Sistemas de ecuaciones lineales y ecuaciones matriciales en R

Daniel Eduardo Macias Estrada

27/8/2020

Resolución de un sistema compatible determinado con la función solve

Dado el sistema de ecuaciones lineales

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

Al representarlo en su forma matricial $AX = b$, podemos darle una solución con el uso de la función **solve(A,b)**, la cual funciona si, y solo si, el sistema de ecuaciones es compatible determinado.

Ejemplo 1

Dado el siguiente sistema de ecuaciones lineales, calculemos su solución

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

En este caso, tanto el número de ecuaciones, así como el número de incógnitas es 3

En su forma matricial, se expresaría en R como:

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

Observamos en el anterior chunk que la función **cbind()** es usada para unir dos matrices en columnas, solo si la dimensión de ambas lo permite. En le documento de **Matrices con R** se menciona la misma función, pero con un uso distinto.

Mediante el teorema de Rouché-Frobenius, y con ayuda de R, verificamos si el rango de la matriz A es igual al rango de la matriz Ampliada, para saber si el sistema es compatible

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

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

Ahora nos aseguramos si este es igual al número de incógnitas, el cual es 3, para saber si es un sistema compatible determinado.

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

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

De esta manera, es posible aplicar la funcíón **solve()** a este caso, por lo que procedemos a realizarla

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

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

La solución que nos devuelve, se puede expresar como:

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

Una forma de comprobar este resultado es sustituyendo días variables en cada una de las ecuaciones del sistema y verificar la igualdad en todas.

Otra forma de comprobar el resultado, es realizando un producto de matrices con la matriz A y el vector de resultados

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

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

o de la siguiente forma

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

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

Así, nos damos cuenta que el vector (1,2,3) es correcto

Resolución de un sistema compatible determinado con la libreria matlib

Otra forma para dar solución a un sistema determinado, se da gracias a la libreria **Matlib**, la cual ofrece variedad de funciones útiles en resolver sistemas de ecuaciones lineales, sobre todos aquellos con 2 o 3 ecuaciones

Ejemplo 2

Dada el siguiente sistema de ecuaciones lineales

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

Al expresarlo en su forma matricial, obtenemos

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

Una vez tenemos la forma matricial, podemos mostrar el sistema con la función **showEqn()** de la libreria **matlib**, la cual recibe como parámetros en primer lugar la matriz de coeficientes y en segundo la matriz de términos independientes

```
library(matlib)
showEqn(A,b)
```

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

Esta misma libreria nos permite obtener el rango de de una matriz con la función **R()**

```
R(A)
```

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

```
R(Ab)
```

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

Gracias a esto podemos determinar el tipo de sistema con el que se trabaja. En este caso, al ser igual los rangos de ambas matrices, del mismo modo que es igual al número de incógnitas, podemos decir que el sistema es compatible determinado.

Por otro lado, con la función **all.equal()** es posible comprobar que un sistema es compatible. Esta función compara los rangos de matrices, y determinar si son iguales

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

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

Finalmente, con la función **Solve()** de matlib podemos resolver el sistema.

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

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

Cabe resaltar que el parámetro **fractions**, que recibe un valor booleano, permite mostrar las soluciones no enteras en forma de fracción, siempre que exista

Representación de sistemas con R

Otro punto importante de la libreria **matlib** es el de representar de manera gráfica las ecuaciones de un sistema lineal. Con las funciones **plotEqn()** para dibujar un sistema lineal de 2 incógnitas, y **plotEqn3d()** para dibujar un sistema lineal de 3 incógnitas.

Los parámtetros que reciben son: la matriz de coeficientes y la matriz de términos independientes

Ejemplo 2

En caso de tener un sistema de dos ecuaciones con dos incógnitas

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

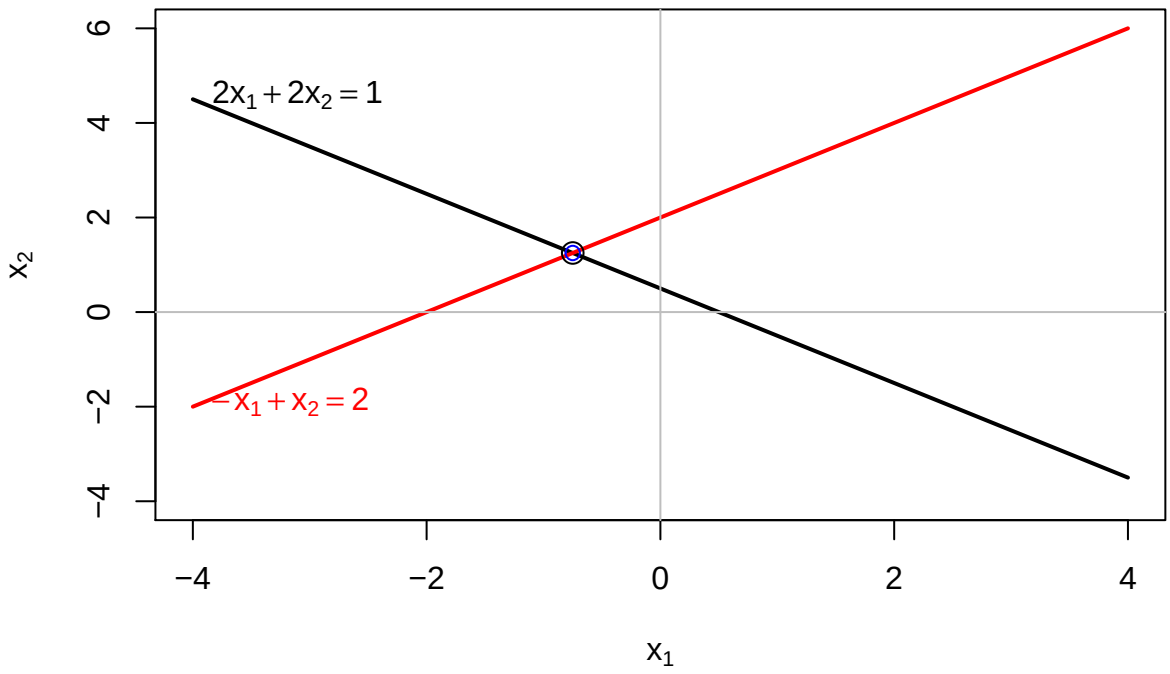
```
##  2*x1 + 2*x2  =  1
## -1*x1 + 1*x2  =  2
```

Su representación sería

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

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

```
points(-3/4, 5/4, col = "blue", pch = 21)
```



La solución que anteriormente habíamos encontrado

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

es el punto donde ambas rectas intersectan

Además, con la función **points()**, se puede indicar un punto dentro del plano. En este caso, se coloco, en el punto de intersección.

Ejemplo 3

En caso de tener un sistema de 3 ecuaciones y 2 variables, como el siguiente caso:

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

Pasándolo a su forma matricial tenermos

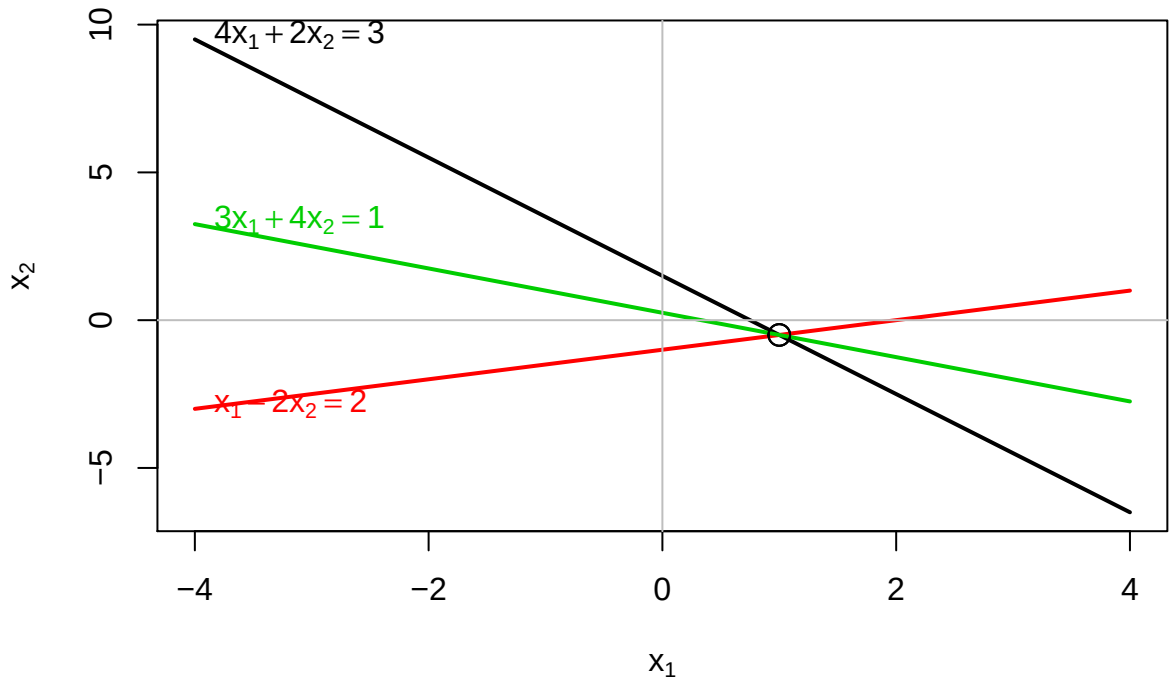
```
A = rbind(c(4,2), c(1,-2), c(3,4))
b = c(3,2,1)
showEqn(A,b)
```

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

Para su representación gráfica, se vuelve a usar la función **plotEqn**

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

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



Debido a que existe un punto de intersección para todas las rectas, significa que el sistema es compatible determinado

Ejemplo 1

Si en lugar de tener un sistema de 2 incógnitas, como los anteriores, tenemos un sistema de 3 ecuaciones como es el casi dek ejemplo 1

```
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
```

Como se menciono antes, este sistema al ser de 3 incognitas, para su representación gráfica haremos uso de la función **plotEqn3d()**,

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

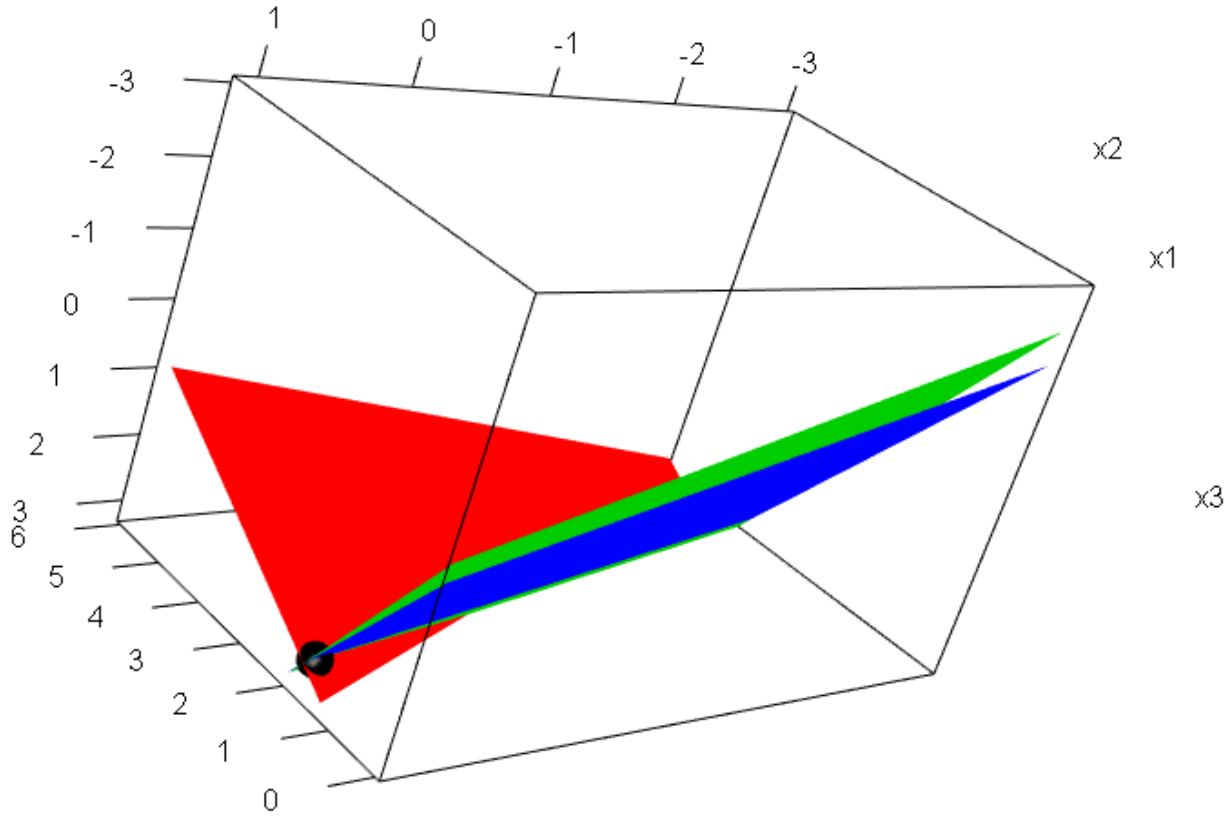


Figure 1: Representación gráfica del primer ejemplo con R

El resultado de la línea de código anterior es interactivo, no una imagen

Como se ve en la gráfica, un punto negro sobresale, el cual indica el punto donde se intersectan las 3 ecuaciones representadas mediante planos. A su vez, este punto indica la solución dada al inicio, es decir, el punto de intersección es aquel ocn las coordenadas $P = (1, 2, 3)$

Método de Gauss

Tal es la potencia de la librería **matlib**, que podemos hacer una implementación del método de Gauss, con el fin de dar una resolución a sistemas de ecuaciones lineales.

Esto se realiza con la función **echelon()**, la cual calcula la escalonada reducida de cualquier matriz

La relevancia de esta función, recae en que al ser la escalonada reducida lo que calcula, nos permite deducir el resultado.

Ejemplo 1

Recuperamos 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)
```

Ahora, podremos usar la función **echelon()** pasándole como parámetro la matriz ampliada

```
echelon(AB)
```

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

De la matriz escalonada reducida, deducimos los resultados para cada incógnita

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

Esto se debe a que cada columna representa una incógnita, mientras que la última representa los términos independientes.

Por otra parte, cabe recalcar que la función anteriormente usada tiene un parámetro **verbose**, que igualado a **TRUE** muestra las operaciones elementales llevado a cabo para conseguir la matriz escalonada reducida.

A su vez, el parámetro **fractions** está presente igualmente en esta función.

```
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
```

Resolución de un sistema compatible indeterminado

Este tipo de sistemas no son posibles de resolver con la función **solve()**, pues solo resuelve sistemas compatibles determinados

Sin embargo, **matlib**, nos ofrece la posibilidad de resolver este tipo de casos,

Ejemplo 4

Dado el siguietne sistema de ecuaciones lineales:

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

Se trata de un sistema de 3 ecuaciones y 3 incógnitas. Ahora lo pasaremos a su forma matricial

```
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)
```

Como lo hemos hecho antes, pasaremos a verificar si el sistema es compatible indeterminado. Comprobamos los rangos de la matriz A y AB

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

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

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

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

Observamos, que el rango es el mismo entre las matrices, pero menor al número de incógnitas del sistema. Por el Teorema de Rouché-Frobenius se clasifica este sistema como compatible indeterminado.

Otra forma de comprobar que, en efecto es un sistema indeterminado, podemos hacer uso de la función **echelon()**

```
echelon(AB)
```

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

El resultado que nos ofrece es interesante, nos indica que la columna de la incógnita z no tiene pivote y, por lo tanto, es un término independiente el cual puede tomar cualquier valor en $\mathbb{R}$

Gracias a la representación anterior, es posible deducir el resultado. no obstante lo podemos tener más claro si hacemos uso de la función **Solve()**

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

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

De esta manera, 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

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

Otra forma de visualizar el resultado del sistema es mediante una representación gráfica del mismo

En este caso, por tener 3 incógnitas, se usará la función **plotEqn3d()**

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

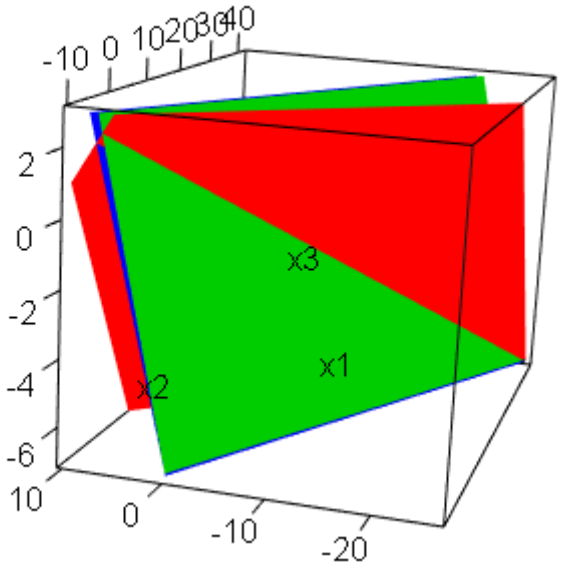


Figure 2: Representación gráfica del ejemplo 4

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

De hecho, podemos representar nuestro conjunto de soluciones por

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

Esto se suele también escribir en formato vectorial, donde el punto (x_1, x_2, x_3) quiere decir que se trata del vector de resultados

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

Sistemas incompatibles

Gracias a las funciones usadas con anterioridad, es fácil determinar si el sistema es incompatible

Ejemplo 5

Supongamos que contamos con el siguiente sistema de ecuaciones lineal, en el cual tenemos 3 ecuaciones y 2 incógnitas:

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

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

Al comparar los rangos de la matriz de coeficientes y de la matriz ampliada, observamos que:

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

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

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

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

Dicha observación, en base al Teoream de Rouché-Frobenius, concluye que el sistema es incompatible.

Tanto con la función **echelon()**, como la función **Solve()** nos ayudan a ver que es así.

```
echelon(AB)
```

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

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

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

En ambos casos nos muestra las siguientes igualdades

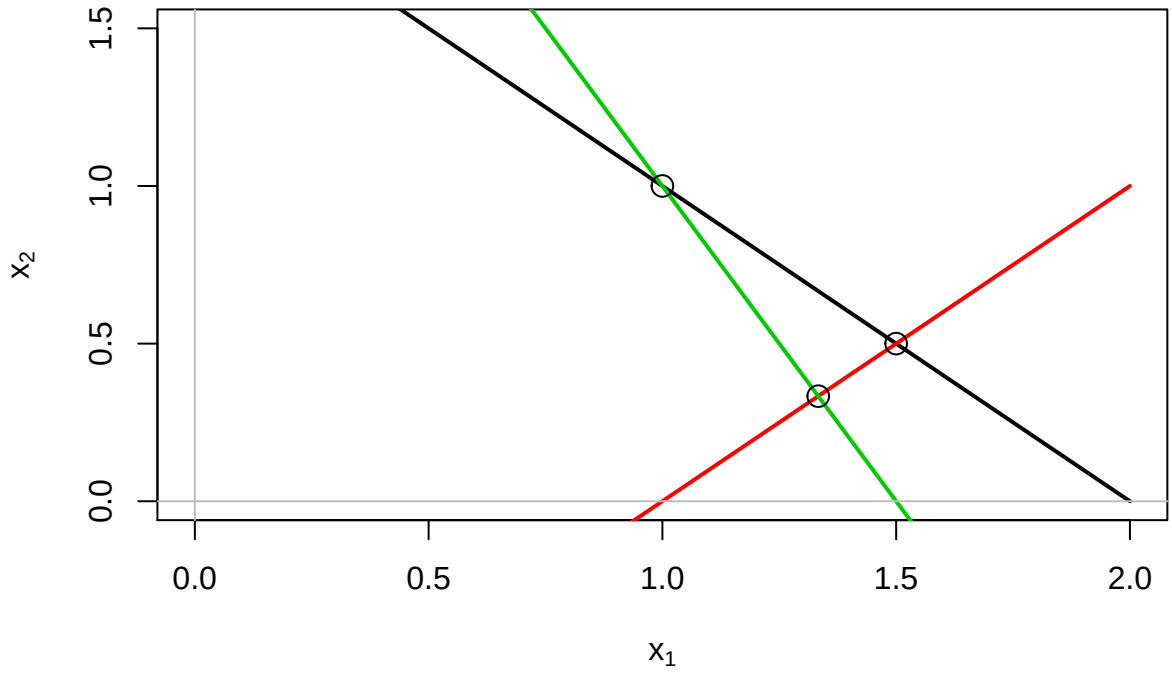
$$0 = 1 \quad \text{y} \quad 0 = \frac{1}{3}$$

Estas, por obviedad, son falsas. De esta manera concluimos que el sistema es incompatible

Otra forma de verificar dicho resultado es de manera gráfica:

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

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



En dicha gráfica, observamos puntos donde 2 rectas intersectan, por lo que podemos definir a dichos puntos, soluciones para cada par de ecuaciones. Sin embargo, el sistema cuenta con 3 ecuaciones, y no existe punto en común para dichas ecuaciones. En otras palabras, podemos concluir lo mismo, el sistema es incompatible.

Ecuaciones matriciales

Para este tema, tenemos que recordar que dada una ecuación matricial, si la tenemos de la forma $AX = B$, donde A, B son matrices, entonces se puede resolver con la función **solve(A,B)**

Ejemplo 6

Sea la ecuación matricial

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

donde

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

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 de la igualdad, además de sacar factor común 3 a la izquierda

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

Debemos de ser conscientes que tanto $(A-(C+D))$ como $3(D-B)+10I_2$ son matrices por si mismas

Ahora procederemos a hacer uso de R para darle una oslución

```
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)

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

Calculadas M y N, la ecuación que resulta es $MX = N$, que como se menciona al comienzo del tema, se puede resolver con la ya mencionada función.

```
X = solve(M,N)
X
```

```
##      [,1] [,2]
## [1,]  5.5    4
## [2,] -4.5    2
```

Ahora pasamos a comprobar dicho resultado

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

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