Ecuaciones y Sistemas Lineales con R

Ramon Ceballos

22/2/2021

Ecuaciones y Sistemas Lineales con R

En esta sección veremos como:

- Trabajar con sistemas compatibles determinados
- Representar ecuaciones de un sistema lineal
- Utilizar el Método de Gauss
- Trabajar con sistemas compatibles indeterminados
- Trabajar con sistemas incompatibles
- Resolver ecuaciones matriciales

1. Sistemas compatibles determinados con `R`

1.1. Empleo de solve() para la resolución

Dado el sistema de ecuaciones lineales:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{cases}$$

Si lo pasamos a su forma matricial, $AX = b$, podremos resolverlo de forma sencilla con la función `solve(A,b)`, siempre que se trate de un sistema compatible determinado.

Ejemplo 1

Dado el siguiente sistema de ecuaciones lineales, calculemos su solución:

$$\begin{cases} x_1 + x_2 + 2x_3 = 9 \\ 2x_1 + 4x_2 - 3x_3 = 1 \\ 3x_1 + 6x_2 - 5x_3 = 0 \end{cases}$$

Observemos que, en este caso, se trata de un sistema de 3 ecuaciones con 3 incógnitas.

Lo pasamos a su forma matricial:

```
A = rbind(c(1,1,2),c(2,4,-3),c(3,6,-5))
b = c(9,1,0)
AB = cbind(A,b)
```

Comprobamos que el rango de A es igual al de la ampliada, para ver si el sistema es compatible.

```
qr(A)$rank==qr(AB)$rank
```

```
## [1] TRUE
```

Ahora, comprobamos si es igual al número de incógnitas, para ver si es compatible determinado.

```
qr(A)$rank==3
```

```
## [1] TRUE
```

Finalmente, una vez visto que se trata de un sistema compatible determinado por el Teorema de Rouché-Frobenius, resolvemos:

```
solve(A,b)
```

```
## [1] 1 2 3
```

En definitiva, la solución de nuestro sistema es:

$$x_1 = 1,\ x_2 = 2,\ x_3 = 3$$

Para comprobar que se trata de la solución correcta, solamente tenéis que sustituir la solución obtenida en cada una de las ecuaciones y ver que se cumplen todas y cada una de las igualdades.

Otra forma de comprobar que la solución obtenida es la correcta, es realizando el siguiente producto de matrices y ver que es igual al vector de términos independientes.

```
solution = c(1,2,3)
A%*%solution
```

```
##      [,1]
## [1,]    9
## [2,]    1
## [3,]    0
```

```
A%*%solution == b
```

```
##      [,1]
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
```

Con lo cual, efectivamente, el vector $(1, 2, 3)$ es solución de nuestro sistema. Además es la única, ya que recordemos que se trataba de un sistema compatible determinado.

1.2. Empleo de matlib para resolverlo

Hay otra forma de resolver sistemas compatibles determinados. La librería **matlib** nos ofrece, aparte de la función **Solve**, muchas otras funciones que pueden sernos de utilidad a la hora de resolver sistemas de ecuaciones lineales. Sobre todo, los de 2 o 3 ecuaciones.

```
library(matlib)
```

Atención. Para los usuarios de Mac, a la hora de utilizar **matlib**, tendréis que ir a xquartz.org e instalar el paquete. Si no, no os funcionará todo lo relativo a esta librería.

Ejemplo 2

Resolvamos esta vez un sistema de dos ecuaciones con dos incógnitas:

$$\begin{cases} 2x_1 & + & 2x_2 & = & 1 \\ -x_1 & + & x_2 & = & 2 \end{cases}$$

Si lo expresamos en su forma matricial, obtenemos:

```
A = rbind(c(2,2),c(-1,1))
b = c(1,2)
AB = cbind(A,b)
```

Una vez lo tenemos en forma matricial, podemos mostrarlo con la función **showEqn()**.

```
showEqn(A, b)
```

```
##  2*x1 + 2*x2  =  1
## -1*x1 + 1*x2  =  2
```

La librería **matlib** también nos permite calcular el rango de una matriz con la función **R()**.

```
R(A)
```

```
## [1] 2
```

```
R(AB)
```

```
## [1] 2
```

Además, mediante la función **all.equal()**, podemos comprobar si el sistema es compatible. Es decir, si los rangos de A y la matriz ampliada AB son iguales:

```
all.equal(R(A),R(AB))
```

```
## [1] TRUE
```

Como en este caso el rango de la matriz de coeficientes y el de la ampliada coinciden entre sí y con el número de incógnitas, por el Teorema de Rouché-Frobenius podemos concluir que se trata de un sistema compatible determinado.

Finalmente, una vez visto que se trata de un sistema compatible determinado, resolvemos:

```
Solve(A, b, fractions = TRUE)
```

```
## x1    =  -3/4
##   x2  =   5/4
```

Observación. El parámetro **fractions** nos permite mostrar las soluciones no enteras en forma de fracción, siempre que esta exista. Su valor por defecto es FALSE.

2. Representación de sistemas con R

Otra cosa interesante que nos proporciona la librería **matlib** es el poder dibujar las ecuaciones de un sistema mediante las funciones **plotEqn()** o **plotEqn3d()**, en función de si tenemos 2 o 3 incógnitas.

En caso de tener un sistema de dos ecuaciones con dos incógnitas, como por ejemplo el del Ejemplo 2, recordemos:

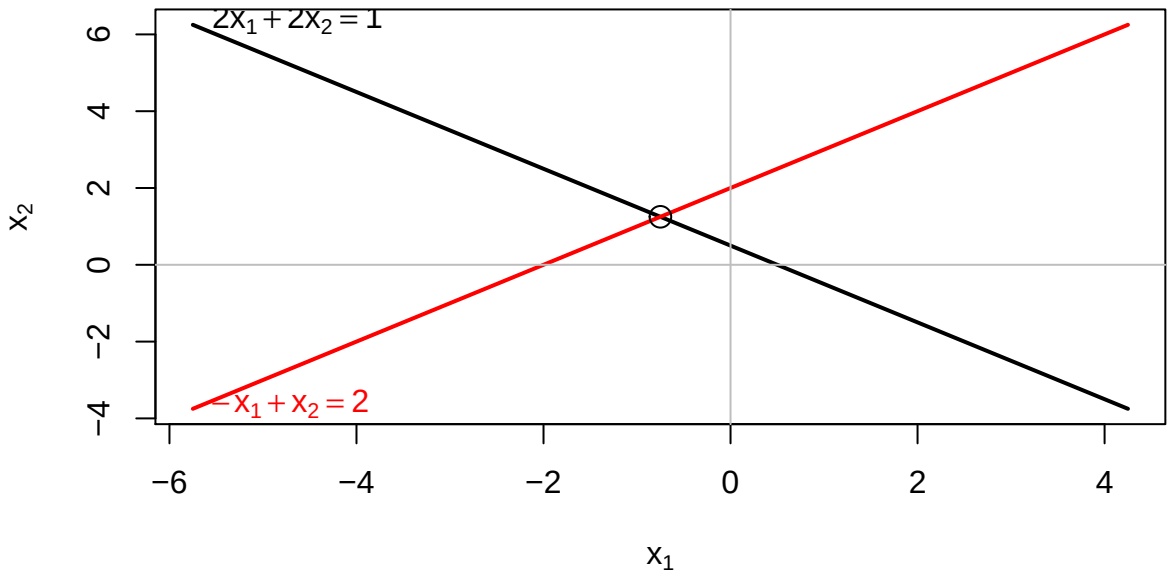
```
showEqn(A,b)
```

```
##  2*x1 + 2*x2  =  1
## -1*x1 + 1*x2  =  2
```

Su representación sería la que se muestra en la siguiente diapositiva.

```
plotEqn(A,b)
```

```
## 2*x[1] + 2*x[2]  =  1
##  -x[1]   + x[2]  =  2
```



La solución que anteriormente habíamos encontrado:

$$\begin{cases} x_1 = -\frac{3}{4} \\ x_2 = \frac{5}{4} \end{cases}$$

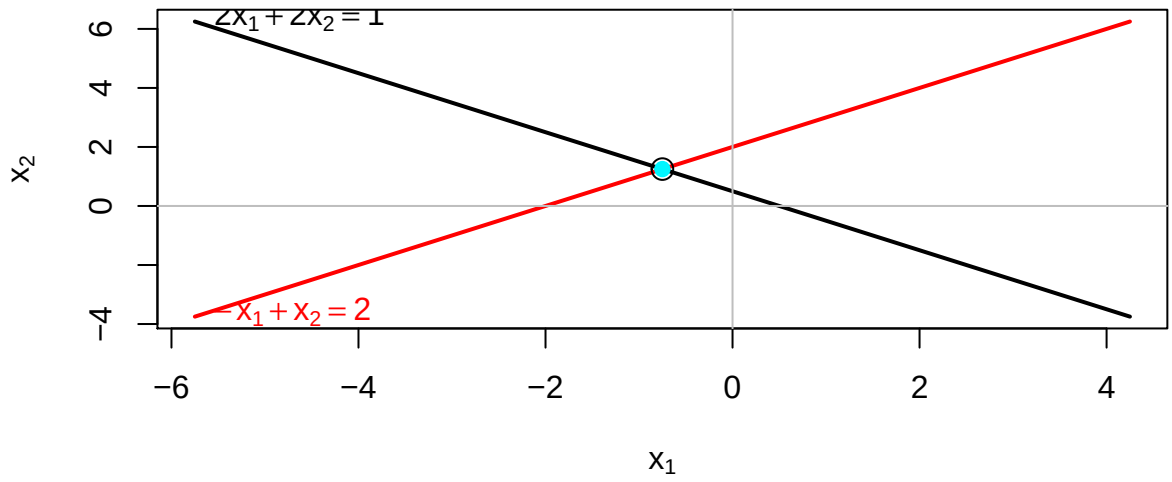
Esta solución es el punto donde ambas rectas intersecan.

Se puede modificar la estética de los ptos representados en el gráfico con **points()**.

```
plotEqn(A,b)
```

```
## 2*x[1] + 2*x[2]  =  1
##  -x[1]   + x[2]  =  2
```

```
points(-3/4,5/4, col = "turquoise1", pch = 19)
```



Ejemplo 3

En caso de tener un **sistema de tres ecuaciones y dos variables**, como por ejemplo:

$$\begin{cases} 4x & + & 2y & = & 3 \\ x & - & 2y & = & 2 \\ 3x & + & 4y & = & 1 \end{cases}$$

Pasándolo a su forma matricial tenemos:

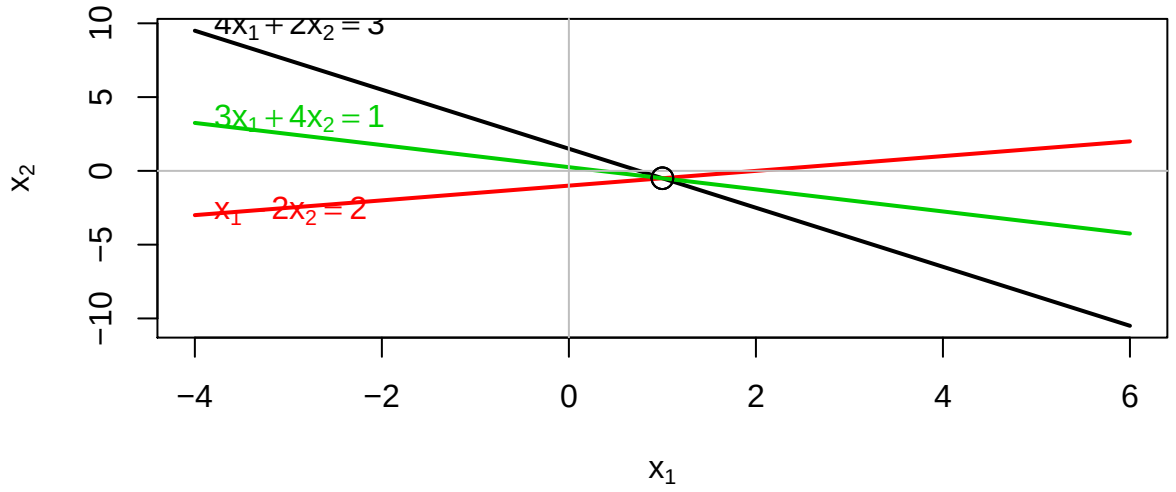
```
A = rbind(c(4,2),c(1,-2),c(3,4))
b = c(3,2,1)
showEqn(A,b)
```

```
## 4*x1 + 2*x2  =  3
## 1*x1 - 2*x2  =  2
## 3*x1 + 4*x2  =  1
```

Y su representación gráfica se consigue, de nuevo, mediante la función **plotEqn()**, ya que volvemos a tener dos incógnitas.

```
plotEqn(A,b)
```

```
## 4*x[1] + 2*x[2]  =  3
##   x[1] - 2*x[2]  =  2
## 3*x[1] + 4*x[2]  =  1
```



Como de nuevo vuelve a haber un punto donde intersecan todas rectas, gráficamente podemos concluir que se trata de un sistema compatible determinado.

Ejercicio 1

- 1. Comprobad que, efectivamente, se trata de un sistema compatible determinado haciendo uso del Teorema de Rouché-Frobenius.
- 2. Calculad su solución.

En primer lugar vemos si los rangos de A y AB coinciden.

```
AB = cbind(A,b)
```

```
#Vía sin librerías
qr(A)$rank == qr(AB)$rank
```

```
## [1] TRUE
```

```r
qr(A)$rank == 2
```

```
## [1] TRUE
```

Como los rangos coinciden y el valor es idéntico al nº de incógnitas concluímos que se trata de un sistema compatible determinado.

Resolvemos el sistema de ecuaciones.

```r
#Vía sin librerías
library(MASS)
as.fractions(qr.solve(A,b))
```

```
## [1]    1 -1/2
```

Ahora empleamos **matlib**.

```r
#matlib

R(A) == R(AB)
```

```
## [1] TRUE
```

```r
R(A) == 2
```

```
## [1] TRUE
```

Como los rangos coinciden y el valor es idéntico al nº de incógnitas concluímos que se trata de un sistema compatible determinado.

Resolvemos el sistema de ecuaciones.

```r
#matlib
Solve(A,b,fractions = TRUE)
```

```
## x1    =     1
##   x2  =  -1/2
##    0  =     0
```

Ejemplo 1

Si, en cambio, tenemos un sistema de 3 ecuaciones con 3 incógnitas, como por ejemplo el sistema del **Ejemplo 1**.

```r
A = rbind(c(1,1,2),c(2,4,-3),c(3,6,-5))
b = c(9,1,0)
showEqn(A,b)
```

```
## 1*x1 + 1*x2 + 2*x3  =  9
## 2*x1 + 4*x2 - 3*x3  =  1
## 3*x1 + 6*x2 - 5*x3  =  0
```

Su representación gráfica se consigue haciendo uso de la función **plotEqn3d**, tal y como se muestra en la siguiente diapositiva.

```
plotEqn3d(A,b, xlim = c(-3,3), ylim = c(0,6))
```

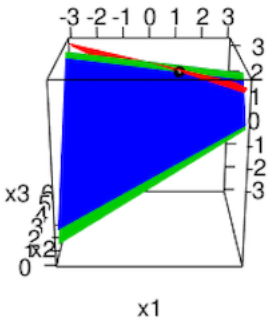


Figure 1: Representación gráfica del sistema del `Ejemplo 1` con `R`

Observación. Esto solo es una imagen, pero el resultado de esta función es interactivo.

En el plot anterior, se puede observar que hay un punto negro. Ese punto, representa la única intersección de los 3 planos. En otras palabras, esa es la solución que habíamos obtenido anteriormente.

Más concretamente, se trata del punto $P = (1, 2, 3)$.

3. Método de Gauss con `R`

La librería **matlib** también nos ofrece la oportunidad de utilizar el Método de Gauss para resolver sistemas de ecuaciones lineales.

La función **echelon()** nos calucla la matriz escalonada reducida de cualquier matriz.

Si reducimos la matriz ampliada a una matriz escalonada reducida equivalente, podremos deducir fácilmente la solución del sistema (suponiendo que la haya).

Ejemplo 1

Recuperemos el sistema de 3 ecuaciones con 3 incógnitas:

$$\begin{cases} x_1 + x_2 + 2x_3 = 9 \\ 2x_1 + 4x_2 - 3x_3 = 1 \\ 3x_1 + 6x_2 - 5x_3 = 0 \end{cases}$$

```
A = rbind(c(1,1,2),c(2,4,-3),c(3,6,-5))
b = c(9,1,0)
AB = cbind(A,b)
```

Entonces, la función **echelon()** aplicada a la matriz ampliada del sistema nos devuelve:

```
echelon(AB)
```

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

De donde, claramente, podemos deducir que la solución es:

$x_1 = 1,\ x_2 = 2$ y $x_3 = 3$

Esto se debe a que la primera columna de la matriz que obtenemos hace referencia a la primera variable; la segunda columna a la segunda variable; la tercera, a la variable x_3; y, la última, al vector de términos independientes (modificado debido a las operaciones realizadas con el Método de Gauss).

Por tanto, que en la primera fila haya un **1** en la primera columna y **0's** en la segunda y tercera columnas, nos indica que la variable x_1 toma el valor que haya en la última columna de esa fila. Es decir, $x_1 = 1$.

Por otro lado, en la segunda fila hay un 1 en la segunda columna (y 0's en la primera y la tercera), lo cual nos informa de que en este caso es la variable x_2 la que tomará el valor que se encuentra en la última columna de esa misma fila, la cual corresponde al vector de términos independientes. En otras palabras, $x_2 = 2$.

Siguiendo el mismo razonamiento, llegamos a la conclusión de que $x_3 = 3$.

La función **`echelon()`** tiene el parámetro **`verbose`** que, igualado a **`TRUE`**, nos muestra las operaciones elementales llevadas a cabo para conseguir la matriz escalonada reducida.

Merece la pena que lo comprobéis vosotros mismos en la consola, ya que en estas diapositivas no cabe toda esa información.

Además, la función **`echelon()`** también cuenta con el parámetro **`fractions`**, muy útil para no estar arrastrando decimales.

```
echelon(AB, verbose = TRUE, fractions = TRUE)
```

```
##
## Initial matrix:
##               b
## [1,]  1  1  2  9
## [2,]  2  4 -3  1
## [3,]  3  6 -5  0
##
## row: 1
##
##  exchange rows 1 and 3
##               b
## [1,]  3  6 -5  0
## [2,]  2  4 -3  1
## [3,]  1  1  2  9
##
##  multiply row 1 by 1/3
##                     b
## [1,]    1    2 -5/3    0
## [2,]    2    4   -3    1
## [3,]    1    1    2    9
##
```

```
##  multiply row 1 by 2 and subtract from row 2
##                     b
## [1,]    1    2 -5/3    0
## [2,]    0    0  1/3    1
## [3,]    1    1    2    9
##
##  subtract row 1 from row 3
##                     b
## [1,]    1    2 -5/3    0
## [2,]    0    0  1/3    1
## [3,]    0   -1 11/3    9
##
## row: 2
##
##  exchange rows 2 and 3
##                     b
## [1,]    1    2 -5/3    0
## [2,]    0   -1 11/3    9
## [3,]    0    0  1/3    1
##
##  multiply row 2 by -1
##                        b
## [1,]     1     2  -5/3     0
## [2,]     0     1 -11/3    -9
## [3,]     0     0   1/3     1
##
##  multiply row 2 by 2 and subtract from row 1
##                        b
## [1,]     1     0  17/3    18
## [2,]     0     1 -11/3    -9
## [3,]     0     0   1/3     1
##
## row: 3
##
##  multiply row 3 by 3
##                        b
## [1,]     1     0  17/3    18
## [2,]     0     1 -11/3    -9
## [3,]     0     0     1     3
##
##  multiply row 3 by 17/3 and subtract from row 1
##                        b
## [1,]     1     0     0     1
## [2,]     0     1 -11/3    -9
## [3,]     0     0     1     3
##
##  multiply row 3 by 11/3 and add to row 2
##           b
## [1,] 1 0 0 1
## [2,] 0 1 0 2
## [3,] 0 0 1 3
```

4. Sistemas compatibles indeterminados con R

En caso de tener sistemas compatibles indeterminados, no podemos aplicar directamente la función **solve()**, ya que ésta solamente está pensada para sistemas compatibles determinados.

No obstante, con todas las funciones presentadas anteriormente, es fácil averiguar si el sistema con el que estamos trabajando es un sistema compatible indeterminado y, además, podemos resolverlo.

Ejemplo 4

Dado el siguiente sistema de ecuaciones lineales:

$$\begin{cases} x & + & y & - & z & = & 2 \\ x & - & y & + & z & = & 1 \\ 3x & + & y & - & z & = & 5 \end{cases}$$

Vemos que se trata de un sistema de 3 ecuaciones con 3 incógnitas. Pasándolo a su forma matricial, obtenemos:

```
A = matrix(c(1,1,-1,1,-1,1,3,1,-1), byrow = TRUE, nrow = 3, ncol = 3)
b = c(2,1,5)
AB = cbind(A,b)
```

Si comprobamos los rangos de la matriz de coeficientes A y la matriz ampliada del sistema, obtenemos:

```
c(R(A),R(AB))
```

```
## [1] 2 2
```

```
all.equal(R(A),R(AB))
```

```
## [1] TRUE
```

Sin embargo, a pesar de que los rangos sean iguales, el número de incógnitas es 3. Por tanto, por el Teorema de Rouché-Frobenius, concluimos que se trata de un sistema compatible indeterminado.

La función **echelon()** puede también indicarnos que se trata de un sistema compatible indeterminado:

```
echelon(AB)
```

```
##               b
## [1,] 1 0  0 1.5
## [2,] 0 1 -1 0.5
## [3,] 0 0  0 0.0
```

En este caso, nos está indicando que la variable x_3 es libre. Es decir, puede tomar cualquier valor en $\mathbb{R}$.

En caso de no verlo tan claro, la función **Solve()** lo muestra más claramente.

```r
Solve(A,b, fractions = TRUE)
```

```
## x1           =  3/2
##   x2 - 1*x3  =  1/2
##           0  =    0
```

Vemos que $x_1 = \frac{3}{2}$, x_3 es libre y x_2 depende del valor que tome x_3.

Es decir, nuestra solución es de la forma:

$$x_1 = \frac{3}{2},\ x_2 = \frac{1}{2} + x_3,\ x_3 \in \mathbb{R}$$

Con lo cual, nuestro conjunto de soluciones para este sistema es infinito.

Existe otra opción, que es dibujar el sistema.

En este caso, al tener 3 variables, necesitamos utilizar `plotEqn3d()`.

```r
plotEqn3d(A,b, xlim = c(-10,10), ylim = c(-10,10), zlim = c(-10,10))
```

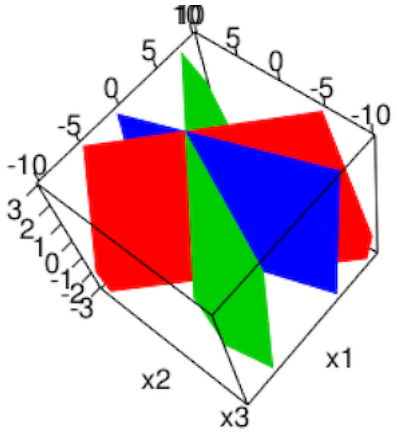


Figure 2: Representación gráfica del sistema del `Ejemplo 4` con `R`

Gráficamente, podemos concluir que el conjunto de soluciones es infinito pues se trata de una recta: la interesección de los 3 planos.

De hecho, podemos representar nuestro conjunto de soluciones por:

$$\left\{x \in \mathbb{R}^3\ :\ x_1 = \frac{3}{2},\ x_2 = \frac{1}{2} + x_3,\ x_3 \in \mathbb{R}\right\}$$

$$(x_1, x_2, x_3) = \left(\frac{3}{2}, \frac{1}{2}, 0\right) + \lambda(0, 1, 1)$$

5. Sistemas incompatibles con `R`

Ejemplo 5

Con todas las funciones presentadas anteriormente, es fácil averiguar si el sistema con el que estamos trabajando es un sistema incompatible.

Supongamos que nos dan el siguiente sistema de ecuaciones lineales:

$$\begin{cases} x & + & y & = & 2 \\ x & - & y & = & 1 \\ 2x & + & y & = & 3 \end{cases}$$

Tenemos un sistema de 3 ecuaciones con dos incógnitas, donde:

```
A = cbind(c(1,1,2),c(1,-1,1))
b = c(2,1,3)
AB = cbind(A,b)
```

Si comprobamos los rangos de la matriz de coeficientes A y la matriz ampliada del sistema, obtenemos:

```
c(R(A),R(AB))
```

```
## [1] 2 3
```

```
all.equal(R(A),R(AB))
```

```
## [1] "Mean relative difference: 0.5"
```

Entonces, como los rangos son diferentes, por el Teorema de Rouché-Frobenius, concluimos que se trata de un sistema incompatible.

Mediante la función **echelon()**, también podemos ver fácilmente que se trata de un sistema incompatible.

```
echelon(AB)
```

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

Si no lo veis tan claro, **Solve()** lo muestra aún más explicitamente:

```
Solve(A,b, fractions = TRUE)
```

```
## x1    =  4/3
##   x2  =  1/3
##    0  =  1/3
```

Tanto con **echelon()** como con **Solve()**, lo que obtenemos son igualdades como:

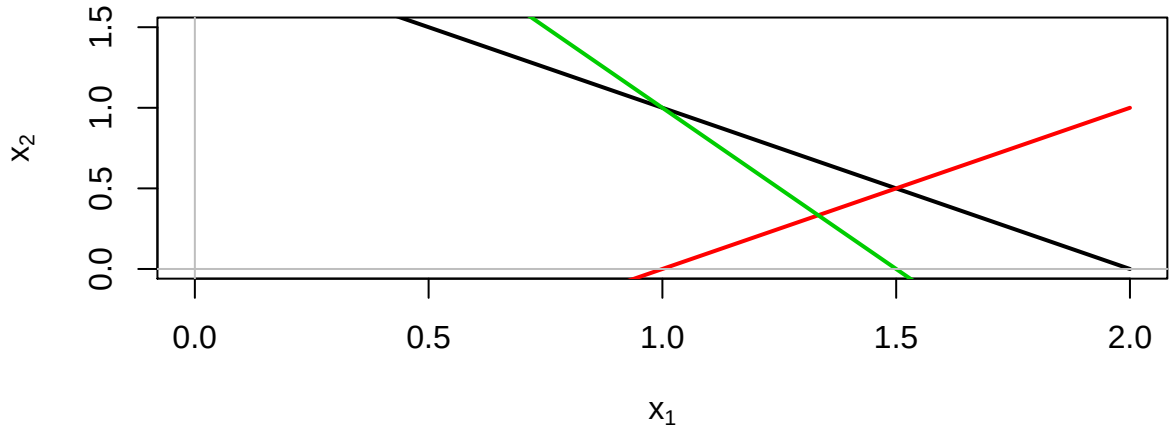
$$0 = 10 = \frac{1}{3}$$

Estas expresiones de igualdad claramente son falsas. Esto es lo que nos lleva a concluir que se trata de un sistema incompatible.

Finalmente, si lo que queréis es una comprobación gráfica, la función **plotEqn()** es la que nos servirá en este caso:

```
plotEqn(A,b, xlim = c(0,2), ylim = c(0,1.5), solution = FALSE)
```

```
##   x[1]   + x[2]  =  2
##   x[1] - 1*x[2]  =  1
## 2*x[1]   + x[2]  =  3
```



En el plot anterior, podemos ver que cada par de rectas tiene solución. Es decir, las rectas intersecan 2 a 2.

No obstante, vemos que no existe punto en que las 3 rectas intersequen.

Este es el motivo por el cual gráficamente podemos concluir que se trata de un sistema incompatible.

6. Ecuaciones matriciales con R

Dada una ecuación matricial, si la tenemos de la siguiente forma $AX = B$, donde A, B son matrices, entonces la podemos resolver haciendo uso de la función **solve(A,B)**.

Ejemplo 6

Sea la ecuación matricial:

$$AX + 3B = (C + D)X + 3D + 10I_2$$

De este ecuación tenemos:

$$A = \begin{pmatrix} 0 & 4 \\ 2 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 2 \\ 3 & -2 \end{pmatrix}, \quad D = \begin{pmatrix} -2 & 1 \\ -1 & 1 \end{pmatrix}, \quad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Pasando los términos multiplicados por X a la izquierda y los que no a la derecha, obtenemos:

$$AX - (C + D)X = 3D - 3B + 10I_2$$

Sacando factor común X por la derecha en el primer término y factor común 3 en el segundo término, tenemos:

$$(A - (C + D))X = 3(D - B) + 10I_2$$

Nuestras matrices son:

```
A = rbind(c(0,4),c(2,1))
B = rbind(c(1,-1),c(2,3))
C = rbind(c(1,2),c(3,-2))
D = rbind(c(-2,1),c(-1,1))
I = diag(1, nrow = 2, ncol = 2)
```

La matriz que multiplica a X, a la que llamaremos M, es:

```
M = A-(C+D)
```

mientras que la matriz del segundo miembro de la igualdad, N, es:

```
N = 3*(D-B)+10*I
```

Una vez calculadas M y N, tenemos la ecuación matricial de la forma $MX = N$. Entonces, la podemos resolver realizando lo siguiente:

```
library(MASS)
X = solve(M,N)
as.fractions(X)
```

```
##      [,1] [,2]
## [1,] 11/2    4
## [2,] -9/2    2
```

Comprobamos nuestro resultado:

```
A%*%X + 3*B == (C+D)%*%X+3*D+10*I
```

```
##      [,1] [,2]
## [1,] TRUE TRUE
## [2,] TRUE TRUE
```