2. Programación Lineal

Javier Ortín

2026-02-12

Ejercicios propuestos

Solución gráfica

Ejercicio 2.1.3 (a)

Buscamos minimizar la función $-x_1 + 3x_2$ con las siguientes restricciones:

$$\begin{cases} x_1 - x_2 \leq 4 \\ x_1 + 2x_2 \geq 4 \\ x_1, x_2 \geq 0 \end{cases}$$

Comenzamos definiendo las restricciones:

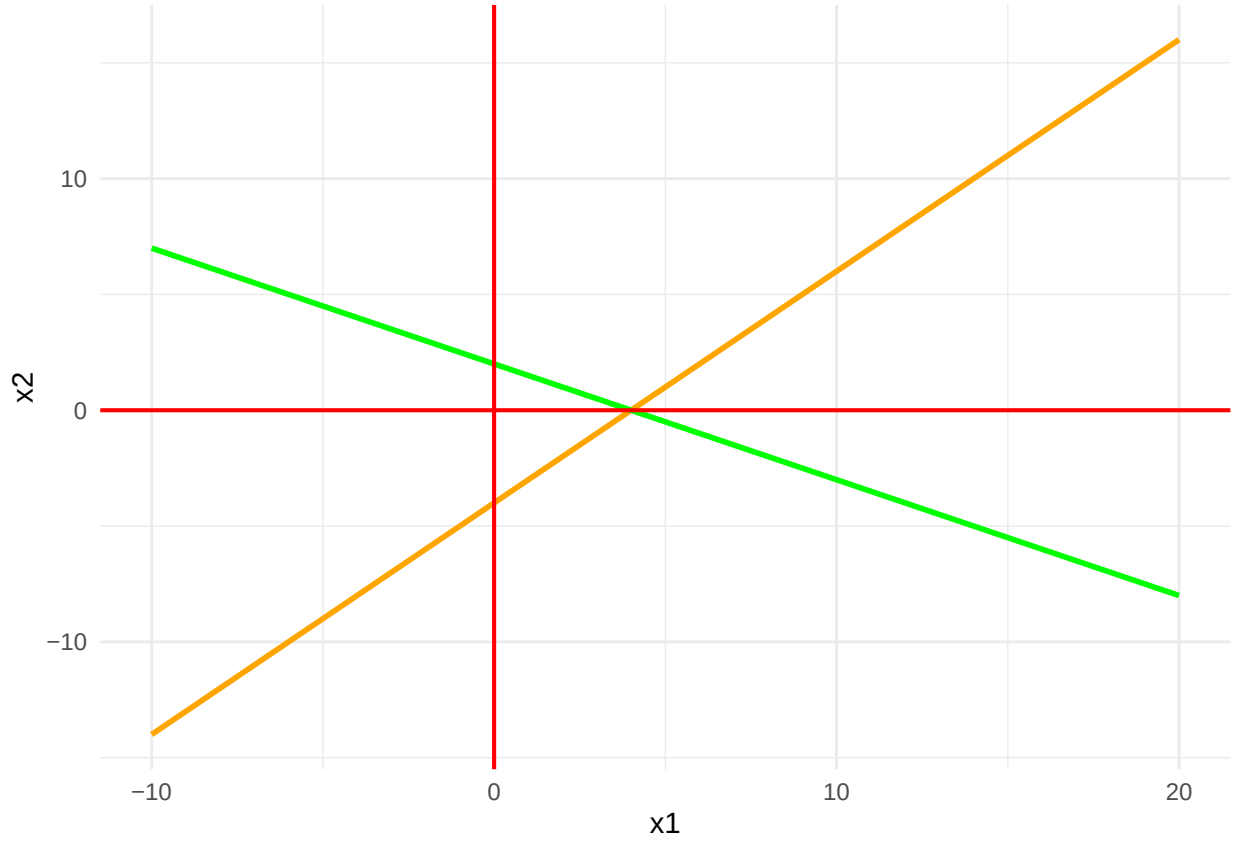
```
R1 = function(x1) x1 - 4
R2 = function(x1) 2 - x1/2
```

Despejando, sololа variable x_1 solo toma valores en $[0, 4]$. Este será nuestro rango de representación para graficar la región factible.

```
x1 = seq(from=-10, to=20, length.out=1000) # Valores de x1 en el rango deseado
datos = data.frame(x1=x1,
                   x2=R1(x1),
                   x2.2=R2(x1))

p = ggplot(data=datos, aes(x=x1)) +
  geom_line(aes(y=x2), col="orange", size=1) +
  geom_line(aes(y=x2.2), col="green", size=1) +
  geom_vline(xintercept = 0, col="red", size=0.72) +
  geom_hline(yintercept = 0, col="red", size=0.72) +
  theme_minimal()
```

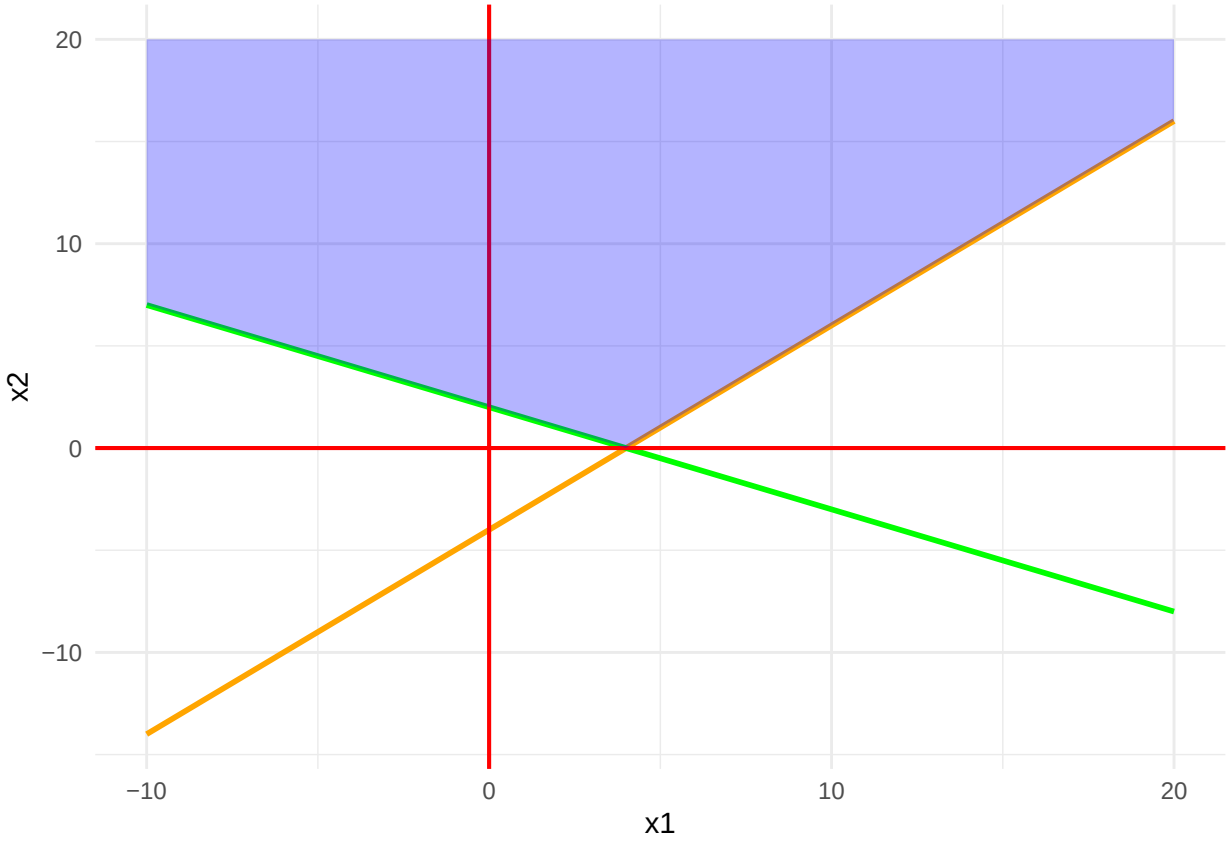
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
p
```



Ya hemos visto las restricciones, queda ver el área plausible:

```
p = p + geom_ribbon(data = transform(datos,
                                     maxY= 20,
                                     minY= pmax(datos$x2, datos$x2.2)),
    aes(ymin=minY, ymax=maxY), fill="blue", alpha=0.3)
p
```



Ahora queda hallar los puntos extremos y evaluar la función en ellos:

```
A = rbind(c(1,-1), # R1
          c(1,2),  # R2
          c(1,0),  # x1 = 0
          c(0,1))  # x2 = 0

# Vector del lado derecho
b = c(4,4,0,0)

# En este caso, la región crítica tiene un solo vértice
punto = solve(A[c(1,2),], b[1:2])

puntos = rbind(punto)
puntos = data.frame(puntos)

f_obj = function(x) - x$X1 + 3*x$X2
puntos[which.min(f_obj(puntos)),]
```

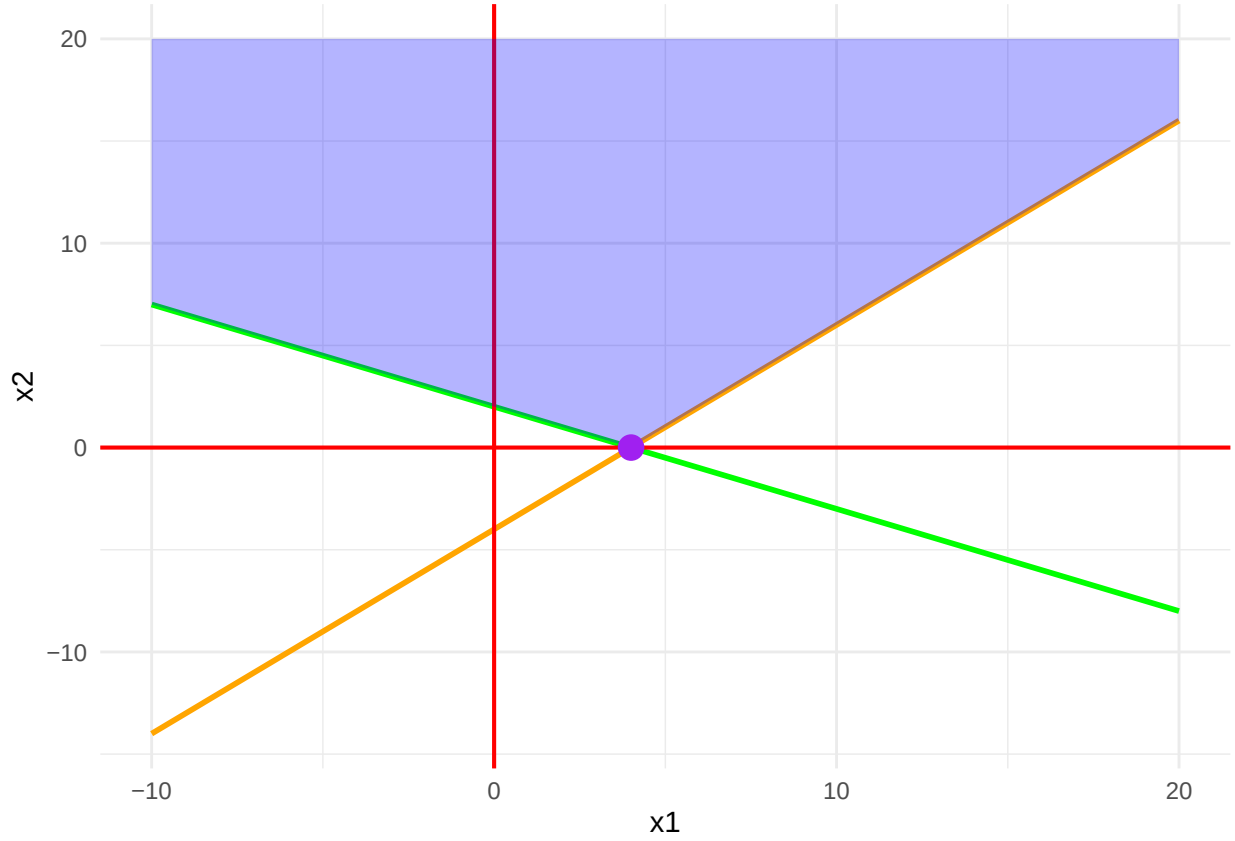
```
##       X1 X2
## punto  4  0
```

```
f_obj(puntos)
```

```
## [1] -4
```

Grafiquemos el resultado:

```
p + geom_point(data=puntos, aes(x=X1, y=X2), col="purple", size=4)
```



La región factible es no acotada, y tiene un único punto extremo.

Ejercicio 2.1.3(b)

Maximizar $4x_1 + x_2$ sujeto a:

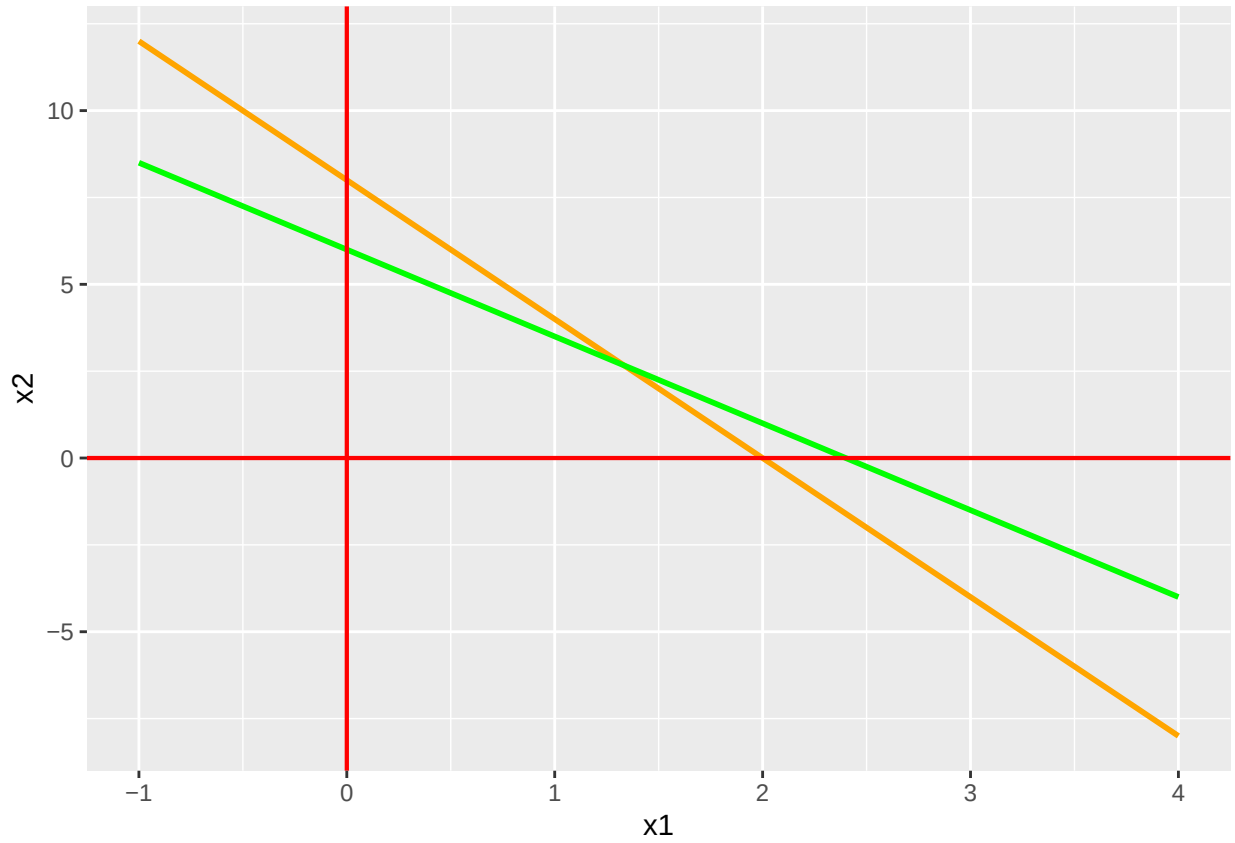
$$\begin{cases} 8x_1 + 2x_2 \leq 16 \\ 5x_1 + 2x_2 \leq 12 \\ x_1, x_2 \geq 0 \end{cases}$$

Al igual que en el caso anterior, comenzamos definiendo las instrucciones:

```
R1 = function(x1) 8 - 4*x1
R2 = function(x1) 6 - 5*x1/2
x1 = seq(-1, 4, length.out=100)

datos = data.frame(x1=x1,
                   x2=R1(x1),
                   x2.2=R2(x1))
```

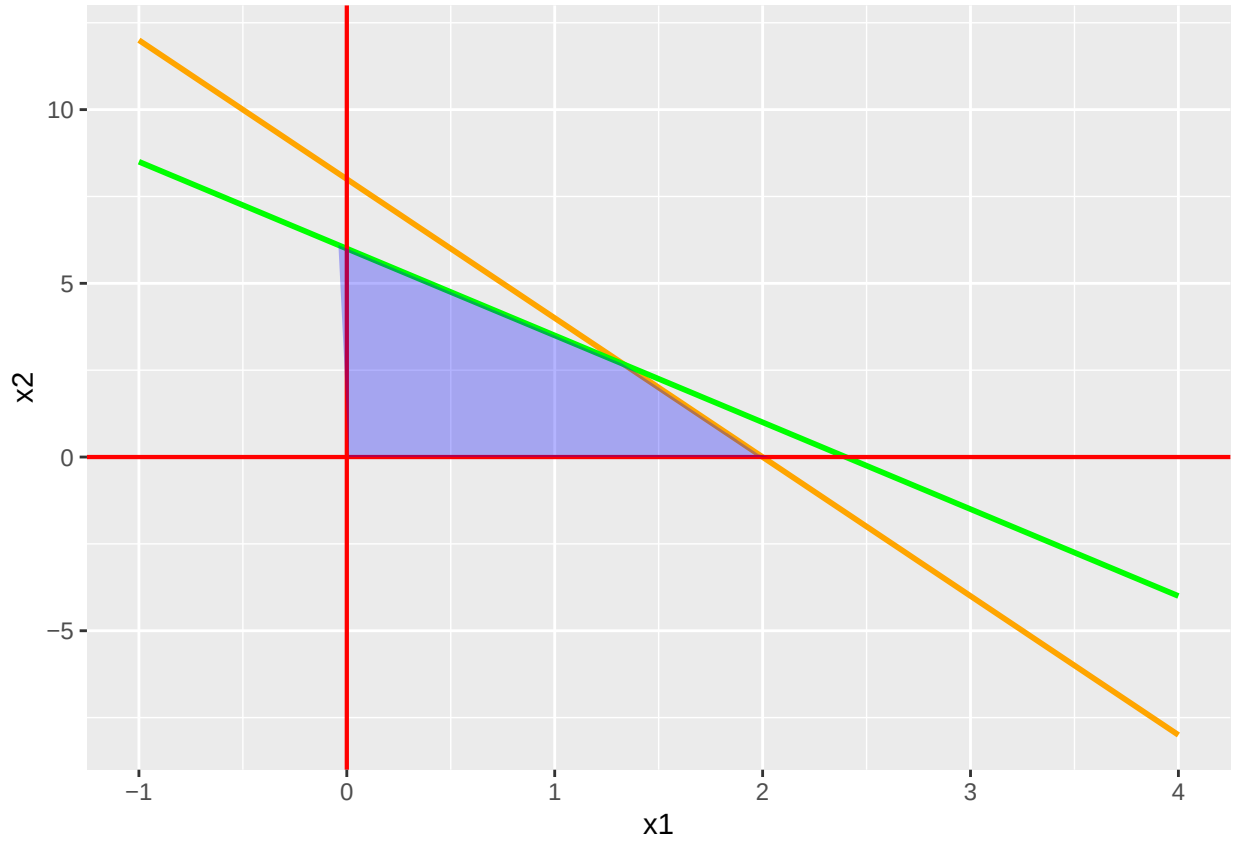
Grafiquemos las restricciones:

```
p = ggplot(data=datos,aes(x=x1)) +
  geom_line(aes(y=x2), col="orange", size=1) +
  geom_line(aes(y=x2.2), col="green", size=1) +
  geom_hline(yintercept=0, col="red", size=0.75) +
  geom_vline(xintercept=0, col="red", size=0.75)

p
```



Pasando ahora a la región factible:

```
p = p + geom_ribbon(data=transform(datos,
                                   yMin=ifelse(datos$x1>=0,0,datos$x2.2),
                                   yMax=ifelse(pmin(datos$x2, datos$x2.2) > 0, pmin(datos$x2, datos$x2.2), 0)),
                    aes(ymin = yMin, ymax=yMax),
                    fill="blue", alpha=0.3)

p
```



Veamos ahora qué puntos extremos tiene el conjunto y cuál es su máximo:

```
A = rbind(c(8,2), # R1
          c(5,2), # R2
          c(1,0), # x1 >= 0
          c(0,1)) # x2 >= 0

b = c(16,12,0,0)

pto.R1R2 = solve(A[c(1,2),], b[c(1,2)])
pto.R10 = solve(A[c(1,4),], b[c(1,4)])
pto.R20 = solve(A[c(2,3),], b[c(2,3)])
pto.00 = solve(A[c(3,4),], b[c(3,4)])

puntos = rbind(pto.R1R2, pto.R10, pto.R20, pto.00)
puntos = as.data.frame(puntos)
puntos
```

```
##                     V1       V2
```

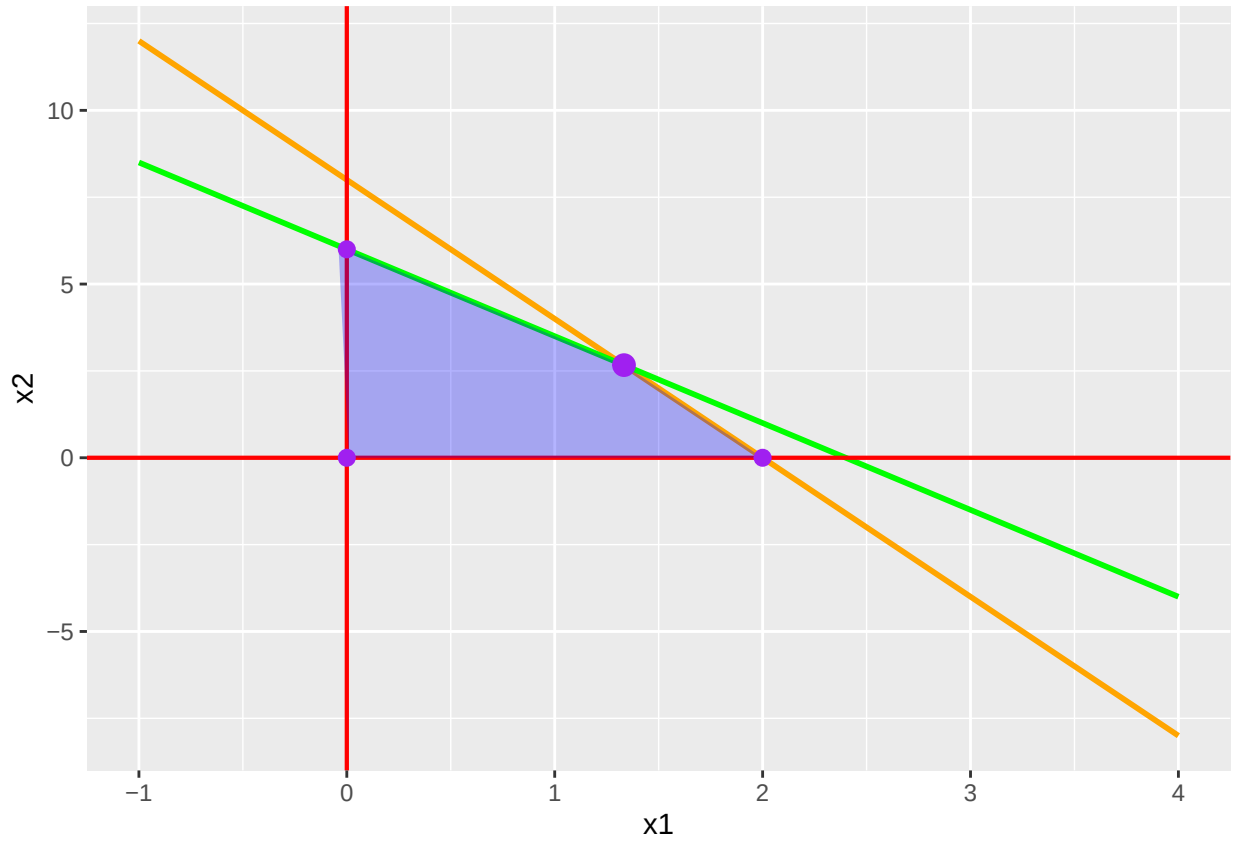
```
## pto.R1R2  1.333333e+00 2.666667
## pto.R10   2.000000e+00 0.000000
## pto.R20  -3.552714e-16 6.000000
## pto.00    0.000000e+00 0.000000
```

```
f_obj = function(x) 4*x$V1 + x$V2
maximo = puntos[which.max(f_obj(puntos)),]
maximo
```

```
##                 V1       V2
## pto.R1R2 1.333333 2.666667
```

Grafiquemos el resultado:

```
p + geom_point(data=puntos, aes(x=V1, y=V2), col="purple", size=2.5) +
  geom_point(data=maximo, aes(x=V1, y=V2), col="purple", size=3.5)
```



Paquete `lpSolve`

Ejercicios 2.4.3

1. Primer ejercicio Maximizar $5x_1 + 7x_2$ sujeto a:

$$\begin{cases} 8x_1 + 14x_2 \leq 63 \\ 10x_1 + 4x_2 \leq 45 \\ x_1, x_2 \in \mathbb{Z}^+ \end{cases}$$

```
coef = c(5,7)
A = rbind(c(8,14), c(10,4))
b = c(63,45)
sol = lp(direction = "max", objective.in = coef, const.mat = A,
         const.rhs = b, const.dir = c("<=", "<="), all.int=TRUE)
sol
```

```
## Success: the objective function is 31
```

```
sol$solution
```

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

Es necesario usar el parámetro `all.int` para garantizar que $x_1, x_2 \in \mathbb{Z}^+$.

2. Segundo ejercicio Maximizar $ x_1 -2x_2 -3x_3 -x_4$ sujeto a:

$$\begin{cases} x_1 - x_2 - 2x_3 - x_4 & \leq 4 \\ 2x_1 + x_3 - 4x_4 & \leq 2 \\ -2x_1 + x_2 + x4 & \leq 1 \\ x_i \geq 0 & i = 1, 2, 3, 4 \end{cases}$$

Modelamos las restricciones:

```
coef = c(1,-2,-3,-1)
A = rbind(c(1, -1,-2,-1),
          c(2,0,1,-4),
          c(-2,1,0,1))
b = c(4,2,1)

sol = lp("max", coef, A, rep("<=", 3), b)
sol
```

```
## Success: the objective function is 4
```

```
sol$solution
```

```
## [1] 7 0 0 3
```

3. Tercer ejercicio Minimizar $-2x_2 + x_3$ sujeto a:

$$\begin{cases} -x_1 - 2x_2 & \geq -3 \\ 4x_1 + x_2 + 7x_3 & \geq -1 \\ 2x_1 - 3x_2 + x_3 & \geq -5 \\ x_i \geq 0 & i = 1, 2, 3 \end{cases}$$

Al igual que en los casos anteriores,

```
coef = c(0,-2,1)
A = rbind(c(-1,-2,0),
          c(4,1,7),
          c(2,-3,1))
b = c(-3,-1,-5)

sol = lp("min", coef, A, rep(">=", 3), b)
sol
```

```
## Success: the objective function is -3
```

```
sol$solution
```

```
## [1] 0.0 1.5 0.0
```

4. Cuarto ejercicio Una compañía está evaluando 5 proyectos a desarrollar durante los próximos 3 años. Cada año dispone de 25 millones de euros para invertir en proyectos. El beneficio esperado (en millones) para cada proyecto, así como la cantidad a invertir cada año (en millones) para el mantenimiento del proyecto vienen dadas por la siguiente tabla:

	inversión / año			
proyecto	1	2	3	beneficio
1	5	1	8	20
2	4	7	10	40
3	3	9	2	20
4	7	4	1	15
5	8	6	10	30

¿Qué proyectos se deberían desarrollar para obtener un beneficio mayor?

Resolución: Podemos modelar el problema usando programación booleana. Para cada i-ésimo proyecto ($i \in \{1,2,3,4,5\}$), la variable booleana x_i será 1 si la empresa decide apostar por él, ó 0 en caso de no hacerlo.

Por tanto, el problema se reduce a maximizar $20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5$ sujeto a:

$$\begin{cases} 5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 & \leq 25 \\ x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 & \leq 25 \\ 8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 & \leq 25 \\ x_i \in \{0,1\} & i \in \{1,\ldots,5\} \end{cases}$$

Usando ahora R para resolver el problema:

```
coef = c(20,40,20,15,30)
A = rbind(c(5,4,3,7,8),
          c(1,7,9,4,6),
          c(8,10,2,1,10))
b = rep(25,3)

sol = lp("max", coef, A, rep("<=", 3), b, all.bin = TRUE)
sol
```

```
## Success: the objective function is 95
```

```r
sol$solution
```

```
## [1] 1 1 1 1 0
```

La empresa puede invertir en los proyectos 1,2,3 y 4; logrando un beneficio esperado de 95 millones.

5. Quinto ejercicio Una empresa esta estudiando llevar a cabo una campaña publicitaria, para ello dispone de 1.000.000 de euros. Puede difundir sus anuncios en dos canales publicitarios distintos, el primero de ellos cobra 15.000 euros cada vez que emite un anuncio, mientras que el segundo cobra el doble. La probabilidad de que un anuncio del primer canal sea visto es del 30 %, mientras que del segundo es del 70 %. Como mínimo deben emitirse 26 anuncios en el primer canal y 13 en el segundo. Determinar el numero de anuncios que debe lanzar en cada canal de manera que maximice la probabilidad de que se vea el anuncio de la empresa, teniendo en cuenta la restriccion presupuestaria y las del número de anuncios.

Resolución Las variables de decisión serán el número de anuncios que se emiten en cada canal. Por tanto, este es un problema de programación entera. Las restricciones vienen dadas por el presupuesto y el mínimo de anuncios en cada cadena. Queda maximizar la probabilidad de que vea un cliente un anuncio.

Si se emiten x_1 anuncios en el primer canal, la probabilidad de que no se vea ninguno de ellos es de $0,7^{x_1}$. Análogamente, si se emiten x_2 anuncios en el segundo medio, la probabilidad de no ver ninguno es de $0,3^{x_2}$. Si las emisiones en ambos canales son independientes, la probabilidad de que se vea al menos un anuncio es de $1 - 0,7^{x_1} \cdot 0,3^{x^2}$.

Este no es, a priori, un problema de optimización lineal, pero podemos transformarlo para que sí lo sea: Maximizar la función anterior equivale a minimizar $0,7^{x_1} \cdot 0,3^{x^2}$. Al ser log una función estrictamente, esto equivale a su vez a minimizar $\log(0.7)x_1 + \log(0.3)x_2$. Este ya es un problema de optimización lineal. Podemos obtener el punto óptimo con `lp` directamente, pero el valor de la función hemos de calcularlo "a mano''.

```r
coef = c(log(0.7), log(0.3))
A = rbind(c(15e3, 30e3), # presupuesto
          c(1,0),
          c(0,1)) # mínimo canal 1

b = c(1e6,26,13)
dir=c("<=", ">=", ">=")
sol = lp("min", coef, A, dir, b, all.int=TRUE)

sol # Podemos observar que no es una probabilidad
```

```
## Success: the objective function is -33.353
```

```r
anuncios = sol$solution
anuncios # Este sí es el punto de interés
```

```
## [1] 26 20
```

Veamos cuál es la probabilidad obtenida real:

```r
probabilidadVerAnuncios = function(x) 1 - 0.7^x[1] * 0.3^x[2]
probabilidadVerAnuncios(anuncios)
```

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

Es casi certero que se vean los anuncios de la empresa. La probabilidad de que no lo vean (calulada así para que R imprima con más decimales) es de:

```r
exp(anuncios[1]*log(0.7) + anuncios[2]*log(0.3))
```

```
## [1] 3.273212e-15
```