Métodos Cuantitativos de la Administración

Lectura Nr. 3

Introducción a R para Programación Lineal

Profesor:
Ricardo Caballero, M.Sc.
✉ ricardo.caballero@utp.ac.pa

Fundamentos básicos: R y RStudio

R es un lenguaje de programación y un entorno de software libre para la computación estadística y gráficos respaldados por la Fundación R para Computación Estadística.

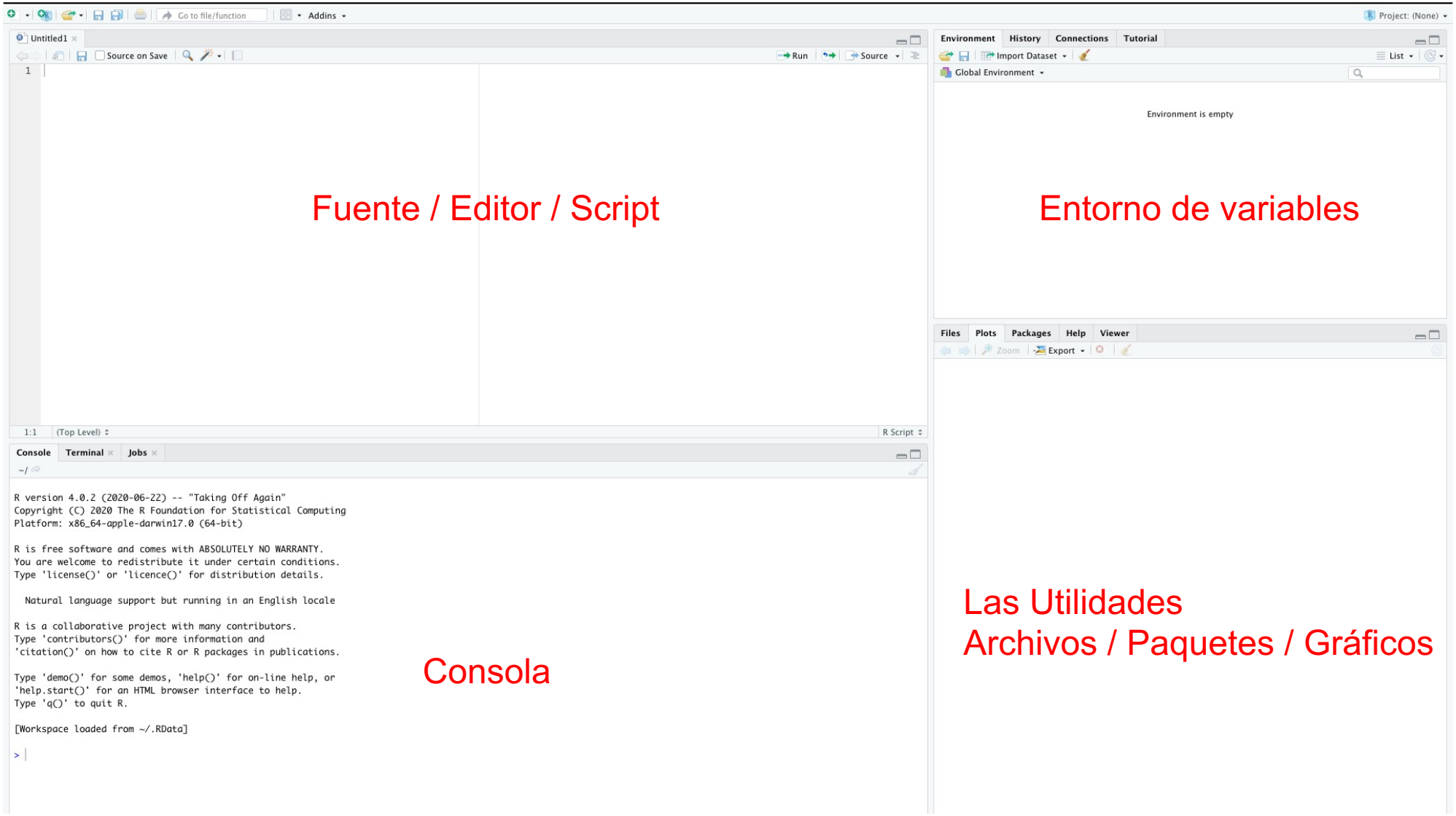
- Lenguaje de programación enfocado en el análisis gráfico y análisis estafístico

- R Studio es un interfaz gráfico para el usuario que permite
  - Escribir, editar y guardar código
  - Generar, ver gráficos
  - Administrar archivos, objetos y conjunto de datos

Para descargar R y Rstudio utilizar los siguientes links:

https://www.r-project.org/
https://rstudio.com/products/rstudio/download/

R Studio y sus partes



Programar con R: Definición de variable e impresión

<- Operador de asignación: asigna un valor a un símbolo

Print() Imprime el valor de una variable

Ejemplo:

Objeto tipo numérico considerado un vector →

```
x <- 1
print(x)
[1] 1
```

O también puede escribir

```
x <- 1
x
[1] 1
```

← No necesariamente se debe usar print para imprimir el valor. R guarda el valor de la variable con <- y al escribir la variable y dar click en ENTER se imprime el valor previamente guardado

Programar con R: Definición de variable e impresión

<- Operador de asignación: asigna un valor a un símbolo

Print() Imprime el valor de una variable

Ejemplo:

Objeto tipo caracter considerado un vector →

```
mensaje <- "hola"
mensaje
[1] "hola"
```

Se utiliza “” para escribir objeto tipo caracter o string

Programar con R: Uso de comentarios

Se utiliza para agregar comentarios

Ejemplo:

```
x <- 5
x
[1] 1
```

#se asigna variable
se imprime automáticamente

Vector

Número

El [1] indica que x es un vector y que 5 es el primer elemento del vector

Programar con R: Uso del operador :

: Se utiliza para crear una secuencia de valores x enteros

Ejemplo:

```
x <- 1:20
x
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Objetos y atributos

Todas las cosas que se escriban o manipulen en R se denominan **objetos**

Los objetos pueden contener cualquier tipo de data como:

- Character (string)
- Numerico (números reales)
- Entero (integer)
- Complejo
- Lógico (True/False)

El objeto más basico es un vector

→ **Un vector solamente puede contener objetos de una misma clase**

Excepción → Listas (contienen conjunto de números, caracteres, etc)

Atributos

- Nombres
- Dimensiones (matrices y arreglos)
- Clases
- Length
- Otros

Vectores y listas

C() Función utilizada para crear vectores de objetos
c – concatenar = union de objetos

Ejemplo:

```
x <- c(5,6)
x
[1] 5 6
```

← Concatenar objetos numéricos de valor 5 y 6

← Vector 5 y 6

Vectores y listas

C() Función utilizada para crear vectores de objetos
c – concatenar = union de objetos

Ejemplo:

```
x <- c(5,6)
x
[1] 5 6
```

Concatenar objetos numéricos de valor 5 y 6

Vector 5 y 6

Ejemplo:

El valor por default es 0

```
x <- vector("numeric", length = 10)
x
[1] 0 0 0 0 0 0 0 0 0 0
```

Matrices

- Utilizan un tipo especial de vector
- Son vectores con **atributos dimensionales**

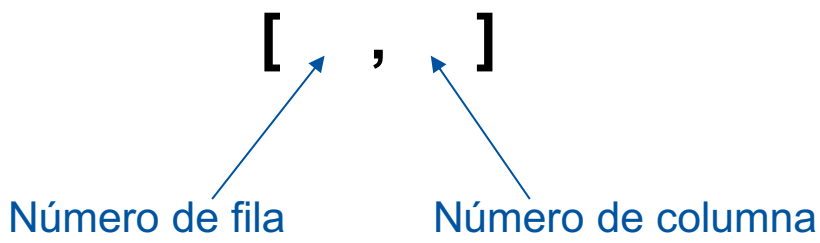
Un atributo dimensional es en sí un vector integer (entero) de longitude (length) 2
Esto quiere decir (nrow, ncol)

matrix(nrow = , ncol =) Define una matriz con n cantidad de filas y columnas

Ejemplo:

```
m <- matrix(nrow = 2, ncol = 3)
m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

NA significa valor no asignado (Not Assigned)



Matrices: dim() y attributes()

dim() Establece la dimensión de un objeto

attributes() Utilizado para obtener todos los atributos de los datos

Ejemplo:

```
attributes(m)
$dim
[1] 2 3
```

```
dim(m)
[1] 2 3
```

$ = Subconjunto de lista, binario

Matrices

matrix(nrow = , ncol =) Define una matriz con n cantidad de filas y columnas

Ejemplo:

```
m <- matrix(1:6, nrow=2, ncol=3)
m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

También se pueden crear matrices creando atributos dimensionales

```
 m <- 1:10
dim(m) <- c(2,5)
 m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Matrices: cbind y rbind

cbind Es una función utilizada se usa para combinar vectores, matrices o conjunto de datos por columnas

rbind Es una función utilizada se usa para combinar vectores, matrices o conjunto de datos por filas

Ejemplo:

```
 x <- 1:3
y <- 10:12

cbind(x,y)
```

```
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
```

```
 x <- 1:3
y <- 10:12

rbind(x,y)
```

```
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Matrices: byrow=TRUE

byrow=TRUE
indica que **la matriz debe ser llenada por filas**.

byrow=FALSE
indica que la matriz debe ser llenada por columnas (predeterminado).

Data.frame

La función data.frame() crea marcos de datos, colecciones de variables estrechamente acopladas que comparten muchas de las propiedades de las matrices y las listas, utilizadas como la estructura de datos fundamental por la mayoría del software de modelado de R.

```
data.frame(…, row.names = NULL, check.rows = FALSE,
           check.names = TRUE, fix.empty.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())
```

Uso de data.frame

Defina los valores para las siguientes variables: nombre, fecha de nacimiento, sexo, edad. Utilice cuatro valores para cada variable y concatenelos en una tabla

```
nombre <- c("Maria","Luis","Juan","Carla")
fecha_nacimiento <- c("26/09/1990", "13/02/2000", "15/04/1993", "20/05/1995")
sexo <- c("F", "M", "M", "F")
edad <- c("31", "22", "28", "26")
estudiante <- data.frame(nombre,fecha_nacimiento, sexo, edad)

estudiante
  nombre fecha_nacimiento sexo edad
1  Maria       26/09/1990         F   31
2   Luis       13/02/2000         M   22
3   Juan       15/04/1993         M   28
4  Carla       20/05/1995         F   26
```

Como se filtran los datos de una tabla

```
 nombre fecha_nacimiento sexo edad
1  Maria       26/09/1990    F   31
2   Luis       13/02/2000    M   22
3   Juan       15/04/1993    M   28
4   Carla      20/05/1995    F   26
```

```
estudiante[2,3]
[1] "M”
```

Filtrar valor de la segunda fila tercera columna

```
estudiante[,3]
[1] "F" "M" "M" "F”
```

Filtrar valores de la tercera columna

```
estudiante$edad
[1] "31" "22" "28" "26”
```

Filtrar valores del subconjunto de lista o columna edad

```
estudiante$edad[3]
[1] "28"
```

Filtrar tercer valor de la columna edad

Limpiar la consola

Sí deseas limpiar todo el código escrito en la consola e iniciar desde cero utiliza

Ctrl + L

LpSolve

Paquete utilizado para resolver problemas de programación lineal y programación entera

Los **paquetes** también se pueden instalar desde la línea de comandos de R.
Este es un enfoque más general que funcionará en todos los entornos.

La instalación del paquete requiere un solo comando:

El paquete lpSolve R:

install.packages("lpSolve")

Instalar el paquete no es suficiente. También debe cargarse en el espacio de memoria R antes de que pueda usarse. Esto se puede hacer con el siguiente comando:

library ("lpSolve")

Para más información puede referirse al siguiente link
https://www.rdocumentation.org/

Argumentos del paquete LpSolve

- **direction**
  indica la dirección de la optimización: "mín."(por default) o "máx."

- **objective.in**
  Vector numérico de los coeficientes de la función objetivo

- **const.mat**
  Matriz de los coeficientes numéricos de las restricciones, una fila por restricción, una columna por variable (a menos que use transpose.constraints = FALSE).

- **const.dir**
  Vector de cadena de caracteres que dan la dirección de la restricción: cada valor debe ser uno de los siguientes "<," "<=," "=," "==," ">," o ">=".

- **const.rhs**
  Vector de valores numéricos para la parte derecha de las restricciones (los recursos)

Para más información puede referirse al siguiente link
https://www.rdocumentation.org/

Ejemplo 1: Problema de maximización con R

Dada la siguiente formulación

$$Max\ f(x,y) = 2x + 3y$$
$$x + y \leq 3$$
$$x + y \geq 0$$

Resuelve el problema de PL utilizando R

Referencia

https://www.supplychaindataanalytics.com/es/resolver-un-problema-de-programacion-lineal-simple-usando-lpsolve-en-r/

Ejemplo 1: Problema de maximización con R

```
library(lpSolve)
f.obj <- c(2,3)                                  # vector de coeficiente de la función objetivo
f.con <- matrix(c(1,1),nrow=1,byrow=TRUE)        # matriz de coeficientes para la matriz de restricciones
f.dir <- c("<=")                                 # vector de dirección de la restricción
f.rhs <- c(3)                                    # valores de la restricción
solution <- lp("max",f.obj,f.con,f.dir,f.rhs)
solution
solution$solution
```

```
Success: the objective function is 9

solution$solution
[1] 0 3
```

Referencia

https://www.supplychaindataanalytics.com/es/resolver-un-problema-de-programacion-lineal-simple-usando-lpsolve-en-r/

Ejemplo 2: Problema de maximización con R

Furniture City fabrica mesas y sillas de bajo precio. El proceso de fabricación de cada producto se parece en que ambos requieren cierto número de horas de trabajo de carpintería, así como cierto número de horas de trabajo en el departamento de pintura y barnizado. Cada mesa requiere de 4 horas de carpintería y 2 horas en el taller de pintura y barnizado. Cada silla requiere de 3 horas de carpintería, y 1 hora en la pintura y barnizado. Durante el periodo de producción actual, hay 240 horas de tiempo de carpintería disponibles, así como 100 horas de tiempo disponibles en pintura y barnizado. Cada mesa vendida genera una utilidad de $70; cada silla fabricada se vende con una utilidad de $50.

El problema de Furniture City es determinar la mejor combinación posible de mesas y sillas a fabricar, con la finalidad de alcanzar la utilidad máxima. La empresa desea que esta situación de mezcla de producción se formule como un problema de programación lineal.

Función objetivo: $$Maximizar\ Z = 70x_1 + 50x_2$$

s.a.:

$$4x_1 + 3x_2 \leq 240$$
$$2x_1 + x_2 \leq 100$$
$$x_i \geq 0 \quad ; i = 1,2$$

Ejemplo 2: Problema de maximización con R

```
library(lpSolve)
f.obj <- c(70,50)                                          # vector de coeficiente de la función objetivo
f.con <- matrix(c(4,3,2,1),nrow=2,byrow=TRUE)              # matriz de coeficientes de restricciones
f.dir <- c("<=")                                           # vector de dirección de la restricción
f.rhs <- c(240,100)                                        # valores de la restricción
solution <- lp("max",f.obj,f.con,f.dir,f.rhs)
solution
solution$solution
```

```
Success: the objective function is 4100

solution$solution
[1] 30 40
```

Ejemplo 3: Problema de transporte con R utilizando lp.transport

Se tienen tres proveedores que pueden enviar mercancía a cuatro clientes diferentes. Cada proveedor tiene las siguientes ofertas 100, 300 y 400 unidades respectivamente . Los clientes tienen una demanda de 100, 100, 200 y 400 unidades respectivamente. El costo de abastecimiento del proveedor i al cliente j se muestra en la matriz de costo abajo. Encontrar la mejor asignación de unidades obteniendo el costo de transporte mínimo.

	Cliente 1	Cliente 2	Cliente 3	Cliente 4	OFERTA
Proveedor 1	$1	$2	$3	$4	100
Proveedor 2	$4	$3	$2	$1	300
Proveedor 3	$1	$4	$3	$2	400
DEMANDA	100	100	200	400	

Referencia

https://www.supplychaindataanalytics.com/solving-bronsons-transport-problem-with-lp-transport-in-r-using-lpsolve/

Ejemplo 3: Problema de transporte con R utilizando lp.transport

```
library(lpSolve)

cost.mat <- matrix(nrow=3,ncol=4)
cost.mat[1,] <- 1:4
cost.mat[2,] <- 4:1
cost.mat[3,] <- c(1,4,3,2)

direction = "min"

row.signs <- rep("<=",3)
row.rhs <- c(100,300,400)

col.signs <- rep(">=",4)
col.rhs <- c(100,100,200,400)

solution <- lp.transport(cost.mat = cost.mat,
                         direction = direction,
                         row.signs = row.signs,
                         row.rhs = row.rhs,
                         col.signs = col.signs,
                         col.rhs = col.rhs)
```

Ejemplo 3: Problema de transporte con R utilizando lp.transport

```
solution <- lp.transport(cost.mat = cost.mat,
                         direction = direction,
                         row.signs = row.signs,
                         row.rhs = row.rhs,
                         col.signs = col.signs,
                         col.rhs = col.rhs)
```

Solution

Success: the objective function is 1400

solution$solution

```
     [,1] [,2] [,3] [,4]
[1,]    0  100    0    0
[2,]    0    0  200  100
[3,]  100    0    0  300
```

El proveedor 1 debe enviar 100 unidades al cliente 2; el proveedor 2 debe enviar 200 unidades al cliente 3 y 100 unidades al cliente 4 mientras que el proveedor 3 debe enviar 100 unidades al cliente 1 y 300 unidades al cliente 4. El costo de transporte con esta asignación será de $1400

Ejemplo 4: Problema de asignación con R utilizando lp.assign

A continuación, se indica el problema para un caso en el que hay 3 trabajos y 3 máquinas. El costo de producir el trabajo 1 en la máquina 1 USD pero 2 USD cuando se produce en la máquina 2. El trabajo 2 cuesta 2 USD en la máquina 1 y 3 USD en la máquina 2. El trabajo 3 cuesta 5 USD en la máquina 1 y 1 USD en la máquina 2. La máquina 3 puede realizar el trabajo 1 por 2 USD y los trabajos 2 y 3 por 3 USD respectivamente. Resolver utilizando R.

$$\min 1x_{11} + 2x_{12} + 3x_{13} + 2x_{21} + 3x_{22} + 3x_{23} + 5x_{31} + 1x_{32} + 3x_{33}$$

$$s.t.$$

$$x_{11} + x_{12} + x_{13} = 1$$

$$x_{21} + x_{22} + x_{23} = 1$$

$$x_{31} + x_{32} + x_{33} = 1$$

$$x \in \mathbb{N}$$

Referencia

https://www.supplychaindataanalytics.com/es/programacion-de-produccion-de-costo-minimo-solucion-del-problema-de-asignacion-con-lpsolve-en-r/

Ejemplo 4: Problema de asignación con R utilizando lp.assign

```
library(lpSolve)

cost.mat <- rbind(c(1,2,3),
                  c(2,3,3),
                  c(5,1,3))

solution <- lp.assign(cost.mat=cost.mat,
                      direction="min")

Solution

## Success: the objective function is 5

solution$solution
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    0    1
[3,]    0    1    0
```

Se debe hacer el trabajo en la máquina 1
el trabajo 2 en la máquina 2 y el trabajo 3 en la
máquina 2 obteniendo un costo total mínimo de $5

Bibliografía

- Render, B. (2016). Métodos cuantitativos para los Negocios. Editorial Pearson.
- Taha, H. (2011). Investigación de Operaciones. Editorial Pearson.
- Hillier, F. & Lieberman, G. (2015). Investigación de Operaciones. McGraw-Hill
- Winston, W. (2004). Operations Research Applications and Algorithms. Thomson Brooks/Cole
- Anderson, D. & Sweeny, D. (2019). Métodos Cuantitativos para los Negocios. Cengage
- Eppen, D. (2000). Investigación de Operaciones en la Ciencia Administrativa. Pearson
- García et al. (2013). Simulación y Análisis de Sistemas con ProModel. Editorial Pearson.
- Srinivasan, G. (2010). Quantitative Models in Operations and Supply Chain Management. PHI Learning Private Limited
- Rardin, R. (2017). Optimization in Operations Research. Pearson
- Carter, M. et al. (2019). Operations Research A Practical Introduction. Taylor & Francis Group
- Aoroto Álvares, C., [et al] (2014) Operations research in business administration and managenent. Valencia: Universitat Politècnica de València
- Ravi Ravindran, A. (2008) Operations Research & Management Science Handbook. Taylor & Francis Group
- Rees, M. (2015). Business Risk and Simulation Modeling in Practice. John Wiley & Sons Ltd
- Sterman, J. (2000). Business Dynamics – Systems Thinking and Modeling for a Complex World. McGraw-Hill
- Winston, W. (2017) Microsoft Excel 2016 – Data Analysis and Busines Modeling. Microsoft press
- Schaffernicht, M. (2006). *Dinámica de Sistemas – Tomo 1: Fundamentos*.
- Alvarez, H. (2011). Introducción a la Simulación. Universidad Tecnológica de Panamá
- https://wps.prenhall.com/bp_taylor_introms_11/220/56508/14466195.cw/content/
- https://rstudio.com/products/rstudio/download/
- https://www.r-project.org/

Contacto

Ricardo Caballero, M.Sc.

Docente Tiempo Completo
Facultad de Ingeniería Industrial
Centro Regional de Chiriquí
Universidad Tecnológica de Panamá

E-mail: ricardo.caballero@utp.ac.pa

https://www.academia.utp.ac.pa/ricardo-caballero