Practica 2

Eloy Bedia & Miguel Moles

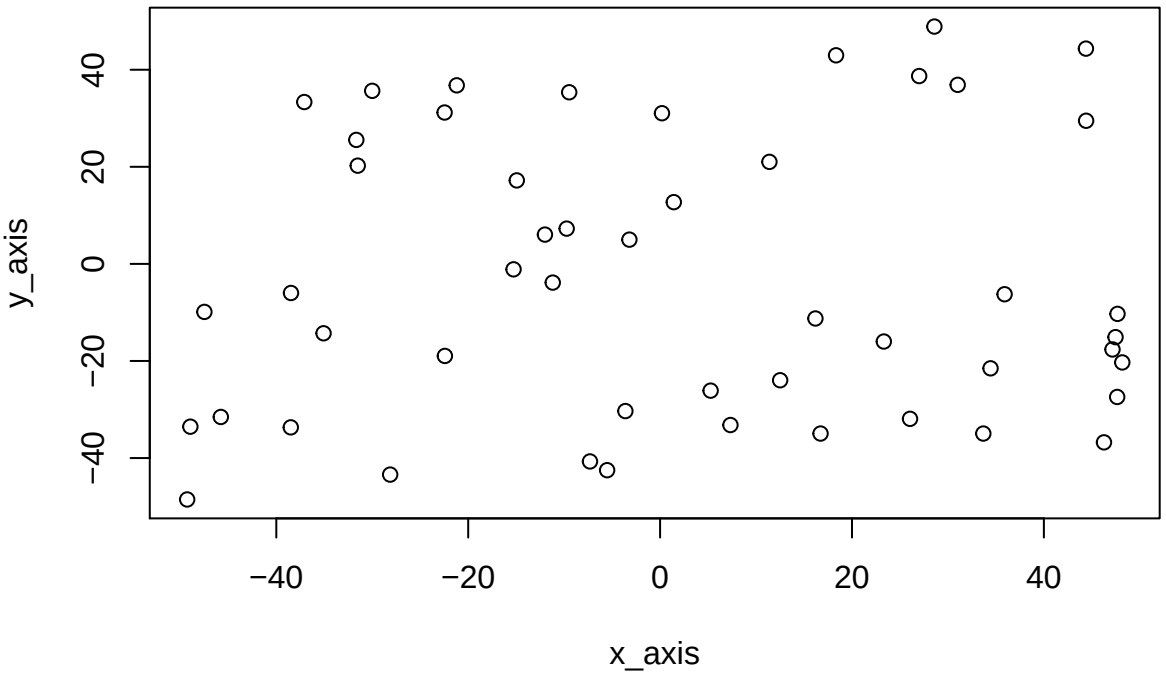
6 de abril de 2018

APARTADO 1

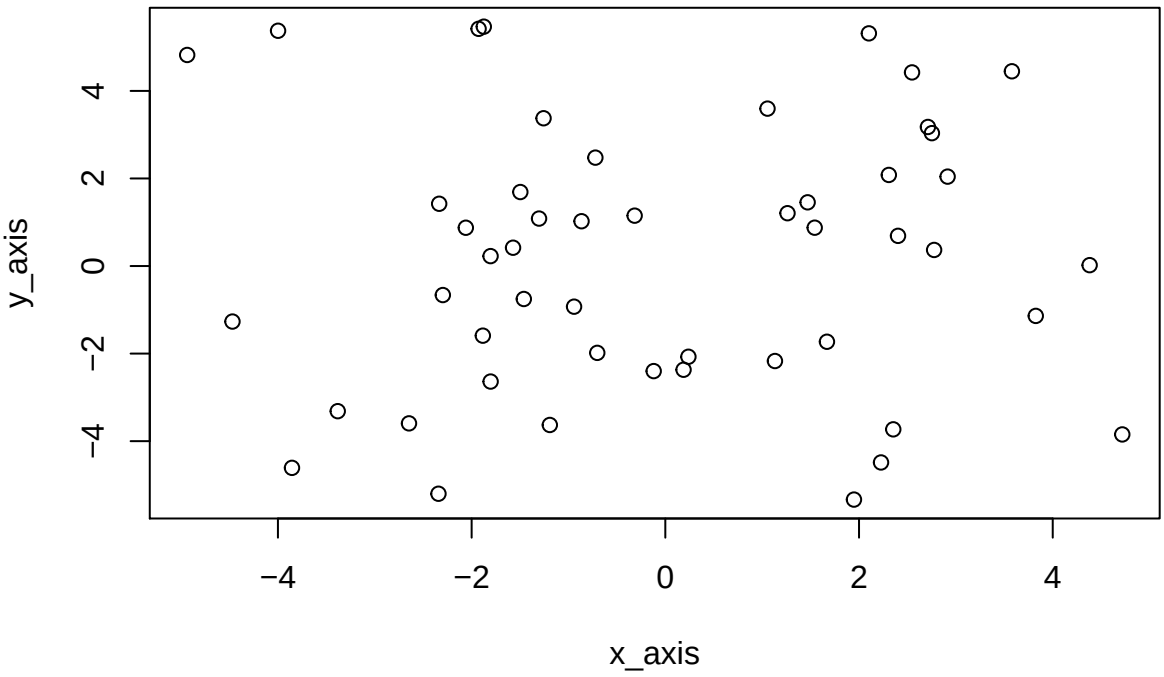
EJERCICIO 1

1. Dibujar una gráfica con la nube de puntos de salida correspondiente.

a) Considere $N = 50$, $dim = 2$, $rango = [-50, +50]$ con $simula\_unif(N, dim, rango)$.



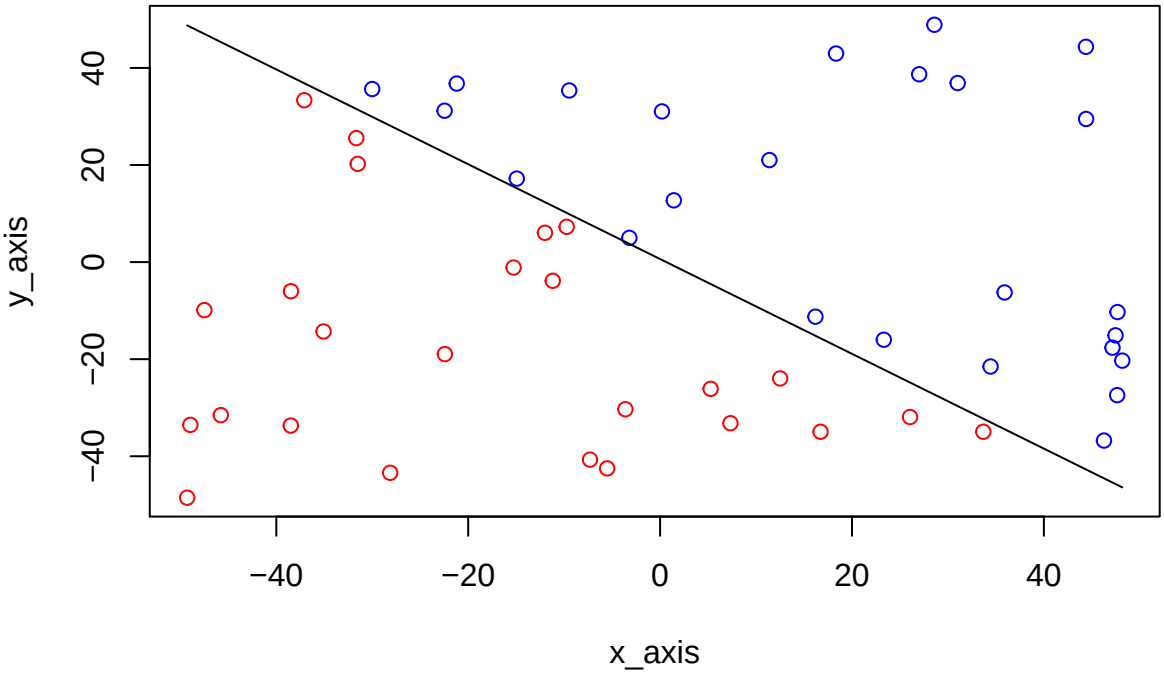
b) Considere $N = 50$, $dim = 2$ y $sigma = [5, 7]$ con $simula\_gaus(N, dim, sigma)$.


y_axis
4
2
0
−2
−4
−4
−2
0
2
4
x_axis

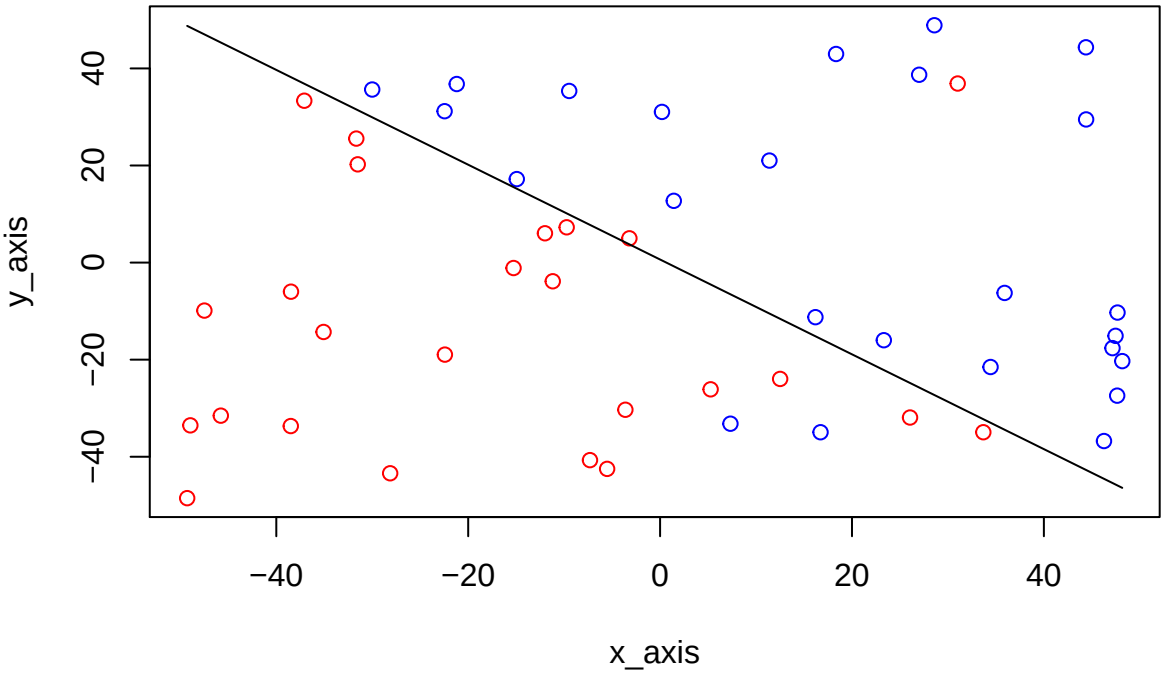
EJERCICIO 2

Con ayuda de la función *simula_unif*() generar una muestra de puntos 2D a los que vamos añadir una etiqueta usando el signo de la función $f(x, y) = y - ax - b$, es decir el signo de la distancia de cada punto a la recta simulada con *simula_recta*().

a) Dibujar una gráfica donde los puntos muestren el resultado de su etiqueta, junto con la recta usada para ello. (Observe que todos los puntos están bien clasificados respecto de la recta)



b) Modifique de forma aleatoria un 10% etiquetas positivas y otro 10% de negativas y guarde los puntos con sus nuevas etiquetas. Dibuje de nuevo la gráfica anterior. (Ahora hay puntos mal clasificados respecto de la recta)



EJERCICIO 3

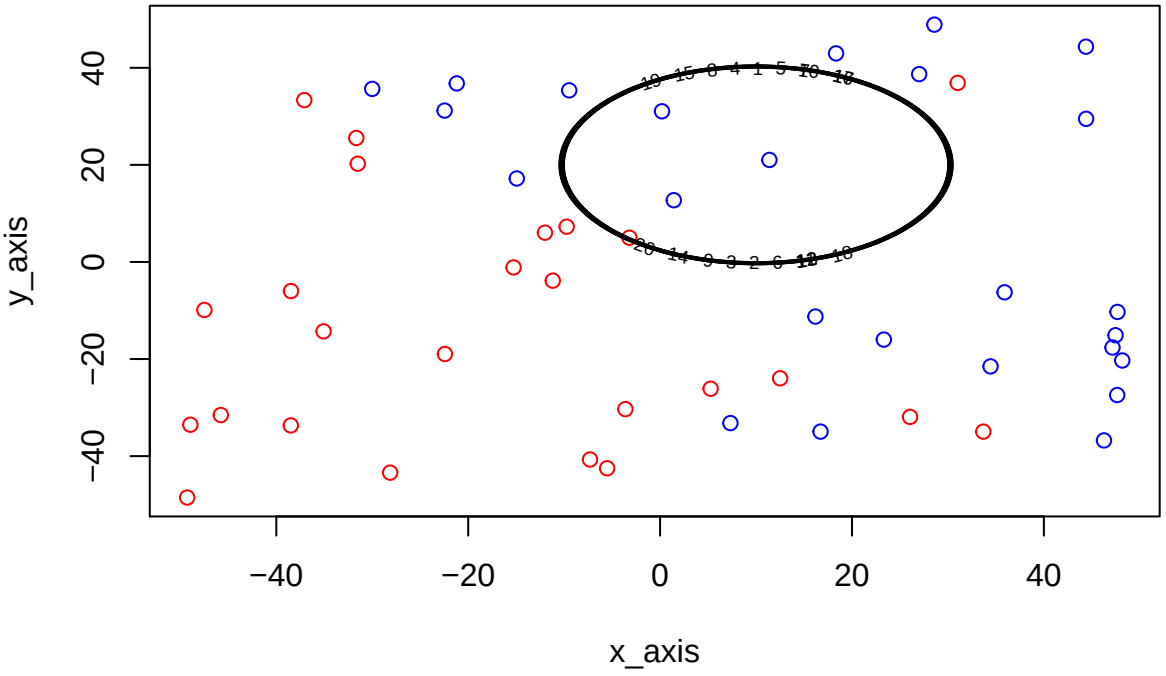
3. Supongamos ahora que las siguientes funciones definen la frontera de clasificación de los puntos de la muestra en lugar de una recta $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$

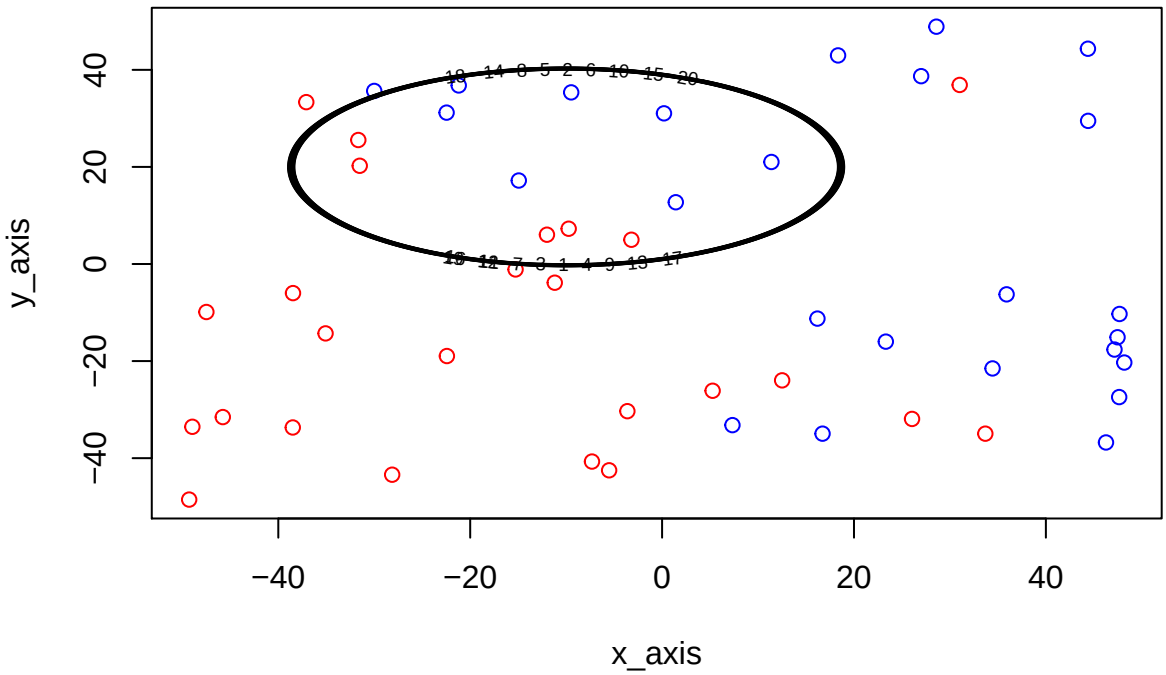
$f(x, y) = 0,5(x + 10)^2 + (y - 20)^2 - 400$

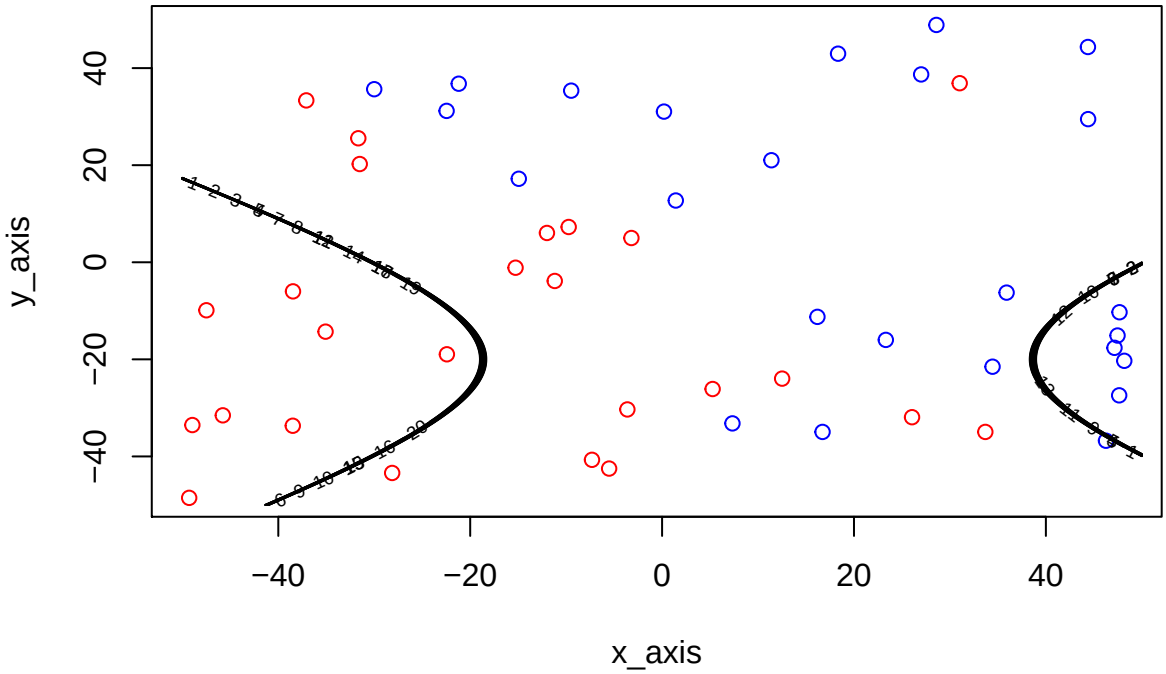
$f(x, y) = 0,5(x - 10)^2 - (y + 20)^2 - 400$

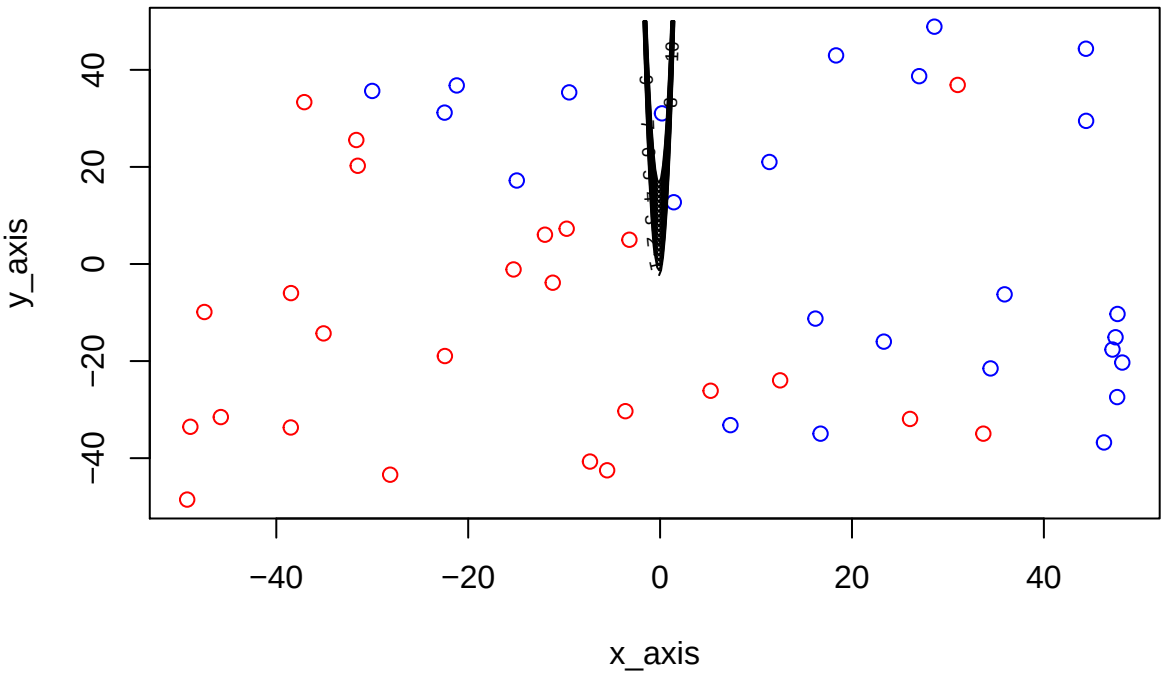
$f(x, y) = y - 20x^2 - 5x + 3$

Visualizar el etiquetado generado en 2b junto con cada una de las gráficas de cada una de las funciones. Comparar las formas de las regiones positivas y negativas de estas nuevas funciones con las obtenidas en el caso de la recta ¿Son estas funciones más complejas mejores clasificadores que la función lineal? ¿En que ganan a la función lineal? Explicar el razonamiento.




y_axis
x_axis
−40
−20
0
20
40


y_axis
x_axis
−40
−20
0
20
40



En todos los casos anteriores, debido a la excesiva complejidad de la clase de funciones elegida ($ax^2 + by^2 + cx + dy + e$), esta no es capaz de adaptarse correctamente a la forma de una recta, que es la funcion que se ha utilizado originalmente para asignarle las etiquetas a esta muestra. A causa de esto, el error dentro de la muestra es muy grande y en consecuencia también lo será fuera de la muestra

APARTADO 2

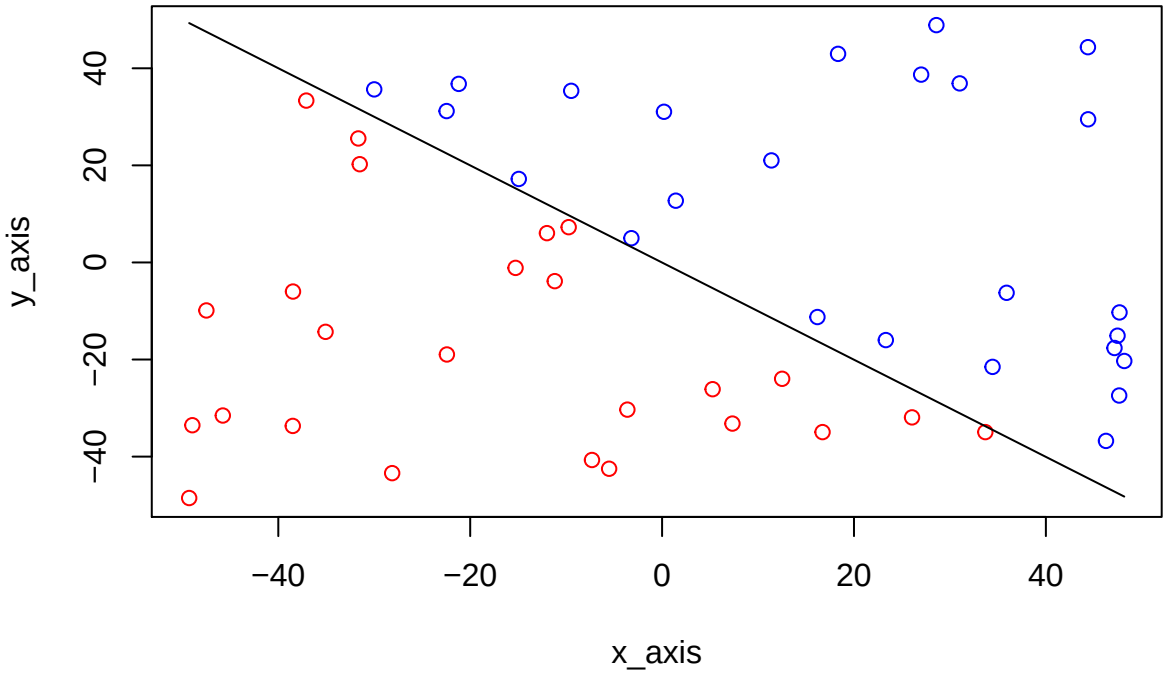
EJERCICIO 1

1. Algoritmo Perceptron: Implementar la función $ajusta\_PLA(datos, label, max\_iter, vini)$ que calcula el hiperplano solución a un problema de clasificación binaria usando el algoritmo PLA. La entrada datos es una matriz donde cada item con su etiqueta está representado por una fila de la matriz, label el vector de etiquetas (cada etiqueta es un valor +1 o -1), max_iter es el número máximo de iteraciones permitidas y $vini$ el valor inicial del vector. La función devuelve los coeficientes del hiperplano.

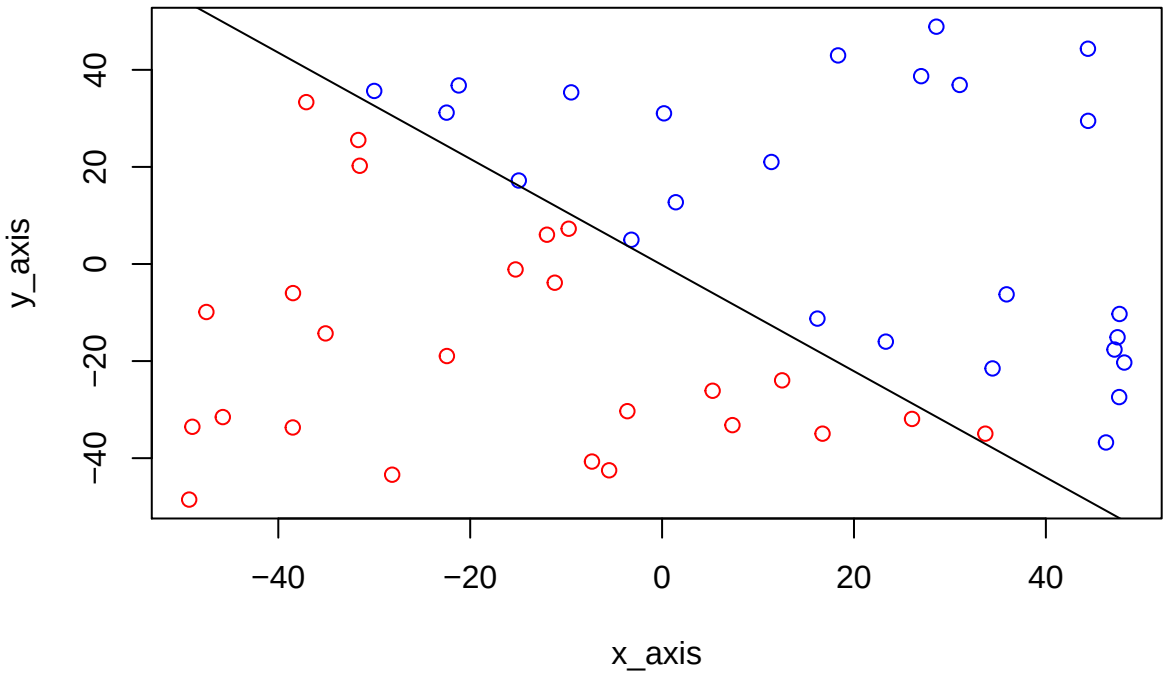
```
ajusta_PLA <- function(datos, label, max_iter, vini){
  label[label==-1] <- 0
  cambio <- TRUE
  datos <- cbind(1, datos)
  iter <- 1

  while(cambio && iter <= max_iter){
    w <- vini
    for(i in 1:length(label)){
      if(signo(t(w)%*%datos[i,]) != label[i]){
        w <- w + label[i]*datos[i,]
      }
    }
    s <- abs(sum(w - vini))
    if(s <= 10e-10)
      cambio = FALSE
    vini <- w
```

```
    iter <- iter + 1
  }
  sol <-pasoARecta(vini)
  list(sol, iter)
}
```

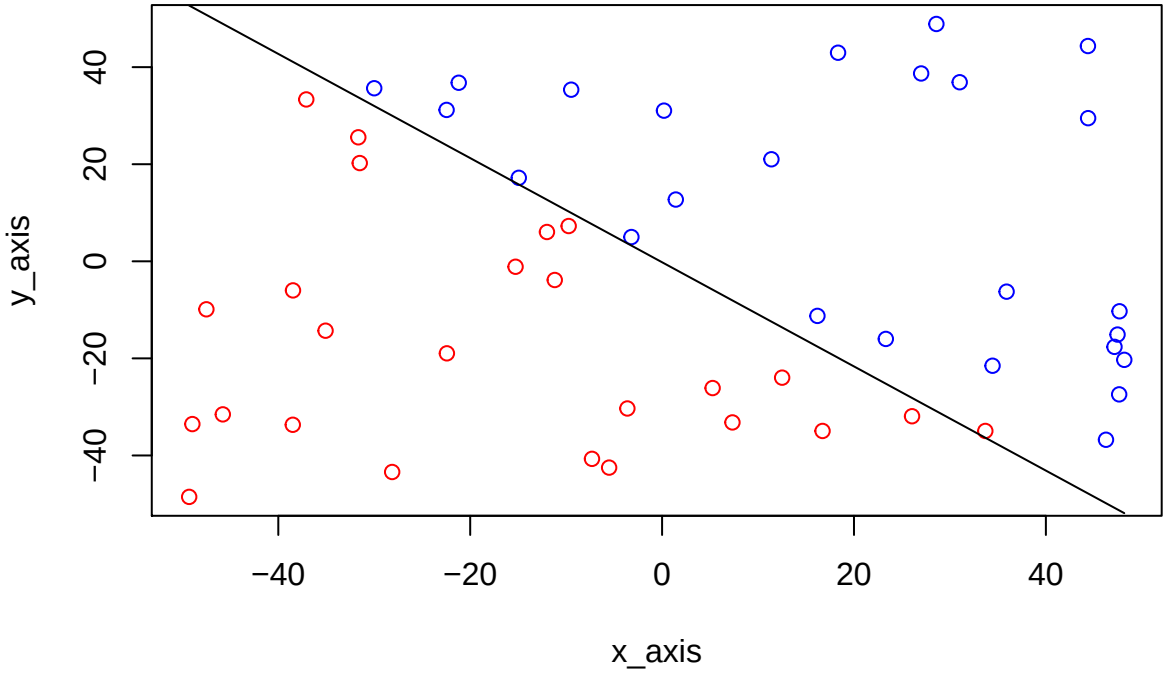
a) Ejecutar el algoritmo PLA con los datos simulados en los apartados $2a$ de la sección.1. Inicializar el algoritmo con: a) el vector cero y, b) con vectores de números aleatorios en $[0, 1]$ (10 veces). Anotar el número medio de iteraciones necesarias en ambos para converger. Valorar el resultado relacionando el punto de inicio con el número de iteraciones.



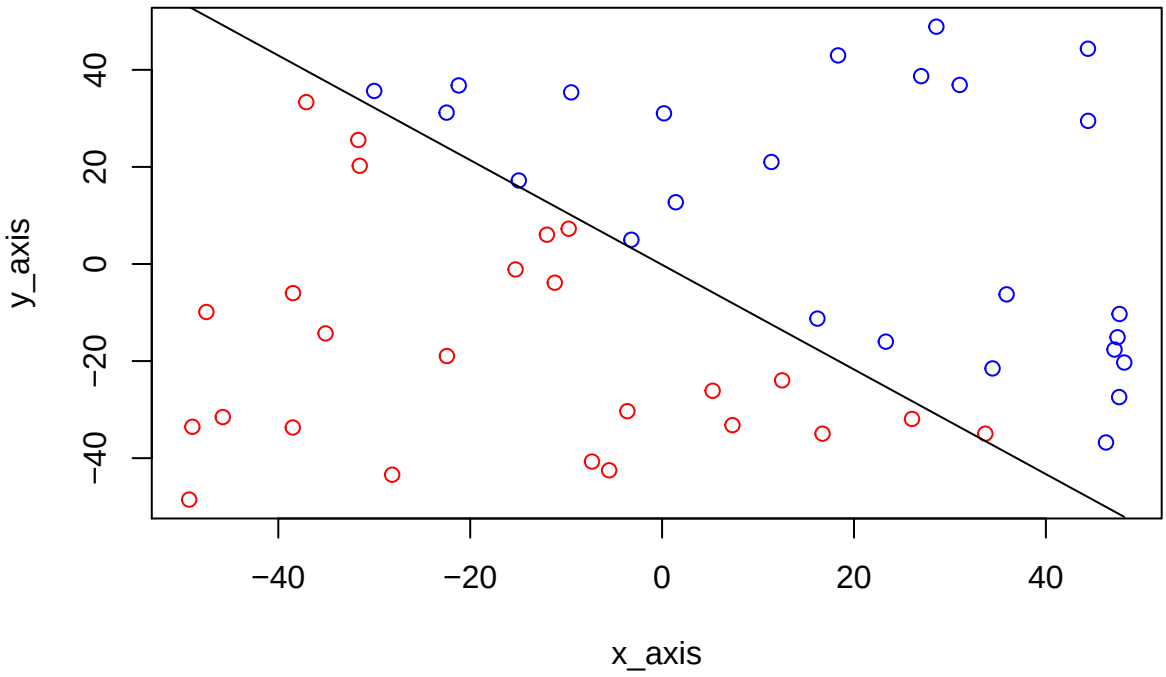

```
## Iteraciones:  3 Value: ( 0 0 0 )
```



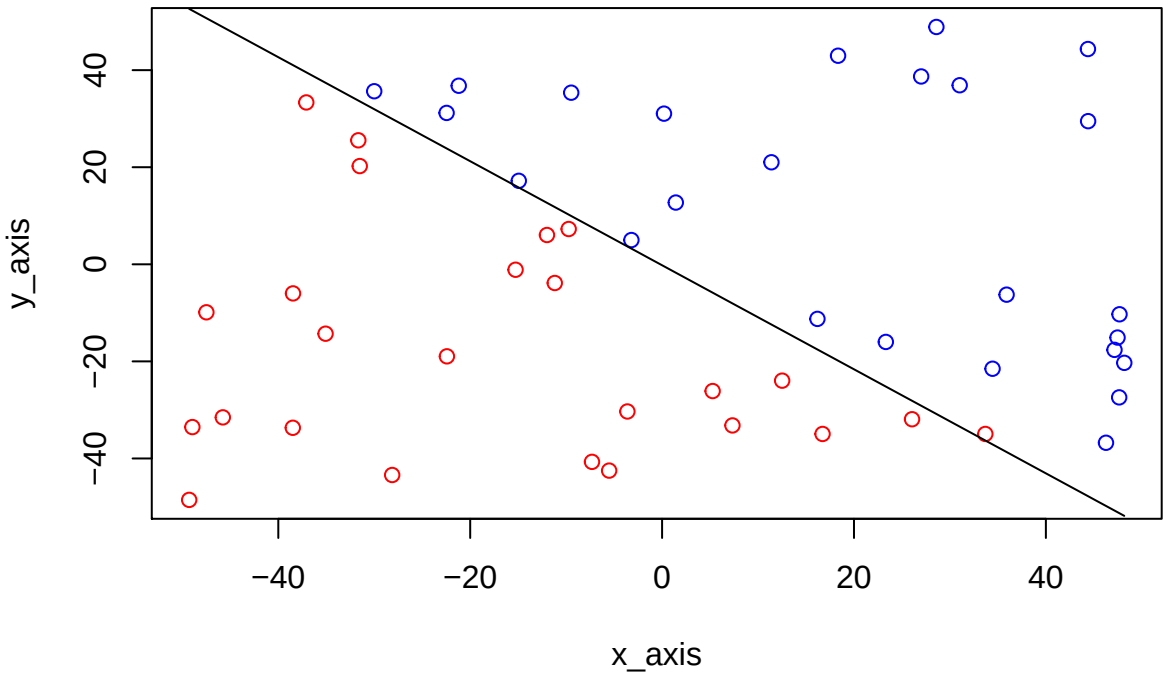

```
## Iteraciones:  4 Inicio: ( 0.8992507 0.04573869 0.08961809 )
```



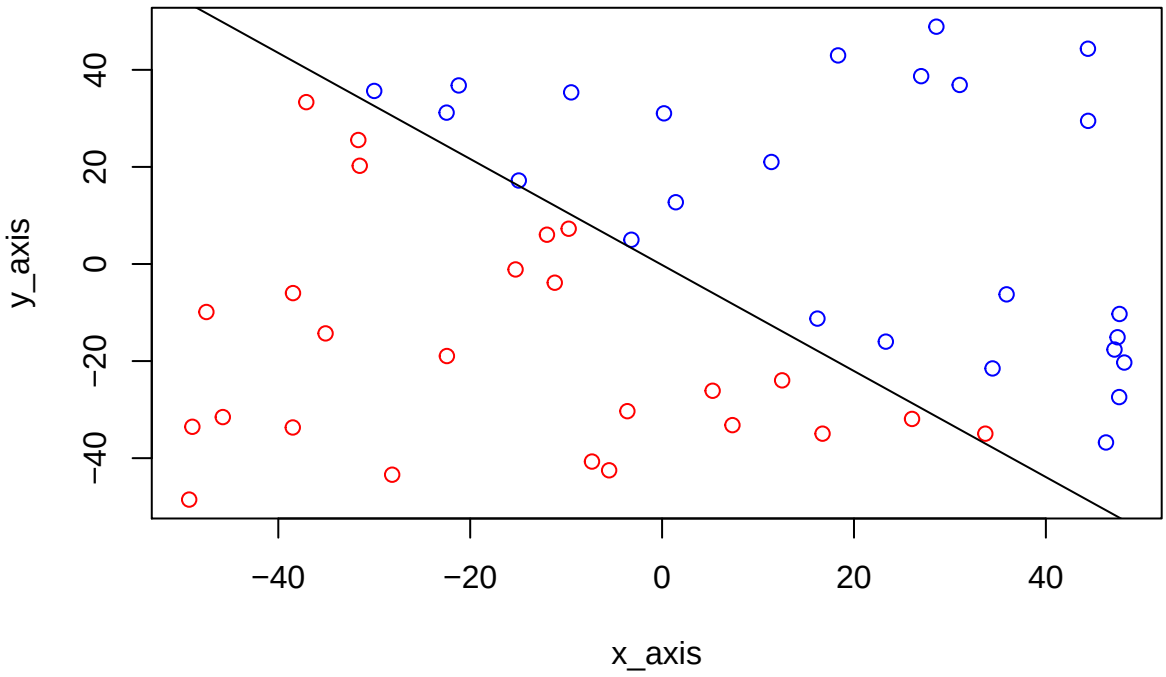

```
## Iteraciones:  5 Inicio: ( 0.4408768 0.7779656 0.1400851 )
```



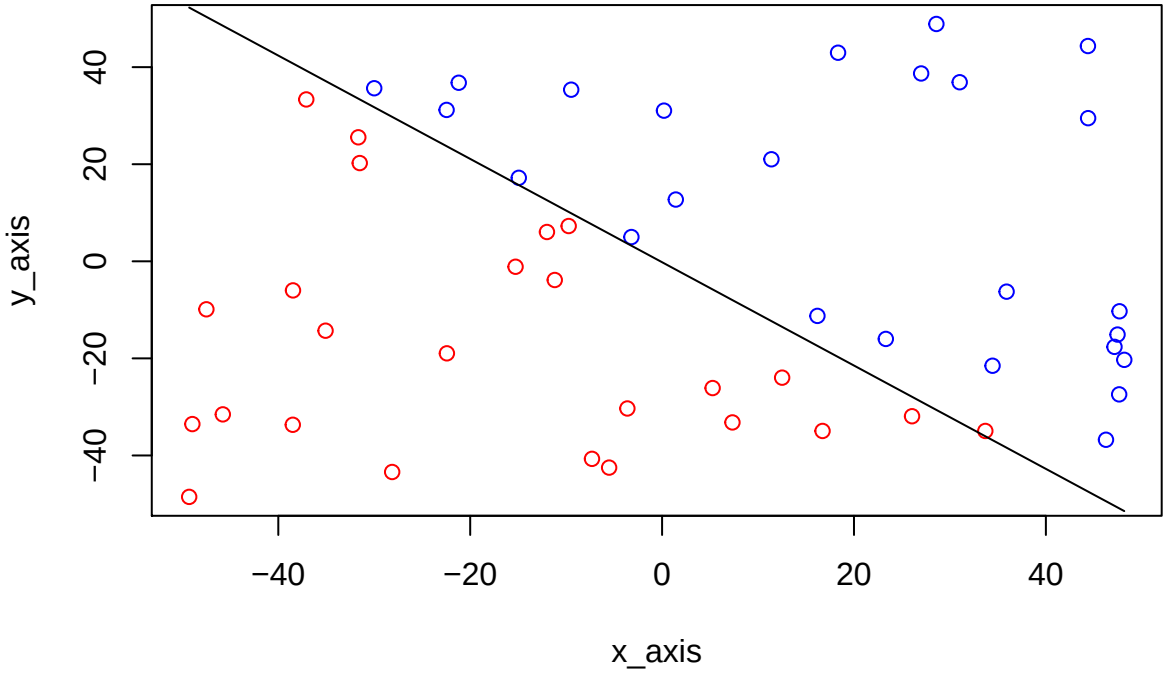

```
## Iteraciones:  4 Inicio: ( 0.6787487 0.04759801 0.6056478 )
```



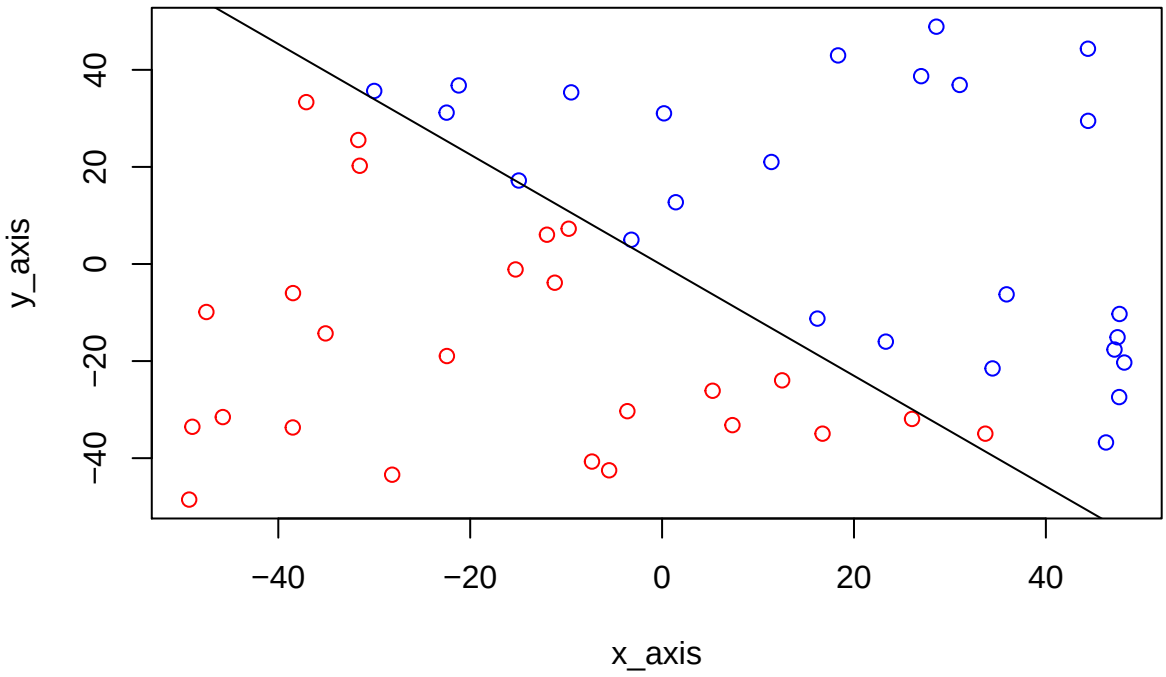

```
## Iteraciones:  5 Inicio: ( 0.4501444 0.7477808 0.1771495 )
```



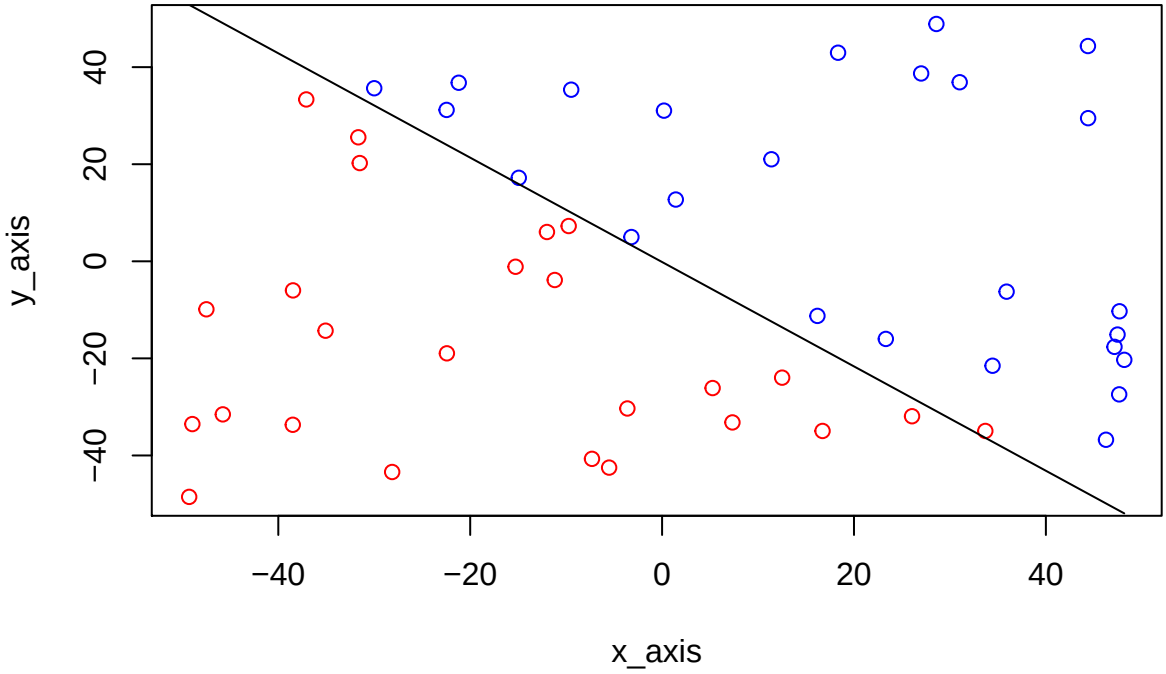

```
## Iteraciones:  4 Inicio: ( 0.6548571 0.1641085 0.2617137 )
```



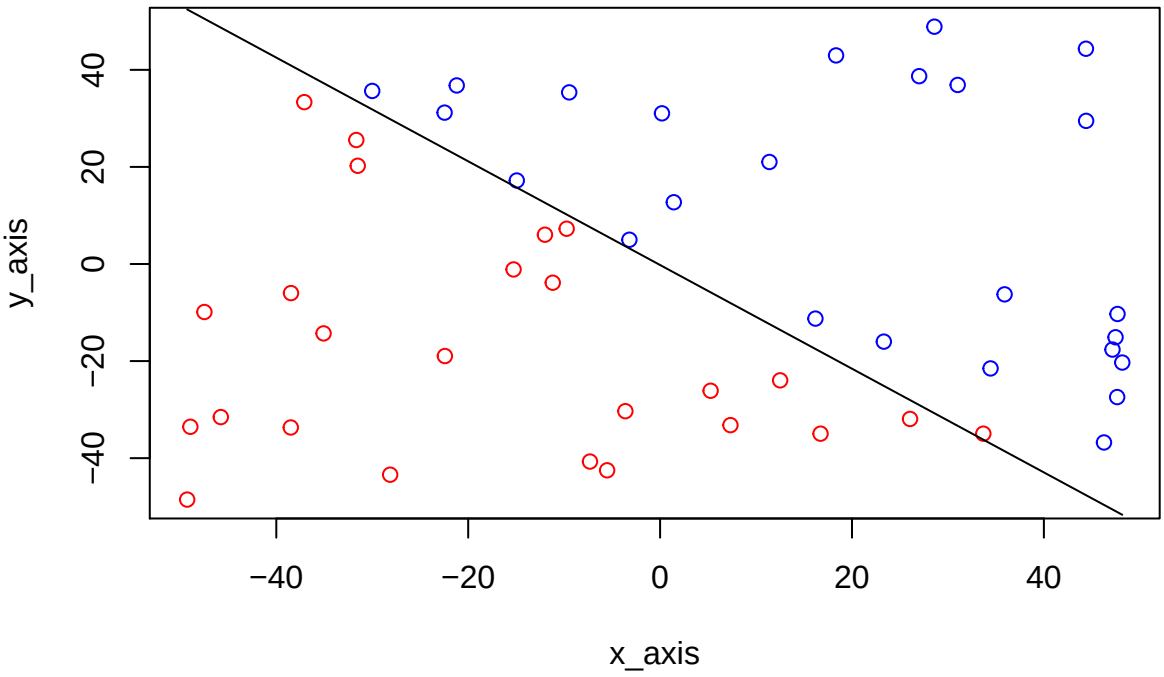

```
## Iteraciones:  5 Inicio: ( 0.09503739 0.3944602 0.2132182 )
```



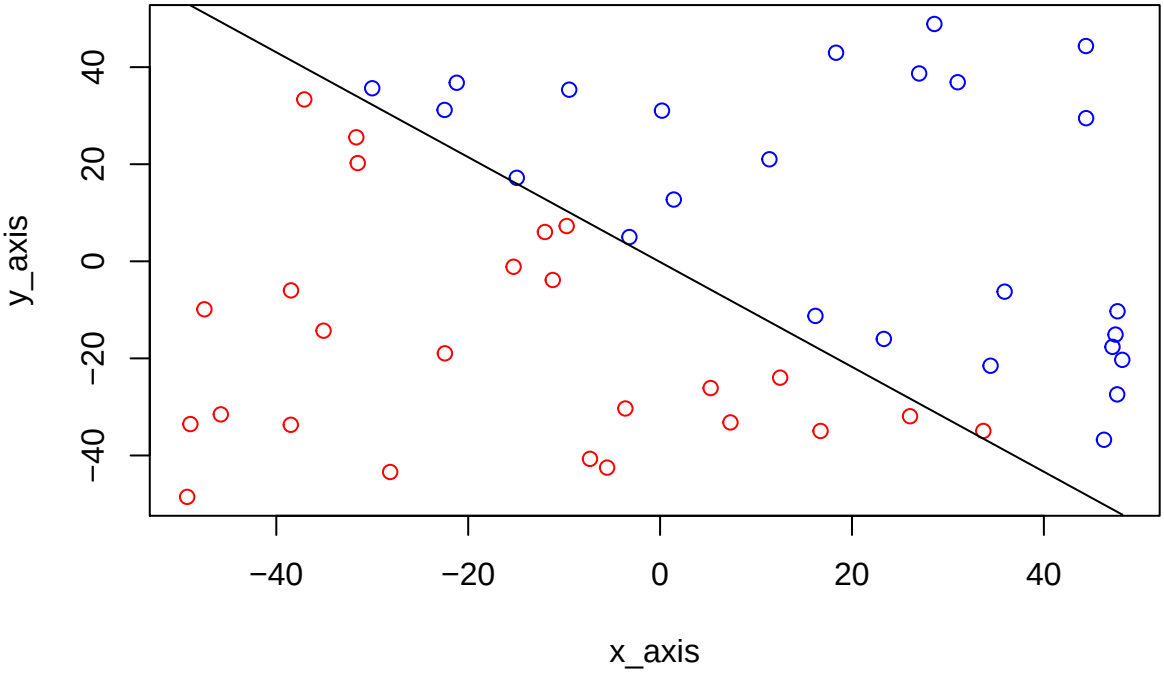

```
## Iteraciones:  9 Inicio: ( 0.6405314 0.2576197 0.09461404 )
```




```
## Iteraciones:  4 Inicio: ( 0.07039834 0.1883563 0.8731385 )
```

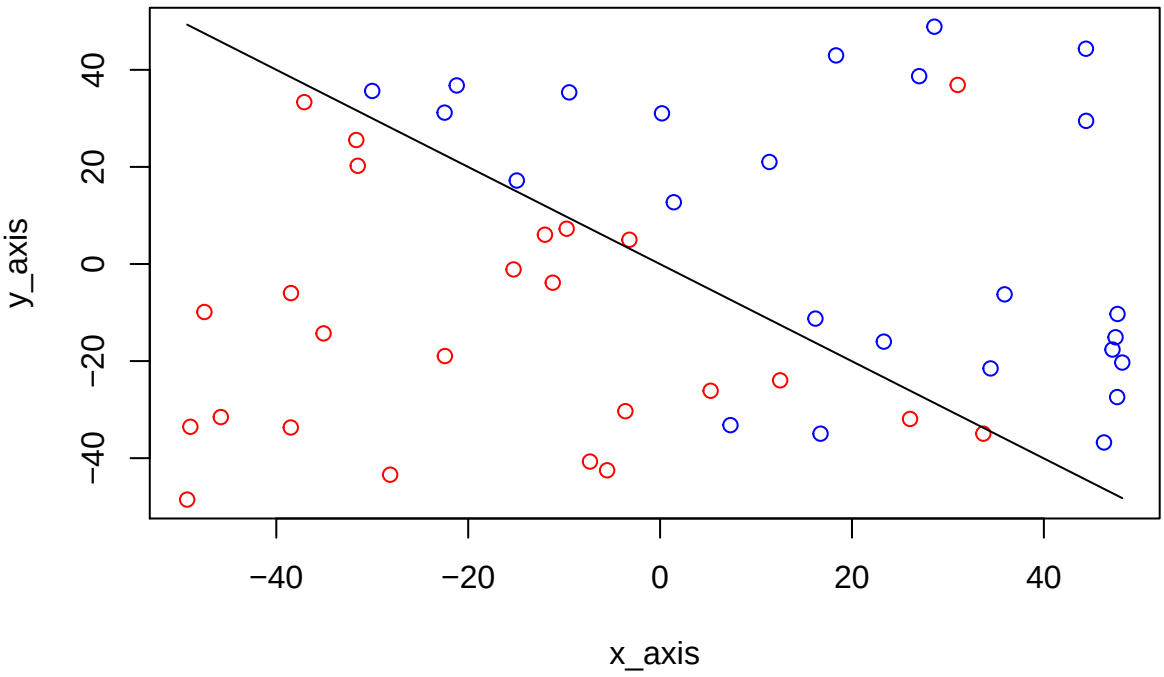



```
## Iteraciones:  5 Inicio: ( 0.9811036 0.3909197 0.01789851 )
```

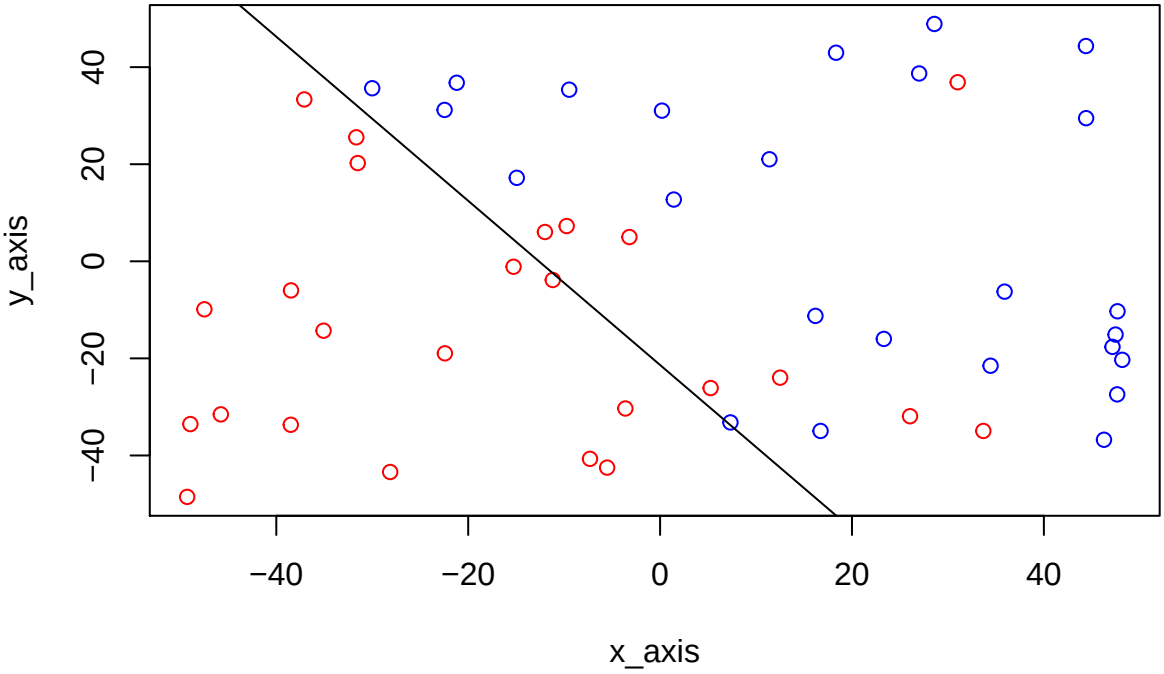



```
## Iteraciones:  4 Inicio: ( 0.0445446 0.1077643 0.5703009 )
## Nº Iteraciones Medio:  4.727273
## El numero medio de iteraciones es pequeño debido a que estamos trabajando con
## un conjunto separable
```

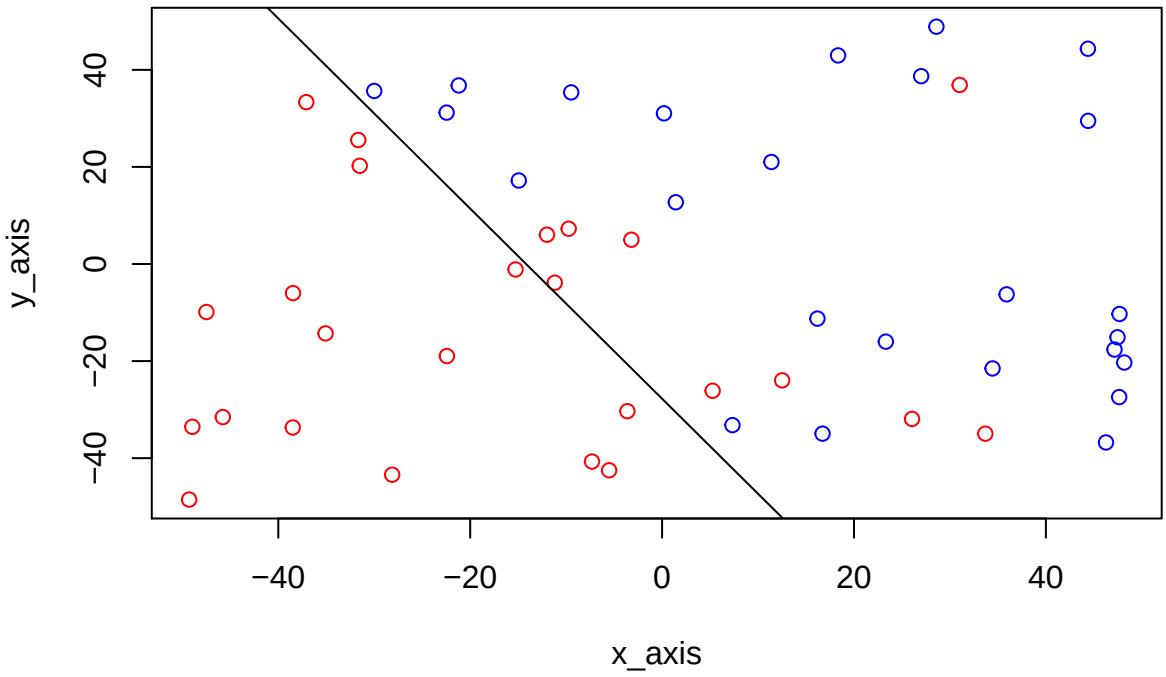
b) Hacer lo mismo que antes usando ahora los datos del apartado 2b de la sección.1. ¿Observa algún comportamiento diferente? En caso afirmativo diga cual y las razones para que ello ocurra.



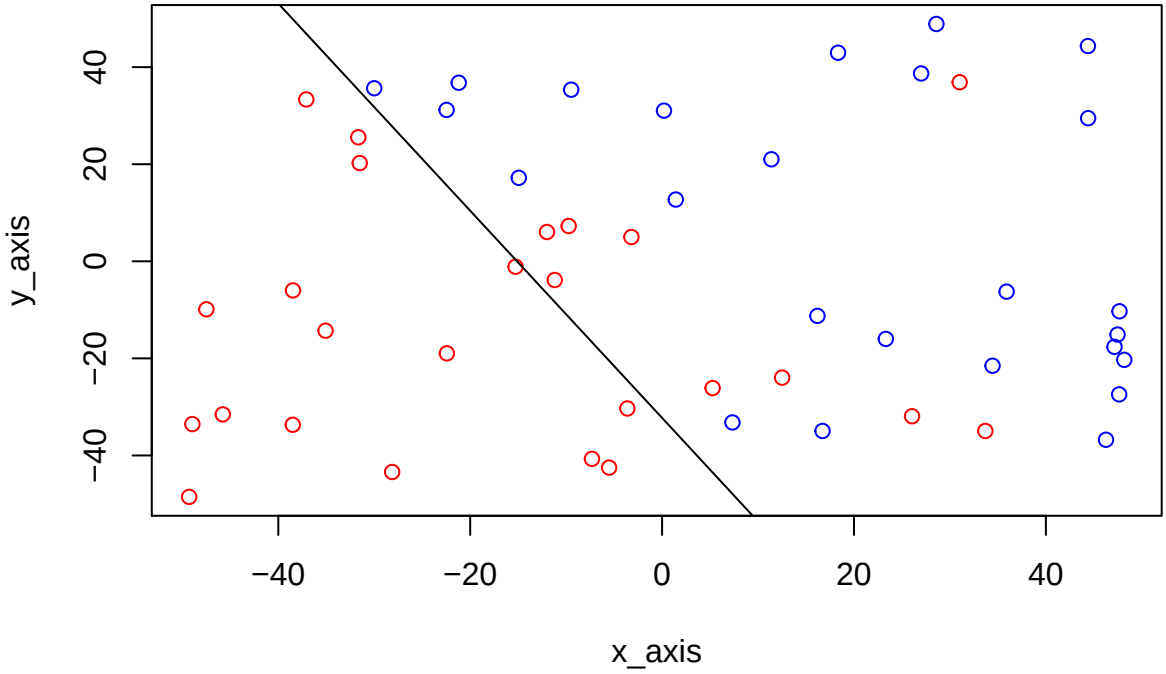

```
## Iteraciones:  3 Value: ( 0 0 0 )
```



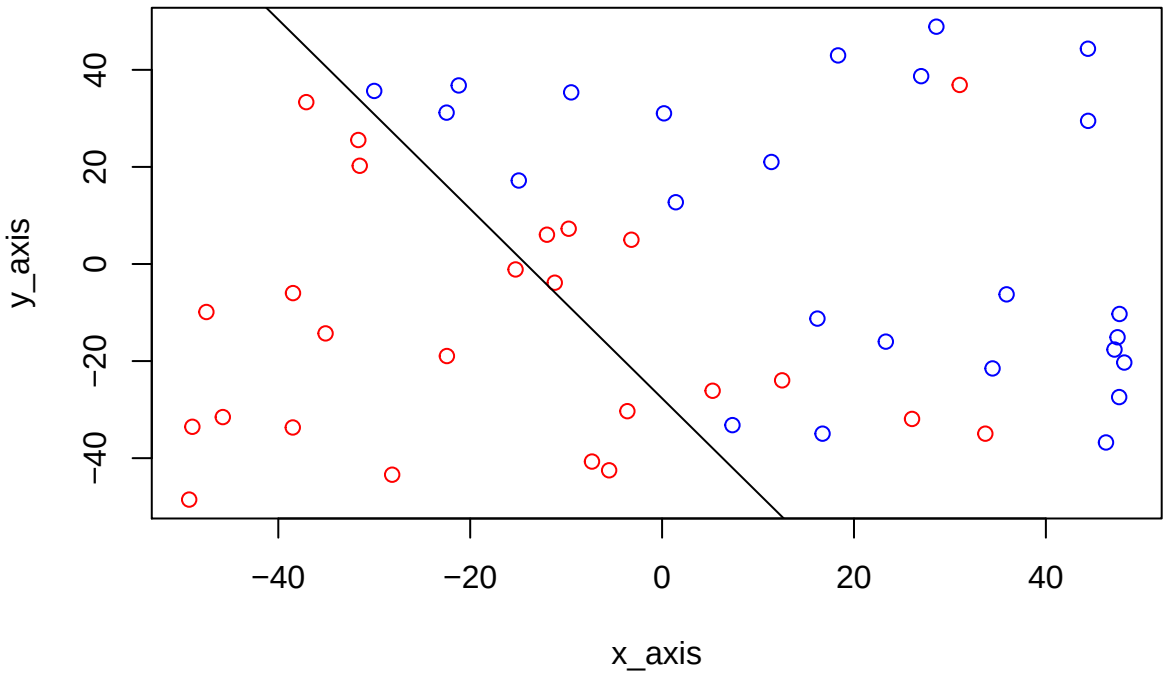

```
## Iteraciones:  90 Inicio: ( 0.8992507 0.04573869 0.08961809 )
```



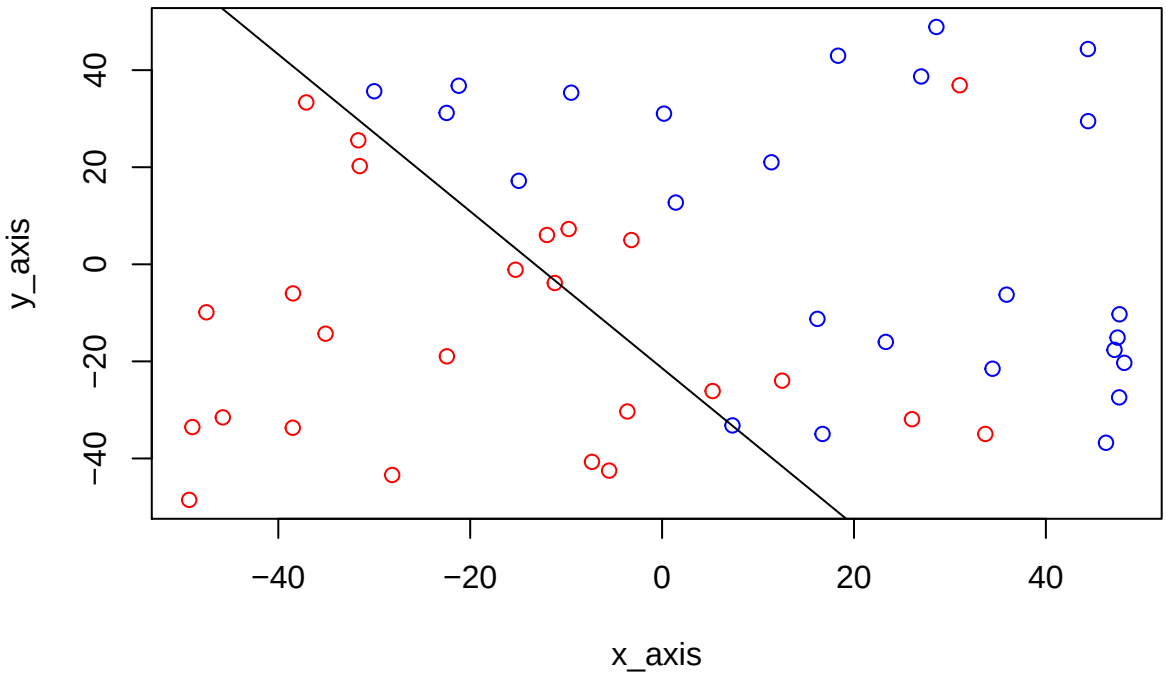

```
## Iteraciones:  91 Inicio: ( 0.4408768 0.7779656 0.1400851 )
```



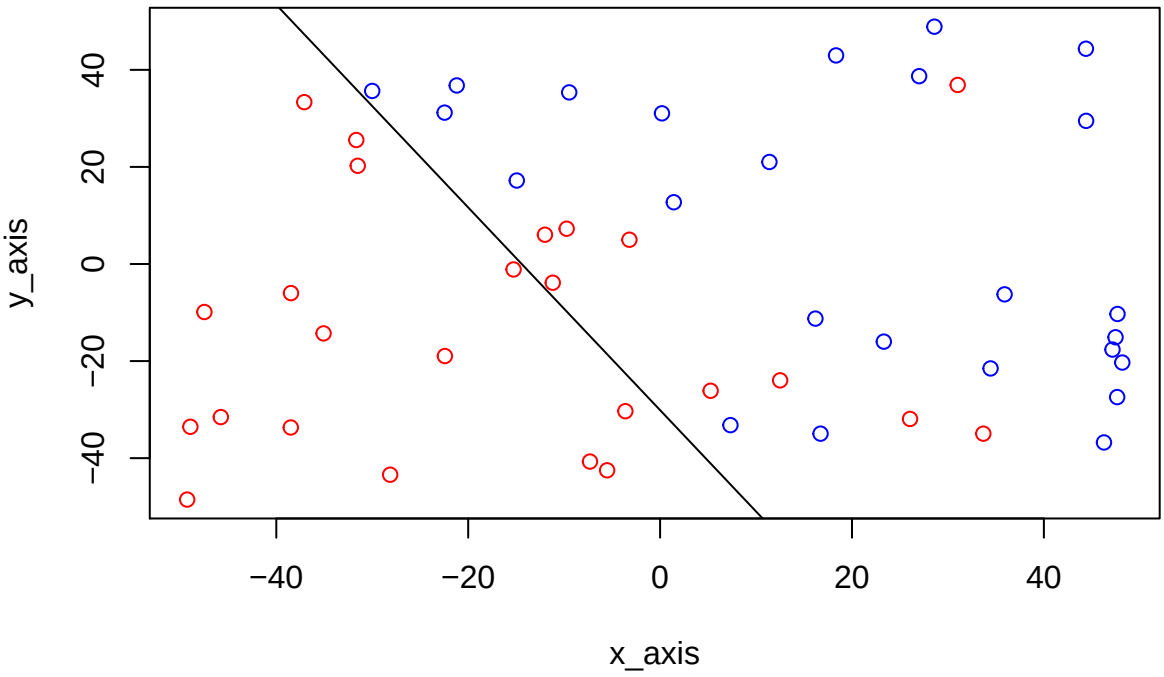

```
## Iteraciones:  99 Inicio: ( 0.6787487 0.04759801 0.6056478 )
```



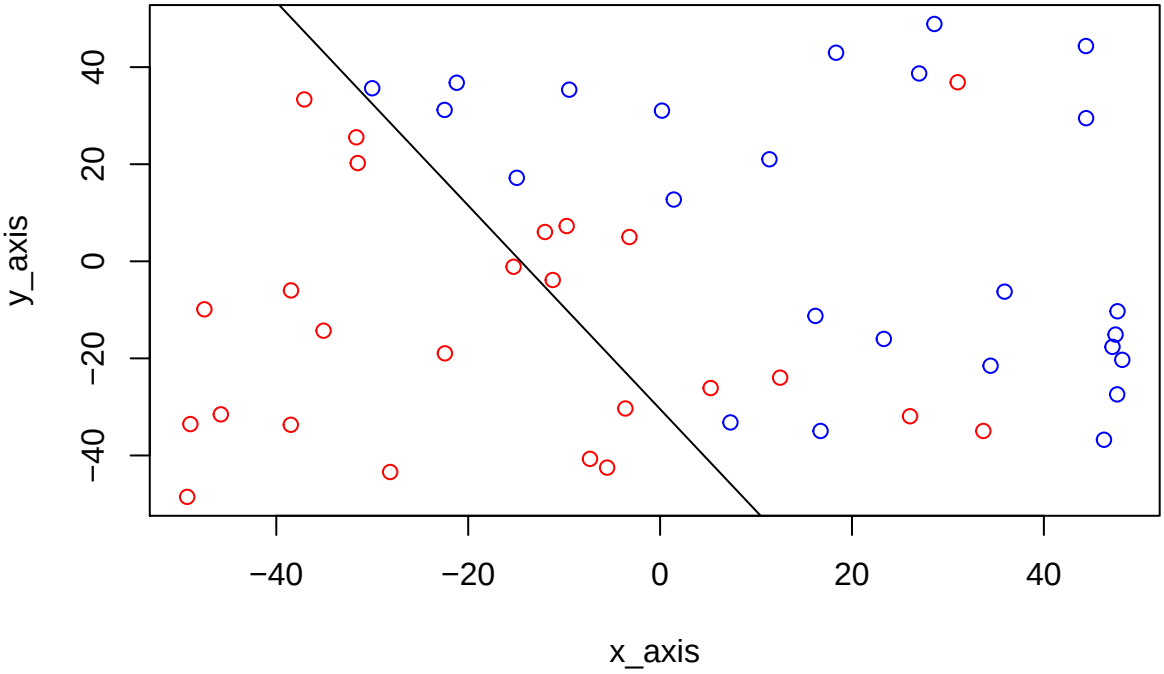

```
## Iteraciones:  91 Inicio: ( 0.4501444 0.7477808 0.1771495 )
```



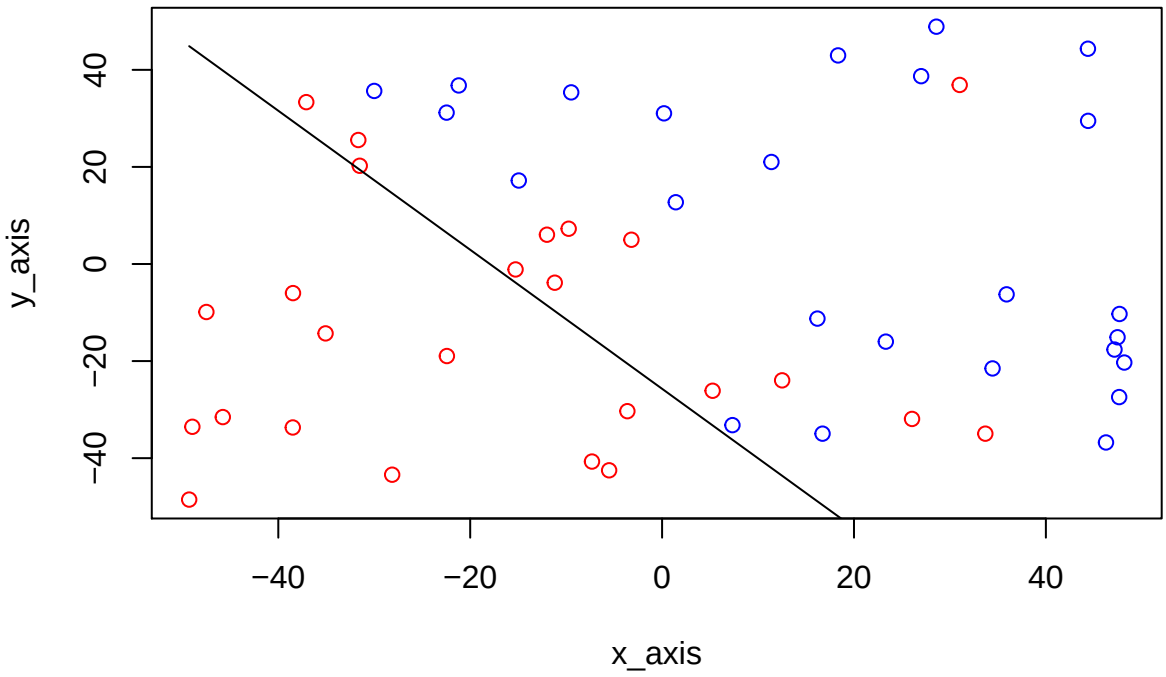

```
## Iteraciones:  96 Inicio: ( 0.6548571 0.1641085 0.2617137 )
```



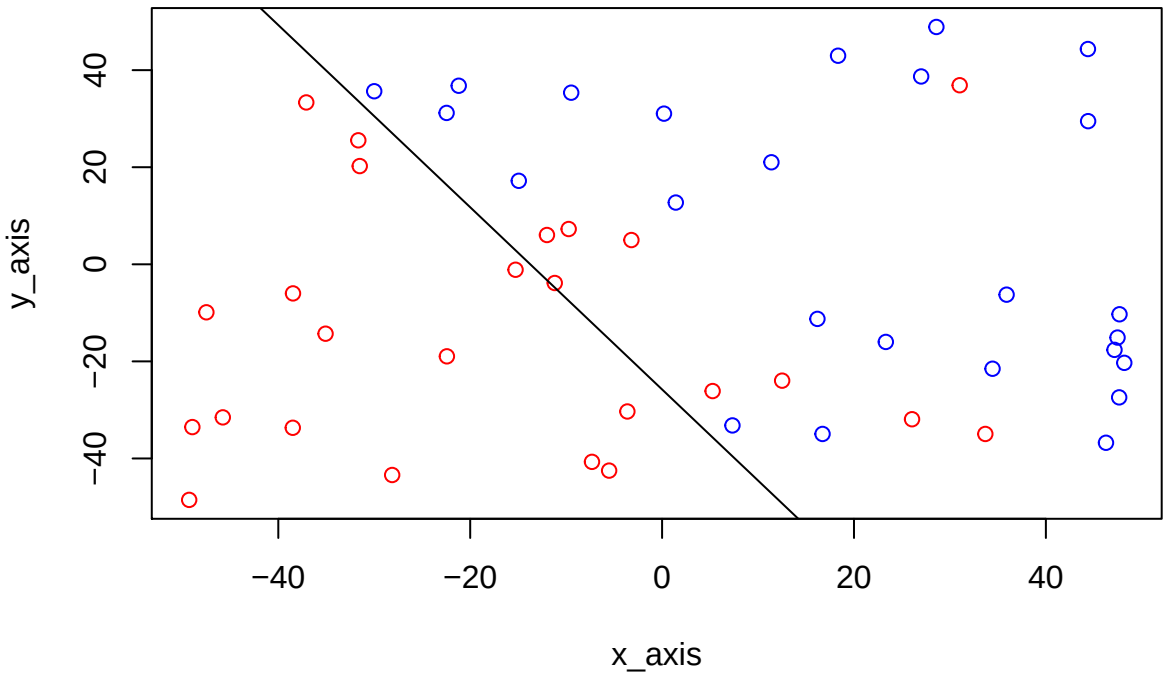

```
## Iteraciones:  93 Inicio: ( 0.09503739 0.3944602 0.2132182 )
```



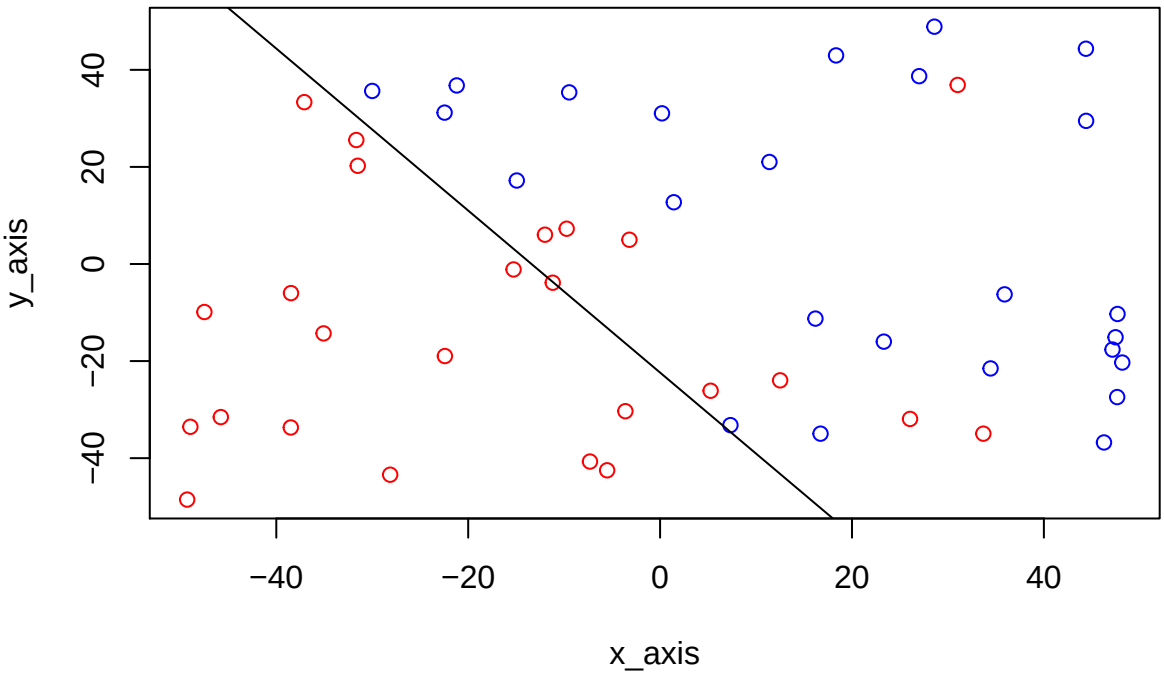

```
## Iteraciones:  93 Inicio: ( 0.6405314 0.2576197 0.09461404 )
```




```
## Iteraciones:  118 Inicio: ( 0.07039834 0.1883563 0.8731385 )
```




```
## Iteraciones:  94 Inicio: ( 0.9811036 0.3909197 0.01789851 )
```




```
## Iteraciones:  126 Inicio: ( 0.0445446 0.1077643 0.5703009 )

## Nº Iteraciones Medio:  90.36364

## El numero medio de iteraciones es más grande debido a que estamos trabajando con
## un conjunto no separable
```

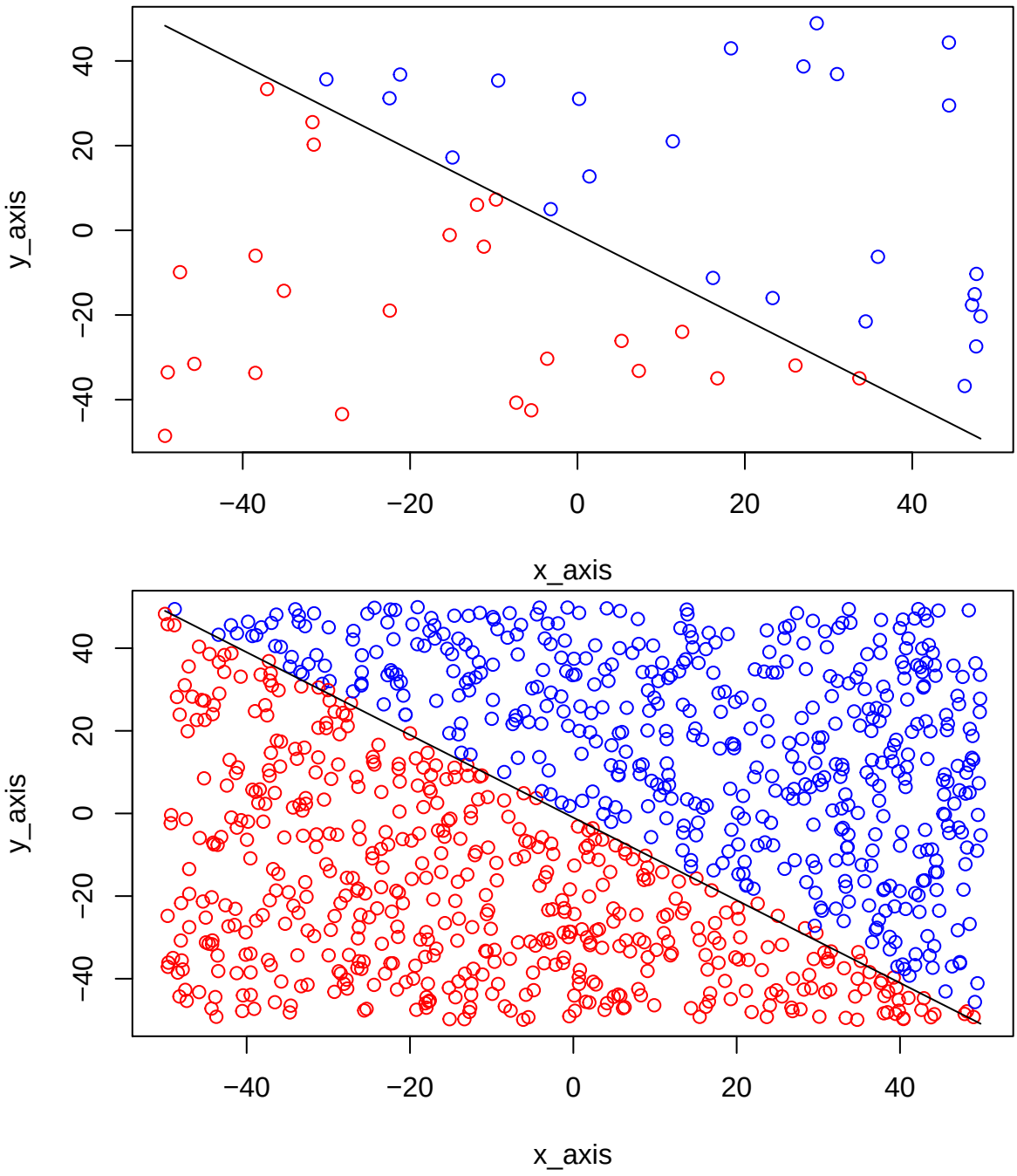
Ejercicio 2 Regresión Logística: En este ejercicio crearemos nuestra propia función objetivo f (una probabilidad en este caso) y nuestro conjunto de datos D para ver cómo funciona regresión logística. Supondremos por simplicidad que f es una probabilidad con valores 0/1 y por tanto que la etiqueta y es una función determinista de x.

Consideremos $d = 2$ para que los datos sean visualizables, y sea $\chi = [0, 2] \times [0, 2]$ con probabilidad uniforme de elegir cada $x\chi$ .Elegir una línea en el plano que pase por χ como la frontera entre $f(x) = 1$ (donde y toma valores +1) y $f(x) = 0$ (donde y toma valores -1), para ello seleccionar dos puntos aleatorios del plano y calcular la línea que pasa por ambos. Seleccionar $N = 100$ puntos aleatorios $\{x_n\}$ de χ y evaluar las respuestas $\{y_n\}$ de todos ellos respecto de la frontera elegida.

a) Implementar Regresión Logística(RL) con Gradiente Descendente Estocástico (SGD) bajo las siguientes condiciones: Inicializar el vector de pesos con valores 0. Parar el algoritmo cuando $||w^{(t-1)} - w^{(t)}|| < 0,01$, donde $w^{(t)}$ denota el vector de pesos al final de la época t. Una época es un pase completo a través de los N datos.

Aplicar una permutación aleatoria, 1, 2, . . ., N , en el orden de los datos antes de usarlos en cada época del algoritmo. Usar una tasa de aprendizaje de $mu = 0,01$

b) Usar la muestra de datos etiquetada para encontrar nuestra solución g y estimar E_{out} usando para ello un número suficientemente grande de nuevas muestras (> 999).




```
## El valor de Eout es 0.3110354
```