Módulo 2 - Lenguaje de programación R-Cran (paradigma de objetos)

Módulo TyHM

Berard, Louise
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
louise.berard28@gmail.com

Ibañez, Celina
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
mcelinaibanez@gmail.com

Mobilia, Pilar
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
pilarmobilia@gmail.com

Torresi, Carla
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
torresi.carla16@gmail.com

Valli, Karima
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
karimavallillalen@gmail.com

Zanella, Bernardita
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
bernizanella2014@gmail.com

Año 2024

Abstract

Este módulo establece los primeros pasos de programación en R-Cran.

***K*eywords** Programing Language · R-Cran · RMarkdown

1 Introducción

En este módulo, exploraremos los siguientes temas:

1. **Introducción a los Algoritmos:**
   - Comprenderemos qué son los algoritmos y cómo se aplican en la resolución de problemas.
   - Analizaremos ejemplos de algoritmos comunes y su importancia en la programación.
2. **Repaso de Conceptos de Cálculo Numérico (Matlab / Octave):**
   - Refrescaremos nuestros conocimientos sobre cálculo numérico, centrándonos en herramientas como Matlab y Octave.
   - Exploraremos cómo aplicar estos conceptos en la programación.
3. **El Lenguaje de Programación R:**
   - Aprenderemos sobre R, un lenguaje ampliamente utilizado en análisis de datos y estadísticas.
   - Veremos cómo instalar R y configurar nuestro entorno de trabajo.
4. **Creación de Notebooks en IDEs (RMarkdown):**
   - Utilizaremos RMarkdown para crear documentos interactivos que combinen código, texto y visualizaciones.

- Exploraremos cómo generar informes y presentaciones utilizando esta herramienta.

5. **Cheat Sheets:**
   - Descubriremos hojas de referencia (cheat sheets) para R y otros lenguajes, que nos ayudarán a recordar funciones y sintaxis clave.
6. **Estructura de Datos en R:**
   - Estudiaremos las estructuras de datos fundamentales en R, como matrices, vectores y dataframes.
   - Practicaremos su manipulación y análisis.
7. **Importación de Datasets (Kaggle):**
   - Aprenderemos a importar conjuntos de datos desde fuentes externas, como Kaggle.
   - Realizaremos ejercicios prácticos con datos reales.
8. **Gestión de Proyectos en GitHub:**
   - Exploraremos cómo utilizar GitHub para gestionar proyectos de programación colaborativos.
   - Veremos cómo trabajar con repositorios, ramas y colaboradores.
9. **Ejercicios Prácticos:**
   - Abordaremos varios ejercicios, incluyendo:
   - La penitencia de Taylor (¿qué será esto?).
   - Algoritmo de ordenación.
   - Multiplicación vectorial.
   - Creación de gráficos.

2 Ecuaciones

You can use directly LaTeX command or Markdown text.

LaTeX command can be used to reference other section. See Section 2. However, you can also use **bookdown** extensions mechanism for this.

2.1 Ecuaciones en bloques

You can use equation in blocks

$$\xi_{ij}(t) = P(x_t = i, x_{t+1} = j|y, v, w; \theta) = \frac{\alpha_i(t) a_{ij}^{w_t} \beta_j(t+1) b_j^{v_{t+1}}(y_{t+1})}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij}^{w_t} \beta_j(t+1) b_j^{v_{t+1}}(y_{t+1})}$$

But also inline i.e $z = x + y$

3 Ejemplos de citaciones, figuras, tablas, referencias

You can insert references. Here is some text (Kour and Saabne 2014b, 2014a) and see Hadash et al. (2018).

The documentation for `natbib` may be found at

You can use custom blocks with LaTeX support from **rmarkdown** to create environment.

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf%7D`

Of note is the command `\citet`, which produces citations appropriate for use in inline text.

You can insert LaTeX environment directly too.

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated...

`https://www.ctan.org/pkg/booktabs`

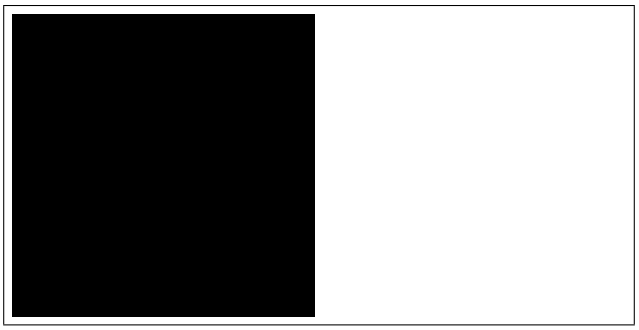

Figure 1: Sample figure caption.

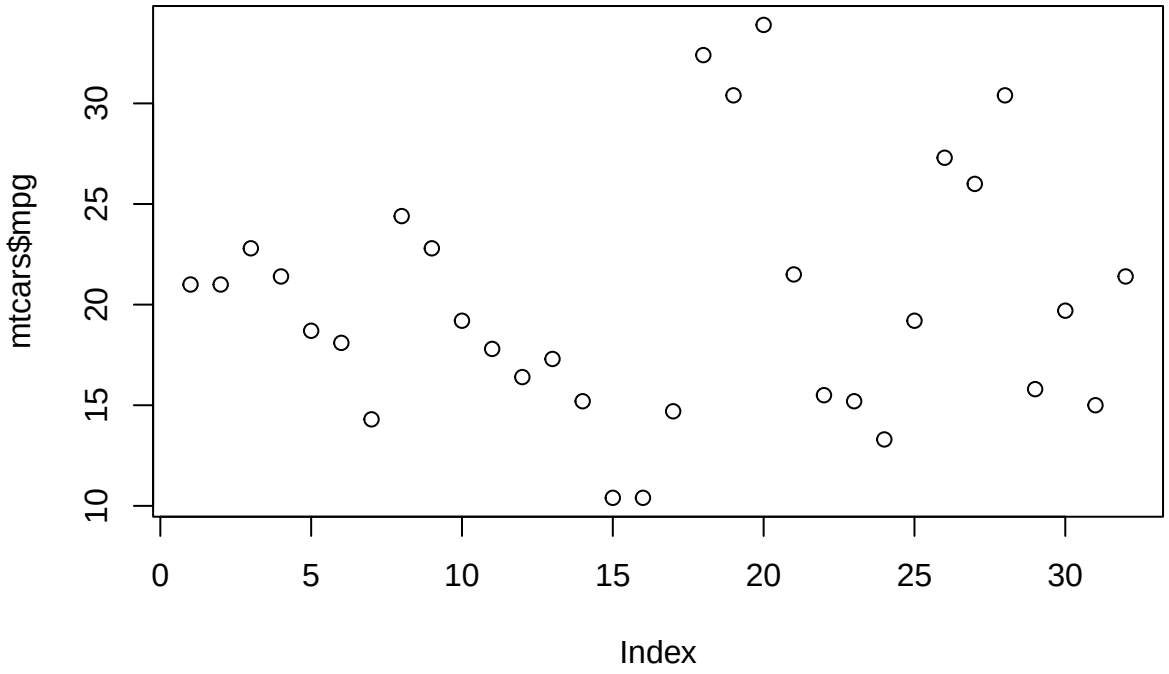

Figure 2: Another sample figure

3.1 Figures

You can insert figure using LaTeX directly.

See Figure 1. Here is how you add footnotes. [^Sample of the first footnote.]

But you can also do that using R.

```
plot(mtcars$mpg)
```

You can use **bookdown** to allow references for Tables and Figures.

3.2 Tables

Below we can see how to use tables.

See awesome Table~1 which is written directly in LaTeX in source Rmd file.

You can also use R code for that.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

```
knitr::kable(head(mtcars), caption = "Head of mtcars table")
```

Table 2: Head of mtcars table

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

3.3 Lists

- Item 1
- Item 2
- Item 3

  Código html w3

```
<html>
<head>
Titulo
</head>
<h1> Titulo </h1>
</head>
</head>
```

Este código es compatible con w3 Consortium Ver: (Consortium et al. 2000) .

Está conformado siguien las regles de paridad da tags. esto quiere decir que todo tag que se abre, luego se cierra.

4 Elementos básicos de programación

En este breve tutorial, exploraremos algunos elementos clave del lenguaje de programación R y cómo podemos utilizarlos para resolver problemas de la vida cotidiana. Nuestro enfoque estará en la eficiencia computacional y la comparación entre soluciones implementadas en R y soluciones conocidas. Veremos: - Generar un vector secuencia - Implementación de una serie Fibonacci - Ordenación de un vector por método burbuja - La penitencia de Taylor - Algoritmo de funciones estadísticas

4.1 Ideas de cómo medir el tiempo de ejecución

El benchmarking es una técnica común para comparar algoritmos y evaluar su eficiencia en términos de tiempo de ejecución, uso de memoria y otros recursos. Aunque no siempre hay un "mejor" algoritmo universal, el benchmarking nos ayuda a seleccionar la opción más adecuada según las características del problema y los recursos disponibles.

- Ejemplo con Matrices en Octave o Matlab: En Octave o Matlab, solíamos medir el tiempo de ejecución de algoritmos utilizando los comandos `tic` y `toc`. Por ejemplo:
  1. Generábamos una matriz **A**.
  2. Iniciábamos el cronómetro con `tic`.
  3. Calculábamos la inversa de **A** utilizando un método (por ejemplo, determinante).
  4. Deteníamos el cronómetro con `toc` y obteníamos el tiempo de ejecución.
  5. Repetíamos el proceso con otro método (por ejemplo, matriz LU).
- Opciones en R para Medir el Tiempo de Ejecución:

1. **`Sys.time()`**:
   - Proporciona una aproximación del tiempo de ejecución total de un bloque de código.
   - Útil para evaluaciones rápidas.
2. **`system.time()`**:
   - Proporciona información más detallada, incluyendo el tiempo de CPU utilizado por el código.
   - Permite medir el tiempo de ejecución de funciones específicas.
3. **Paquetes R para Benchmarking**:
   - **`rbenchmark`**, **`microbenchmark`** y **`tictoc`** son paquetes que miden el tiempo y también informan sobre el uso de memoria y CPU.

4.1.1 Usando Sys.time

El tiempo de ejecución de un fragmento de código se puede medir tomando la diferencia entre el tiempo al inicio y al final del fragmento de código leyendo los registros del RTC (Real Time Clock).

Cálculo del tiempo que se demora en armar la matriz el algoritmo:

```
mi_vector_d <- seq(1:100)
start_time <- Sys.time()
mi_matriz_e <- matrix(mi_vector_d, nrow=10, byrow=TRUE)
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 0.001264811 secs
```

Si se usa el comando dentro de un documento en R-Studio te demorarás mucho tiempo cuando compiles un PDF o una presentación.

4.1.2 Método tictoc

Esto de usar una biblioteca es llamar u cargar una procedimientos que generará comando nuevos en R. Como ya fue comentado, cargar una biblioteca implica ejecutar el comando install.packages() o usar en r-studio el menú de Herramientas y Luego Instalar paquetes. Las funciones tic y toc son de la misma biblioteca de Octave/Matlab y se usan de la misma manera para la evaluacion comparativa que el tiempo de sistema recién demostrado. Sin embargo, tictoc agrega mucha más comodidad al usuario y armonía al conjunto.

```
library(tictoc)
mi_vector_f <- seq(1:100)
tic("Tiempo que se demora en hacer la matriz g")
mi_matriz_g <- matrix(mi_vector_d, nrow=10, byrow=TRUE)
mi_vector_f
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

```
mi_matriz_g
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    2    3    4    5    6    7    8    9    10
##  [2,]   11   12   13   14   15   16   17   18   19    20
##  [3,]   21   22   23   24   25   26   27   28   29    30
##  [4,]   31   32   33   34   35   36   37   38   39    40
##  [5,]   41   42   43   44   45   46   47   48   49    50
##  [6,]   51   52   53   54   55   56   57   58   59    60
##  [7,]   61   62   63   64   65   66   67   68   69    70
##  [8,]   71   72   73   74   75   76   77   78   79    80
##  [9,]   81   82   83   84   85   86   87   88   89    90
## [10,]   91   92   93   94   95   96   97   98   99   100
```

```
toc()
```

```
## Tiempo que se demora en hacer la matriz g: 0.003 sec elapsed
```

4.2 Vectores

Un vector es una estructura de datos que almacena números de doble presición.

```
mi_vector_a <- c(12,34,12,54,23,12,65,34,12,56,66)
mi_vector_b <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)#seq(1:16)

mi_vector_a
```

```
##  [1] 12 34 12 54 23 12 65 34 12 56 66
```

```
mi_vector_b
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
```

4.2.1 Suma de vectores

```
sum(mi_vector_a,mi_vector_b)
```

```
## [1] 516
```

4.2.2 Generar vector secuencia

```
A <- 0
for (i in 1:50000) { A[i] <- (i*2)}
head (A)
```

```
## [1]  2  4  6  8 10 12
```

Secuencias generada con for - [1] 2 4 6 8 10 12

```
tail (A)
```

```
## [1]  99990  99992  99994  99996  99998 100000
```

- [1] 99990 99992 99994 99996 99998 100000

Secuencia generada con R - [1] 1 3 5 7 9 11 - [1] 999989 999991 999993 999995 999997 999999

4.3 Serie Fibonacci

En matemáticas, la sucesión o serie de Fibonacci es la siguiente sucesión infinita de números naturales: 0,1,1,2,3,5,8 ... 89,144,233 ... La sucesión comienza con los números 0 y 1,2 a partir de estos, cada término es la suma de los dos anteriores, es la relación de recurrencia que la define. A los elementos de esta sucesión se les llama números de Fibonacci. Tiene numerosas aplicaciones en ciencias de la computación, matemática y teoría de juegos. También aparece en configuraciones biológicas, como por ejemplo en las ramas de los árboles, en la disposición de las hojas en el tallo, en las flores de alcachofas y girasoles, en las inflorescencias del brécol romanesco, en la configuración de las piñas de las coníferas, en la reproducción de los conejos y en como el ADN codifica el crecimiento de formas orgánicas complejas. De igual manera, se encuentra en la estructura espiral del caparazón de algunos moluscos, como el nautilus.

```
start_time <- Sys.time()
a<-0
b<-1
c<-a+b

  while (c<=1000000) {
    a<-b
    b<-c
    c<-a+b
  }
c
```

```
## [1] 1346269
```

```
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 0.007968426 secs
```

4.4 Método Burbuja

La Ordenación de burbuja (Bubble Sort en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas burbujas. También es conocido como el método del intercambio directo. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo uno de los más sencillos de implementada.

```
x<-sample(1:100,10)
start_time <- Sys.time()
burbuja <- function(x){
n<-length(x)
for(j in 1:(n-1)){
for(i in 1:(n-j)){
if(x[i]>x[i+1]){
temp<-x[i]
x[i]<-x[i+1]
x[i+1]<-temp
```

```
}
}
}
return(x)
}
res<-burbuja(x)
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 0.01252508 secs
```

4.5 Penitencia de Gauss

A este método lo realizamos de 2 formas:

4.5.1 Sumas y multiplicación

```
start_time <- Sys.time()
suma <- 0
n<-10000
for (i in 1:n) {
    suma <- suma + i
}
suma
```

```
## [1] 50005000
```

```
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 0.006286144 secs
```

```
n<-500
mi_vector_b<- seq(1:n)
S1<-0
R<-0
S1<-mi_vector_b[1]+mi_vector_b[n]
R<-(n-1)/2*S1
R
```

```
## [1] 124999.5
```

4.5.2 For con bucle

```
m<-500
mi_vector_a<- seq(1:m)
R <- 0

for (i in 1:m) {
  R <- R + mi_vector_a[i]
}
R
```

```
## [1] 125250
```

4.6 Matrices

Las matrices se parecen a los vectores, pero tienen filas y columnas. Se alimentan de vectores.

```r
mi_matriz_c <- matrix(mi_vector_b, nrow=4, byrow=4)
mi_matriz_c
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    1    2    3    4    5    6    7    8    9    10    11    12    13    14
## [2,]  126  127  128  129  130  131  132  133  134   135   136   137   138   139
## [3,]  251  252  253  254  255  256  257  258  259   260   261   262   263   264
## [4,]  376  377  378  379  380  381  382  383  384   385   386   387   388   389
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]    15    16    17    18    19    20    21    22    23    24    25    26
## [2,]   140   141   142   143   144   145   146   147   148   149   150   151
## [3,]   265   266   267   268   269   270   271   272   273   274   275   276
## [4,]   390   391   392   393   394   395   396   397   398   399   400   401
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]    27    28    29    30    31    32    33    34    35    36    37    38
## [2,]   152   153   154   155   156   157   158   159   160   161   162   163
## [3,]   277   278   279   280   281   282   283   284   285   286   287   288
## [4,]   402   403   404   405   406   407   408   409   410   411   412   413
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]    39    40    41    42    43    44    45    46    47    48    49    50
## [2,]   164   165   166   167   168   169   170   171   172   173   174   175
## [3,]   289   290   291   292   293   294   295   296   297   298   299   300
## [4,]   414   415   416   417   418   419   420   421   422   423   424   425
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]    51    52    53    54    55    56    57    58    59    60    61    62
## [2,]   176   177   178   179   180   181   182   183   184   185   186   187
## [3,]   301   302   303   304   305   306   307   308   309   310   311   312
## [4,]   426   427   428   429   430   431   432   433   434   435   436   437
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]    63    64    65    66    67    68    69    70    71    72    73    74
## [2,]   188   189   190   191   192   193   194   195   196   197   198   199
## [3,]   313   314   315   316   317   318   319   320   321   322   323   324
## [4,]   438   439   440   441   442   443   444   445   446   447   448   449
##      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
## [1,]    75    76    77    78    79    80    81    82    83    84    85    86
## [2,]   200   201   202   203   204   205   206   207   208   209   210   211
## [3,]   325   326   327   328   329   330   331   332   333   334   335   336
## [4,]   450   451   452   453   454   455   456   457   458   459   460   461
##      [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
## [1,]    87    88    89    90    91    92    93    94    95    96    97    98
## [2,]   212   213   214   215   216   217   218   219   220   221   222   223
## [3,]   337   338   339   340   341   342   343   344   345   346   347   348
## [4,]   462   463   464   465   466   467   468   469   470   471   472   473
##      [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## [1,]    99    100    101    102    103    104    105    106    107    108
## [2,]   224    225    226    227    228    229    230    231    232    233
## [3,]   349    350    351    352    353    354    355    356    357    358
## [4,]   474    475    476    477    478    479    480    481    482    483
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## [1,]    109    110    111    112    113    114    115    116    117    118
## [2,]    234    235    236    237    238    239    240    241    242    243
## [3,]    359    360    361    362    363    364    365    366    367    368
## [4,]    484    485    486    487    488    489    490    491    492    493
##      [,119] [,120] [,121] [,122] [,123] [,124] [,125]
## [1,]    119    120    121    122    123    124    125
## [2,]    244    245    246    247    248    249    250
## [3,]    369    370    371    372    373    374    375
## [4,]    494    495    496    497    498    499    500
```

4.6.1 Llenar por fila o columna la matriz

byrow=TRUE, me llena por fila byrow=FALSE, me llena por columna

4.6.2 ¿Cómo accedo a un elemento de la matriz?

```
mi_matriz_c[2,4]
```

```
## [1] 129
```

4.6.3 ¿Cómo traer una fila completa?

```
mi_matriz_c[2, ]
```

```
##   [1] 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
##  [19] 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
##  [37] 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
##  [55] 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
##  [73] 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
##  [91] 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
## [109] 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250
```

4.6.4 ¿Cómo traer una columna completa?

```
mi_matriz_c[ ,3]
```

```
## [1]   3 128 253 378
```

4.6.5 ¿Cómo accedo a toda la matriz menos la fila/columna 2?

```
mi_matriz_c[-2, ]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    1    2    3    4    5    6    7    8    9    10    11    12    13    14
## [2,]  251  252  253  254  255  256  257  258  259   260   261   262   263   264
## [3,]  376  377  378  379  380  381  382  383  384   385   386   387   388   389
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]    15    16    17    18    19    20    21    22    23    24    25    26
## [2,]   265   266   267   268   269   270   271   272   273   274   275   276
## [3,]   390   391   392   393   394   395   396   397   398   399   400   401
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]    27    28    29    30    31    32    33    34    35    36    37    38
## [2,]   277   278   279   280   281   282   283   284   285   286   287   288
## [3,]   402   403   404   405   406   407   408   409   410   411   412   413
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]    39    40    41    42    43    44    45    46    47    48    49    50
## [2,]   289   290   291   292   293   294   295   296   297   298   299   300
## [3,]   414   415   416   417   418   419   420   421   422   423   424   425
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]    51    52    53    54    55    56    57    58    59    60    61    62
## [2,]   301   302   303   304   305   306   307   308   309   310   311   312
## [3,]   426   427   428   429   430   431   432   433   434   435   436   437
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]    63    64    65    66    67    68    69    70    71    72    73    74
## [2,]   313   314   315   316   317   318   319   320   321   322   323   324
## [3,]   438   439   440   441   442   443   444   445   446   447   448   449
```

```
##      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86]
## [1,]    75    76    77    78    79    80    81    82    83    84    85    86
## [2,]   325   326   327   328   329   330   331   332   333   334   335   336
## [3,]   450   451   452   453   454   455   456   457   458   459   460   461
##      [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
## [1,]    87    88    89    90    91    92    93    94    95    96    97    98
## [2,]   337   338   339   340   341   342   343   344   345   346   347   348
## [3,]   462   463   464   465   466   467   468   469   470   471   472   473
##      [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107] [,108]
## [1,]    99    100    101    102    103    104    105    106    107    108
## [2,]   349    350    351    352    353    354    355    356    357    358
## [3,]   474    475    476    477    478    479    480    481    482    483
##      [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117] [,118]
## [1,]    109    110    111    112    113    114    115    116    117    118
## [2,]    359    360    361    362    363    364    365    366    367    368
## [3,]    484    485    486    487    488    489    490    491    492    493
##      [,119] [,120] [,121] [,122] [,123] [,124] [,125]
## [1,]    119    120    121    122    123    124    125
## [2,]    369    370    371    372    373    374    375
## [3,]    494    495    496    497    498    499    500
```

5 Referencias Bibliográficas

https://themys.sid.uncu.edu.ar/rpalma/TyHM/Benchmark/Elementos-de-Programaci%c3%b3n.pdf

.

Consortium, W3 et al. 2000. "Extensible Markup Language (Xml) 1.0." *Http://Www.w3.org/TR/1998/REC-Xml-20001006/*.

Hadash, Guy, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. 2018. "Estimate and Replace: A Novel Approach to Integrating Deep Neural Networks with Existing Applications." *arXiv Preprint arXiv:1804.09028*.

Kour, George, and Raid Saabne. 2014a. "Fast Classification of Handwritten on-Line Arabic Characters." In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, 312–18. IEEE.

———. 2014b. "Real-Time Segmentation of on-Line Handwritten Arabic Script." In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, 417–22. IEEE.