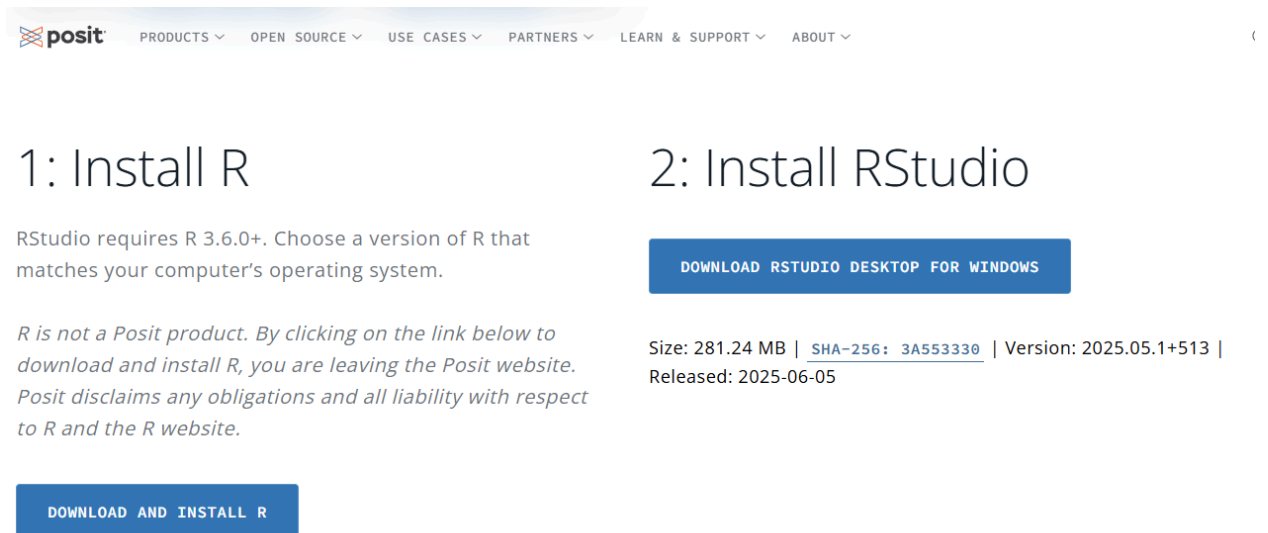
`mi_funcion(3)` llama a la función `mi_funcion` con el argumento 3 y almacena el resultado en la variable `resultado`.”

DESARROLLO

Descarga de RStudio

Nos dirigimos al link de descarga del lenguaje R y de la interfaz RStudio Desktop para trabajar con el lenguaje en un ambiente amigable

<https://www.rstudio.com/products/rstudio/download/>



posit PRODUCTS OPEN SOURCE USE CASES PARTNERS LEARN & SUPPORT ABOUT

1: Install R

RStudio requires R 3.6.0+. Choose a version of R that matches your computer's operating system.

R is not a Posit product. By clicking on the link below to download and install R, you are leaving the Posit website. Posit disclaims any obligations and all liability with respect to R and the R website.

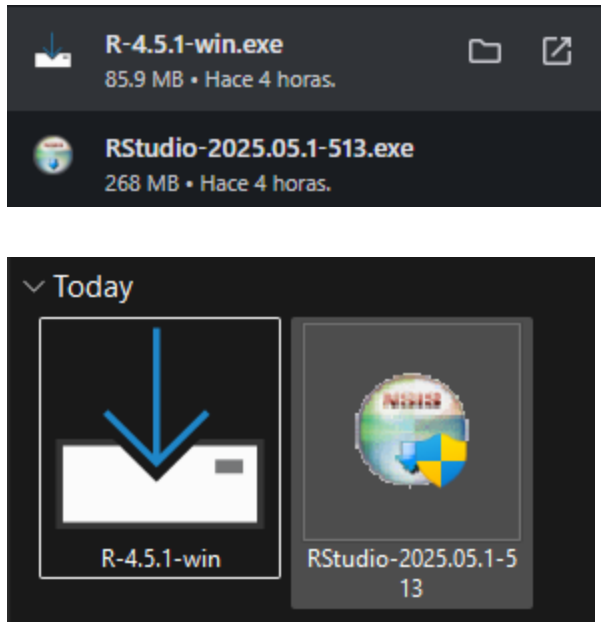
DOWNLOAD AND INSTALL R

2: Install RStudio

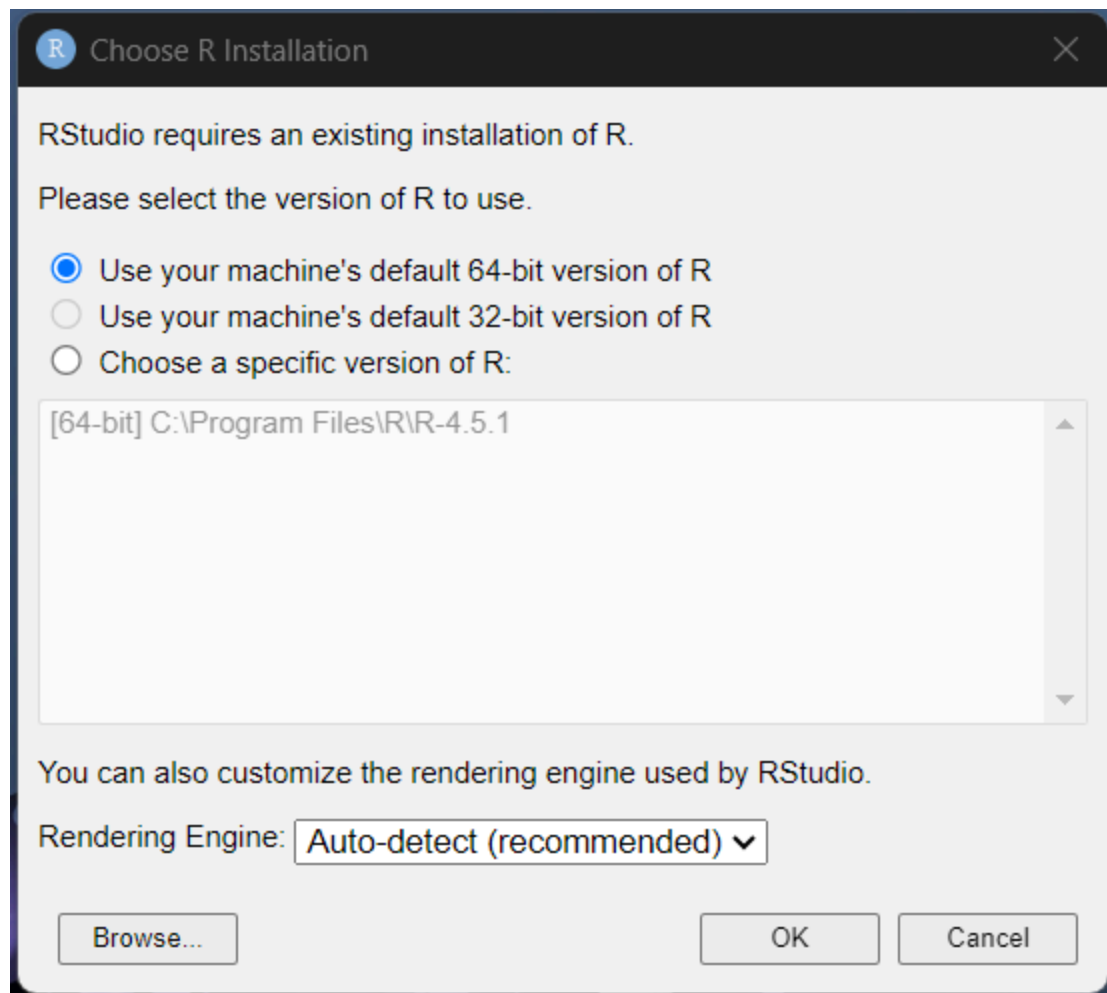
DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

Size: 281.24 MB | [SHA-256: 3A553330](#) | Version: 2025.05.1+513 | Released: 2025-06-05

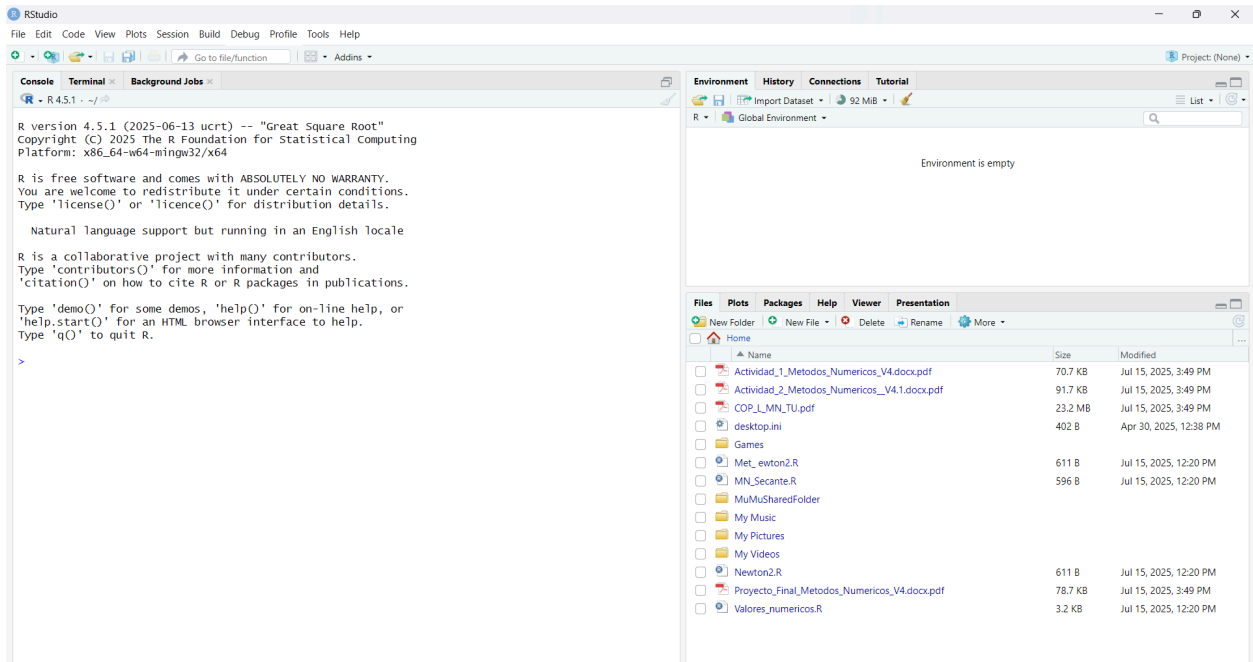
Comprobamos que ambos archivos estan descargados y procedemos primero con la instalación de R-4.5.1.win.exe que es la instalación del lenguaje R para Windows, posterior a la instalación se instala el RStudio también en su versión para Windows.



Una vez instalado R y RStudio nos pedirá una última configuración dentro de RStudio con la versión de R que deseamos utilizar, en mi caso selecciono 64bits ya que mi equipo de cómputo cuenta con esa arquitectura.

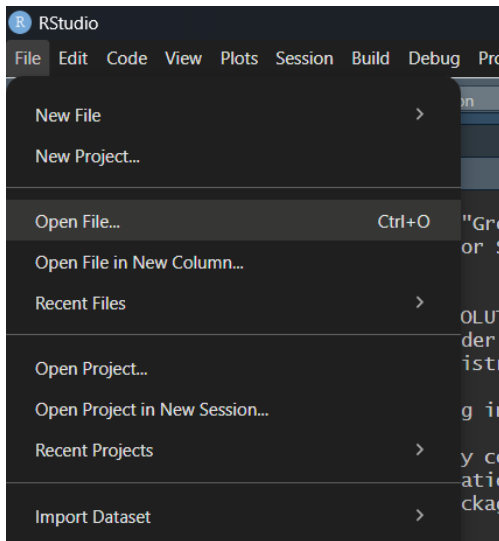


Aquí tenemos todo preparado RStudio con lenguaje R de 64 bits cargado y listo para trabajar.

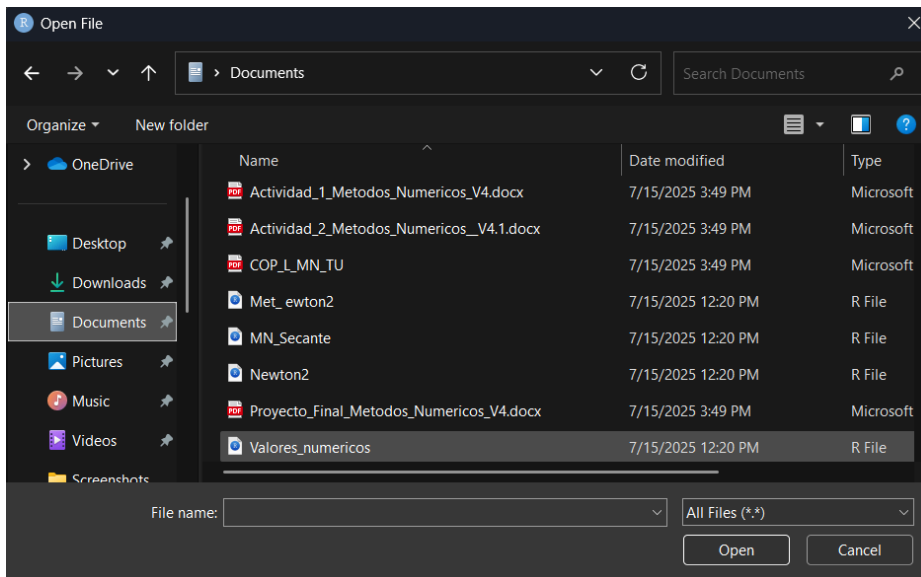


Carga de Valores_numéricos.R

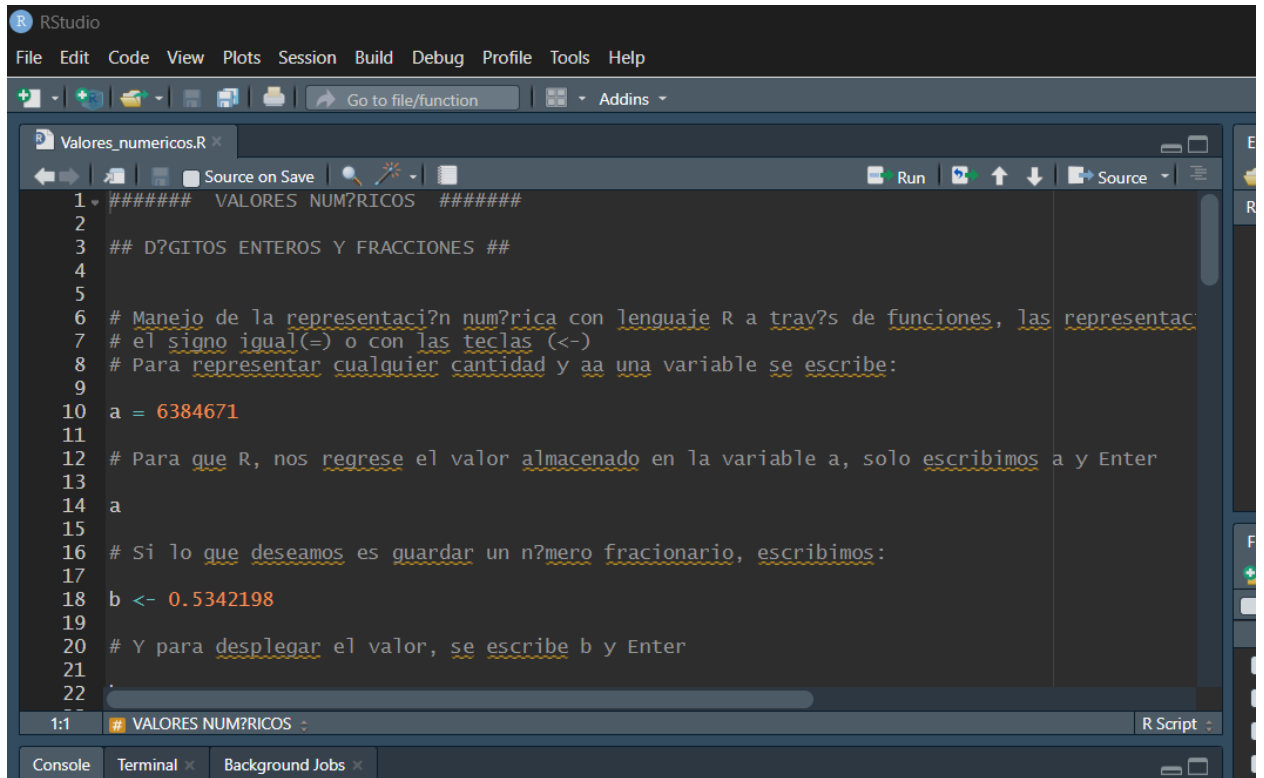
Para cargar el archivo Valores Numéricos nos dirigimos a File, Open File y seleccionamos el archivo Valores_numericos que nos compartieron para la actividad



Seleccionamos el archivo y abrimos



En la sección de arriba a la izquierda podemos ver como se ha cargado correctamente el archivo y se muestra la información del archivo



The screenshot shows the RStudio environment with a script editor open. The script is titled 'Valores_numericos.R' and contains the following R code:

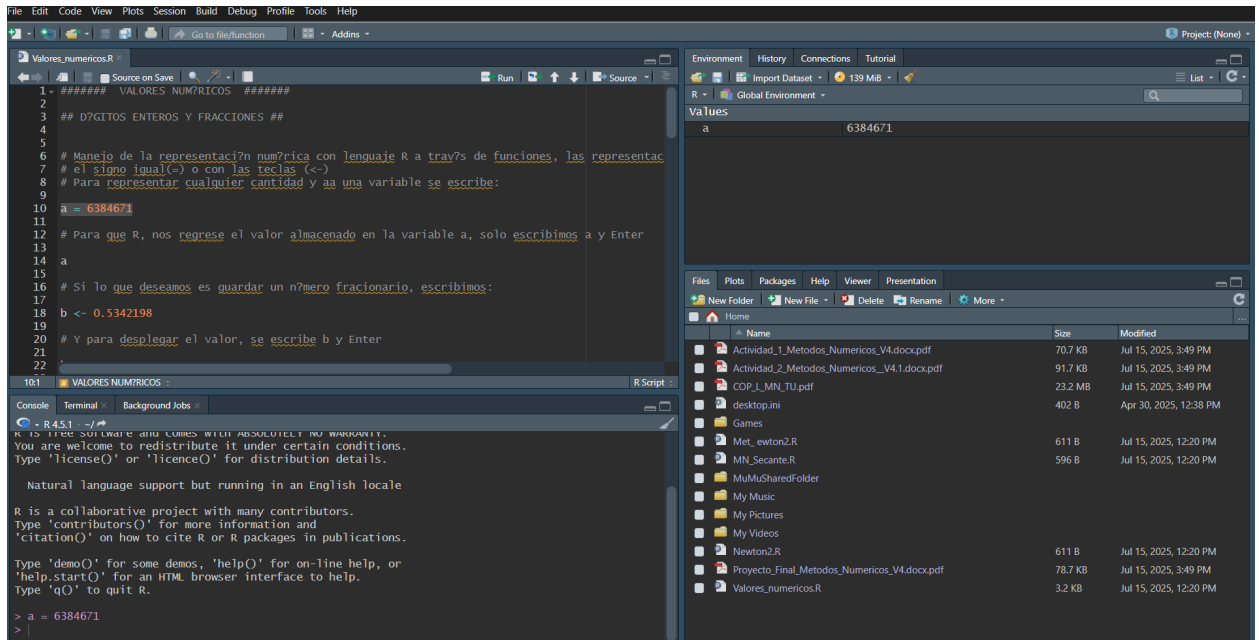
```
1 ##### VALORES NUMERICOS #####
2
3 ## DÍGITOS ENTEROS Y FRACCIONES ##
4
5
6 # Manejo de la representación numérica con lenguaje R a través de funciones, las representac
7 # el signo igual(=) o con las teclas (<=)
8 # Para representar cualquier cantidad y a una variable se escribe:
9
10 a = 6384671
11
12 # Para que R, nos regrese el valor almacenado en la variable a, solo escribimos a y Enter
13
14 a
15
16 # Si lo que deseamos es guardar un número fraccionario, escribimos:
17
18 b <- 0.5342198
19
20 # Y para desplegar el valor, se escribe b y Enter
21
22 .
```

The script is saved in the 'Source' pane. The 'Console' pane at the bottom shows the output of the script, which is the text 'VALORES NUMERICOS :'. The 'Terminal' and 'Background Jobs' panes are also visible at the bottom.

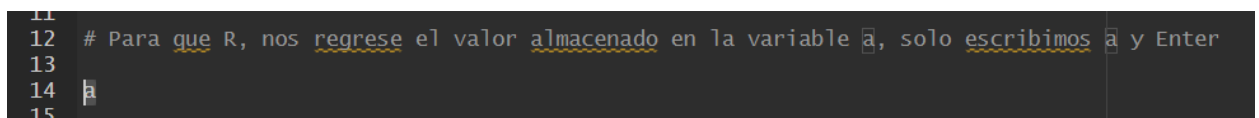
Ejecución de Valores_numéricos.R

Dígitos Enteros y Fracciones

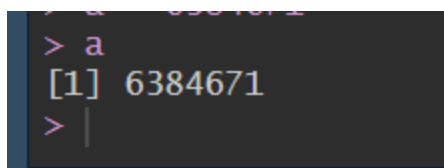
Asignamos la variable ‘a’ con el valor de 6384671 por medio del signo igual ‘=’



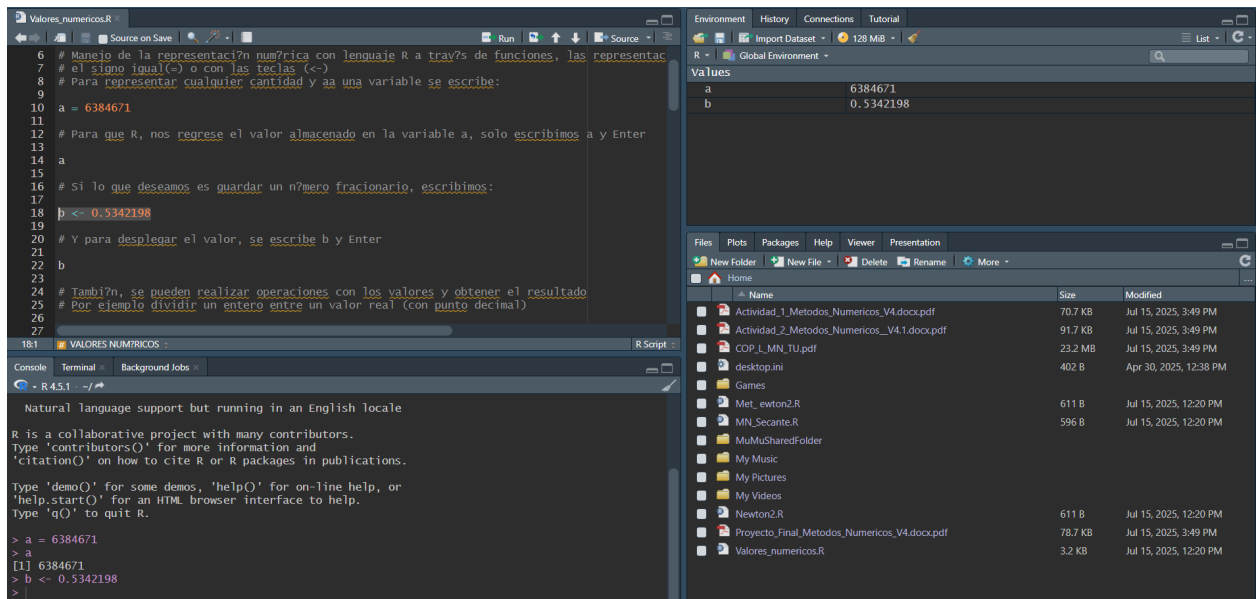
Para conocer el valor de la variable sólo la mencionamos por su nombre “a”



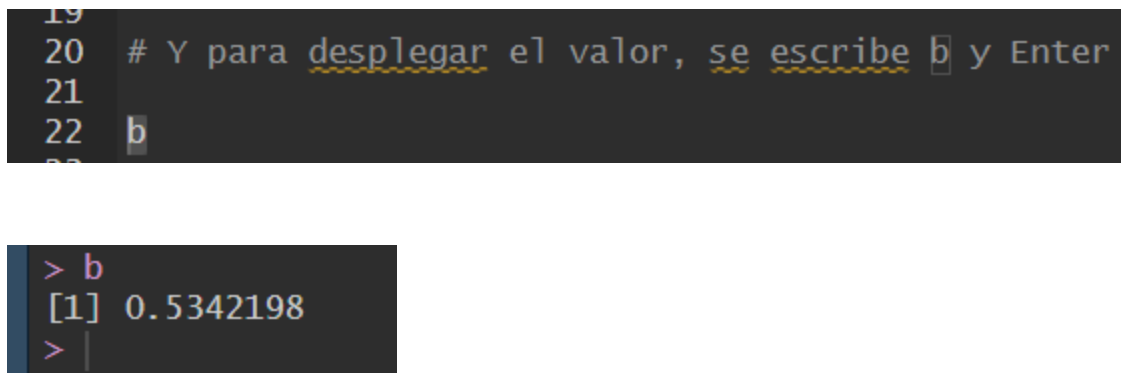
y la consola nos mostrará el valor que le asignamos previamente



Otra manera de asignar valores a las variables sería por medio de los signos ' \leftarrow ', en el ejemplo asignaremos a la variable 'b' un valor fraccionario 0.5342198



Retornamos el valor de 'b' escribiendo su nombre



En la siguiente imagen realizamos una operación de división de números enteros y nos muestra su resultado al correr la operación.


```

23
24 # También, se pueden realizar operaciones con los valores y obtener el resultado
25 # Por ejemplo dividir un entero entre un valor real (con punto decimal)
26
27 1/3.0
28

```

```

> 1/3.0
[1] 0.3333333
>

```

Si queremos modificar la cantidad de decimales que nos arroja el resultado, podemos utilizar la función `options()` y el modificador `digits=n` con la cantidad de decimales que queremos, para este ejemplo son 3 y observamos cómo se reducen a 3 los decimales.

```

29 # Se puede observar que el resultado nos regresa 7 decimales, pero se puede modificar esa ca
30 # Usando la función options() y el modificador digits = n; donde n es la cantidad de decimal
31
32 options(digits=3)
33 1/3.0
34

```

```

[1] 0.3333333
> options(digits=3)
> 1/3.0
[1] 0.333
>

```

La cantidad de decimales continuará bajo esta configuración de 3 decimales hasta que reiniciemos el software o modifiquemos nuevamente con `options()` y el modificador `digits=n`, modificamos nuevamente a sus 7 decimales

```

34
35 # La cantidad de decimales permanece con esa cantidad hasta que se modifique o se reinicie R
36
37 options(digits=7)
38 1/3.0
39

```

```
[1] 0.3333333
> options(digits=7)
> 1/3.0
[1] 0.3333333
> |
```

Redondeo

Para redondear utilizaremos la función `round(x,n)` donde `x` es el valor del número y `n` la cantidad de decimales, si omitimos `n` se descartan las decimales automáticamente como en el siguiente ejemplo:

```
43
44 # La función round(x,n); donde x es el valor y n es la cantidad de decimales, sin n es sin d
45 # Si tenemos un valor de 54.2 y lo redondeamos, obtenemos:
46
47 round(54.2)
48
```

```
[1] 54
> round(54.2)
[1] 54
> |
```

y utilizando la función `round(97.5684197, 2)` le estamos pidiendo que redondee a solo 2 decimales

```
48
49 # Si escribimos una cantidad con más decimales y le pedimos un número particular de decimale
50 # nos proporcionar ese número de decimales solicitado y redondeado con el siguiente dígito
51
52 round(97.5684197, 2)
53
```

```
> round(97.5684197, 2)
[1] 97.57
> |
```

Dígitos Significativos

Similar a la función anterior, existe la función `signifi(x,n)` que redondee el valor de `x` a `n` cifras significativas, si se omite `n` el valor default de `n=6` a diferencia del redondeo donde se omitían decimales.

```
56 # signif(x,n) redondea a x, con n dígitos significativos (default n=6)
57
58 signif(27.384956102)
```

```
> signif(27.384956102)
[1] 27.385
> |
```

```
53
54 ## DÍGITOS SIGNIFICATIVOS ##
55
56 # signif(x,n) redondea a x, con n dígitos significativos (default n=6)
57
58 signif(27.384956102)
59
60 signif(39.6429304521, 5)
61
```

```
[1] 27.385
> signif(39.6429304521, 5)
[1] 39.643
> |
```

```

54 ## D?GITOS SIGNIFICATIVOS ##
55
56 # signif(x,n) redondea a x, con n d?gitos significativos (default n=6)
57
58 signif(27.384956102)
59
60 signif(39.6429304521, 5)
61
62 signif(61.378045912, 2)
63

```

```

> signif(61.378045912, 2)
[1] 61
> |

```

```

54 ## D?GITOS SIGNIFICATIVOS ##
55
56 # signif(x,n) redondea a x, con n d?gitos significativos (default n=6)
57
58 signif(27.384956102)
59
60 signif(39.6429304521, 5)
61
62 signif(61.378045912, 2)
63
64 signif(316.6971243547, 3)
65

```

```

> signif(316.6971243547, 3)
[1] 317
> |

```

Definir Variables y Asignar Valores

Vamos a dale el valor exponencial de uno a la variable 'e'

```

67
68 e <- exp(1) # Asigna un valor a la variable (Base de los logaritmos naturales e = 2.71828
69 e          # Imprime el valor que tenga la variable

```

```
[1] 2.71828
> e <- exp(1)
> |
```

y vemos que correctamente se agrega el valor de 2.71828182845905 que es el valor aproximado del exponencial.

Values	
a	6384671
b	0.5342198
e	2.71828182845905

Mostramos el valor de la variable 'e' introduciendola y corriendo y nos devuelve el valor

```
66 ### DEFINIR VARIABLES Y ASIGNAR VALORES
67
68 e <- exp(1) # Asigna un valor a la variable e
69 e          # Imprime el valor de e
70
71 x = 0.005   # Tambi?n se puede usar '=' para asignar
```

```
> e <- exp(1)
> e
[1] 2.718282
> |
```

Asignamos una variable x con un valor de 0.005 ya sea con el signo '=' o '<-'

```

70
71 x = 0.005      # También se puede usar el símbolo <- en lugar de igual
72

```

e	2.718281828
x	0.005

Ahora a la variable x0 se le asignará una función

```

1 x = 0.005      # También se puede usar el símbolo <- en lugar de igual
2
3 x0 = e ** (2*x)      # Se la asigna una función a x0
4

```

```

> x = 0.005
> x0 = e ** (2*x)
>

```

Vamos a llamar una variable tex y asignarle una cadena de caracteres

```

74
75 tex = "El valor de x0 es: "      # A la variable tex, se le asigna una cadena
76

```

```

> x0 = e ** (2*x)
> tex = "El valor de x0 es: "
>

```

y por medio de la función cat() vamos a concatenar e imprimir en consola la variable tex

y la variable x0 mostrada como cadena de caracteres

```

9
7 cat(tex, x0)
8

```

```
cat(x0, x1)
El valor de x0 es: 1.01005
> |
```

Otro ejemplo de concatenar donde damos un nuevo valor a x0

```
80
81 x0 = 1
82
```

```
El valor de x
> x0 = 1
> |
```

Agregamos la variable x1 y le asignamos un valor

```
82
83 x1 = x0 - pi * x0 + 1
84
```

```
> x1 = x0 - pi * x0 + 1
> |
```

y lo llamamos para que se muestre el resultado

```
84
85 x1
86
```

```
> x1
[1] -1.141593
> |
```

Ahora por medio de la función cat() retornamos los valores de x0 y x1

```

5
7 cat("x0 =", x0, "\n", "x1 =", x1) # Si usamos "\n" se cambia de rengl?n
8

```

```

x1 = -1.141593
> cat("x0 =", x0, "\n", "x1 =", x1)
x0 = 1
x1 = -1.141593
> |

```

Definir Funciones y Pasar Parámetros

Ahora vamos a definir la variable 'd' con una función

```

90
91 d = function(a,b,c) b^2-4*a*c
92

```

```

x1 = -1.141593
> d = function(a,b,c) b^2-4*a*c
> |

```

Para poder llamar al resultado de esta función, tenemos que introducir la variable seguida de los parámetros que le asignaremos a las incógnitas de la función.

```

92
93 d(2,2,1)
94

```

siendo a=2, b=2, c=1 de la función


```

> d(2,2,1)
[1] -4
> |

```

Graficación de Funciones

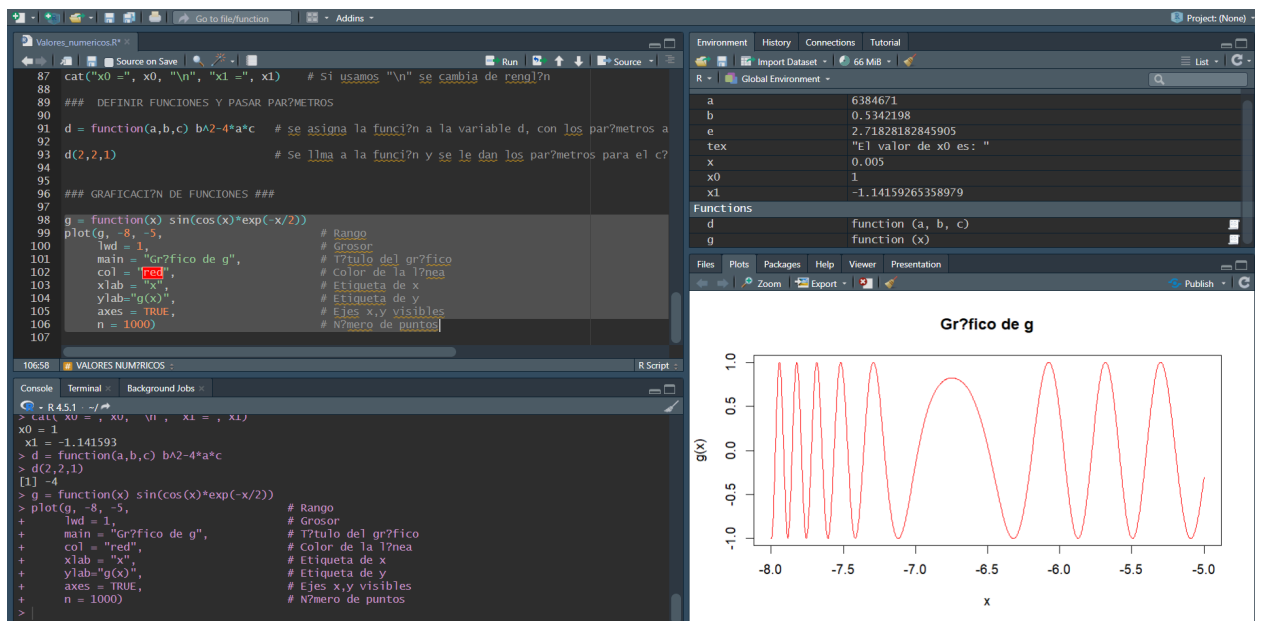
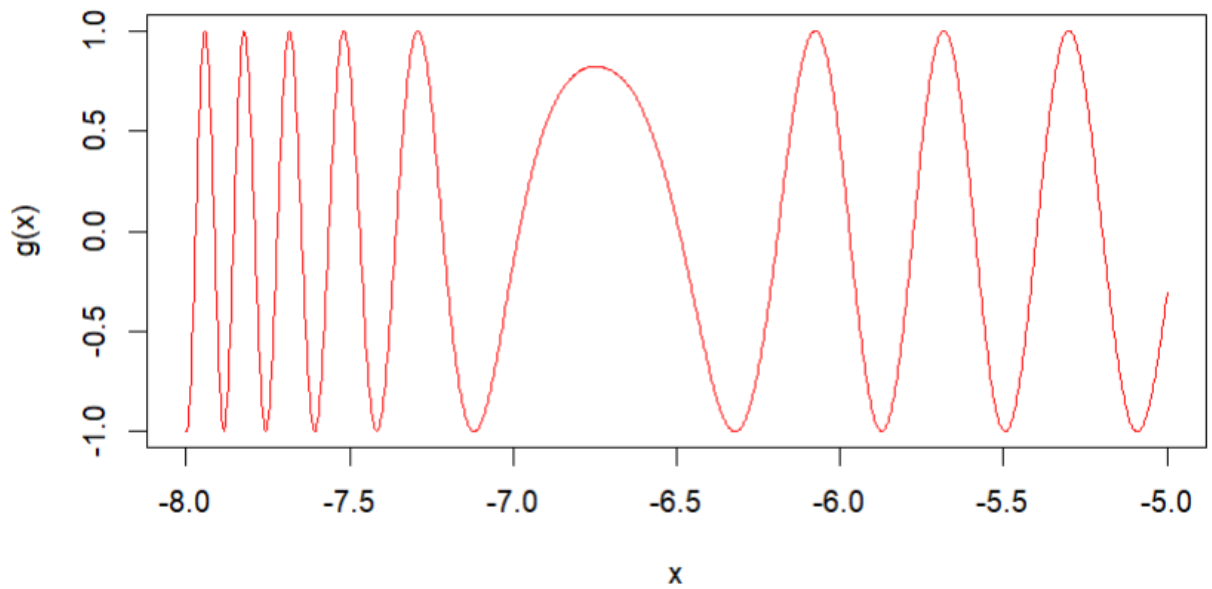
Vamos a graficar una función 'g' en un rango desde -8 hasta el -5 en el eje x que se llamará Gráfico de g con una línea de grosor 1 y color rojo y se evaluará mil veces para trazar mil puntos sobre la línea de la función.

```

96  ### GRAFICACIÓN DE FUNCIONES ###
97
98  g = function(x) sin(cos(x)*exp(-x/2))
99  plot(g, -8, -5,                                # Rango
100      lwd = 1,                                    # Grosor
101      main = "Gráfico de g",                      # Título del gráfico
102      col = "red",                                # Color de la línea
103      xlab = "x",                                  # Etiqueta de x
104      ylab="g(x)",                                # Etiqueta de y
105      axes = TRUE,                                # Ejes x,y visibles
106      n = 1000)                                   # Número de puntos
107

```

Gráfico de g



CONCLUSIÓN

En el desarrollo de esta tarea hemos logrado realizar la instalación de un ambiente amigable diseñado para poder trabajar con el lenguaje R, conocimos la estructura para trabajar con R y vimos muchas de las operaciones y funciones básicas con las que vamos a estar trabajando a lo largo de la materia.

Uno de los puntos de los que aprendimos que considero es de los más importantes para poder iniciar nuestros métodos numéricos es la graficación de la función con la que vamos a trabajar, pues si sabemos interpretar correctamente el comportamiento de la gráfica de la función podemos concluir rápidamente ciertos valores como si la función tiene o no solución, en qué punto es probable que se encuentre la solución o entre que puntos, lo que nos ayuda a determinar valores iniciales, etc.

Para mi esta mas que claro que hay que utilizar las herramientas que existen para poder facilitarnos la resolución de problemas, y el utilizar sistemas computacionales para la resolución de problemas, en lugar de utilizar métodos tradicionales de realizarlos a lápiz y papel es una gran ventaja que todos deberíamos de aprovechar, con eso minimizamos errores, y hacemos más eficiente nuestro trabajo.

REFERENCIAS

Noguera, I. B., & Noguera, I. B. (2025, 14 febrero). ¿Qué son los métodos numéricos?
Ingeniería Química Reviews.

<https://www.ingenieriaquimicareviews.com/2020/10/metodos-numericos.html>

Blasco, J. L. (2024, 4 abril). Introducción y primeros pasos con el lenguaje R.
OpenWebinars.net. <https://openwebinars.net/blog/introduccion-lenguaje-r/>