Universidad Corporativa Coppel | Programas Académicos

Ingeniería en Desarrollo de Software

Actividad | 1 | Análisis de Conceptos

Nombre del curso

Ingeniería en Desarrollo de Software

TUTOR: Miguel Angel Rodriguez Vega.

ALUMNO: Uziel de Jesús López Ornelas.

FECHA: 1 de Noviembre del 2024.

Índice

Introducción

Los métodos numéricos son aplicaciones de algoritmos mediante las cuales es posible formular y solucionar problemas matemáticos utilizando operaciones aritméticas menos complejas, esta definición nos da la bienvenida a esta materia en la que estaremos viendo y resolviendo varios ejercicios en el software de RStudio, en esta primera actividad descargaremos el programa antes mencionado para ejecutarlo y mandar capturas del proceso en el cual nosotros lo instalaremos en nuestros ordenadores, después abriremos un documento .txt donde se incluirán diferentes instrucciones las cuales las mandaremos al software para ejecutarlas, de esta manera nos familiarizaremos de poco en poco a este sistema, para lograr resolver las diferentes actividades que se harán en posteriores días. Todo esto será anexado con capturas de pantalla con explicaciones sencillas para que tengan coherencia en lo que se quiere explicar, se dará una conclusión final con lo que se ha aprendido en la actividad presentada para que tengamos mayor experiencia y conocimiento.

Descripción

Existen en el mercado varios programas con los que se pueden realizar análisis numéricos, por ejemplo, Octave, que es un lenguaje de programación científico. Este lenguaje tiene mucha similitud con las instrucciones y funcionamiento de MATLAB, incluso se dice que existe una parte del motor de MATLAB que está contenida en Octave. Este software es de libre distribución. Actualmente, uno de los programas informáticos para realizar análisis de datos en general es el Lenguaje R, ya que facilita el manejo de los datos. Además, es un software libre y de fácil comprensión. La interfaz del Lenguaje R no es muy elegante, por lo que se recomienda usarlo con el software RStudio, la cual es de distribución gratuita y sencilla de utilizar. Antes de instalarla, debemos de instalar la versión de Lenguaje R para cualquier plataforma: Linux, Windows, MacOs, etc. El software funciona bajo el modelo de ventanas y usa una serie de comandos, que pueden ser manejados con versatilidad y sencillez.

Justificación

Como ya se mencionó anteriormente los métodos numéricos son necesarios para diferentes áreas que pueden abarcar las matemáticas, la ingeniería e inclusive la ciencia, ya que gracias a estos nos permite resolver problemas complejos de manera aproximada pero muy precisa, con esto estamos diciendo que su utilidad va más allá de lo que en ocasiones nos imaginamos, resolver problemas complejos, llegar a resultados de una aproximación que puede no ser exacta pero muy cercana al valor real y la eficiencia con la que se trabaja con este tupo de métodos son sin duda una de las pocas cosas que se pueden realizar en este ámbito de las matemáticas y la ingeniería. El software RStudio en una excelente opción si queremos trabajar con este tipo de cálculos en el lenguaje de programación R ya que nos otorga muchas herramientas que sin duda son un gran apoyo en lo que respecta a la resolución de problemas en los que se tenga que utilizar métodos y ecuaciones relacionadas con los métodos numéricos.

Desarrollo

Descarga de Rstudio:

Para descargar el software de Rstudio es importante acceder al link de dicho programa, en este caso tenemos el siguiente que nos lo proporciona la Universidad:

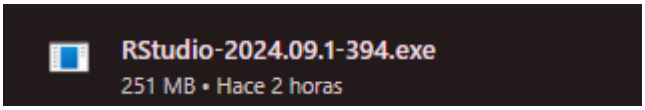
Recursos
RStudio: https://www.rstudio.com/products/rstudio/download/

Cuando damos clic en el enlace nos dirige automáticamente a la Página principal:

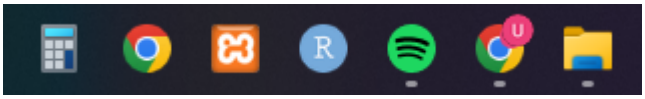
En este apartado tendremos la posibilidad de descargar aquel que se ajuste a nuestro sistema operativo de preferencia o el que nos interese más, en mi caso será el de Windows:

Sistema operativo	Descargar	Tamaño	SHA-256
Windows 10/11	RSTUDIO-2024.09.1-394.EXE	263,71 MB	2C3CF96A
macOS 12+	RSTUDIO-2024.09.1-394.DMG	613,31 MB	F1AAC1C8
Ubuntu 20/Debian 11	RSTUDIO-2024.09.1-394-AMD64.DEB	202,43 MB	2890EDE2
Ubuntu 22/Debian 12	RSTUDIO-2024.09.1-394-AMD64.DEB	202,43 MB	94896D5D

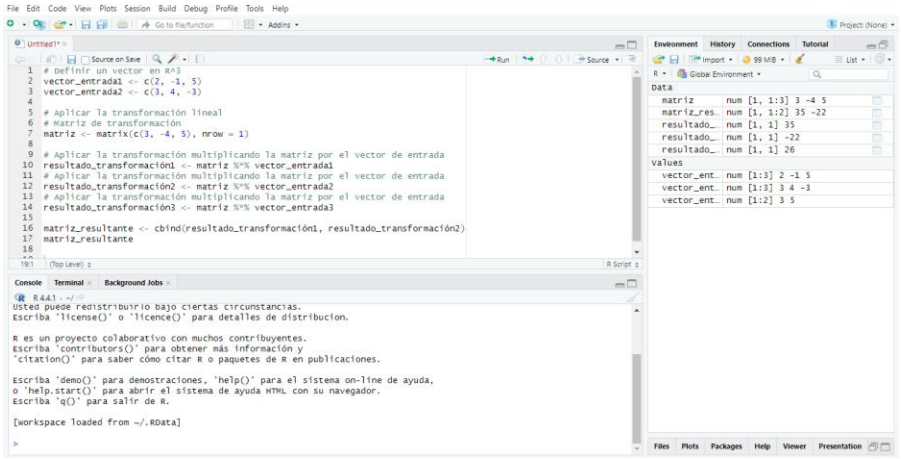
Seleccionaremos el link para comenzar la descarga y se nos descargara el programa .exe para su ejecución:



Al seleccionarlo nos pedirá que ubiquemos el archivo de origen para que se incorpore en nuestro ordenador para después finalizar la instalación, después de todo esto lo tendremos en nuestro escritorio:



Al abrirlo nos mandara a la página principal del programa, en este caso tenía una actividad de otra materia hecha por lo que se guardó en automático, con estos pasos ya tendríamos listo nuestro programa:



Carga de Valores_numericos.R:

Para ello seleccionamos el archivo que nos proporciona la Universidad en la actividad correspondiente:

> **Archivo Valores Numéricos R:**
> http://umi.edu.mx/coppel/IDS/plataforma/files/programasenr/Valores_numericos.R

Cuando seleccionamos el archivo se nos abrirá en formato .txt en el cual observaremos instrucciones de cómo utilizar Rstudio:

```
#######  VALORES NUM□RICOS  #######

## D□GITOS ENTEROS Y FRACCIONES ##


# Manejo de la representaci□n num□rica con lenguaje R a trav□s de funciones, las representaciones pueden ser con:
# el signo igual(=) o con las teclas (<-)
# Para representar cualquier cantidad y aa una variable se escribe:

a = 6384671

# Para que R, nos regrese el valor almacenado en la variable a, solo escribimos a y Enter

a

# Si lo que deseamos es guardar un n□mero fracionario, escribimos:

b <- 0.5342198

# Y para desplegar el valor, se escribe b y Enter

b

# Tambi□n, se pueden realizar operaciones con los valores y obtener el resultado
# Por ejemplo dividir un entero entre un valor real (con punto decimal)

1/3.0

# Se puede observar que el resultado nos regresa 7 decimales, pero se puede modificar esa cantidad de decimales
# Usando la funci□n options() y el modificador digits = n; donde n es la cantidad de decimales

options(digits=3)
1/3.0

# La cantidad de decimales permanece con esa cantidad hasta que se modifique o se reinicie R

options(digits=7)
1/3.0
```

```
## REDONDEO ##

# La funci□n round(x,n); donde x es el valor y n es la cantidad de decimales, sin n es sin decimales
# Si tenemos un valor de 54.2 y lo redondeamos, obtenemos:

round(54.2)

# Si escribimos una cantidad con m□s decimales y le pedimos un n□mero particular de decimales,
# nos proporcionar□ ese n□mero de decimaeles solicitado y redondeado con el siguiente d□gito.

round(97.5684197, 2)
```

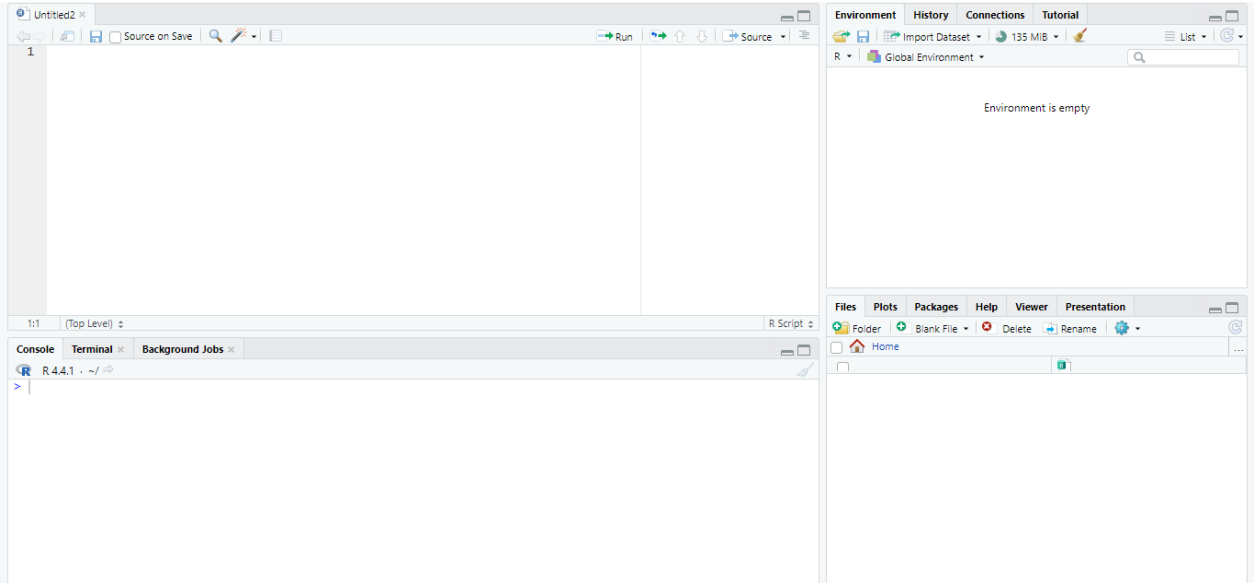
```
## D◆GITOS SIGNIFICATIVOS ##

# signif(x,n)  redondea a x, con n d◆gitos significativsos (default n=6)

signif(27.384956102)

signif(39.6429304521, 5)

signif(61.378045912, 2)

signif(316.6971243547, 3)

### DEFINIR VARIABLES Y ASIGNAR VALORES ###

e <- exp(1)    # Asigna un valor a la variable (Base de los logaritmos naturales e = 2.718281)
e              # Imprime el valor que tenga la variable

x = 0.005     # Tambi◆n se puede usar el simbolo <- en lugar de igual

x0 = e ** (2*x)         #  Se la asigna una funci◆n a x0

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

cat(tex, x0)                # Se obtine los resultados con la instrucci◆n cat, que concatena y convierte a string

# Otro ejemplo:

x0 = 1

x1 = x0 - pi * x0 + 1

x1

cat("x0 =", x0, "\n", "x1 =", x1)    # Si usamos "\n" se cambia de rengl◆n
```

```
###  DEFINIR FUNCIONES Y PASAR PAR◆METROS

d = function(a,b,c) b^2-4*a*c   # se asigna la funci◆n a la variable d, con los par◆metros a,b,c; y la funci◆n b cuadrada menos 4 por a por c

d(2,2,1)                        # Se llma a la funci◆n y se le dan los par◆metros para el c◆lculo y regresa el resultado


### GRAFICACI◆N DE FUNCIONES ###

g = function(x) sin(cos(x)*exp(-x/2))
plot(g, -8, -5,                         # Rango
     lwd = 1,                           # Grosor
     main = "Gr◆fico de g",              # T◆tulo del gr◆fico
     col = "red",                       # Color de la l◆nea
     xlab = "x",                        # Etiqueta de x
     ylab="g(x)",                       # Etiqueta de y
     axes = TRUE,                       # Ejes x,y visibles
     n = 1000)                          # N◆mero de puntos
```

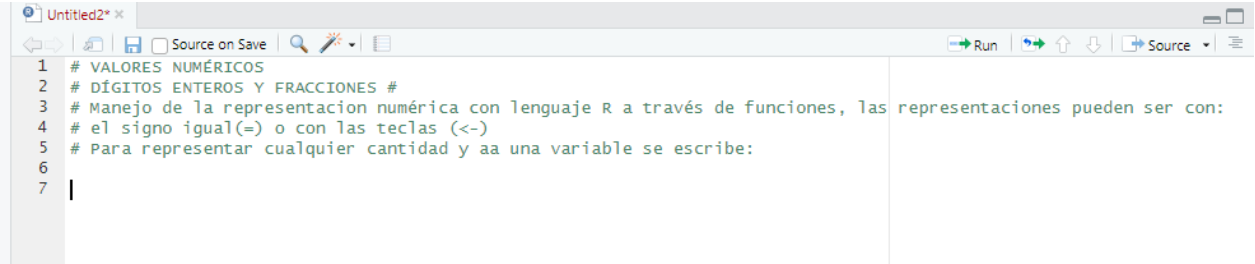
El archivo abre en una página de internet, pero en el siguiente paso se anexará la información al programa junto con la ejecución de la instrucción.

Ejecución de Valores_numericos.R:

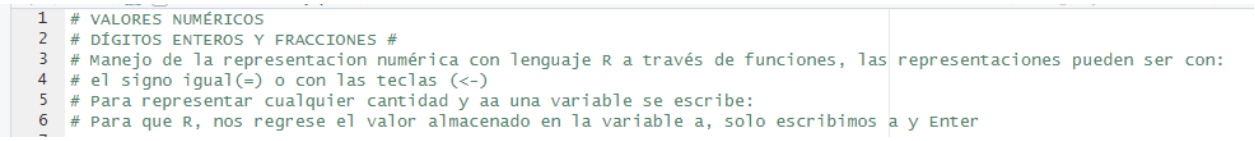
Abriremos el programa para iniciar con las instrucciones:



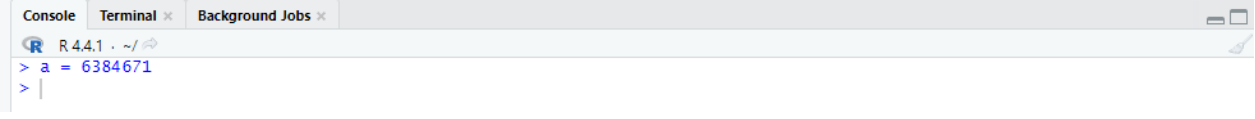
Empezaremos con la primera instrucción, de dígitos enteros y fracciones:

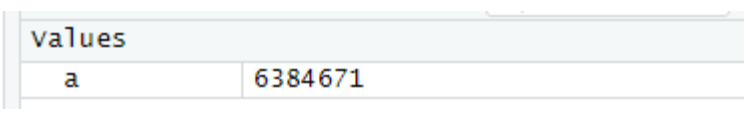


Crearemos nuestra ejecución de la primera instrucción:



Para hacer que este comando se ejecute presionamos control + enter para que se muestre en la ventana inferior izquierda, es decir, en la consola:





Pasamos con la siguiente instrucción:

```
10  # Si lo que deseamos es guardar un número fracionario, escribimos:
11  # Y para desplegar el valor, se escribe b y Enter
```

Ejecutamos la instrucción de acuerdo a que nos dice para que aparezca en la consola:

```
> b <- 0.5342198
```

b	0.5342198

La siguiente instrucción es la siguiente:

```
15  # También, se pueden realizar operaciones con los valores y obtener el resultado
16  # Por ejemplo dividir un entero entre un valor real (con punto decimal)
```

Al ejecutarla nos arroja en la consola con el comando y el resultado:

```
> 1/3.0
[1] 0.3333333
```

Continuaremos con la siguiente instrucción:

```
20  # Se puede observar que el resultado nos regresa 7 decimales, pero se puede modificar esa cantidad de decimales
21  # Usando la función options() y el modificador digits = n; donde n es la cantidad de decimales
```

Daremos la siguiente indicación en la consola, como podemos observar los decimales, donde anteriormente eran 7, ahora con el nuevo comando se convirtieron en 3:

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

La siguiente indicación es la siguiente:

```
# La cantidad de decimales permanece con esa cantidad hasta que se modifique o se reinicie R
```

La ejecución consta de que el número decimal puede ser devuelto al original:

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

Continuamos con la siguiente instrucción que en este caso sería “Redondeo”:

```
# REDONDEO #
# La función round(x,n); donde x es el valor y n es la cantidad de decimales, sin n es sin decimales
# Si tenemos un valor de 54.2 y lo redondeamos, obtenemos:
```

Y hacemos lo mismo, ejecutamos la acción de la instrucción, esta vez se redondea el número, en este caso como el 54.2 es más cercano al “54” que al “55” es por ello que se redondea a ese número:

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

Seguiremos con las siguientes instrucciones que se nos da la actividad:

```
# Si escribimos una cantidad con más decimales y le pedimos un número particular de decimales,
# nos proporcionará ese número de decimaeles solicitado y redondeado con el siguiente dígito.
```

De acuerdo a la instrucción generaremos la acción, en donde al igual que la instrucción anterior redondea el número a dos decimales ya que el “2” indica el número de decimales que va a tener:

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

La instrucción siguiente nos indica realizar los “Dígitos Significativos”:

```
# DÍGITOS SIGNIFICATIVOS #
# signif(x,n)  redondea a x, con n dígitos significativsos (default n=6)
```

Con los siguientes números de ejemplo:

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

> signif(39.6429304521, 5)
[1] 39.643

> signif(61.378045912, 2)
[1] 61

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

Ahora definiremos variables y les asignaremos valores:

```
# DEFINIR VARIABLES Y ASIGNAR VALORES #
```

Pasaremos a realizar la instrucción:

```
# Asigna un valor a la variable (Base de los logaritmos naturales e = 2.718281)
# Imprime el valor que tenga la variable
```

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

También se puede realizar de la siguiente manera:

```
# También se puede usar el simbolo <- en lugar de igual
```

```
> x <- 0.005
```

Asignaremos valores a las funciones:

```
#  Se la asigna una función a x0
```

```
x0 = e ** (2*x)
```

A la variable tex se le asigna una cadena:

```
# A la variable tex, se le asigna una cadena

tex = "El valor de x0 es: "
```

Se pueden obtener resultados con la instrucción cat:

```
# Se obtine los resultados con la instrucción cat, que concatena y convierte a string

cat(tex, x0)
```

Aquí podemos visualizar un ejemplo similar

```
# Otro ejemplo:

x0 = 1

x1 = x0 - pi * x0 + 1

x1

# Si usamos "\n" se cambia de renglón

cat("x0 =", x0, "\n", "x1 =", x1)
```

```
x0 = 1.01005
 x1 = -1.141593
```

También podemos definir funciones y pasar parámetros:

```
# DEFINIR FUNCIONES Y PASAR PARÁMETROS
# se asigna la función a la variable d, con los parámetros a,b,c; y la función b cuadrada menos 4 por a por c

d = function(a,b,c) b^2-4*a*c
```

```
# Se llama a la función y se le dan los parámetros para el cálculo y regresa el resultado

d(2,2,1)
```

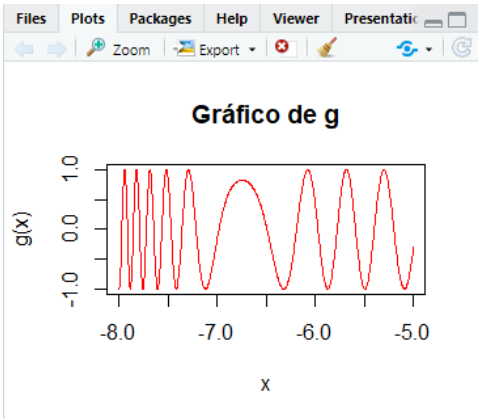
```
> d = function(a,b,c) b^2-4*a*c
>
> d(2,2,1)
[1] -4
>
```

Ahora toca graficar las funciones para eso tomaremos diferentes parámetros:

```
# GRAFICACIÓN DE FUNCIONES #

g = function(x) sin(cos(x)*exp(-x/2))
plot(g, -8, -5,                          # Rango
     lwd = 1,                            # Grosor
     main = "Gráfico de g",              # Título del gráfico
     col = "red",                        # Color de la línea
     xlab = "x",                         # Etiqueta de x
     ylab="g(x)",                        # Etiqueta de y
     axes = TRUE,                        # Ejes x,y visibles
     n = 1000)                           # Número de puntos
```

Y este es el resultado que podemos apreciar en el software, tanto la respuesta de la consola como la gráfica que esta genera:

```
> g = function(x) sin(cos(x)*exp(-x/2))
> plot(g, -8, -5,
+      lwd = 1,
+      main = "Gráfico de g",
+      col = "red",
+      xlab = "x",
+      ylab="g(x)",
+      axes = TRUE,
+      n = 1000)
```



Conclusión

En esta primera actividad tenemos varias instrucciones en las que nosotros con el software de RStudio tenemos que poner en práctica, desde como descargarlo, ejecutarlo, otorgarle una carpeta de origen, finalizar con la descarga hasta familiarizarnos con los términos antes mencionados para colocar todos los scripts en la ventana correspondiente y después de ello verificar en la consola de resultados la ejecución del comando de acuerdo a la sintaxis que había colocado anteriormente, sin duda una gran herramienta en la que nosotros nos podemos apoyar para realizar este tipo de trabajos, pero lo más importante es que en este mismo software podemos crear graficas en las que le podemos asignar valores que nosotros consideremos apropiados, claro todo esto sin la utilización de una ecuación como problema, solo son datos que nosotros colocaremos para crearla con nuestros criterios, desde el color de la línea, los diferentes parámetros, los ejes, y demás son al igual unas de las muchas opciones que tenemos para lograr personalizar nuestras gráficas para que cada una tenga una identidad propia por así decirlo.

Link para GitHub:

https://github.com/Leyzu-Ing/M-todos-Num-ricos.git

Referencias

Rafa. (2020, 25 febrero). *Qué es R y RStudio?* Rafa González Gouveia. https://gonzalezgouveia.com/que-es-r-y-rstudio/