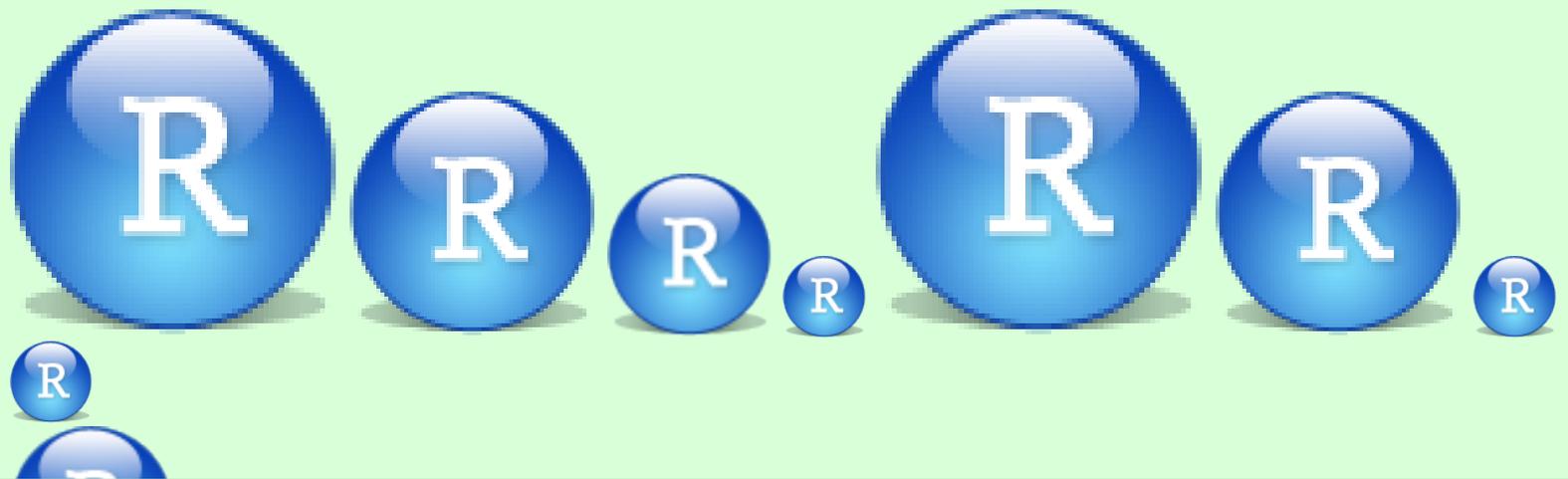


# Modelación Aplicada a las Ciencias Animales: Generalidades de R-project



**Mario Fernando Cerón-Muñoz**  
**Luis Fernando Galeano Vasco**  
**Jeanneth Mosquera Rendón**

**Ciencias Animales**



Cerón-Muñoz, MF; Galeano Vasco, LF; Mosquera Rendón, J

Modelación aplicada a las ciencias animales: generalidades de R-project / Mario Fernando Cerón Muñoz, Luis Fernando Galeano, Jeanneth Mosquera Rendón. - Medellín, Colombia: Editorial Biogénesis, [2013]

175 p.: il; 27 cm

Incluye referencias bibliográficas e índice

ISBN: 978-958-8790-70-1

1. Modelación animal. 2. Distribuciones

005

©Cerón-Muñoz MF; Galeano Vasco LF y Mosquera Rendón J

Primera Edición: Junio de 2013

ISBN: 978-958-8790-70-1

Autores:

Mario Fernando Cerón-Muñoz, Zootecnista, Dr en Zootecnia

Luis Fernando Galeano Vasco, Zootecnista, cDr en Ciencias Animales

Jeanneth Mosquera Rendón, Ingeniera Biológica.

Facultad de Ciencias Agrarias

Universidad de Antioquia

Corrección de textos:

Diego García Sierra

Evaluación de contenido:

Diego Fernando Lemus Polanía

Ingeniero Industrial, MS en Estadística

Luis Gabriel González Herrera

Médico Veterinario Zootecnista, Dr en Genética y Mejoramiento Animal

Elkin Mauricio Arboleda Zapata

Zootecnista, MS en Ciencias Animales

Diseño y diagramación en *TEX*:

Mario Fernando Cerón-Muñoz

Asesoría en Diseño:

Sandra María Arango Mejía

Todos los derechos reservados, puede ser reproducido en todo o en parte y por cualquier medio, citando la fuente. Esta publicación contó con el apoyo de la Universidad de Antioquia y del CODI sostenibilidad 2011-2012 del grupo GAMMA.



©Fondo Editorial Biogénesis

Universidad de Antioquia

Facultad de Ciencias Agrarias

Ciudadela de Robledo, Carrera 75 No 65-87

Medellín, Colombia

## Prólogo

Retomando las palabras de Menigno Hidalgo Matos, “El objetivo final cambiará: ya no consistirá en obtener un diploma, sino en disfrutar del aprendizaje a lo largo de toda una vida”, nuestro objetivo es apoyar el desempeño profesional de los futuros egresados de las ciencias animales, mediante la enseñanza de un programa computacional para el manejo de hojas de datos, operaciones matemáticas, análisis estadísticos y gráficos.

El entorno R se ha convertido en una herramienta estadística agradable, amigable, potente, coherente, confiable, con un amplio soporte científico, versátil y renovable, que le ha permitido difundirse y tener una amplia red de usuarios de diferentes campos profesionales.

Nuestro libro contiene aspectos relacionados con operaciones matriciales, generación de números pseudoaleatorios, creación de algoritmos, manejo de hojas de datos y generación de gráficos. Se utilizan variables y casos de sistemas de producción animal, buscando que nuestros lectores se familiaricen con R-project y vean su aplicabilidad en el campo pecuario.

No incluimos en este libro información sobre la instalación y el manejo básico debido a que el R-project es constantemente actualizado y adaptado a otros programas. Además, en internet existen páginas web de introducción al R-project, principalmente la página oficial: <http://www.r-project.org/>.

Hemos procurado no cometer errores en la escritura del texto; sin embargo, solicitamos a los lectores que nos informen la existencia de errores o sobre comandos y funciones más adecuados, con el objeto de corregirlos en ediciones futuras.

Este libro fue realizado en el sistema de composición de textos *L<sup>A</sup>T<sub>E</sub>X*, una gran herramienta para la diagramación de libros.

*Mario Fernando Cerón-Muñoz*  
Zootecnista, Dr.

# Contenido

<b>1. Matrices</b>	<b>1</b>
1.1. Arreglos matriciales . . . . .	1
1.2. Sumatorias al interior de una matriz . . . . .	6
1.2.1. Suma por filas . . . . .	6
1.2.2. Suma por columnas . . . . .	7
1.2.3. Suma de valores de algunas filas y columnas . . . . .	8
1.2.4. Operaciones utilizando la función <i>apply</i> . . . . .	9
1.3. Traza de una matriz . . . . .	11
1.4. Suma y resta de matrices . . . . .	12
1.5. Multiplicación de matrices . . . . .	13
1.6. Multiplicación de una matriz por un escalar . . . . .	15
1.7. Multiplicación directa de matrices . . . . .	16
1.8. Inversa de matrices . . . . .	18
1.9. Ecuación característica, autovalores y autovectores de una matriz . . . . .	20
1.10. Inversa generalizada de una matriz . . . . .	25
1.11. Algunas aplicaciones de matrices en la producción animal . . . . .	26
1.11.1. Pago por calidad de leche . . . . .	26
1.11.2. Formulación de raciones . . . . .	31
1.11.3. Formulación de raciones con restricción de nutrientes . . . . .	33
1.11.4. Precio del huevo . . . . .	35
<b>2. Números generados aleatoriamente</b>	<b>38</b>
2.1. Números pseudoaleatorios con distribución uniforme discreta . . . . .	39
2.2. Números pseudoaleatorios con distribución binomial . . . . .	40
2.3. Números pseudoaleatorios con distribución de Poisson . . . . .	46
2.4. Números pseudoaleatorios con distribución hipergeométrica . . . . .	52
2.5. Números pseudoaleatorios con distribución geométrica . . . . .	55
2.6. Números pseudoaleatorios con distribución uniforme continua . . . . .	57
2.7. Números pseudoaleatorios con distribución normal . . . . .	58
2.8. Números pseudoaleatorios con distribución continua lognormal . . . . .	64
2.9. Números pseudoaleatorios con distribución continua logística . . . . .	66
2.10. Números pseudoaleatorios con distribución multinomial . . . . .	69
2.11. Números pseudoaleatorios con distribución normal bivalente . . . . .	71

<b>3. Algoritmos y creación de funciones en R-project</b>	<b>76</b>
3.1. Algoritmos . . . . .	76
3.2. Comandos de R-project para algoritmos . . . . .	76
3.2.1. Comando <i>for</i> . . . . .	77
3.2.2. Comando <i>while</i> . . . . .	78
3.2.3. Comandos <i>if</i> , <i>repeat</i> y <i>break</i> . . . . .	80
3.3. Creación de funciones en R-project . . . . .	81
3.3.1. Ejemplo del peso de lechones . . . . .	82
3.3.2. Ejemplo para la obtención de un valor mínimo . . . . .	82
3.3.3. Ejemplo de una función que relaciona dos variables . . . . .	83
3.3.4. Ejemplo de una función con los comandos <i>for</i> y <i>seq</i> . . . . .	84
3.3.5. Ejemplo de una función con el comando <i>while</i> . . . . .	85
3.3.6. Ejemplo de una función con los comandos <i>repeat</i> , <i>if</i> y <i>break</i> . . . . .	86
3.3.7. Creación de una función <i>Lag</i> . . . . .	87
<b>4. Creación y montaje de hojas de datos</b>	<b>90</b>
4.1. Generalidades . . . . .	90
4.2. Hoja de datos a partir de matrices y vectores . . . . .	90
4.3. Estructura de una hoja de datos . . . . .	92
4.4. Importación de hojas de datos . . . . .	93
4.4.1. Archivos <i>txt</i> . . . . .	94
4.4.2. Archivos <i>csv</i> . . . . .	97
4.4.3. Archivos de <i>Excel</i> . . . . .	98
4.5. Manipulación de bases de datos . . . . .	99
4.5.1. Unión de hojas de datos . . . . .	99
4.5.2. Creación de nuevas variables . . . . .	102
4.5.3. Condicionales para operar variables . . . . .	104
4.6. Ejemplos de construcción de hojas de datos en producción animal . . . . .	107
4.6.1. Montaje de genealogías . . . . .	107
4.6.2. Cálculo de la producción de leche por lactancia . . . . .	118
4.6.3. Cálculo de la producción de leche hasta los 305 días . . . . .	129
<b>5. Montaje de gráficos</b>	<b>136</b>
5.1. Generalidades . . . . .	136
5.2. Comando <i>plot</i> para generar gráficos . . . . .	138
5.3. Comando <i>par</i> para especificaciones previas de los gráficos . . . . .	139
5.3.1. Argumentos para localización y posición . . . . .	140
5.3.2. Localización de los nombres de los ejes . . . . .	143
5.3.3. Líneas de los ejes . . . . .	144
5.3.4. Tamaño de letra y valores de los ejes . . . . .	145
5.3.5. Colores . . . . .	148
5.3.6. Tipos de letra . . . . .	150
5.3.7. Forma rectangular y cuadrada del área de trazado . . . . .	152
5.3.8. Grosor de los ejes y puntos de dato . . . . .	152
5.3.9. Varios gráficos en la misma área total . . . . .	153

5.4. Inclusión de líneas . . . . .	153
5.5. Símbolos de puntos de dato . . . . .	156
5.6. Divisiones de los ejes . . . . .	158
5.7. Nombres en el gráfico . . . . .	159
5.8. Gráficos de relación de variables <i>pairs</i> . . . . .	161
5.9. Gráficos <i>stem</i> . . . . .	161
5.10. Histogramas de frecuencia generados con el comando <i>histogram</i> . . . . .	162
5.11. Gráficos <i>qqplot</i> . . . . .	163
5.12. Gráficos <i>boxplot</i> , <i>barplot</i> y <i>pie</i> . . . . .	164
5.13. Gráficos <i>scatterplot</i> y <i>coplot</i> . . . . .	167
5.14. Gráficos de Cleveland . . . . .	171

## Capítulo 1

# Matrices

### 1.1. Arreglos matriciales

Un arreglo de valores dispuestos en  $m$  filas y  $n$  columnas se conoce como matriz. Las matrices se representan con letras mayúsculas y su tamaño u orden está dado por el número de filas ( $m$ ) y el número de columnas ( $n$ ).

Cada elemento incluido en la matriz tiene una posición de fila ( $i$ ) y de columna ( $j$ ), como se indica en este ejemplo:

$$A_{mn} = \begin{bmatrix} a_{i=1,j=1} & a_{i=1,j=2} & a_{i=1,j=3} \\ a_{i=2,j=1} & a_{i=2,j=1} & a_{i=2,j=1} \end{bmatrix}$$

$$A_{mn} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

En el caso anterior, la matriz  $A$  es de tamaño ( $m = 2$  por  $n = 3$ ). Su primer elemento está en la posición de la primera fila ( $i = 1$ ) y la primera columna ( $j = 1$ ). Veamos un arreglo matricial:

$$A_{mn} = \begin{bmatrix} 4 & -4 & 2 \\ 5 & 0.4 & 0 \end{bmatrix}$$

La matriz  $A$  es de tamaño  $2 \times 3$  con 6 elementos ( $A_{2,3}$ ), donde el elemento  $i = 1, j = 2$  es  $-4$ , y su expresión es  $a_{1,2} = -4$ .

Para la creación de la matriz  $A$  en R-project (R Core Team, 2012b), especificamos: *matrix* para dar la orden de creación de una matriz, *nrow* para indicar el número de filas, *ncol* para indicar el número de columnas y *data = c()* para ingresar los valores correspondientes a los elementos de la matriz, así:



```
A=matrix(nrow=2,ncol=3, data=c(4,5,-4,0.4,2,0))
```

```
A
```

```
      [,1] [,2] [,3]
[1,]    4 -4.0    2
[2,]    5  0.4    0
```

El argumento  $data = c(4, 5, -4, 0.4, 2, 0)$  generó la disposición de los datos completando sucesivamente las columnas, donde el 4 estuvo en la posición  $i, j = 1, 1$  y el 5 en  $i, j = 1, 2$ , para completar la primera columna. Si los datos estuviesen en la forma  $data = c(4, -4, 2, 5, 0.4, 0)$ , se requeriría incluir el argumento  $byrow = TRUE$ , para que la disposición de los datos complete primero las filas, de forma que el 4 esté en la posición  $i, j = 1, 1$ , el  $-4$  en  $i, j = 1, 2$  y el 2 en  $i, j = 3, 1$  (primera fila completa). Veamos:

```
R Studio

A=matrix(nrow=2,ncol=3, data=c(4,-4,2, 5,0.4,0), byrow=TRUE)
A
      [,1] [,2] [,3]
[1,]    4 -4.0    2
[2,]    5  0.4    0
```

Las matrices de una sola fila ( $1 \times n$ ) o una sola columna ( $m \times 1$ ) se denominan vectores y se denotan con letras minúsculas; veamos un ejemplo para cada caso:

$$b_{1,3} = [-5 \quad -4 \quad 2]$$

$$c_{3,1} = \begin{bmatrix} -5 \\ 0.34 \\ -3 \end{bmatrix}$$

La programación en R-project para la creación del vector fila  $b$  y el vector columna  $c$  sería:

```
R Studio

b=matrix(nrow=1,ncol=3, data=c(-5,-4,2))
b
c=matrix(nrow=3,ncol=1, data=c(-5,.3,-3))
c
> b
      [,1] [,2] [,3]
[1,]   -5   -4    2
> c
      [,1]
[1,] -5.0
[2,]  0.3
[3,] -3.0
```

Veamos un arreglo matricial con elementos relacionados con la producción de huevos entre la primera y sexta semana de postura, de gallinas que estaban alojadas en granjas de un proyecto de una comunidad rural:

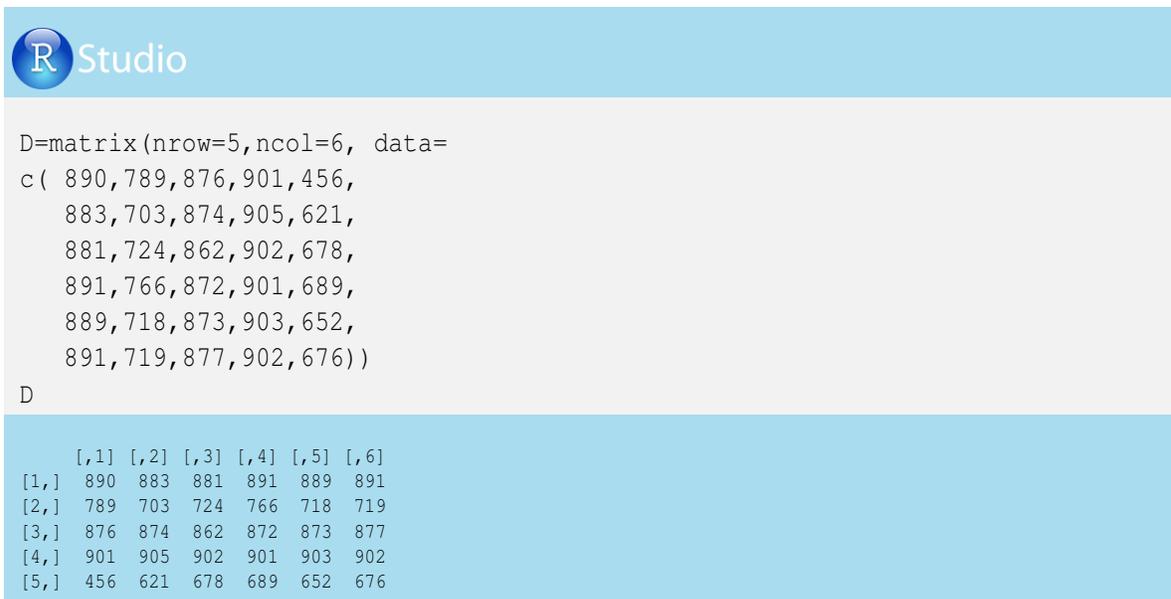
Granja	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6
1	890	883	881	891	889	891
2	789	703	724	766	718	719
3	876	874	862	872	873	877
4	901	905	902	901	903	902
5	456	621	678	689	652	676

Esta información permite construir una matriz de producciones semanales de huevo en cinco galpones (filas,  $m = 5$ ) por seis semanas (columnas,  $n = 6$ ):

$$D_{5,6} = \begin{bmatrix} 890 & 883 & 881 & 891 & 889 & 891 \\ 789 & 703 & 724 & 766 & 718 & 719 \\ 876 & 874 & 862 & 872 & 873 & 877 \\ 901 & 905 & 902 & 901 & 903 & 902 \\ 456 & 621 & 678 & 689 & 652 & 676 \end{bmatrix}$$

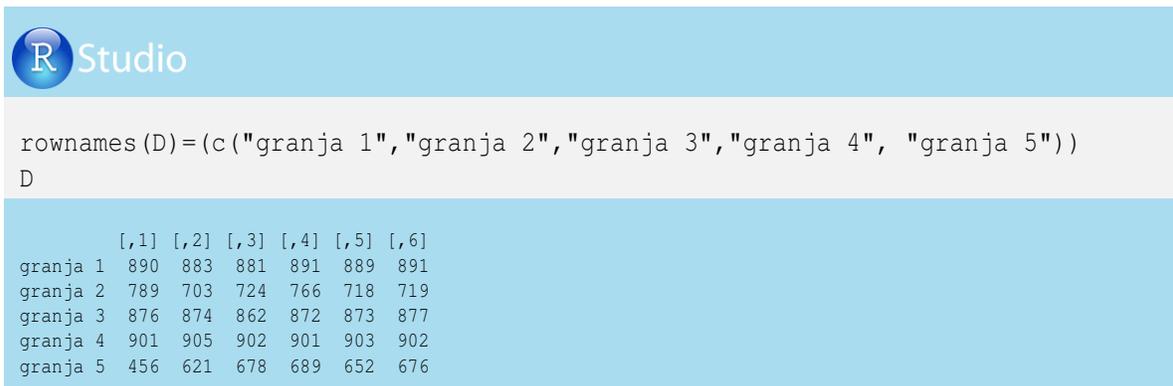
La matriz  $D$  tiene 30 elementos, ubicados en las posiciones  $d_{11}$  a  $d_{56}$ .

Con esta información, la creación de la matriz  $D$  en R-project se realiza de la siguiente manera:



```
D=matrix(nrow=5,ncol=6, data=
c( 890,789,876,901,456,
  883,703,874,905,621,
  881,724,862,902,678,
  891,766,872,901,689,
  889,718,873,903,652,
  891,719,877,902,676))
D
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 890  883  881  891  889  891
[2,] 789  703  724  766  718  719
[3,] 876  874  862  872  873  877
[4,] 901  905  902  901  903  902
[5,] 456  621  678  689  652  676
```

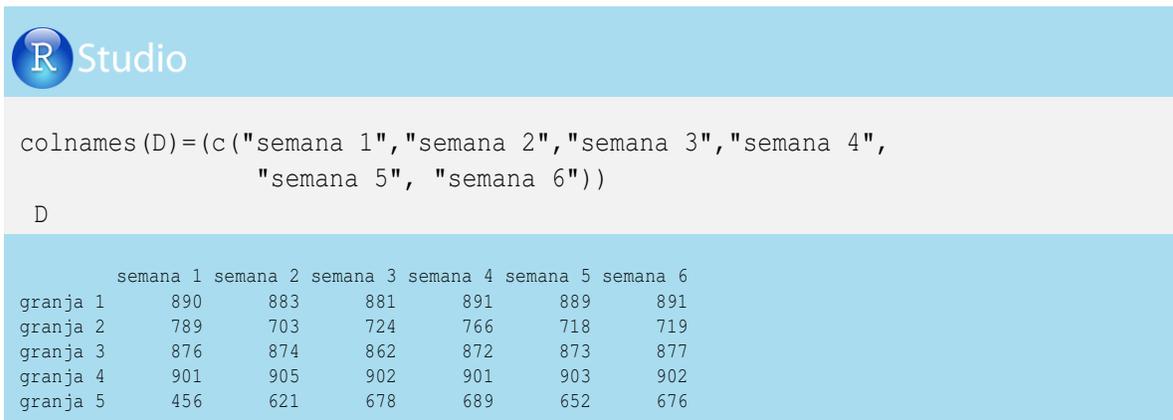
Para asignarles nombres a las granjas (filas) utilizamos el comando *rownames*:



```
rownames(D)=(c("granja 1","granja 2","granja 3","granja 4", "granja 5"))
D
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
granja 1	890	883	881	891	889	891
granja 2	789	703	724	766	718	719
granja 3	876	874	862	872	873	877
granja 4	901	905	902	901	903	902
granja 5	456	621	678	689	652	676

También les podemos asignar nombres a las columnas (semana de postura), utilizando el comando *colnames*:



```
colnames(D)=(c("semana 1","semana 2","semana 3","semana 4",
               "semana 5", "semana 6"))
D
```

	semana 1	semana 2	semana 3	semana 4	semana 5	semana 6
granja 1	890	883	881	891	889	891
granja 2	789	703	724	766	718	719
granja 3	876	874	862	872	873	877
granja 4	901	905	902	901	903	902
granja 5	456	621	678	689	652	676

Podemos generar un vector a partir de la matriz *D*, que contenga la información de la producción de huevos de la primera semana; a ese vector lo llamaremos *y*, así:

$$y_{5,1} = \begin{bmatrix} 890 \\ 789 \\ 876 \\ 901 \\ 456 \end{bmatrix}$$

Para obtener el vector columna  $y$  de la matriz  $D$  en R-project, utilizamos la siguiente programación:

```
R Studio
y=matrix(D[,1])
y
      [,1]
[1,] 890
[2,] 789
[3,] 876
[4,] 901
[5,] 456
```

Al interior de las matrices o vectores se pueden cambiar los valores de las filas por los valores de las columnas. La transpuesta de la matriz  $D_{5,6}$  es una matriz que se obtiene al colocar como filas las columnas, dando como resultado una matriz de tamaño  $i, j = 6, 5$ , cuya notación será  $D'_{6,5}$  y sus valores son:

$$D'_{6,5} = \begin{bmatrix} 890 & 789 & 876 & 901 & 456 \\ 883 & 703 & 874 & 905 & 621 \\ 881 & 724 & 862 & 902 & 678 \\ 891 & 766 & 872 & 901 & 689 \\ 889 & 718 & 873 & 903 & 652 \\ 891 & 719 & 877 & 902 & 676 \end{bmatrix}$$

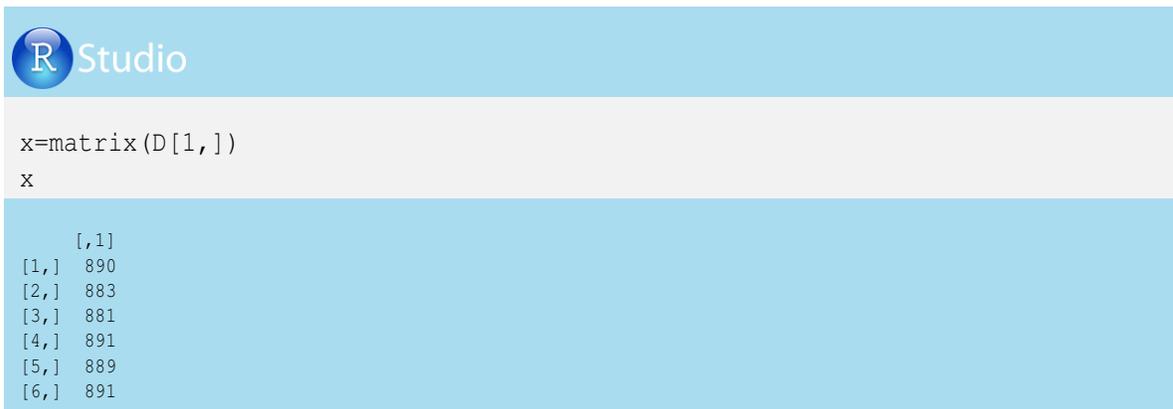
La programación en R-project para la transpuesta de una matriz o vector requiere escribir una  $t$  antes de los paréntesis que contienen el nombre de la matriz a transponer; veamos:

```
R Studio
Dprima=t (D)
Dprima
      granja 1 granja 2 granja 3 granja 4 granja 5
[1,]      890      789      876      901      456
[2,]      883      703      874      905      621
[3,]      881      724      862      902      678
[4,]      891      766      872      901      689
[5,]      889      718      873      903      652
[6,]      891      719      877      902      676
```

Retomando el ejercicio de obtener un vector columna  $y$  de la matriz  $D$ , también podemos generar un vector de la primera fila, al cual llamaremos  $x$  y estaría dado por:

$$x_{1,6} = [890 \ 883 \ 881 \ 891 \ 889 \ 891]$$

La programación para la generación de un vector de la primera fila de  $D$  sería:



```

R Studio

x=matrix(D[1,])
x

      [,1]
[1,] 890
[2,] 883
[3,] 881
[4,] 891
[5,] 889
[6,] 891
    
```

Pero  $x$  quedó como un vector columna, y se requiere que quede como vector fila, para esto utilizamos el comando  $t$  que genera la transpuesta, así:



```

R Studio

x=t(x)
x

      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 890 883 881 891 889 891
    
```

## 1.2. Sumatorias al interior de una matriz

### 1.2.1. Suma por filas

Si deseamos calcular la producción total de huevos durante las seis semanas en la primera granja, tenemos que sumar los elementos de la primera fila de la matriz  $D_{5,6}$ , así:

$$d_{1.} = \sum_{j=1}^n d_{1j}$$

Donde  $d_{1.}$  representa todos los valores de la matriz  $D_{5,6}$  que cumplan con la condición de estar en la fila 1 y en todas las columnas. En otras palabras, el 1 nos fija en la primera fila de  $D_{5,6}$  y el punto (.) nos indica que son todos los datos de las columnas.

En el ejemplo estudiado,  $j$  varía de 1 hasta  $n$ , donde  $n = 6$  y estaría dado por:

$$d_{1.} = \sum_{j=1}^n d_{1j} = (d_{11} + d_{12} + d_{13} + d_{14} + d_{15} + d_{16})$$

Remplazando por los valores numéricos, tenemos que la producción total de huevos en la primera granja estaría dada por:

$$d_{1.} = (890 + 883 + 881 + 891 + 889 + 891) = 5325$$

Veamos la programación en R-project para obtener la suma de los valores que están en la primera fila de la matriz  $D$ .



```
R Studio
d1.=sum(D[1,])
d1.
[1] 5325
```

### 1.2.2. Suma por columnas

Supongamos que se desea calcular la sumatoria de producción total de huevos en la cuarta semana; en este caso, tendríamos que realizar el siguiente cálculo:

$$d_{.4} = \sum_{i=1}^m d_{i4}$$

El signo punto (.) en  $d_{.4}$  representa a todos los valores de todas las cinco filas (cinco granjas) de la matriz  $D_{5,6}$  y el 4 nos fija en la cuarta columna (cuarta semana).

En el ejemplo estudiado, la  $i$  varía de 1 hasta  $m$ , donde  $m = 5$ . La suma de la cuarta columna estaría dada por:

$$d_{.4} = \sum_{i=1}^m d_{i4} = (d_{14} + d_{24} + d_{34} + d_{44} + d_{54})$$

Entonces la producción total de huevos de las cinco granjas en la cuarta semana sería:

$$d_{.4} = (891 + 766 + 872 + 901 + 689) = 4119$$

Veamos la programación en R-project para obtener la suma de los valores que están en la cuarta columna de la matriz  $D_{5,6}$ :



```
R Studio
d.=sum(D[,4])
d.4
[1] 4119
```

### 1.2.3. Suma de valores de algunas filas y columnas

Supongamos que es de particular interés calcular la producción de huevos en los tres primeros galpones durante las dos primeras semanas. En este caso tendríamos que realizar la siguiente doble sumatoria:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij}$$

Donde la  $i$  varía de 1 hasta 3 (tres primeros galpones) y la  $j$  varía de 1 hasta 2 (dos primeras semanas), y la suma estaría dada por:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij} = (d_{11} + d_{12} + d_{21} + d_{22} + d_{31} + d_{32})$$

Con los valores numéricos (producción de huevos), se tiene:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij} = (890 + 883 + 789 + 703 + 876 + 874) = 5015$$

Para obtener la suma de los valores que están en las tres primeras filas y en las dos primeras columnas de la matriz  $D_{5,6}$ , debemos utilizar la siguiente programación:



```

R Studio

di13j12=sum(D[1:3,1:2])
di13j12

[1] 5015
    
```

Ahora obtengamos la sumatoria de los valores del ejemplo anterior al cuadrado, es decir:

$$\sum_{i=1}^3 \sum_{j=1}^2 (d_{ij})^2$$

La sumatoria de los valores al cuadrado estaría dada por:

$$\sum_{i=1}^3 \sum_{j=1}^2 (d_{ij})^2 = (d_{11})^2 + (d_{12})^2 + (d_{21})^2 + (d_{22})^2 + (d_{31})^2 + (d_{32})^2$$

La sumatoria al cuadrado de la producción de huevos será:

$$\sum_{i=1}^3 \sum_{j=1}^2 (d_{ij})^2 = 890^2 + 883^2 + 789^2 + 703^2 + 876^2 + 874^2 = 4219771$$

La programación para calcular esta suma de cuadrados en R-project es:

```
R Studio
di13j12a12=sum(D[1:3,1:2]^2)
di13j12a12
[1] 4219771
```

Ahora realicemos la sumatoria de las dos primeras semanas, sin tener en cuenta la granja 2. En este caso tendríamos que realizar la siguiente doble sumatoria, sin incluir la fila 2:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij}, i \neq 2$$

La  $i$  varía de 1 hasta 3, pero sin la fila 2 (la granja 1 y la 3), y la  $j$  varía de 1 hasta 2 (las dos primeras semanas). La suma estaría dada por:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij}, i \neq 2 = d_{11} + d_{12} + d_{31} + d_{32}$$

La suma de producción de huevos será:

$$\sum_{i=1}^3 \sum_{j=1}^2 d_{ij}, i \neq 2 = 890 + 883 + 876 + 874 = 3523$$

Para obtener la suma de los valores que están en la filas 1 y 3 y además en las dos primeras columnas de la matriz  $D_{5,6}$ , se procede de la siguiente forma:

```
R Studio
di13j12sini2=sum(D[1:3,1:2], -D[2,1:2])
di13j12sini2
[1] 3523
```

#### 1.2.4. Operaciones utilizando la función *apply*

La suma de filas y columnas se puede realizar con la función *apply*. Esta función también permite encontrar los valores mínimo y máximo y calcular la media, la mediana y otros valores de una serie de datos.

Veamos el caso de la suma de la producción de huevos por granja (por filas de la matriz  $D_{5,6}$ ). Pero antes, detallemos la siguiente expresión para la programación en R-project:

$$[1 : 5, 1 : 6], 1, sum$$

Donde  $1 : 5$  indica que se tendrá en cuenta desde la fila 1 hasta la 5 y  $1 : 6$  que se tendrá en cuenta desde la columna 1 hasta la 6. El siguiente valor (en este caso el 1) ordena que se realice la operación por filas, y el argumento *sum* genera una sumatoria.

```
R Studio  
dsumaporfila=apply (D[1:5,1:6],1,sum)  
dsumaporfila  
granja 1 granja 2 granja 3 granja 4 granja 5  
5325 4419 5234 5414 3772
```

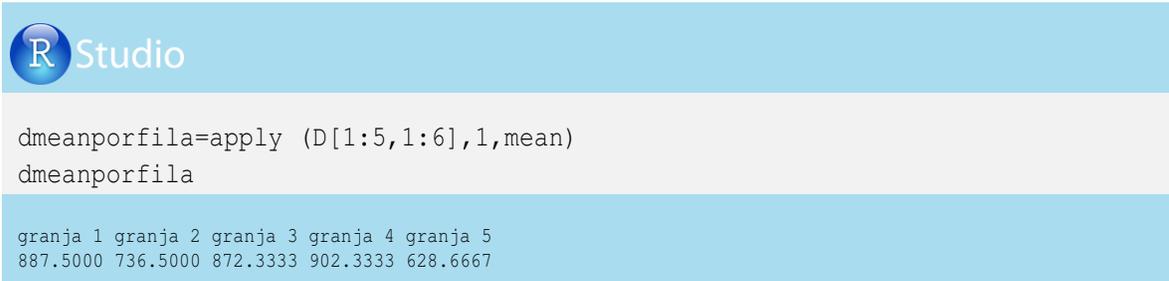
Como estamos indicando que se tienen en cuenta todos los valores de la matriz ( $i = 1..5$  y  $j = 1..6$ ) de  $D_{5,6}$ , entonces no es necesario especificar  $D[1 : 5, 1 : 6]$ , solamente que es  $D$ , de modo que queda *apply(D, 1, sum)*, como se indica a continuación:

```
R Studio  
dsumaporfila=apply (D,1,sum)  
dsumaporfila  
granja 1 granja 2 granja 3 granja 4 granja 5  
5325 4419 5234 5414 3772
```

Para realizar la suma por columnas, en la expresión cambiamos  $[1 : 5, 1 : 6], 1, sum$  por  $[1 : 5, 1 : 6], 2, sum$ , reemplazando únicamente el 1 de fila por el 2 de columna; veamos el ejercicio en R-project:

```
R Studio  
dsumaporcol=apply (D[1:5,1:6],2,sum)  
dsumaporcol  
semana 1 semana 2 semana 3 semana 4 semana 5 semana 6  
3912 3986 4047 4119 4035 4065
```

Para realizar el promedio de las filas, en la expresión cambiamos  $[1 : 5, 1 : 6], 1, sum$  por  $[1 : 5, 1 : 6], 1, mean$ ; veamos el ejercicio en R-project:



```
dmeanporfila=apply (D[1:5,1:6],1,mean)
dmeanporfila
```

```
granja 1 granja 2 granja 3 granja 4 granja 5
887.5000 736.5000 872.3333 902.3333 628.6667
```

Se puede utilizar simplemente la expresión  $apply(D, 1, mean)$ .

### 1.3. Traza de una matriz

En las matrices cuadradas (donde el número de filas es igual al número de columnas,  $m = n$ ) se puede calcular la traza de una matriz ( $tr$ ), que está dada por la sumatoria de los elementos de su diagonal principal (donde  $i = j$ ).

En una matriz ( $E_{m,n}$ ), la  $tr(E)$  se define por medio de la siguiente expresión:

$$tr(E) = \sum_{i=1}^m e_{ii} = e_{1,1} + e_{2,2} + e_{3,3} + \dots + e_{m,m}$$

Antes de realizar un ejemplo de cálculo de traza, construyamos la matriz  $E$  a partir de la matriz  $D$  con el propósito de obtener una matriz cuadrada. Esta matriz contiene la información de las tres primeras granjas y las tres primeras semanas:

$$E_{3,3} = \begin{bmatrix} 890 & 883 & 881 \\ 789 & 703 & 724 \\ 876 & 874 & 862 \end{bmatrix}$$

La traza de la matriz  $E$  estaría dada por la sumatoria de todos los elementos donde  $i = j$ , con la siguiente expresión:

$$tr(E) = \sum_{i=1}^3 e_{ii} = e_{1,1} + e_{2,2} + e_{3,3}$$

Los valores de producción de huevos de la traza de  $E_{3,3}$ , relacionados con tres granjas y tres semanas, serían:

$$tr(E) = \sum_{i=1}^3 e_{ii} = 890 + 703 + 862 = 2455$$

Iniciemos con la obtención de la matriz  $E$  a partir de la matriz  $D$  en R-project:

```
R Studio
E=(D[1:3,1:3])
E
```

	semana 1	semana 2	semana 3
granja 1	890	883	881
granja 2	789	703	724
granja 3	876	874	862

Para calcular la traza debemos obtener la diagonal principal de la matriz  $E$  mediante el comando *diag*, generando un vector de los tres valores de la diagonal; luego procedemos a sumar esos tres valores con el comando *sum*, como se indica a continuación:

```
R Studio
diagonaldeE=diag(E)
diagonaldeE
trazaE=sum(diagonaldeE)
trazaE
```

```
> diagonaldeE
[1] 890 703 862
> trazaE
[1] 2455
```

#### 1.4. Suma y resta de matrices

Para sumar y restar matrices se requiere que ambas matrices tengan la misma dimensión u orden, y la operación de interés se realizará entre elementos de la misma posición.

Veamos el caso de la resta de la producción de huevos semanales por granja de la matriz  $E$  y el número de huevos rotos (matriz  $F$ ), que origina el número de huevos aptos para la venta (matriz  $G$ ).

$$E_{33} - F_{33} = G_{33}$$

$$\begin{bmatrix} 890 & 883 & 881 \\ 789 & 703 & 724 \\ 876 & 874 & 862 \end{bmatrix} - \begin{bmatrix} 0 & 10 & 11 \\ 25 & 22 & 24 \\ 70 & 76 & 43 \end{bmatrix} = \begin{bmatrix} 890 & 873 & 870 \\ 764 & 681 & 700 \\ 806 & 798 & 819 \end{bmatrix}$$

La programación en R-project requiere de la definición de  $F$  para realizar el cálculo, y sería:

```

R Studio

F=matrix(nrow=3,ncol=3,c(0,25,70,10,22,76,11,24,43))
F
G=E-F
G

> F
      [,1] [,2] [,3]
[1,]    0   10   11
[2,]   25   22   24
[3,]   70   76   43
> G
      semana 1 semana 2 semana 3
granja 1     890     873     870
granja 2     764     681     700
granja 3     806     798     819
    
```

### 1.5. Multiplicación de matrices

Para multiplicar dos matrices ( $G_{m,n} * H_{k,l}$ ), se requiere que el número de columnas de la primera matriz sea igual al número de filas de la segunda ( $n = k$ ). Considerando que no existe la propiedad conmutativa, el producto será una matriz de orden  $m,l$ , así:

$$G_{m,n} * H_{k,l} = M_{m,l}$$

El valor obtenido para cada elemento estará dado por:

$$G_{m,n} * H_{k,l} = M_{m,l}$$

$$\begin{bmatrix} g_{11} & g_{12} & \dots & g_{1n} \\ g_{21} & g_{22} & \dots & g_{2n} \\ \dots & \dots & \dots & \dots \\ g_{m1} & g_{m21} & \dots & g_{mn} \end{bmatrix} * \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1l} \\ h_{21} & h_{22} & \dots & h_{2l} \\ \dots & \dots & \dots & \dots \\ h_{k1} & h_{k2} & \dots & h_{kl} \end{bmatrix} =$$

$$\begin{bmatrix} (g_{11} * h_{11}) + (g_{12} * h_{21}) + \dots + (g_{1n} * h_{k1}) & \dots & (g_{21} * h_{11}) + (g_{22} * h_{21}) + \dots + (g_{2n} * h_{kl}) \\ (g_{21} * h_{11}) + (g_{22} * h_{21}) + \dots + (g_{2n} * h_{k1}) & \dots & (g_{21} * h_{1l}) + (g_{22} * h_{2l}) + \dots + (g_{2n} * h_{kl}) \\ \dots & \dots & \dots \\ (g_{m1} * h_{11}) + (g_{m2} * h_{21}) + \dots + (g_{mn} * h_{k1}) & \dots & (g_{m1} * h_{1l}) + (g_{m2} * h_{2l}) + \dots + (g_{mn} * h_{kl}) \end{bmatrix}$$

Veamos el caso de los ingresos generados por la venta de huevos para tres galpones, si el precio del huevo estuvo en \$302, \$304 y \$299 para la primera, segunda y tercera semana, respectivamente. La matriz  $G_{m,n}$  relaciona el número de huevos vendidos en las tres granjas ( $m = 3$ , filas) en las tres semanas ( $n = 3$ , columnas), y el vector  $h_{k,l}$  relaciona los precios semanales.

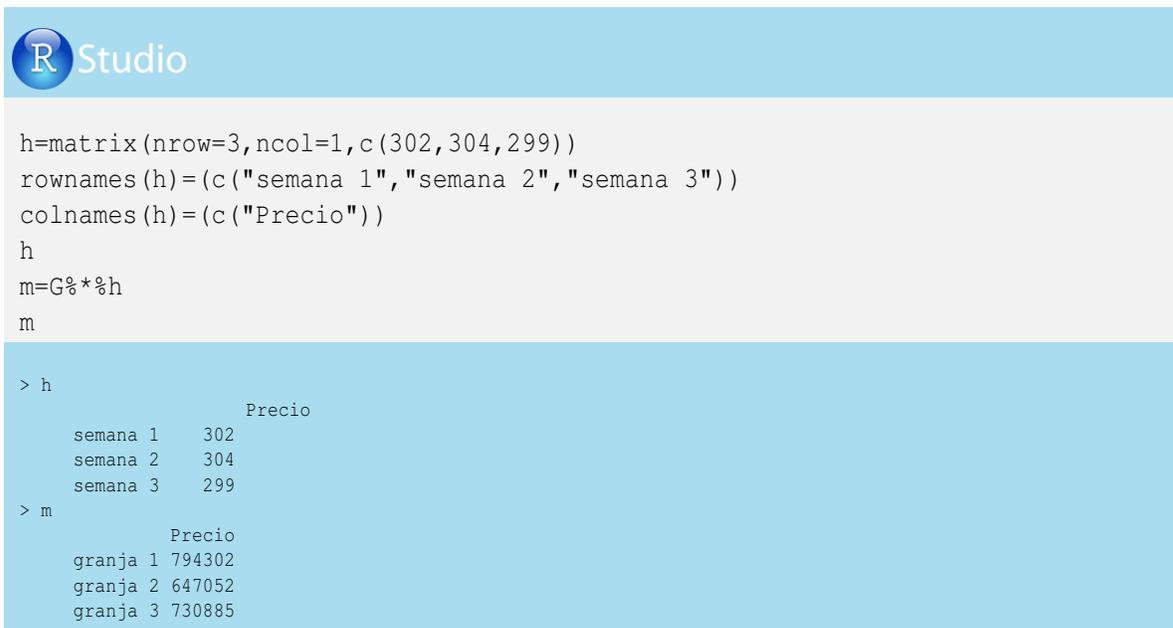
El producto de  $G * h$  originará el vector  $m_{m,l}$ , el cual presenta en cada fila los ingresos totales en cada semana. La operación se puede realizar porque  $n = k = 3$ . Entonces los datos serían:

$$G_{33} * h_{31} = m_{31}$$

$$\begin{bmatrix} 890 & 873 & 870 \\ 764 & 681 & 700 \\ 806 & 798 & 819 \end{bmatrix} * \begin{bmatrix} 302 \\ 304 \\ 299 \end{bmatrix} = \begin{bmatrix} (890 * 302) + (873 * 304) + (870 * 299) \\ (764 * 302) + (681 * 304) + (700 * 299) \\ (806 * 302) + (798 * 304) + (819 * 299) \end{bmatrix} = \begin{bmatrix} 794302 \\ 647052 \\ 730885 \end{bmatrix}$$

Los ingresos en pesos colombianos fueron \$794302, \$647052 y \$730885 para la primera, segunda y tercera semana, respectivamente.

En R-project hay que tener en cuenta que para multiplicar dos matrices se utiliza `%*%`:



```

R Studio

h=matrix(nrow=3,ncol=1,c(302,304,299))
rownames(h)=(c("semana 1","semana 2","semana 3"))
colnames(h)=(c("Precio"))
h
m=G%*%h
m

> h
      Precio
semana 1  302
semana 2  304
semana 3  299

> m
      Precio
granja 1 794302
granja 2 647052
granja 3 730885
    
```

Si se desea saber cuáles fueron los ingresos por granja, se tiene que transponer la matriz  $G$  y realizar el siguiente producto matricial:

$$G'_{33} * h_{31} = n_{31}$$

$$\begin{bmatrix} 890 & 764 & 806 \\ 873 & 681 & 798 \\ 870 & 700 & 819 \end{bmatrix} * \begin{bmatrix} 302 \\ 304 \\ 299 \end{bmatrix} = \begin{bmatrix} (890 * 302) + (764 * 304) + (806 * 299) \\ (873 * 302) + (681 * 304) + (798 * 299) \\ (870 * 302) + (700 * 304) + (819 * 299) \end{bmatrix} = \begin{bmatrix} 742030 \\ 709272 \\ 720421 \end{bmatrix}$$

Los ingresos fueron \$742030, \$709272 y \$720421 para la primera, segunda y tercera granja, respectivamente.

La programación en R-project sería:

```

R Studio

Gprima=t(G)
Gprima
n=Gprima%*%h
n

> Gprima
      granja 1 granja 2 granja 3
semana 1    890    764    806
semana 2    873    681    798
semana 3    870    700    819

> n
      Precio
semana 1 742030
semana 2 709272
semana 3 720421
    
```

### 1.6. Multiplicación de una matriz por un escalar

En el caso de la multiplicación de una matriz por un escalar, se realiza de forma directa, mediante el producto directo o producto de Kronecker, cuyo símbolo es  $\otimes$ .

Realicemos la multiplicación de la matriz  $G_{mn}$  con el escalar  $q_{11}$ . Para esto, se multiplica directamente cada elemento de la matriz por el valor del escalar, y el producto directo será una matriz de orden  $m \times n$ , así:

$$(G_{mn} \otimes q_{11}) = P_{mn}$$

$$\begin{bmatrix} g_{11} & \cdot & g_{1n} \\ g_{21} & \cdot & g_{2n} \\ \cdot & \cdot & \cdot \\ g_{m1} & \cdot & g_{mn} \end{bmatrix} \otimes [q_{11}] = \begin{bmatrix} g_{11} * q_{11} & \cdot & g_{1n} * q_{11} \\ g_{21} * q_{11} & \cdot & g_{2n} * q_{11} \\ \cdot & \cdot & \cdot \\ g_{m1} * q_{11} & \cdot & g_{mn} * q_{11} \end{bmatrix}$$

De un estudio previo en nuestras granjas, en el que se tuvo en cuenta la edad de postura, la línea genética y la alimentación de las aves, se espera que el 10% sean huevos de la categoría AAA (entre 67 y 77.9 gramos). Si multiplicamos la producción de huevos aptos para la venta por el porcentaje de huevos AAA, tendremos el número de huevos AAA producidos.

Retomemos la matriz  $G_{mn}$  de huevos aptos para la venta en las  $m = 3$  granjas y en las  $n = 3$  semanas y multipliquémosla por un escalar  $q_{11}$  de valor 0.1. Esta multiplicación debe realizarse mediante el producto directo ( $G_{mn} \otimes q_{11}$ ) y originará la matriz  $P_{mn}$ , relacionada con número de huevos AAA producidos, así:

$$G_{33} \otimes h_{11} = P_{33}$$

$$\begin{bmatrix} 890 & 873 & 870 \\ 764 & 681 & 700 \\ 806 & 798 & 819 \end{bmatrix} \otimes [0.1] = \begin{bmatrix} (890 * 0.1) & (873 * 0.1) & (870 * 0.1) \\ (764 * 0.1) & (681 * 0.1) & (700 * 0.1) \\ (806 * 0.1) & (798 * 0.1) & (819 * 0.1) \end{bmatrix} = \begin{bmatrix} 89 & 87 & 87 \\ 76 & 68 & 70 \\ 81 & 80 & 82 \end{bmatrix}$$

En R-project, el producto de Kronecker está dado por la expresión `%x%`, diferente a la expresión de la multiplicación, que es `%*%`. En la siguiente programación le incluimos el comando `round` para dejar los resultados sin decimales:

```

R Studio

q=matrix(nrow=1,ncol=1,c(0.1))
q
El producto es una matriz (3x3)
P=G%x%q
P
P=round (P)
colnames(P)=(c("semana 1","semana 2","semana 3"))
rownames(P)=(c("granja 1","granja 2","granja 3"))
P

      [,1]
[1,] 0.1
> P
      [,1] [,2] [,3]
[1,] 89.0 87.3 87.0
[2,] 76.4 68.1 70.0
[3,] 80.6 79.8 81.9
> P
      semana 1 semana 2 semana 3
granja 1      89      87      87
granja 2      76      68      70
granja 3      81      80      82

```

El número de huevos tipo AAA varió en la granja 1 de 87 a 89 en las tres semanas, y el menor número de huevos de esta categoría lo presentó la granja 2 (68 huevos).

### 1.7. Multiplicación directa de matrices

Sean  $G$  una matriz de tamaño  $m, n$  y  $R$  una matriz de tamaño  $k, l$ ; el producto de Kronecker de estas dos matrices es una matriz bloque de dimensión  $m * k, n * l$ , así:

$$G_{mn} \otimes R_{kl} = S_{m*k, n*l}$$

Si  $G$  es una matriz de tamaño  $m = 3$  filas y  $n = 3$  columnas y  $R$  una matriz de tamaño  $k = 2$  filas y  $l = 2$  columnas, el producto directo de estas dos matrices será la matriz  $S$  con  $m * k = 3 * 2 = 6$  filas y  $n * l = 3 * 2 = 6$  columnas ( $S_{6,6}$ ), y el valor de cada elemento estará dado por:

$$G_{33} \otimes R_{22} = S_{66}$$

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \otimes \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} g_{11}r_{11} & g_{11}r_{12} & g_{12}r_{11} & g_{12}r_{12} & g_{13}r_{11} & g_{13}r_{12} \\ g_{11}r_{21} & g_{11}r_{22} & g_{12}r_{21} & g_{12}r_{22} & g_{13}r_{21} & g_{13}r_{22} \\ g_{21}r_{11} & g_{21}r_{12} & g_{22}r_{11} & g_{22}r_{12} & g_{23}r_{11} & g_{23}r_{12} \\ g_{21}r_{21} & g_{21}r_{22} & g_{22}r_{21} & g_{22}r_{22} & g_{23}r_{21} & g_{23}r_{22} \\ g_{31}r_{11} & g_{31}r_{12} & g_{32}r_{11} & g_{32}r_{12} & g_{33}r_{11} & g_{33}r_{12} \\ g_{31}r_{21} & g_{31}r_{22} & g_{32}r_{21} & g_{32}r_{22} & g_{33}r_{21} & g_{33}r_{22} \end{bmatrix}$$

Dicho de otra forma:

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \otimes \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} g_{11}R & g_{12}R & g_{13}R \\ g_{21}R & g_{22}R & g_{23}R \\ g_{31}R & g_{32}R & g_{33}R \end{bmatrix}$$

Retomando el ejemplo del tipo de huevo, se espera que el 10% de los huevos aptos para la venta sean AAA, el 60% sean AA y el 28% A.

Para conocer el número de huevos por categoría en cada granja y en cada semana, utilizamos la matriz  $G_{mn}$ , que representa el número de huevos aptos para la venta en las tres granjas ( $m = 3$  filas) y las tres semanas ( $n = 3$  columnas), y la multiplicamos con un vector columna  $r_{1 \times 3}$  de valores 0.1, 0.6 y 0.28, mediante el producto de Kronecker, así:

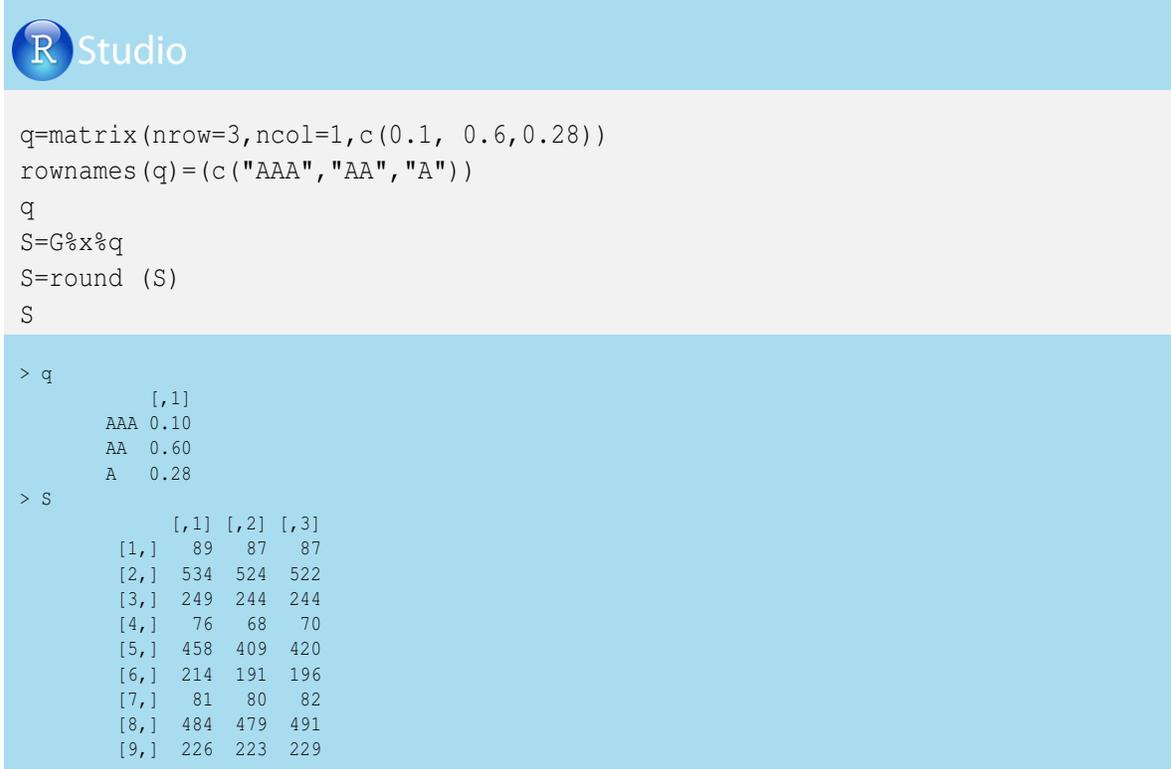
$$G_{33} \otimes R_{31} = S_{93}$$

$$\begin{bmatrix} 890 & 873 & 870 \\ 764 & 681 & 700 \\ 806 & 798 & 819 \end{bmatrix} \otimes \begin{bmatrix} 0.1 \\ 0.6 \\ 0.28 \end{bmatrix} = \begin{bmatrix} (890 * 0.1) & (873 * 0.1) & (870 * 0.1) \\ (890 * 0.6) & (873 * 0.6) & (870 * 0.6) \\ (890 * 0.28) & (873 * 0.28) & (870 * 0.28) \\ (764 * 0.1) & (681 * 0.1) & (700 * 0.1) \\ (764 * 0.6) & (681 * 0.6) & (700 * 0.6) \\ (764 * 0.28) & (681 * 0.28) & (700 * 0.28) \\ (806 * 0.1) & (798 * 0.1) & (819 * 0.1) \\ (806 * 0.6) & (798 * 0.6) & (819 * 0.6) \\ (806 * 0.28) & (798 * 0.28) & (819 * 0.28) \end{bmatrix} = \begin{bmatrix} 89 & 87 & 87 \\ 534 & 524 & 522 \\ 249 & 244 & 244 \\ 76 & 68 & 70 \\ 458 & 409 & 420 \\ 214 & 191 & 196 \\ 81 & 80 & 82 \\ 484 & 479 & 491 \\ 226 & 223 & 229 \end{bmatrix}$$

La matriz  $S_{93}$  representa el número de huevos AAA, AA y A en cada granja y semana, y estaría dada por:

Granja	Tipo de huevo	Semana 1	Semana 2	Semana 3
1	AAA	89	87	87
1	AA	534	524	522
1	A	249	244	244
2	AAA	76	68	70
2	AA	458	409	420
2	A	214	191	196
3	AAA	81	80	82
3	AA	484	479	491
3	A	226	223	229

La programación en R-project sería:



```

R Studio

q=matrix(nrow=3,ncol=1,c(0.1, 0.6,0.28))
rownames(q)=(c("AAA","AA","A"))
q
S=G%x%q
S=round(S)
S

> q
      [,1]
AAA 0.10
AA  0.60
A   0.28

> S
      [,1] [,2] [,3]
[1,]  89   87   87
[2,] 534  524  522
[3,] 249  244  244
[4,]  76   68   70
[5,] 458  409  420
[6,] 214  191  196
[7,]  81   80   82
[8,] 484  479  491
[9,] 226  223  229

```

## 1.8. Inversa de matrices

Para realizar división entre matrices, es necesario invertir la matriz del denominador y efectuar la multiplicación entre dicha inversa y la matriz del numerador. Una matriz tiene inversa si es cuadrada y su determinante es diferente de cero; en caso contrario, se utilizará la inversa generalizada, que se describirá más adelante.

En el caso de una matriz  $2 \times 2$ , su determinante está dado por el producto de los elementos de la diagonal principal, menos el producto de los elementos de la diagonal secundaria, así:

$$\det R_{mm} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = r_{11}r_{22} - r_{21}r_{12}$$

Veamos un ejemplo:

$$\det R_{mm} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = (0 * 3) - (2 * 1) = -2$$

Como el determinante es diferente de cero ( $det \neq 0$ ), entonces la matriz  $R$  tiene inversa.

La programación en R-project para encontrar el determinante de una matriz sería:

```

RStudio

R=matrix(ncol=2,nrow=2,c(0,2,1,3))
R
Rdet= det(R)
Rdet

      [,1] [,2]
[1,]    0    1
[2,]    2    3
> Rdet
[1] -2
    
```

El fundamento para obtener la inversa de una matriz es que el producto de esta por su inversa genere una matriz identidad (matriz de unos en la diagonal principal y de ceros fuera de esta).

Tomemos como ejemplo la matriz  $R_{mm}$ , cuya inversa sería  $R_{mm}^{-1}$ , y una matriz identidad ( $I_{mm}$ ), para decir:

$$R_{mm} * R_{mm}^{-1} = I_{mm}$$

$$\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (r_{11} * a) + (r_{12} * c) & (r_{11} * b) + (r_{12} * d) \\ (r_{21} * a) + (r_{22} * c) & (r_{21} * b) + (r_{22} * d) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Por consiguiente, se tendrían cuatro ecuaciones con cuatro incógnitas. Para mayor claridad, veamos un ejemplo:

$$R_{mm} * R_{mm}^{-1} = I_{mm}$$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (0 * a) + (1 * c) & (0 * b) + (1 * d) \\ (2 * a) + (3 * c) & (2 * b) + (3 * d) \end{bmatrix}$$

En el elemento  $i_{11}$ :

$$\begin{aligned} (0 * a) + (1 * c) &= 1 \\ 0 + c &= 1 \\ c &= 1 \end{aligned}$$

En el elemento  $i_{21}$ , reemplazamos  $c$  por 1:

$$\begin{aligned} (2 * a) + (3 * c) &= 0 \\ (2 * a) + (3 * 1) &= 0 \\ 2a + 3 &= 0 \\ a &= -\frac{3}{2} \end{aligned}$$

En el elemento  $i_{12}$ :

$$\begin{aligned} (0 * b) + (1 * d) &= 0 \\ 1d &= 0 \\ d &= 0 \end{aligned}$$

En el elemento  $i_{22}$ , reemplazamos  $d$  por 0:

$$\begin{aligned}(2 * b) + (3 * d) &= 1 \\(2 * b) + (3 * 0) &= 1 \\2b &= 1 \\b &= \frac{1}{2}\end{aligned}$$

Por consiguiente:

$$R_{mm} * R_{mm}^{-1} = I_{mm}$$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} -\frac{3}{2} & \frac{1}{2} \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Para encontrar la inversa de una matriz en R-project, se puede utilizar la librería *MASS* (Model Applied Statistics with S) de Venables y Ripley (2002), de operaciones básicas en R-project, y utilizamos el comando *solve*, como se describe a continuación:



```

R Studio

install.packages ("MASS")
library (MASS)
Rinv= solve (R)
Rinv

      [,1] [,2]
[1,] -1.5  0.5
[2,]  1.0  0.0
    
```

En R-project se puede comprobar si el producto de una matriz por su inversa genera una matriz identidad. Para el ejemplo anterior, la programación en R-project sería:



```

R Studio

I=R%*%Rinv
I

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

### 1.9. Ecuación característica, autovalores y autovectores de una matriz

Toda matriz cuadrada obedece a una ecuación característica. Para obtener la ecuación característica igualamos el determinante a cero. En el caso de la matriz  $R$ , estaría dada por:

$$|R_{mm} - \lambda I_{mm}| = 0$$

Cuya solución sería:

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} -\lambda & 1 \\ 2 & 3 - \lambda \end{bmatrix} = 0$$

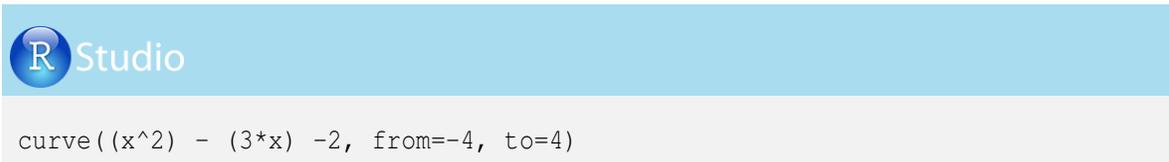
El desarrollo de la ecuación característica será:

$$\begin{aligned} -\lambda(3 - \lambda) - (2 * 1) &= 0 \\ -3\lambda + \lambda^2 - 2 &= 0 \\ \lambda^2 - 3\lambda - 2 &= 0 \end{aligned}$$

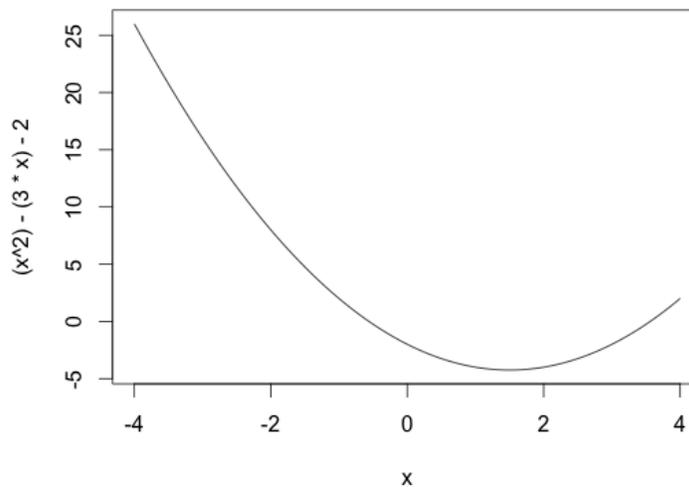
Para construir la gráfica en R-project utilizamos el comando `curve` con estos argumentos:

- ♣ la ecuación a graficar, que sería  $\lambda^2 - 3\lambda - 2 = 0$ , pero cambiamos  $\lambda$  por  $x$ , de modo que  $x^2 - 3x - 2 = 0$ ,
- ♣ el valor mínimo del eje  $x$  con la expresión `from =`, y
- ♣ el valor máximo del eje  $x$  con la expresión `to =`

La gráfica de la ecuación característica en R-project sería:



```
curve((x^2) - (3*x) - 2, from=-4, to=4)
```



Los valores que asumiría  $\lambda$  estarían dados por:

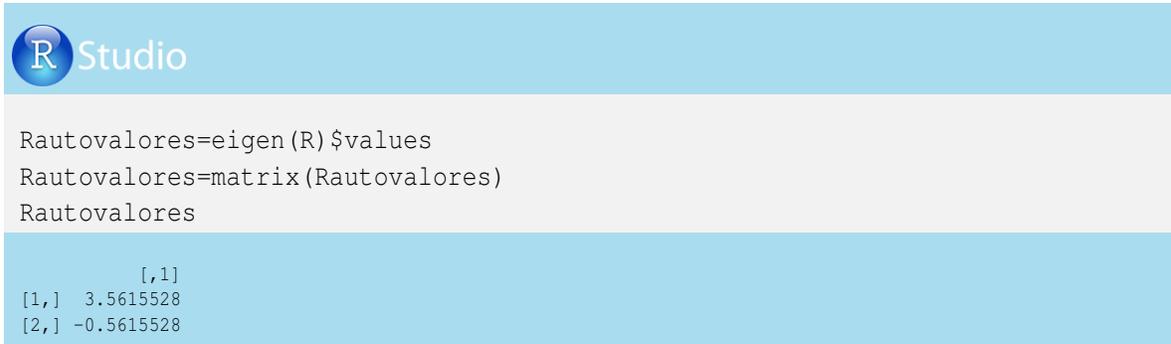
$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\lambda_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-3 + \sqrt{9 - 8}}{2} = 3.56$$

$$\lambda_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-3 - \sqrt{9 - 8}}{2} = -0.56$$

Los valores de  $\lambda_1 = 3.56$  y  $\lambda_2 = -0.56$  de la ecuación característica de la matriz en estudio, se denominan valores característicos, autovalores o *eigenvalues*.

La programación para obtener los autovalores en R-project sería:



```

R Studio

Rautovalores=eigen(R) $values
Rautovalores=matrix(Rautovalores)
Rautovalores

      [,1]
[1,]  3.5615528
[2,] -0.5615528
    
```

Estos autovalores están asociados a vectores propios ( $v$ ), donde cada autovalor y su respectivo autovector son operadores lineales de la matriz original.

La programación para obtener los autovectores en R-project sería:



```

R Studio

Rautovectores=eigen(R) $vectors
Rautovectores

      [,1]      [,2]
[1,] -0.2703230 -0.8719282
[2,] -0.9627697  0.4896337
    
```

Por tanto, los autovectores asociados a los autovalores  $\lambda_1$  y  $\lambda_2$  son, respectivamente:

$$\lambda_1 \rightarrow v_1 = \begin{bmatrix} -0.2703 \\ -0.9628 \end{bmatrix}$$

$$\lambda_2 \rightarrow v_2 = \begin{bmatrix} -0.8719 \\ -0.4896 \end{bmatrix}$$

Separemos el primer autovalor ( $\lambda_1$ ) y su autovector asociado.

```

R Studio

Lamda1=(Rautovalores [1,1])
Lamda1
autovector1=(Rautovectores[1:2,1])
autovector1

[1] 3.561553
> autovector1
[1] -0.2703230 -0.9627697

```

Realicemos la misma operación con  $\lambda_2$  y su autovector.

```

R Studio

Lamda2=(Rautovalores [2,1])
Lamda2
autovector2=(Rautovectores[1:2,2])
autovector2

[1] -0.5615528
> autovector2
[1] -0.8719282 0.4896337

```

Un autovalor multiplicado por su autovector es igual a la matriz por ese autovector. En el caso de la matriz  $R_{mm}$  con sus autovalores  $\lambda_1$  y  $\lambda_2$  asociados a sus respectivos autovectores, se tendrían las siguientes igualdades:

$$\lambda_1 I_{mm} v_1 = R_{mm} v_1$$

$$\lambda_2 I_{mm} v_2 = R_{mm} v_2$$

Donde  $I$  es una matriz identidad,  $v_1 \neq 0$  y  $v_2 \neq 0$ .

Para el caso de  $\lambda_1$  se tendría:

$$\lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * v_1 = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * v_1$$

$$3.56151528 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.2703 \\ -0.9628 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} -0.2703 \\ -0.9628 \end{bmatrix}$$

$$\begin{bmatrix} 3.56151528 & 0 \\ 0 & 3.56151528 \end{bmatrix} * \begin{bmatrix} -0.2703 \\ -0.9628 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} -0.2703 \\ -0.9628 \end{bmatrix}$$

$$\begin{bmatrix} -0.96277 \\ -3.42895 \end{bmatrix} = \begin{bmatrix} -0.96277 \\ -3.42895 \end{bmatrix}$$

La programación en R-project para obtener el lado izquierdo (lo llamaremos Lizq1) y el lado derecho (lo llamaremos Lder1) de la igualdad es la siguiente:



```

R Studio

Lizq1=R%*%autovector1
Lizq1
Lder1=Lamda1%x%I%*%autovector1
Lder1

> Lizq1
      [,1]
[1,] -0.9627697
[2,] -3.4289551
> Lder1
      [,1]
[1,] -0.9627697
[2,] -3.4289551
    
```

Para el caso de  $\lambda_2$  se tendría:

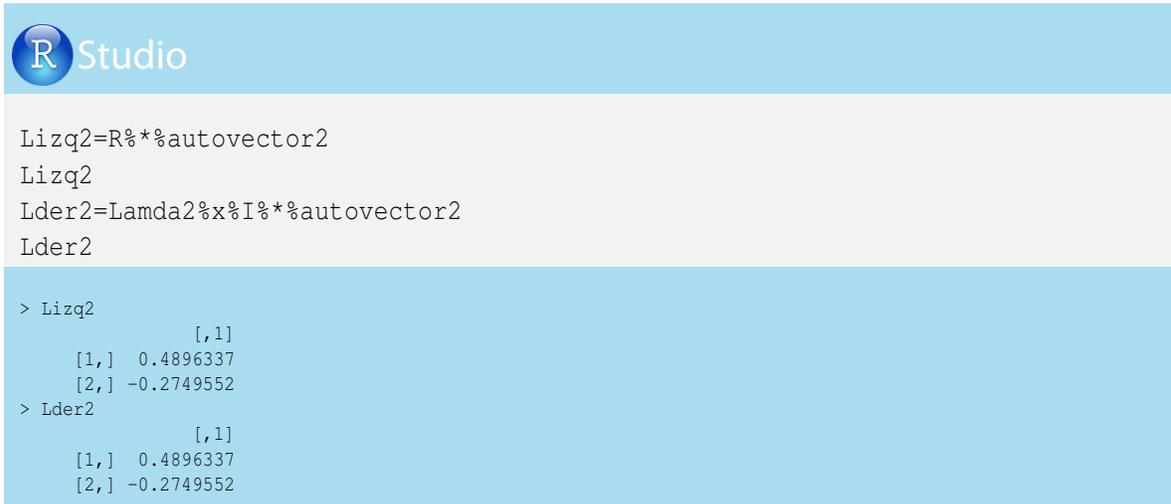
$$\lambda_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * v_2 = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * v_2$$

$$-0.561515 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.8719 \\ -0.4896 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} -0.8719 \\ -0.4896 \end{bmatrix}$$

$$\begin{bmatrix} -0.561515 & 0 \\ 0 & -0.561515 \end{bmatrix} * \begin{bmatrix} -0.8719 \\ -0.4896 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} -0.8719 \\ -0.4896 \end{bmatrix}$$

$$\begin{bmatrix} 0.4896 \\ -0.27496 \end{bmatrix} = \begin{bmatrix} 0.4896 \\ -0.27496 \end{bmatrix}$$

La programación en R-project para obtener el lado izquierdo y el lado derecho de la segunda igualdad es la siguiente:



```

R Studio

Lizq2=R*%autovector2
Lizq2
Lder2=Lamda2%x%I*%autovector2
Lder2

> Lizq2
      [,1]
[1,] 0.4896337
[2,] -0.2749552
> Lder2
      [,1]
[1,] 0.4896337
[2,] -0.2749552

```

### 1.10. Inversa generalizada de una matriz

Hay casos en que las matrices no tienen inversa porque su determinante es igual a cero. Veamos un ejemplo de este caso, considerando la matriz  $S_{mn}$ :

$$S_{mn} = \begin{bmatrix} 0 & 1 \\ 0 & 3 \end{bmatrix}$$

$$\text{Det}S_{mn} = |(0 * 3) - (0 * 1)| = 0$$

Por otro lado, al multiplicar  $S * S^{-1}$  no tendríamos una matriz identidad, por lo que resulta:

$$\begin{bmatrix} 0 & 1 \\ 0 & 3 \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (0 * a) + (1 * c) & (0 * b) + (1 * d) \\ (0 * a) + (3 * c) & (0 * b) + (3 * d) \end{bmatrix}$$

En el elemento  $i_{11}$

$$\begin{aligned} (0 * a) + (1 * c) &= 1 \\ 0 + c &= 1 \\ c &= 1 \end{aligned}$$

En el elemento  $i_{21}$

$$\begin{aligned} (0 * a) + (3 * c) &= 0 \\ 0 + (3 * 1) &= 0 \\ 0 + 3 &\neq 0 \end{aligned}$$

En el elemento  $i_{12}$

$$\begin{aligned} (0 * b) + (1 * d) &= 0 \\ 1d &= 0 \\ d &= 0 \end{aligned}$$

En el elemento  $i_{22}$

$$\begin{aligned} (0 * b) + (3 * d) &= 1 \\ (0 * b) + (3 * 0) &= 1 \\ 0 &\neq 1 \end{aligned}$$

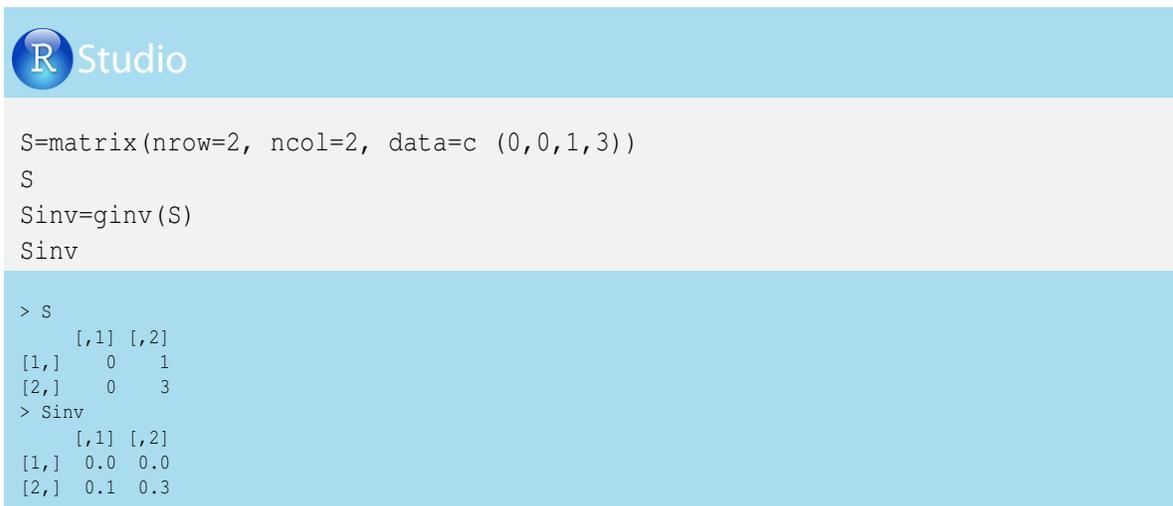
Por consiguiente:

$$S_{mn} * S_{mn}^{-1} \neq I_{mn}$$

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} * \begin{bmatrix} N & N \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

A pesar de lo presentado anteriormente, se puede obtener la inversa de  $S_{mn}$ , la cual se llama inversa generalizada, que se denota con  $S^{-}$ , lo que resuelve el problema de las matrices que presentan determinante igual a cero o que son matrices no cuadradas. Sin embargo, hay que tener en cuenta que una matriz puede tener infinitas matrices inversas.

Para encontrar la inversa generalizada utilizamos la librería *MASS* del R-project, con el comando *ginv*.



```

R Studio

S=matrix(nrow=2, ncol=2, data=c (0,0,1,3))
S
Sinv=ginv(S)
Sinv

> S
      [,1] [,2]
[1,]    0    1
[2,]    0    3
> Sinv
      [,1] [,2]
[1,]  0.0  0.0
[2,]  0.1  0.3
    
```

## 1.11. Algunas aplicaciones de matrices en la producción animal

Las matrices son comúnmente usadas en sistemas de producción animal y resultan de bastante utilidad para resolver situaciones como balanceo de raciones, costos mínimos, cálculos de consumo, cálculos de dimensión de estanques y potreros, evaluaciones genéticas, entre otros, empleando los sistemas de ecuaciones lineales. Veamos algunos ejemplos:

### 1.11.1. Pago por calidad de leche

Veamos el siguiente caso: En las colillas de pago que recibe un productor de leche, le reportaron que un litro de leche tenía 35 g de grasa y 30 g de proteína y le pagaron \$761.25, y en el siguiente mes la leche tenía 30 g de grasa y 28 g de proteína y le pagaron \$694.26. El

productor desea saber qué precio tienen un gramo de grasa y un gramo de proteína por litro de leche producido. Las ecuaciones para resolver la incógnita del productor son las siguientes:

$$35g + 30p = 761.25$$

$$30g + 28p = 694.26$$

La forma matricial del sistema de ecuaciones planteado es:

$$\begin{bmatrix} 35 & 30 \\ 30 & 28 \end{bmatrix} * \begin{bmatrix} g \\ p \end{bmatrix} = \begin{bmatrix} 761.25 \\ 694.26 \end{bmatrix}$$

Desarrollemos el ejercicio:

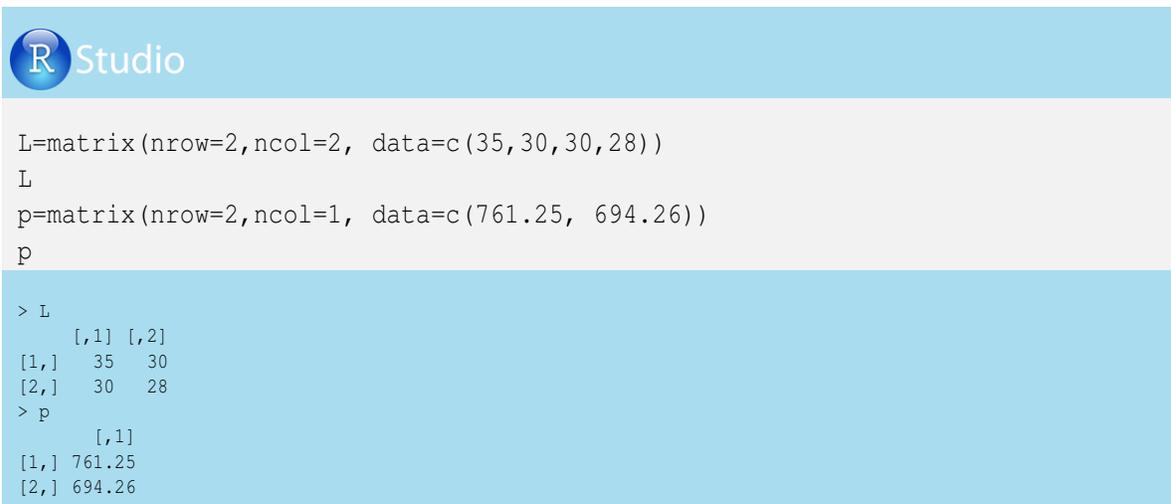
$$\begin{bmatrix} g \\ p \end{bmatrix} = \begin{bmatrix} 35 & 30 \\ 30 & 28 \end{bmatrix}^{-1} * \begin{bmatrix} 761.25 \\ 694.26 \end{bmatrix}$$

$$\begin{bmatrix} g \\ p \end{bmatrix} = \begin{bmatrix} 0.35 & -0.375 \\ -0.375 & 0.4375 \end{bmatrix} * \begin{bmatrix} 761.25 \\ 694.26 \end{bmatrix}$$

$$\begin{bmatrix} g \\ p \end{bmatrix} = \begin{bmatrix} 6.09 \\ 18.27 \end{bmatrix}$$

La empresa le está pagando al productor \$6.09 por gramo de grasa y \$18.27 por gramo de proteína.

Desarrollemos el ejercicio en R-project. Primero creamos la matriz  $L$ , que relaciona los meses con las cantidades de grasa y proteína (gramos) y un vector columna  $p$  que relaciona el pago por litro de leche durante dos meses.



```

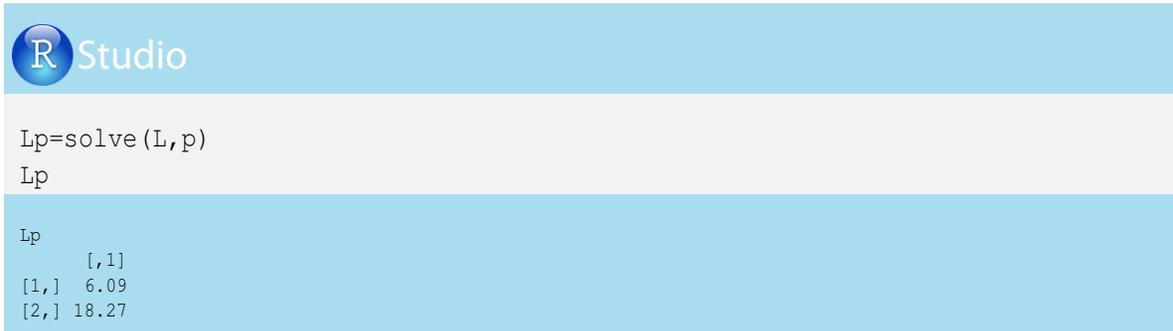
R Studio

L=matrix(nrow=2,ncol=2, data=c(35,30,30,28))
L
p=matrix(nrow=2,ncol=1, data=c(761.25, 694.26))
p

> L
      [,1] [,2]
[1,]  35   30
[2,]  30   28
> p
      [,1]
[1,] 761.25
[2,] 694.26

```

Ahora utilizaremos el comando *solve*, al cual se le determina entre paréntesis la matriz que se invertirá y la matriz o vector a multiplicársele:



```

R Studio

Lp=solve(L,p)
Lp

Lp
      [,1]
[1,]  6.09
[2,] 18.27
    
```

Al siguiente mes el productor recibe una bonificación por certificarse como hato libre de Brucella, pero no sabe cuál es el valor de dicha bonificación. Para calcularla, el sistema de ecuaciones sería:

$$35g + 30p + 0b = 761.25$$

$$30g + 28p + 0b = 694.26$$

$$33g + 30p + 1b = 759.07$$

La forma matricial del sistema de ecuaciones planteado es:

$$\begin{bmatrix} 35 & 30 & 0 \\ 30 & 28 & 0 \\ 33 & 30 & 1 \end{bmatrix} * \begin{bmatrix} g \\ p \\ b \end{bmatrix} = \begin{bmatrix} 761.25 \\ 694.26 \\ 759.07 \end{bmatrix}$$

Desarrollemos el ejercicio:

$$\begin{bmatrix} g \\ p \\ b \end{bmatrix} = \begin{bmatrix} 35 & 30 & 0 \\ 30 & 28 & 0 \\ 33 & 30 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} 761.25 \\ 694.26 \\ 759.07 \end{bmatrix}$$

$$\begin{bmatrix} g \\ p \\ b \end{bmatrix} = \begin{bmatrix} 0.35 & -0.375 & 0 \\ -0.375 & 0.4375 & 0 \\ -0.3 & -0.75 & 1 \end{bmatrix} * \begin{bmatrix} 761.25 \\ 694.26 \\ 759.07 \end{bmatrix}$$

$$\begin{bmatrix} g \\ p \\ b \end{bmatrix} = \begin{bmatrix} 6.09 \\ 18.27 \\ 10 \end{bmatrix}$$

La empresa le está pagando al productor una bonificación de \$10 por litro de leche producido, por ser un hato libre de Brucella.

Ahora desarrollemos el ejercicio en R-project:

```

R Studio

L1=matrix(nrow=3,ncol=3, data=c(35,30,33,30,28,30,0,0,1))
L1
p1=matrix(nrow=3,ncol=1, data=c(761.25, 694.26, 759.07))
p1
L1p1=solve(L1,p1)
L1p1

> L1
      [,1] [,2] [,3]
[1,]  35   30   0
[2,]  30   28   0
[3,]  33   30   1
> p1
      [,1]
[1,] 761.25
[2,] 694.26
[3,] 759.07
> L1p1
      [,1]
[1,]  6.09
[2,] 18.27
[3,] 10.00

```

En los siguientes dos meses, el productor recibe otra bonificación por hato libre de tuberculosis ( $t$ ), pero se le anuncia que lo están castigando por alto recuento de Unidades Formadoras de Colonia ( $u$ ).

Con las colillas de pago se generó el siguiente sistema de ecuaciones:

$$\begin{aligned}
 35g + 30p + 0b + 0t + 0u &= 761.25 \\
 30g + 28p + 0b + 0t + 0u &= 694.26 \\
 33g + 30p + 1b + 0t + 0u &= 759.07 \\
 35g + 30p + 1b + 1t + 1u &= 766.98 \\
 32g + 29p + 1b + 1t + 1u &= 729.71 \\
 30g + 28p + 1b + 1t + 0u &= 714.26
 \end{aligned}$$

El sistema de forma matricial sería:

$$\begin{bmatrix} 35 & 30 & 0 & 0 & 0 \\ 30 & 28 & 0 & 0 & 0 \\ 33 & 30 & 1 & 0 & 0 \\ 35 & 30 & 1 & 1 & 1 \\ 32 & 29 & 1 & 1 & 1 \\ 30 & 28 & 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} g \\ p \\ b \\ t \\ u \end{bmatrix} = \begin{bmatrix} 761.25 \\ 694.26 \\ 759.07 \\ 766.98 \\ 729.71 \\ 714.26 \end{bmatrix}$$

Obtengamos inicialmente la matriz  $L2$  y el vector  $p2$  en R-project:

```

R Studio

L2=matrix(nrow=6,ncol=5, data=c(35, 30, 33, 35, 32, 30, 30, 28, 30,
30, 29, 28, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,0, 0, 1, 1, 0))
L2
p2=matrix(nrow=6,ncol=1, data=c(761.25, 694.26, 759.07, 766.25,
729.71,714.26))
p2

> L2
      [,1] [,2] [,3] [,4] [,5]
[1,]  35  30   0   0   0
[2,]  30  28   0   0   0
[3,]  33  30   1   0   0
[4,]  35  30   1   1   1
[5,]  32  29   1   1   1
[6,]  30  28   1   1   0
> p2
      [,1]
[1,] 761.25
[2,] 694.26
[3,] 759.07
[4,] 766.25
[5,] 729.71
[6,] 714.26

```

Para resolver este caso, es necesario utilizar la inversa generalizada porque la matriz no es cuadrada. Utilizaremos la librería *limSolve* de Soetaert et al. (2009) y Van den Meersche et al. (2009) con el comando *lsei*.

```

R Studio

install.packages ("limSolve")
library (limSolve)
L2p2=lsei(L2,p2, fulloutput=TRUE)$X
L2p2=matrix(L2p2)
L2p2

L2p2
      [,1]
[1,]  6.09
[2,] 18.27
[3,] 10.00
[4,] 10.00
[5,] -15.00

```

La empresa le está pagando al productor una bonificación de \$10 por hato libre de tuberculosis y lo está castigando en \$15 por alto recuento de Unidades Formadoras de Colonia.

### 1.11.2. Formulación de raciones

Veamos un ejemplo de formulación de concentrado para vacas de producción de leche. Iniciamos realizando una mezcla que contenga un 15% de proteína y que incluya maíz (con 7.5% de proteína) y soya (con 36.8% de proteína). Para resolver este caso, se requiere el diseño de un sistema de ecuaciones que considere una ecuación que relacione la mezcla de maíz ( $m$ ) y de soya ( $s$ ) para producir un kilo de concentrado y otra ecuación que relacione los porcentajes de proteína de  $m$  y  $s$  para generar un 15% de proteína en la ración. Veamos:

$$\begin{aligned} m + s &= 1 \\ 7.5m + 36.8s &= 15 \end{aligned}$$

La forma matricial de este sistema de ecuaciones es:

$$\begin{bmatrix} 1 & 1 \\ 7.5 & 36.8 \end{bmatrix} * \begin{bmatrix} m \\ s \end{bmatrix} = \begin{bmatrix} 1 \\ 15 \end{bmatrix}$$

Desarrollemos el ejercicio:

$$\begin{bmatrix} m \\ s \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 7.5 & 36.8 \end{bmatrix}^{-1} * \begin{bmatrix} 1 \\ 15 \end{bmatrix}$$

$$\begin{bmatrix} m \\ s \end{bmatrix} = \begin{bmatrix} 1.26 & -0.034 \\ -0.26 & 0.034 \end{bmatrix} * \begin{bmatrix} 1 \\ 15 \end{bmatrix}$$

$$\begin{bmatrix} m \\ s \end{bmatrix} = \begin{bmatrix} 0.744 \\ 0.256 \end{bmatrix}$$

Se requieren 744 g de maíz y 256 g de soya para obtener un kilo de concentrado con 15% de proteína. En otras palabras, se requiere 74.4% de maíz y 25.6% de soya para producir una ración con el 15% de proteína.

Ahora en R-project creamos la matriz  $U$  con las cantidades de maíz y soya y sus respectivos porcentajes de proteína, y el vector  $v$  relacionado con las cantidades de proteína de la mezcla.

```


U=matrix(nrow=2,ncol=2, data=c(1,7.5,1,36.8))
U
v=matrix(nrow=2,ncol=1, data=c(1,15))
v

> U
  [,1] [,2]
[1,]  1.0  1.0
[2,]  7.5 36.8
> v
  [,1]
[1,]  1
[2,] 15

```

Para resolver el sistema de ecuaciones en R-project, podemos utilizar el comando *solve* o el comando *lsei*. Aquí utilizaremos el comando *lsei*:

```

R Studio

Uv=lsei(U,v, fulloutput=TRUE)$X
Uv=matrix(Uv)
Uv

> Uv
      [,1]
[1,] 0.7440273
[2,] 0.2559727
    
```

Supongamos que se desea una nueva formulación de concentrado que tenga en cuenta un aporte energético de 1750 kcal/kg. El maíz utilizado tiene una energía de 1800 kcal/kg, y la soya de 2240 kcal/kg. Adicionalmente, tenemos la posibilidad de incluirle cebada (*c*), que tiene un 11.3 % de proteína y 1655 kcal/kg de energía.

$$\begin{aligned}
 m + s + c &= 1 \\
 7.5m + 36.8s + 11.3c &= 15 \\
 1800m + 2240s + 1655c &= 1750
 \end{aligned}$$

La forma matricial de este sistema de ecuaciones es:

$$\begin{bmatrix} 1 & 1 & 1 \\ 7.5 & 36.8 & 11.3 \\ 1800 & 2240 & 1655 \end{bmatrix} * \begin{bmatrix} m \\ s \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 15 \\ 1750 \end{bmatrix}$$

Desarrollemos el ejercicio:

$$\begin{bmatrix} m \\ s \\ c \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 7.5 & 36.8 & 11.3 \\ 1800 & 2240 & 1655 \end{bmatrix}^{-1} * \begin{bmatrix} 1 \\ 15 \\ 1750 \end{bmatrix}$$

Los resultados encontrados por medio del paquete *limSolve* serían:

$$\begin{bmatrix} m \\ s \\ c \end{bmatrix} = \begin{bmatrix} 0.043 \\ 0.152 \\ 0.805 \end{bmatrix}$$

Según los resultados de este ejemplo, un kilo de concentrado tendría 43 g de maíz, 152 g de soya y 805 g de cebada. En otras palabras, en la fabricación del concentrado se requiere 4.3 % de maíz, 15.2 % de soya y 80.5 % de cebada.

Vemos la programación en R-project. Primero creamos la matriz  $U1$  y el vector  $v1$ :

```

R Studio

U1=matrix(nrow=3,ncol=3, data=c(1,7.5,1800,1,36.8,2240,
1,11.3,1655))
U1
v1=matrix(nrow=3,ncol=1, data=c(1,15,1750))
v1

> U1
      [,1] [,2] [,3]
[1,]  1.0  1.0  1.0
[2,]  7.5 36.8 11.3
[3,] 1800.0 2240.0 1655.0
> v1
      [,1]
[1,]    1
[2,]   15
[3,]  1750

```

Ahora utilizaremos el comando *lsei* para generar la solución al sistema de ecuaciones:

```

R Studio

U1v1=lsei(U1,v1, fulloutput=TRUE)$X
U1v1=matrix(U1v1)
U1v1

      [,1]
[1,] 0.04357777
[2,] 0.15159201
[3,] 0.80483016

```

### 1.11.3. Formulación de raciones con restricción de nutrientes

Cuando se realiza una ración, es necesario limitar algunos nutrientes. En este caso, el balanceamiento de raciones estará dado por un sistema de ecuaciones e inecuaciones (ecuaciones con desigualdades, mayor o menor que).

Continuemos con el ejemplo anterior y decidamos que los límites máximos de la ración serían de 40% de maíz, 15% de soya y 75% de cebada.

Al sistema anterior se le incluye el siguiente sistema de inecuaciones:

$$\begin{aligned}
 m &\leq 40 \\
 s &\leq 15 \\
 c &\leq 75
 \end{aligned}$$

Para montar las inecuaciones tenemos:

$$\begin{aligned} m + 0 + 0 &\leq 40 \\ 0 + s + 0 &\leq 15 \\ 0 + 0 + c &\leq 75 \end{aligned}$$

Lo que es igual a:

$$\begin{aligned} -m + 0 + 0 &\geq -40 \\ 0 - s + 0 &\geq -15 \\ 0 + 0 - c &\geq -75 \end{aligned}$$

La forma matricial de este sistema de inecuaciones es:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} m \\ s \\ c \end{bmatrix} \geq \begin{bmatrix} -0.40 \\ -0.15 \\ -0.75 \end{bmatrix}$$

Para resolver el sistema anterior de ecuaciones con la inclusión de inecuaciones, utilizamos la librería *limSolve* en R-project. Los resultados obtenidos se presentan a continuación:

$$\begin{bmatrix} m \\ s \\ c \end{bmatrix} = \begin{bmatrix} 0.096 \\ 0.150 \\ 0.750 \end{bmatrix}$$

Respetando las restricciones impuestas al sistema de ecuaciones, la ración estaría compuesta por 9.6% de maíz, 15% de soya y 75% de cebada.

Veamos la programación en R-project, donde creamos una matriz *U2* que relaciona las restricciones de maíz, soya y cebada, y un vector *v2* que relaciona las cantidades máximas de maíz, soya y cebada.

```


R Studio

U2=matrix(nrow=3,ncol=3, data=c(-1, 0, 0, 0, -1, 0, 0, 0, -1))
U2
v2=matrix(nrow=3,ncol=1, data=c(-0.4,-.15,-.75))
v2

      [,1] [,2] [,3]
[1,]  -1   0   0
[2,]   0  -1   0
[3,]   0   0  -1
> v2
      [,1]
[1,] -0.40
[2,] -0.15
[3,] -0.75

```

El comando *lsei* funcionaría incluyendo cuatro matrices: dos relacionadas con el sistema de ecuaciones (en este caso  $U1$  y  $v1$ ) y dos relacionadas con el sistema de inecuaciones  $U2$  y  $v2$ . Veamos el desarrollo de este ejercicio en R-project:

```

R Studio

U1v1U2v2=lsei(U1,v1,G=U2,H=v2)$X
U1v1U2v2=matrix(U1v1U2v2)
U1v1U2v2

      [,1]
[1,] 0.09597288
[2,] 0.15000000
[3,] 0.75000000
    
```

Ahora, comprobemos que las cantidades de materia prima (maíz, soya y cebada) obtenidas anteriormente no superan los límites máximos y generan un kilo de concentrado con 15% de proteína y 1750 kcal/kg (se tendrán valores aproximados):

```

R Studio

comprobacion=U1%*%U1v1U2v2
comprobacion

      [,1]
[1,] 0.9959729
[2,] 14.7147966
[3,] 1750.0011906
    
```

Es de anotar que el uso de la inversa generalizada permite obtener diversas soluciones, las cuales cumplen con las condiciones establecidas en el sistema de ecuaciones.

#### 1.11.4. Precio del huevo

Retomando el ejemplo del precio promedio de los huevos por semana, consideremos la matriz  $G$ , relacionada con los huevos buenos por granja (fila) y por semana (columna), y un vector columna  $m$  del ingreso bruto por venta de huevos por semana.

$$G_{mn} * h_{kl} = m_{ml}$$

$$\begin{bmatrix} 890 & 873 & 870 \\ 764 & 681 & 700 \\ 806 & 798 & 819 \end{bmatrix} * \begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 794302 \\ 647052 \\ 730885 \end{bmatrix}$$

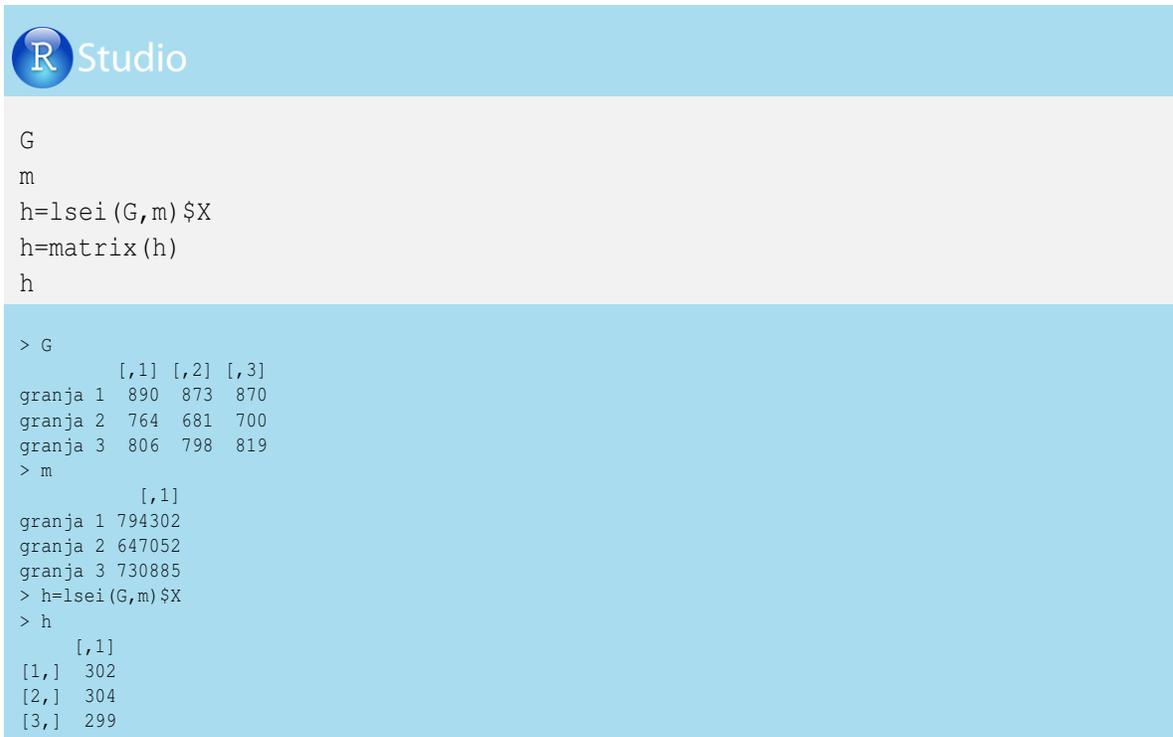
Para resolver el anterior sistema de ecuaciones con las incógnitas ( $h$ ) se requiere que la matriz  $G$  pase al otro lado como inversa, así:

$$h_{kl} = G_{mn}^{-1} * m_{ml}$$

$$\begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 0.0005 & 0.013 & -0.0117 \\ 0.0388 & -0.017 & -0.026 \\ -0.038 & 0.004 & 0.038 \end{bmatrix} \begin{bmatrix} 794302 \\ 647052 \\ 730885 \end{bmatrix}$$

$$\begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 302 \\ 304 \\ 299 \end{bmatrix}$$

La programación en R-project para resolver el ejercicio es:



```

R Studio

G
m
h=lsei(G,m)$X
h=matrix(h)
h

> G
      [,1] [,2] [,3]
granja 1 890 873 870
granja 2 764 681 700
granja 3 806 798 819
> m
      [,1]
granja 1 794302
granja 2 647052
granja 3 730885
> h=lsei(G,m)$X
> h
      [,1]
[1,] 302
[2,] 304
[3,] 299
    
```

Veamos otro ejemplo del cálculo del precio promedio de los huevos por granja. Para esto utilizamos la transpuesta de la matriz  $G$ , que relaciona el número de huevos buenos por semana (filas) y por granja (columnas) y el vector columna  $n$  que está relacionado con el ingreso bruto por venta de huevos por granja. El sistema de ecuaciones sería:

$$G'_{mn} * h_{kl} = n_{ml}$$

$$\begin{bmatrix} 890 & 764 & 806 \\ 873 & 681 & 798 \\ 870 & 700 & 819 \end{bmatrix} * \begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 742030 \\ 709272 \\ 720421 \end{bmatrix}$$

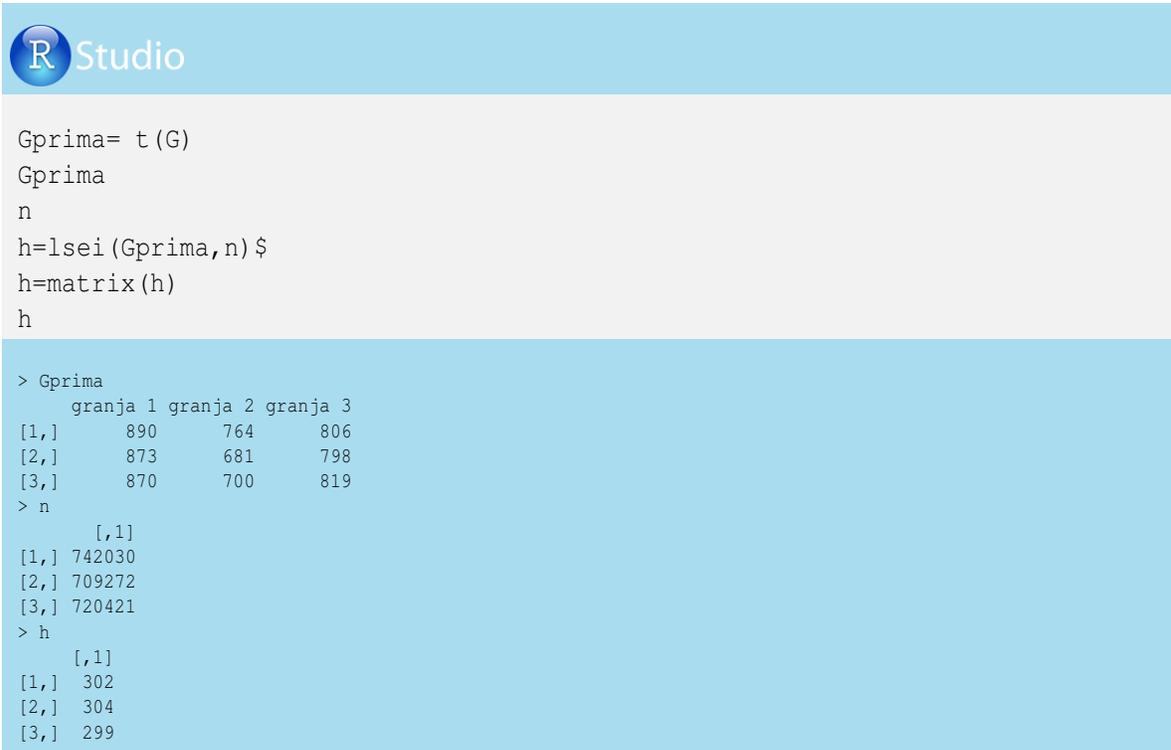
El anterior sistema de ecuaciones con las incógnitas ( $h$ ) requiere que la matriz  $G$  pase al otro lado como inversa, de modo que queda así:

$$h_{kl} = G'^{-1}_{mn} * n_{ml}$$

$$\begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 0.0005 & 0.038 & -0.038 \\ 0.0130 & -0.0174 & -0.004 \\ -0.01174 & -0.026 & 0.038 \end{bmatrix} \begin{bmatrix} 742030 \\ 709272 \\ 720421 \end{bmatrix}$$

$$\begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} 302 \\ 304 \\ 299 \end{bmatrix}$$

Veamos la programación en R-project:



```

R Studio

Gprima= t(G)
Gprima
n
h=lsei(Gprima,n)$
h=matrix(h)
h

> Gprima
      granja 1 granja 2 granja 3
[1,]      890      764      806
[2,]      873      681      798
[3,]      870      700      819
> n
      [,1]
[1,] 742030
[2,] 709272
[3,] 720421
> h
      [,1]
[1,] 302
[2,] 304
[3,] 299

```

## Capítulo 2

### Números generados aleatoriamente

En R-project se puede obtener un grupo de números que obedecen a una distribución específica, mediante un proceso de generación al azar (todo número tiene la misma probabilidad de ser elegido). Estos números se denominan aleatorios, pero por el hecho de ser obtenidos por procesos computacionales (algoritmos) que pueden generar los mismos valores si existen las mismas condiciones de arranque (semillas), son llamados pseudoaleatorios.

La generación de números al azar ayuda a realizar procesos de simulación. En sistemas de producción podemos simular los cambios poblacionales de un hato o granja en el tiempo, los cambios en la producción cuando se modifican los aspectos nutricionales o de manejo, los ingresos percibidos en un sistema de producción en el tiempo, etc.

Toda variable aleatoria de tipo discreto (elementos que siguen una numeración en secuencia) o de tipo continuo (valores dentro de un intervalo de números reales) obedece a un tipo de distribución, por ejemplo:

- ♣ La producción de leche de  $n$  vacas o el peso al sacrificio de  $n$  novillos, por ejemplo, pueden ser modelados por medio de una distribución normal.
- ♣ La presencia de mastitis en un hato o la mortalidad al nacimiento de los animales obedecen a ensayos de Bernoulli, y varios ensayos de Bernoulli conllevan a una distribución binomial.
- ♣ El tamaño de camada, la mortalidad de aves por jaula en un galpón o la mortalidad en el tiempo pueden obedecer a una distribución de Poisson.

En este capítulo estudiaremos las principales distribuciones que se utilizan en producción y salud animal. Las programaciones en R-project incluyen el comando *set.seed* (valor de semilla) para generar la misma secuencia de números pseudoaleatorios, para que el lector pueda obtener los mismos resultados de este libro. También se realizarán gráficos sencillos, los cuales pueden ser mejorados por el lector con las explicaciones del capítulo 5.

## 2.1. Números pseudoaleatorios con distribución uniforme discreta

Sea  $x$  una variable aleatoria discreta con distribución uniforme, la cual asume  $x$  valores enteros consecutivos, con igual probabilidad de ocurrir. Un ejemplo clásico de este tipo de distribución es el lanzamiento de un dado, donde los valores 1, 2, 3, 4, 5 y 6 tienen la misma probabilidad de salir.

La distribución de probabilidad o función de masa para este tipo de variable aleatoria discreta está dada por:

$$f(x;k) = \frac{1}{k}$$

Donde  $k$  representa los valores que puede asumir  $x$ . Esta función nos permite estimar la probabilidad de obtener un valor en un rango determinado.

Tomemos como ejemplo el sexo de 100 embriones, los cuales pueden ser machos o hembras. Esta variable discreta puede seguir una distribución uniforme discreta con dos posibles valores.

Para esta variable aleatoria, la probabilidad de obtener un valor en el rango determinado es  $f(1;2) = 1/2 = 0.5$ , donde  $k = 2$  es el número de valores que puede asumir  $x$ .

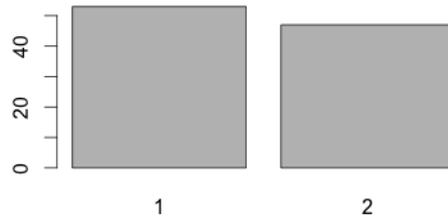
La programación en R-project para simular el género de 100 embriones, de los cuales se espera un 50% machos ( $sexo = 1$ ) y 50% hembras ( $sexo = 2$ ), está dada por el comando `sample`, seguido de un paréntesis con el valor inicial y final que puede asumir la variable y el número de repeticiones. También es necesario especificar si existe un muestreo con remplazo, con el comando `replace = TRUE`. Veamos la programación utilizando como semilla el número 10:

```
R Studio
set.seed (10)
SexoEmbrion=sample(1:2,100,replace=TRUE)
SexoEmbrion

[1] 2 1 1 2 1 1 1 1 2 1 2 2 1 2 1 1 1 1 1 2 2 2 1 1 2 2 1 1 2 2 1 2 1 2 1 1 2 1 2
[37] 2 2 2 2 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 2 1 1 1 2 2 2 1 1 1 1 2
[73] 2 2 1 2 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 2 2 2 1 2 1 2 2 1
```

Generemos un gráfico de barras con las frecuencias de los sexos, utilizando el comando `barplot` e indicando que el gráfico es el producto de una tabla de frecuencias (`table`):

```
R Studio
barplot (table (SexoEmbrion))
```



## 2.2. Números pseudoaleatorios con distribución binomial

Antes de generar números aleatorios, iniciamos estudiando la distribución binomial, que es una distribución de probabilidad discreta, dada por el número de sucesos en una secuencia de  $n$  tentativas independientes, cuyo resultado puede ser un éxito (con probabilidad  $p$ ) o un fracaso (con probabilidad  $1 - p$ ). Veamos un ejemplo relacionado con la tuberculosis en bovinos, causada por la bacteria *Mycobacterium bovis*: en un estudio previo de determinada región se encontró que la probabilidad de hallar bovinos libres de tuberculosis es del 70% ( $p = 0.7$  y  $q = 0.3$ ).

Si pretendemos realizar la prueba de tuberculina en 5 vacas escogidas aleatoriamente del hato, la probabilidad de tener  $x = 0, x = 1, x = 2, x = 3, x = 4$  y  $x = 5$  vacas libres de la bacteria estaría dada por las siguientes situaciones:

- ♣ No se encuentran vacas libres de la bacteria ( $x = 0$ ). En este caso se tendrían 5 vacas positivas a la prueba de tuberculina ( $\prod(q) = 0.3 * 0.3 * 0.3 * 0.3 * 0.3 = 0.00243$ ) o ( $q^5 = 0.3^5 = 0.00243$ ).

Veamos el siguiente esquema:

Identificación de las vacas					$\prod(p) * \prod(q)$	Veces	$\sum(\prod(p) * \prod(q))$
1	2	3	4	5			
Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	0.00243	1	0.00243

Dicho de otra forma:

$$f_x = \binom{n}{x} * p^x * q^{n-x}$$

$$f_{x=0} = \binom{5}{0} * 0.7^0 * 0.3^{5-0}$$

$$f_{x=0} = \frac{5!}{(5-0)! * 0!} * 1 * 0.00243$$

$$f_{x=0} = \frac{5 * 4 * 3 * 2}{5 * 4 * 3 * 2} * 1 * 0.00243$$

$$f_{x=0} = 0.00243$$

♣ Si se presenta una vaca libre de la bacteria ( $x = 1$ ):

Si la primera vaca está libre de la bacteria:

$$\begin{aligned} p &= 0.7 \\ \prod(p) &= 0.7 \\ p^1 &= 0.7^1 \end{aligned}$$

Si las otras vacas son positivas:

$$\begin{aligned} \prod(q) &= 0.3 * 0.3 * 0.3 * 0.3 = 0.0081 \\ p^4 &= 0.3^4 = 0.0081 \end{aligned}$$

El producto sería:

$$\prod(p) * \prod(q) = 0.00567$$

También se podría tener que la primera vaca sea positiva, la segunda vaca esté libre y las otras positivas, y así sucesivamente, como se indica a continuación:

Identificación de las vacas					$\prod(p) * \prod(q)$	Veces	$\sum(\prod(p) * \prod(q))$
1	2	3	4	5			
Neg (p=0.7)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	0.00567	5	0.02835
Tub (p=0.3)	Neg (p=0.7)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	0.00567		
Tub (p=0.3)	Tub (p=0.3)	Neg (p=0.7)	Tub (p=0.3)	Tub (p=0.3)	0.00567		
Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Neg (p=0.7)	Tub (p=0.3)	0.00567		
Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Neg (p=0.7)	0.00567		

Dicho de otra forma:

$$f(1) = \binom{5}{1} * 0.7^1 * 0.3^{5-1}$$

$$f(1) = \frac{5!}{(5-1)! * 1!} * 0.7 * 0.0081$$

$$f(1) = \frac{5 * 4 * 3 * 2}{4 * 3 * 2} * 0.00567$$

$$f(1) = 5 * 0.00567 = 0.02835$$

♣ En el caso de dos vacas libres, se tendría:

Identificación de las vacas					$\Pi(p) * \Pi(q)$	Veces	$\Sigma(\Pi(p) * \Pi(q))$
1	2	3	4	5			
Neg (p=0.7)	Neg (p=0.7)	Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	0.01323	10	0.1323
Neg (p=0.7)	Tub (p=0.3)	Neg (p=0.7)	Tub (p=0.3)	Tub (p=0.3)	0.01323		
.	.	.	.	.	..		
Tub (p=0.3)	Tub (p=0.3)	Tub (p=0.3)	Neg (p=0.7)	Neg (p=0.7)	0.01323		

De otra forma:

$$f(2) = \binom{5}{2} * 0.7^2 * 0.3^3$$

$$f(2) = \frac{5!}{(5-2)! * 2!} * 0.49 * 0.027$$

$$f(2) = \frac{5*4*3*2}{3*2*2} * 0.01323$$

$$f(2) = 10 * 0.01323 = 0.1323$$

♣ Con tres vacas libres, se tendría:

Identificación de las vacas					$\Pi(p) * \Pi(q)$	Veces	$\Sigma(\Pi(p) * \Pi(q))$
1	2	3	4	5			
Tub (q=0.3)	Tub (q=0.3)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	0.03087	10	0.3087
Neg (p=0.7)	Tub (q=0.3)	Tub (q=0.3)	Neg (p=0.7)	Neg (p=0.7)	0.03087		
.	.	.	.	.	..		
Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Tub (q=0.3)	Tub (q=0.3)	0.03087		

♣ Si se presentan cuatro vacas libres:

Identificación de las vacas					$\Pi(p) * \Pi(q)$	Veces	$\Sigma(\Pi(p) * \Pi(q))$
1	2	3	4	5			
Tub (q=0.3)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	0.07203	5	0.36015
Neg (p=0.7)	Tub (q=0.3)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	0.07203		
Neg (p=0.7)	Neg (p=0.7)	Tub (q=0.3)	Neg (p=0.7)	Neg (p=0.7)	0.07203		
Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Tub (q=0.3)	Neg (p=0.7)	0.07203		
Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Tub (q=0.3)	0.07203		

♣ Si todas las vacas están libres de la bacteria:

En este caso  $x = 5$ , donde se tendrían 5 vacas negativas a la prueba de tuberculina ( $\prod(p) = 0.7 * 0.7 * 0.7 * 0.7 * 0.7 = 0.16807$ ) o ( $p^5 = 0.7^5 = 0.16807$ ). Veamos el siguiente esquema:

Identificación de las vacas					$\prod(p)$
1	2	3	4	5	
Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	Neg (p=0.7)	0.16807

Dicho de otra forma:

$$f_x = \binom{n}{x} * q^x * p^{x-n}$$

$$f_{x=0} = \binom{5}{5} * 0.7^5 * 0.7^{5-5}$$

$$f_{x=0} = \frac{5!}{(5-5)! * 5!} * 0.16807 * 1$$

$$f_{x=0} = \frac{5 * 4 * 3 * 2}{1 * 5 * 4 * 3 * 2} * 0.16807$$

$$f_{x=0} = 0.16807$$

En general, la probabilidad de encontrar 0, 1, 2, 3, 4 y 5 vacas libres de tuberculosis es:

$$f_{(x=0)} = 0.00243$$

$$f_{(x=1)} = 0.02835$$

$$f_{(x=2)} = 0.1323$$

$$f_{(x=3)} = 0.3087$$

$$f_{(x=4)} = 0.36015$$

$$f_{(x=5)} = 0.16807$$

Para estimar las probabilidades puntuales de una distribución binomial utilizamos el comando *dbinom*, seguido de un paréntesis donde se define el número de casos posibles de vacas libres de tuberculosis ( 0, 1, 2, 3, 4 y 5 o  $x = 0 : 5$ ), con un tamaño muestral de 5 (*size=5*) y con una probabilidad de estar libre de la bacteria de *prob = 0.7*.

```

R Studio

ProbBin=dbinom(x=0:5,size=5,prob=0.70)
ProbBin

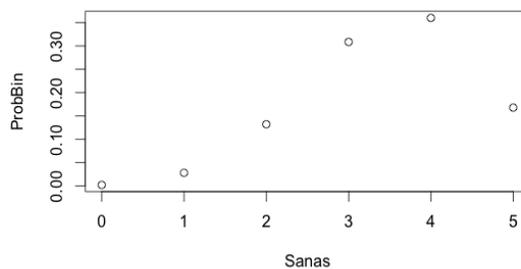
[1] 0.00243 0.02835 0.13230 0.30870 0.36015 0.16807
    
```

Modifiquemos el argumento  $x = 0 : 5$  por un vector creado al que llamaremos *Sanas*:

```
R Studio  
  
Sanas=c(0,1,2,3,4,5)  
Sanas  
ProbBin=dbinom(x=Sanas,size=5,prob=0.70)  
ProbBin  
  
> Sanas  
[1] 0 1 2 3 4 5  
> ProbBin  
[1] 0.00243 0.02835 0.13230 0.30870 0.36015 0.16807
```

Generemos un gráfico con el número de vacas sanas y sus respectivas probabilidades puntuales:

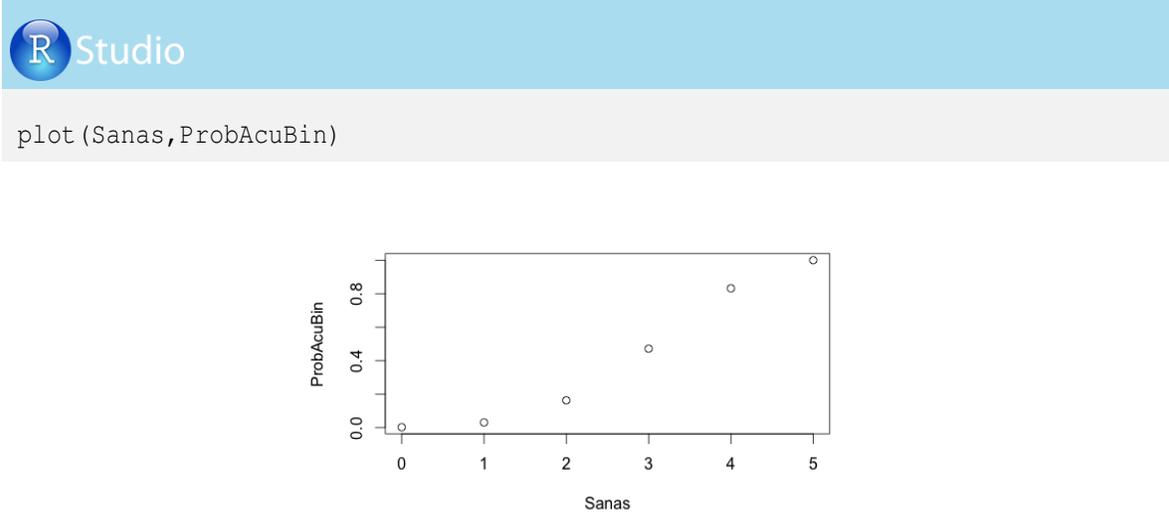
```
R Studio  
  
plot(Sanas,ProbBin)
```



Para conocer la distribución de probabilidad acumulada de una variable aleatoria binomial, se utiliza el comando *pbinom*:

```
R Studio  
  
ProbAcuBin=pbinom(q=0:5,size=5,prob=0.70)  
ProbAcuBin  
  
[1] 0.00243 0.03078 0.16308 0.47178 0.83193 1.00000
```

Generemos el gráfico de distribución de probabilidad acumulada:



Ahora generemos un número pseudoaleatorio con una distribución binomial, donde se tienen cinco tentativas ( $size=5$ ) o cinco vacas que pueden estar libres de tuberculosis, con una probabilidad de 0.7. Recordemos que utilizaremos una semilla para que el lector pueda obtener estos resultados en sus ejercicios:

```
R Studio
set.seed(3)
AleaBin=rbinom(1,size=5,prob=0.7)
AleaBin
[1] 4
```

En el caso anterior, la observación generada indica que en cinco tentativas un animal presentó reacción a la prueba de tuberculina (4 libres de la enfermedad). Generemos ahora otra observación, cambiando el valor de la semilla.

```
R Studio
set.seed(4)
AleaBin=rbinom(1,size=5,prob=0.7)
AleaBin
[1] 5
```

En el caso anterior, la observación generada indica que, en cinco tentativas, cinco vacas estuvieron libres de tuberculosis. Generemos ahora tres observaciones (3):

```
R Studio  
set.seed(4)  
AleaBin=rbinom(3, size=5, prob=0.7)  
AleaBin  
[1] 3 5 4
```

En la generación de las tres observaciones se encontró que en la primera observación se hallaron tres vacas sanas, en la segunda observación todas las vacas estuvieron sanas y en la tercera observación se encontraron cuatro vacas sanas.

Generemos ahora 10 observaciones, pero con 100 tentativas (100 vacas):

```
R Studio  
set.seed(18)  
AleaBin=rbinom(10, size=100, prob=0.7)  
AleaBin  
[1] 78 73 76 76 70 71 70 77 73 68
```

En esta caso, con diez observaciones se encontró que el número de vacas que estaban libres de la bacteria varió entre 68 y 78.

### 2.3. Números pseudoaleatorios con distribución de Poisson

La distribución de probabilidad discreta de Poisson parte de la distribución binomial con muestras grandes y con probabilidades de éxito bajas ( $p < 0.10$ ). Permite determinar la probabilidad de ocurrencia de un número determinado de eventos ( $k$ ) en un intervalo regular de tiempo, longitud, superficie, etc; y se caracteriza porque los eventos son independientes.

La función de masa de probabilidad de una variable con distribución de Poisson es:

$$f(k; \lambda) = \frac{e^{-\lambda} \lambda^k}{k!}$$

Donde  $k$  es el número entero positivo relacionado con los eventos ocurridos, y  $\lambda$  es el número esperado de eventos en un intervalo determinado y es constante en cada intervalo.

Las siguientes variables pueden presentar distribución de Poisson: número de lechones por camada y cerda, número de personas que ingresan a un consultorio o almacén agropecuario en un tiempo determinado, número de animales que ingresan a un bebedero en una hora determinada, número de células somáticas por placa leída, recuento de células en una placa histológica, entre otras.

Veamos dos ejemplos: el primero relacionado con el periodo de rumia de vacunos, dado por el “número de masticaciones por bolo” observado en 30 vacas, y otro relacionado con el “número de lechones por parto” que tiene una cerda en una granja de 25 matrices.

♣ Ejemplo de número de masticaciones por bolo:

La media de masticaciones es de 50 por bolo ( $\lambda = 50$ ). Si aplicamos la función de masa de probabilidad de una distribución de Poisson, podemos estimar el número de masticaciones por bolo con diferentes valores de  $k$  (40, 55, 60, 65 y 70), de la siguiente manera:

$$f(40; 50) = \frac{e^{-50} 50^{40}}{40!} = 0.021$$

$$f(45; 50) = \frac{e^{-50} 50^{45}}{45!} = 0.046$$

$$f(50; 50) = \frac{e^{-50} 50^{50}}{50!} = 0.056$$

$$f(55; 50) = \frac{e^{-50} 50^{55}}{55!} = 0.042$$

$$f(60; 50) = \frac{e^{-50} 50^{60}}{60!} = 0.020$$

$$f(65; 50) = \frac{e^{-50} 50^{65}}{65!} = 0.006$$

La probabilidad de encontrar una hembra con 40 masticaciones es de 0.02, con 50 es de 0.056 y con 65 de 0.006.

En R-project generamos un vector relacionado con el número de masticaciones por bolo (de 30 a 75) y posteriormente utilizamos el comando `dpois` para encontrar las probabilidades de escoger una vaca que realice entre 30 y 75 masticaciones, donde el promedio de masticaciones es de  $\lambda = 50$ , teniendo en cuenta la distribución de Poisson.

```
R Studio

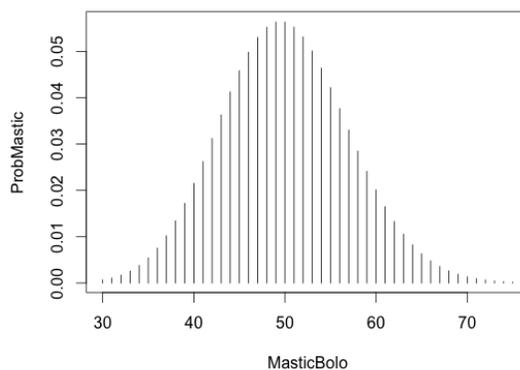
MasticBolo=c(30:75)
MasticBolo
set.seed(11)
ProbMastic=dpois(MasticBolo,lambda=50)
Masticaciones=cbind(MasticBolo,ProbMastic)
Masticaciones

> MasticBolo
 [1] 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
[25] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
> Masticaciones
      MasticBolo  ProbMastic
[1,]          30 0.0006771985
[2,]          31 0.0010922556
.
.
[19,]         48 0.0551985062
[20,]         49 0.0563250063
.
.
[45,]         74 0.0003086781
[46,]         75 0.0002057854
```

Gráficamente tendríamos:

```
R Studio

plot(MasticBolo,ProbMastic,type="h")
```



Con el comando `rpois` podemos generar 30 números pseudoaleatorios relacionados con la cantidad de masticaciones por bolo de 30 vacas, con un promedio de  $\lambda = 50$  masticaciones/bolo, así:

```

R Studio

set.seed (9)
Mastican30Vacas=rpois (n=30,lambda=50)
Mastican30Vacas

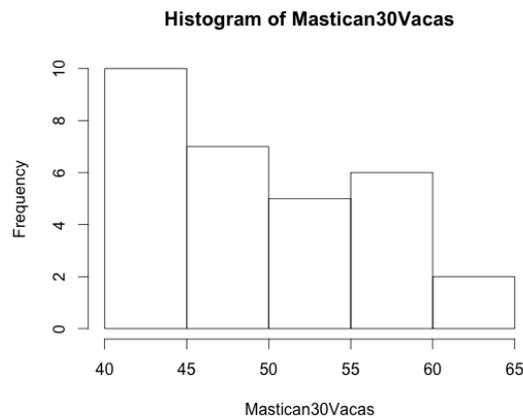
[1] 44 44 48 53 41 46 48 47 53 63 44 59 58 61 54 54 43 56 45 50 58 59 54 42
[25] 47 45 50 42 56 41
    
```

Generemos el gráfico de la distribución de probabilidad de los números pseudoaleatorios simulados:

```

R Studio

hist (Mastican30Vacas)
    
```



♣ Ejemplo del número de lechones nacidos por cerda en un parto:

La media de lechones por cerda y parto es de 11 ( $\lambda = 11$ ). Si aplicamos la función de masa de probabilidad de una distribución de Poisson, podemos estimar el número de lechones por

camada con valores de  $k$  de 8, 9, 10, 11, 12 y 13 lechones, así:

$$f(8;11) = \frac{e^{-11}11^8}{8!} = 0.089$$

$$f(9;11) = \frac{e^{-11}11^9}{9!} = 0.109$$

$$f(10;11) = \frac{e^{-11}11^{10}}{10!} = 0.119$$

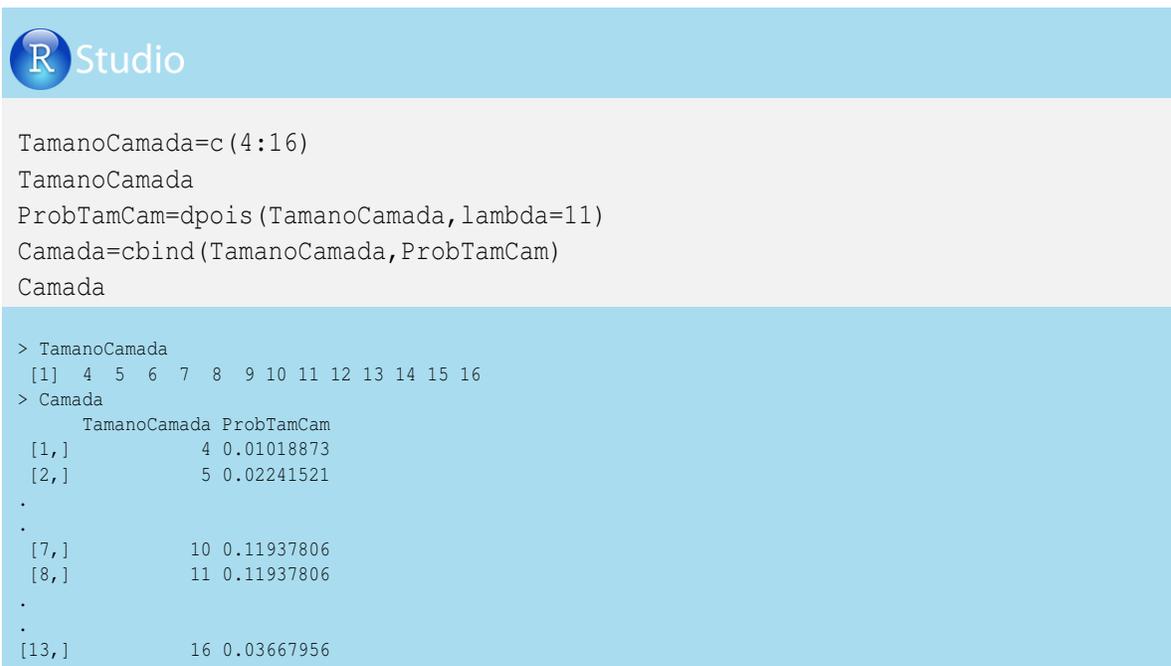
$$f(11;11) = \frac{e^{-11}11^{11}}{11!} = 0.119$$

$$f(12;11) = \frac{e^{-11}11^{12}}{12!} = 0.109$$

$$f(13;11) = \frac{e^{-11}11^{13}}{13!} = 0.092$$

La probabilidad de encontrar una cerda con 8 lechones es de 0.089, con 11 lechones de 0.119 y con 13 lechones de 0.092.

En R-project generamos un vector relacionado con el tamaño de camada por parto y cerda (de 4 a 16 lechones) y utilizamos el comando *dpois* para calcular las probabilidades de escoger una cerda que tenga una camada de 4 a 16 lechones, donde el promedio es de  $\lambda = 11$ :



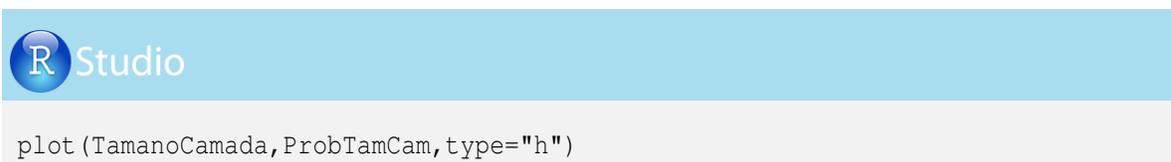
```

R Studio

TamanoCamada=c(4:16)
TamanoCamada
ProbTamCam=dpois(TamanoCamada,lambda=11)
Camada=cbind(TamanoCamada,ProbTamCam)
Camada

> TamanoCamada
[1] 4 5 6 7 8 9 10 11 12 13 14 15 16
> Camada
      TamanoCamada ProbTamCam
[1,]           4 0.01018873
[2,]           5 0.02241521
.
.
.
[7,]          10 0.11937806
[8,]          11 0.11937806
.
.
.
[13,]         16 0.03667956
    
```

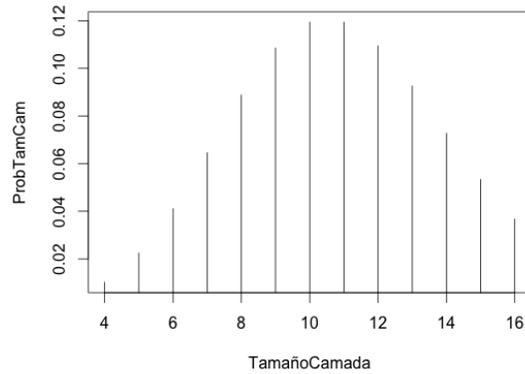
El gráfico de las probabilidades es:



```

R Studio

plot(TamanoCamada,ProbTamCam,type="h")
    
```



Con el comando *rpois* generamos 20 números pseudoaleatorios relacionados con el número de lechones por parto y cerda con un promedio de  $\lambda = 11$  lechones por camada, obedeciendo a una distribución de Poisson:

```

R Studio

set.seed (1)
TamanoCamada20Cerdas=rpois (n=20, lambda=11)
TamanoCamada20Cerdas

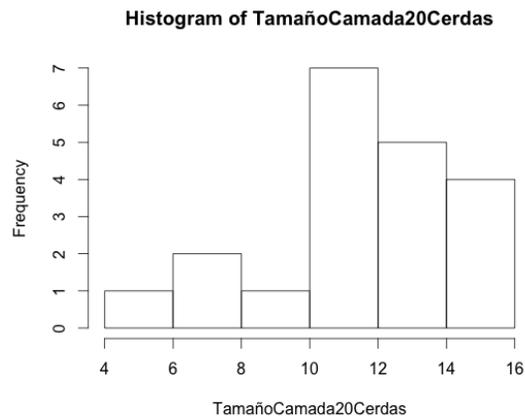
[1] 8 15 15 12 15 13 12 9 16 12 13 11 8 13 12 14 13 11 4 12
    
```

Realicemos el gráfico de la distribución de probabilidad de los números pseudoaleatorios simulados:

```

R Studio

hist (TamanoCamada20Cerdas)
    
```

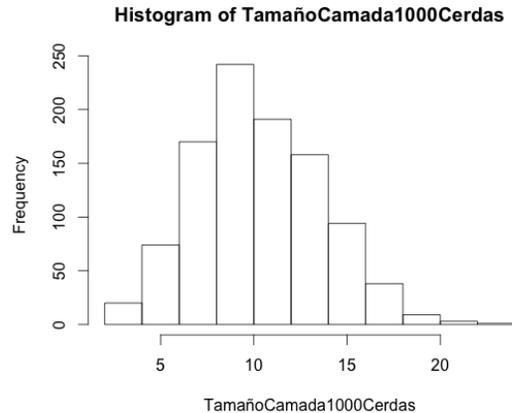


Ahora generaremos 1000 números pseudoaleatorios relacionados con el número de lechones por parto y cerda con un promedio de 11 lechones por camada, obedeciendo a una distribución de Poisson:

```
R Studio  
TamanoCamada1000Cerdas=rpois (n=1000, lambda=11)  
TamanoCamada1000Cerdas  
[1] 14 8 9 12 15 10 12 10 6 14 9 15 10 16 12 9 8 8 5 13 10 13 12  
[24] 8 8 15 17 9 7 6 18 10 13 11 8 6 15 11 18 12 8 10 13 14 11 10  
.  
.  
[967] 15 7 9 11 9 15 11 13 3 10 10 13 15 5 12 11 11 8 15 10 13 11 9  
[990] 16 11 10 12 12 8 9 13 8 13 12
```

El gráfico de la distribución de probabilidades es:

```
R Studio  
hist (TamanoCamada1000Cerdas)
```



#### 2.4. Números pseudoaleatorios con distribución hipergeométrica

La distribución hipergeométrica es una distribución discreta que se caracteriza porque el proceso muestral no tiene reposición.

Consideremos el caso de un muestreo de aves que presentan daño hepático. Durante el proceso de muestreo las aves son sacrificadas y no regresan al respectivo galpón, por tanto, no hay reposición.

La variable, en principio, tendría una distribución binomial, pero por no existir reposición

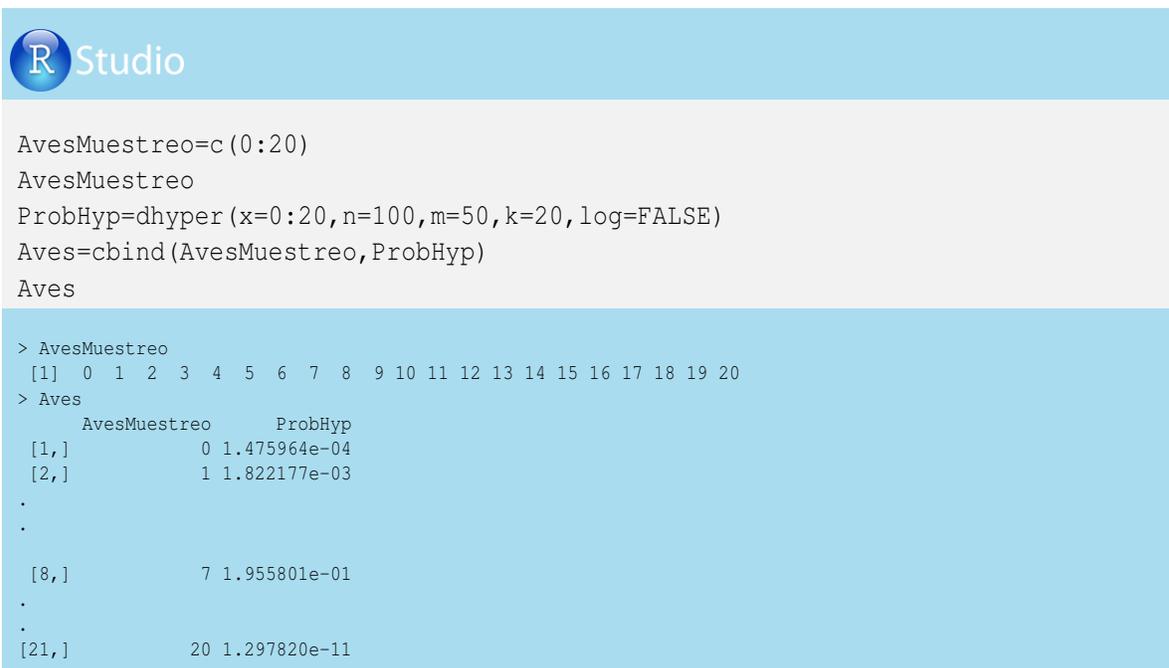
tiene una distribución hipergeométrica. Sin embargo, cuando el tamaño de la población es grande, esta distribución se asemeja a la binomial.

La función de masa de probabilidad para una variable aleatoria con distribución hipergeométrica es:

$$f(x;n;m;k) = \frac{\binom{m}{x} \binom{n}{k-x}}{\binom{m+n}{k}}$$

Donde  $x$  es el número de éxitos en la muestra,  $n$  es el número de fracasos del espacio muestral,  $m$  es el número de éxitos en el espacio muestral y  $k$  es el número de ensayos.

Veamos el siguiente ejemplo de un experimento con 150 aves, donde se espera encontrar 100 aves con daño hepático ( $n = 100$ ) y 50 sanas ( $m = 50$ ). Se tomará una muestra del lote y se sacrificarán 20 aves ( $k = 20$ ) para medir diferentes órganos internos afectados (creamos un vector con especificaciones 0:20). Se desea saber cuál es la probabilidad de encontrar de 0 a 20 aves enfermas. Utilizaremos en R-project el comando *dhyper*, indicando que no se aplique logaritmo a las probabilidades ( $\log = FALSE$ ):



```

R Studio

AvesMuestreo=c(0:20)
AvesMuestreo
ProbHyp=dhyper(x=0:20,n=100,m=50,k=20,log=FALSE)
Aves=cbind(AvesMuestreo,ProbHyp)
Aves

> AvesMuestreo
[1] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> Aves
      AvesMuestreo      ProbHyp
[1,]              0 1.475964e-04
[2,]              1 1.822177e-03
.
.
[8,]              7 1.955801e-01
.
.
[21,]             20 1.297820e-11

```

La gráfica de la distribución de probabilidades es:

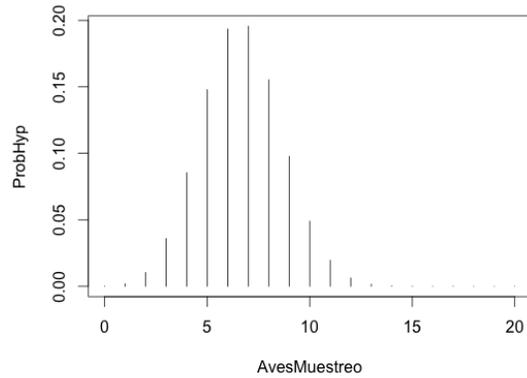


```

R Studio

plot(AvesMuestreo,ProbHyp,type="h")

```



Con el comando *rhyper* podemos simular 40 observaciones ( $nn = 40$ ), relacionadas con lotes o experimentos de muestreo de 20 aves ( $k = 20$ ) cada una. El número de aves sanas en cada muestreo es:

```

R Studio

set.seed (8)
AvesProb=rhyper (nn=40,m=50,n=100,k=20)
AvesProb

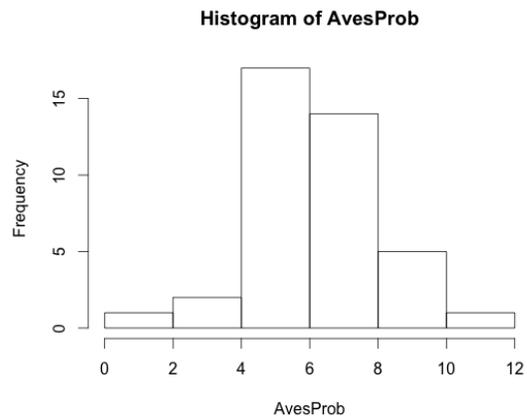
[1] 6 5 8 7 6 8 6 10 8 7 6 4 6 7 5 10 1 5 5 7 5 6 7 5
[25] 7 11 4 7 7 8 5 7 10 5 6 6 10 10 6 8
    
```

El gráfico de la distribución de probabilidad de los números pseudoaleatorios simulados es:

```

R Studio

hist (AvesProb)
    
```



## 2.5. Números pseudoaleatorios con distribución geométrica

La distribución geométrica es un caso especial de la distribución binomial y su distribución de probabilidad está dada por las siguientes dos situaciones:

- ♣ Sea  $X$  el número de ensayos de Bernoulli necesarios para obtener un éxito, después de un conjunto de sucesos no exitosos, con una probabilidad  $p$  de ocurrencia del suceso. La función de masa de probabilidad es:

$$p(X = x) = (1 - p)^{x-1} p, \text{ para } x = \{1, 2, 3, \dots\}$$

- ♣ Sea  $Y$  el número de fallas antes de obtener el primer evento exitoso. En este caso, la variable  $Y$  depende de  $X$  porque  $Y = X - 1$ . La función de masa de probabilidad es:

$$p(Y = y) = (1 - p)^y p, \text{ para } y = \{0, 1, 2, 3, \dots\}$$

En estos dos casos la secuencia de probabilidades es una progresión geométrica, que se caracteriza porque los eventos son independientes, no existe un número determinado de repeticiones y está relacionada con tiempos de espera o con intervalos regulares de tiempo.

Veamos un ejemplo: Una empresa de biotecnología reproductiva bovina implanta el 10% de los embriones producidos. Si se desea saber las probabilidades de tener éxito cuando se transfieren embriones en 1, 2, 3 y 4 vacas receptoras, se tendría el siguiente esquema:

Ensayos	Fallas	Éxito	Vacas transferidas				$\prod(1 - p)$	$p$	$\prod(1 - p) * p$
			1	2	3	4			
1	0	1	p=0.1					0.1	0.1
2	1	1	q=0.9	p=0.1			0.9	0.1	0.09
3	2	1	q=0.9	q=0.9	p=0.1		0.81	0.1	0.081
4	3	1	q=0.9	q=0.9	q=0.9	p=0.1	0.729	0.1	0.0729

Dicho de otra forma, la función masa de probabilidad, cuando se consideran los dos casos de distribución anteriormente mencionados, sería:

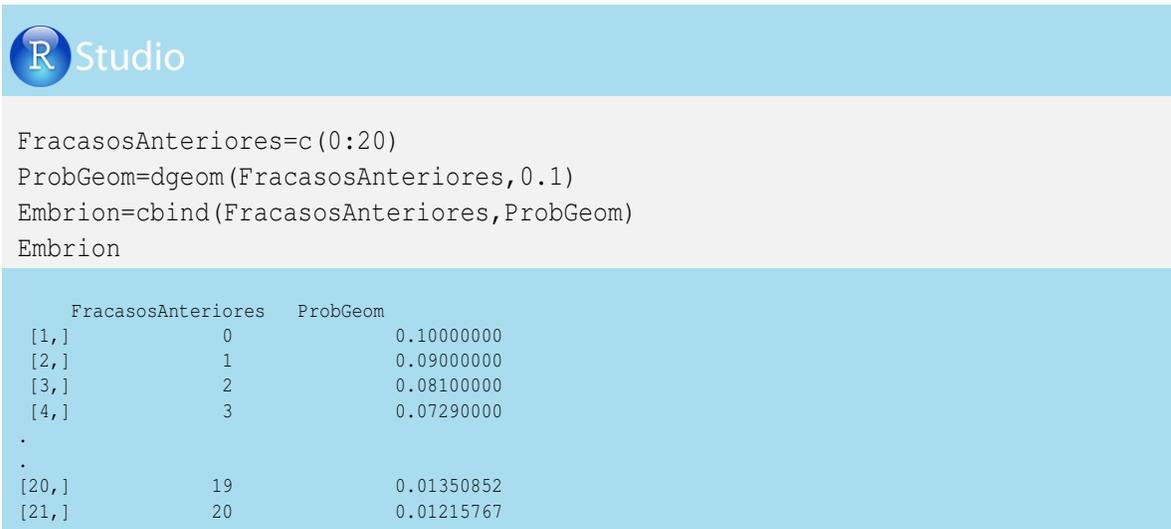
- ♣ Número de ensayos de Bernoulli necesarios para obtener un éxito:

$$\begin{aligned} f_{x=1} &= (1 - 0.1)^{1-1} * 0.1 = 1 * 0.1 = 0.1 \\ f_{x=2} &= (1 - 0.1)^{2-1} * 0.1 = 0.9 * 0.1 = 0.09 \\ f_{x=3} &= (1 - 0.1)^{3-1} * 0.1 = 0.81 * 0.1 = 0.081 \\ f_{x=4} &= (1 - 0.1)^{4-1} * 0.1 = 0.729 * 0.1 = 0.0729 \end{aligned}$$

- ♣ Número de fallas antes de obtener el primer evento exitoso:

$$\begin{aligned} f_{y=0} &= (1 - 0.1)^0 * 0.1 = 1 * 0.1 = 0.1 \\ f_{y=1} &= (1 - 0.1)^1 * 0.1 = 0.9 * 0.1 = 0.09 \\ f_{y=2} &= (1 - 0.1)^2 * 0.1 = 0.81 * 0.1 = 0.081 \\ f_{y=3} &= (1 - 0.1)^3 * 0.1 = 0.729 * 0.1 = 0.0729 \end{aligned}$$

Utilicemos el comando *dgeom* para obtener las probabilidades del número de fracasos ( $y = 0, 1, 2, 3, 4, \dots, 20$ ) en la implantación de embriones antes de obtener un éxito. Inicialmente generamos un vector relacionado con el número de embriones no implantados y luego aplicamos el comando *dgeom* indicando el vector que contiene el número de fracasos y la probabilidad de éxito ( $p = 0.1$ ):

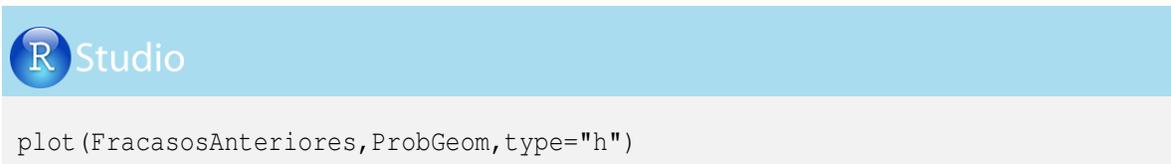


```
R Studio

FracasosAnteriores=c(0:20)
ProbGeom=dgeom(FracasosAnteriores,0.1)
Embrion=cbind(FracasosAnteriores,ProbGeom)
Embrion
```

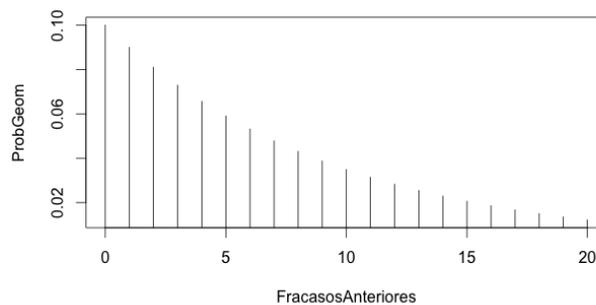
	FracasosAnteriores	ProbGeom
[1,]	0	0.10000000
[2,]	1	0.09000000
[3,]	2	0.08100000
[4,]	3	0.07290000
.		
.		
[20,]	19	0.01350852
[21,]	20	0.01215767

La gráfica de la distribución de probabilidad es:



```
R Studio

plot(FracasosAnteriores,ProbGeom,type="h")
```



Con el comando `rgeom` podemos simular 50 números pseudoaleatorios ( $n = 50$ ) relacionados con el número de fracasos hasta lograr el implante de un embrión:

```

R Studio

set.seed (9)
EmbrioProb=rgeom(n=50,prob=0.10)
EmbrioProb

[1] 19  4  4 10 18 16  2  0 13  1 14  1 14  6  1  6 11  1 13
[20]  9  2  7 18  0 13  4 22  3  7  9  2 11  2  7  4  5  0 20
[39]  0  3  8 19 15  4  5  1 10  3 34  2

```

El primer valor indica que se logró implantar el embrión después de 19 intentos fallidos, en el segundo caso se esperaron 4 transferencias para lograr un éxito, y así sucesivamente.

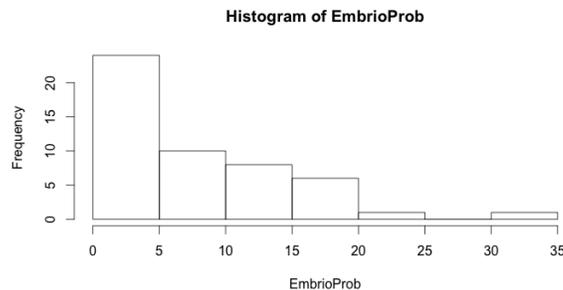
El gráfico de la distribución de probabilidad de los números pseudoaleatorios generados es:

```

R Studio

hist(EmbrioProb)

```



## 2.6. Números pseudoaleatorios con distribución uniforme continua

La distribución de probabilidad uniforme continua considera que una variable aleatoria puede asumir  $x$  valores en un intervalo con igual probabilidad de ocurrir.

La función densidad de probabilidad de un valor en un rango determinado está dada por:

$$f(x) = \frac{1}{(max - min)}$$

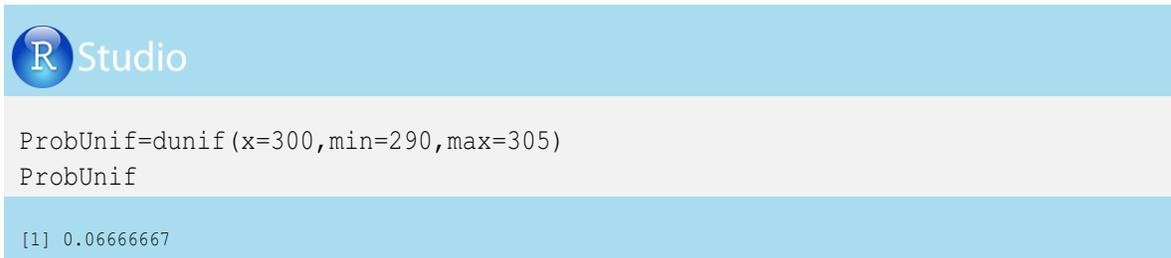
Donde  $x$  es el valor posible que se puede asumir en un intervalo que tiene un valor mínimo ( $min$ ) y uno máximo ( $max$ ).

Veamos un ejemplo: Una panadería compra huevos yumbo a un precio de entre \$290 y \$305 por huevo. ¿Cuál es la probabilidad de que a usted como nuevo proveedor le compren a \$300, si se sabe que el precio para el pago del huevo sigue una distribución uniforme?

La probabilidad de que le compren los huevos a \$300 por unidad en el rango determinado es:

$$f(300) = \frac{1}{(305 - 290)} = 0.067$$

Veamos cómo obtener la probabilidad de interés en R-project:



```

R Studio

ProbUnif=dunif(x=300,min=290,max=305)
ProbUnif

[1] 0.06666667
    
```

La probabilidad de que el precio del huevo sea de \$300 es de 0.067.

La distribución uniforme continua con un rango entre 0 y 1 es utilizada ampliamente en los procesos de simulación con cualquier tipo de distribución.

## 2.7. Números pseudoaleatorios con distribución normal

La distribución normal es una de las distribuciones de probabilidad de variables continuas que permite modelar numerosos fenómenos naturales. Esta distribución está completamente caracterizada por dos parámetros: la media ( $\mu$ ) y la varianza ( $\sigma^2$ ), cuya raíz cuadrada es la desviación estándar de la distribución. La función densidad para una variable  $X$  con distribución normal ( $X \sim N(\mu, \sigma^2)$ ) está dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

La curva en forma de campana que genera la distribución normal es simétrica, siendo el punto central el valor de  $\mu_x$  con dos puntos de inflexión (en  $x = \mu - \sigma$  y  $x = \mu + \sigma$ ).

Otras propiedades de la distribución normal son:

- ♣  $\mu_x$  es igual a la *mediana*<sub>x</sub> y a la *moda*<sub>x</sub>,
- ♣ el 68.26 % de los valores de  $x$  están entre  $x = \mu_x - \sigma_x$  y  $x = \mu_x + \sigma_x$ ,
- ♣ el 95.44 % de los valores de  $x$  están entre  $x = \mu_x - 2\sigma_x$  y  $x = \mu_x + 2\sigma_x$ ,
- ♣ el 99.74 % de los valores de  $x$  están entre  $x = \mu_x - 3\sigma_x$  y  $x = \mu_x + 3\sigma_x$ ,
- ♣ si  $X \sim N(\mu, \sigma^2)$  y  $\beta$  es un número real, entonces  $(X + \beta) \sim N(\mu + \beta, \sigma^2)$ , y
- ♣ si  $X \sim N(\mu, \sigma^2)$  y  $\beta$  es un número real, entonces  $(\beta X) \sim N(\beta\mu, \beta^2\sigma^2)$ .

Generemos en R-project las probabilidades de encontrar individuos con pesos entre 100 y 300 kg, donde la variable peso tiene una distribución normal con media de 190 kg y desvío de 4 kg.

```

R Studio

Peso=c(100:300)
Peso
ProbNorm=dnorm(x=Peso, mean=190, sd=4)
Pesaje=cbind(Peso,ProbNorm)
Pesaje

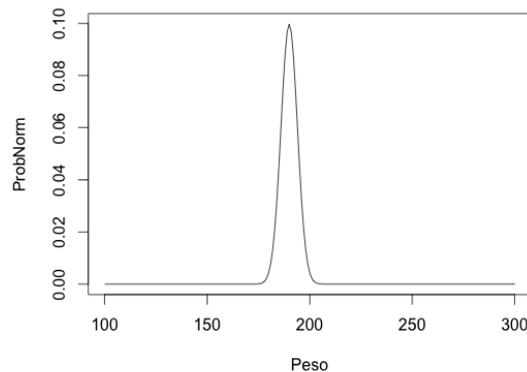
> Peso
 [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
[19] 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
.
.
[199] 298 299 300
> Pesaje
      Peso      ProbNorm
[1,]  100 1.169659e-111
[2,]  101 3.143360e-109
.
.
[97,]  196 3.237940e-02
[98,]  197 2.156933e-02
.
.
[200,] 299 5.668289e-163
[201,] 300 6.042951e-166
    
```

El gráfico de la distribución normal generado es el siguiente:

```

R Studio

plot(Peso,ProbNorm,type="l")
    
```



En el siguiente ejemplo modificaremos los parámetros de la distribución normal, considerando una media de peso de 190 kg y una desviación estándar de 30 kg, con el objetivo de mostrar cómo cambia el gráfico de la función de probabilidad dependiendo de los valores que asumen los parámetros. Veamos la programación en R-project, considerando las probabilidades de encontrar individuos con peso entre 100 y 300 kg:

```
R Studio

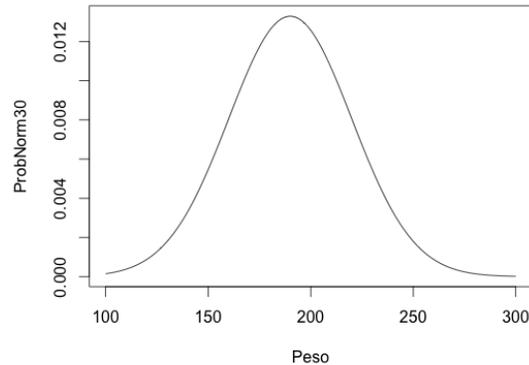
Peso=c(100:300)
Peso
ProbNorm30=dnorm(x=Peso, mean=190, sd=30)
ProbNorm30
Pesaje=cbind(Peso,ProbNorm30)
Pesaje

> Peso
 [1] 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
 [21] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
 .
 .
 [201] 300
> Pesaje
      Peso ProbNorm30
 [1,]  100 1.477283e-04
 [2,]  101 1.631743e-04
 .
 .
 [103,]  202 1.227567e-02
 [104,]  203 1.210635e-02
 .
 .
 [200,]  299 1.808023e-05
 [201,]  300 1.600902e-05
```

El gráfico para la distribución normal con media de 190 kg y desvío de 30 kg es el siguiente:

```
R Studio

plot(Peso,ProbNorm30,type="l")
```



En este nuevo ejercicio se presenta la generación de números pseudoaleatorios del peso de terneros al destete, la cual presenta distribución normal. Los terneros pertenecen a tres haciendas con 500, 50 y 5 animales destetados. El peso promedio es de 180 kg y se tiene un desvío de 4 kg. Veamos la generación de los pesos en los tres casos en R-project, recordando que el comando *set.seed* fija la semilla que permite obtener los mismos resultados presentados en este libro:

♣ En el caso de la hacienda con 500 animales:

```

R Studio

set.seed(9)
PesoDestete500=rnorm(n=500,mean=180,sd=4)
PesoDestete500
summary(PesoDestete500)

> PesoDestete500
 [1] 176.9328 176.7342 179.4339 178.8896 181.7452 175.2525 184.7679 179.9272
 [9] 179.0077 178.5483 185.1103 178.1244 180.2842 178.9358 187.3810 176.6422
 .
 .
 [489] 180.7558 177.9910 181.1691 177.8301 182.1293 179.7124 173.4065 184.2952
 [497] 188.8007 178.1974 185.0189 177.6971
> summary(PesoDestete500)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 167.8  177.4   179.9   180.0  182.4   191.1

```

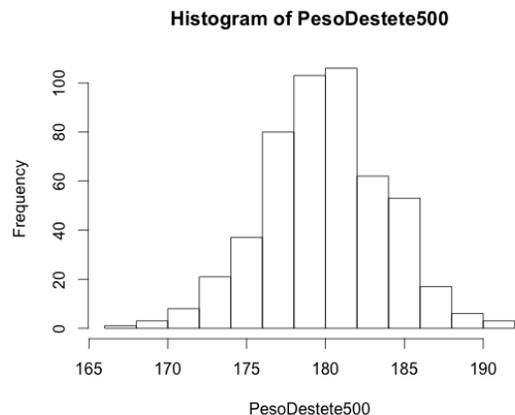
Veamos el histograma de frecuencia generado para el peso al destete simulado de los 500 animales:

```

R Studio

hist(PesoDestete500)

```



♣ En el caso de la hacienda con 50 animales, la programación en R-project es la siguiente:

```

R Studio

set.seed(9)
PesoDestete50=rnorm(n=50,mean=180,sd=4)
PesoDestete50
summary(PesoDestete50)

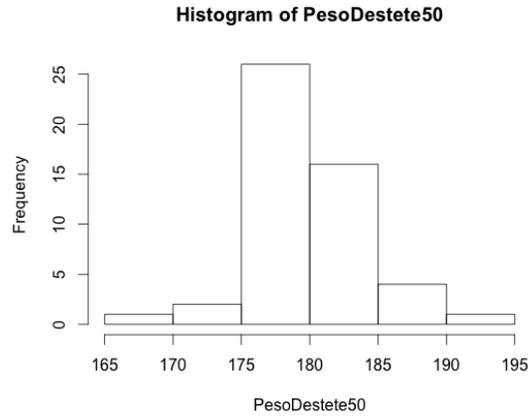
[1] 176.9328 176.7342 179.4339 178.8896 181.7452 175.2525 184.7679 179.9272
 [9] 179.0077 178.5483 185.1103 178.1244 180.2842 178.9358 187.3810 176.6422
[17] 179.6902 169.5292 183.5515 177.1700 187.0280 180.7290 178.9324 183.7057
[25] 177.2267 190.7280 180.8901 177.1733 181.6689 181.4782 176.3449 178.7323
[33] 184.1962 180.6724 180.1258 175.9587 181.5310 176.7212 181.4468 180.3735
[41] 176.7662 171.9225 177.0451 181.5315 186.9235 179.1877 176.0144 174.7739
[49] 176.0087 177.7399
> summary(PesoDestete50)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 169.5  177.0   179.1   179.6   181.5   190.7
    
```

El histograma de frecuencia resultante es:

```

R Studio

hist(PesoDestete50)
    
```



♣ En el caso de la hacienda con 5 animales, la programación en R-project es la siguiente:

```

R Studio

set.seed(9)
PesoDestete5=rnorm(n=5,mean=180,sd=4)
PesoDestete5
summary(PesoDestete5)

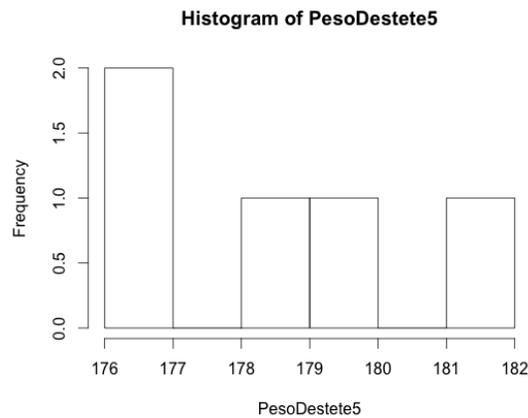
[1] 176.9328 176.7342 179.4339 178.8896 181.7452
> summary(PesoDestete5)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 176.7  176.9  178.9  178.7  179.4  181.7
    
```

El histograma de frecuencia es:

```

R Studio

hist(PesoDestete5)
    
```



## 2.8. Números pseudoaleatorios con distribución continua lognormal

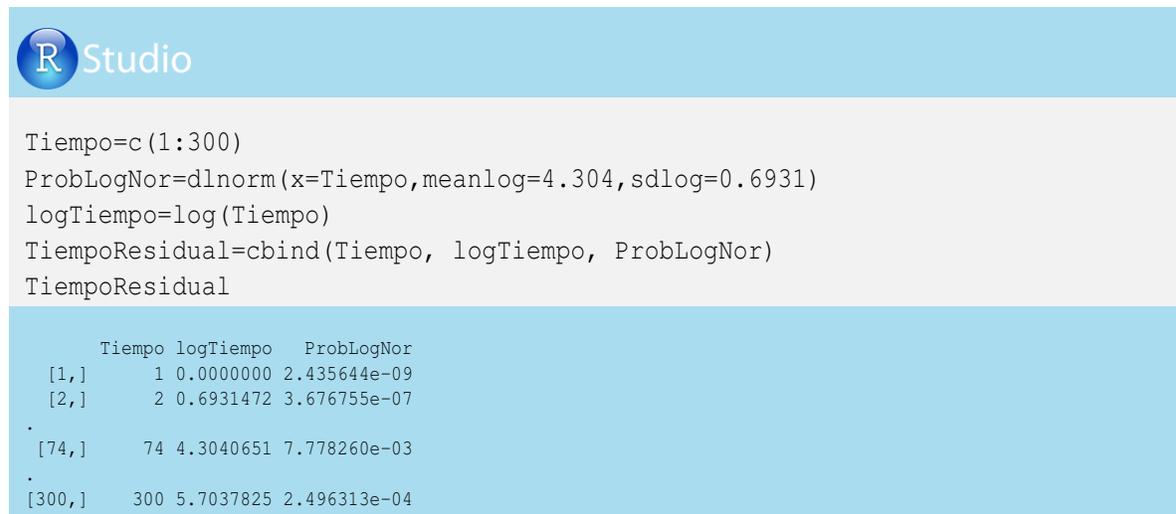
Esta distribución es frecuentemente utilizada para explicar variables que presentan asimetría, donde la mayoría de valores ocurren en las proximidades del valor mínimo o del valor máximo, lo que genera que la media aritmética esté desplazada, haciendo que la mediana sea una mejor medida para la valoración del centro de los datos.

Una variable aleatoria continua  $X$  es lognormalmente distribuida ( $X \sim \text{LogN}(\mu, \sigma^2)$ ), si  $\log(X)$  tiene una distribución normal ( $\log(X) \sim N(\mu, \sigma^2)$ ). La función densidad de probabilidad de una variable aleatoria con distribución lognormal está dada por:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(\ln(x)-\mu)^2}, x > 0$$

Este tipo de distribución es ampliamente utilizada en la medicina veterinaria y en la microbiología, en estudios de tiempos de supervivencia, tiempos de incubación, tiempos de presentación de residuos en leche, títulos de anticuerpos, entre otros. Consideremos como ejemplo el siguiente caso:

En un estudio previo se demostró que la presencia en el tiempo ( $t$  en horas) de residuos de antibióticos en la leche de vacas tratadas para mastitis sigue una distribución lognormal con media logarítmica natural de  $\ln(t) = 4.304$ , cuyo valor inverso es  $t = e^{4.303} = 74$  horas y una desviación estándar de  $\ln(t) = 0.6931$ , cuyo valor inverso es  $t = e^{0.6931} = 2$  horas. La programación en R-project para generar la función densidad de una distribución lognormal para una media de 4.304 y un desvío estándar de 0.6931, desde la primera hora de aplicación del antibiótico ( $\ln(t) = 0$ , donde  $t = e^0 = 1$  hora) hasta  $\ln(t) = 5.7037825$ , donde  $t = e^{5.7037825} = 300$  horas de aplicado el producto, es el siguiente:



```

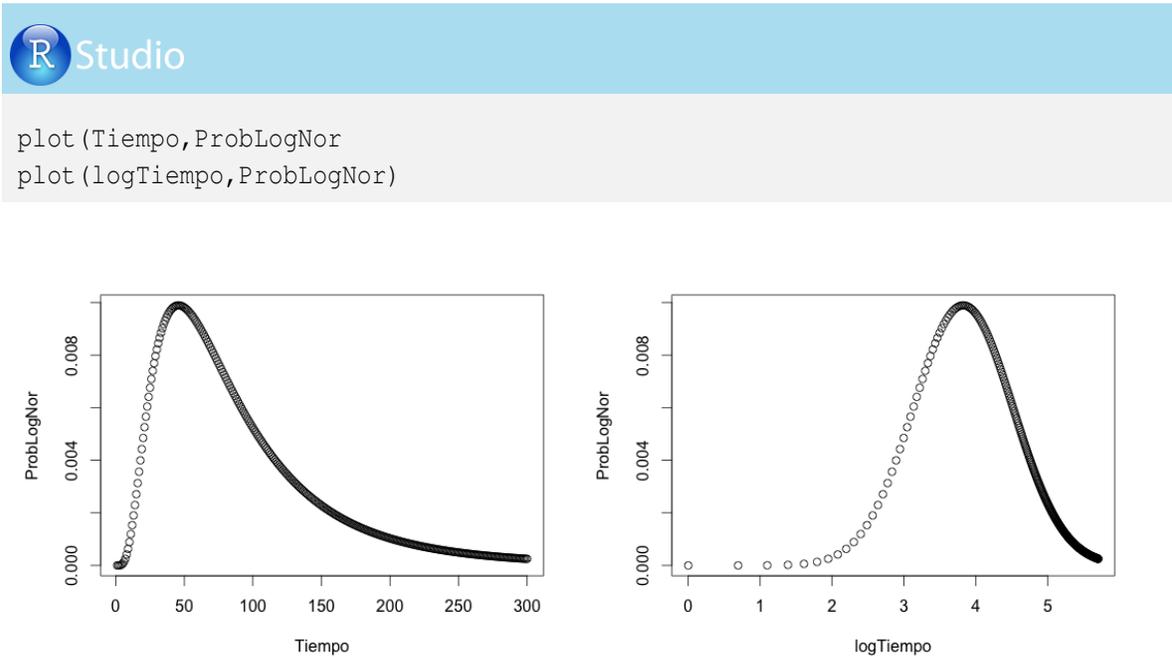
R Studio

Tiempo=c(1:300)
ProbLogNor=dlnorm(x=Tiempo,meanlog=4.304,sdlog=0.6931)
logTiempo=log(Tiempo)
TiempoResidual=cbind(Tiempo, logTiempo, ProbLogNor)
TiempoResidual

      Tiempo logTiempo  ProbLogNor
[1,]      1 0.0000000 2.435644e-09
[2,]      2 0.6931472 3.676755e-07
.
[74,]     74 4.3040651 7.778260e-03
.
[300,]    300 5.7037825 2.496313e-04

```

Los gráficos de las distribuciones de las variables tiempo y lognormal del tiempo son los siguientes:



A continuación se presenta la generación de números pseudoaleatorios de la variable horas de retiro, que presenta una distribución lognormal, con escala de 4.304 y forma 0.6931, como se indicó anteriormente. Generemos en R-project 500 números pseudoaleatorios:

```

R Studio

set.seed (7)
TiempoRetiro=rlnorm(n=500,meanlog=4.304,sdlog=0.6931)
TiempoRetiro
summary(TiempoRetiro)
sd(TiempoRetiro)

> TiempoRetiro
 [1] 361.149142  32.282250  45.731432  55.603145  37.759092  38.376307
.
[499] 137.992793  53.928591
> summary(TiempoRetiro)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 9.424  47.840  75.100  96.870 122.600 497.800
> sd(TiempoRetiro)
 [1] 73.69112
    
```

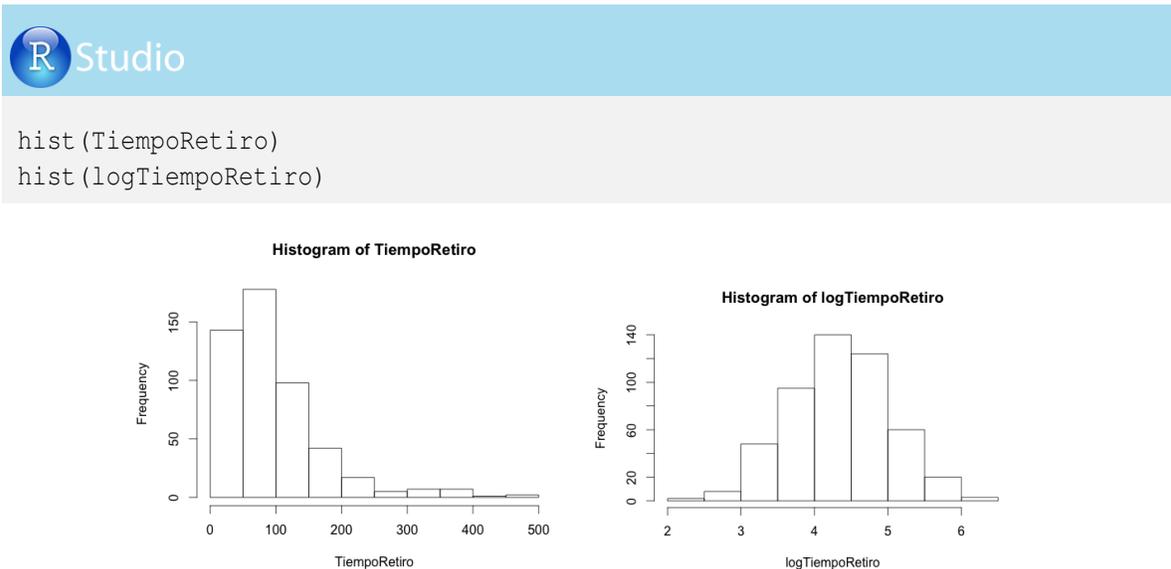
Si generamos el logaritmo natural de los datos de la variable *TiempoRetiro*, tenemos:

```
R Studio

logTiempoRetiro=log(TiempoRetiro)
logTiempoRetiro
summary(logTiempoRetiro)
sd(logTiempoRetiro)

> logTiempoRetiro
[1] 5.8893 3.4745 3.8228 4.0182 3.6312 3.6474 4.8225 4.2229 4.4098
.
496] 3.5293 3.7252 5.3442 4.9272 3.9877
> summary(logTiempoRetiro)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 2.243  3.868   4.319   4.335   4.809   6.210
> sd(logTiempoRetiro)
[1] 0.6918144
```

Los histogramas de frecuencias del tiempo de retiro y el logaritmo del tiempo de retiro son los siguientes:



## 2.9. Números pseudoaleatorios con distribución continua logística

La distribución logística se caracteriza porque su función pertenece a la familia logística que aparece en el contexto de regresión *log*. Esta distribución es utilizada principalmente en estudios de crecimiento poblacional (demografía); por ejemplo, el crecimiento de hongos y bacterias en cajas de Petri.

La función densidad de la distribución logística presenta una forma muy semejante a la

función densidad de la distribución normal, pero con colas más pesadas. Su parámetro de posición es  $m$  y el parámetro de escala o dispersión es  $s$ , y la función densidad de la distribución logística está dada por:

$$f(x) = \frac{e^{-\frac{(x-m)}{s}}}{s(1 + e^{-\frac{x-m}{s}})^2}$$

Veamos el ejemplo de la tasa de crecimiento anual de la población vacuna en Colombia, donde se espera que la población vacuna aumente en 10 % (posición) y con una escala de 2. Asumiendo que la variable tasa de crecimiento anual tiene una distribución logística, generaremos en R-project las probabilidades para las diferentes tasas de crecimiento poblacional de vacunos en Colombia (desde un decrecimiento del 10 %, hasta una tasa de crecimiento del 30 %); donde la variable tasa de crecimiento tiene una distribución logística con posición 10 y escala 2. Veamos la programación en R-project:



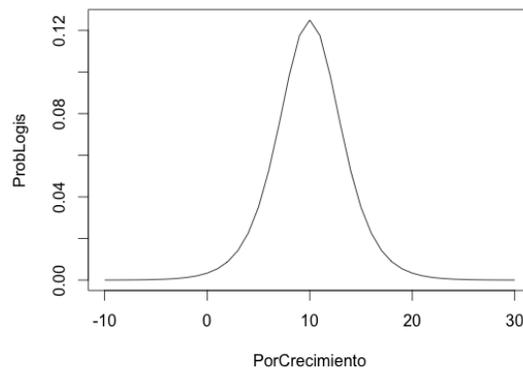
```
PorCrecimiento=c(-10:30)
ProbLogis=dlogis(x=PorCrecimiento,location=10,scale=2)
Creces=cbind(PorCrecimiento,ProbLogis)
Creces
```

```
PorCrecimiento  ProbLogis
[1,]           -10 2.269790e-05
[2,]            -9 3.742031e-05
.
[41,]           30 2.269790e-05
```

El gráfico de la curva de probabilidad de la distribución logística con parámetro de posición 10 y de escala 2, para el ejemplo propuesto, es:



```
plot(PorCrecimiento,ProbLogis,type="l")
```



A continuación se presenta la generación de 30 números pseudoaleatorios de la distribución logística en R-project. Para tal fin, consideremos la variable tasa de crecimiento poblacional de vacunos en Colombia, en dos posibles escenarios. El primero con un parámetro de posición 10 y de escala 2, y el segundo con un parámetro de posición 10 y escala 4:

♣ Tasa de crecimiento con distribución logística con parámetro de posición 10 y escala 2:

```
R Studio

set.seed (7)
TasCrecpos10escala2=rlogis (n=40, location=10, scale=2)
TasCrecpos10escala2
summary (TasCrecpos10escala2)

 [1]  5.349601 13.554117  8.275166 -1.083889 11.528707 10.737728 14.069514 13.842486
 [9]  9.677844 14.559937  6.338606 15.635942  5.504145 12.502710 14.318408 14.183571
[17]  7.834313 11.359673  8.006590 10.923016  9.319242  8.972021  8.963944  7.763423
[25]  7.897938 14.990589 11.080331 -2.343958  7.214033 20.998240  4.490015 10.121862
[33]  8.995850 10.272398 12.152634  8.851489 10.418314 10.893675  8.861294  6.784160
> summary (TasCrecpos10escala2)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.344  7.882   9.900   9.845 12.240  21.000
```

♣ Tasa de crecimiento con distribución logística con parámetro de posición 10 y escala 4:

```
R Studio

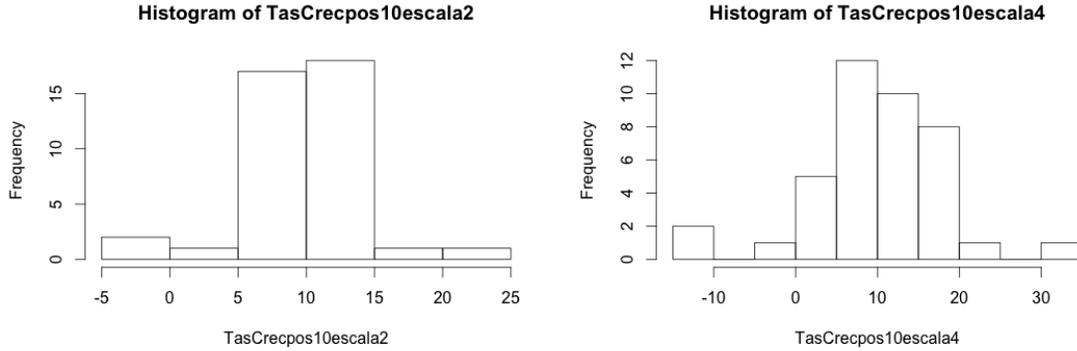
set.seed (7)
TasCrecpos10escala4=rlogis (n=40, location=10, scale=4)
TasCrecpos10escala4
summary (TasCrecpos10escala4)

 [1]  0.699202 17.108234  6.550332 -12.167777 13.057414 11.475457 18.139028
 [8] 17.684972  9.355687 19.119874  2.677211 21.271883  1.008291 15.005420
[15] 18.636815 18.367142  5.668626 12.719346  6.013179 11.846032  8.638483
[22]  7.944042  7.927889  5.526845  5.795877 19.981178 12.160661 -14.687917
[29]  4.428066 31.996480 -1.019970 10.243724  7.991701 10.544796 14.305268
[36]  7.702978 10.836628 11.787350  7.722588  3.568320
> summary (TasCrecpos10escala4)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-14.690  5.764   9.800   9.691 14.480  32.000
```

Los histogramas de frecuencia de los conjuntos de datos generados son:

```
R Studio

hist (TasCrecpos10escala2)
hist (TasCrecpos10escala4)
```



**2.10. Números pseudoaleatorios con distribución multinomial**

La distribución multinomial es una generalidad de la distribución binomial, cuando se tienen más de dos categorías o sucesos independientes en cada ensayo ( $A_1, A_2, \dots, A_k$ , con  $k > 2$ ), con probabilidades de ocurrencia  $p_i$  que cumplen la siguiente propiedad:

$$P(A_1) = p_1; P(A_2) = p_2; \dots; P(A_k) = p_k : k \text{ con } \sum_{i=1}^k p_i = 1$$

Supongamos que bajo condiciones independientes se tienen  $n$  pruebas o ensayos y deseamos conocer la probabilidad de que el suceso  $A_1$  aparezca  $x_1$  veces, el suceso  $A_2$  aparezca  $x_2$  veces y así sucesivamente hasta el suceso  $A_k$  que aparece  $x_k$  veces; en este caso tendríamos:

$$P[(A_1 = x_1) \cap (A_2 = x_2) \cap \dots \cap (A_k = x_k)]$$

La función densidad de la distribución multinomial de parámetros  $n$  y  $p_i = (p_1 \dots p_k)$  está dada por:

$$f(x_1, \dots, x_k; n; p_1, \dots, p_k) = \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}$$

Con:

$$\sum_{i=1}^k p_i = 1 \text{ y } \sum_{i=1}^k x_i = n$$

Veamos un ejemplo práctico de la distribución multinomial: En un estudio previo en una ganadería, se encontró que el 20% de las vacas presentaron mastitis subclínica y el 10% presentaron mastitis clínica. Si pretendemos realizar la prueba de CMT (California Mastitis Test) en 20 vacas escogidas aleatoriamente del hato, ¿cuál sería la probabilidad de encontrar estos resultados?:

♣ 10 vacas sanas, 10 con mastitis subclínica y ninguna con mastitis clínica:

$$f(10, 10, 0; 20; 0.7, 0.2, 0.1) = \frac{20!}{10!10!0!} 0.7^{10} 0.2^{10} 0.1^0 = 0.00053$$

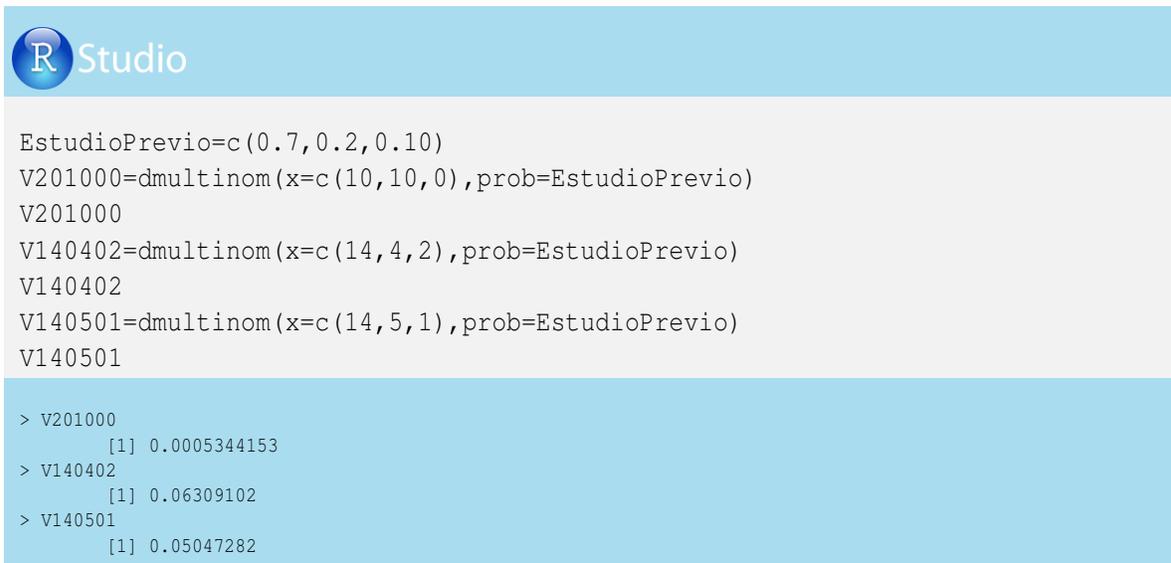
♣ 14 vacas sanas, 4 con mastitis subclínica y 2 con mastitis clínica:

$$f(14, 4, 2; 20; 0.7, 0.2, 0.1) = \frac{20!}{14!4!2!} 0.7^{14} 0.2^4 0.1^2 = 0.06309$$

♣ 14 vacas sanas, 5 con mastitis subclínica y 1 con mastitis clínica:

$$f(14, 5, 1; 20; 0.7, 0.2, 0.1) = \frac{20!}{14!5!1!} 0.7^{14} 0.2^5 0.1^1 = 0.0504$$

Con el comando *dmultinom* del R-project podemos calcular las probabilidades anteriormente enunciadas. En el argumento *prob* = indicamos un vector con las probabilidades encontradas en el estudio previo de mastitis en una ganadería, al que llamamos *EstudioPrevio*. Veamos:



```

R Studio

EstudioPrevio=c(0.7,0.2,0.10)
V201000=dmultinom(x=c(10,10,0),prob=EstudioPrevio)
V201000
V140402=dmultinom(x=c(14,4,2),prob=EstudioPrevio)
V140402
V140501=dmultinom(x=c(14,5,1),prob=EstudioPrevio)
V140501

> V201000
      [1] 0.0005344153
> V140402
      [1] 0.06309102
> V140501
      [1] 0.05047282
    
```

A continuación se presenta la generación de números pseudoaleatorios de la distribución multinomial en R-project. Consideremos un hato de 100 vacas donde la variable salud de la ubre sigue una distribución multinomial con parámetros  $p = (70\%, 20\%, 10\%)$ , que indica la probabilidad de tener 70% de vacas sanas, 20% de vacas con mastitis subclínica y 10% de vacas con mastitis clínica.

Utilizaremos el comando *rmultinom*, con el argumento *n* que indica el número de datos simulados (crearemos 20 datos simulados), el argumento *size* que es el número de observaciones (100 vacas) y el argumento *prob*, que indica las probabilidades de los sucesos:

```

R Studio

Probabilidades=c(0.7,0.2,0.10)
AleaMultin=rmultinom(n=20,size=100,prob=Probabilidades)
AleaMultin=t(AleaMultin)
colnames(AleaMultin)=(c("Sana","Subclinica","Clinica"))
AleaMultin
summary(AleaMultin)

> AleaMultin
  Sana Subclinica Clinica
[1,]  67         17    16
[2,]  76         14    10
.
.
[19,] 66         23    11
[20,] 68         16    16
> summary(AleaMultin)
  Sana      Subclínica      Clínica
Min.   :65.00  Min.   :14.00  Min.   : 5.00
1st Qu.:67.75  1st Qu.:17.75  1st Qu.: 9.00
Median :69.00  Median :19.00  Median :11.00
Mean   :69.40  Mean   :19.50  Mean   :11.10
3rd Qu.:71.25  3rd Qu.:20.50  3rd Qu.:13.25
Max.   :76.00  Max.   :27.00  Max.   :16.00

```

**2.11. Números pseudoaleatorios con distribución normal bivalente**

La variable bidimensional de dos variables continuas (x y y) tiene una distribución normal bivariada si su función de probabilidad conjunta está dada por la siguiente expresión:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}[(\frac{x-\mu_x}{\sigma_x})^2 - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + (\frac{y-\mu_y}{\sigma_y})^2]}$$

La función de probabilidad anterior depende de cinco parámetros: las medias de x y y (μ<sub>x</sub> y μ<sub>y</sub>), las desviaciones estándar de x y y (σ<sub>x</sub> y σ<sub>y</sub>) y la correlación entre las dos variables (ρ<sub>x,y</sub>). Las variables x y y pueden asumir valores de -∞ hasta ∞, σ > 0 y -1 ≤ ρ ≤ 1.

Veamos un ejemplo de vacunos que serán vendidos para sacrificio según el peso (media de 400 kg, con desviación estándar de 10 kg) y espesor de grasa de cadera (con media 0.45 cm, con desviación estándar de 0.05 cm). La correlación entre estas dos variables es de 0.80 (cuya covarianza es de 0.4).

Para la programación en R-project iniciamos con la generación de un vector con la media del peso y la media de la grasa (400 y 0.45 kg, respectivamente), como se indica a continuación:



```
Media=c(400,0.45)
```

Luego generamos una matriz de tamaño 2\*2 que tendrá en su diagonal principal las varianzas de las variables consideradas (peso y grasa) y fuera de la diagonal principal la covarianza entre las variables, así:

$$\begin{bmatrix} \sigma_{\text{peso}}^2 & \sigma_{\text{peso,grasa}} \\ \sigma_{\text{peso,grasa}} & \sigma_{\text{grasa}}^2 \end{bmatrix} = \begin{bmatrix} 10^2 & 0.4 \\ 0.4 & 0.05^2 \end{bmatrix} = \begin{bmatrix} 100 & 0.4 \\ 0.4 & 0.0025 \end{bmatrix}$$

La programación en R-project es la siguiente:



```
covar=matrix(c(100,0.4,0.4,0.0025),ncol=2)
covar
```

```
      [,1] [,2]
[1,] 100.0 0.4000
[2,]  0.4 0.0025
```

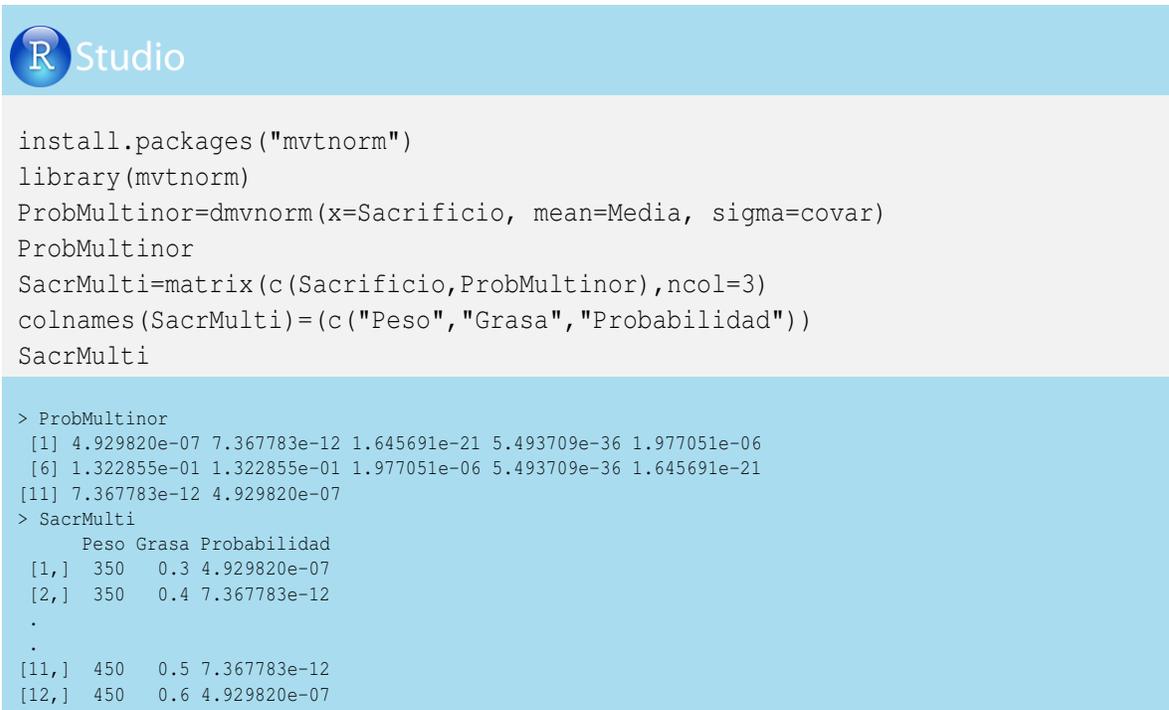
Generamos algunos valores que pueden asumir el peso y el espesor de grasa en una matriz con dos columnas, la primera relacionada con el peso (350 hasta 450) y la otra con la grasa (0.3 a 0.6):



```
Sacrificio=matrix(ncol=2,
c(350,350,350,350,400,400,400,400,450,450,450,450,
  0.3,0.4,0.5,0.6,0.3,0.4,0.5,0.6,0.3,0.4,0.5,0.6))
colnames(Sacrificio)=(c("peso","grasa"))
Sacrificio
```

```
> Sacrificio
      peso grasa
[1,] 350  0.3
[2,] 350  0.4
[3,] 350  0.5
[4,] 350  0.6
[5,] 400  0.3
.
.
[12,] 450  0.6
```

Para la distribución normal bivariante en R-project debemos instalar el paquete *mvtnorm* (Multivariate Normal and t Distributions de Genz et al., 2012 y Genz y Bretz, 2009). Para el cálculo de las probabilidades utilizaremos el comando *dmvnorm*, el cual requiere indicar de forma adecuada los argumentos. Primero se escribe el nombre de la hoja de datos que contiene los valores de las dos variables (en este caso la hoja de datos *Sacrificio*), y después se escriben los valores medios (en este caso *Media*) y se debe especificar la matriz de varianzas y covarianzas (en este caso *covar*). A continuación crearemos una hoja de datos de los pesos y el espesor de grasa de los animales y la respectiva probabilidad conjunta, a la cual llamamos *SacrMulti*:



```

install.packages("mvtnorm")
library(mvtnorm)
ProbMultinor=dmvnorm(x=Sacrificio, mean=Media, sigma=covar)
ProbMultinor
SacrMulti=matrix(c(Sacrificio,ProbMultinor),ncol=3)
colnames(SacrMulti)=(c("Peso","Grasa","Probabilidad"))
SacrMulti

> ProbMultinor
[1] 4.929820e-07 7.367783e-12 1.645691e-21 5.493709e-36 1.977051e-06
[6] 1.322855e-01 1.322855e-01 1.977051e-06 5.493709e-36 1.645691e-21
[11] 7.367783e-12 4.929820e-07
> SacrMulti
      Peso Grasa Probabilidad
[1,]  350   0.3 4.929820e-07
[2,]  350   0.4 7.367783e-12
.
.
[11,] 450   0.5 7.367783e-12
[12,] 450   0.6 4.929820e-07

```

Ahora simularemos los pesos y los espesores de grasa de 500 animales, directamente desde R-project, mediante el comando *rmvnorm*. Los argumentos de este comando son: el número de datos ( $n = 500$ ), el vector de medias y la matriz de varianzas-covarianzas que generamos anteriormente; veamos la programación:

```

R Studio

AnimalesSacrif=rmvnorm(n=500,mean=Media,sigma=covar)
colnames (AnimalesSacrif)=(c("Peso","Grasa"))
AnimalesSacrif

> AnimalesSacrif
      Peso      Grasa
[1,] 414.6867 0.5272045
[2,] 395.3344 0.4769984
.
.
[499,] 397.2186 0.4369669
[500,] 395.3380 0.4595097
    
```

Ahora veamos las medias, las varianzas y la covarianza de las dos variables:

```

R Studio

summary(AnimalesSacrif)
var(AnimalesSacrif)

> summary(AnimalesSacrif)
      Peso      Grasa
Min.   :373.8   Min.   :0.2910
1st Qu.:392.5   1st Qu.:0.4154
Median :399.2   Median :0.4467
Mean   :399.4   Mean   :0.4476
3rd Qu.:406.1   3rd Qu.:0.4812
Max.   :431.4   Max.   :0.5953
> var(AnimalesSacrif)
      Peso      Grasa
Peso  95.87757 0.397940026
Grasa 0.39794 0.002576751
    
```

Las varianzas y la covarianza de los datos generados fueron:

$$\begin{aligned}
 \sigma_{peso}^2 &= 95.877 \\
 \sigma_{grasa}^2 &= 0.0026 \\
 \sigma_{peso,grasa} &= 0.398
 \end{aligned}$$

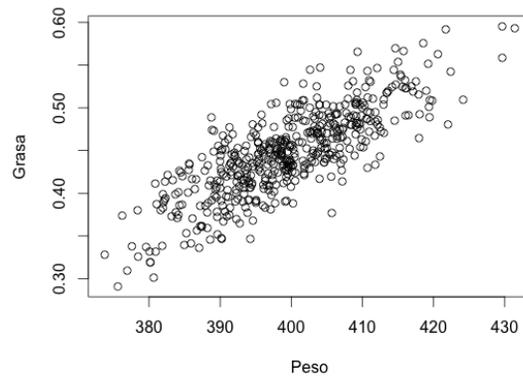
Las varianzas y la covarianza fueron similares a los parámetros de entrada. La correlación de estas dos variables fue:

$$\rho_{peso,grasa} = \frac{\sigma_{peso,grasa}}{\sqrt{\sigma_{peso}^2 * \sigma_{grasa}^2}} = \frac{0.39794}{\sqrt{95.87757 * 0.002577}} = 0.80$$

Ahora generaremos un gráfico de dispersión que relacione el peso y la grasa de los animales, el cual nos indica la alta correlación positiva entre los datos simulados de la distribución normal bivariada:



```
plot(AnimalesSacrif)
```



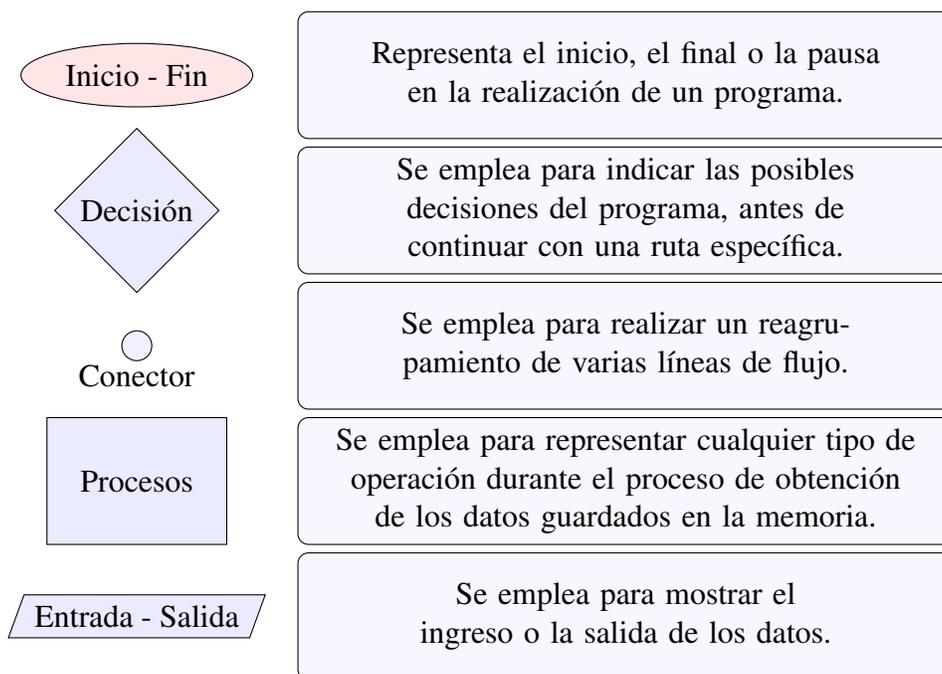
## Capítulo 3

# Algoritmos y creación de funciones en R-project

### 3.1. Algoritmos

Un algoritmo es una secuencia de acciones que se debe realizar ordenadamente para ejecutar de un programa desde el inicio hasta el fin. La representación gráfica de un algoritmo se conoce como diagrama de flujo (o flujograma), el cual incorpora símbolos que, conectados mediante líneas de flujo, indican la secuencia lógica establecida para el funcionamiento del algoritmo.

Los símbolos más utilizados en un flujograma son:



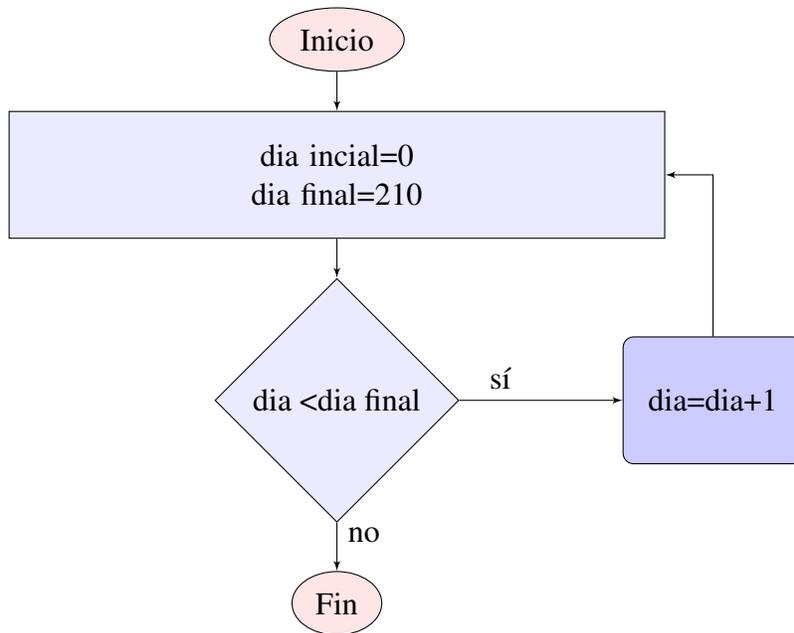
### 3.2. Comandos de R-project para algoritmos

En R-project los comandos *for*, *if*, *repeat* y *while* son los más usados para la construcción de algoritmos. Describiremos algunos ejemplos con estos comandos:

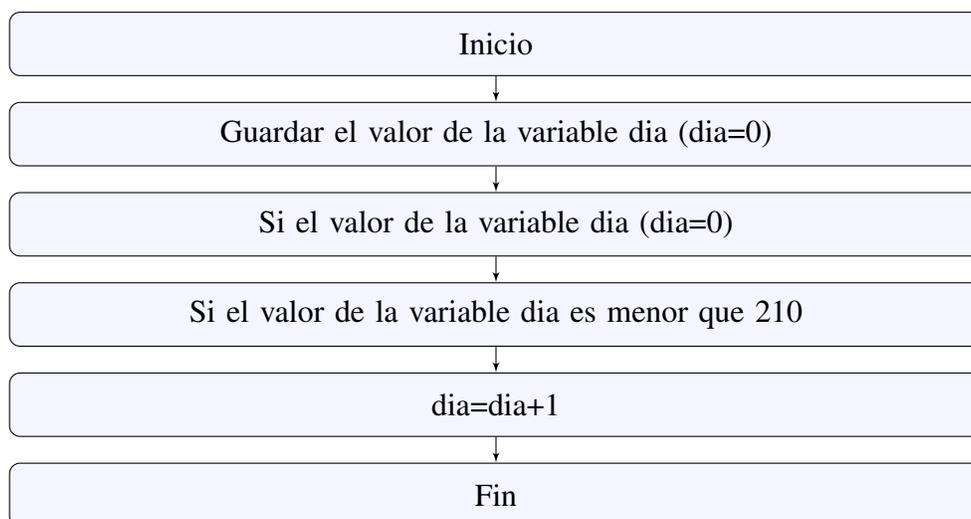
### 3.2.1. Comando *for*

El comando *for* (vector) {expresión} permite realizar una operación de forma secuencial, la cual se ejecutará un número específico de veces.

Partamos del siguiente ejemplo para crear un flujograma y usar el comando *for*. Deseamos generar un vector que tenga valores secuenciales desde 0 hasta 210, con intervalos de un día. Estos valores están relacionados con la edad del ternero desde el nacimiento (día 0) hasta el destete (210 días). El flujograma sería:



Veamos la secuencia de la programación en R-project:



Ahora generemos en R-project una variable que indica la edad del animal, desde el nacimiento ( $dia = 0$ ) hasta el destete ( $dia = 210$ ). El nombre de la variable es *dia* (sin tilde).



```

R Studio

for(dia in 0:210)
{print(dia)}

[1] 0
[1] 1
.
[1] 209
[1] 210
    
```

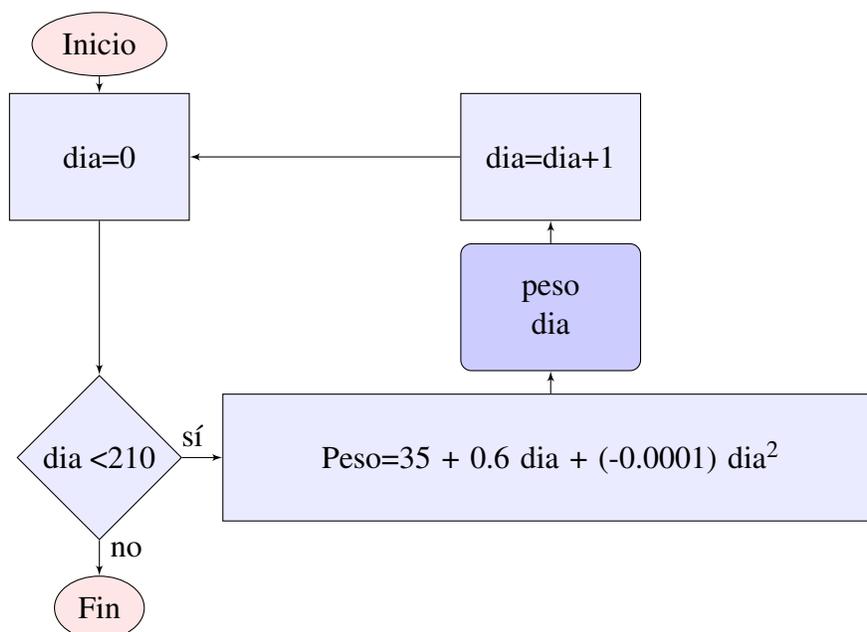
### 3.2.2. Comando *while*

El comando *while* (condición) {expresión} permite generar secuencias repetidas sin prefijar el número de veces. Realicemos un ejemplo de peso de los terneros desde el nacimiento hasta el destete, utilizando una ecuación cuadrática de crecimiento, donde el intercepto es de  $\beta_0 = 35$  kg (peso al nacimiento), un parámetro lineal  $\beta_1 = 0.600$  kg (incremento) y un parámetro cuadrático  $\beta_2 = -0.0001$ kg (desaceleración). Teniendo que  $y$  es el peso del ternero y  $x$  es la edad, la representación del modelo cuadrático es:

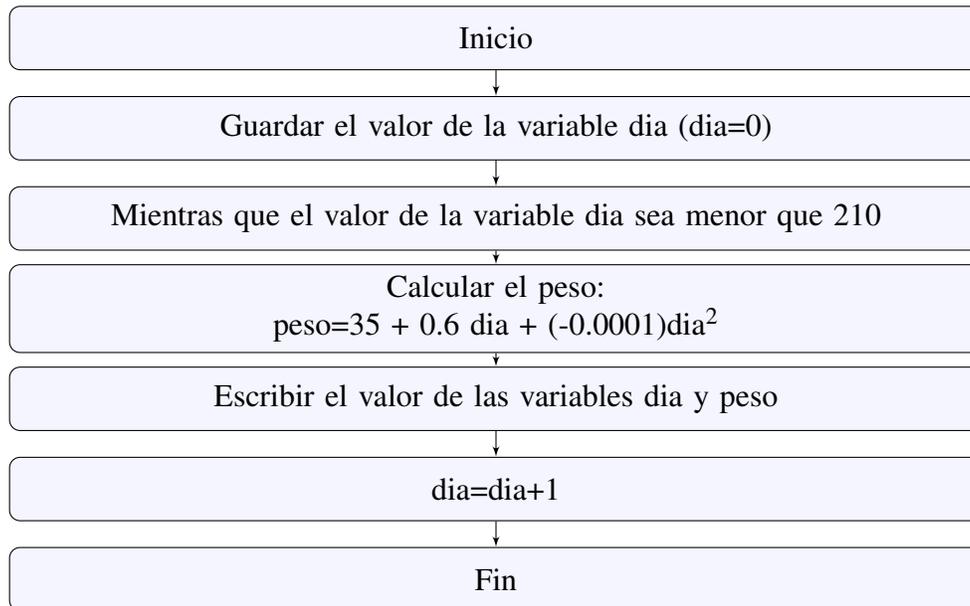
$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

$$y = 35kg + 0.600x + (-0.0001x^2)$$

Ahora generemos un algoritmo utilizando el comando *while* para el modelo anterior, que permita calcular el peso diario de los terneros durante la etapa de predestete. El diagrama de flujo es:



Veamos la secuencia de la programación en R-project:



En R-project inicialmente creamos la variable `dia = 0` (día sin tilde). Posteriormente utilizamos el comando `while` para definir la condición entre paréntesis de una secuencia de 210 días y unos argumentos entre corchetes.

Los argumentos entre corchetes están relacionados con: la ecuación requerida, la visualización en pantalla de los días con sus respectivos pesos y la generación de un nuevo día (`dia + 1`). Veamos la programación:

```

R Studio

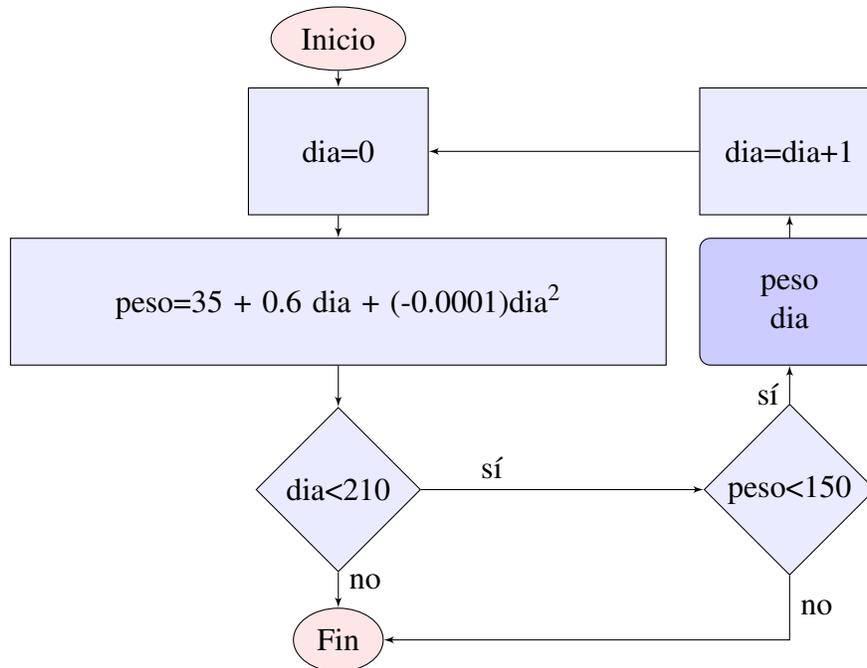
dia=0
while(dia<=210)
{
  peso=35 + (0.6*dia) + (-0.0001*dia^2)
  print(dia)
  print(peso)
  dia=dia +1
}

[1] 0
[1] 35
[1] 1
[1] 35.5999
[1] 2
[1] 36.1996
.
[1] 210
[1] 156.59
  
```

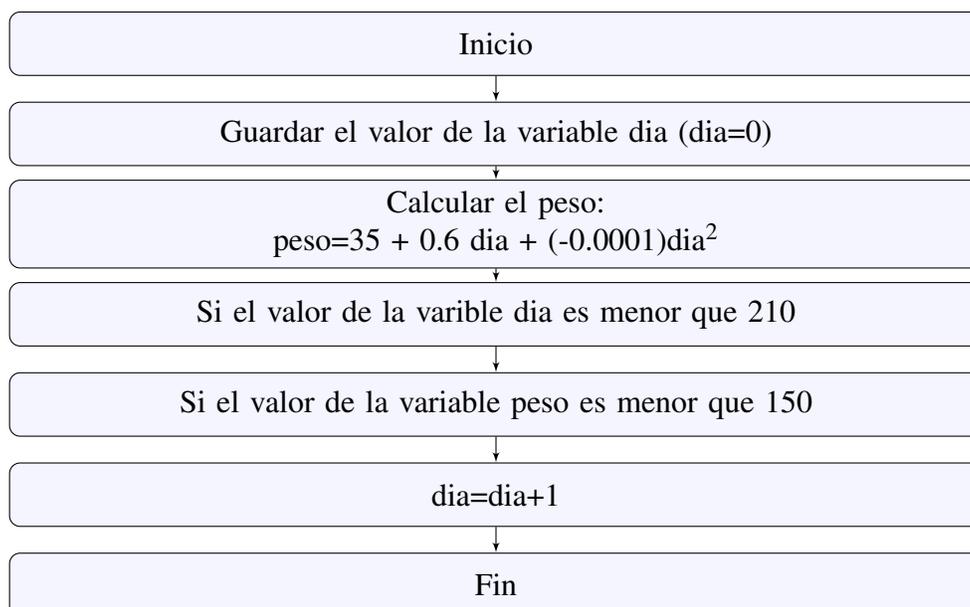
### 3.2.3. Comandos *if*, *repeat* y *break*

El comando *if* (condición) {expresión} genera condiciones para el cumplimiento de una secuencia, *repeat* (condición) {expresión} repite secuencias y el comando *break* condiciona a parar una repetición.

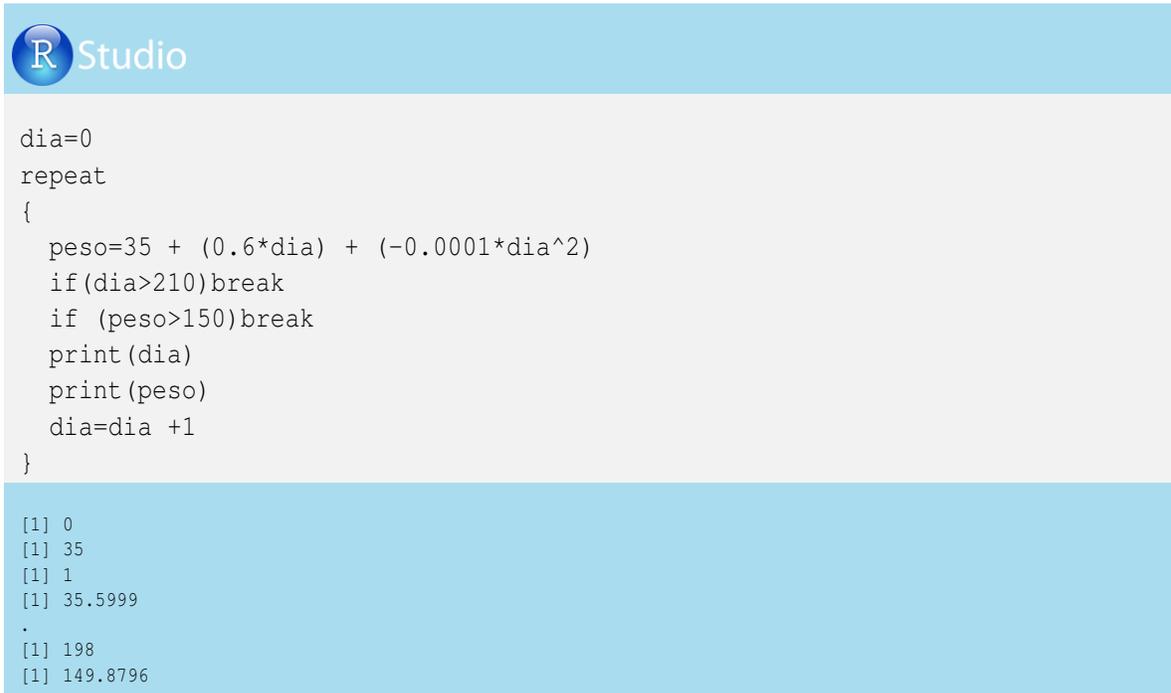
Supongamos que ahora deseamos generar la información de edad y peso de los terneros hasta que se cumplan dos condiciones. La primera es que el ternero no sobrepase los 150 kg o que no alcance los 210 días de edad. Para tal fin, utilizaremos el siguiente flujograma:



La secuencia para utilizar los comandos *repeat*, *if* y *break* en R-project es:



Para este ejemplo en R-project creamos inicialmente la variable *dia* y le asignamos el primer valor del día al nacimiento, o sea  $dia = 0$ . Para cumplir con la restricción de interés, le incluimos el comando *repeat* que permite generar una rutina de repeticiones, la cual se especifica dentro de los corchetes. Para pausar el proceso si llegamos a los 210 días o 150 kg de peso, lo realizamos con el comando *if* (si alguna condición se cumple) y el comando *break* para parar la repetición.



```

R Studio

dia=0
repeat
{
  peso=35 + (0.6*dia) + (-0.0001*dia^2)
  if(dia>210)break
  if (peso>150)break
  print(dia)
  print(peso)
  dia=dia +1
}

[1] 0
[1] 35
[1] 1
[1] 35.5999
.
[1] 198
[1] 149.8796

```

### 3.3. Creación de funciones en R-project

Los procedimientos que realiza R-project se denominan funciones (es el caso de *rnorm*, *matrix*, etc). R-project permite crear funciones de usuario, cuando se realizan tareas repetitivas para incluirlas en la programación que se está efectuando.

Una función tiene un nombre, argumentos (datos de entrada para poder ejecutar la función) y expresiones (instrucciones para la ejecución), empleando la siguiente estructura:

Nombre=*function* (argumentos) {expresiones}

Para utilizar el comando *function* es necesario respetar la estructura general de los argumentos y las expresiones y sus respectivos paréntesis y corchetes.

### 3.3.1. Ejemplo del peso de lechones

Tomemos como ejemplo la obtención del peso de lechones en la fase inicial de crecimiento según la edad, mediante la ecuación:

$$y = \beta_0 + (\beta_1x) + (\beta_2x^2)$$
$$y = 1.1 + (0.2x) + (.001x^2)$$

Donde  $y$  es el peso y  $x$  son los días de edad de los lechones durante la primera semana de vida.

En R-project creamos inicialmente la función (*function*) cuyo nombre es *EcuCua*:

```
R Studio  
EcuCua=function(x)  
{  
  y=1.1+(0.1*x) + (0.001*x^2)  
  print(y)  
}
```

Probaremos la anterior función creando un vector de edad en días, con posterior ejecución:

```
R Studio  
Edad=c(1,2,3,4,5,6,7)  
Edad  
Peso=EcuCua(Edad)  
  
> Edad  
[1] 1 2 3 4 5 6 7  
> Peso=EcuCua(Edad)  
[1] 1.301 1.504 1.709 1.916 2.125 2.336 2.549
```

### 3.3.2. Ejemplo para la obtención de un valor mínimo

A continuación crearemos una función que nos permita encontrar el valor mínimo de un vector de datos y le asignaremos el nombre *MINIMO*. En este ejemplo, la  $y$  entre paréntesis nos indica el vector al cual se le hallará el valor mínimo, y la expresión entre corchetes permite encontrar el valor mínimo del vector  $y$ .

```
R Studio  
MINIMO=function(y){min(c(y))}
```

Ahora implementaremos esta función para encontrar el peso mínimo de un cerdo en la primera semana de vida (ejemplo anterior). *MinPeso* será el dato generado con el peso mínimo, la palabra *MINIMO* permitirá llamar la función creada anteriormente con ese nombre y entre paréntesis indicamos el nombre del vector que tiene los pesos del ejercicio anterior, que se llama *Peso*. El código de R-project y el resultado son:



```

R Studio

MinPeso=MINIMO(Peso)
MinPeso

[1] 1.301

```

Podemos cambiar dentro de la función el comando *min*, por los comandos *max* para generar el valor máximo, *mean* para generar el valor medio, etc, dependiendo de las necesidades.

### 3.3.3. Ejemplo de una función que relaciona dos variables

Generemos una función que relacione las variables  $x$  y  $y$ , por medio de la siguiente expresión:

$$\frac{x * y}{100}$$

En R-project la función que obedece a la anterior expresión sería:



```

R Studio

Relac=function(x,y)
{
  (x*y)/100
}

```

En el siguiente ejercicio se tiene la variable kilos de leche ( $x$ ) y la variable porcentaje grasa ( $y$ ) de hatos lecheros y deseamos saber cuántos kilos de grasa se produce en los hatos analizados. En R-project generaremos un vector con la producción de leche en kilos por hato (*LecheHato*) y un vector con sus respectivos porcentajes de grasa (*GrasaHato*). A esta función la llamamos *Relac* y entre paréntesis indicamos los nombres de los vectores que contienen los valores de  $x$  y  $y$ . El vector con los kilos de grasa se llama *Kilosgrasa*:



```
LecheHato= c(1550, 300, 124)
GrasaHato= c(3.2, 4.1, 4)
Kilosgrasa= Relac (LecheHato, GrasaHato)
Kilosgrasa
```

```
1] 49.60 12.30 4.96
```

### 3.3.4. Ejemplo de una función con los comandos *for* y *seq*

El siguiente ejemplo está relacionado con el algoritmo desarrollado en la sección 3.2.1, para crear una función que genere la edad de un animal, desde su nacimiento hasta un día determinado ( $x$ ) asignado por el usuario, utilizando el comando *for*.



```
edadanimal=function(x) for(dia in 1:x)
{print(dia)}
```

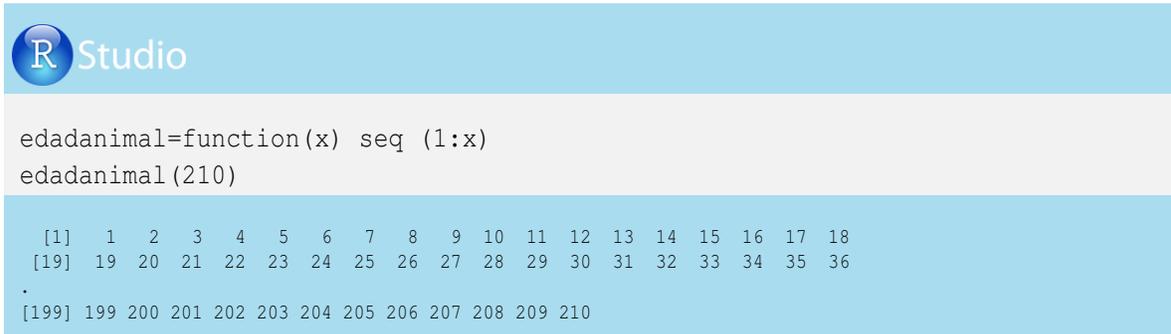
Emplearemos en R-project la función creada para generar un vector con los primeros 210 días de edad de un ternero. Para esto, llamaremos la función ya creada (*edadanimal*) y le escribimos entre paréntesis el número de datos generados, que en este caso es (210), de la siguiente forma:



```
edadanimal(210)
```

```
[1] 1
[1] 2
[1] 3
.
[1] 209
[1] 210
```

También podemos generar la misma secuencia de datos con el comando *seq*. Veamos la programación en R-project y los resultados con el ejemplo anterior:



```

R Studio

edadanimal=function(x) seq (1:x)
edadanimal(210)

 [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
.
[199] 199 200 201 202 203 204 205 206 207 208 209 210

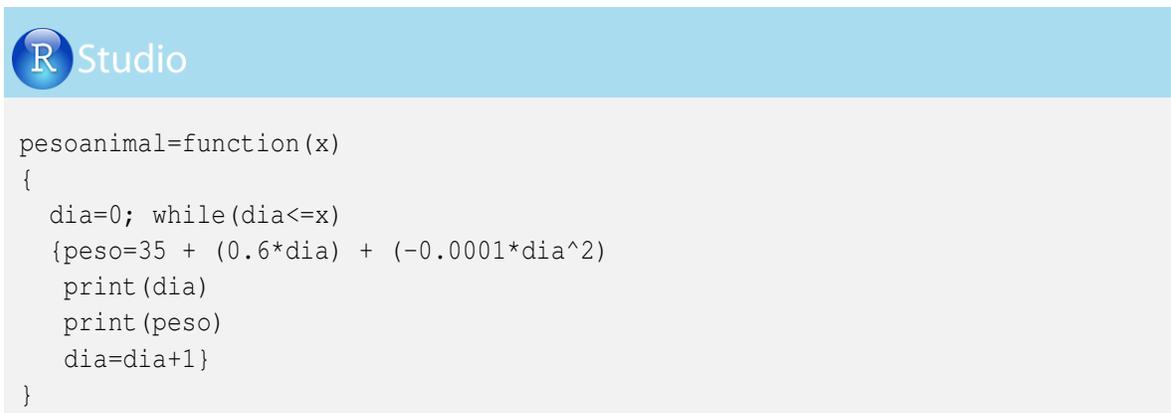
```

### 3.3.5. Ejemplo de una función con el comando *while*

A continuación se presenta la función que usa el comando *while*, que permite generar el peso del ternero del ejemplo de la sección 3.2.2, mediante la ecuación:

$$\text{peso} = 35 + (0.6 * \text{dia}) + (-0.0001 * \text{dia}^2)$$

La función creada con el comando *while* en R-project es:



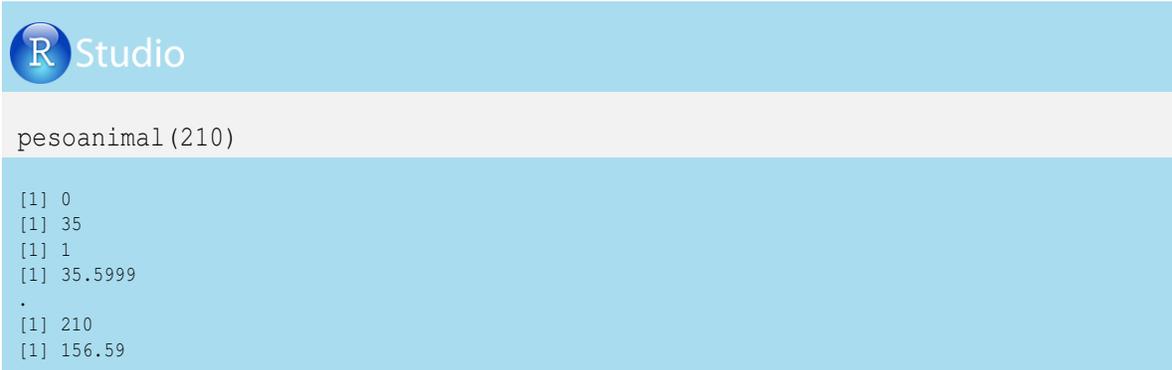
```

R Studio

pesoanimal=function(x)
{
  dia=0; while(dia<=x)
  {peso=35 + (0.6*dia) + (-0.0001*dia^2)
  print(dia)
  print(peso)
  dia=dia+1}
}

```

Apliquemos la función que llamamos *pesoanimal* para obtener el peso de los animales desde el nacimiento hasta los 210 días:



```
R Studio
pesoanimal(210)
[1] 0
[1] 35
[1] 1
[1] 35.5999
.
[1] 210
[1] 156.59
```

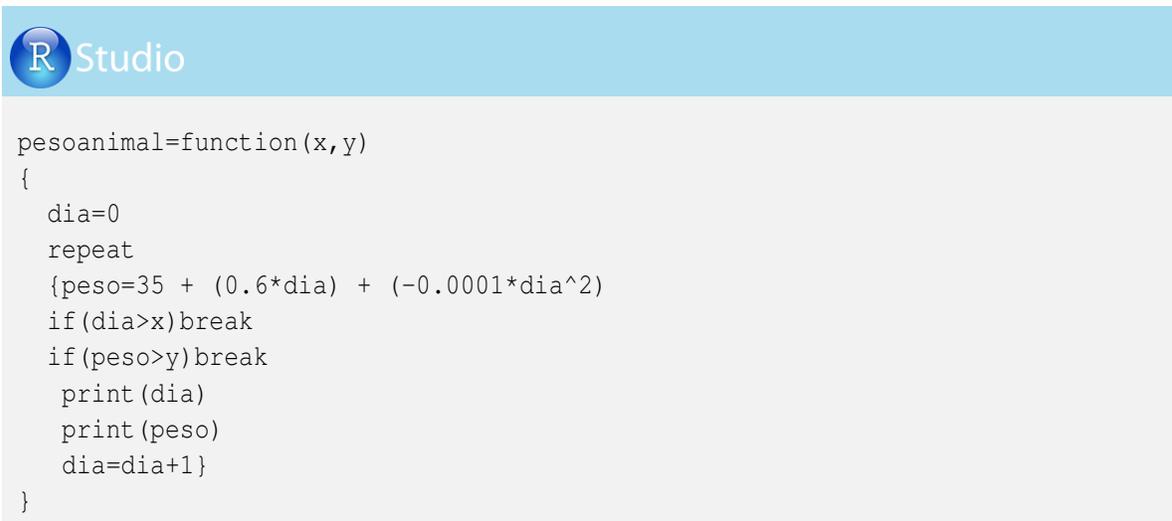
Vemos que se generó un vector iniciando con el día 0, seguido de su respectivo peso en kilos (35), del día 1 con el peso 35.599 hasta el día 210 con peso 156.59.

### 3.3.6. Ejemplo de una función con los comandos *repeat*, *if* y *break*

Ahora se presenta una función creada con el comando *repeat*, acompañado de *if* y *break* para generar los pesos ( $x$ ) de animales según la edad ( $y$ ), cuando se cumple alguna o las dos condiciones del ejemplo de la sección 3.2.2, las cuales son:

$$\begin{aligned}x &\leq 150 \\ y &\leq 210\end{aligned}$$

La programación en R-project es:



```
R Studio
pesoanimal=function(x,y)
{
  dia=0
  repeat
  {peso=35 + (0.6*dia) + (-0.0001*dia^2)
  if(dia>x)break
  if(peso>y)break
  print(dia)
  print(peso)
  dia=dia+1}
}
```

Emplearemos la función anterior para generar la información de edad y peso del ternero, teniendo como condición que sea menor o igual que 210 días o tener 150 kg o menos:



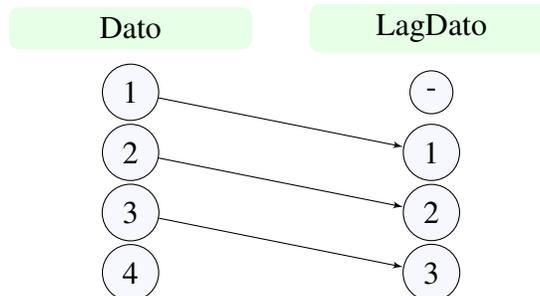
```

R Studio
pesoanimal (210, 150)
[1] 0
[1] 35
[1] 1
[1] 35.5999
.
[1] 198
[1] 149.8796

```

### 3.3.7. Creación de una función *Lag*

En muchas oportunidades es necesario realizar operaciones entre filas de una misma columna (mayores detalles en la sección 4.5.3). Una de las formas para realizar este tipo de operaciones, es crear una nueva variable corriendo cuantos espacios o filas sean necesarias. Por ejemplo, a partir de la variable *Dato*, generaremos una variable *LagDato*, con la siguiente caracterización:



Generemos una función a la que llamaremos *FuncLag* en R-project.

```
FuncLag=function (Lagx) {expresión }
```

La expresión estaría dada por:

La creación de un vector (lo llamaremos *MenosUltimo*, el cual le quitará a la columna *Lagx* el último valor), así:

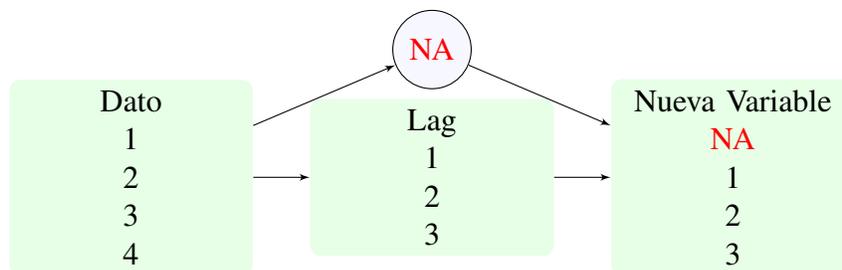
```
MenosUltimo= matrix([1:(length (Lagx)-1]
```

Posteriormente creamos un vector donde el primer valor es *NA* y le pegamos el vector *MenosUltimo*, con el comando *rbind(NA,MenosUltimo)*.

Veamos en R-project la estructura para crear la función *FuncLag*:

```
R Studio  
  
FuncLag=function(LAG)  
{  
  MenosUltimo=matrix(LAG[1:(length(LAG)-1)]);  
  rbind(NA,MenosUltimo)  
}
```

Apliquemos la función *FuncLag* a un vector de cuatro datos llamado *Dato*:



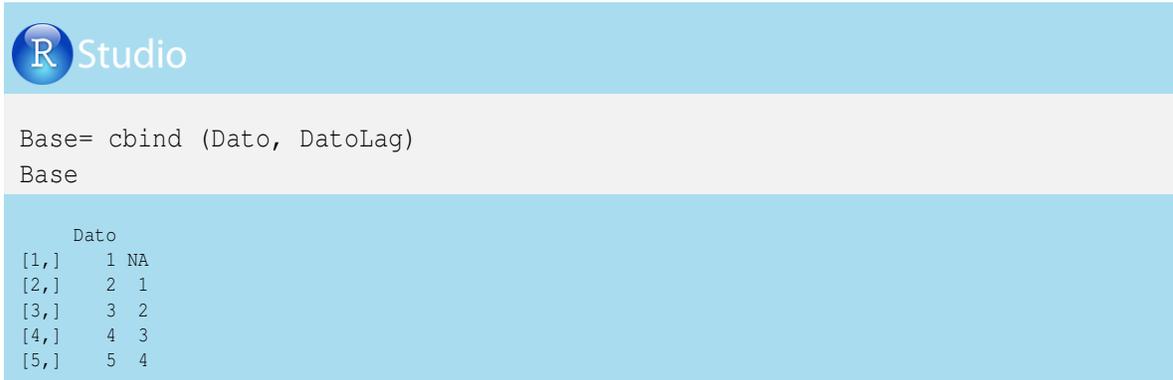
Generemos inicialmente el vector *Dato* en R-project:

```
R Studio  
  
Dato=c(1,2,3,4,5)  
Dato  
  
[1] 1 2 3 4 5
```

Ahora apliquemos la función *FuncLag* al vector *Dato* y creamos el vector *DatoLag*:

```
R Studio  
  
DatoLag=FuncLag(Dato)  
DatoLag  
  
  [,1]  
[1,] NA  
[2,]  1  
[3,]  2  
[4,]  3  
[5,]  4
```

Vamos a crear una matriz de dos columnas con información del vector *Dato* y el vector *DatoLag*, mediante el comando *cbind*:



```
R Studio  
Base= cbind (Dato, DatoLag)  
Base  
      Dato  
[1,]  1 NA  
[2,]  2  1  
[3,]  3  2  
[4,]  4  3  
[5,]  5  4
```

Podemos apreciar que el primer dato de la primera columna (1) quedó relacionado con el *NA* de la segunda columna y que al dato 5 de la primera columna le corresponde el dato 4 de la segunda columna.

## Capítulo 4

### Creación y montaje de hojas de datos

#### 4.1. Generalidades

En el capítulo 1 vimos la creación de vectores y matrices; ahora nos preparamos para generar conjuntos de datos de tipo numérico o alfanumérico estructurados por filas (casos) y columnas (variables), a los que denominaremos hojas de datos o *dataframe*.

Inicialmente generaremos hojas de datos desde matrices y vectores utilizando la función *data.frame*, y en secciones posteriores mostraremos la importación de hojas de datos con diversos formatos y la manipulación de datos con ejemplos en producción animal.

#### 4.2. Hoja de datos a partir de matrices y vectores

Como ejemplo, consideremos los pesos (kg) de tres toros:

Toro	Peso
MONKY	445
LITIO	567
MORENO	643

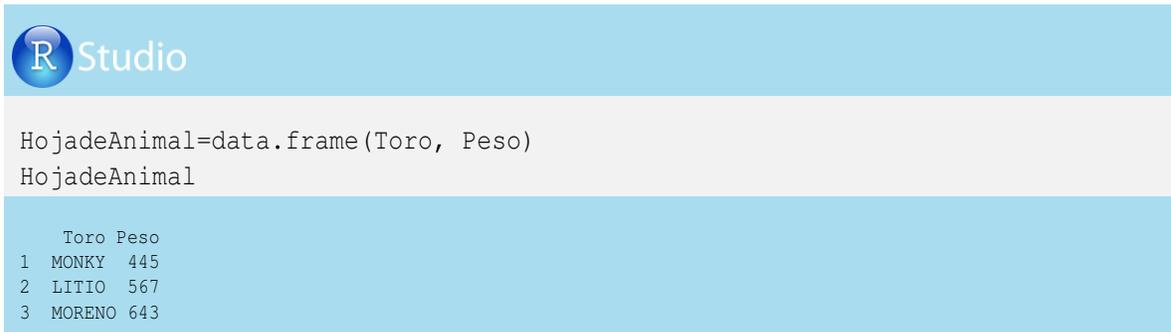
Para la construcción de una hoja de datos en R-project, primero crearemos un vector alfanumérico con los nombres de los toros (*Toro*) y otro vector numérico con los pesos de los toros (*Peso*):

```
R Studio

Toro=c("MONKY", "LITIO", "MORENO")
Toro
Peso=c(445, 567, 643)
Peso

> Toro
[1] "MONKY" "LITIO" "MORENO"
> Peso
[1] 445 567 643
```

Ahora crearemos la hoja de datos a la que llamaremos *HojadeAnimal*, que contiene los vectores *Toro* y *Peso*, mediante el comando *data.frame*:



```
R Studio

HojadeAnimal=data.frame(Toro, Peso)
HojadeAnimal

  Toro Peso
1 MONKY 445
2 LITIO 567
3 MORENO 643
```

Ahora crearemos la matriz que contiene el número de hijos de cada toro en tres fincas, teniendo en cuenta la siguiente estructura:

Toro	Montana	Vegas	Progreso
MONKY	3	23	5
LITIO	5	62	17
MORENO	0	12	3

En R-project sería:

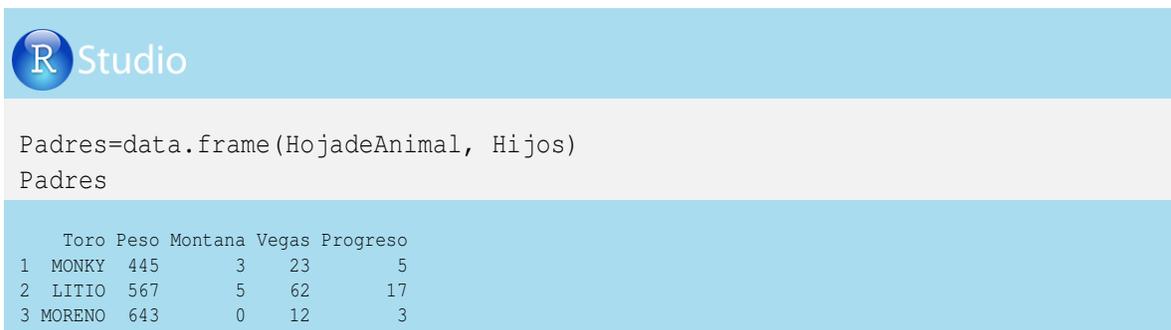


```
R Studio

Hijos=matrix(nrow=3,ncol=3, data=
  c( 3,23,5,5,62,17,0,12,3),byrow=TRUE)
colnames(Hijos)=(c("Montana", "Vegas", "Progreso"))
Hijos

  Montana Vegas Progreso
[1,]      3    23      5
[2,]      5    62     17
[3,]      0    12      3
```

Juntamos la hoja de datos *HojadeAnimal* con la matriz *Hijos*, para crear la hoja de datos *Padres*:



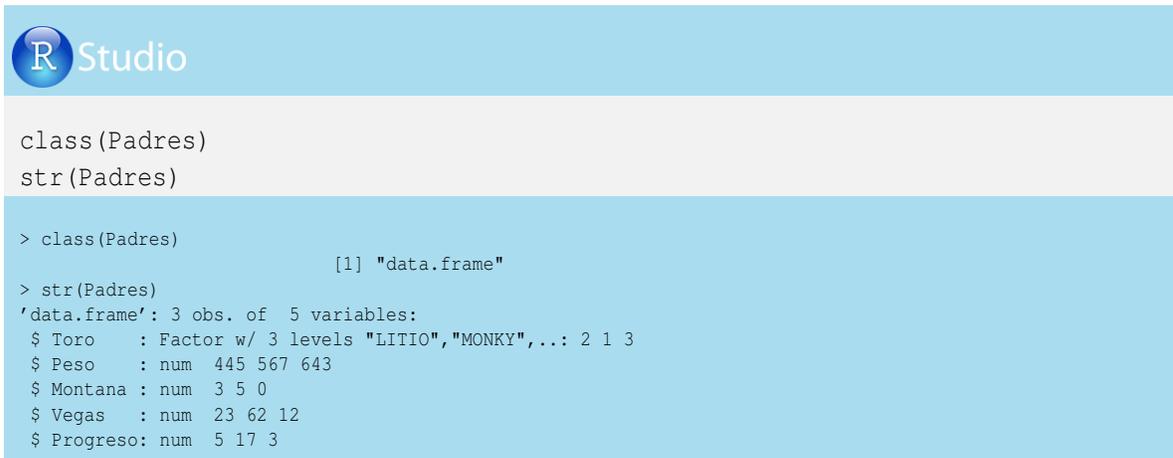
```
R Studio

Padres=data.frame(HojadeAnimal, Hijos)
Padres

  Toro Peso Montana Vegas Progreso
1 MONKY 445      3    23      5
2 LITIO 567      5    62     17
3 MORENO 643      0    12      3
```

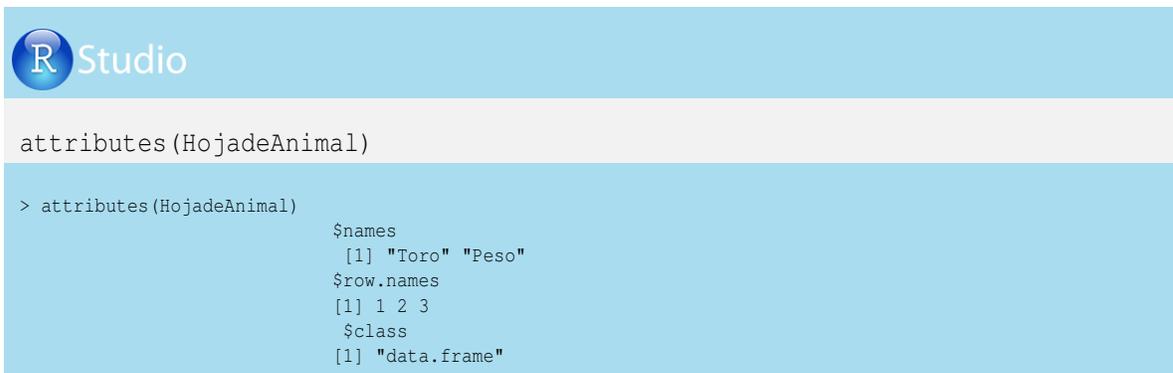
### 4.3. Estructura de una hoja de datos

Con el propósito de manipular adecuadamente una hoja de datos, debemos conocer su composición y las características de sus componentes. El comando *class* nos muestra si es verdaderamente una hoja de datos, y el comando *str* indica el número de observaciones y las especificaciones de cada variable con los datos respectivos. Comprobemos la estructura de la hoja de datos *Padres*:



```
R Studio  
class(Padres)  
str(Padres)  
  
> class(Padres)  
[1] "data.frame"  
  
> str(Padres)  
'data.frame': 3 obs. of 5 variables:  
 $ Toro   : Factor w/ 3 levels "LITIO","MONKY",...: 2 1 3  
 $ Peso   : num  445 567 643  
 $ Montana : num  3 5 0  
 $ Vegas  : num  23 62 12  
 $ Progreso: num  5 17 3
```

El comando *attributes* generamos la información del tipo de variables del *data.frame*:



```
R Studio  
attributes(HojadeAnimal)  
  
> attributes(HojadeAnimal)  
$names  
 [1] "Toro" "Peso"  
$row.names  
 [1] 1 2 3  
$class  
 [1] "data.frame"
```

El comando *summary* muestra un resumen de los datos indicando la media, la mediana, los cuartiles y los valores mínimo y máximo en el caso de variables numéricas, y frecuencias en el caso de variables alfanuméricas.

```
summary(Padres)

summary(Padres)
  Toro      Peso      Montana      Vegas      Progreso
LITIO :1  Min.    :445.0    Min.    :0.000    Min.    :12.00   Min.    : 3.000
MONKY :1  1st Qu.:506.0    1st Qu.:1.500    1st Qu.:17.50   1st Qu.: 4.000
MORENO:1  Median :567.0    Median :3.000    Median :23.00   Median : 5.000
        Mean  :551.7    Mean   :2.667    Mean   :32.33   Mean   : 8.333
        3rd Qu.:605.0    3rd Qu.:4.000    3rd Qu.:42.50   3rd Qu.:11.000
        Max.  :643.0    Max.   :5.000    Max.   :62.00   Max.   :17.000
```

#### 4.4. Importación de hojas de datos

En R-project existen diferentes formas de importar datos. Algunos de los comandos más empleados para la importación de datos son: *read.table*, *read.csv*, *read.xls*, *xlConnect* y *RODBC* o utilizando librerías como la *foreign* (R Core Team, 2012a) que permite importar hojas de datos de formatos especiales como **Epinfo**, **Dbase**, **Minitab**, **SPSS**, **SAS**, **Stata**, entre otros.

Nuestra recomendación es utilizar formatos planos separados por tabulaciones o espacios (*.txt*) y por comas (*.csv*) o archivos en Excel (*.xls* o *.xlsx*). Por eso, en este capítulo trabajaremos la importación de este tipo de archivos.

El argumento inicial de cada comando de importación es la ruta del computador donde se encuentra el archivo, con su respectivo nombre y extensión. También se requieren otros argumentos para importar adecuadamente la información de las hojas de datos. Algunos de estos argumentos son:

Argumento	Especificaciones
<i>header</i>	Utilice = <i>TRUE</i> cuando en la primera fila de la base a importar están los nombres de las variables
<i>sep</i>	Hay que indicar si las columnas están separadas por espacio, doble espacio, tabulaciones u otros signos.
<i>dec</i>	Especifica el número de decimales que tiene cada variable.
<i>row.names</i>	Permite poner los nombres a las variables. Si no desea incluirlos, utilice <i>NULL</i> , y se generarán nombres automáticamente.
<i>col.names</i>	Si se desea poner nombres. Si se usa la <i>V</i> , por defecto se incluye una columna numerada.
<i>as.is</i>	Con este argumento y el nombre de las variables, se mantiene el formato original de las variables, ya que por defecto <i>read.table</i> deja todas las variables como caracteres.
<i>na.strings</i>	Se especifica que la hoja de datos tiene valores en blanco <i>NA</i>
<i>nrows</i>	Si desea especificar el número de filas que se leerán
<i>check.names</i>	Permite verificar si existen columnas con nombres repetidos

#### 4.4.1. Archivos *.txt*

Para importar datos separados por tabulaciones (*.txt*) podemos utilizar el comando *read.table*, seguido de un paréntesis donde se especifica la ruta donde está el archivo y se incluyen otras especificaciones o argumentos de interés.

Realicemos un ejemplo de importación de hojas de datos separado por tabulaciones. Para esto, originemos una base de datos en Excel. La llamaremos **Pesaje2005.excel** y posteriormente la guardamos en formato “*.txt*”, el cual tendrá las columnas relacionadas con el nombre de los terneros, el sexo, las fechas de pesaje y su respectivo peso:

Pesaje2005.xlsx									
	A	B	C	D	E	F	G	H	I
1	ANIMAL	SEXO	PESAJE	PESO					
2	44/05	H	2005/08/15	88,40					
3	44/05	H	2005/08/25	88,30					
4	44/05	H	2005/11/30	144,00					
5	44/05	H	2005/12/23	150,00					
6	45/05	M	2005/08/25	30,20					
7	45/05	M	2005/11/30	84,10					
8	45/05	M	2005/12/23	96,00					
9	45/05	M	2005/01/15	100,88					
10	45/05	M	2005/01/30	119,00					
11	49/05	H	2005/09/10	30,00					
12	49/05	H	2005/11/30	77,00					
13	49/05	H	2005/12/23	88,00					
14									

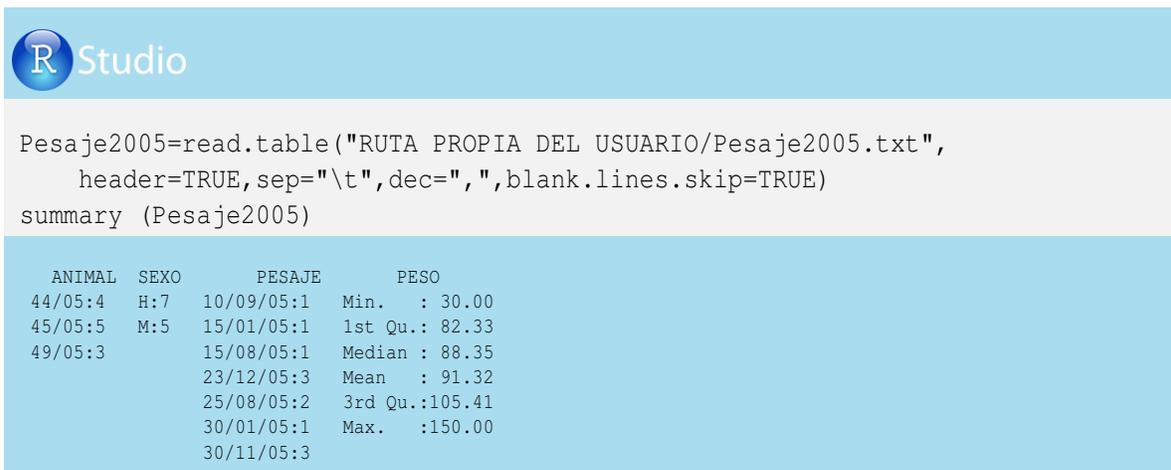
En Excel se da la opción “guardar como” y se busca el formato *.txt*, y lo visualizamos en un editor de texto para observar si su estructura está correcta:

Editor Pesaje2005.txt				
ANIMAL	SEXO	PESAJE	PESO	
44/05	H	2005/08/15	88,40	
44/05	H	2005/08/25	88,30	
44/05	H	2005/11/30	144,00	
44/05	H	2005/12/23	150,00	
45/05	M	2005/08/25	30,20	
45/05	M	2005/11/30	84,10	
45/05	M	2005/12/23	96,00	
45/05	M	2005/01/15	100,88	
45/05	M	2005/01/30	119,00	
49/05	H	2005/09/10	30,00	
49/05	H	2005/11/30	77,00	
49/05	H	2005/12/23	88,00	

Para importar la hoja de datos *Pesaje2005.txt* utilizamos el comando *read.table* indicando inicialmente la ruta específica para cada usuario, seguido por el nombre del archivo *Pesaje2005.txt*.

También incluimos los argumentos: *header* que permite indicar si la primera fila contiene los nombres de las variables, *sep* = “\t” para especificar que el archivo está separado por tabulaciones, *dec* = “,” para indicar que los decimales en las variables numéricas están separados por coma, y *blank.lines.skip* para eliminar filas que estén vacías.

Veamos cómo se importa en R-project una hoja de datos con el comando *read.table*:



```

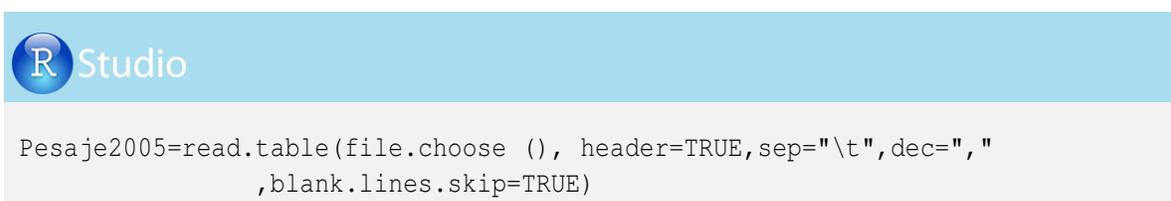
R Studio

Pesaje2005=read.table("RUTA PROPIA DEL USUARIO/Pesaje2005.txt",
  header=TRUE, sep="\t", dec=",", blank.lines.skip=TRUE)
summary (Pesaje2005)

```

ANIMAL	SEXO	PESAJE	PESO
44/05:4	H:7	10/09/05:1	Min. : 30.00
45/05:5	M:5	15/01/05:1	1st Qu.: 82.33
49/05:3		15/08/05:1	Median : 88.35
		23/12/05:3	Mean : 91.32
		25/08/05:2	3rd Qu.:105.41
		30/01/05:1	Max. :150.00
		30/11/05:3	

En caso de no conocer la ruta donde está el archivo, se puede utilizar el argumento *file.choose()*, el cual busca en su computador la localización del archivo deseado para importación, así:



```

R Studio

Pesaje2005=read.table(file.choose (), header=TRUE, sep="\t", dec=",",
  ,blank.lines.skip=TRUE)

```

Si utiliza *R-Studio* puede realizar la importación desde la ventana de *Workspace*.

Con la información suministrada por el comando *summary* verificamos si el proceso de importación de la hoja de datos fue adecuada. Encontramos que hay tres animales (con 3 o más datos), dos sexos, pesajes realizados durante el 2005 y el peso, que está entre 30 kg y 150 kg con una media de 91.32 kg.

En el caso de variables tipo fecha (como en la variable *PESAJE*) es necesario modificar su estructura para que R-project pueda realizar cálculos entre fechas. Para esto utilizamos el comando *as.Date*.

Veamos en R-project el uso de *as.Date* y la estructura de la hoja de datos con el comando *str* antes y después de la modificación de la variable *PESAJE*.



```
str (Pesaje2005)
Pesaje2005$PESAJE=(as.Date (Pesaje2005$PESAJE, "%d/%m/%y"))
str (Pesaje2005)

> str (Pesaje2005)
' data.frame': 12 obs. of 4 variables:
 $ ANIMAL: Factor w/ 3 levels "44/05","45/05",...: 1 1 1 1 2 2 2 2 3 ...
 $ SEXO : Factor w/ 2 levels "H","M": 1 1 1 1 2 2 2 2 1 ...
 $ PESAJE: Factor w/ 7 levels "10/09/05","15/01/05",...: 3 5 7 4 5 7 4 2 6 1 ...
 $ PESO : num 88.4 88.3 144 150 30.2 ...

> str (Pesaje2005)
' data.frame': 12 obs. of 4 variables:
 $ ANIMAL: Factor w/ 3 levels "44/05","45/05",...: 1 1 1 1 2 2 2 2 3 ...
 $ SEXO : Factor w/ 2 levels "H","M": 1 1 1 1 2 2 2 2 1 ...
 $ PESAJE: Date, format: "2005-08-15" "2005-08-25" "2005-11-30" ...
 $ PESO : num 88.4 88.3 144 150 30.2 ...
```

Si la variable original (en formato *Factor*) contiene la información del año con dos dígitos (por ejemplo: 02/12/00), se utiliza el comando *as.Date* con el argumento “%d/%m/%y”(y minúscula). Pero si el año contiene cuatro dígitos (por ejemplo: 02/12/2000) se utiliza el argumento “%d/%m/%Y”(Y mayúscula). Si la variable original tiene el mes en letras (*jan*, *feb*, *nov*, etc), por ejemplo *2dec00*, el argumento es “%d/%b/%y”.

Para exportar o guardar la hoja de datos como archivo *txt*, utilizamos el comando *write.table*, especificándole el nombre de la hoja de datos, la ruta y el nombre del archivo a guardar con la extensión *txt* y otros argumentos de utilidad, como es el caso de dejar en la primera fila los nombres de las columnas (*row.names = TRUE*) y la separación de los decimales por medio de un punto (*dec = “.”*):



```
write.table (Pesaje2005, file="RUTA PROPIA DEL USUARIO/pesaje.txt",
            col.names=TRUE, na="NA",dec=".")
```

#### 4.4.2. Archivos *csv*

Ahora importemos una hoja de datos en formato CSV, separado por punto y coma. Primero crearemos una hoja de datos en Excel con los pesajes del año 2006, con el nombre *Pesaje2006.xlsx* y la guardamos en formato *csv*, con el nombre *Pesaje2006.csv*.

Pesaje2006.xlsx									
	A	B	C	D	E	F	G	H	I
1	ANIMAL	SEXO	PESAJE	PESO					
2	45/05	M	15/012/006	100					
3	45/05	M	30/01/2006	119					
4									
5									

En la opción “guardar como” de Excel, escogemos el formato CSV, de modo que obtenemos un archivo *Pesaje2006.csv*, el cual se presenta de la siguiente forma:

Editor Pesaje2006.csv
ANIMAL;SEXO;PESAJE;PESO
45/05;M;15/012/006;100
45/05;M;30/01/2006;119

El comando *read.csv* de R-project permite importar la hoja de datos de interés indicando la ruta específica, el nombre y la extensión del archivo y otros argumentos que se requieran. Ahora importemos la hoja de datos *Pesaje2006.csv* y modifiquemos el formato de las variables tipo fecha:

```

R Studio

Pesaje2006=read.csv("RUTA DEL USUARIO/Pesaje2006.csv",
  header=TRUE, sep=";", dec=",")
Pesaje2006$PESAJE=(as.Date(Pesaje2006$PESAJE,"%d/%m/%y"))
summary(Pesaje2006)

  ANIMAL SEXO PESAJE PESO
45/05:2 M:2 Min. :2006-01-15 Min. :100.0
1st Qu.:2006-01-18 1st Qu.:104.8
Median :2006-01-22 Median :109.5
Mean :2006-01-22 Mean :109.5
3rd Qu.:2006-01-26 3rd Qu.:114.2
Max. :2006-01-30 Max. :119.0
    
```

Utilizamos el comando *write.csv* para guardar la hoja de datos, indicando la ruta y el nombre del archivo a guardar con la extensión *csv*:

```
write.csv (Pesaje2005, file="RUTA PROPIA DEL USUARIO/pesaje.csv")
```

#### 4.4.3. Archivos de Excel

Hay varias formas de importar archivos en formato Excel, como es el caso de los comandos *xlConnect*, *RODBC* y *read.xls*, o directamente de la opción exportar datos de la ventana *workspace* si se está trabajando en *R-Studio*. Aunque es fácil importar datos, hay que tener cuidado con la versión de Excel que se esté utilizando y la de la plataforma de R-project (Windows, Mac y Linux), porque se pueden presentar problemas en la importación. Nuestra sugerencia para el lector es trabajar la importación de archivos de datos en formatos *txt* y *csv*.

A continuación haremos la importación de datos utilizando la librería *RODBC*. Para esto construiremos el archivo *madres.xlsx* que contiene la hoja de datos *informaciones* con los nombres de las madres y las fechas de nacimiento de los animales que aparecen en las hojas de datos anteriores (*Pesaje2005.txt* y *Pesaje2006.csv*):

MADRES.xlsx									
	A	B	C	D	E	F	G	H	I
1	ANIMAL	MADRE	NACIMIENTO						
2	44/05	37/08	12/05/05						
3	45/05	23/00	25/08/05						
4	49/05	115/08	10/09/05						
5	50/06	37/08	10/07/05						
6									

Instalamos la librería *RODBC* (ODBC Database Access de Ripley, s.f), que nos permite extraer una base de datos que está en Excel 2007 y que tiene varias hojas. Del archivo *MADRES.xlsx* importamos la hoja llamada *informaciones*. Advertimos que es muy probable que esta importación no funcione por las versiones de Excel y de R-project. En este caso, se recomienda consultar la ayuda de R-project que esté utilizando o convertir la base a *csv*:



```
install.packages("RODBC")
library(RODBC)
Madre=odbcConnectExcel2007("RUTA DE USUARIO/MADRES.xlsx")
sqlTables(Madre)
sqlFetch(Madre,"informaciones")
summary (Madre)
```

ANIMAL	MADRE	NACIMIENTO
44/05:1	115/08:1	Min. :2005-05-12
45/05:1	23/00 :1	1st Qu.:2005-07-29
49/05:1	37/08 :2	Median :2005-09-02
50/06:1		Mean :2005-10-21
		3rd Qu.:2005-11-24
		Max. :2006-07-10

## 4.5. Manipulación de bases de datos

En muchas oportunidades es necesario unir hojas de datos, crear variables a partir de las existentes y manipular registros que cumplan (o no cumplan) con algunas condiciones.

### 4.5.1. Unión de hojas de datos

En muchas ocasiones es necesario unir hojas de datos que comparten alguna información. En R-project existen varias alternativas para unir hojas de datos dependiendo del número de variables comunes entre ellas. A continuación presentaremos una forma de unir las hojas de pesaje de animales del año 2005 y 2006 (*Pesaje2005*, *Pesaje2006*) y la hoja de información de madres (*Madre*).

Primero combinemos las hojas que contienen los pesajes, para lo cual es necesario confirmar si las hojas de datos tienen las mismas variables. Para ello utilizaremos el comando *colnames*:



```
colnames (Pesaje2005)
colnames (Pesaje2006)
```

```
> colnames (Pesaje2005)
[1] "ANIMAL" "SEXO" "PESAJE" "PESO"
> colnames (Pesaje2006)
[1] "ANIMAL" "SEXO" "PESAJE" "PESO"
```

Se puede apreciar que las dos hojas de datos tienen exactamente las mismas variables (*ANIMAL*, *SEXO*, *PESAJE* y *PESO*). Además es necesario verificar si las variables tienen el mismo formato, con el comando *str*. En caso contrario, hay que estandarizar el mismo formato utilizando los comandos *as.Date*, *as.numeric* o *as.factor*.

Juntemos las hojas de datos con el comando *rbind*, para crear una nueva hoja de datos llamada *Pesaje*.

```
R Studio

Pesaje= rbind (Pesaje2005, Pesaje2006)
Pesaje
```

	ANIMAL	SEXO	PESAJE	PESO
1	44/05	H	2005-08-15	88.40
2	44/05	H	2005-08-25	88.30
.				
14	45/05	M	2006-01-30	119.00

Apreciamos que la hoja de datos *Pesaje* tiene 14 filas (registros de pesaje) producto de la unión de la hoja de datos *Pesaje2005* que tiene 12 registros y la hoja de datos *Pesaje2006* que tiene 2 registros.

Ahora juntemos la hoja de datos *Pesaje* con la información de la hoja de datos *Madre*. Las dos hojas tienen en común la variable *ANIMAL*. Veamos, con el comando *colnames*, las variables de estas dos hojas:

```
R Studio

colnames (Madre)
colnames (Pesaje)
```

```
> colnames (Madre)
[1] "ANIMAL"      "MADRE"      "NACIMIENTO"
> colnames (Pesaje)
[1] "ANIMAL" "SEXO"      "PESAJE" "PESO"
```

Como sabemos que existen variables en común en las dos hojas, utilizaremos el comando *merge* para juntarlas. Al incluir los argumentos *all.x = TRUE* (relacionado con la primera hoja) y *all.y = TRUE* (relacionado con la segunda hoja) se juntarán las dos hojas sin importar si el animal no aparece en alguna de las dos. En nuestro caso, la hoja de datos *Pesaje* tiene pesos de tres animales que no aparecen en la hoja de datos *Madres* (el animal 50/06 no tiene información de pesaje), por consiguiente generará los correspondientes *NAs* en las variables *PESAJE* y *PESO*.



```
MadresCrias=merge (Madre, Pesaje, all.x=TRUE, all.y=TRUE)
colnames (MadresCrias)
MadresCrias
```

```
> colnames (MadresCrias)
[1] "ANIMAL"      "MADRE"       "NACIMIENTO" "SEXO"        "PESAJE"      "PESO"
> MadresCrias
  ANIMAL  MADRE  NACIMIENTO SEXO  PESAJE  PESO
1  44/05  37/08  2005-05-12   H 15/08/05  88.40
2  44/05  37/08  2005-05-12   H 25/08/05  88.30
3  44/05  37/08  2005-05-12   H 30/11/05 144.00
4  44/05  37/08  2005-05-12   H 23/12/05 150.00
5  45/05  23/00  2005-08-25   M 25/08/05  30.20
6  45/05  23/00  2005-08-25   M 30/11/05  84.10
7  45/05  23/00  2005-08-25   M 23/12/05  96.00
8  45/05  23/00  2005-08-25   M 15/01/05 100.88
9  45/05  23/00  2005-08-25   M 30/01/05 119.00
10 45/05  23/00  2005-08-25   M   <NA> 119.00
11 45/05  23/00  2005-08-25   M   <NA> 100.00
12 49/05 115/08  2005-09-10   H 30/11/05  77.00
13 49/05 115/08  2005-09-10   H 23/12/05  88.00
14 49/05 115/08  2005-09-10   H 10/09/05  30.00
15 50/06  37/08  2006-07-10 <NA>   <NA>   NA
```

Si queremos que no se tengan en cuenta los animales que, a pesar de tener madre (primera hoja), no tienen pesaje, entonces incluimos únicamente el argumento `all.y = TRUE`. En este caso saldrá el animal 50/06 de la hoja de datos `MadreCrias1`.



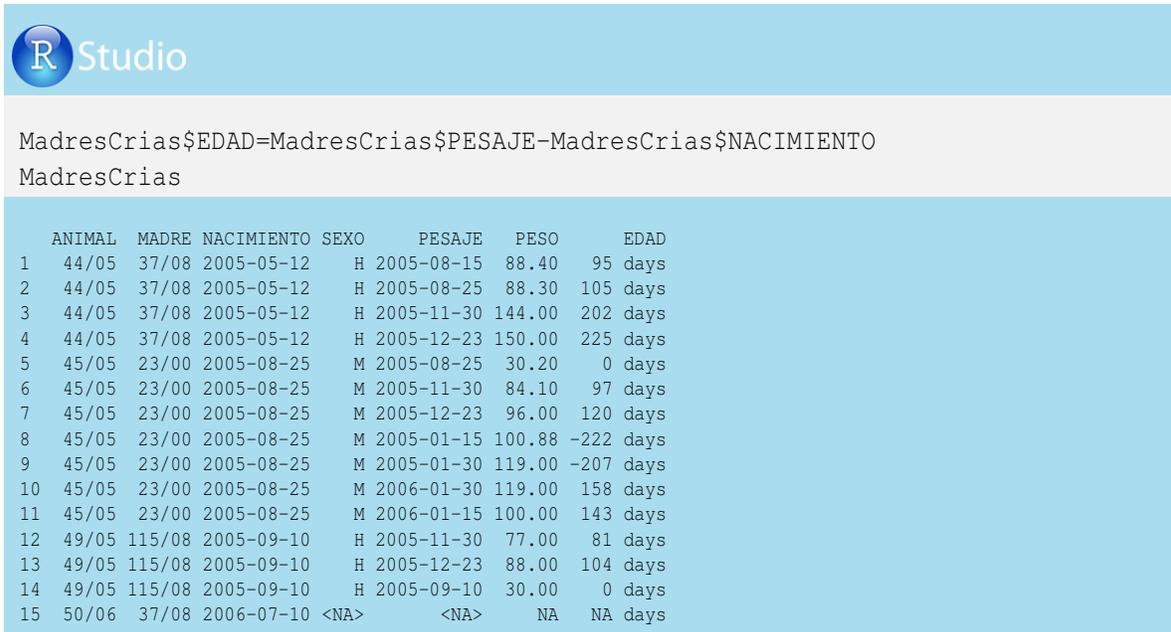
```
MadresCrias1=merge (Madre, Pesaje, all.y=TRUE)
MadresCrias1
```

```
  ANIMAL  MADRE  NACIMIENTO SEXO  PESAJE  PESO
1  44/05  37/08  2005-05-12   H 15/08/05  88.40
2  44/05  37/08  2005-05-12   H 25/08/05  88.30
3  44/05  37/08  2005-05-12   H 30/11/05 144.00
4  44/05  37/08  2005-05-12   H 23/12/05 150.00
5  45/05  23/00  2005-08-25   M 25/08/05  30.20
6  45/05  23/00  2005-08-25   M 30/11/05  84.10
7  45/05  23/00  2005-08-25   M 23/12/05  96.00
8  45/05  23/00  2005-08-25   M 15/01/05 100.88
9  45/05  23/00  2005-08-25   M 30/01/05 119.00
10 45/05  23/00  2005-08-25   M   <NA> 119.00
11 45/05  23/00  2005-08-25   M   <NA> 100.00
12 49/05 115/08  2005-09-10   H 30/11/05  77.00
13 49/05 115/08  2005-09-10   H 23/12/05  88.00
14 49/05 115/08  2005-09-10   H 10/09/05  30.00
```

#### 4.5.2. Creación de nuevas variables

Usualmente para obtener información adicional de las hojas de datos debemos crear nuevas variables, las cuales son el resultado de operaciones entre dos o más variables o entre variables y constantes.

Pongamos el ejemplo del cálculo de la edad de los animales al momento del pesaje de la hoja de datos *MadresCrias*. En este caso la edad estaría dada por la diferencia de la fecha de pesaje y la fecha de nacimiento:



```

R Studio

MadresCrias$EDAD=MadresCrias$PESAJE-MadresCrias$NACIMIENTO
MadresCrias

  ANIMAL  MADRE NACIMIENTO SEXO   PESAJE  PESO   EDAD
1  44/05  37/08 2005-05-12   H 2005-08-15  88.40  95 days
2  44/05  37/08 2005-05-12   H 2005-08-25  88.30 105 days
3  44/05  37/08 2005-05-12   H 2005-11-30 144.00 202 days
4  44/05  37/08 2005-05-12   H 2005-12-23 150.00 225 days
5  45/05  23/00 2005-08-25   M 2005-08-25  30.20   0 days
6  45/05  23/00 2005-08-25   M 2005-11-30  84.10  97 days
7  45/05  23/00 2005-08-25   M 2005-12-23  96.00 120 days
8  45/05  23/00 2005-08-25   M 2005-01-15 100.88 -222 days
9  45/05  23/00 2005-08-25   M 2005-01-30 119.00 -207 days
10 45/05  23/00 2005-08-25   M 2006-01-30 119.00 158 days
11 45/05  23/00 2005-08-25   M 2006-01-15 100.00 143 days
12 49/05 115/08 2005-09-10   H 2005-11-30  77.00  81 days
13 49/05 115/08 2005-09-10   H 2005-12-23  88.00 104 days
14 49/05 115/08 2005-09-10   H 2005-09-10  30.00   0 days
15 50/06  37/08 2006-07-10 <NA>   <NA>   NA  NA days

```

Observando detenidamente los datos generados, encontramos que la edad de algunos animales es negativa, lo cual es un claro error de digitación. La recomendación que le damos al lector es estar muy pendiente de posibles errores de digitación en las hojas de datos. En este caso, se digitó mal la información de fecha de pesaje del animal 45/05. Lamentablemente no tenemos la fecha correcta de los pesajes, por este motivo procedemos a eliminar los registros y además eliminamos los registros que no tengan edad, como es el caso del animal 50/06.

Podemos crear una nueva hoja de datos que llamaremos *MadresCriasCorrecta*, con la información correcta de la hoja de datos *MadresCrias*, mediante el comando *subset*, donde indicamos en los argumentos que dejaremos los animales con edad *mayor que cero* o animales con edad *igual que cero*. Para *mayor que* utilizamos el signo  $>$ , para la disyunción (o) utilizamos el signo  $|$  y para *igual que* utilizamos el signo  $==$ , como se indica a continuación:



```
MadresCrias=subset (MadresCrias, MadresCrias$EDAD > 0 | MadresCrias$EDAD == 0)
MadresCrias
```

	ANIMAL	MADRE	NACIMIENTO	SEXO	PESAJE	PESO	EDAD
1	44/05	37/08	2005-05-12	H	2005-08-15	88.4	95 days
2	44/05	37/08	2005-05-12	H	2005-08-25	88.3	105 days
3	44/05	37/08	2005-05-12	H	2005-11-30	144.0	202 days
4	44/05	37/08	2005-05-12	H	2005-12-23	150.0	225 days
5	45/05	23/00	2005-08-25	M	2005-08-25	30.2	0 days
6	45/05	23/00	2005-08-25	M	2005-11-30	84.1	97 days
7	45/05	23/00	2005-08-25	M	2005-12-23	96.0	120 days
10	45/05	23/00	2005-08-25	M	2006-01-30	119.0	158 days
11	45/05	23/00	2005-08-25	M	2006-01-15	100.0	143 days
12	49/05	115/08	2005-09-10	H	2005-11-30	77.0	81 days
13	49/05	115/08	2005-09-10	H	2005-12-23	88.0	104 days
14	49/05	115/08	2005-09-10	H	2005-09-10	30.0	0 days

El problema con la variable *EDAD* es que quedó con el formato de fecha en días (*days*), por lo cual no es posible estimar algunas de las medidas estadísticas (por ejemplo, promedios). Para esto, modificamos el formato de fecha por formato numérico con el comando *as.numeric*. Apliquemos este comando en R-project y veamos los dos resúmenes de la variable *EDAD*, antes y después de aplicado:



```
summary (MadresCriasCorrecto$EDAD)
MadresCriasCorrecto$EDAD=as.numeric (MadresCriasCorrecto$EDAD)
summary (MadresCriasCorrecto$EDAD)
```

```
> summary (MadresCriasCorrecto$EDAD)
      Length Class      Mode
      12 difftime numeric
> summary (MadresCriasCorrecto$EDAD)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0.0   91.5   104.5   110.8   146.8   225.0
```

Luego de realizar las operaciones de interés, puede ser de utilidad para el analista presentar la hoja de datos ordenada por alguna(s) de la(s) variable(s). Organicemos la hoja anterior por sexo, nombre, nacimiento y fecha de pesaje de la cría, mediante la función de R-project *do.call*.

Ahora crearemos la hoja de datos con la información organizada, a la que llamaremos *Organizado*, seguida de un igual (=) y del nombre de la hoja de datos (*MadreCriasCorrecto*). Posteriormente abrimos un corchete donde indicamos que usaremos la función *do.call*, seguido de un paréntesis que contiene la función *order* y la lista de variables para ordenar de *MadreCriasCorrecto*, que en nuestro caso las variables serían *SEXO*, *ANIMAL*,

**NACIMIENTO** y **PESAJE**. Hay que asegurarse de cerrar todos los paréntesis, los corchetes y las comillas, como se indica a continuación:

```

R Studio

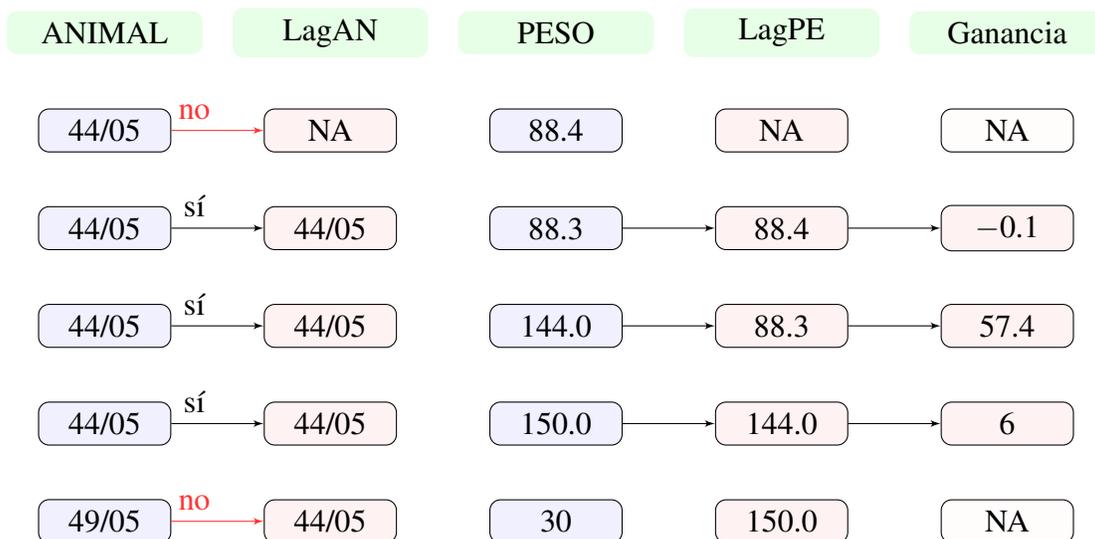
Organizado=MadresCriasCorrecto[do.call(order, MadresCriasCorrecto[
  c("SEXO", "ANIMAL", "NACIMIENTO", "PESAJE")]),]
Organizado

  ANIMAL MADRE NACIMIENTO SEXO PESAJE PESO EDAD
1  44/05  37/08 2005-05-12   H 2005-08-15  88.4  95
2  44/05  37/08 2005-05-12   H 2005-08-25  88.3 105
3  44/05  37/08 2005-05-12   H 2005-11-30 144.0 202
4  44/05  37/08 2005-05-12   H 2005-12-23 150.0 225
14 49/05 115/08 2005-09-10   H 2005-09-10  30.0   0
12 49/05 115/08 2005-09-10   H 2005-11-30  77.0  81
13 49/05 115/08 2005-09-10   H 2005-12-23  88.0 104
5  45/05  23/00 2005-08-25   M 2005-08-25  30.2   0
6  45/05  23/00 2005-08-25   M 2005-11-30  84.1  97
7  45/05  23/00 2005-08-25   M 2005-12-23  96.0 120
11 45/05  23/00 2005-08-25   M 2006-01-15 100.0 143
10 45/05  23/00 2005-08-25   M 2006-01-30 119.0 158
    
```

El lector puede utilizar otro tipo de función disponible en R-project para organizar datos.

### 4.5.3. Condicionales para operar variables

Ahora calculemos la ganancia de peso y la diferencia en días entre pesajes incluyendo condicionales para efectuar operaciones. Para esto utilizaremos la función *FuncLag* del capítulo anterior, con el propósito de generar tres variables con el nombre del animal, la diferencia de edad entre pesajes (días) y las ganancias de peso (kg). Veamos el esquema de una de las formas de calcular las ganancias de peso propuesta en este libro.



Vemos que tanto el primer registro del animal 44/05 como el primer registro del animal 49/05, no muestran ganancias de peso, dado que no tienen un pesaje anterior con el cual establecer la diferencia. De la misma forma aplicamos la variable *EDAD* para obtener los días de intervalo de pesajes.

Antes de realizar el proceso, recordemos la función *FuncLag* del capítulo 3, pero con algunas modificaciones:

```

R Studio

FuncLag= function (LAG) {
  a= matrix (LAG [1:(length(LAG)-1)]);
  rbind(NA,a)
}

```

Emplearemos esta función para crear tres variables rezagadas, así:

```

R Studio

Organizado$LagPE=FuncLag(Organizado$PESO)
Organizado$LagAN=FuncLag(Organizado$ANIMAL)
Organizado$LagED=FuncLag(Organizado$EDAD)
Organizado

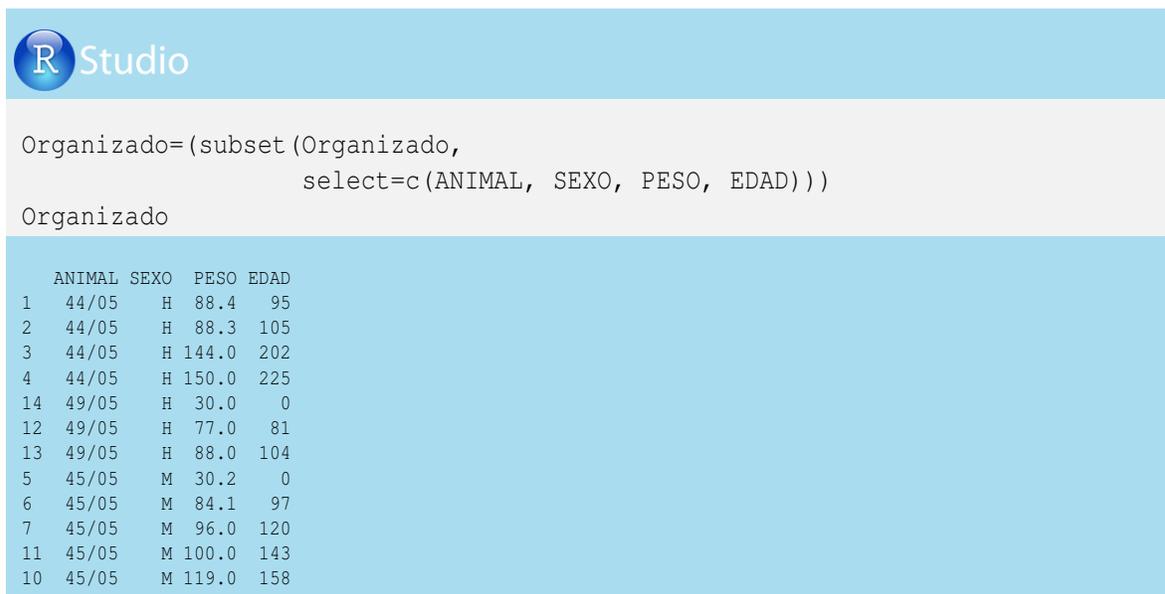
```

	ANIMAL	MADRE	NACIMIENTO	SEXO	PESAJE	PESO	EDAD	LagPE	LagED	LagAN
1	44/05	37/08	2005-05-12	H	2005-08-15	88.4	95	NA	NA	<NA>
2	44/05	37/08	2005-05-12	H	2005-08-25	88.3	105	88.4	95	44/05
3	44/05	37/08	2005-05-12	H	2005-11-30	144.0	202	88.3	105	44/05
4	44/05	37/08	2005-05-12	H	2005-12-23	150.0	225	144.0	202	44/05
14	49/05	115/08	2005-09-10	H	2005-09-10	30.0	0	150.0	225	44/05
12	49/05	115/08	2005-09-10	H	2005-11-30	77.0	81	30.0	0	49/05
13	49/05	115/08	2005-09-10	H	2005-12-23	88.0	104	77.0	81	49/05
5	45/05	23/00	2005-08-25	M	2005-08-25	30.2	0	88.0	104	49/05
6	45/05	23/00	2005-08-25	M	2005-11-30	84.1	97	30.2	0	45/05
7	45/05	23/00	2005-08-25	M	2005-12-23	96.0	120	84.1	97	45/05
11	45/05	23/00	2005-08-25	M	2006-01-15	100.0	143	96.0	120	45/05
10	45/05	23/00	2005-08-25	M	2006-01-30	119.0	158	100.0	143	45/05

Ahora podemos construir la variable ganancia de peso en el periodo (*GP*), la cual está dada por la diferencia entre *PESO* y *LagPE*, la variable intervalo entre pesajes *IntP*, que está dada por la diferencia entre la variable *EDAD* y el rezago de la edad con la variable *LagED*, y la variable ganancia diaria, que está dada por la división entre *GP* e *IntP*.

Primero eliminemos las variables *NACIMIENTO*, *MADRE* y *PESAJE*, porque no las utilizaremos en esta parte del ejercicio y ello nos permitirá visualizar mejor las salidas del R-project. Para esto utilizaremos el comando *subset*, indicando en los argumentos el nombre

de la hoja de datos y seleccionando las variables que quedarían en la nueva hoja de datos:

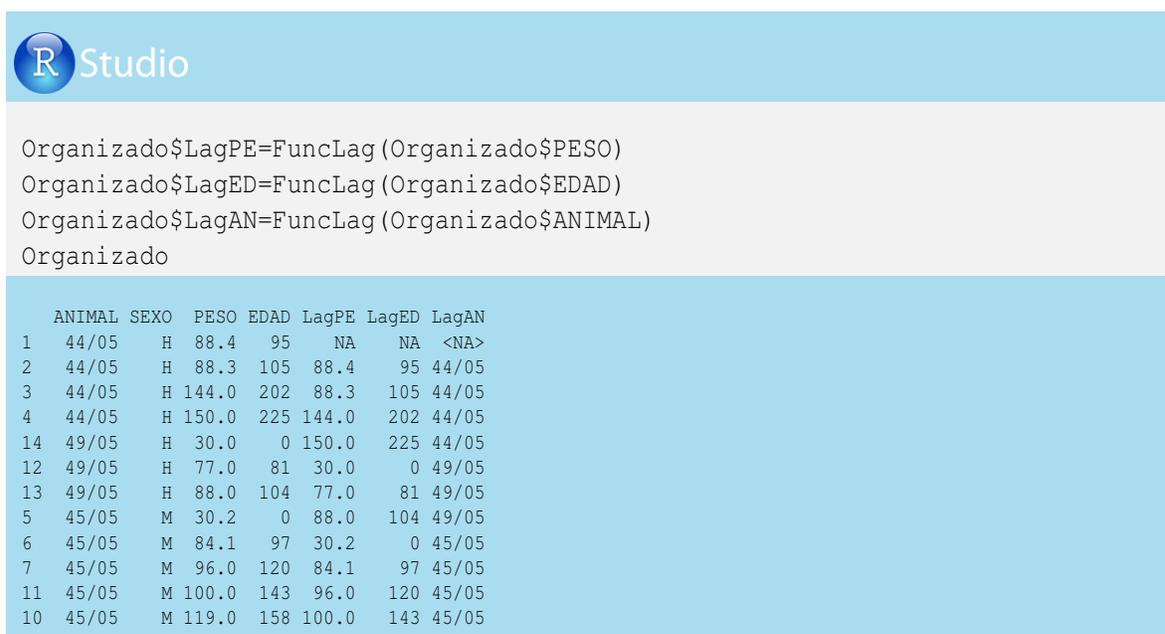


```

Organizado=(subset (Organizado,
                    select=c (ANIMAL, SEXO, PESO, EDAD)))
Organizado
    
```

	ANIMAL	SEXO	PESO	EDAD
1	44/05	H	88.4	95
2	44/05	H	88.3	105
3	44/05	H	144.0	202
4	44/05	H	150.0	225
14	49/05	H	30.0	0
12	49/05	H	77.0	81
13	49/05	H	88.0	104
5	45/05	M	30.2	0
6	45/05	M	84.1	97
7	45/05	M	96.0	120
11	45/05	M	100.0	143
10	45/05	M	119.0	158

Apliquemos la función *FuncLag*:

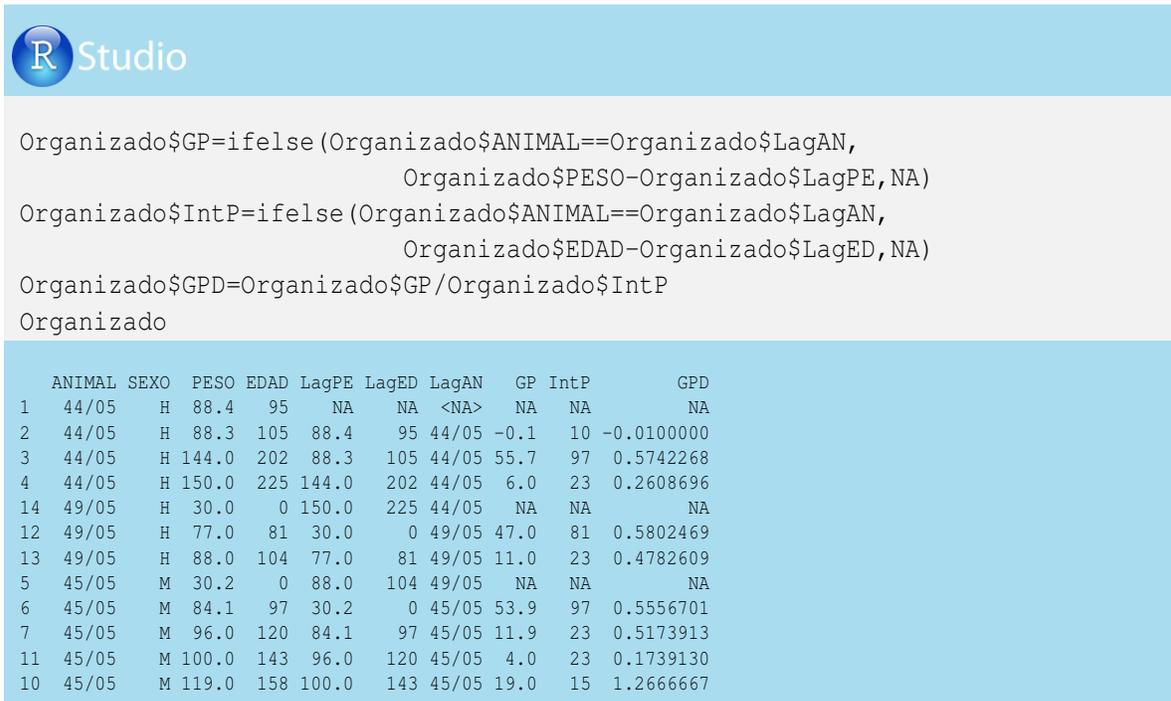


```

Organizado$LagPE=FuncLag (Organizado$PESO)
Organizado$LagED=FuncLag (Organizado$EDAD)
Organizado$LagAN=FuncLag (Organizado$ANIMAL)
Organizado
    
```

	ANIMAL	SEXO	PESO	EDAD	LagPE	LagED	LagAN
1	44/05	H	88.4	95	NA	NA	<NA>
2	44/05	H	88.3	105	88.4	95	44/05
3	44/05	H	144.0	202	88.3	105	44/05
4	44/05	H	150.0	225	144.0	202	44/05
14	49/05	H	30.0	0	150.0	225	44/05
12	49/05	H	77.0	81	30.0	0	49/05
13	49/05	H	88.0	104	77.0	81	49/05
5	45/05	M	30.2	0	88.0	104	49/05
6	45/05	M	84.1	97	30.2	0	45/05
7	45/05	M	96.0	120	84.1	97	45/05
11	45/05	M	100.0	143	96.0	120	45/05
10	45/05	M	119.0	158	100.0	143	45/05

Finalmente, creamos las variables *GP*, *IntP* y *GPD*:



```

Organizado$GP=ifelse (Organizado$ANIMAL==Organizado$LagAN,
                      Organizado$PESO-Organizado$LagPE, NA)
Organizado$IntP=ifelse (Organizado$ANIMAL==Organizado$LagAN,
                        Organizado$EDAD-Organizado$LagED, NA)
Organizado$GPD=Organizado$GP/Organizado$IntP
Organizado

```

	ANIMAL	SEXO	PESO	EDAD	LagPE	LagED	LagAN	GP	IntP	GPD
1	44/05	H	88.4	95	NA	NA	<NA>	NA	NA	NA
2	44/05	H	88.3	105	88.4	95	44/05	-0.1	10	-0.0100000
3	44/05	H	144.0	202	88.3	105	44/05	55.7	97	0.5742268
4	44/05	H	150.0	225	144.0	202	44/05	6.0	23	0.2608696
14	49/05	H	30.0	0	150.0	225	44/05	NA	NA	NA
12	49/05	H	77.0	81	30.0	0	49/05	47.0	81	0.5802469
13	49/05	H	88.0	104	77.0	81	49/05	11.0	23	0.4782609
5	45/05	M	30.2	0	88.0	104	49/05	NA	NA	NA
6	45/05	M	84.1	97	30.2	0	45/05	53.9	97	0.5556701
7	45/05	M	96.0	120	84.1	97	45/05	11.9	23	0.5173913
11	45/05	M	100.0	143	96.0	120	45/05	4.0	23	0.1739130
10	45/05	M	119.0	158	100.0	143	45/05	19.0	15	1.2666667

## 4.6. Ejemplos de construcción de hojas de datos en producción animal

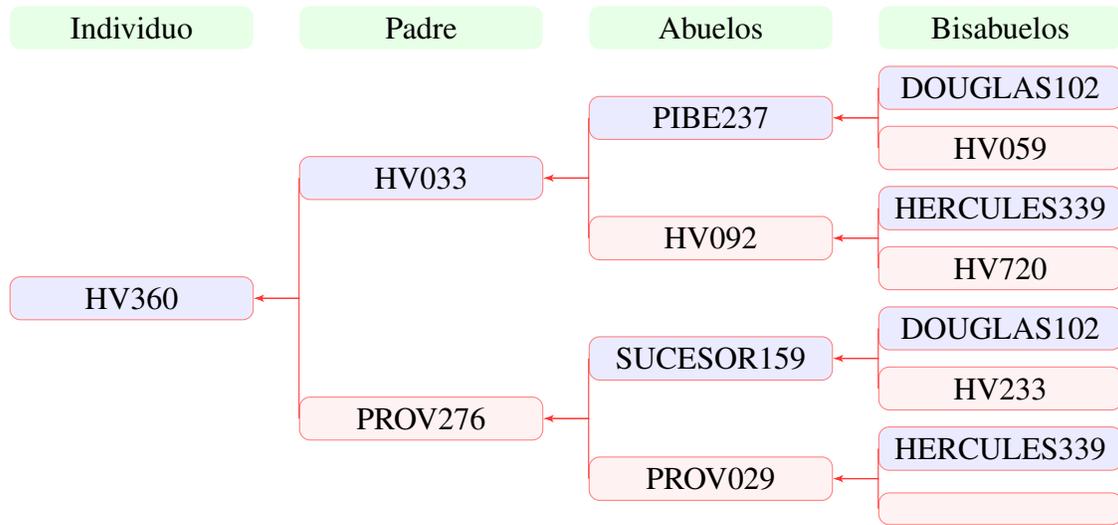
En esta sección veremos algunos ejemplos relacionados con montaje de genealogías y cálculos de la producción de leche, las cuales se pueden aplicar y modificar dependiendo de las circunstancias de interés para el analista en el manejo de hojas de datos en ciencias animales.

### 4.6.1. Montaje de genealogías

En los sistemas de producción en los cuales es necesario hacer control de la paternidad de los individuos, se requiere construir árboles genealógicos o genealogías con el fin de controlar los apareamientos de los animales o generar información de parentesco entre ellos. La construcción de genealogías también es fundamental en la realización de evaluaciones genéticas.

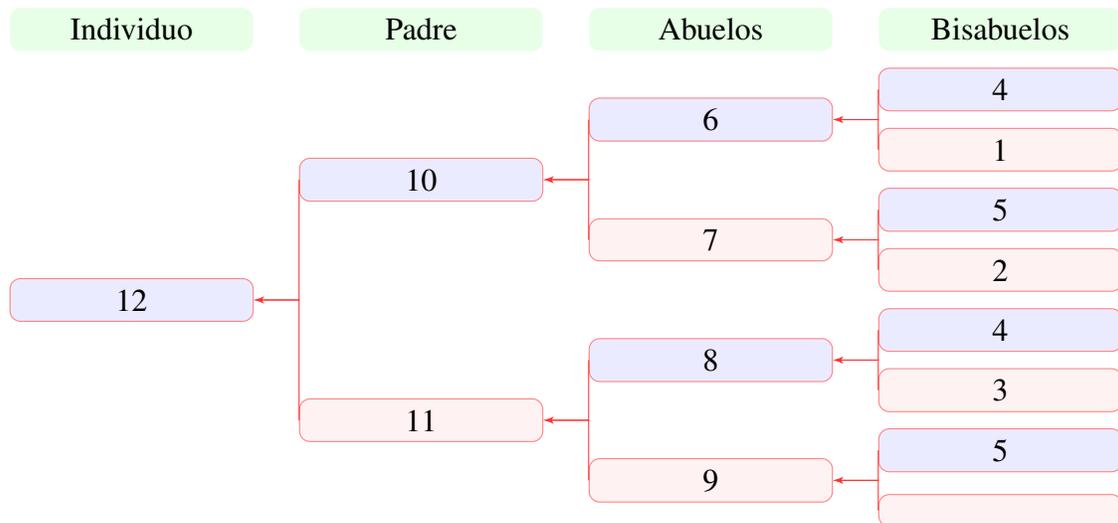
En este apartado usaremos como ejemplo la construcción de la genealogía del individuo *HV360*, perteneciente a la raza Blanco Oreginegro. Este individuo tiene información de su padre, su madre, los cuatro abuelos y siete bisabuelos.

A continuación se presentan las cuatro generaciones conocidas de este individuo, pero falta la información de un bisabuelo. Identificamos con color azul a los machos y con color rosado a las hembras para una mejor visualización:



Podemos observar que los ancestros *DOUGLAS102* y *HERCULES339* están presentes en la línea materna y en la línea paterna, lo que indica que el animal *HV360* es endogámico.

Aparentemente aparecen 14 animales, pero hay animales repetidos en la genealogía; por esto es necesario numerar los individuos, iniciando con los animales de los que no se tiene información de sus padres (en este caso las hembras *HV059*, *HV720* y *HV233* y los machos *DOUGLAS102* y *HERCULES339*), seguido por los hijos de los anteriores animales, hasta llegar al animal que nació de último.



En la renumeración de animales, los padres siempre tendrán una numeración menor que la de sus hijos. La información suministrada por el pedigree de los animales se puede ubicar en tres columnas, donde se relacionan todos los animales con sus respectivos progenitores. En caso de no tener información sobre sus progenitores, se deja un espacio en blanco, pero en el caso de R-project quedará como *NA*, según veremos más adelante. Por el momento, se realizará la genealogía en tres columnas:

Número	Animal	Madre	Padre
1	HV059		
2	HV720		
3	HV233		
4	DOUGLAS102		
5	HERCULES339		
6	PIBE237	HV059	DOUGLAS102
7	HV092	HV720	HERCULES339
8	SUCESOR159	HV233	DOUGLAS102
9	PROV029		HERCULES339
10	HV033	HV092	PIBE237
11	PROV276	PROV029	SUCESOR159
12	HV360	PROV276	HV033

Veamos cómo queda la estructura de la genealogía con los animales reenumerados:

ANIMAL Renumerado	Madre Renumerada	Padre Renumerado
1		
2		
3		
4		
5		
6	1	4
7	2	5
8	3	4
9		5
10	7	6
11	9	8
12	11	10

Ingredamos en R-project la información de la genealogía, creando tres vectores: *id* relacionada con la identificación del animal, *sire* con el nombre del padre y *dam* con el nombre de la madre, que posteriormente se unirán en la hoja de datos llamada *junto*:



```

id   =c("HV360", "HV033", "PROV276", "PIBE237", "HV092", "SUCESOR159", "PROV029")
sire =c("HV033", "PIBE237", "SUCESOR159", "DOUGLAS102", "HERCULES339", "DOUGLAS102",
        , "HERCULES339")
dam  =c("PROV276", "HV092", "PROV029", "HV059", "HV720", "HV233", NA)
junto =data.frame (id,dam,sire)
junto

```

	id	dam	sire
1	HV360	PROV276	HV033
2	HV033	HV092	PIBE237
3	PROV276	PROV029	SUCESOR159
4	PIBE237	HV059	DOUGLAS102
5	HV092	HV720	HERCULES339
6	SUCESOR159	HV233	DOUGLAS102
7	PROV029	<NA>	HERCULES339

En R-project existen varias librerías que permiten realizar análisis de genealogías en animales y plantas. En este libro utilizaremos la librería *pedantics*, desarrollada por Morrissey (2012) para estudios de genealogía en poblaciones, y la librería *glmm* (Muestreo Monte Carlo con cadenas de Markov para modelos mixtos lineales generalizados y multivariados) de Hadfield (2010), con énfasis en el estudio de efectos aleatorios correlacionados derivados de pedigrees y filogenias, desarrollado por Hadfield (2010).

A continuación se presenta la programación en R-project para instalar el paquete anteriormente mencionado:



```

install.packages("MCMCglmm")
install.packages("pedantics")
library(pedantics)

```

Utilizaremos el comando *fixPedigree*, seguido del nombre de la hoja de datos y la posición donde se encuentran las tres columnas, para generar la lista de los animales, iniciando con aquellos que no tienen información de progenitores y continuando en forma ascendente por orden de nacimiento; veamos:



```
pedigree=fixPedigree(junto[,1:3])
pedigree
```

	id	dam	sire
25	HV059	<NA>	<NA>
26	HV720	<NA>	<NA>
27	HV233	<NA>	<NA>
46	DOUGLAS102	<NA>	<NA>
47	HERCULES339	<NA>	<NA>
4	PIBE237	HV059	DOUGLAS102
5	HV092	HV720	HERCULES339
6	SUCESOR159	HV233	DOUGLAS102
7	PROV029	<NA>	HERCULES339
2	HV033	HV092	PIBE237
3	PROV276	PROV029	SUCESOR159
1	HV360	PROV276	HV033

Ahora crearemos una columna denominada *rid* con la numeración correspondiente al orden presentado en el paso anterior. Utilizaremos el comando *seq* para generar la secuencia. Después de *seq* va un 1 entre paréntesis que indica el primer valor de la secuencia, seguido de una coma para incluir el valor final de la secuencia. El valor final se obtiene con el comando *length* que indica la longitud de una variable (en este caso para *id*) con el objetivo de indicar que el número de animales con identificación es 12:



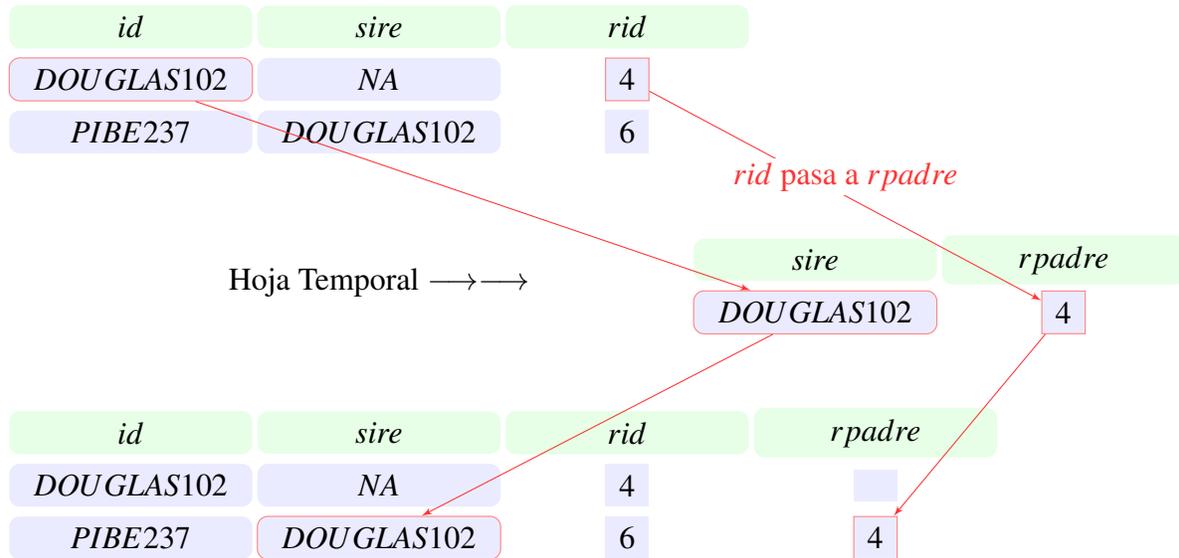
```
pedigree$rid= seq ( 1, length (pedigree$id))
pedigree
```

	id	dam	sire	rid
25	HV059	<NA>	<NA>	1
26	HV720	<NA>	<NA>	2
27	HV233	<NA>	<NA>	3
46	DOUGLAS102	<NA>	<NA>	4
47	HERCULES339	<NA>	<NA>	5
4	PIBE237	HV059	DOUGLAS102	6
5	HV092	HV720	HERCULES339	7
6	SUCESOR159	HV233	DOUGLAS102	8
7	PROV029	<NA>	HERCULES339	9
2	HV033	HV092	PIBE237	10
3	PROV276	PROV029	SUCESOR159	11
1	HV360	PROV276	HV033	12

Ahora realizamos un artificio para generar la renumeración de los padres, los cuales aparecen anteriormente como animales. Por ejemplo, al animal *DOUGLAS102*, que aparece con la numeración *rid* = 4, se le debe poner en la columna de padre *rpadre* = 4.

Para esto generamos una hoja de datos temporal donde convertimos los animales a padres,

y posteriormente la combinamos con la hoja inicial, donde se incluirá la columna con la numeración de padre. Veamos el esquema:



Aplicaremos el esquema anterior a la hoja de datos *pedigree*. Primero creamos la hoja temporal *padreTemporal* y luego combinamos las dos hojas por la columna *sire* para generar la hoja de datos *padre*. Creamos la hoja temporal de la siguiente manera:

```

R Studio

padreTemporal= data.frame (sire=pedigree$id,
                           rpadre=pedigree$rid)
padreTemporal

   sire rpadre
1  HV059     1
2  HV720     2
3  HV233     3
4 DOUGLAS102  4
5 HERCULES339 5
6  PIBE237     6
7  HV092     7
8  SUCESOR159 8
9  PROV029     9
10 HV033     10
11 PROV276    11
12 HV360     12
    
```

Ahora combinemos las hojas de datos *pedigree* y *padreTemporal* para obtener la hoja de datos llamada *padre*:

```

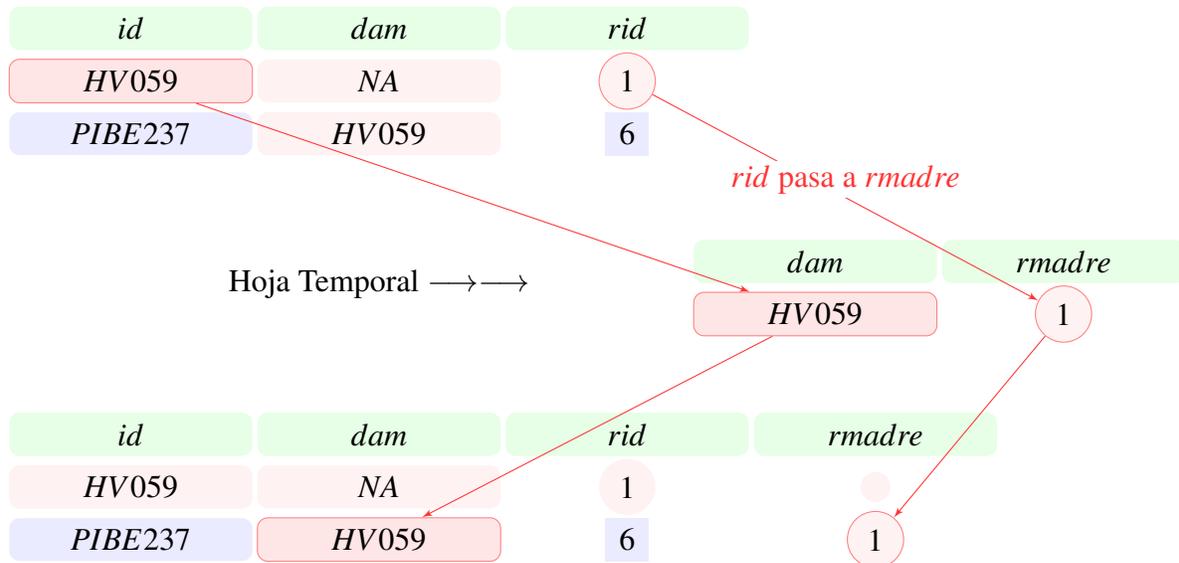
R Studio

padre= merge (pedigree, padreTemporal, all.x=TRUE)
padre

```

	sire	id	dam	rid	rpadre
1	DOUGLAS102	PIBE237	HV059	6	4
2	DOUGLAS102	SUCESOR159	HV233	8	4
3	HERCULES339	HV092	HV720	7	5
4	HERCULES339	PROV029	<NA>	9	5
5	HV033	HV360	PROV276	12	10
6	PIBE237	HV033	HV092	10	6
7	SUCESOR159	PROV276	PROV029	11	8
8	<NA>	HV059	<NA>	1	NA
9	<NA>	HV720	<NA>	2	NA
10	<NA>	HV233	<NA>	3	NA
11	<NA>	DOUGLAS102	<NA>	4	NA
12	<NA>	HERCULES339	<NA>	5	NA

Realizamos el mismo procedimiento para reenumerar las madres. A continuación presentaremos el esquema para la hembra *HV059*, la cual aparece con la numeración 1 y después aparece como madre de *PIBE237*:



De manera similar al caso de los padres, creamos el archivo temporal *madreTemporal* en R-proyect:

```
R Studio

madreTemporal= data.frame (dam=pedigree$id, rmadre=pedigree$rid)
madreTemporal
```

1	HV059	1
2	HV720	2
3	HV233	3
4	DOUGLAS102	4
5	HERCULES339	5
6	PIBE237	6
7	HV092	7
8	SUCESOR159	8
9	PROV029	9
10	HV033	10
11	PROV276	11
12	HV360	12

Finalmente combinamos la hoja de datos *padre* con la hoja de datos *madreTemporal* y de esta forma obtenemos la reenumeración de los animales:

```
R Studio

madre= merge (padre, madreTemporal, all.x=TRUE)
madre
```

	dam	sire	id	rid	rpadre	rmadre
1	HV059	DOUGLAS102	PIBE237	6	4	1
2	HV092	PIBE237	HV033	10	6	7
3	HV233	DOUGLAS102	SUCESOR159	8	4	3
4	HV720	HERCULES339	HV092	7	5	2
5	PROV029	SUCESOR159	PROV276	11	8	9
6	PROV276	HV033	HV360	12	10	11
7	<NA>	<NA>	HV720	2	NA	NA
8	<NA>	<NA>	HV233	3	NA	NA
9	<NA>	<NA>	DOUGLAS102	4	NA	NA
10	<NA>	HERCULES339	PROV029	9	5	NA
11	<NA>	<NA>	HV059	1	NA	NA
12	<NA>	<NA>	HERCULES339	5	NA	NA

En el siguiente cuadro se muestra la hoja de datos reenumerada y organizada cronológicamente según el nacimiento de los animales:

```
R Studio

ordenados=madre[do.call(order,madre[c("rid")]),]
ordenados
```

	dam	sire	id	rid	rpadre	rmadre
11	<NA>	<NA>	HV059	1	NA	NA
7	<NA>	<NA>	HV720	2	NA	NA
8	<NA>	<NA>	HV233	3	NA	NA
9	<NA>	<NA>	DOUGLAS102	4	NA	NA
12	<NA>	<NA>	HERCULES339	5	NA	NA
1	HV059	DOUGLAS102	PIBE237	6	4	1
4	HV720	HERCULES339	HV092	7	5	2
3	HV233	DOUGLAS102	SUCESOR159	8	4	3
10	<NA>	HERCULES339	PROV029	9	5	NA
2	HV092	PIBE237	HV033	10	6	7
5	PROV029	SUCESOR159	PROV276	11	8	9
6	PROV276	HV033	HV360	12	10	11

El proceso anterior es muy importante para generar una hoja de datos con genealogía reenumerada, la cual es necesaria para ejecutar programas específicos de evaluaciones genéticas.

Regresemos ahora a los comandos específicos de la librería *pedantics*, con el objetivo de generar un resumen numérico de la genealogía. Utilizaremos el comando *pedigreeStats* con una hoja de datos nueva que llamaremos *statorden* y solo contiene información de animal, madre y padre, con las variables *id*, *dam* y *sire*, respectivamente. Veamos:

```
R Studio

statorden=data.frame(id=ordenados$id,dam=ordenados$dam,sire=ordenados$sire)
statorden
```

```
> statorden
  id      dam      sire
1 HV059 <NA> <NA>
2 HV720 <NA> <NA>
3 HV233 <NA> <NA>
4 DOUGLAS102 <NA> <NA>
5 HERCULES339 <NA> <NA>
6 PIBE237 HV059 DOUGLAS102
7 HV092 HV720 HERCULES339
8 SUCESOR159 HV233 DOUGLAS102
9 PROV029 <NA> HERCULES339
10 HV033 HV092 PIBE237
11 PROV276 PROV029 SUCESOR159
12 HV360 PROV276 HV033
```

Con *pedigreeStats*, obtengamos una hoja de datos que llamaremos *Resumen* y contendrá información importante de la genealogía. Es necesario incluir el argumento *graphicalReport*, seguido de = “y” o = “n” para generar o no gráficos de la genealogía, respectivamente.

```
R Studio  
Resumen=pedigreeStats (statorden, graphicalReport=' n' )
```

Ahora solicitemos información relevante de esa genealogía, como es el caso del número total de animales, el número de hijos y el número de animales que son padres. Veamos:

```
R Studio  
Resumen$totalSampleSize  
Resumen$totalPaternities  
Resumen$paternalSibships  
  
> Resumen$totalSampleSize  
[1] 12  
> Resumen$totalPaternities  
[1] 7  
> Resumen$paternalSibships  
      Var1 Freq  
1 DOUGLAS102 2  
2 HERCULES339 2  
3 HV033 1  
4 PIPE237 1  
5 SUCESOR159 1
```

En esta programación le solicitamos que indique:

Número total de animales (*Resumen\$totalSampleSize = 12*),  
Número de animales con padre (*Resumen\$totalPaternities = 7*) y  
Número de hijos por toro (*Resumen\$paternalSibships*).

También podemos conocer el número de animales con madre (*Resumen\$totalMaternities = 6*) y el número de hijos por madre (*Resumen\$maternalSibships*):



```
Resumen$totalMaternities
Resumen$maternalSibships
```

```
> Resumen$totalMaternities
[1] 6
> Resumen$maternalSibships
      Var1 Freq
1 HV059     1
2 HV092     1
3 HV233     1
4 HV720     1
5 PROV029   1
6 PROV276   1
```

Ahora generemos una hoja de datos con información del porcentaje de endogamia (*inbreeding coefficient*) de los animales. Por tener un ancestro común por línea materna y paterna, el animal *HV360* presenta un porcentaje de endogamia del 6.25 %.



```
Endogamia=data.frame(pedigree, endogamia=
                      (Resumen$inbreedingCoefficients*100))
Endogamia
```

	id	dam	sire	rid	endogamia
25	HV059	<NA>	<NA>	1	0.00
26	HV720	<NA>	<NA>	2	0.00
27	HV233	<NA>	<NA>	3	0.00
46	DOUGLAS102	<NA>	<NA>	4	0.00
47	HERCULES339	<NA>	<NA>	5	0.00
4	PIBE237	HV059	DOUGLAS102	6	0.00
5	HV092	HV720	HERCULES339	7	0.00
6	SUCESOR159	HV233	DOUGLAS102	8	0.00
7	PROV029	<NA>	HERCULES339	9	0.00
2	HV033	HV092	PIBE237	10	0.00
3	PROV276	PROV029	SUCESOR159	11	0.00
1	HV360	PROV276	HV033	12	6.25

En la genealogía de *HV033*, cinco animales no tienen información de sus progenitores (generación 0 o población base), cuatro animales provienen de la generación 0, o sea, primera generación (*PIBE237*, *HV092*, *SUCESOR159* y *PROV029*), en la segunda generación tenemos dos animales (*HV033* y *PROV276*) y nuestro *HV033* aparece en la tercera generación.

Generemos una tabla de resumen con la información de individuos por generación, que indica que en las generaciones 0, 1, 2 y 3 hay 5, 4, 2 y 1 animales, respectivamente:

```

R Studio
Resumen$pedigreeDepth
0 1 2 3
5 4 2 1
    
```

Le recomendamos al lector explorar esta interesante librería y otras disponibles para análisis de pedigrée, filogenia, evaluaciones genéticas, evaluaciones moleculares, entre otras.

#### 4.6.2. Cálculo de la producción de leche por lactancia

En producción animal es necesario calcular la producción total de leche de una vaca mediante pesajes diarios de leche a lo largo de la lactancia, con el propósito de tomar decisiones sobre su continuidad en el rebaño. Hay diferentes métodos para calcular la producción total de leche, que incluyen principalmente ajustes al inicio y al final de la lactancia o el uso de modelos de curva de lactancia como Wood, Gompertz, entre otros. Sin embargo, el método más utilizado es el recomendado por el Comité Internacional de Registro Animal (ICAR), el cual no realiza ajustes al inicio y al final de la lactancia y tiene en cuenta los promedios de los pesajes de leche mediante la siguiente fórmula:

$$y_{total} = P_1D_1 + \left[ \sum_{i=2}^n \frac{P_i + P_{i-1}}{2} (D_i - D_{i-1}) \right] + (D_s - D_u)P_u$$

Donde  $y_{total}$  es la producción total de leche en una lactancia, y  $P_1D_1$  hace referencia a la producción de leche desde el parto hasta el primer pesaje o control lechero;  $P_1$  es la producción de leche en el primer control y  $D_1$  es el número de días de este periodo.

$\left[ \sum_{i=2}^n \frac{P_i + P_{i-1}}{2} (D_i - D_{i-1}) \right]$  se refiere a la sumatoria de las producciones parciales ocurridas entre controles, donde  $\frac{P_i + P_{i-1}}{2}$  es el promedio de producción de leche entre el pesaje actual ( $P_i$ ) y el anterior ( $P_{i-1}$ ) y  $D_i - D_{i-1}$  es el intervalo entre controles sucesivos.

$(D_s - D_u)P_u$  es la producción de leche ocurrida entre el último pesaje y el día de secado.

Veamos el ejemplo de tres vacas que tienen datos de una lactancia.

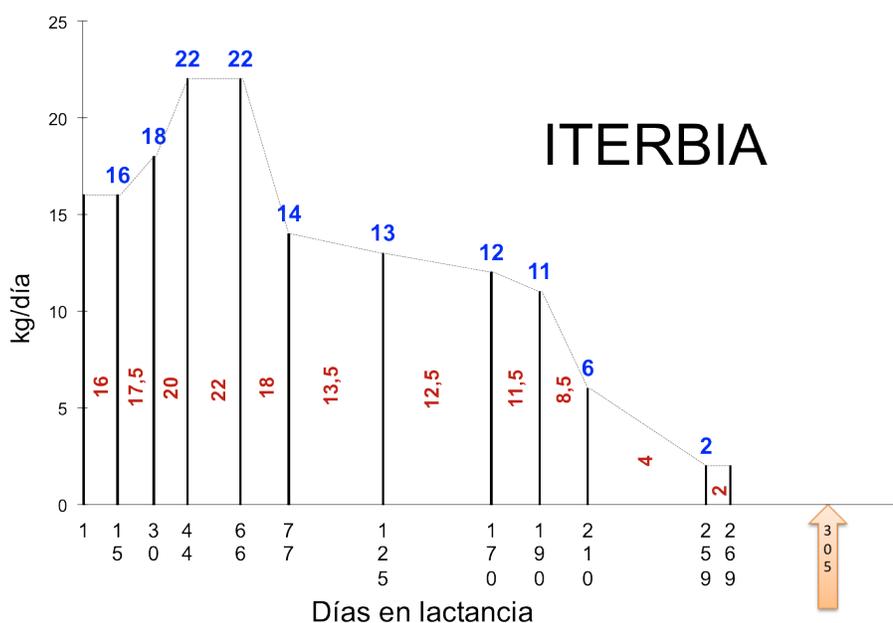
VACA.xlsx						
	A	B	C	D	E	F
1	Nombre	FechaParto	FechaSecado			
2	ITERBIA	12/01/10	8/10/10			
3	ESCANDIA	24/01/10	28/12/10			
4	HOLMIA	12/01/10	15/11/10			
5						

Según las fechas de parto (*FechaParto*) y la fechas de terminación de su lactancia (*FechaSecado*), las tres vacas parieron en enero de 2010 y se secaron entre octubre y diciembre de 2012. En sistemas de producción especializados para leche, lo normal es tener lactancias que duran 305 días. En este caso, tenemos lactancias inferiores a 305 días, cercanas a los 305 días (307 días) y superiores a los 305 días (338 días).

Iniciemos observando el desempeño productivo de la vaca *ITERBIA*.

LECHE.xlsx						
	A	B	C	D	E	F
1	Nombre	FechaControl	Leche			
2	ITERBIA	27/01/10	16			
3	ITERBIA	11/02/10	18			
4	ITERBIA	25/02/10	22			
5	ITERBIA	19/03/10	22			
6	ITERBIA	30/03/10	14			
7	ITERBIA	17/05/10	13			
8	ITERBIA	1/07/10	12			
9	ITERBIA	21/07/10	11			
10	ITERBIA	10/08/10	6			
11	ITERBIA	28/09/10	2			

Esta vaca tuvo una lactancia corta, donde en su primer control lechero produjo 16 lt/día, subió su producción a 22 lt a los dos meses de lactancia, y tuvo una posterior reducción de la producción de leche hasta llegar a 2 lt en su último control a los 259 días. Se dejó de ordeñar a los 269 días. Para el cálculo de la producción de leche total, observemos detalladamente la siguiente figura.



Podemos dividir el cálculo de la producción de leche en tres momentos. El primer momento está dado por la sumatoria de la producción de leche diaria desde el parto hasta el primer control. Cabe aclarar que la hembra produce calostro en sus primeros días, pero se asumió como criterio para el cálculo como si fuera leche.

El segundo momento está dado por los promedios de producciones de leche entre periodos multiplicados por el número de días, y el tercer periodo está dado por la producción de leche en el último control por el intervalo de días ocurrido desde el último control hasta el secado.

El cálculo de la producción de leche total sería:

$$y_{total} = 16 * 15 + \left[ \left( \frac{18+16}{2} * 15 \right) + \left( \frac{22+18}{2} * 14 \right) + \dots + \left( \frac{2+6}{2} * 49 \right) \right] + 10 * 2$$

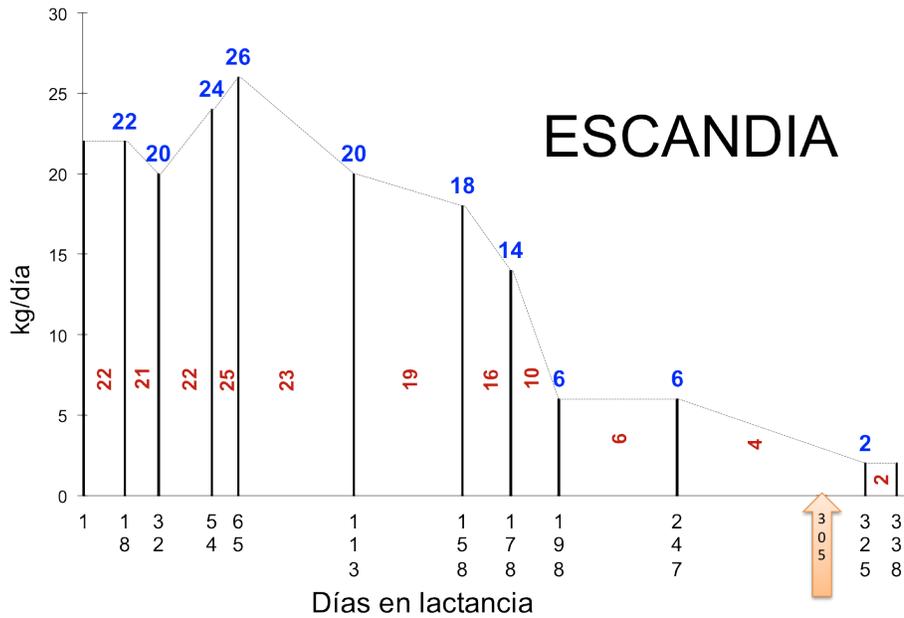
$$y_{total} = 240 + [(17.5 * 15) + (20 * 14) + \dots + (4 * 49)] + 10 * 2$$

$$y_{total} = 3283.5lt$$

Ahora miremos la producción de leche de la vaca *ESCANDIA*

LECHE.xlsx						
	A	B	C	D	E	F
12	ESCANDIA	11/02/10	22			
13	ESCANDIA	25/02/10	20			
14	ESCANDIA	19/03/10	24			
15	ESCANDIA	30/03/10	26			
16	ESCANDIA	17/05/10	20			
17	ESCANDIA	1/07/10	18			
18	ESCANDIA	21/07/10	14			
19	ESCANDIA	10/08/10	6			
20	ESCANDIA	28/09/10	6			
21	ESCANDIA	15/12/10	2			

Esta vaca tuvo una lactancia de 338 días, donde en su primer control lechero produjo 22 lt/día, bajó su producción al mes de lactancia, y tuvo un pico de producción de 26 lt. Su último control fue a los 325 días y se dejó de ordeñar a los 338 días. Veamos el gráfico del desempeño productivo de *ESCANDIA*:



Su producción total de leche en esta lactancia fue:

$$y_{total} = 22 * 18 + \left[ \left( \frac{20+22}{2} * 14 \right) + \left( \frac{24+20}{2} * 22 \right) + \dots + \left( \frac{2+6}{2} * 78 \right) \right] + 13 * 2$$

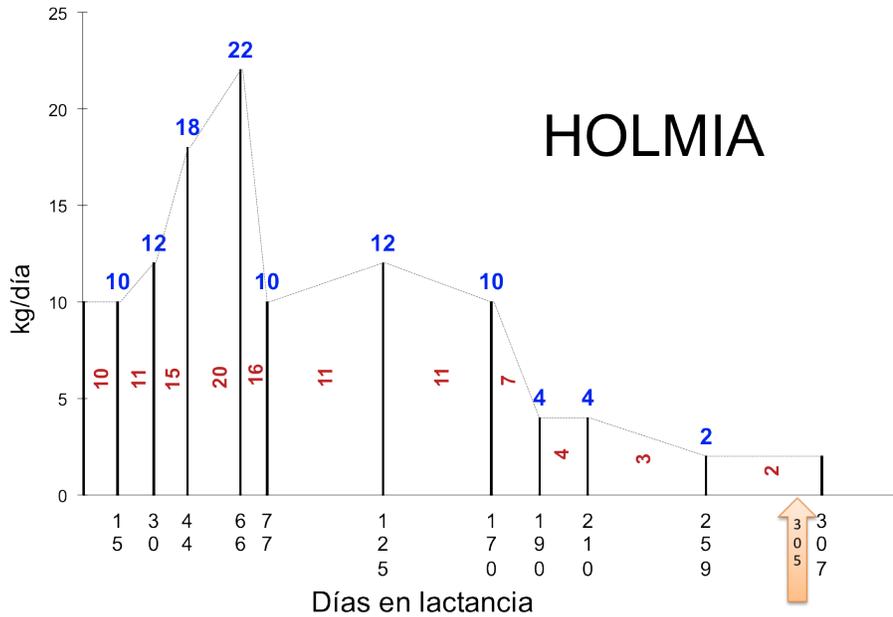
$$y_{total} = 396 + [(21 * 14) + (22 * 22) + \dots + (4 * 78)] + 13 * 2$$

$$y_{total} = 4560lt$$

Por último, miremos el desempeño de *HOLMIA*:

LECHE.xlsx						
	A	B	C	D	E	F
22	HOLMIA	27/01/10	10			
23	HOLMIA	11/02/10	12			
24	HOLMIA	25/02/10	18			
25	HOLMIA	19/03/10	22			
26	HOLMIA	30/03/10	10			
27	HOLMIA	17/05/10	12			
28	HOLMIA	1/07/10	10			
29	HOLMIA	21/07/10	4			
30	HOLMIA	10/08/10	4			
31	HOLMIA	28/09/10	2			

*HOLMIA* tuvo una lactancia de 307 días, y su pico fue de 22 lt a los dos meses; veamos gráficamente su desempeño:



El cálculo de la producción de leche total es:

$$y_{total} = 10 * 15 + \left[ \left( \frac{12+10}{2} * 15 \right) + \left( \frac{18+12}{2} * 14 \right) + \dots + \left( \frac{2+4}{2} * 49 \right) \right] + 48 * 2$$

$$y_{total} = 150 + [(11 * 15) + (15 * 14) + \dots + (3 * 49)] + 48 * 2$$

$$y_{total} = 2627lt$$

Después de estas explicaciones, procedamos a realizar el cálculo de la producción de leche en R-project. Iniciemos creando la función *FuncLag*, como se describió en el capítulo anterior.

```

R Studio

FuncLag=function(LAG)
{
  MenosUltimo=matrix(LAG[1:(length(LAG)-1)]);
  rbind(NA,MenosUltimo)
}
    
```

Ahora llamemos las dos hojas de datos, que en nuestro caso están en formato .csv. El lector podrá construirla e importarla de la forma que desee, como se indicó al inicio de este capítulo. La primera hoja tiene información de las vacas con sus fechas de parto y secado. Hay que

tener en cuenta que las fechas se deben convertir en formato fecha con el comando *as.Date*, con los argumentos nombre de la variable y el formato de la fecha. Nosotros utilizaremos el formato día (%d), mes (%m) y año (%y):

```
R Studio

Vaca =read.csv("RUTA DEL LECTOR/Vaca.csv", sep=";", dec=",", header=TRUE)
Vaca$FechaParto = (as.Date (Vaca$FechaParto, "%d/%m/%y"))
Vaca$FechaSecado = (as.Date (Vaca$FechaSecado, "%d/%m/%y"))
Vaca
```

	Nombre	FechaParto	FechaSecado
1	ITERBIA	2010-01-12	2010-10-08
2	ESCANDIA	2010-01-24	2010-12-28
3	HOLMIA	2010-01-12	2010-11-15

La segunda hoja contiene la información de producciones de leche:

```
R Studio

Leche=read.csv("RUTA DEL LECTOR/LECHE.csv", sep=";", dec=",", header=TRUE)
Leche$FechaControl= (as.Date (Leche$FechaControl, "%d/%m/%y"))
Leche
```

	Nombre	FechaControl	Leche
1	ITERBIA	2010-01-27	16
2	ITERBIA	2010-02-11	18
3	ITERBIA	2010-02-25	22
.			
29	HOLMIA	2010-08-10	4
30	HOLMIA	2010-09-28	2

Veamos la estructura de las dos hojas de datos para verificar los nombres de las variables. Podemos observar que las dos hojas tienen en común la variable *Nombre*, la cual nos permitirá juntar las fechas de parto, fechas de control, fechas de secado y los pesajes, mediante el comando *merge*:

```
R Studio

colnames (Vaca)
colnames (Leche)
```

```
> colnames (Vaca)
[1] "Nombre" "FechaParto" "FechaSecado"
> colnames (Leche)
[1] "Nombre" "FechaControl" "Leche"
```

Juntemos las hojas de datos:

```
R Studio

Pdn=merge (Vaca, Leche, all.x=TRUE, all.y=TRUE)
Pdn
```

	Nombre	FechaParto	FechaSecado	FechaControl	Leche
1	ESCANDIA	2010-01-24	2010-12-28	2010-02-11	22
2	ESCANDIA	2010-01-24	2010-12-28	2010-02-25	20
3	ESCANDIA	2010-01-24	2010-12-28	2010-03-19	24
.					
29	ITERBIA	2010-01-12	2010-10-08	2010-08-10	6
30	ITERBIA	2010-01-12	2010-10-08	2010-09-28	2

Calculemos los días en leche según la fecha de control, y la duración de la lactancia según la fecha de secado:

```
R Studio

Pdn$Del=Pdn$FechaControl-Pdn$FechaParto
Pdn$Del=as.numeric(Pdn$Del)
Pdn$Duración=(Pdn$FechaSecado-Pdn$FechaParto)
Pdn$Duración=as.numeric(Pdn$Duración)
Pdn
```

	Nombre	FechaParto	FechaSecado	FechaControl	Leche	Del	Duración
1	ESCANDIA	2010-01-24	2010-12-28	2010-02-11	22	18	338
2	ESCANDIA	2010-01-24	2010-12-28	2010-02-25	20	32	338
3	ESCANDIA	2010-01-24	2010-12-28	2010-03-19	24	54	338
.							
29	ITERBIA	2010-01-12	2010-10-08	2010-08-10	6	210	269
30	ITERBIA	2010-01-12	2010-10-08	2010-09-28	2	259	269

Es posible que los animales y las fechas de control no estén ordenados, por lo que procedemos a ordenarlos (comando *do.call*), y para efectos de visualización dejamos las variables que utilizaremos en nuestros cálculos de producción de leche (*subset*) para después utilizar la función *FunLag*:



```
Pdn=Pdn[do.call(order,Pl[c("Nombre", "FechaControl")]),]
Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración))
Pdn
```

	Nombre	Leche	Del	Duración
1	ESCANDIA	22	18	338
2	ESCANDIA	20	32	338
3	ESCANDIA	24	54	338
.				
28	ITERBIA	11	190	269
29	ITERBIA	6	210	269
30	ITERBIA	2	259	269

Una vez verifiquemos que las hembras y los controles estén ordenados, identificamos el primer registro de cada vaca, que corresponde al primer control lechero (menor número de días en lactancia), mediante el comando `!duplicated`, que le pondrá `TRUE` (verdadero) al primer registro de cada vaca.

Nótese que al comando `duplicated` le añadimos el signo `!`; si no se incluye, pondrá `FALSE` al primer registro y `TRUE` a los otros registros de la vaca. En este caso creamos la variable `Primero` y al aplicar `ifelse` calculamos la producción de leche al inicio, que está dada por la multiplicación de los días en leche por la producción de leche en el primer control:



```
Pdn$Primero=!duplicated(Pdn$Nombre)
Pdn$Inicio=ifelse(Pdn$Primero=="TRUE", (Pdn$Leche*Pdn$Del), 0)
Pdn
```

	Nombre	Leche	Del	Duración	Primero	Inicio
1	ESCANDIA	22	18	338	TRUE	396
2	ESCANDIA	20	32	338	FALSE	0
.						
10	ESCANDIA	2	325	338	FALSE	0
11	HOLMIA	10	15	307	TRUE	150
.						
20	HOLMIA	2	259	307	FALSE	0
21	ITERBIA	16	15	269	TRUE	240
.						
30	ITERBIA	2	259	269	FALSE	NA

Ahora aplicamos nuevamente el comando `duplicated` para ponerle al último registro de cada hembra (último control lechero realizado) un `TRUE` o `FALSE` que lo distinga de los otros registros de producción de leche, creando la variable `Fin`.

Escogimos utilizar *FALSE* para diferenciar este último registro por vaca, calculamos los días en leche ocurridos desde el último control y el día de secado, creando la variable *Fina* y calculamos la producción de leche en este intervalo en la variable *Final*. Veamos la programación en R-project para obtener la última producción de leche de cada una de las vacas:

```
R Studio

Pdn$Fin=duplicated(Pdn$Nombre, MARGIN = 1, fromLast = TRUE)
Pdn$Fina=ifelse(Pdn$Fin=="FALSE", (Pdn$Duración-Pdn$Del),0)
Pdn$Final=(Pdn$Fina*Pdn$Leche)
Pdn
```

	Nombre	Leche	Del	Duración	Primero	Inicio	Fin	Fina	Final
1	ESCANDIA	22	18	338	TRUE	396	TRUE	0	0
.	.	.	.	.	.	.	.	.	.
10	ESCANDIA	2	325	338	FALSE	0	FALSE	13	26
11	HOLMIA	10	15	307	TRUE	150	TRUE	0	0
.	.	.	.	.	.	.	.	.	.
20	HOLMIA	2	259	307	FALSE	0	FALSE	48	96
21	ITERBIA	16	15	269	TRUE	240	TRUE	0	0
.	.	.	.	.	.	.	.	.	.
29	ITERBIA	6	210	269	FALSE	0	TRUE	0	0
30	ITERBIA	2	259	269	FALSE	0	FALSE	10	20

Para efectos de visualización y manejo de los datos, creamos una nueva hoja de datos con las variables *Nombre*, *Leche*, *Del*, *Duración*, *Inicio* y *Final*, las cuales serán utilizadas en los siguientes pasos. Veamos:

```
R Studio

Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración, Inicio, Final))
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final
1	ESCANDIA	22	18	338	396	0
2	ESCANDIA	20	32	338	0	0
.	.	.	.	.	.	.
29	ITERBIA	6	210	269	0	0
30	ITERBIA	2	259	269	0	20

Empleando la función *FunLac*, en la hoja de datos *Pdn* podemos crear tres variables con la información de la producción de leche del control anterior, los días en leche y el rezago del nombre, con el propósito de estimar las producciones parciales entre los intervalos de control

lechero. Veamos la programación en R-project:

```
R Studio

Pdn$LagL=FuncLag (Pdn$Leche)
Pdn$LagD=FuncLag (Pdn$Del)
Pdn$LagN=FuncLag (Pdn$Nombre)
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	LagL	LagD	LagN
1	ESCANDIA	22	18	338	396	0	NA	NA	<NA>
2	ESCANDIA	20	32	338	0	0	22	18	ESCANDIA
.	.	.	.	.	.	.	.	.	.
29	ITERBIA	6	210	269	0	0	11	190	ITERBIA
30	ITERBIA	2	259	269	0	20	6	210	ITERBIA

A continuación creamos tres columnas: *prom* que es el promedio de la producción de leche entre controles, *dias* que es el intervalo de tiempo entre controles y *Mitad* que es la producción parcial de leche en cada periodo, veamos:

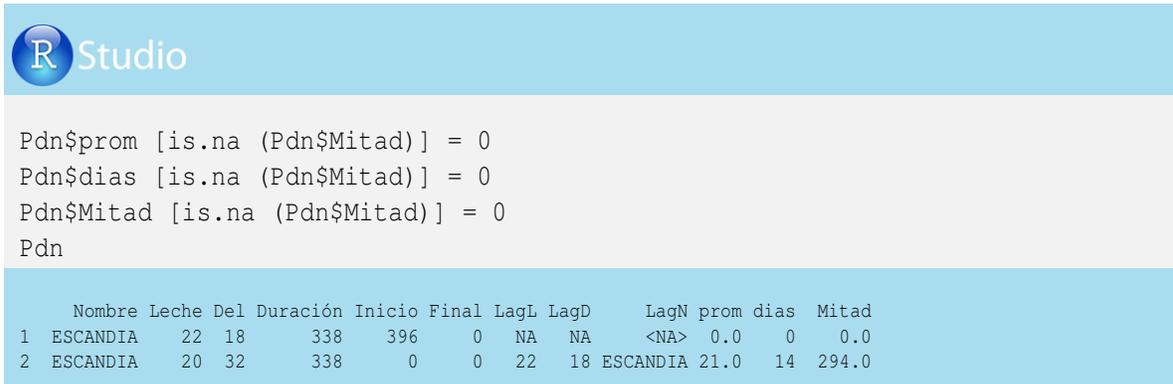
```
R Studio

Pdn$prom=ifelse (Pdn$Nombre==Pdn$LagN, ((Pdn$Leche+Pdn$LagL) / 2) , 0)
Pdn$dias=ifelse (Pdn$Nombre==Pdn$LagN, Pdn$Del-Pdn$LagD, 0)
Pdn$Mitad= (Pdn$prom*Pdn$dias)
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	LagL	LagD	LagN	prom	dias	Mitad
1	ESCANDIA	22	18	338	396	0	NA	NA	<NA>	NA	NA	NA
2	ESCANDIA	20	32	338	0	0	22	18	ESCANDIA	21.0	14	294.0
3	ESCANDIA	24	54	338	0	0	20	32	ESCANDIA	22.0	22	484.0
.	.	.	.	.	.	.	.	.	.	.	.	.
21	ITERBIA	16	15	269	240	0	2	259	HOLMIA	0.0	0	0.0
.	.	.	.	.	.	.	.	.	.	.	.	.
29	ITERBIA	6	210	269	0	0	11	190	ITERBIA	8.5	20	170.0
30	ITERBIA	2	259	269	0	20	6	210	ITERBIA	4.0	49	196.0

En esta tabla de datos podemos observar que en el primer registro el comando *ifelse* aplicado en la programación anterior no funcionó, porque la variable *LagN* tiene un *NA*, lo que a su vez genera un *NA* en las variables *prom*, *dias* y *Mitad*. Para convertir ese *NA* en 0 se emplea

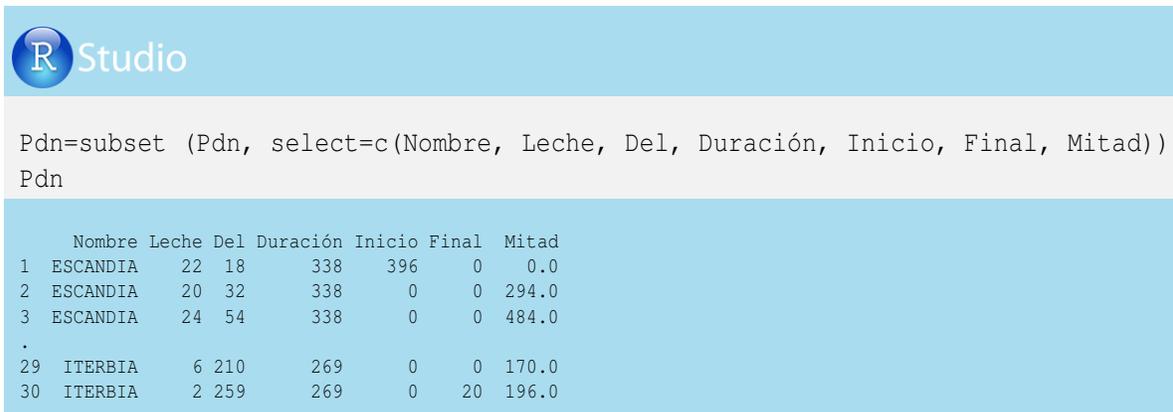
el comando *is.na*, como se indica a continuación:



```
Pdn$prom [is.na (Pdn$Mitad)] = 0
Pdn$dias [is.na (Pdn$Mitad)] = 0
Pdn$Mitad [is.na (Pdn$Mitad)] = 0
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	LagL	LagD	LagN	prom	dias	Mitad
1	ESCANDIA	22	18	338	396	0	NA	NA	<NA>	0.0	0	0.0
2	ESCANDIA	20	32	338	0	0	22	18	ESCANDIA	21.0	14	294.0

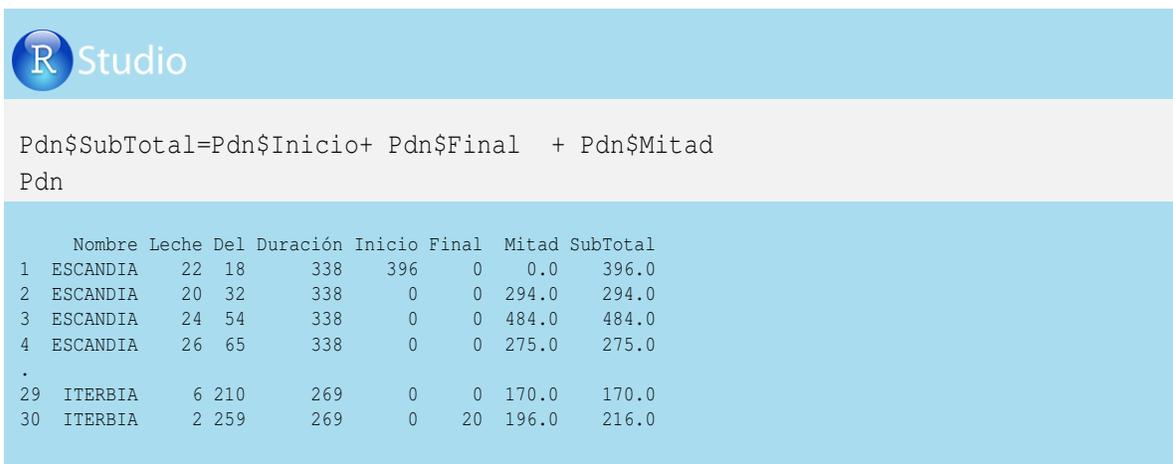
Ahora dejamos las variables que necesitamos para los siguientes cálculos:



```
Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración, Inicio, Final, Mitad))
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	Mitad
1	ESCANDIA	22	18	338	396	0	0.0
2	ESCANDIA	20	32	338	0	0	294.0
3	ESCANDIA	24	54	338	0	0	484.0
.							
29	ITERBIA	6	210	269	0	0	170.0
30	ITERBIA	2	259	269	0	20	196.0

A continuación calculamos el subtotal de la producción de leche por fila de la hoja de datos. Veamos la programación y la tabla de resultados:

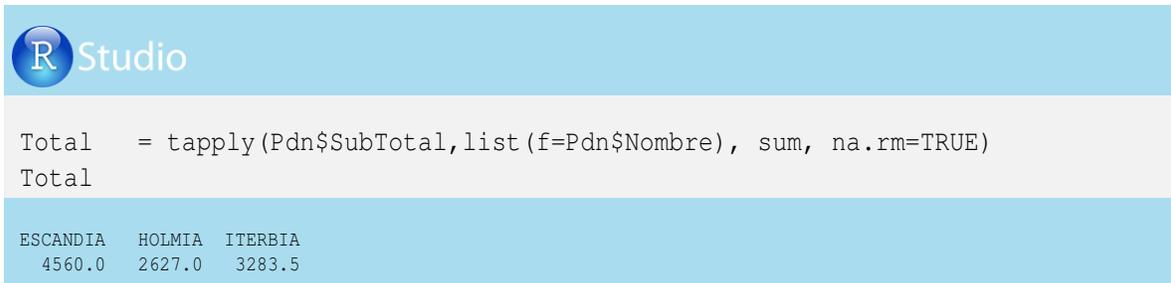


```
Pdn$SubTotal=Pdn$Inicio+ Pdn$Final + Pdn$Mitad
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	Mitad	SubTotal
1	ESCANDIA	22	18	338	396	0	0.0	396.0
2	ESCANDIA	20	32	338	0	0	294.0	294.0
3	ESCANDIA	24	54	338	0	0	484.0	484.0
4	ESCANDIA	26	65	338	0	0	275.0	275.0
.								
29	ITERBIA	6	210	269	0	0	170.0	170.0
30	ITERBIA	2	259	269	0	20	196.0	216.0

Recordemos que se pueden utilizar diferentes comandos y secuencias para calcular la producción de leche. En este ejercicio hemos tratado de usar diferentes comandos y mostrar su utilidad para otros procedimientos. Ahora utilizaremos el comando *tapply* que nos permitirá

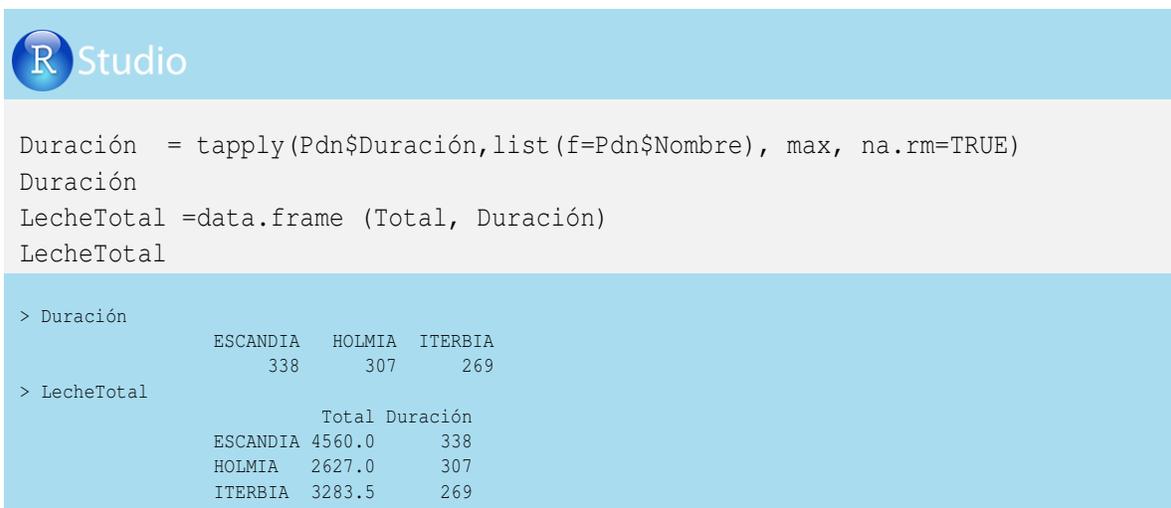
generar otras hojas de datos con información específica de la hoja original. En este caso emplearemos dicho comando para calcular la suma de los distintos periodos por vaca, y con los argumentos *sum* y *list(f = Pdn\$Nombre)* podemos calcular la producción de leche total de cada vaca. Veamos:



```
Total = tapply(Pdn$SubTotal, list(f=Pdn$Nombre), sum, na.rm=TRUE)
Total
```

ESCANDIA	HOLMIA	ITERBIA
4560.0	2627.0	3283.5

De manera similar al procedimiento para calcular la producción total de leche, utilizaremos el comando *tapply* para calcular la duración de la lactancia de cada una de las vacas, empleando el argumento *max*. Finalmente, construiremos una hoja de datos con los resultados finales de producción total de leche y duración de la lactancia:



```
Duración = tapply(Pdn$Duración, list(f=Pdn$Nombre), max, na.rm=TRUE)
Duración
LecheTotal =data.frame (Total, Duración)
LecheTotal
```

```
> Duración
```

	ESCANDIA	HOLMIA	ITERBIA
	338	307	269

```
> LecheTotal
```

	Total	Duración
ESCANDIA	4560.0	338
HOLMIA	2627.0	307
ITERBIA	3283.5	269

El lector puede realizar otro tipo de programaciones o adecuaciones para la producción de leche o para sus constituyentes (grasa, proteína, lactosa, etc.).

#### 4.6.3. Cálculo de la producción de leche hasta los 305 días

Es muy común hacer el cálculo de la producción de leche hasta los 305 días, pues es una medida ampliamente utilizada en razas especializadas y permite hacer una comparación de animales en un periodo de lactancia adecuado.

La producción hasta 305 días se confunde con la producción de leche ajustada a 305 días, que es una forma de proyectar la lactancia cuando una vaca no la ha terminado o cuando sale del hato sin llegar a los 305 días de lactancia. En esta sección trabajaremos con la

producción de leche hasta los 305 días (denotada como  $y_{305}$ ), en la cual se presentan tres posibles situaciones:

♣ **La vaca terminó su lactancia antes de los 305 días:**

En este caso la producción de leche hasta los 305 días es igual a la producción de leche total (caso de la vaca *ITERBIA*) y no es necesario hacer ajuste a los 305 días porque fisiológicamente la hembra terminó su lactancia. Hay que tener cuidado con este ajuste porque puede conducir a engaños en la producción real de leche.

$$y_{305} = y_{total}$$

Retomando la producción de leche total y la duración de lactancia de la vaca *ITERBIA*, presentada en la sección anterior, tendremos:

$$y_{total} = y_{305} = 3283.5 \text{ lt, con una duración de 269 días en lactancia.}$$

♣ **La vaca tiene controles de leche después de los 305 días:**

En este caso se promedian los controles que involucran los 305 días (los controles inmediatamente anterior e inmediatamente posterior a los 305 días) y se multiplican por los días que pasaron desde el control inmediatamente anterior a los 305 días e inmediatamente posterior a los 305 días, con las siguientes especificaciones:

$$y_{305} = P_1 D_1 + \left[ \sum_{i=2}^a \frac{P_i + P_{i-1}}{2} (D_i - D_{i-1}) \right] + (305 - D_a) * \left( \frac{P_d + P_a}{2} \right)$$

Donde los controles  $a$  y  $d$  son los que ocurrieron inmediatamente antes e inmediatamente después de los 305 días.

En el caso de *ESCANDIA*, la cual tuvo una lactancia de 338 días, sus pesajes inmediatamente anterior y posterior ocurrieron en los días 247 y 325, con 6 y 2 lt, respectivamente. Emplearemos la fórmula anterior para estimar la producción de leche de *ESCANDIA* hasta los 305 días:

$$y_{305} = 22 * 18 + \left[ \left( \frac{20+22}{2} * 14 \right) + \left( \frac{24+20}{2} * 22 \right) + \dots + \left( \frac{6+6}{2} * 49 \right) \right] + (305 - 247) * \frac{2+6}{2}$$

$$y_{305} = 396 + [(21 * 14) + (22 * 22) + \dots + (6 * 49)] + 58 * 4$$

$$y_{305} = 4454 \text{ lt}$$

♣ **La vaca tiene su último control antes de los 305 días, pero se seca después de los 305 días:**

Es el caso de la vaca *HOLMIA*, cuyo último control ocurrió el día 259, pero su duración de lactancia fueron 307 días. En este caso se tiene en cuenta la producción de leche del último pesaje y se multiplica por la diferencia entre 305 días y los días en leche del último control, con las siguientes especificaciones.

$$y_{305} = P_1 D_1 + \left[ \sum_{i=2}^n \frac{P_i + P_{i-1}}{2} (D_i - D_{i-1}) \right] + (305 - D_u) * P_u$$

Ahora calcularemos la producción de leche de la vaca *HOLMIA* hasta 305 días:

$$y_{305} = 10 * 15 + \left[ \left( \frac{12+10}{2} * 15 \right) + \left( \frac{18+12}{2} * 14 \right) + \dots + \left( \frac{2+4}{2} * 49 \right) \right] + (305 - 259) * 2$$

$$y_{305} = 150 + \left[ (11 * 15) + (15 * 14) + \dots + (3 * 49) \right] + 46 * 2$$

$$y_{305} = 2623lt$$

Para realizar el cálculo de la producción de leche hasta 305 días en R-project, tomaremos parte de la programación del ejercicio de la sección anterior.



```
FuncLag=function(LAG)
{
  MenosUltimo=matrix(LAG[1:(length(LAG)-1)]);
  rbind(NA,MenosUltimo)
}
Leche=read.csv("RUTA DEL LECTOR/LECHE.csv", sep=";", dec=",", header=TRUE)
Leche$FechaControl= (as.Date (Leche$FechaControl, "%d/%m/%y"))
Vaca =read.csv("RUTA DEL LECTOR/Vaca.csv", sep=";", dec=",", header=TRUE)
Vaca$FechaParto= (as.Date (Vaca$FechaParto, "%d/%m/%y"))
Vaca$FechaSecado= (as.Date (Vaca$FechaSecado, "%d/%m/%y"))
Pdn=merge (Vaca, Leche, all.x=TRUE, all.y=TRUE)
Pdn$Del=Pdn$FechaControl-Pdn$FechaParto
Pdn$Del=as.numeric(Pdn$Del)
Pdn=Pdn[do.call(order,P1[c("Nombre","FechaControl")]),]
Pdn$Duración=(Pdn$FechaSecado-Pdn$FechaParto)
Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración))
Pdn$Primero=!duplicated(Pdn$Nombre)
Pdn$Inicio=ifelse (Pdn$Primero=="TRUE", (Pdn$Leche*Pdn$Del),0)
Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración, Inicio))
```

Veamos la hoja de datos que tenemos, para luego continuar con el cálculo de la producción de leche hasta 305 días:

	Nombre Leche	Del	Duración	Inicio
1	ESCANDIA	22 18 days	338 days	396
2	ESCANDIA	20 32 days	338 days	0
.	.	.	.	.
10	ESCANDIA	2 325 days	338 days	0
11	HOLMIA	10 15 days	307 days	150
.	.	.	.	.
20	HOLMIA	2 259 days	307 days	0
21	ITERBIA	16 15 days	269 days	240
.	.	.	.	.
29	ITERBIA	6 210 days	269 days	0
30	ITERBIA	2 259 days	269 days	0

Apliquemos nuestra función de rezagos *FuncLag* a las variables de producción de leche, días en leche y nombre de la vaca, mediante la siguiente programación en R-project:

```
Pdn$LagL=FuncLag(Pdn$Leche)
Pdn$LagD=FuncLag(Pdn$Del)
Pdn$LagN=FuncLag(Pdn$Nombre)
Pdn
```

	Nombre Leche	Del	Duración	Inicio	LagL	LagD	LagN
1	ESCANDIA	22 18 days	338 days	396	NA	NA	<NA>
2	ESCANDIA	20 32 days	338 days	0	22	18	ESCANDIA
3	ESCANDIA	24 54 days	338 days	0	20	32	ESCANDIA
.	.	.	.	.	.	.	.
30	ITERBIA	2 259 days	269 days	0	6	210	ITERBIA

Emplearemos el comando *duplicated* para generar una columna que indique como falso al último control de leche que tuvo la hembra en ese parto. Posteriormente, utilizaremos el comando *ifelse* para calcular la producción de leche desde el último control hasta el día de secado, si se cumple la condición de que los días en lactancia son menos de 305 días. En este caso, se calculará la producción parcial al final de la lactancia de la vaca *ITERBIA* dada por  $2 * (269 - 259) = 20lt$ .



```
Pdn$Fin=duplicated(Pdn$Nombre, MARGIN = 1, fromLast = TRUE)
Pdn$Ulti1=ifelse(Pdn$Fin=="FALSE" & Pdn$Duración<305,
                 ((Pdn$Duración-Pdn$Del)*Pdn$Leche), 0)
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	LagL	LagD	LagN	Fin	Ulti1
1	ESCANDIA	22	18 days	338 days	396	NA	NA	<NA>	TRUE	0
2	ESCANDIA	20	32 days	338 days	0	22	18	ESCANDIA	TRUE	0
.										
30	ITERBIA	2	259 days	269 days	0	6	210	ITERBIA	FALSE	20

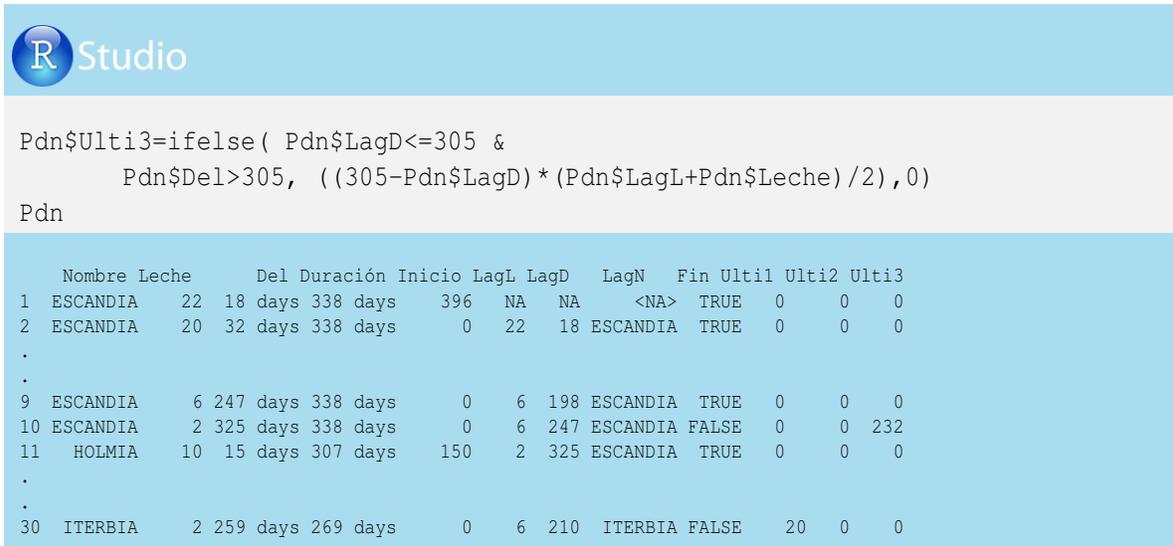
A continuación calcularemos la producción de leche al final de la lactancia, de una vaca cuyo último control fue antes de los 305 días pero cuya fecha de secado fue posterior a los 305 días (caso de la vaca *HOLMIA*); veamos:



```
Pdn$Ulti2=ifelse(Pdn$Fin=="FALSE" & Pdn$Del<=305 &
                 Pdn$Duración>305, ((305-Pdn$Del)*Pdn$Leche), 0)
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	LagL	LagD	LagN	Fin	Ulti1	Ulti2
1	ESCANDIA	22	18 days	338 days	396	NA	NA	<NA>	TRUE	0	0
2	ESCANDIA	20	32 days	338 days	0	22	18	ESCANDIA	TRUE	0	0
.											
19	HOLMIA	4	210 days	307 days	0	4	190	HOLMIA	TRUE	0	0
20	HOLMIA	2	259 days	307 days	0	4	210	HOLMIA	FALSE	0	92
21	ITERBIA	16	15 days	269 days	240	2	259	HOLMIA	TRUE	0	0
.											
30	ITERBIA	2	259 days	269 days	0	6	210	ITERBIA	FALSE	20	0

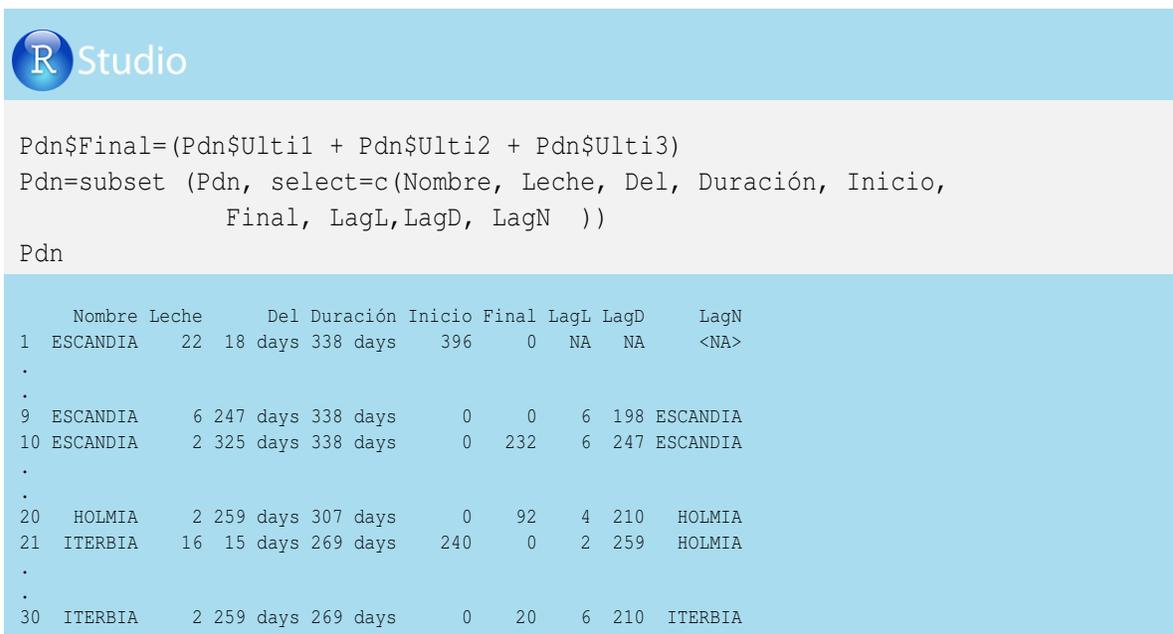
Para las vacas que tienen controles posteriores a 305 días, realizamos el siguiente cálculo del periodo ocurrido inmediatamente anterior y el periodo inmediatamente posterior a los 305 días:



```
Pdn$Ulti3=ifelse( Pdn$LagD<=305 &
  Pdn$Del>305, ((305-Pdn$LagD) * (Pdn$LagL+Pdn$Leche) /2), 0)
Pdn
```

	Nombre Leche	Del	Duración	Inicio	LagL	LagD	LagN	Fin	Ulti1	Ulti2	Ulti3	
1	ESCANDIA	22	18 days	338 days	396	NA	NA	<NA>	TRUE	0	0	0
2	ESCANDIA	20	32 days	338 days	0	22	18	ESCANDIA	TRUE	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
9	ESCANDIA	6	247 days	338 days	0	6	198	ESCANDIA	TRUE	0	0	0
10	ESCANDIA	2	325 days	338 days	0	6	247	ESCANDIA	FALSE	0	0	232
11	HOLMIA	10	15 days	307 days	150	2	325	ESCANDIA	TRUE	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
30	ITERBIA	2	259 days	269 days	0	6	210	ITERBIA	FALSE	20	0	0

Ahora crearemos la variable *Final* con la suma de las tres variables creadas anteriormente (*Ulti1*, *Ulti2* y *Ulti3*) y de esta forma obtendremos la producción en el periodo que involucra los 305 días. Luego generaremos una nueva hoja de datos con las variables que nos interesan. Veamos la programación en R-project:



```
Pdn$Final=(Pdn$Ulti1 + Pdn$Ulti2 + Pdn$Ulti3)
Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración, Inicio,
  Final, LagL,LagD, LagN ))
Pdn
```

	Nombre Leche	Del	Duración	Inicio	Final	LagL	LagD	LagN	
1	ESCANDIA	22	18 days	338 days	396	0	NA	NA	<NA>
.	.	.	.	.	.	.	.	.	.
9	ESCANDIA	6	247 days	338 days	0	0	6	198	ESCANDIA
10	ESCANDIA	2	325 days	338 days	0	232	6	247	ESCANDIA
.	.	.	.	.	.	.	.	.	.
20	HOLMIA	2	259 days	307 days	0	92	4	210	HOLMIA
21	ITERBIA	16	15 days	269 days	240	0	2	259	HOLMIA
.	.	.	.	.	.	.	.	.	.
30	ITERBIA	2	259 days	269 days	0	20	6	210	ITERBIA

Finalmente, calcularemos los promedios de leche, los intervalos entre controles y las producciones de leche entre controles, como se realizó en el ejercicio del cálculo de la producción de leche total en la sección anterior. Destacamos que únicamente se tienen en cuenta controles

anteriores a los 305 días:

```
R Studio

Pdn$prom=ifelse(Pdn$N==Pdn$LagN & Pdn$Del <=305, ((Pdn$Leche+Pdn$LagL)/2), 0)
Pdn$dias=ifelse(Pdn$Nombre==Pdn$LagN, Pdn$Del-Pdn$LagD, 0)
Pdn$Mitad=(Pdn$prom*Pdn$dias)
Pdn$Mitad [is.na (Pdn$Mitad)] = 0
Pdn
```

	Nombre	Leche	Del	Duración	Inicio	Final	LagL	LagD	LagN	prom	dias	Mitad
1	ESCANDIA	22	18 days	338 days	396	0	NA	NA	<NA>	NA	NA	0.0
2	ESCANDIA	20	32 days	338 days	0	0	22	18	ESCANDIA	21.0	14	294.0
.												
29	ITERBIA	6	210 days	269 days	0	0	11	190	ITERBIA	8.5	20	170.0
30	ITERBIA	2	259 days	269 days	0	20	6	210	ITERBIA	4.0	49	196.0

Para terminar el ejercicio, emplearemos el comando *tapply*, de la misma forma que lo hicimos para el cálculo de la producción de leche total. De esta manera obtendremos la producción de leche hasta 305 días. Veamos la programación en R-project para tal fin:

```
R Studio

Pdn=subset (Pdn, select=c(Nombre, Leche, Del, Duración, Inicio, Final, Mitad))Pdn
Pdn$SubTotal=Pdn$Inicio+ Pdn$Final + Pdn$Mitad
L305= tapply(Pdn$SubTotal, list(f=Pdn$Nombre), sum, na.rm=TRUE)
L305
```

ESCANDIA	HOLMIA	ITERBIA
4454.0	2623.0	3283.5

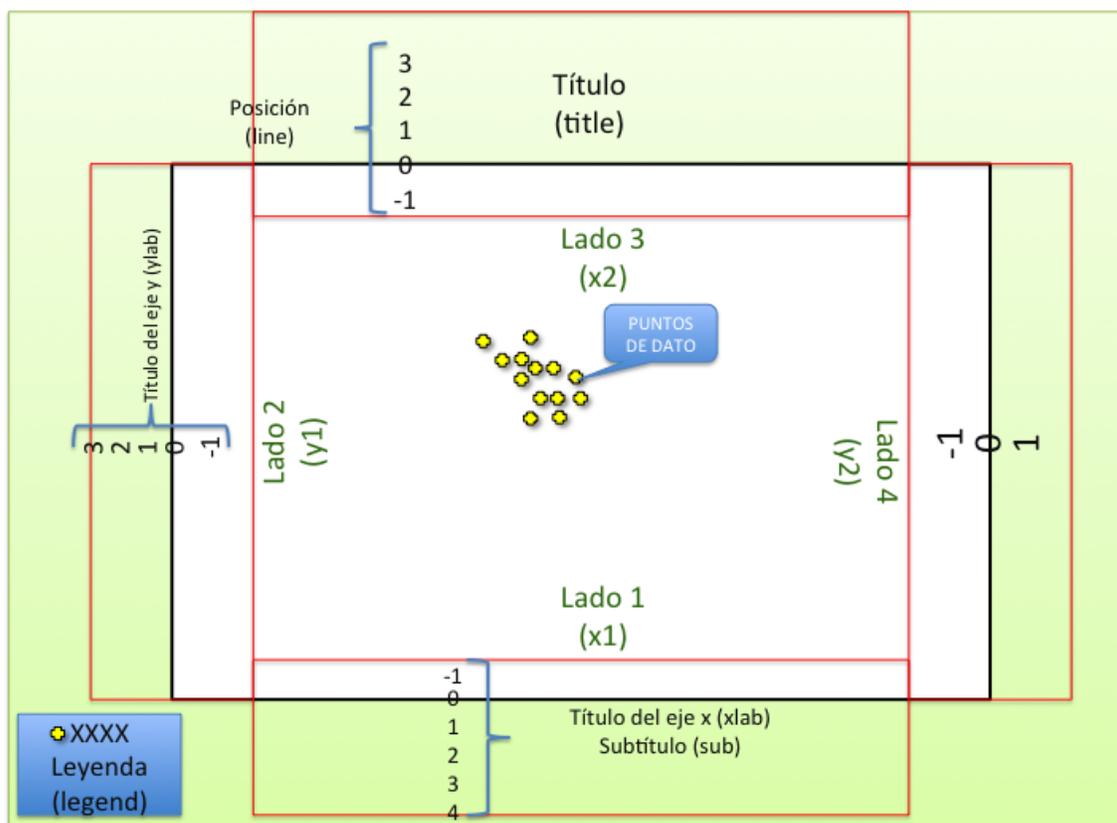
## Capítulo 5

# Montaje de gráficos

### 5.1. Generalidades

El área total de un gráfico de R-project comprende el título principal (*title*), los títulos de los ejes (*xlab* y *ylab*), los subtítulos (*sub*), la leyenda (*legend*) y el área de trazado donde están los puntos de dato y todas las especificaciones que ilustran los datos dentro de los cuatro lados (*x1*, *x2*, *y1* y *y2*).

En la siguiente ilustración podemos ver la localización de los componentes de un gráfico. La secuencia de números (-1 hasta 4) corresponde a las posibles posiciones o distancias al margen de la página donde pueden localizarse los textos de los títulos:



En la mayoría de las librerías de R-project se manejan diferentes tipos de gráficos que permiten potencializar los análisis realizados. Este capítulo se enfocará en la obtención de los gráficos más comunes con los comandos *plot*, *stem*, *hist*, *scatterplot*, *pie*, *boxplot*, *qqplot* y *dotchat*) y con la manipulación de líneas, textos, escalas y puntos de dato.

Para una mejor ilustración emplearemos hojas de datos que nos permitirán realizar los gráficos. A continuación se presenta la primera hoja que utilizaremos. Esta contiene información sobre el peso (en kilogramos), la estatura (medida en clases de *frame*), el espesor de grasa de cadera (en centímetros) y el tamaño del cuerno (en centímetros) de vacunos al momento del sacrificio.

ANIMAL	PESO(kg)	GRASA(cm)	CUERNO(cm)	FRAME(clases)
1	406	0.456	9.66	3
2	396	0.403	10.09	4
3	417	0.524	11.56	4
4	396	0.414	11.82	4
5	402	0.442	8.58	4
6	393	0.399	11.25	4
7	403	0.466	10.23	4
8	408	0.498	9.74	4
9	417	0.539	10.81	4
10	398	0.448	10.32	4

Veamos la construcción de la hoja de datos en R-project:

```

R Studio

peso=c(406,396,417,396,402,393,403,408,417,398)
grasa=c(0.456,0.403,0.524,0.414,0.442,0.399,0.466,0.498,0.539,0.448)
cuerno=c(9.66,10.09,11.56,11.82,8.58,11.25,10.23,9.74,10.81,10.32)
frame=c(3,4,4,4,4,4,4,4,4,4)
machos=data.frame(peso, grasa, cuerno, frame)
machos

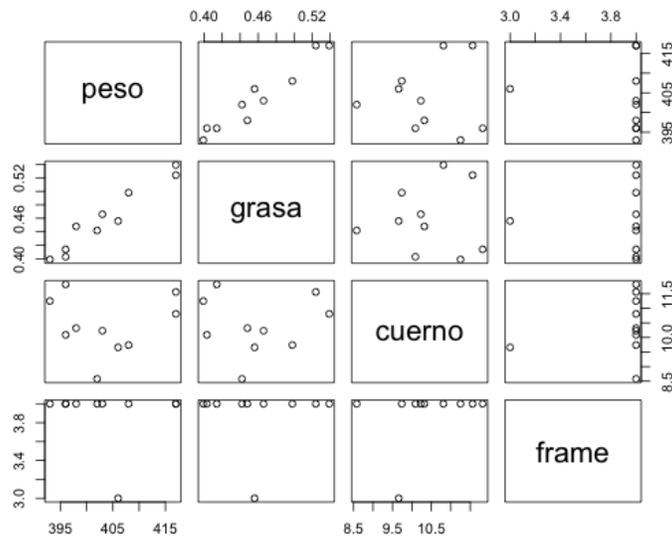
  peso grasa cuerno frame
1  406 0.456   9.66     3
2  396 0.403  10.09     4
3  417 0.524  11.56     4
.
9  417 0.539  10.81     4
10 398 0.448  10.32     4

```

## 5.2. Comando *plot* para generar gráficos

El comando *plot* se utiliza para crear figuras en las que se presenta la relación de las variables numéricas de la hoja de datos. Veamos el resultado de este comando para la hoja de datos *machos*:

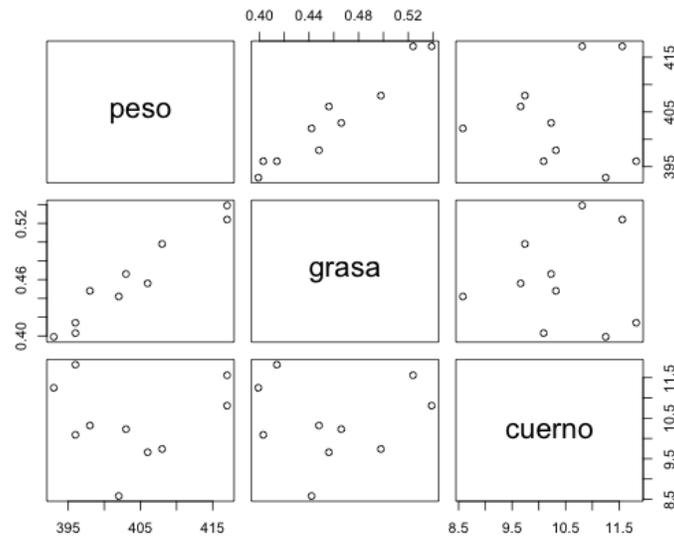
```
R Studio  
plot(machos)
```



Este gráfico nos da una idea de la estructura y de la relación de variables. Por ejemplo: a medida que el peso aumenta, el espesor de grasa también aumenta, y el tamaño del cuerno no está relacionado con el peso y la grasa.

Ahora graficaremos la relación entre las tres primeras variables de la hoja de datos *machos*:

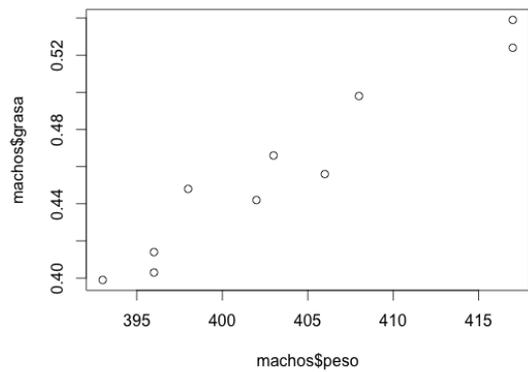
```
R Studio  
plot(machos[1:3])
```



El comando *plot* también permite realizar gráficos de relación entre dos variables de una hoja de datos, así:

```

R Studio
plot(machos$peso,machos$grasa)
    
```

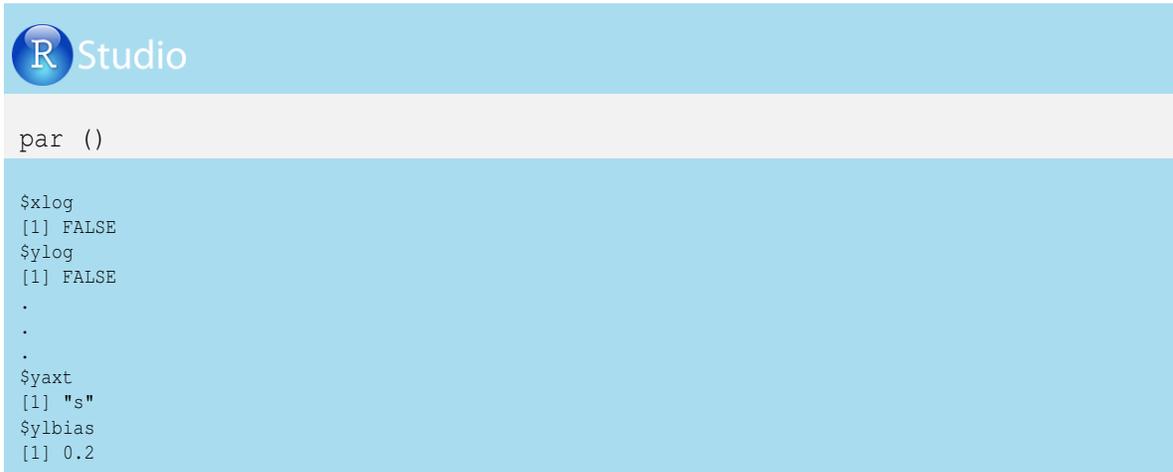


### 5.3. Comando *par* para especificaciones previas de los gráficos

El comando *par* es una función auxiliar de los comandos de generación de gráficos y permite modificar y establecer los distintos aspectos de los mismos, como color, tipo de letra o posiciones.

Para ver todas las especificaciones de los gráficos se utiliza el comando *par* sin argumento.

Realizaremos este procedimiento, pero no entraremos a detallar sus especificaciones, porque la mayoría de estas se trabajarán en las siguientes secciones:



```

R Studio

par ()

$xlog
[1] FALSE
$ylog
[1] FALSE
.
.
.
$yaxt
[1] "s"
$ylbias
[1] 0.2
    
```

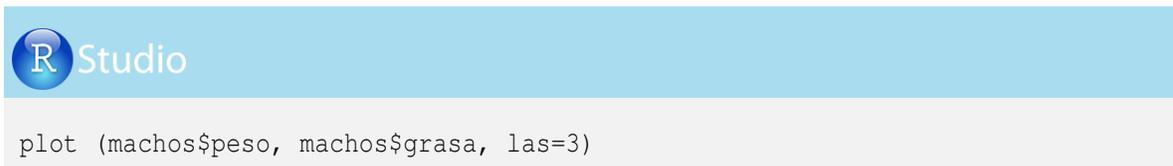
El comando *par* fija las características de los gráficos siguientes. Sin embargo, se pueden escribir las especificaciones dentro de los argumentos de los comandos de graficación, para modificar únicamente al gráfico solicitado. Por ejemplo, miremos el argumento *las* = dentro del comando *par* para fijarlo en los gráficos siguientes, o *las* = dentro del comando *plot* para modificar solamente el gráfico plot de la relación entre grasa y peso de los animales:



```

R Studio

par(las=3)
plot (machos$peso, machos$grasa)
    
```



```

R Studio

plot (machos$peso, machos$grasa, las=3)
    
```

### 5.3.1. Argumentos para localización y posición

El comando *par* con el argumento *las*, seguido de = 0, = 1, = 2 o = 3, determina la posición de giro de los valores de la escala de los ejes, así:

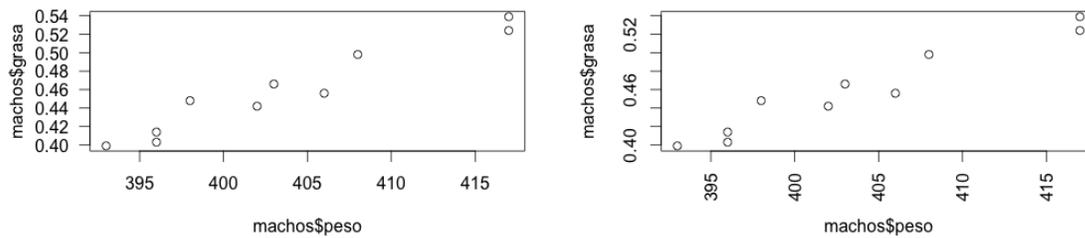
Argumento	Posición en el eje y	Posición en el eje x
<i>las</i> = 0	vertical	horizontal
<i>las</i> = 1	horizontal	horizontal
<i>las</i> = 2	horizontal	vertical
<i>las</i> = 3	vertical	vertical

Veamos la programación en R-project, donde especificamos para el primer gráfico que los nombres de los ejes estén en forma horizontal (*las = 1*) y para el segundo gráfico en forma vertical (*las = 3*):

```

R Studio

par(las=1)
plot(machos$Peso,machos$Grasa)
par(las=3)
plot(machos$Peso,machos$Grasa)
    
```



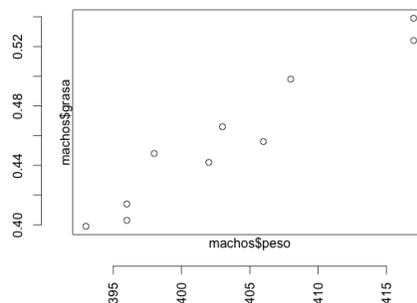
Como vimos al inicio de este capítulo, hay unas distancias que involucran el área de trazado (desde  $-1$  a  $4$ ). Con el comando *mgp* seguido de *c(valor, valor, valor)* podemos determinar o especificar la distancia que asumirán los nombres, los valores y las líneas de los ejes *x* y *y*, respectivamente.

En el siguiente ejemplo especificaremos que el nombre del eje esté cercano al área de gráfico (0), que los valores estén muy lejanos (3) y las líneas estén entre el nombre y los valores (2):

```

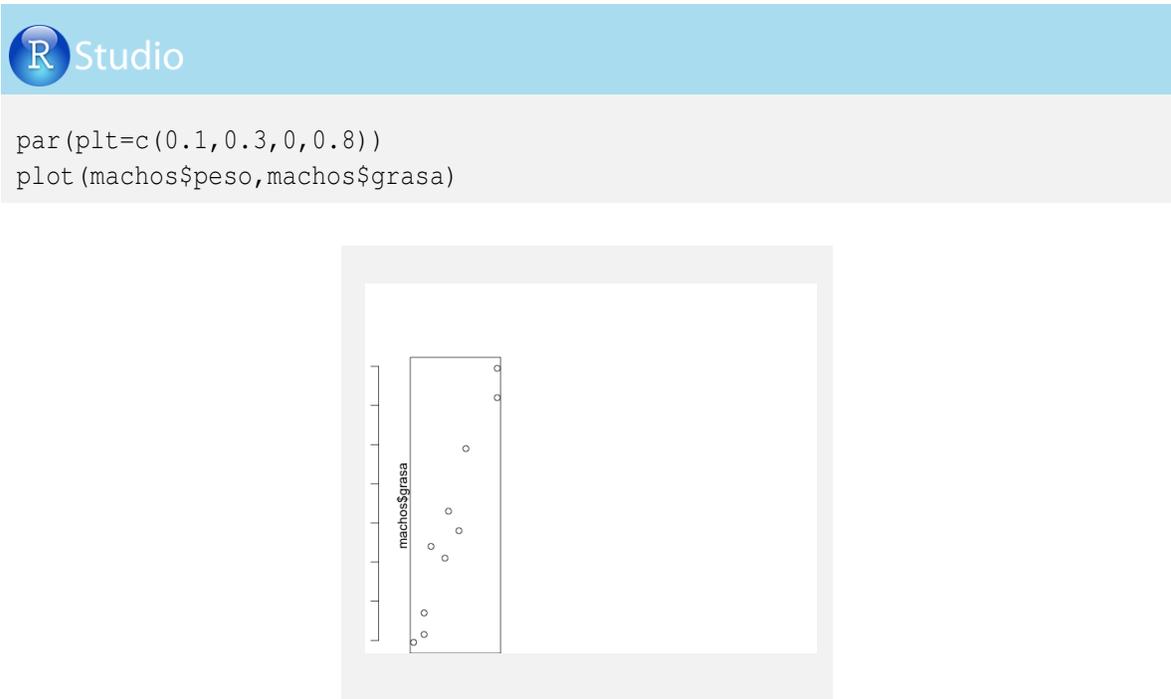
R Studio

par(mgp=c(0, 3, 2))
plot(machos$peso,machos$grasa)
    
```

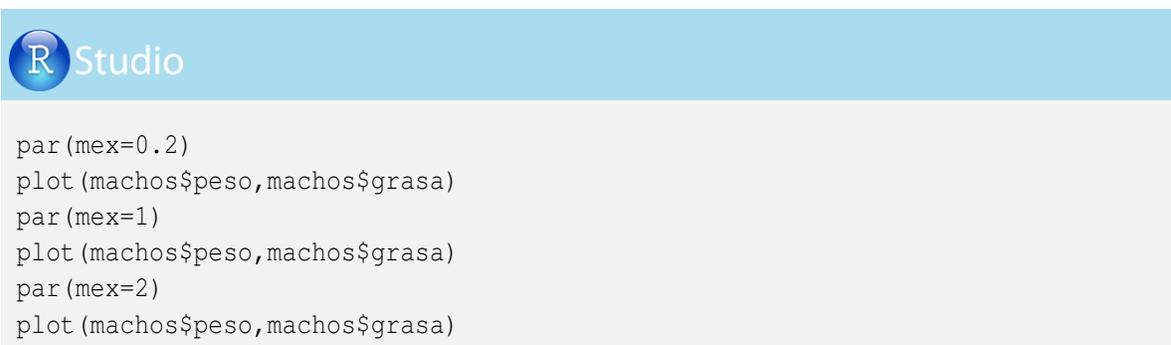


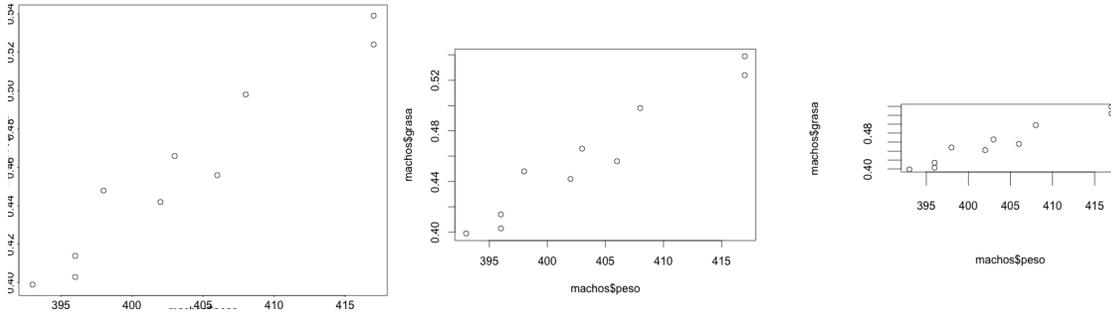
El área del trazado puede cambiar de tamaño y de posición dentro del área total del gráfico. Para esto se utiliza el argumento *plt* con las coordenadas *c*(valor de la parte inferior, valor de la parte superior, valor a la derecha, valor a la izquierda). Todos los valores enunciados deben estar entre 0 y 1.

En el siguiente ejemplo, el gráfico estará en la parte izquierda del área total, ocupando desde el 10% de la margen izquierda hasta el 30% de la margen hacia la derecha. También estará desplegado en el piso (0%) y llegará cerca a la parte superior de la figura (80%).



El argumento *mex* determina el tamaño del área de trazado con relación al área total. Si el valor es cercano a 0, los ejes estarán muy cerca a los límites del área total, y si es cercano a 3, el área de trazado estará reducido al centro del área total. Este argumento procura mantener la distancia de los nombres de los ejes. Veamos el ejemplo con *mex = 0.2*, *mex = 1* y *mex = 2*:



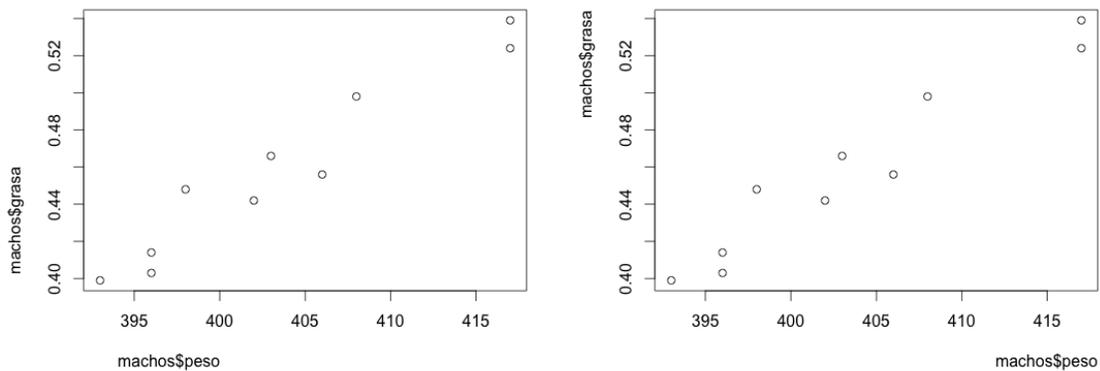


### 5.3.2. Localización de los nombres de los ejes

El argumento *adj* permite alinear los textos de izquierda (0) a derecha (1). Veamos el funcionamiento de este argumento mediante dos ejemplos: el primero alineando los textos completamente a la izquierda y el segundo con la alineación del texto a la derecha de los ejes cartesianos:

```

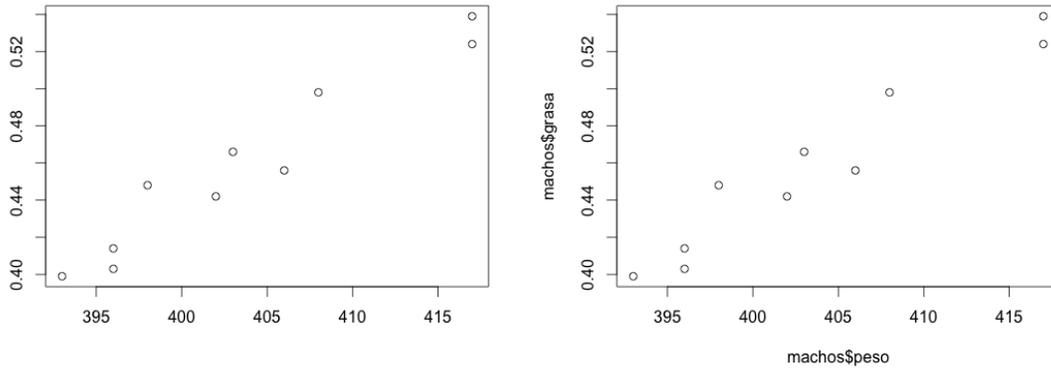
R Studio
par (adj=0.1)
plot (machos$peso,machos$grasa)
par (adj=1)
plot (machos$peso,machos$grasa)
    
```



Otro argumento útil es *ann*, al cual se le agrega = *TRUE* o = *FALSE* para incluir o excluir los nombres de los ejes. Veamos un ejemplo en R-project:

```

R Studio
par (ann=FALSE)
plot (machos$peso,machos$grasa)
par (ann=TRUE)
plot (machos$peso,machos$grasa)
    
```

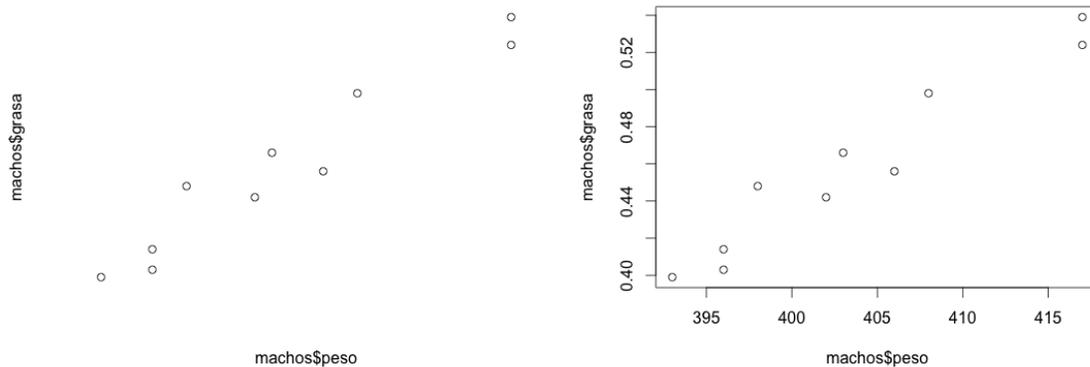


### 5.3.3. Líneas de los ejes

El argumento *axes* seguido de *= TRUE* o *= FALSE* incluye o excluye todas las líneas de los ejes del gráfico generado; veamos la implementación en R-project:



```
plot(machos$peso, machos$grasa, axes=FALSE)
plot(machos$peso, machos$grasa, axes=TRUE)
```



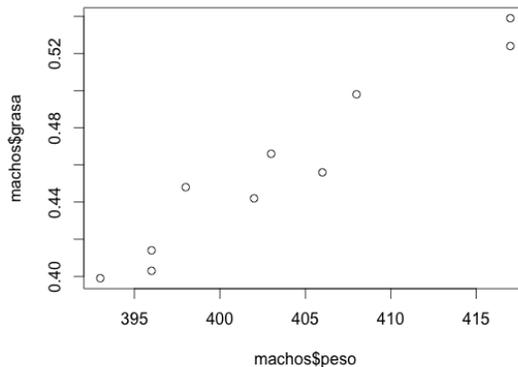
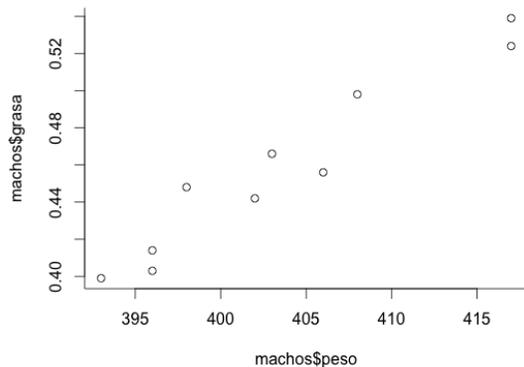
El argumento *bty* permite incluir o excluir las líneas de los ejes. Para que aparezcan las líneas de los ejes *y1* y *x1*, use *bty = "l"*, para los ejes *y2* y *x2* use *bty = "7"*, para *y1*, *x1* y *x2* use *bty = "c"*, para *y1*, *y2* y *x1* use *bty = "u"*, para *y2*, *x1* y *x2* use *bty = "]"* y para los cuatro

lados use el *bty* = "o". Veamos la implementación de los casos enunciados:

```

R Studio
par (bty="u")
plot (machos$peso,machos$grasa)
par (bty="c")
plot (machos$peso,machos$grasa)

```



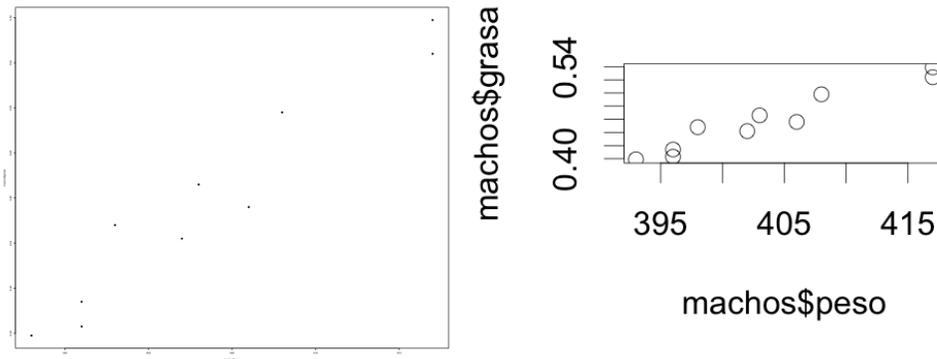
### 5.3.4. Tamaño de letra y valores de los ejes

El argumento *cex* permite modificar el tamaño relativo de los textos y los valores de los ejes con respecto al área de trazado. Veamos dos ejemplos de su implementación en R-project: en el primero hacemos imperceptible los textos y los valores de los ejes con *cex* = 0.2, caso contrario al siguiente gráfico con *cex* = 2:

```

R Studio
par (cex=0.2)
plot (machos$peso,machos$grasa)
par (cex=2)
plot (machos$peso,machos$grasa)

```

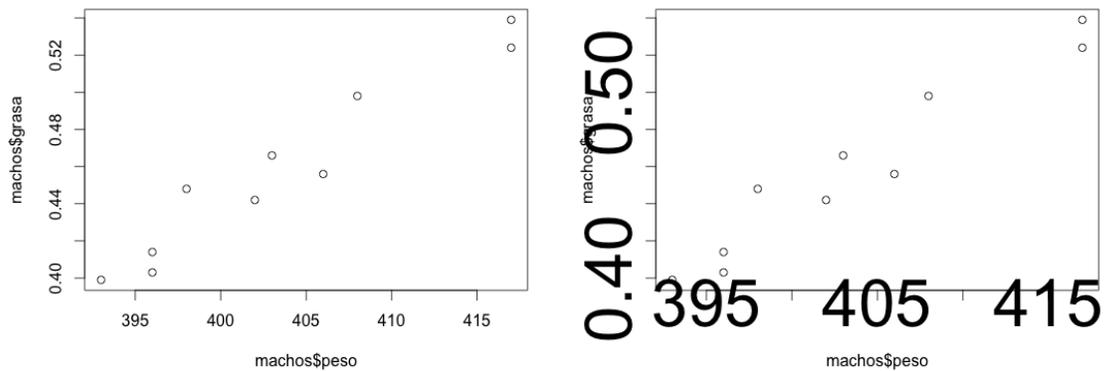


Mediante el argumento *cex.axis* se puede modificar únicamente el tamaño de los valores de los ejes. Veamos dos ejemplos de su implementación en R-project de manera similar a como se realizó con el argumento *cex*:

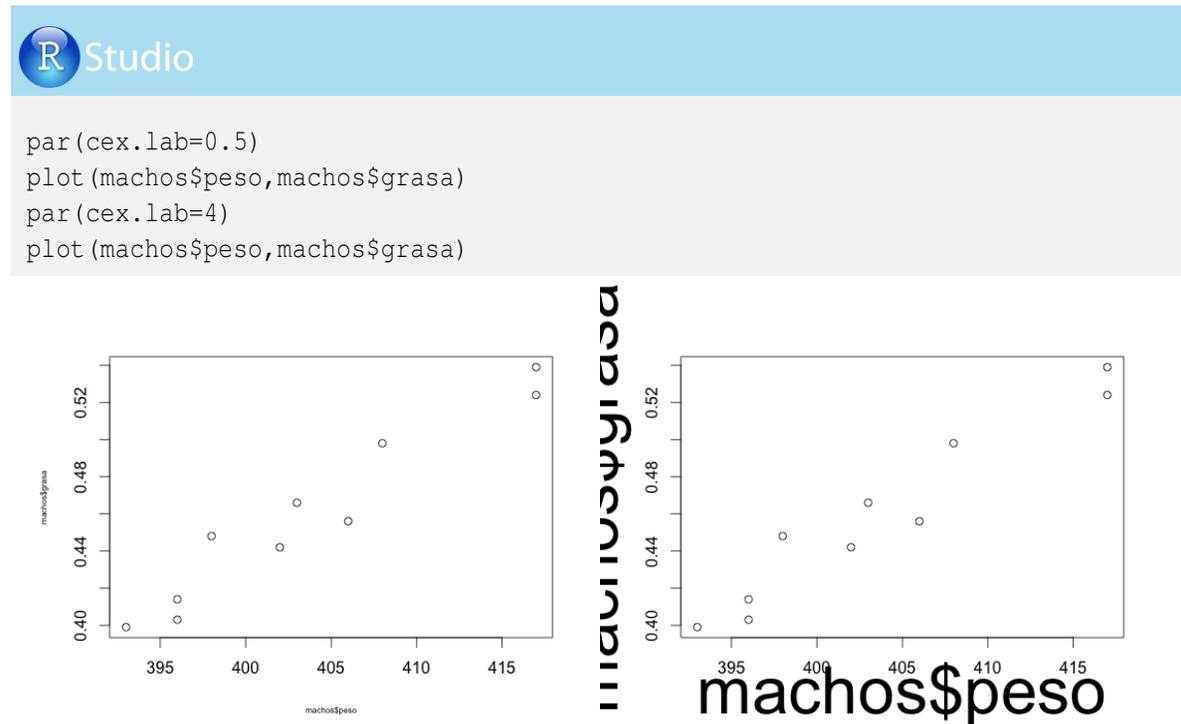
```

R Studio
par(cex.axis=1)
plot(machos$peso,machos$grasa)
par(cex.axis=4)
plot(machos$peso,machos$grasa)

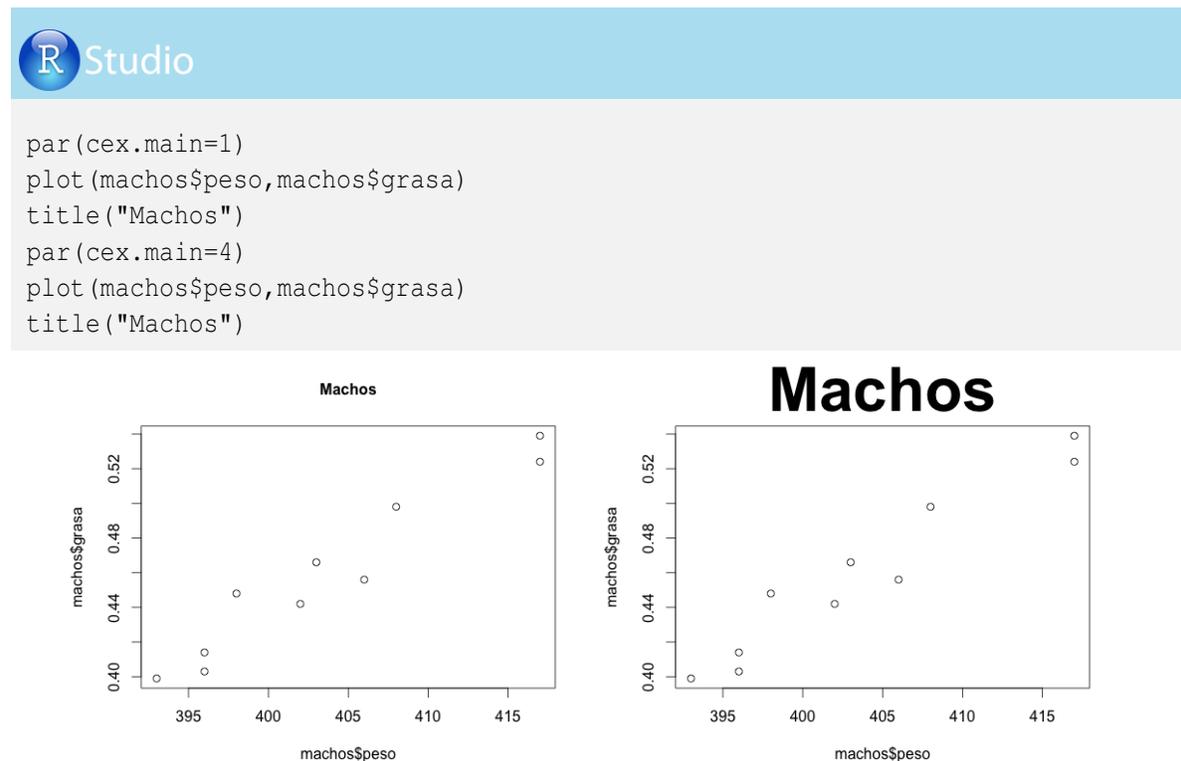
```



Por medio del argumento *cex.lab* se puede modificar el tamaño del texto del nombre de los ejes solamente; veamos dos ejemplos:



El argumento *main* = “ ” o el argumento *title* = “ ” incorporan un título al gráfico, y mediante el argumento *cex.main* podemos modificar el tamaño del texto del título; veamos:

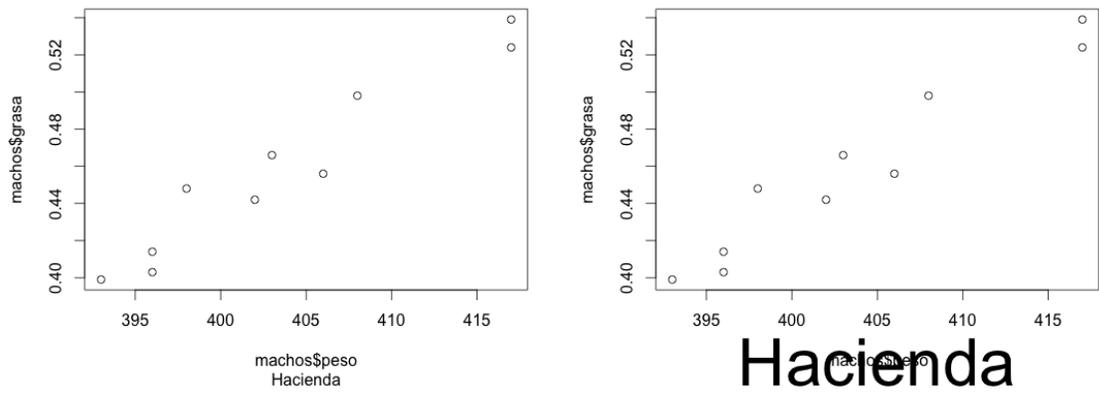


Empleando el argumento `sub = ""` podemos incorporar un subtítulo en la parte inferior del gráfico, y mediante el argumento `cex.sub` podemos determinar el tamaño del subtítulo. Veamos en R-project el uso de estos dos argumentos por medio de dos ejemplos extremos (tamaño muy pequeño y tamaño muy grande):

```

R Studio

par(cex.sub=1)
plot(machos$peso,machos$grasa, sub="Hacienda")
par(cex.sub=4)
plot(machos$peso,machos$grasa, sub="Hacienda")
    
```



### 5.3.5. Colores

Para incorporar colores en R-project debemos conocer algunos de los argumentos, dependiendo de los aspectos que se deseen colorear. Antes de iniciar este proceso, recomendamos al lector ejecutar el comando `colours()` para ver la lista de los 657 colores disponibles; veamos:

```

R Studio

colours()

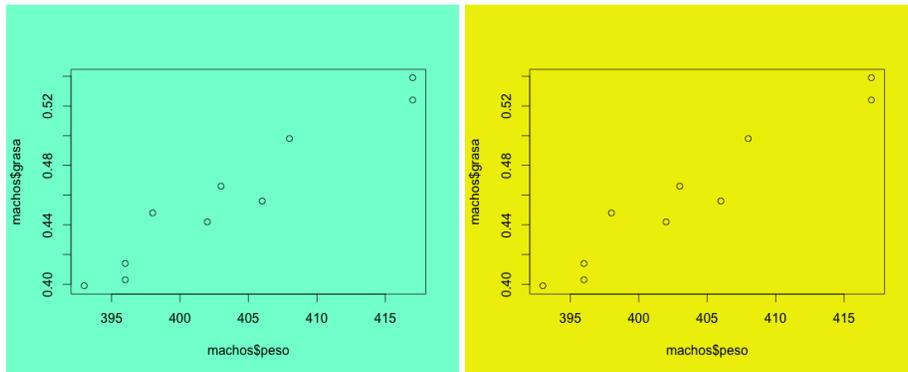
[1] "white"           "aliceblue"       "antiquewhite"    "antiquewhite1"
[5] "antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
[9] "aquamarine1"   "aquamarine2"    "aquamarine3"    "aquamarine4"
.
[653] "yellow1"        "yellow2"         "yellow3"         "yellow4"
[657] "yellowgreen"
    
```

El argumento *bg* seguido del color de interés para el usuario, el cual debe estar entre comillas, permite cambiar el color del fondo general del gráfico. Veamos dos ejemplos:

```

R Studio
par(bg="aquamarine")
plot(machos$peso,machos$grasa)
par(bg="yellow2")
plot(machos$peso,machos$grasa)

```

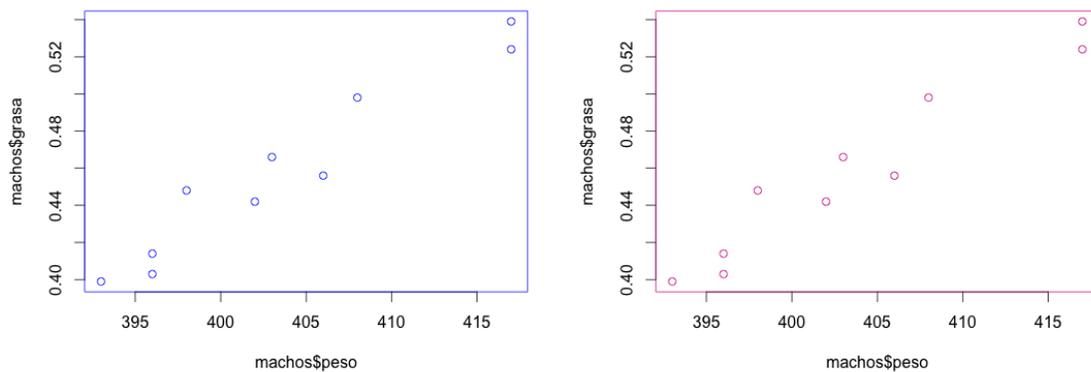


De manera similar, con el argumento *col* podemos fijar el color interno del área de trazado; veamos dos casos:

```

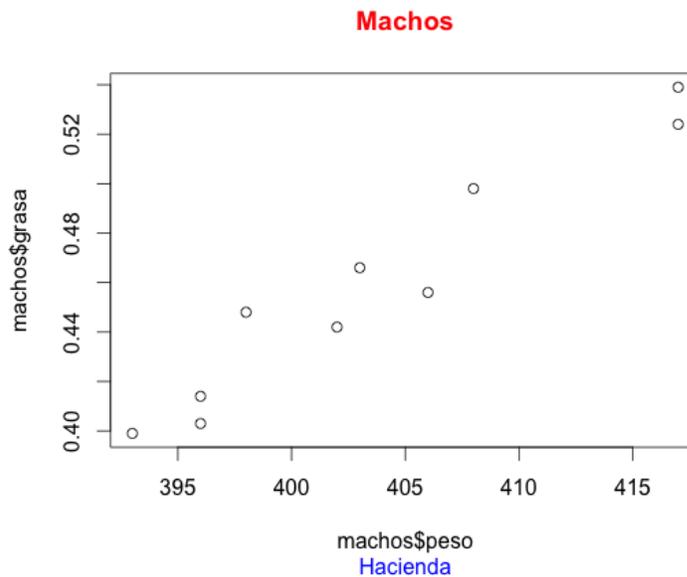
R Studio
par(col="blue")
plot(machos$peso,machos$grasa)
par(col="violetred")
plot(machos$peso,machos$grasa)

```



Mediante el argumento *col.main* y *col.sub* podemos fijar el color del título y del subtítulo del gráfico; veamos:

```
R Studio  
par(col.main="red")  
par(col.sub="blue")  
plot(machos$peso,machos$grasa, sub="Hacienda")  
title("Machos")
```



Dentro del comando *par* podemos escribir los dos argumentos anteriores, como se indica a continuación:

```
R Studio  
par(col.main="red", col.sub="blue")
```

Finalmente, mediante los argumentos *col.axis* y *col.lab* se pueden modificar los colores de las líneas y de los nombres de los ejes, respectivamente. Dichos comandos se utilizan de manera semejante a lo visto anteriormente.

### 5.3.6. Tipos de letra

El argumento *family* determina el tipo de letra del texto del gráfico. Este argumento seguido de = “*serif*”, = “*sans*” y = “*mono*” define los tipos de letra más utilizados. Simultáneamente se pueden emplear los argumentos *font.lab* = y *font.axis* = para determinar la fuente del nombre y de los valores de los ejes, respectivamente, con una de las siguientes cinco opciones:

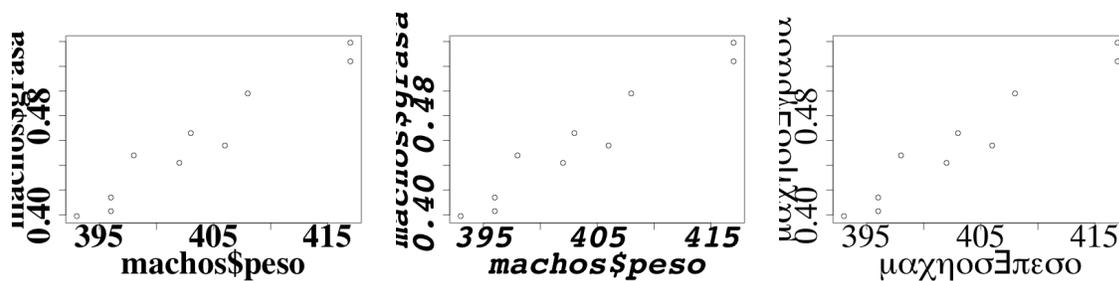
- ♣ 1 corresponde a letra normal,
- ♣ 2 para letra normal y negrita,
- ♣ 3 para letra itálica,
- ♣ 4 para letra negrita e itálica, y
- ♣ 5 para texto simbólico.

A continuación se presentan tres ejemplos: el primero con tipo de letra *serif* y negrita, el segundo con tipo de letra *mono* y cursiva, y el tercero con letra *mono* y tipo símbolo. Para mejor visualización de los nombres y valores de los ejes, aumentaremos su tamaño con los argumentos `cex.lab = 3` y `cex.axis = 3`:

```

R Studio

par (cex.axis=3)
par (cex.lab=3)
par (family="serif")
par (font.lab=2)
par (font.axis=2)
plot (machos$peso,machos$grasa)
par (family="mono")
par (font.lab=4)
par (font.axis=4)
plot (machos$peso,machos$grasa)
par (family="mono")
par (font.lab=5)
par (font.axis=5)
plot (machos$peso,machos$grasa)
    
```



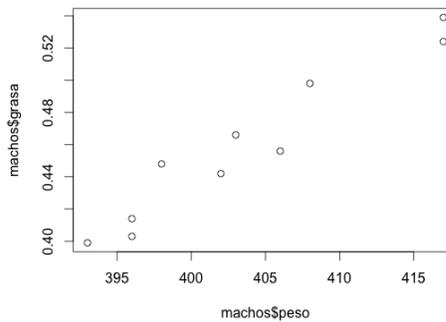
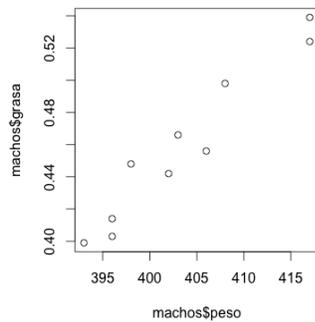
### 5.3.7. Forma rectangular y cuadrada del área de trazado

Empleando el argumento *pty*, seguido de la opción = “s” o de la opción = “m”, se obtiene un gráfico de forma cuadrada o rectangular, respectivamente. Veamos estos dos casos en R-project:

```

R Studio

par(pty="s")
plot(machos$peso,machos$grasa)
par(pty="m")
plot(machos$peso,machos$grasa)
    
```



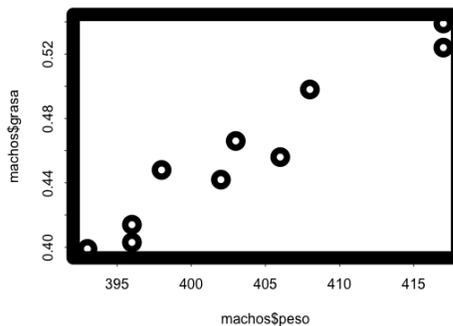
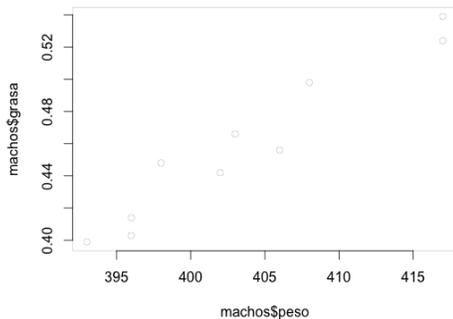
### 5.3.8. Grosor de los ejes y puntos de dato

El argumento *lwd*, seguido de un número mayor que 0, fija el grosor de las líneas de los ejes y de los puntos de datos; veamos su implementación.

```

R Studio

par(lwd=0.2)
plot(machos$peso,machos$grasa)
par(lwd=20)
plot(machos$peso,machos$grasa)
    
```

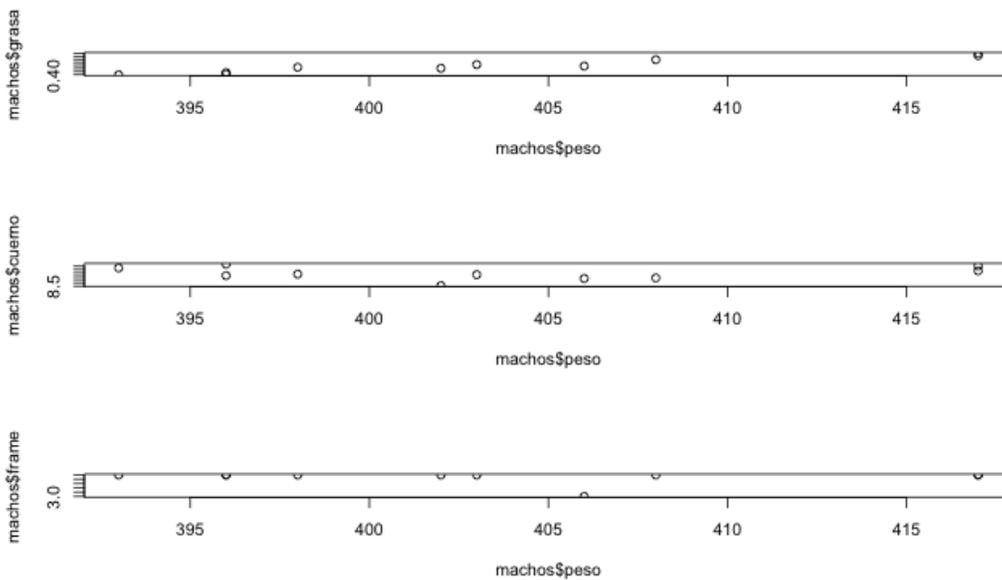


### 5.3.9. Varios gráficos en la misma área total

Se pueden generar varios gráficos en una misma área mediante el argumento *mfrow* del comando *par*. Se debe construir un vector que contiene el número de figuras distribuidas en filas y columnas. La siguiente programación permite agrupar 3 gráficos distribuidos en 3 filas y 1 columna.



```
par(mfrow=c(3,1))
plot(machos$peso,machos$grasa)
plot(machos$peso,machos$cuerno)
plot(machos$peso,machos$frame)
```



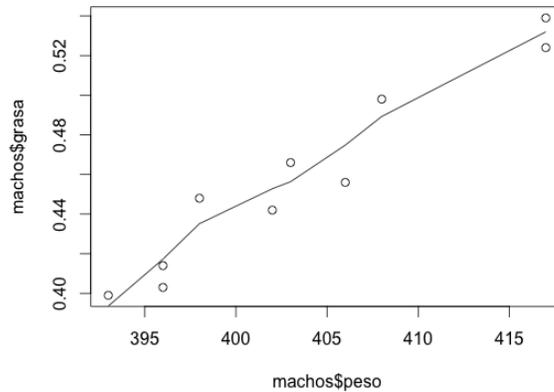
Con los argumentos  $CEX = 0$  ó  $MEX = 0$  se restablece el área a un solo gráfico por plano.

### 5.4. Inclusión de líneas

En muchas ocasiones resulta necesario incluirle líneas al gráfico de interés. R-project tiene los comandos *line* y *abline* para tal fin. Dado que el comando *line* es usualmente empleado en el diseño de gráficos de este capítulo, centraremos nuestra atención en la generación de una línea suavizada, obtenida por una regresión local, mediante la ponderación lineal de regresión de mínimos cuadrados (*lowess*, locally weighted scatter plot smoothing); veamos la programación:



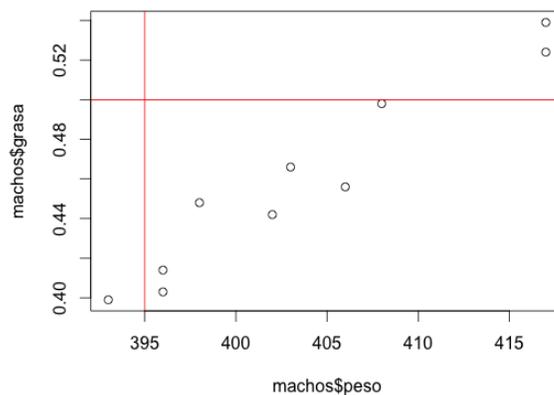
```
plot(machos$peso,machos$grasa)
lines(lowess(machos$peso,machos$grasa))
```



Ahora, mediante el comando *abline* se pueden incluir líneas fijas de orientación horizontal y vertical mediante los argumentos  $h =$  y  $v =$ , respectivamente, seguidos de valores determinados por el usuario. A continuación trazaremos dos líneas, una horizontal relacionada con el espesor de grasa (0.5 cm) y una vertical relacionada con el peso (390 kg), valores que la empresa considera adecuados para el sacrificio de los animales; veamos:



```
plot(machos$peso,machos$grasa)
abline(h=0.50, v=395, col="red")
```



Finalmente, por medio del argumento *lty* se generan los siguientes tipos de trazado para las líneas incluidas en el gráfico:

- ♣ = 0 o = “*blank*” (blanco)
- ♣ = 1 o = “*solid*” (continuo)
- ♣ = 2 o = “*dashed*” (discontinuo)
- ♣ = 3 o = “*dotted*” (puntos)
- ♣ = 4 o = “*dotdash*” (punto y guión)
- ♣ = 5 o = “*longdash*” (guión largo)
- ♣ = 6 o = “*twodash*” (dos guiones).

Veamos su implementación en R-project mediante ejemplos aplicados al caso del peso y el espesor de grasa adecuados para el sacrificio de un animal por parte del frigorífico:

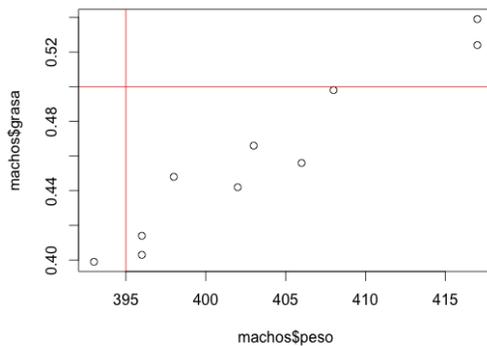
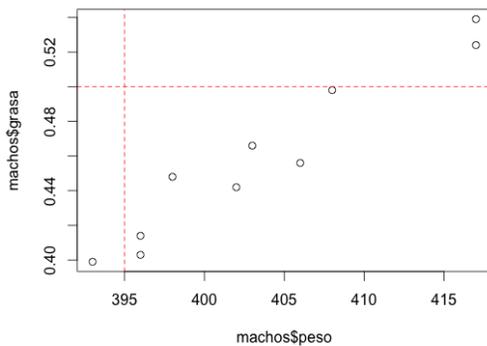
```

R Studio

plot(machos$peso,machos$grasa)
par(lty="dashed")
abline(h=0.50, v=390, col="red")

plot(machos$peso,machos$grasa)
par(lty="solid")
abline(h=0.50, v=390, col="red")

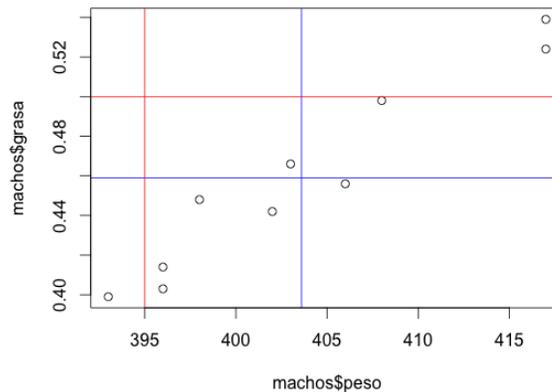
```



En el siguiente gráfico se incorporan cuatro líneas de trazado: las dos azules indican los promedios del espesor de grasa y del peso al sacrificio, y las dos líneas rojas indican los pesos ideales para el frigorífico. Para generar las líneas de los promedios, creamos dos vectores con el comando *mean* e incorporamos estos resultados al argumento *abline*, y para el color incluimos el argumento *col*; veamos su implementación en R-project:



```
pesomedio=mean(machos$peso)
grasamedia=mean(machos$grasa)
plot(machos$peso,machos$grasa)
abline(h=(grasamedia),v=(pesomedio),col="blue")
abline(h=0.50, v=395, col="red")
```



### 5.5. Símbolos de puntos de dato

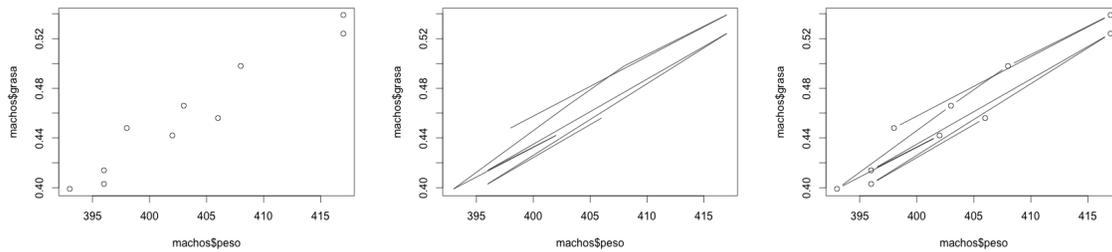
Los símbolos y líneas que permiten visualizar los puntos de datos pueden ser modificados por el argumento *type* en los comandos que generan gráficos. Los tipos más utilizados son:

- ♣ “p” para puntos,
- ♣ “l” para líneas,
- ♣ “b” para puntos y líneas,
- ♣ “c” genera líneas con espacios en blanco donde irían los puntos,
- ♣ “o” para puntos y líneas sobrepuestas,
- ♣ “h” para líneas verticales, y
- ♣ “n” para no mostrar los puntos de dato en el gráfico:

Veamos la implementación del argumento *type* dentro del comando *plot* en R-project, generando tres gráficos con puntos de datos representados por puntos, líneas y puntos y líneas:



```
plot(machos$peso, machos$grasa, type="p")
plot(machos$peso, machos$grasa, type="l")
plot(machos$peso, machos$grasa, type="b")
```

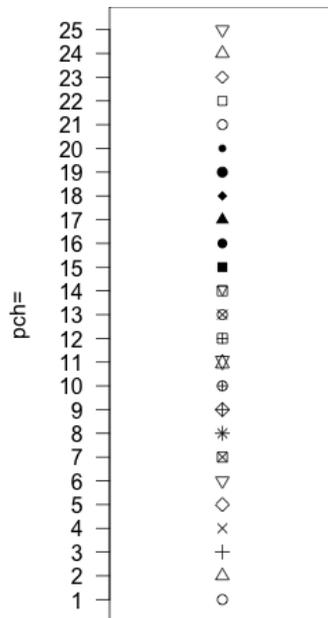


R-project tiene una lista de símbolos para identificar los puntos de datos (estrellas, triángulos, círculos, etc.). Mediante la siguiente programación generaremos un gráfico con los códigos de los símbolos y su visualización disponibles:

```

R Studio

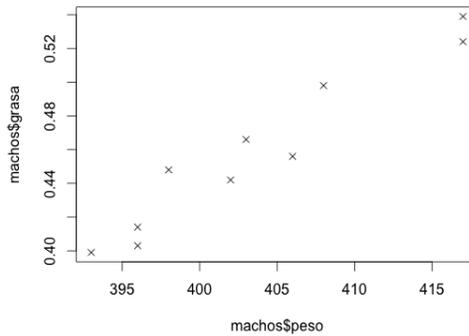
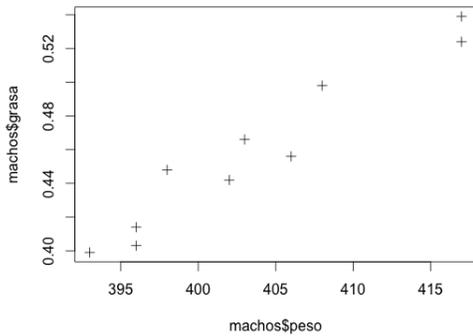
x=(1)
y=(1:25)
g=merge(x,y)
par(lab=c(1,25,25))
par(plt=c(0.3,0.6,0,1))
par(las=1)
plot(g$x,g$y, pch=1:25, ylab="pch=",xlab="")
    
```



Para modificar los símbolos de los puntos de datos se utiliza el comando *pch*, seguido del número que genera uno de los símbolos presentados en el gráfico anterior. En el siguiente ejemplo utilizaremos el símbolo +, al cual le corresponde el número 3 y otro gráfico con el símbolo x al que le corresponde el número 4:



```
plot(machos$peso, machos$grasa, pch=3)
plot(machos$peso, machos$grasa, pch=4)
```



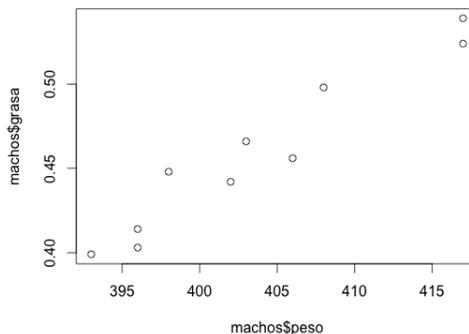
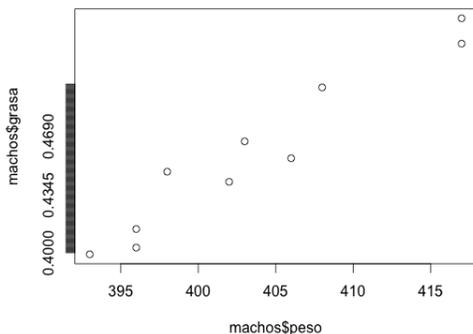
### 5.6. Divisiones de los ejes

La escala de los ejes obedece a un valor mínimo, un valor máximo y al número de divisiones. Para modificar la dimensión de los ejes  $y$  y  $x$  se utilizan los argumentos  $yaxp$  y  $xaxp$ , respectivamente, seguido de las siguientes especificaciones  $= c(\text{valor mínimo}, \text{valor máximo}, \text{número de divisiones})$ .

En el siguiente ejemplo generaremos dos gráficos donde especificamos que el eje  $y$  varíe de 0.4 a 0.5 con 200 divisiones y otro gráfico que mantenga los mismos valores mínimo y máximo, pero que genere dos divisiones en la escala.



```
plot(machos$peso, machos$grasa, yaxp=c(0.4, 0.5, 200))
plot(machos$peso, machos$grasa, yaxp=c(0.4, 0.5, 2))
```

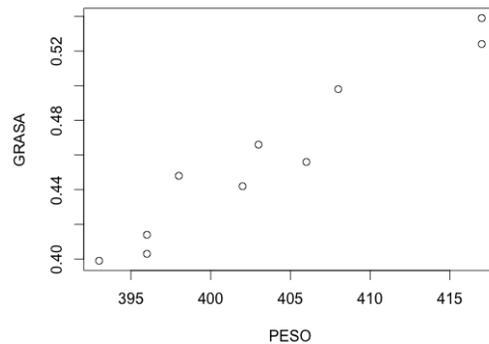


## 5.7. Nombres en el gráfico

Para incluir o modificar los nombres de los ejes  $x$  y  $y$  se utilizan los argumentos  $xlab$  y  $ylab$ , respectivamente. En el siguiente ejemplo le modificamos los títulos a los ejes, donde el nombre para  $x$  es *PESO* y para  $y$  es *GRASA*; veamos su implementación:



```
plot(machos$peso,machos$grasa, xlab="PESO", ylab="GRASA")
```

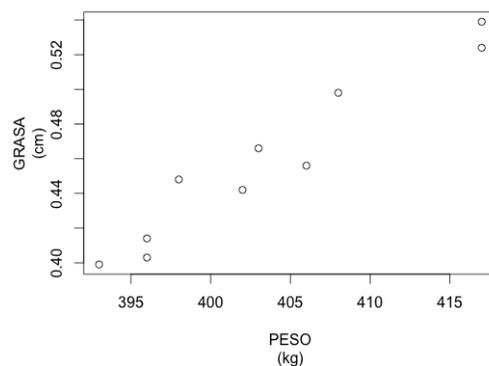


Para incluir líneas adicionales a los nombres de los ejes en posiciones posteriores a la línea donde están los nombres, el usuario debe recordar claramente las distancias desde el entorno del área total y el área de los trazos, como se indicó en el primer gráfico de este capítulo.

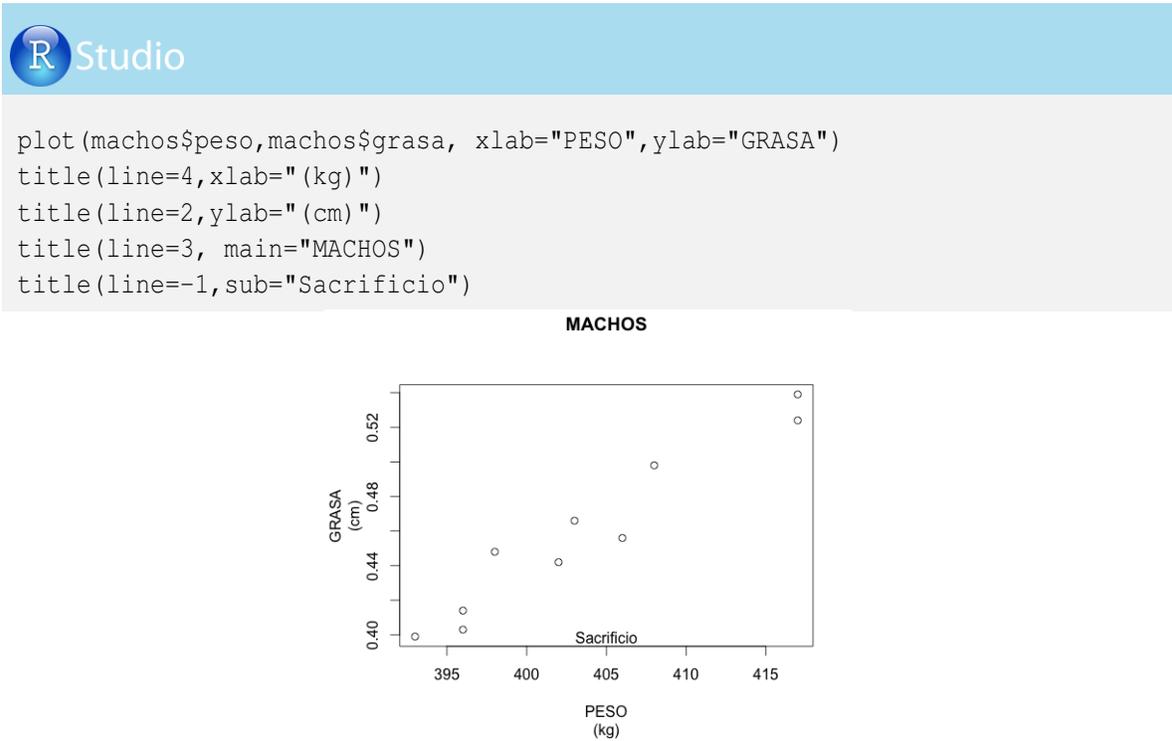
En la siguiente programación se presenta la forma como se puede incluir un línea adicional al título de los ejes con el objetivo de incluir las unidades de medida en los dos ejes:



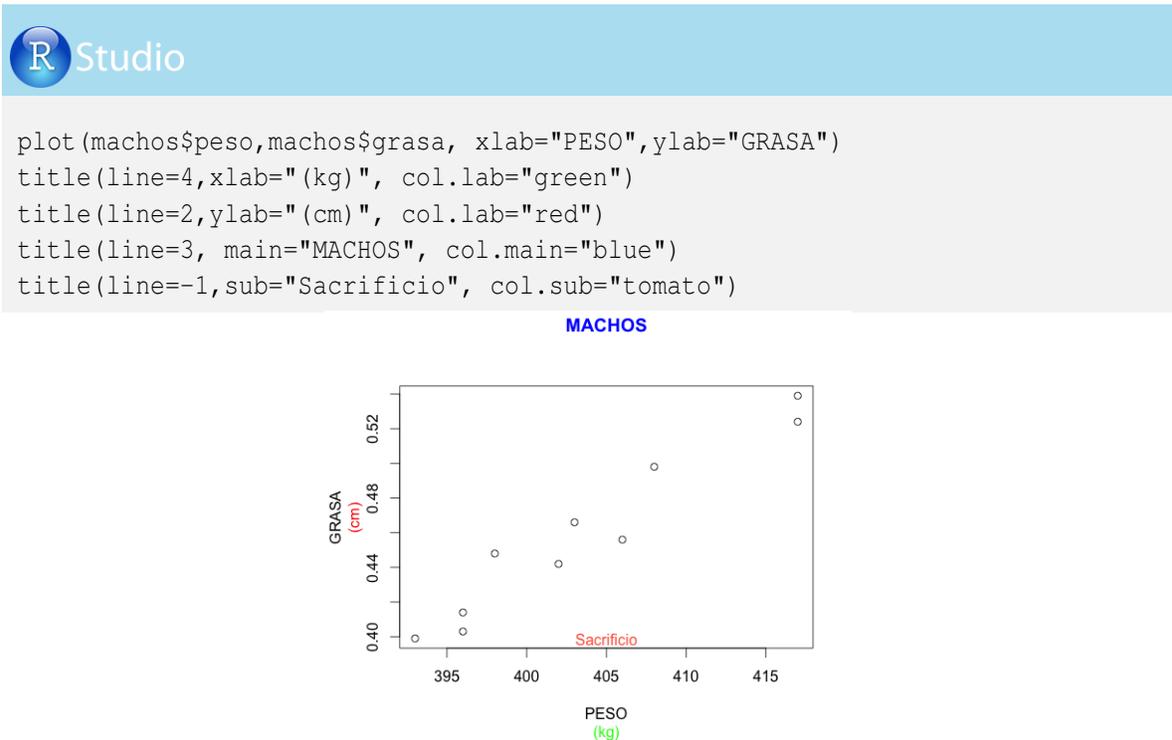
```
plot(machos$peso,machos$grasa, xlab="PESO",ylab="GRASA")
title(line=4,xlab="(kg) ")
title(line=2,ylab="(cm) ")
```



A continuación utilizaremos el comando *title* para agregar un título general (*MACHOS*), mediante el argumento *main* y un subtítulo (*Sacrificio*), con el argumento *sub*. También indicamos la posición de estos nombres con el argumento *line*; veamos la programación:



Finalmente les asignaremos colores a los títulos, mediante el argumento *col.lab* para los nombres de los ejes, el argumento *col.main* para el título general y *col.sub* para el subtítulo:

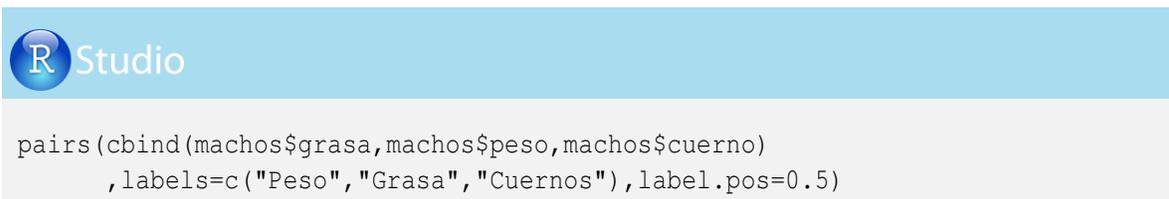


## 5.8. Gráficos de relación de variables *pairs*

El comando *pairs* permite generar gráficos de relación de variables, semejantes al gráfico obtenido con el comando *plot*. En el siguiente ejemplo presentaremos la relación visual entre las variables *peso*, *grasa* y *cuerno*, definidas con *cbind* dentro del comando *pairs*. Generaremos un gráfico de nueve figuras, donde las figuras que están fuera de la diagonal presentan los puntos de dato que relacionan dos variables y en las figuras de la diagonal aparecerán los nombres de las variables, escritas en la programación con el argumento *labels*.

Se puede modificar la posición del nombre de las variables dentro de las figuras mediante el argumento *label.pos* seguido por un valor entre 0 y 1. Un valor cercano a 0 posiciona el nombre cerca al eje *x*, y si el valor es próximo a 1 el nombre quedará en el borde superior de la figura.

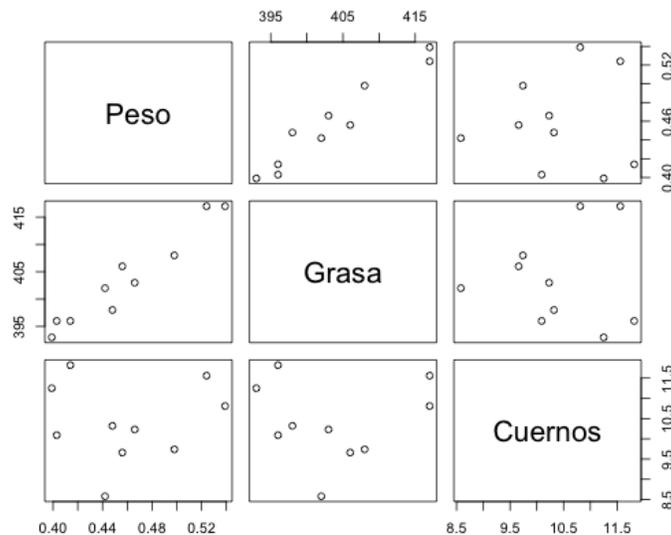
Veamos la programación del comando *pairs*, indicando las variables y los nombres de las variables posicionados en la mitad de las figuras (*label.pos = 0.5*):



```

pairs(cbind(machos$grasa,machos$peso,machos$cuerno)
      ,labels=c("Peso", "Grasa", "Cuernos"),label.pos=0.5)

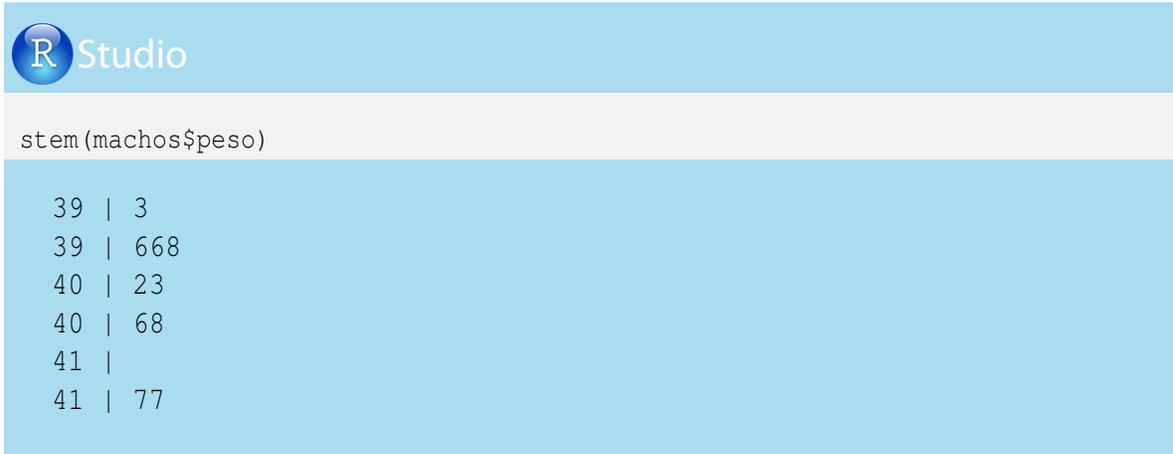
```



## 5.9. Gráficos *stem*

Un diagrama frecuentemente utilizado en estudios exploratorios y descriptivos es el gráfico de tallos y hojas (*stem and leaf diagram*), el cual permite observar la distribución empírica de los datos de una variable. Se denomina *hoja* al conjunto de dígitos de la derecha y tallo al

conjunto de cifras de la izquierda. En R-project podemos utilizar el comando *stem* seguido por la variable de interés; veamos un ejemplo:



```

R Studio

stem(machos$peso)

39 | 3
39 | 668
40 | 23
40 | 68
41 |
41 | 77
    
```

Para entender la distribución de los datos, recordemos que la variable *peso* varía entre 393 y 417; entonces el comando *stem* generó un gráfico a la izquierda, de valores 39, 40 y 41, correspondiente a las decenas y centenas de esta variable. Los valores a la derecha corresponden a las unidades, distribuidos de la siguiente forma:

- ♣ el valor mínimo es 393, generando 39|3,
- ♣ dos valores de 396 y uno de 398, generando 39|668,
- ♣ un valor de 402 y uno de 403, generando 40|23,
- ♣ un valor de 406 y uno 408, generando 40|68,
- ♣ no hay valores entre 410 y 414, generando 41| , y
- ♣ dos valores de 417, generando 41|77

### 5.10. Histogramas de frecuencia generados con el comando *histogram*

El gráfico de histograma permite observar la distribución específica de la frecuencia de los valores muestrales de una variable aleatoria, organizados en clases o intervalos. Es muy útil para hacerse una idea general del tipo de distribución de los valores muy lejanos y de la concentración de datos.

El comando *histogram* de la librería *lattice* de Deepayan (2008) permite obtener histogramas de frecuencia de un conjunto de datos. Este comando tiene un conjunto de argumentos que determinan la presentación visual de los datos, veamos algunos de ellos:

- ♣ El argumento *break* = determina el número aproximado de intervalos de clase en los que se realiza la distribución de los datos. Para determinar el número de clases, generalmente se emplean las reglas de Sturges, Scott y Freeman-Diaconis. También el usuario puede definir un número entero de clases, crear su propia función para determinarla o utilizar un vector con los intervalos que se desean.
- ♣ El argumento *type* = seguido de la palabra “percent” genera el histograma de frecuencias absolutas. Este argumento con la palabra “count” muestra el histograma relacionado con

el número datos distribuidos en los intervalos y con “*density*” se visualiza la densidad de probabilidad de los datos.

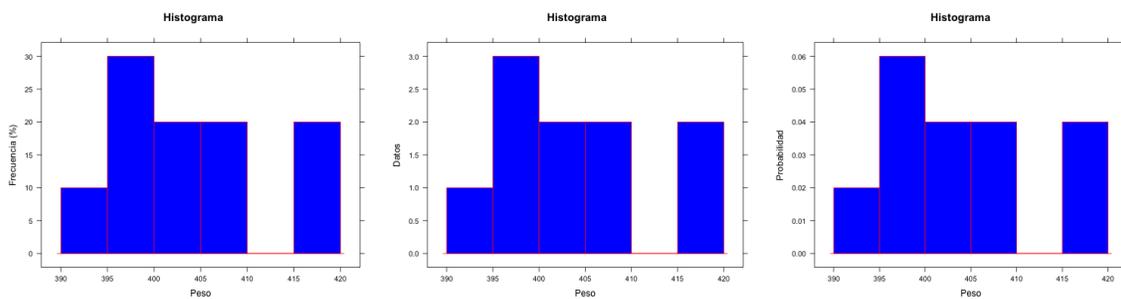
En el siguiente ejemplo utilizaremos la variable *peso* de la hoja de datos *machos*, donde generaremos los tres tipos de histogramas indicados en el argumento *type*, y utilizaremos la regla de Sturges para determinar el número de clases. Además le agregamos los argumentos para incluir los nombres y el color de las barras y el borde; veamos:

```

R Studio

histogram(machos$peso, type="percent", breaks="Sturges", col="blue",
  border="red", main="Histograma", xlab="Peso", ylab="Frecuencia (%)")
histogram(machos$peso, type="count", breaks="Sturges", col="blue",
  border="red", main="Histograma", xlab="Peso", ylab="Datos")
histogram(machos$peso, type="density", breaks="Sturges", col="blue",
  border="red", main="Histograma", xlab="Peso", ylab="Probabilidad")

```



### 5.11. Gráficos qqplot

Una de las medidas de posición no central más importante es el cuantil (denominado como Q). Los cuantiles dividen la distribución de probabilidad empírica de los datos en partes iguales, es decir, en intervalos de valores equiparables. Generalmente se utiliza el *cuartil* (cuatro grupos), el *quintil* (cinco grupos) y el *percentil* (100 grupos).

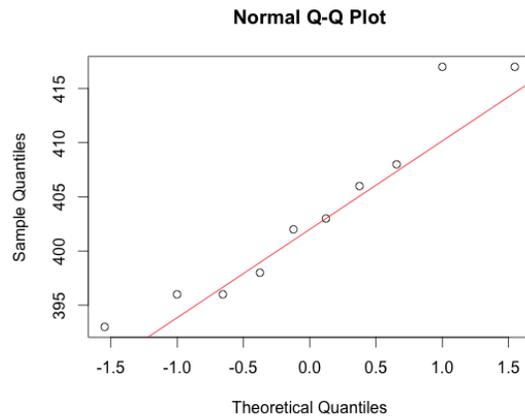
En el caso del *cuartil*, la distribución de los datos se divide en cuatro partes (cuantiles que contienen el 25% de los datos), originando los cuantiles 0.25, 0.50 y 0.75. Es decir, para la variable *X* se tendrían:

- ♣  $X_{0.25}$  = El 25% de los datos es menor o igual que este valor,
- ♣  $X_{0.50}$  = El 50% de los datos es menor o igual que este valor, y
- ♣  $X_{0.75}$  = El 75% de los datos es menor o igual que este valor.

Un gráfico *QQ* indica las diferencias de cada punto de datos de la relación del valor observado con el valor esperado sobre la distribución. Si los datos siguen una distribución, se espera que se generen puntos en el gráfico cercanos a una línea recta. Veamos a continuación un ejemplo utilizando el comando *qqnorm* para generar un gráfico *QQ* específico para una distribución normal y le adicionamos una línea recta de color rojo con el comando *qqline*.



```
qqnorm(machos$peso)
qqline(machos$peso, col="red")
```



### 5.12. Gráficos *boxplot*, *barplot* y *pie*

Un gráfico *boxplot* o gráfico de caja es una herramienta de la estadística descriptiva que proporciona una visión general de la simetría de la distribución de los datos mediante un resumen numérico de las cantidades, de los valores mínimo y máximo y de los cuartiles:  $Q_{0.25}$ ,  $Q_{0.50}$  o mediana y el  $Q_{0.75}$ . También suministra información sobre la simetría de los datos y la presencia de datos atípicos. Este gráfico es útil para comparar las distribuciones de datos que provienen de poblaciones distintas, o datos de una misma población bajo condiciones experimentales diferentes.

Para mostrar la relevancia de este tipo de gráfico, vamos a incluirle a la hoja de datos de *machos* información productiva de hembras con el fin de comparar los dos sexos. Los datos de las hembras son:

ANIMAL	PESO(kg)	GRASA(cm)	CUERNO(cm)	FRAME(clases)
11	350	0.498	8.66	4
12	333	0.465	7.10	3
13	412	0.526	8.11	3
14	401	0.444	7.11	3
15	434	0.424	7.88	3
16	323	0.333	7.11	4
17	360	0.413	8.10	3
18	308	0.568	8.98	3
19	316	0.533	5.10	3
20	398	0.424	3.10	3

Generemos la hoja de datos en R-project con información de las hembras:

```
R Studio

peso=c(350,333,412,401,434,323,360,308,316,398)
grasa=c(0.498,0.465,0.526,0.444,0.424,0.333,0.413,0.568,0.533,0.424)
cuerno=c(8.66,7.10,8.11,7.11,7.88,7.11,8.10,8.98,5.10,3.10)
frame=c(4,3,3,3,3,4,3,3,3,3)
hembras=data.frame(peso, grasa, cuerno, frame)
hembras

  peso grasa cuerno frame
1  350 0.498   8.66     4
2  333 0.465   7.10     3
.
10 398 0.424   3.10     3
```

Ahora juntamos la hoja de datos *hembras* con la hoja de datos *machos*, pero antes creamos la variable *sexo* en las dos hojas de datos:

```
R Studio

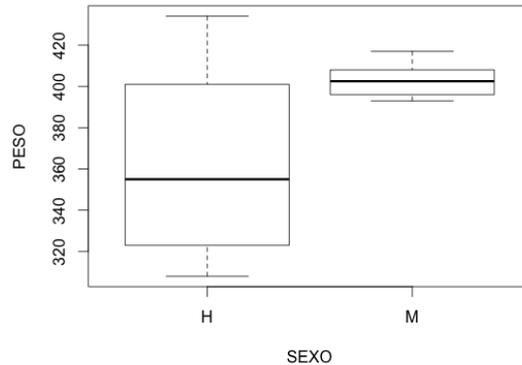
hembras$sexo="H"
machos$sexo="M"
todos=rbind(machos,hembras)
todos=data.frame(todos)
todos

  peso grasa cuerno frame sexo
1  406 0.456   9.66     3    M
2  396 0.403  10.09     4    M
.
10 398 0.448  10.32     4    M
11 350 0.498   8.66     4    H
.
20 398 0.424   3.10     3    H
```

Con esta hoja de datos de 20 animales, generaremos un gráfico *boxplot* (gráfico de Box-Whisker) para la variable *peso* de los animales agrupados por la variable *sexo*. El gráfico resultante muestra las medidas descriptivas: mediana, primer cuartil, segundo cuartil o mediana, tercer cuartil y valores extremos. En el primer argumento de este comando se utiliza el símbolo  $\sim$  para relacionar la variable *peso* con la variable *sexo*:

```
R Studio

boxplot(todos$peso~todos$sexo,ylab="PESO",xlab="SEXO",data=todos)
```

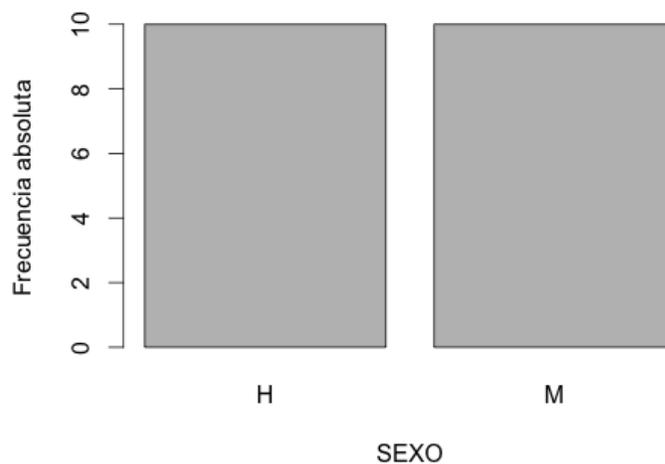


El gráfico nos indica que existió mayor variación de peso en las hembras, cuya mediana estuvo cercana a los 360 kg, diferente a los machos que fue cercana a los 400 kg.

Otra herramienta de la estadística descriptiva es el diagrama de barras, el cual se utiliza para representar las clases de una variable categórica o valores cuantitativos de una variable discreta.

En el eje horizontal de este tipo de diagrama se presentan usualmente las categorías o valores de la variable discreta, y en el eje y se presentan las frecuencias absolutas o relativas de cada grupo formado. Veamos en R-project un ejemplo para obtener el diagrama de barras del número de animales de cada sexo:

```
R Studio  
barplot(table(todos$sexo), xlab="SEXO", ylab="Frecuencia absoluta")
```



El diagrama circular o gráfico pastel tiene la misma funcionalidad del diagrama de barras, pero en forma circular. Mediante el comando *pie* se puede generar este tipo de gráfico para clases de una variable categórica o valores de una variable discreta. Veamos la implementación en R-project:



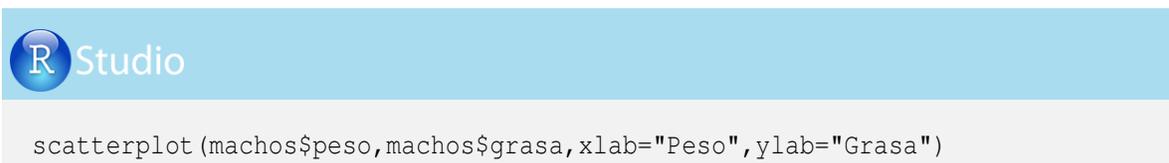
### 5.13. Gráficos *scatterplot* y *coplot*

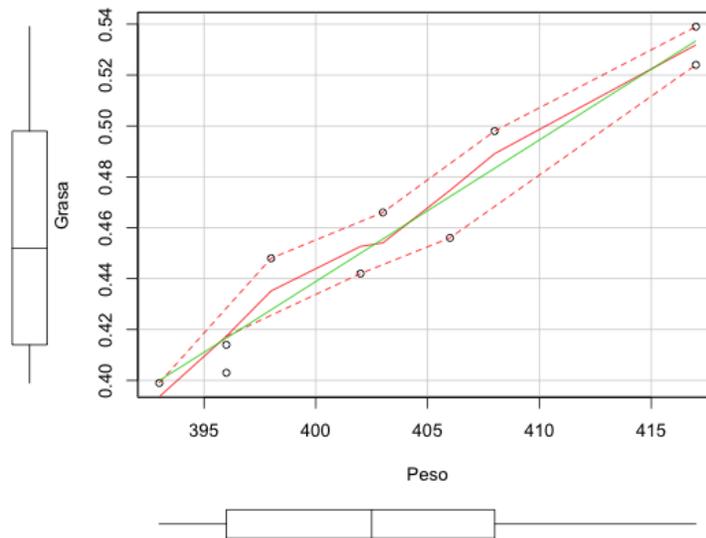
El gráfico *scatterplot* permite observar la relación de dos variables, la recta de regresión lineal entre las dos variables, una línea suavizada obtenida con una regresión no paramétrica de Loess y unas líneas suavizadas (inferior y superior) que representan el cuadrado medio de los residuos de la regresión de Loess. Adicionalmente genera un diagrama de caja para cada variable fuera del área de trazado, que permite observar la dispersión de los datos.

Antes de realizar los gráficos *scatterplot* debemos instalar las librerías *car* de Fox y Weisberg (2012) y *nnet* de Venables y Ripley (2002).



Ahora generemos un gráfico *scatterplot* con la hoja de datos *machos* para la relación entre el peso y la grasa, mediante la siguiente programación en R-project:





En el anterior gráfico se pueden apreciar las distribuciones de los datos de grasa y peso en forma de *boxplot*, la cual está cercana a los nombres de las variables y fuera del área de trazado (incluya *boxplot = FALSE*, si no desea las cajas).

La línea verde indica la recta de regresión lineal para la relación entre el peso y la grasa de los machos (use *reg.line = FALSE* si no la desea).

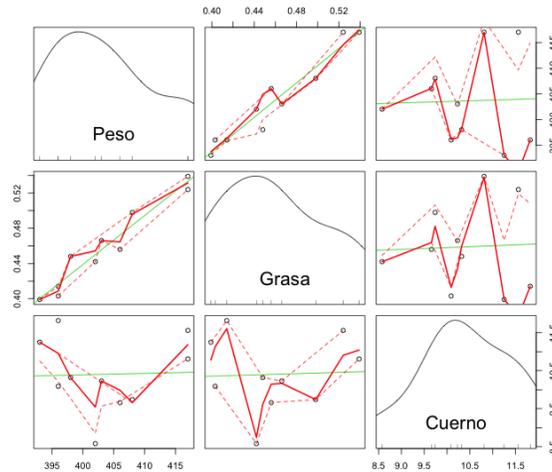
La línea roja continua indica la regresión no paramétrica Loess entre el peso y la grasa (escriba el argumento *smoth = FALSE* si no desea ver esta línea).

Las líneas rojas punteadas indican los cuadrados medios de los residuos a lo largo de la línea de regresión de Loess (use *spread = FALSE* para no mostrar estas líneas).

El comando *scatterplotMatrix* genera gráficos de dispersión de diversas variables, incluyendo en la diagonal la densidad de probabilidad empírica de cada una de ellas; veamos:



```
scatterplotMatrix(cbind(machos$peso,machos$grasa,machos$cuerno)
,diag="density",var.labels=c("Peso","Grasa","Cuerno"),label.pos=0.2)
```



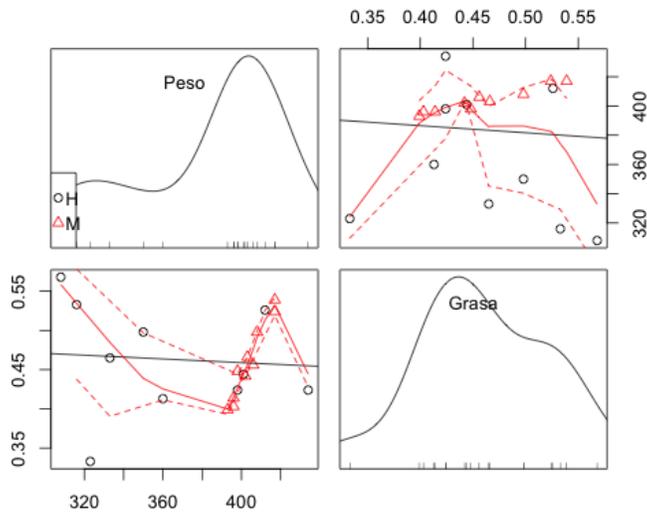
También se pueden generar gráficos de dispersión con las variables agrupadas por otra variable, mediante el comando *scatterplotMatrix* y *scaterplot*. En este caso utilizaremos la hoja de datos *todos* y analizaremos la relación del peso y la grasa dentro de cada sexo. A continuación se presenta la programación con el comando *scatterplotMatrix*. El primer argumento contiene el símbolo  $\sim$ , el nombre de las variables a comparar separadas por el signo  $+$ , el símbolo  $|$  para indicar que los datos están agrupados por una variable y el nombre de variables que se agrupan. Incluiremos otros argumentos como *data* para indicar el nombre de la hoja de datos y el argumento *var.labels* para generar los nombres de las variables en el gráfico; veamos:

```

R Studio

scatterplotMatrix(~todos$peso+todos$grasa|todos$sexo,data=todos,
var.labels=c("Peso", "Grasa"),cex.labels=1)

```

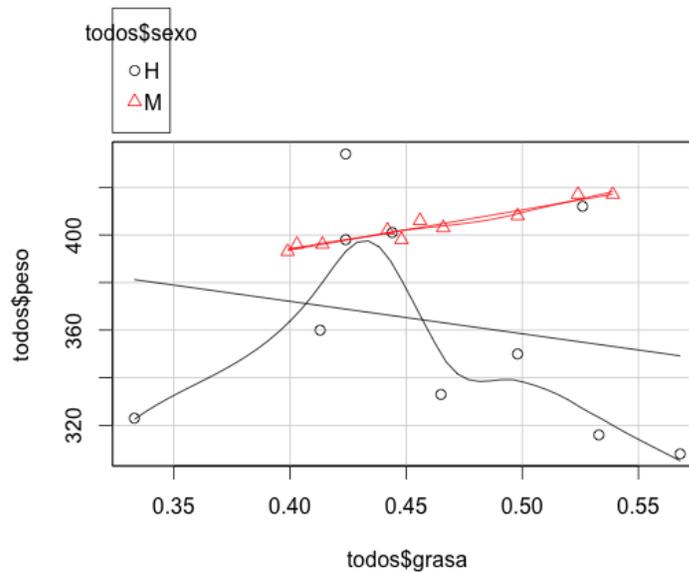


En las figuras de la diagonal está la distribución de los datos para cada variable, sin importar el sexo, y las figuras por fuera de la diagonal muestran los puntos de datos de la relación entre las dos variables (círculo para hembras y triángulo para machos) y las líneas explicadas en los anteriores gráficos.

Ahora veamos un gráfico muy semejante al anterior, obtenido con el comando *scatterplot*:

```

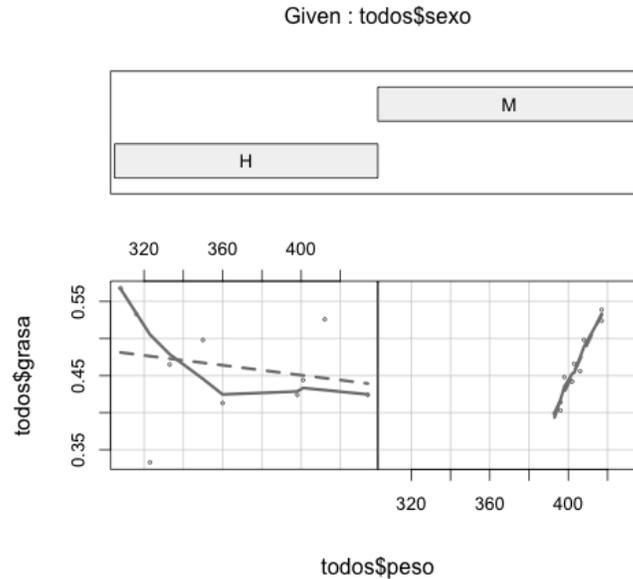
R Studio
scatterplot (todos$peso~todos$grasa|todos$sexo)
    
```



Otro gráfico que hace énfasis en las cajas de dispersión de los datos por grupo es el *coplot*. A continuación presentaremos la programación para mostrar la relación de las variables *peso* y *grasa*, condicionadas a la variable *sexo*:

```

R Studio
coplot ((todos$grasa)~todos$peso|todos$sexo, panel=panel.car,
col=gray(.5), lwd=3, cex=.4)
    
```



Este gráfico muestra en la parte superior una caja, donde se visualiza a grosso modo que los machos son más pesados que las hembras, y dos figuras en la parte inferior con los puntos de datos de la relación de la grasa con el peso para cada sexo.

### 5.14. Gráficos de Cleveland

En ocasiones es necesario generar gráficos de múltiples niveles de tipo Cleveland. Para explicar este tipo de gráficos crearemos una hoja de datos que contiene información del número de animales de dos fincas, repartidas en cuatro categorías de animales (novillas, vacas de primer parto, vacas de segundo parto y vacas adultas). Veamos la hoja de datos:

	LA HERRADURA	LA MONTAÑA
Novillas preñadas	12	11
Vacas 1er parto	6	13
Vacas 2 partos	0	11
Vacas adultas	0	13

Generemos la hoja de datos Fincas en R-project mediante el comando `data.frame`.

```

R Studio

Fincal=c(12,6,0,0)
Finca2=c(11,13,11,13)
Fincas=data.frame(Fincal,Finca2)
    
```

Incluyamos los nombres de las filas y de las columnas, mediante los comandos *rownames* y *colnames*, respectivamente.

```

R Studio

colnames(Fincas)=c("La Herradura", "La Montaña")
rownames(Fincas)=
  c("Novillas preñadas", "Vacas 1er parto", "Vacas 2 partos", "Vacas adultas")
attach(Fincas)
Fincas

```

	La Herradura	La Montaña
Novillas preñadas	12	11
Vacas 1er parto	6	13
Vacas 2 partos	0	11
Vacas adultas	0	13

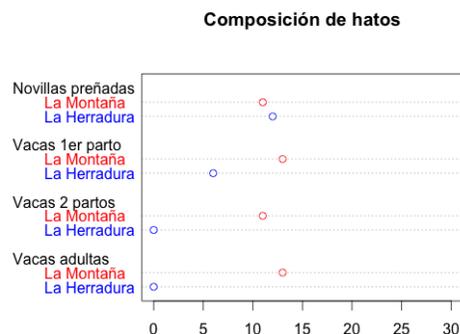
Esta hoja de datos (*Fincas*) tiene cuatro filas que contienen el número de animales por categoría reproductiva y dos columnas relacionadas con las dos fincas. Además se incluyen los nombres de filas y columnas. Esta tabla de categoría reproductiva y fincas será reproducida en un gráfico de tipo Cleveland, mediante el comando *dotchart* con los argumentos: *t* seguido de la hoja de datos, el título del gráfico con el argumento *main*, el color asignado a los datos de la primera finca (azul) y a la segunda finca (rojo) y el argumento *xlim* para indicar los límites que se asumirán en el eje *x* (de 0 a 30); veamos:

```

R Studio

dotchart(t(Fincas), main="Composición de hatos",
         color=c("blue", "red"), xlim=c(0, 30))

```



Nuestro objetivo con este capítulo consistió en mostrar los comandos básicos para la generación de gráficos sencillos; el lector podrá consultar otro tipo de gráficos dentro de comandos o librerías específicas de análisis estadísticos.

## Bibliografía

La siguiente bibliografía fue citada o consultada para la elaboración del libro:

**Cerón-Muñoz MF, Arboleda ZE 2008** Parentesco, endogamia y deriva genética. En MA Elzo y MF Cerón-Muñoz (eds.), Modelación aplicada a las ciencias animales. Medellín: Editorial Biogénesis, Pp. 67-104.

**Conesa GDV sf** Curso Introducción R: Sesión 2. Grup d'Estadística Espacial i Temporal en Epidemiologia i Medi Ambient. Universitat de València. <http://www.uv.es/conesa/CursoR/material/handout-sesion2.pdf>

\_\_\_\_\_ **sf** Curso Introducción R: Sesión 5. Grup d'Estadística Espacial i Temporal en Epidemiologia i Medi Ambient. Universitat de València. <http://www.uv.es/conesa/CursoR/material/handout-sesion5.pdf>

**Deepayan S 2008** lattice: Multivariate data visualization with R. R package version 0.20-10, Springer, Nueva York. ISBN 978-0-387-75968-5. <http://lmdvr.r-forge.r-project.org>

**Di Rienzo JA, Casanoves F, González LA, Tablada EM, Díaz MP, Robledo CW, Balzarini MG 2005** Estadística para las ciencias agropecuarias. 6<sup>a</sup> ed. (edición electrónica). Córdoba Argentina: Editorial las Brujas. 329p. <http://es.scribd.com/doc/52816014/201/Unidad-experimental>

**Fox J, Weisberg S 2011** An R Companion to Applied Regression, 2<sup>a</sup> ed. Thousand Oaks CA: Sage. URL: <http://socserv.socsci.mcmaster.ca/jfox/Books/Companion>

**García-Ligero MJ, Román RP sf** Simulación con R. [http://www.ugr.es/~proman/PDF/Simulacion\\_R.pdf](http://www.ugr.es/~proman/PDF/Simulacion_R.pdf)

**Genz A, Bretz F 2009** Computation of Multivariate Normal and t Probabilities. Lecture Notes in Statistics, Vol. 195., Springer-Verlage, Heidelberg.

**Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T 2012** mvtnorm: Multivariate Normal and t Distributions. R package version 0.9-9993. <http://CRAN.R-project.org/package=mvtnorm>

**González RJM 2011** Simulación. Ensayo: UNIDAD II: Números Aleatorios y Pseudo aleatorios, UNIDAD III: Variables Aleatorias-Abasolo MA, Carranza GAJ, De Luna PNY, Hernández MD, Medrano MER. Ingeniería Industrial, Instituto Tecnológico de Reynosa. <http://www.slideshare.net/albertojea/numeros-pseudoaleatorios-y-variables-aleatorias>

<http://www.slideshare.net/albertojea/numeros-pseudoaleatorios-y-variables-aleatorias>

**Hadfield JD 2010** MCMC Methods for Multi-Response Generalized Linear Mixed Models: The MCMCglmm R Package. *Journal of Statistical Software*, 33(2): 1-22. <http://www.jstatsoft.org/v33/i02/>

**Heitlinger EG 2010** Transcript of Mick Crawley's R course 2010. Imperial College London, Silwood Park. <http://www.docstoc.com/docs/32130429/Transcript-of-Mick-Crawleys-R-course-2010-Imperial-College>

**Lane DM, Scott D, Hebl M, Guerra R, Osherson D, Zimmer H** sf Online statistics education: A multimedia course of study. [http://onlinestatbook.com/Online\\\_Statistics\\\_Education.pdf](http://onlinestatbook.com/Online\_Statistics\_Education.pdf)

**Maindonald JH 2008** Using R for Data Analysis and Graphics: Introduction, Code and Commentary. Australian National University. <http://cran.r-project.org/doc/contrib/usingR.pdf>

**Mendoza H, Bautista G 2002** Probabilidad y estadística. Bogotá: Universidad Nacional de Colombia. Licencia Creative Commons BY-NC-ND. <http://www.virtual.unal.edu.co/cursos/ciencias/2001065/>

**Morrissey M 2012** pedantics: Functions to facilitate power and sensitivity analyses for genetic studies of natural populations. R package version 1.03. <http://CRAN.R-project.org/package=pedantics>

**Provette DB, da Silva FR, Souza TG 2011** Estatística aplicada à ecologia usando o R. Pós-graduação em biologia animal. Universidade Estadual Paulista. São José do Rio Preto. [http://cran.r-project.org/doc/contrib/Provette-Estatistica\\\_aplicada.pdf](http://cran.r-project.org/doc/contrib/Provette-Estatistica\_aplicada.pdf)

**R Core Team 2012a** foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, .... R package version 0.8-51. <http://CRAN.R-project.org/package=foreign>

**R Core Team 2012b** R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. <http://www.R-project.org/>

**Ricci V 2005** Fitting distributions with R. <http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf>

**Ripley B, and from 1999 to Oct 2002 Lapsley M 2012** RODBC: ODBC Database Access. R package version 1.3-6. <http://CRAN.R-project.org/package=RODBC>

**S.A. 2012** The personality project: Using R for psychological research. A simple guide to an elegant package. Versión 4. <http://personality-project.org/r/r.guide.html>

**Searle RS, Willett LS 2001** Matrix Algebra for Applied Economics. Nueva York: Wiley and Sons, Inc.

**Searle SR 1971** Linear Models. Nueva York: Wiley and Sons, Inc. Wiley classics library edition. Published 1997.

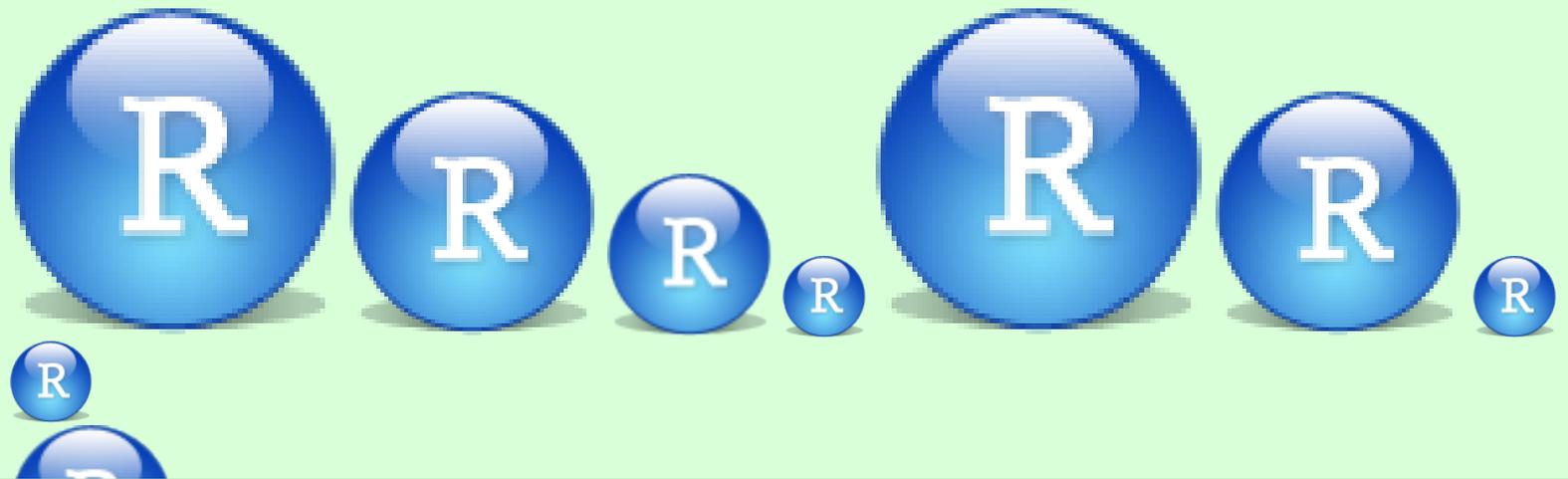
**Searle SR 1982** Matrix Algebra Useful for Statistics (Wiley Series in Probability and Statistics). Nueva York: Wiley and Sons, Inc.

**Soetaert K, Van den Meersche K, Van Oevelen D 2009** limSolve: Solving Linear Inverse Models. R-package version 1.5.1.

**Van den Meersche K, Soetaert K, Van Oevelen D 2009** xsample(): An R Function for Sampling Linear Inverse Problems. Journal of Statistical Software, Code Snippets, 30: 1-15.

**Venables WN, Ripley BD 2002** Modern Applied Statistics with S. 4<sup>a</sup> ed. Nueva York: Springer. <http://www.stats.ox.ac.uk/pub/MASS4>

**Vergara GO, Hurtado-Lugo N 2008** Álgebra matricial. En: MA Elzo y MF Cerón-Muñoz (eds), Modelación aplicada a las ciencias animales. Medellin: Editorial Biogénesis, Pp 13-38.



Mario Fernando Cerón-Muñoz



Luis Fernando Galeano Vasco



Jeanneth Mosquera Rendón