Introducción a R para Ciencias C1-S2-clase3

Contents

1 Práctica de Casa 01 - Programación Básica con R

Realiza los siguientes ejercicios durante tu tiempo libre para mejorar tus habilidades de manejo del lenguaje de programación R.

Recuerda realizar esta práctica luego de haber aprendido la información de la clase 1 y 2 de la semana 1, y la clase 1 de la semana 2, es decir, luego de haber desarrollado:

- R-Notebook-C1-S1-clase1.Rmd
- R-Notebook-C1-S1-clase2.Rmd
- Funciones útiles.R (Script)

  Nota: Si necesitas crear un code chunk los atajos en el teclado son en WINDOWS: `Crtl+Alt+i`, y en MAC: `Command+Alt+i`.

1.1 Ejercicio 1:

Crea una variables llamada `tlf` que contengan la suma de los dígitos de tu número de celular, dividido entre el total de números que tiene. Visualiza el resultado ejecutando el nombre de la variable. Redondea esta cifra a dos decimales usando una de las funciones que aprendiste en el script "Funciones útiles".

```
tlf <- sum(9+3+2+0+9+3+2+8+5)/9
tlf
```

```
## [1] 4.555556
```

```
round(tlf)
```

```
## [1] 5
```

1.2 Ejercicio 2:

Crea los vectores siguientes:

```
# 2.1.
# Números del (1,2,3,4,...,20,21) en ese orden
1:21
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
```

```r
# 2.2.
# Números del (21,20,19,18,...,2,1) en ese orden
# Tip: usa el operador para rangos
21:1
```

```
##  [1] 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```r
# 2.3.
# Prueba creando un vector con vectores dentro
# Crea un vector que contenga lo números
# (1,2,3,4,...,20,21,21,20,19,18,...,2,1)
c(1:21,21:1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 21 20 19 18
## [26] 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```r
# 2.4.
# Asigna a temperatura el valor 24.3
temperatura <- 24.3

# 2.5.
# Asigna a precipitacion los valores 202.7, 10.5, 209.2
precipitacion <- c(202.7, 10.5, 209.2)

# 2.6.
# Además de bases de datos precargadas por defecto, en R exiten
# algunos vectores creados por defecto en la memoria del programa.
# El vector letters y el vector LETTERS contienen las letras
# del alfabeto. Podemos acceder a ellas simplemente ejecutando
# su nombre
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```r
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```r
# Ahora, indexa las letras del 10 al 20 en mayúsculas,
# y las letras a,b,c,d en las letras minúsculas
# Recuerda que estos objetos son vectores.
LETTERS[10:20]
```

```
##  [1] "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T"
```

```r
letters[1:4]
```

```
## [1] "a" "b" "c" "d"
```

```r
# 2.7.
# Usa la función paste() para crear un vector que contenga
# los valores ("Parcela 1", "Parcela 2", ....., "Parcela 30").
# otro con los valores ("Especie1", "Especie2", ....., "Especie10")
# y otro con los valores ("Hospital-A", "Hospital-B", "Hospital-C", "Hospital-D")
paste("Parcela", 1:30)
```

```
##  [1] "Parcela 1"  "Parcela 2"  "Parcela 3"  "Parcela 4"  "Parcela 5"
##  [6] "Parcela 6"  "Parcela 7"  "Parcela 8"  "Parcela 9"  "Parcela 10"
```

```
## [11] "Parcela 11" "Parcela 12" "Parcela 13" "Parcela 14" "Parcela 15"
## [16] "Parcela 16" "Parcela 17" "Parcela 18" "Parcela 19" "Parcela 20"
## [21] "Parcela 21" "Parcela 22" "Parcela 23" "Parcela 24" "Parcela 25"
## [26] "Parcela 26" "Parcela 27" "Parcela 28" "Parcela 29" "Parcela 30"
```

```
paste0("Especie",1:10)
```

```
##  [1] "Especie1"  "Especie2"  "Especie3"  "Especie4"  "Especie5"  "Especie6"
##  [7] "Especie7"  "Especie8"  "Especie9"  "Especie10"
```

```
paste("Hospital", c("A","B","C"), sep="-")
```

```
## [1] "Hospital-A" "Hospital-B" "Hospital-C"
```

```
paste("Hospital", LETTERS[1:4], sep="-")
```

```
## [1] "Hospital-A" "Hospital-B" "Hospital-C" "Hospital-D"
```

1.3 Ejercicio 3:

Practica la indexación de vectores, factores y data.frames (lo mismo aplica para matrices y tibbles):

```
# Carga la base de datos airquality
data("airquality")

# 3.1.
# Indexa la posición columna 4, fila 100
airquality[100,4]
```

```
## [1] 90
```

```
# 3.2.
# Indexa la columna Ozone y las filas del 100 al 120
airquality[100:120,"Ozone"]
```

```
##  [1]  89 110  NA  NA  44  28  65  NA  22  59  23  31  44  21   9  NA  45 168  73
## [20]  NA  76
```

```
# 3.3.
# Indexa las columnas Ozone y Temp en el mismo
# rango de filas que el ejercicio 3.2.
# Asigna el resultado al nombre aire
aire <- airquality[100:120,c("Ozone","Temp")]

# 3.4.
# Halla la suma de los valores de la columna Temp
# de la variable aire que creaste en 3.3.
sum(aire$Temp)
```

```
## [1] 1728
```

```
# 3.5.
# Ahora trata de hacer lo mismo para ambas columnas
# de la variable aire, pero usando la función colSums()
# aprendida en el script "Funciones Útiles.R"
colSums(aire)
```

```
## Ozone  Temp
##    NA  1728
```

Si observas la suma de valores de `Ozone`, que debería de ser un número, aparece como `NA` (valor perdido). Cuando una columna tiene valores perdidos cualquier operación matemática o estadística se convierte en

NA. Para evitarlo, usa dentro de colSums() el argumento na.rm = TRUE. Este argumento cuando es TRUE remueve los NA (de ello derivan las siglas na.rm).

```
# 3.6.
# Usa na.rm = TRUE dentro de colSums() y obtén
#la suma de ambas columnas  en la variable aire.
colSums(aire, na.rm = T)
```

```
## Ozone  Temp
##   907  1728
```

```
# Calcula el promedio de ambas columnas en aire,
# usando la función apply() aprendida en el script "Funciones Útiles.R"
# usa na.rm=TRUE como cuarto argumento de la función apply.
apply(aire, 2, mean, na.rm=TRUE)
```

```
##    Ozone     Temp
## 56.68750 82.28571
```

```
# 3.7.
# Extrae la columna Species de la base de datos y guárdala
# con el nombre "esp".
data(iris)
esp <- iris$Species

# Visualiza esp
esp
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
# De esp, indexa la posicion 78 ¿A qué especie corresponde?
esp[78]
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

```
# Dado que en la parte inferior del resulta en la consola aparece
# Levels: setosa versicolor virginica
# Debo pensar que este vector de datos en realidad es un factor.
# Compruébalo usando la función is()
is(esp)
```

```
## [1] "factor"              "integer"             "oldClass"
## [4] "double"              "numeric"             "vector"
## [7] "data.frameRowLabels"
```

```
# Ahora coerciona esp de factor a vector tipo caracter
# Y guárdalo asignándole el nombre char.
char <- as.character(esp)

# Indexa la misma posición, 78, en char y verifica
# si el resultado te muestra niveles.
# Si la respuestas es no, es porque ya no es factor,
# ahora es un vector tipo carácter.
char[78]
```

```
## [1] "versicolor"
```

1.3.1 "Coder Tip" para trabajar con factores dentro de tablas

Es muy importante saber reemplazar el nombre de los niveles de un factor con otros valores, ya sea como variables independientes o como una columna dentro de una tabla. Para este ejercicio necesitaremos cargar la base de datos `airquality`. Trabajaremos con la columna `Month`, meses en español. Veamos la tabla:

```
data("airquality")
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

Veamos los valores únicos de la columna `Month` con la función `unique()`:

```
unique(airquality$Month)
```

```
## [1] 5 6 7 8 9
```

La columna `Month` se puede considerar un vector numérico, con valores del 5 al 9 repetidos varias veces. Cada cifra representa los meses (según su número de orden en el año) Mayo, Junio, Julio, Agosto, Septiembre. Podemos usar la función `factor()` para convertir la columna en factor y, a la vez, aprovechar reemplazar los números por sus respectivos nombres de mes, utilizando el argumento `labels`. Cada nivel debe ser reemplazado con un valor como lo menciona el código a continuación:

```
factor(airquality$Month, labels = c("5"="Mayo", "6"="Junio", "7"="Julio",
                                    "8"="Agosto","9"="Septiembre"))
```

```
##   [1] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##   [7] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
```

```
##  [13] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [19] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [25] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [31] Mayo       Junio      Junio      Junio      Junio      Junio
##  [37] Junio      Junio      Junio      Junio      Junio      Junio
##  [43] Junio      Junio      Junio      Junio      Junio      Junio
##  [49] Junio      Junio      Junio      Junio      Junio      Junio
##  [55] Junio      Junio      Junio      Junio      Junio      Junio
##  [61] Junio      Julio      Julio      Julio      Julio      Julio
##  [67] Julio      Julio      Julio      Julio      Julio      Julio
##  [73] Julio      Julio      Julio      Julio      Julio      Julio
##  [79] Julio      Julio      Julio      Julio      Julio      Julio
##  [85] Julio      Julio      Julio      Julio      Julio      Julio
##  [91] Julio      Julio      Agosto     Agosto     Agosto     Agosto
##  [97] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [103] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [109] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [115] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [121] Agosto     Agosto     Agosto     Septiembre Septiembre Septiembre
## [127] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [133] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [139] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [145] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [151] Septiembre Septiembre Septiembre
## Levels: Mayo Junio Julio Agosto Septiembre
```

Como viste, a pesar de que los niveles fueron números, estos deben ser colocados con comillas `""`, y con símbolo de igual `=` designar con qué valor va a ser reemplazado. Sin embargo, cuando revisamos cual es la clase de la columna `Month` en `airquality` vemos que sigue siendo numérica.

```
airquality$Month
```

```
##   [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6
##  [38] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7
##  [75] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
## [112] 8 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
## [149] 9 9 9 9 9
```

Lo que sucede es que con el código de la función `factor()` solo creaste un factor, pero no lo usaste para reemplazar a la columna `Month`. Para ello debes asignarle el factor que crees a la columna que deseas reemplazar.

```
airquality$Month <- factor(airquality$Month,
                           labels = c("5"="Mayo", "6"="Junio", "7"="Julio",
                                      "8"="Agosto","9"="Septiembre"))
```

Ahora revisemos el contenido de dicha columna

```
airquality$Month
```

```
##   [1] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##   [7] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [13] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [19] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [25] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [31] Mayo       Junio      Junio      Junio      Junio      Junio
##  [37] Junio      Junio      Junio      Junio      Junio      Junio
##  [43] Junio      Junio      Junio      Junio      Junio      Junio
```

```
##  [49] Junio      Junio      Junio      Junio      Junio      Junio
##  [55] Junio      Junio      Junio      Junio      Junio      Junio
##  [61] Junio      Julio      Julio      Julio      Julio      Julio
##  [67] Julio      Julio      Julio      Julio      Julio      Julio
##  [73] Julio      Julio      Julio      Julio      Julio      Julio
##  [79] Julio      Julio      Julio      Julio      Julio      Julio
##  [85] Julio      Julio      Julio      Julio      Julio      Julio
##  [91] Julio      Julio      Agosto     Agosto     Agosto     Agosto
##  [97] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [103] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [109] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [115] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [121] Agosto     Agosto     Agosto     Septiembre Septiembre Septiembre
## [127] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [133] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [139] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [145] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [151] Septiembre Septiembre Septiembre
## Levels: Mayo Junio Julio Agosto Septiembre
```

O veamos la tabla

```
View(airquality)
```

Algo que también es importante es reordenar los niveles de un factor. Cambia el orden del factor `Month`en la base de datos `airquality` para que comience por Septiembre y termine en Mayo, usando el argumento `levels`. Este orden, a pesar de que parece no afectar en nada a la tabla, tiene un gran impacto en cómo se verán gráficas o análisis basados en esta columna.

```
airquality$Month <- factor(airquality$Month,
                           levels = c("Septiembre","Agosto","Julio","Junio","Mayo"))
```

Revisemos nuevamente el contenido de `Month`

```
airquality$Month
```

```
##   [1] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##   [7] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [13] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [19] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [25] Mayo       Mayo       Mayo       Mayo       Mayo       Mayo
##  [31] Mayo       Junio      Junio      Junio      Junio      Junio
##  [37] Junio      Junio      Junio      Junio      Junio      Junio
##  [43] Junio      Junio      Junio      Junio      Junio      Junio
##  [49] Junio      Junio      Junio      Junio      Junio      Junio
##  [55] Junio      Junio      Junio      Junio      Junio      Junio
##  [61] Junio      Julio      Julio      Julio      Julio      Julio
##  [67] Julio      Julio      Julio      Julio      Julio      Julio
##  [73] Julio      Julio      Julio      Julio      Julio      Julio
##  [79] Julio      Julio      Julio      Julio      Julio      Julio
##  [85] Julio      Julio      Julio      Julio      Julio      Julio
##  [91] Julio      Julio      Agosto     Agosto     Agosto     Agosto
##  [97] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [103] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [109] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [115] Agosto     Agosto     Agosto     Agosto     Agosto     Agosto
## [121] Agosto     Agosto     Agosto     Septiembre Septiembre Septiembre
```

```
## [127] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [133] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [139] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [145] Septiembre Septiembre Septiembre Septiembre Septiembre Septiembre
## [151] Septiembre Septiembre Septiembre
## Levels: Septiembre Agosto Julio Junio Mayo
```

1.4 Ejercicio 4:

Crea un **code chunk**. Luego, carga la base de datos `trees` y responde:

- ¿Cuál es el número de columnas de la data frame `trees`?
- ¿Cuál es el número de filas de la data frame `trees`?
- Crea un subconjunto la tabla `trees` usando las filas de 10 a las 20 y las columnas 1 y 3. Guarda la tabla resultante con el nombre `trees.sub`. Visualiza la data frame `tree.sub`.

[*Crea el code chunk aquí*]

```
# 4.1.
# Data trees
data(trees)

# 4.2.
# N columnas
ncol(trees)
```

```
## [1] 3
```

```
# 4.3.
# N filas
nrow(trees)
```

```
## [1] 31
```

```
# 4.4.
# Subconjunto
trees[1:10,c(1,3)]
```

```
##    Girth Volume
## 1    8.3   10.3
## 2    8.6   10.3
## 3    8.8   10.2
## 4   10.5   16.4
## 5   10.7   18.8
## 6   10.8   19.7
## 7   11.0   15.6
## 8   11.0   18.2
## 9   11.1   22.6
## 10  11.2   19.9
```

1.4.1 "Coder Tip" para eliminar columnas

Puedes eliminar una columna de una tabla si le asignas a esa columna el valor `NULL` usando el operador de asignación: `<- NULL`. Por ejemplo, al cargar la base de datos iris con `data(iris)`, podemos eliminar la

columna `Species` de la siguiente manera:

```
# Carga la base de datos
data("iris")

# Visualicemos la tabla con head()
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# Utiliza $ para seleccionar la columna Species
# y eliminala
iris$Species <- NULL

# Visualicemos nuevamente para verificar que
# Species fue eliminada
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2
## 4          4.6         3.1          1.5         0.2
## 5          5.0         3.6          1.4         0.2
## 6          5.4         3.9          1.7         0.4
```

Ahora tú, elimina la columna `Height` de la base de datos `trees`.

```
# Escribe aquí tu código
trees$Height <- NULL

# Visualiza la tabla
head(trees)
```

```
##   Girth Volume
## 1   8.3   10.3
## 2   8.6   10.3
## 3   8.8   10.2
## 4  10.5   16.4
## 5  10.7   18.8
## 6  10.8   19.7
```

1.5 Ejercicio 5:

Por último, trabaja identificando valores perdidos para lidiar con ellos más adelante.

```
# Ejecuta este código para crear una data frame
# conteniendo valores perdidos (NAs)
df <- data.frame(Col1 = c(1,2,3,NA,5,6,NA,8,9,10),
           Col2 = 20:29,
           Col3 = c(LETTERS[1:5],NA,"J","K","I","NA"))
```

```r
# Visualiza df
View(df)
```

Los valores `NA` emulan una celda en blanco en una tabla de excel. Cuando uno coloca manualmente `NA` como texto en las celdas en blanco de un Excel, R las reconocerá como texto `"NA"`, tal cual se puso intencionalmente en la columna `Col3`, un `NA` y un `"NA"`. Al ubicar los valores perdidos, las siguientes fórmulas obviarán los textos `"NA"` que hayas escrito en tu Excel. Es recomendable NO colocar `"NA"` y simplemente dejar las celdas vacías si vamos a cargar nuestra tabla a RStudio.

Obviando que dicho `NA` no es un valor perdido real, realiza las siguientes indicaciones:

```r
# Cuántos NA hay en toda la tabla
sum(is.na(df))
```

```
## [1] 3
```

```r
# Identifica dónde (filas y columnas) aparecen las NA
which(is.na(df), arr.ind = TRUE)
```

```
##      row col
## [1,]   4   1
## [2,]   7   1
## [3,]   6   3
```

```r
# Identifica cuántos NA tiene cada columna
colSums(is.na(df))
```

```
## Col1 Col2 Col3
##    2    0    1
```

```r
# Ver cuál filas son casos completos (sin NA)
which(complete.cases(df))
```

```
## [1]  1  2  3  5  8  9 10
```

```r
# Ver cuál filas NO son casos completos (tienen al menos un NA)
which(!complete.cases((df)))
```

```
## [1] 4 6 7
```

```r
# Elimina todas las filas con al menos un NA
# y asigna la nueva tabla a df2
df2 <- na.omit(df)

# Visualiza df2
View(df2)
```

1.6