Práctica 1 Aprendizaje Automático

Jacinto Carrasco Castillo

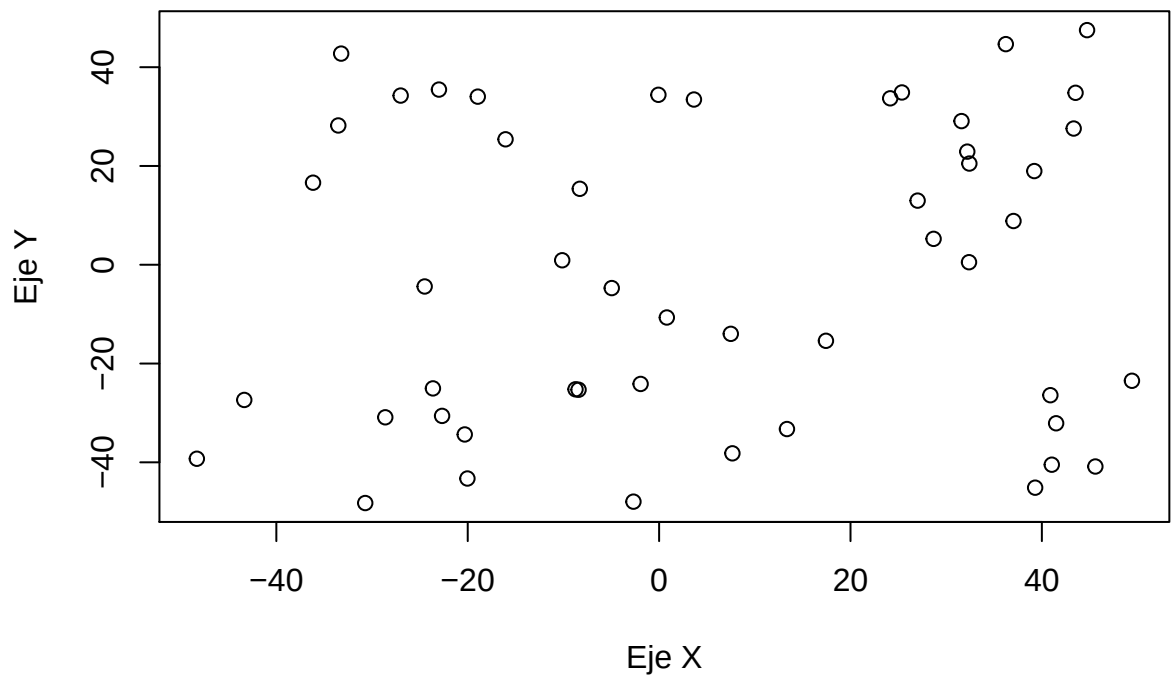
Generación y visualización de datos

```
set.seed(3141592)
simula_unif <- function(N, dim, rango){
  lista <- matrix(runif(N*dim, min =  rango[1], max = rango[2]), dim, N)
  return(lista)
}
```

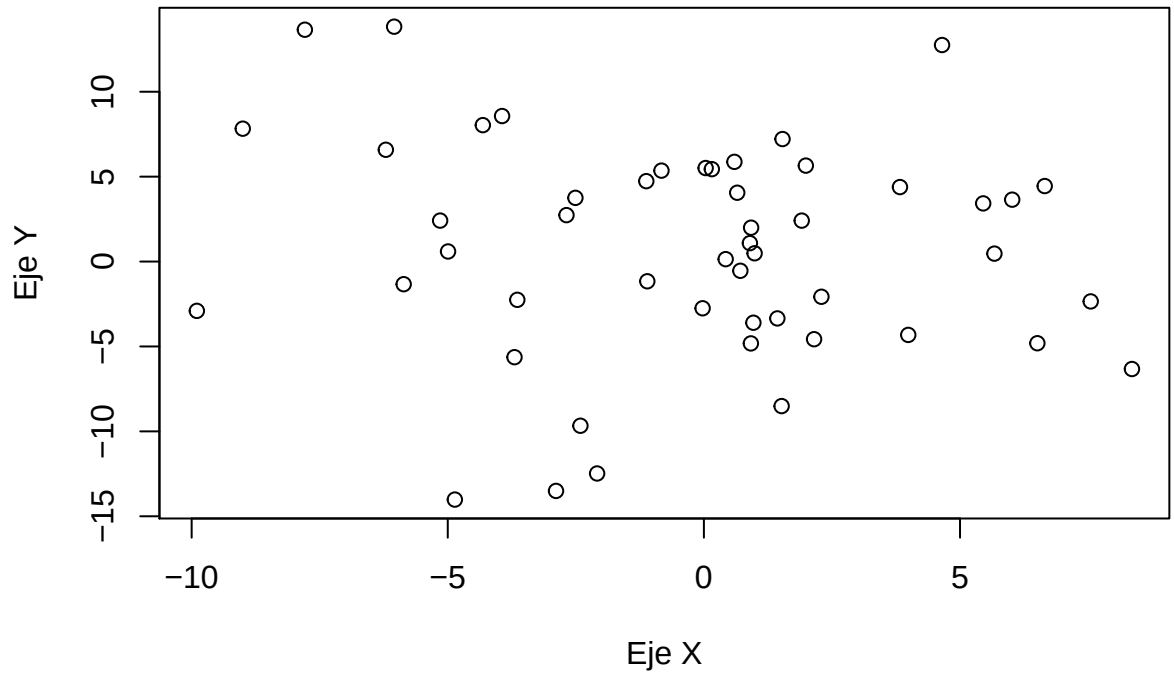
```
simula_gauss <- function(N, dim, sigma){
  lista <- matrix(rnorm(N*dim, sd = sigma), dim, N)
  return(lista)
}
```

```
lista_unif <- simula_unif(50, 2, c(-50, 50))
lista_gauss <- simula_gauss(50, 2, c(5,7))
```

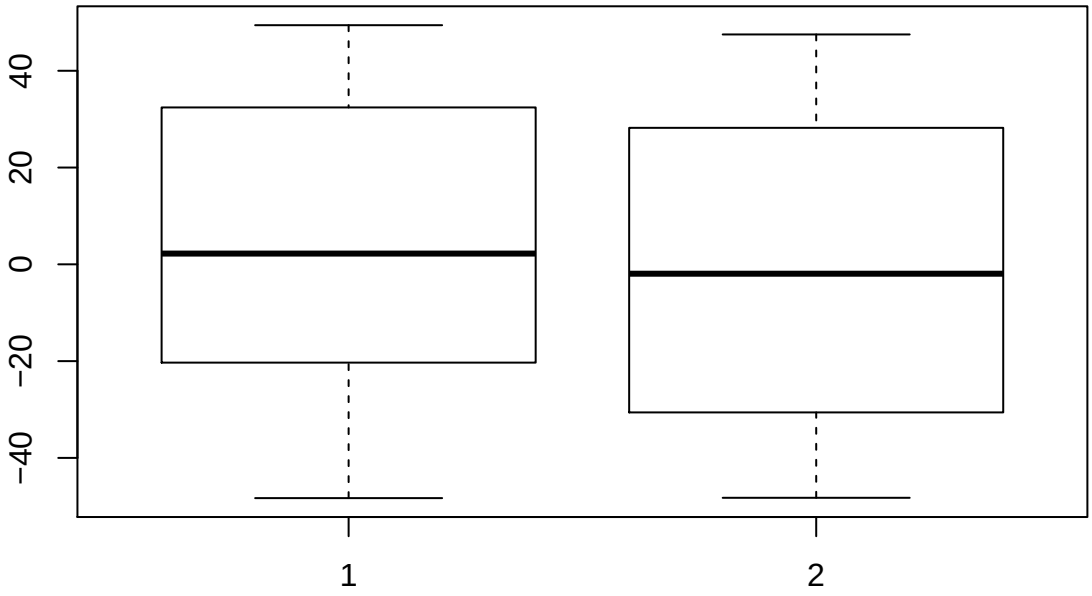
```
plot(lista_unif[1,], lista_unif[2,], xlab="Eje X", ylab="Eje Y")
```



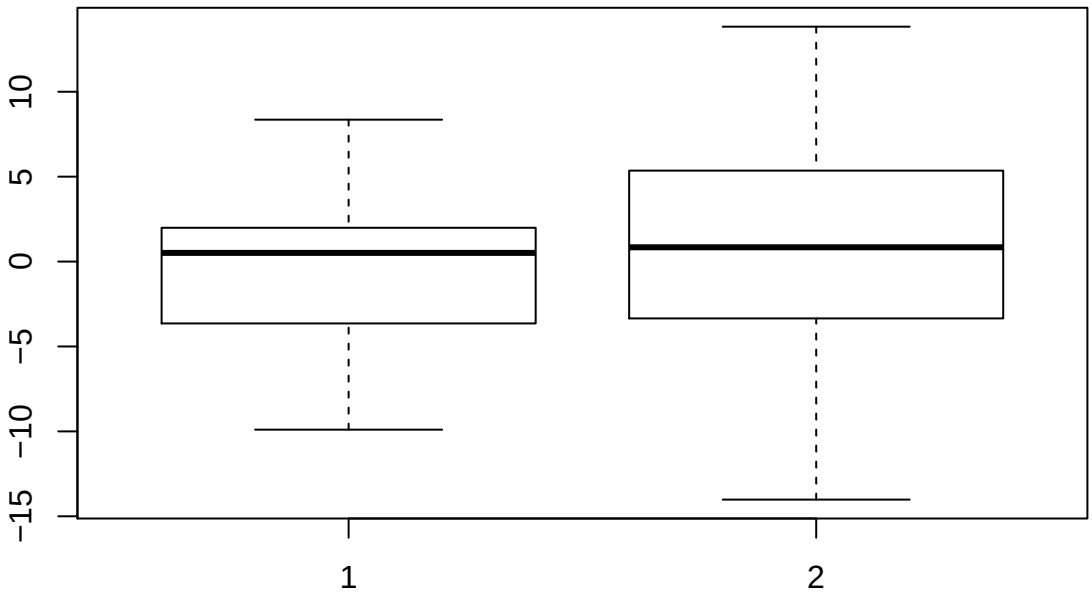

```
plot(lista_gauss[1,],lista_gauss[2,], xlab="Eje X", ylab="Eje Y")
```




```
boxplot(lista_unif[1,],lista_unif[2,])
```

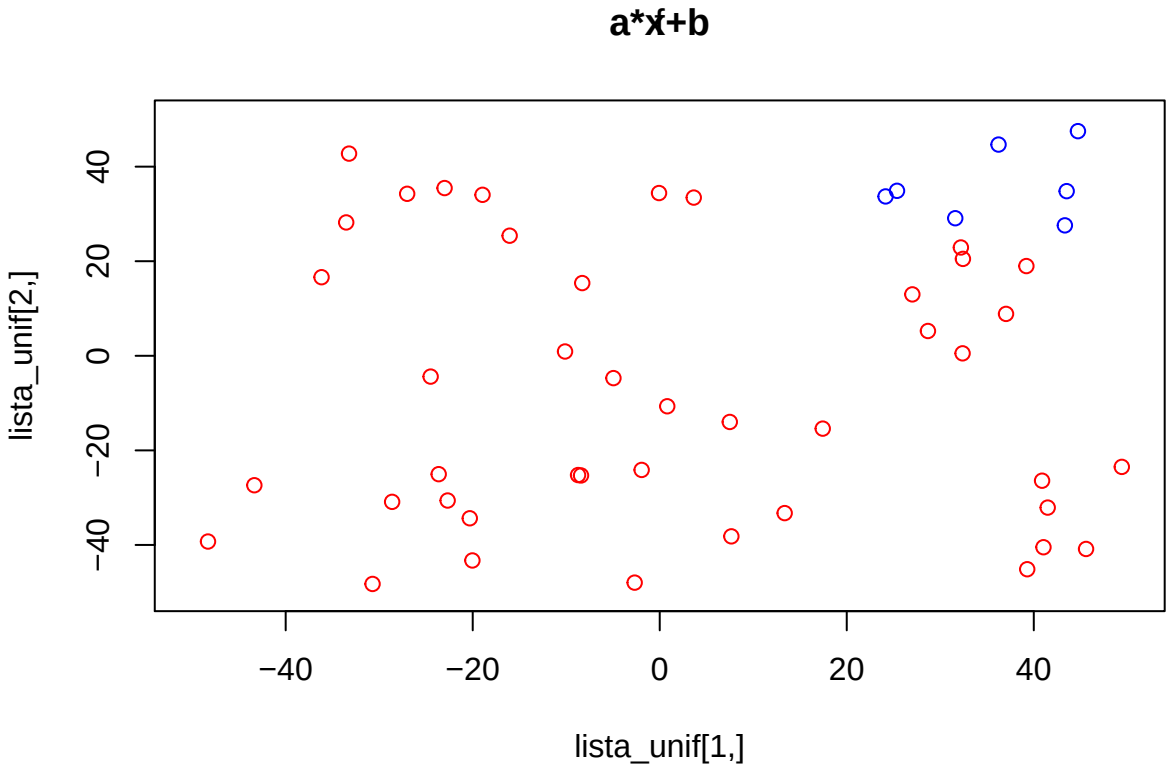



```
boxplot(lista_gauss[1,],lista_gauss[2,])
```

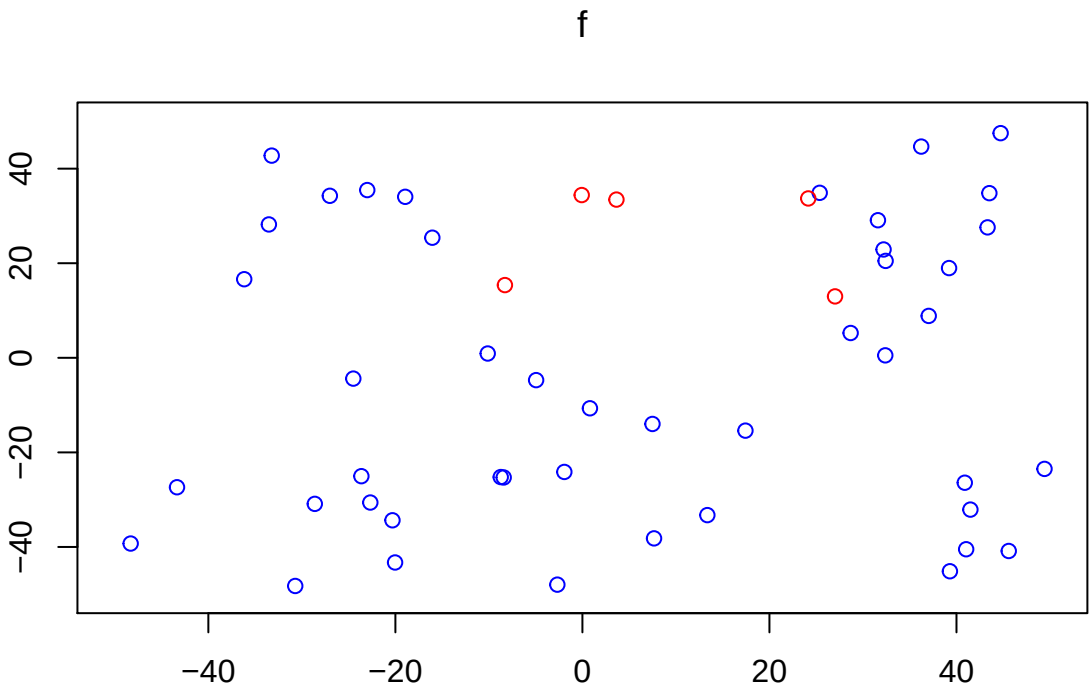



```
simula_recta <- function(intervalo){
  puntos <- simula_unif(2,2,intervalo)
  a <- (puntos[2,2]-puntos[2,1])/(puntos[1,2]-puntos[1,1])
  b <- puntos[2,1] - a * puntos[1,1]
  return(c(a,b))
}
```

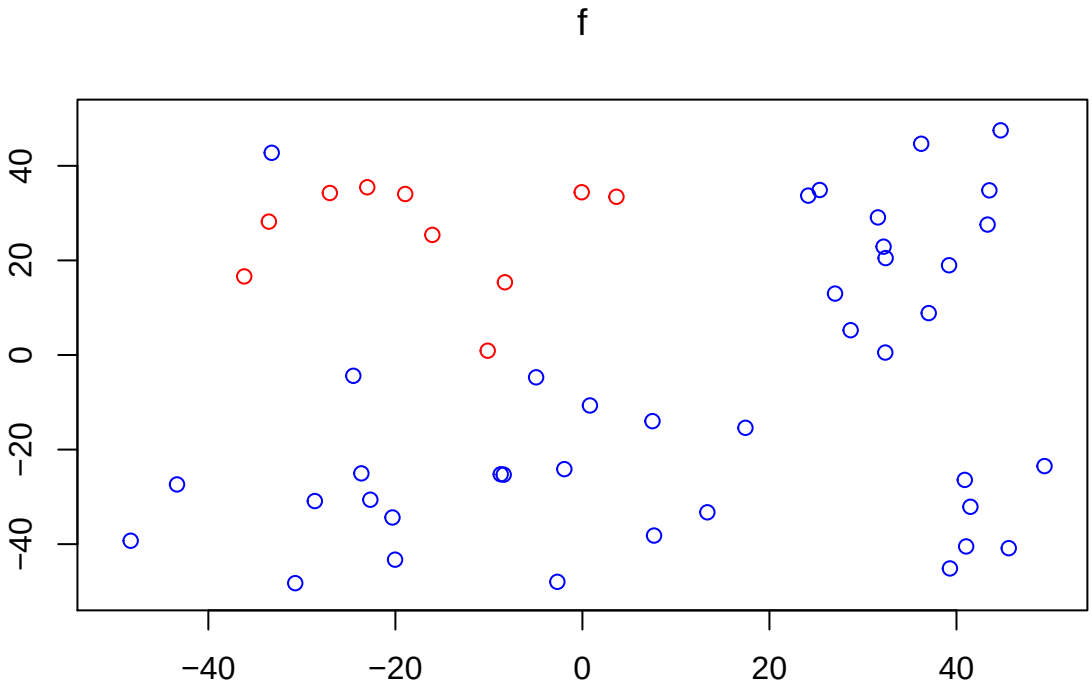
```
intervalo = c(-50, 50)
coefs <- simula_recta(intervalo)
a <- coefs[1]
b <- coefs[2]

f <- function(X){
  return(X[2] - a*X[1] -b)
}


etiqueta = apply(lista_unif, 2, function(X) sign(f(X)))


draw_function <- function(interval_x, interval_y, f, levels = 0, col = 0, add = FALSE){
  x <- seq(interval_x[1],interval_x[2],length=1000)
  y <- seq(interval_y[1],interval_y[2],length=1000)
  z <- outer(x,y, f)

  contour(x,y,z, levels=0, main=expression(f), col = col, add = add)
}


draw_function(c(-50,50), c(-50,50),function(x,y) return(y - a*x -b))
title(main="a*x+b", ylab = "lista_unif[2,]", xlab = "lista_unif[1,]")
points(lista_unif[1,], lista_unif[2,], col = (etiqueta+3))
```



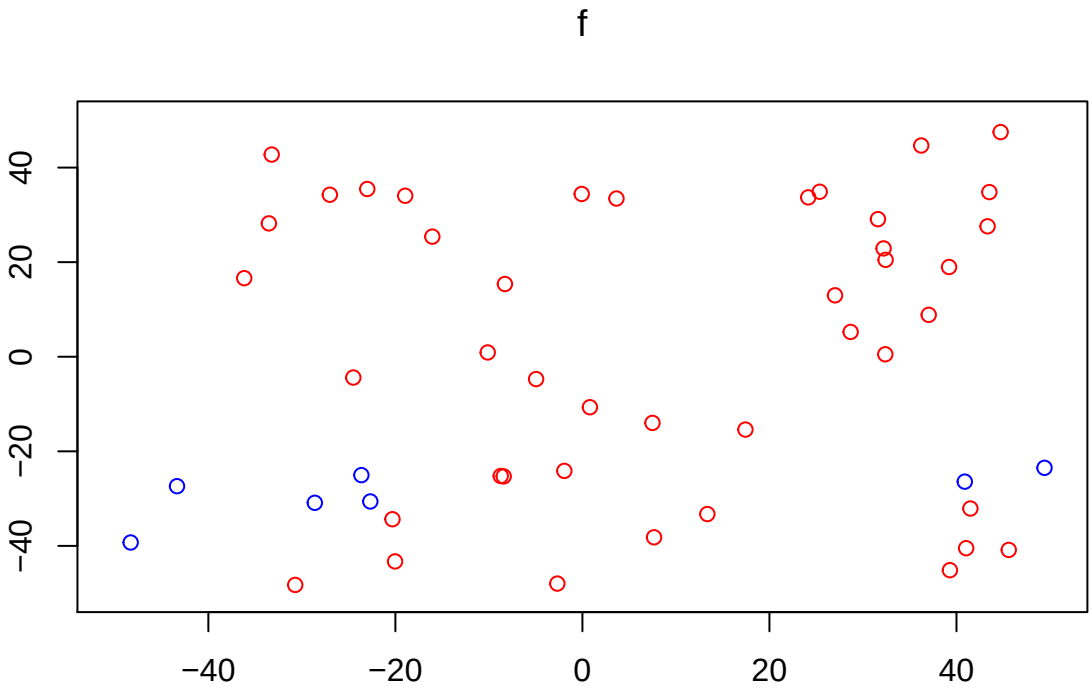

```
f_1 <- function(X){
  return((X[1] - 10)^2 + (X[2] - 20)^2 - 400)
}

etiqueta_1 = apply(lista_unif, 2, function(X) sign(f_1(X)))

draw_function(c(-50,50), c(-50,50), function(x,y) (x - 10)^2 + (y - 20)^2 - 400)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_1+3))
```



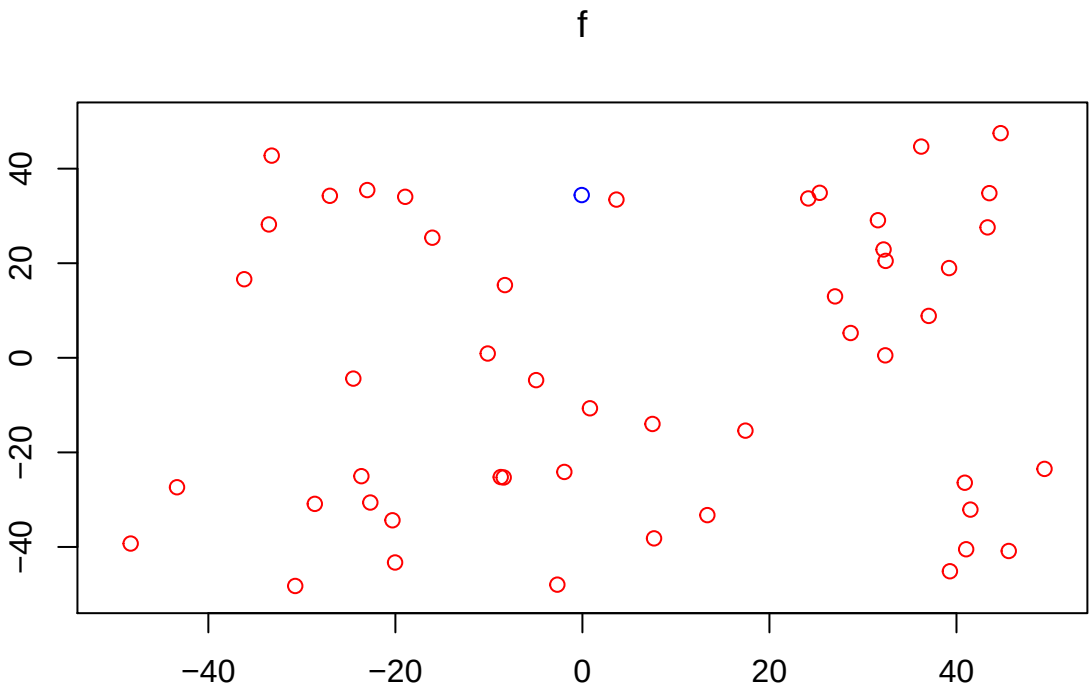

```
f_2 <- function(X){
  return(0.5*(X[1] + 10)^2 + (X[2] - 20)^2 - 400)
}

etiqueta_2 = apply(lista_unif, 2, function(X) sign(f_2(X)))

draw_function(c(-50,50), c(-50,50),function(x,y) 0.5*(x + 10)^2 + (y - 20)^2 - 400)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_2+3))
```



```r
f_3 <- function(X){
  return(0.5*(X[1] - 10)^2 - (X[2] + 20)^2 - 400)
}

etiqueta_3 = apply(lista_unif, 2, function(X) sign(f_3(X)))

draw_function(c(-50,50), c(-50,50), function(x,y) 0.5*(x - 10)^2 - (y + 20)^2 - 400)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_3+3))
```

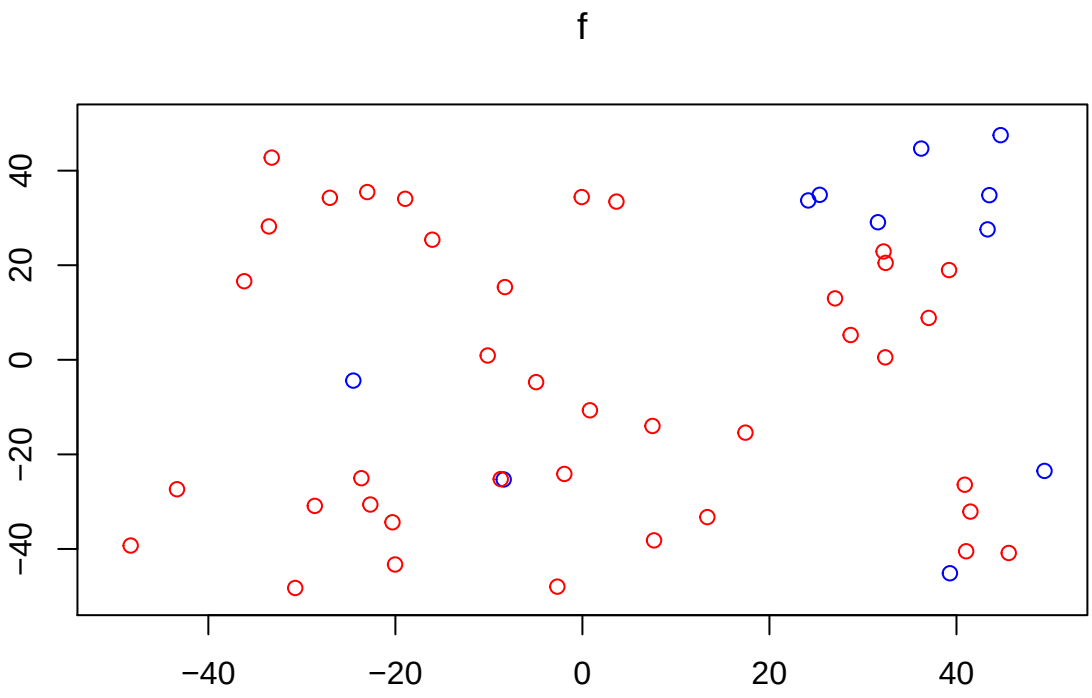



```r
f_4 <- function(X){
  return(X[2] - 20*X[1]^2- 5*X[1] + 3)
}

etiqueta_4 = apply(lista_unif, 2, function(X) sign(f_4(X)))

draw_function(c(-50,50), c(-50,50),function(x,y) y - 20*x^2 - 5*x + 3)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_4+3))
```

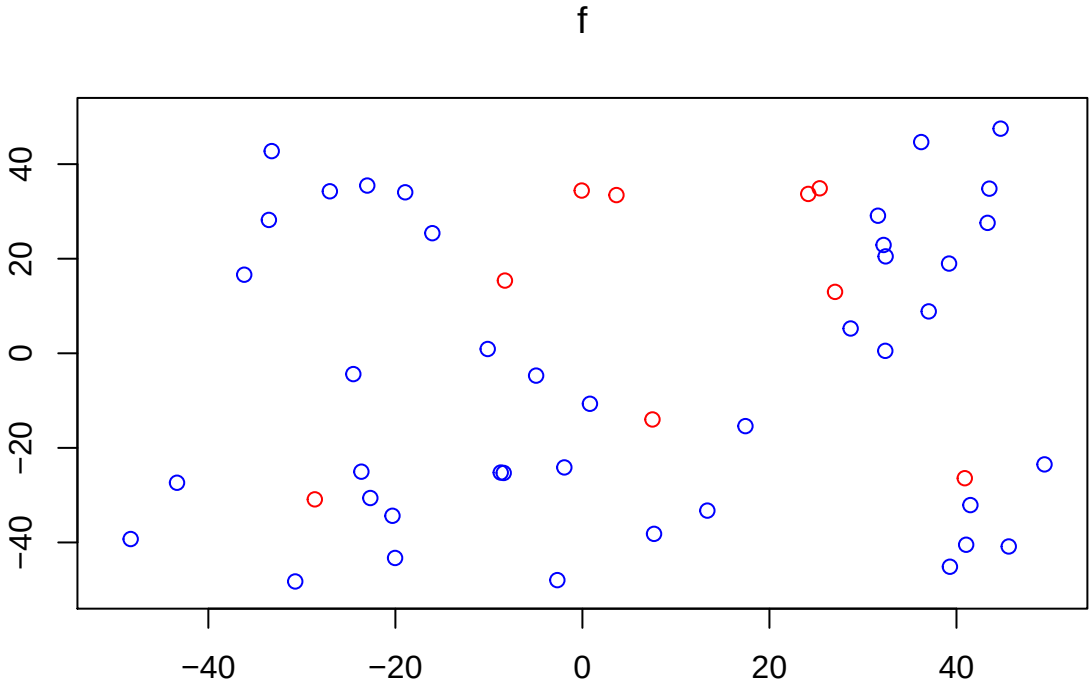



```
modify_rnd_bool_subvector <- function(v, perc = 0.1){
  mod_v <- v

  if( length(which(v == 1)) >= 10){
    to_change <- sample(which(v == 1), perc*length(which(v == 1)) )
    mod_v[to_change] = -1
  }

  if( length(which(v == -1)) >= 10){
    to_change <- sample(which(v == -1), perc*length(which(v == -1)) )
    mod_v[to_change] = 1
  }

  return(mod_v)
}
```

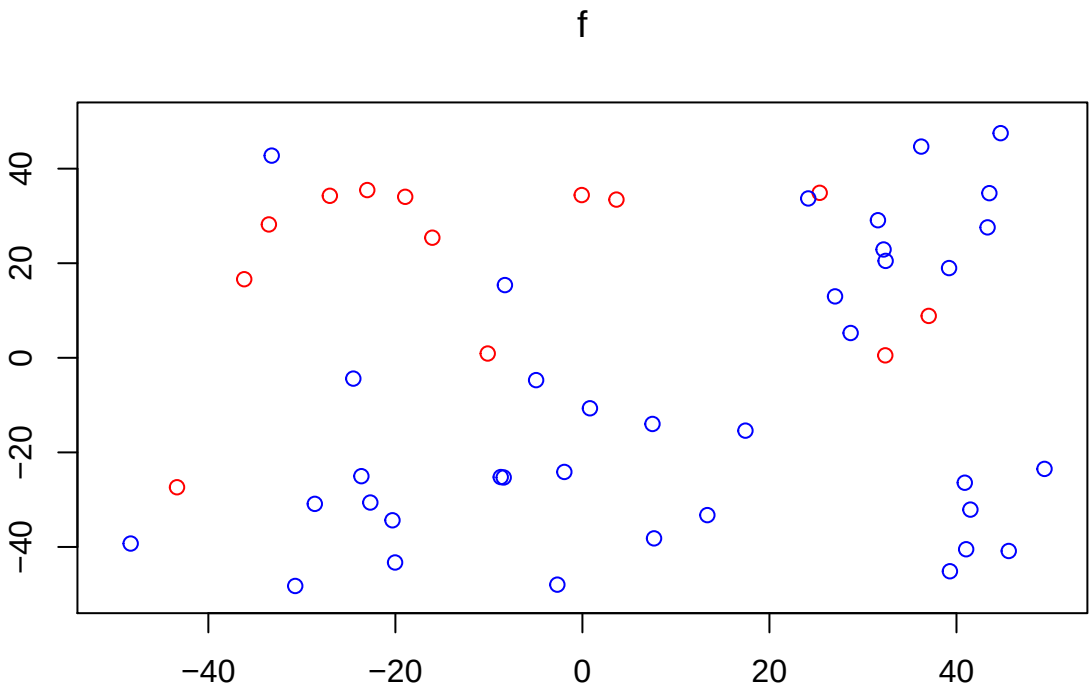
```
etiqueta_mod <- modify_rnd_bool_subvector(etiqueta)

draw_function(c(-50,50), c(-50,50), function(x,y) y-a*x-b)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_mod+3))
```



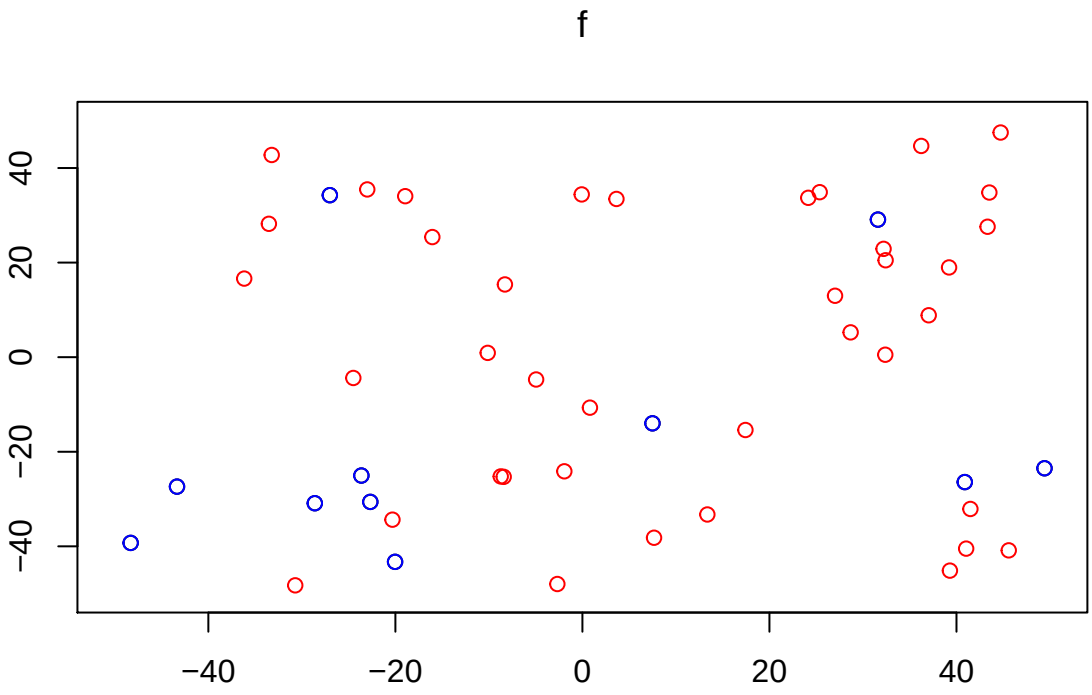

```
etiqueta_mod_1 <- modify_rnd_bool_subvector(etiqueta_1)

draw_function(c(-50,50), c(-50,50),function(x,y) (x - 10)^2 + (y - 20)^2 - 400)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_mod_1+3))
```




```
etiqueta_mod_2 <- modify_rnd_bool_subvector(etiqueta_2)

draw_function(c(-50,50), c(-50,50),function(x,y) 0.5*(x + 10)^2 + (y - 20)^2 - 400)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_mod_2+3))
```

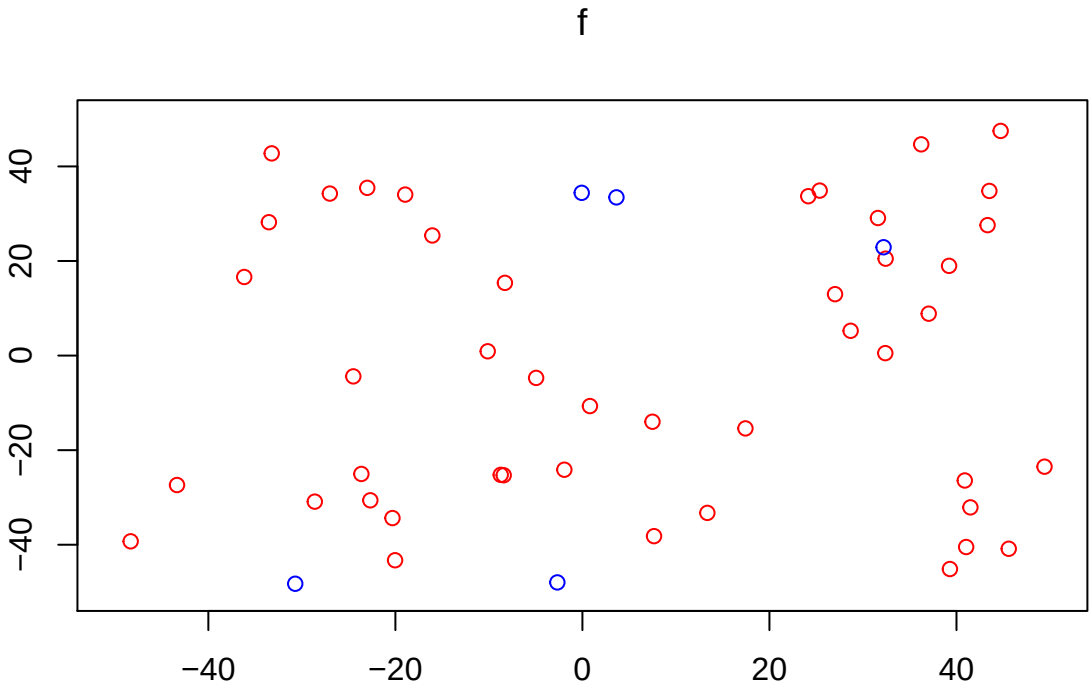



```
etiqueta_mod_3 <- modify_rnd_bool_subvector(etiqueta_3)

draw_function(c(-50,50), c(-50,50),function(x,y) 0.5*(x - 10)^2 - (y + 20)^2 - 400)
points(lista_unif[1,etiqueta_mod_3==1], lista_unif[2,etiqueta_mod_3==1])
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_mod_3+3))
```




```
etiqueta_mod_4 <- modify_rnd_bool_subvector(etiqueta_4)

draw_function(c(-50,50), c(-50,50),function(x,y) y - 20*x^2 - 5*x + 3)
```

```
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_mod_4+3))
```



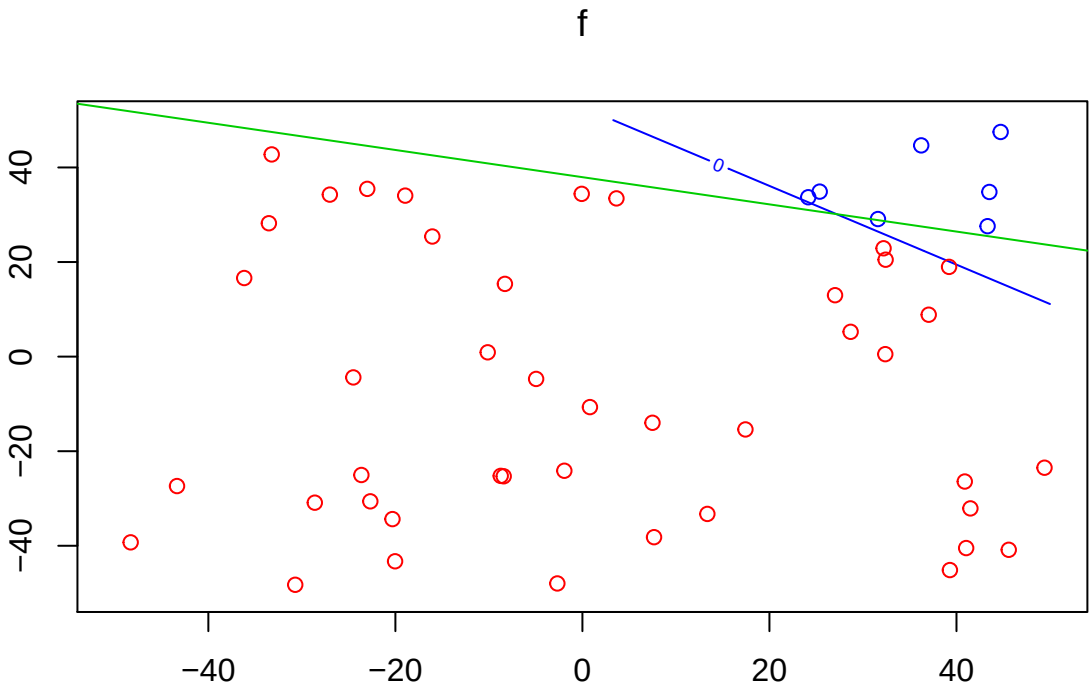
Ajuste del algoritmo Perceptron

```
ajusta_PLA <- function(datos, label, max_iter, vini, draw_iterations = FALSE, col = 0){
  sol <- vini
  iter <- 0
  changed <- TRUE

  while ( iter < max_iter && changed){

    changed <- FALSE
    for( inner_iter in 1:ncol(datos) ){
      x <- c(datos[,inner_iter],1)

      if( sign(crossprod(x,sol)) != label[inner_iter]){
        sol <- sol + label[inner_iter] * x
        changed <- TRUE
      }
    }
    if(draw_iterations){
      if(iter < 10) name = paste('000', iter, 'plot.png', sep='')
      else if(iter < 100) name = paste('00', iter, 'plot.png', sep='')
      else name = paste('0', iter, 'plot.png', sep='')
      png(name)
      draw_function(c(-50,50), c(-50,50), function(x,y) y +sol[1]/sol[2]*x
                    +sol[3]/sol[2], col = 4)
      points(lista_unif[1,], lista_unif[2,], col = (label+3))
```

```
      dev.off()
    }
    iter <- iter+1
  }
  sol <- sol/sol[length(sol)-1]

  return( list( hiperplane = sol, iterations = iter))
}

sol_pla <- ajusta_PLA(lista_unif,etiqueta, 7000,rep(0,3))$hiperplane
```

Representación de ejecución del algoritmo PLA

```
draw_function(c(-50,50), c(-50,50), function(x,y) y+sol_pla[1]*x+sol_pla[3], col = 4)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta+3))
abline(b, a, col=3)
```



Cálculo de la media para 10 ejecuciones del PLA con el vector (0,0,0) y vectores aleatorios:

```
iterations <- numeric()
length(iterations) <- 10

iterations[1] <- ajusta_PLA(lista_unif,etiqueta, 10000,rep(0,3))$iterations

for(i in 2:10){
  v_ini <- runif(3)
  iterations[i] <- ajusta_PLA(lista_unif,etiqueta, 10000, v_ini)$iterations
}

print(c("Media de iteraciones ", mean(iterations)))
```

```
## [1] "Media de iteraciones " "64.8"
```

```
count_errors <- function(f, datos, label){
  #Conteo de errores
  signes <- apply(datos, 2, function(x) return(sign(f(c(x,1)))))
  v <- signes != label
  return(length(v[v]))
}
```

```
hiperplane_to_function <- function( vec ){
  f <- function(x){
    return( crossprod(vec,x) )
  }

  return(f)
}
```

Conteo de errores

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 10,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

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

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 100,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 12
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod, 1000,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 13
```

Conteo de errores función cuadrática

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 10,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 29
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 100,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 22
```

```
sol <- ajusta_PLA(lista_unif,etiqueta_mod_1, 1000,rep(0,3))$hiperplane
count_errors(hiperplane_to_function(sol), lista_unif, etiqueta_mod )
```

```
## [1] 39
```

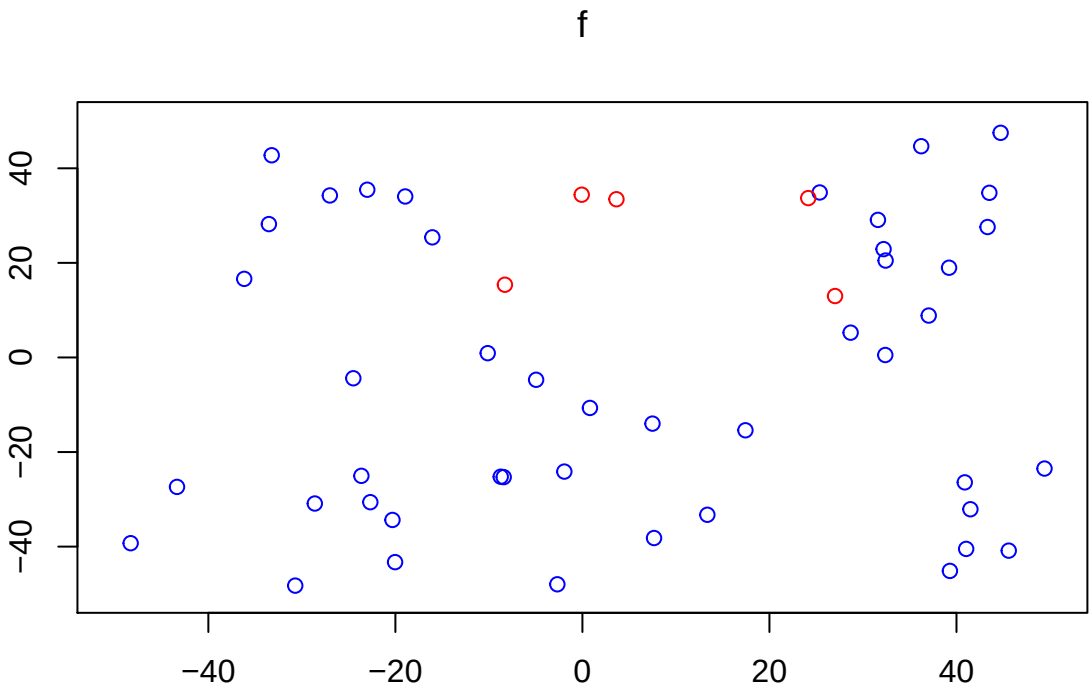
Algoritmo PLA Modificado

```
ajusta_PLA_MOD <- function(datos, label, max_iter, vini, draw_iterations = FALSE){
  sol <- vini
  iter <- 0
  changed <- TRUE
  current_sol <- sol

  while ( iter < max_iter && changed){
    current_errors <- count_errors(hiperplane_to_function(sol), datos, label )
    changed <- FALSE

    for( inner_iter in 1:ncol(datos) ){
      x <- c(datos[,inner_iter],1)

      if( sign(crossprod(x,current_sol)) != label[inner_iter]){
        current_sol <- current_sol + label[inner_iter] * x
        changed <- TRUE
      }
    }

    if(draw_iterations){
      if(iter < 10) name = paste('000', iter, 'plot.png', sep='')
      else if(iter < 100) name = paste('00', iter, 'plot.png', sep='')
      else name = paste('0', iter, 'plot.png', sep='')
      png(name)
      draw_function(c(-50,50), c(-50,50),
                    function(x,y) y +sol[1]/sol[2]*x +sol[3]/sol[2], col = 4)
      draw_function(c(-50,50), c(-50,50),
                    function(x,y) y +current_sol[1]/current_sol[2]*x +current_sol[3]/current_sol[2], co
      points(lista_unif[1,], lista_unif[2,], col = (label+3))
      dev.off()
    }

    if( current_errors >= count_errors(hiperplane_to_function(current_sol),
                                       datos, label )){
      sol <- current_sol
    }

    iter <- iter+1
  }
  sol <- sol/sol[length(sol)-1]

  return( list( hiperplane = sol, iterations = iter))
}
```

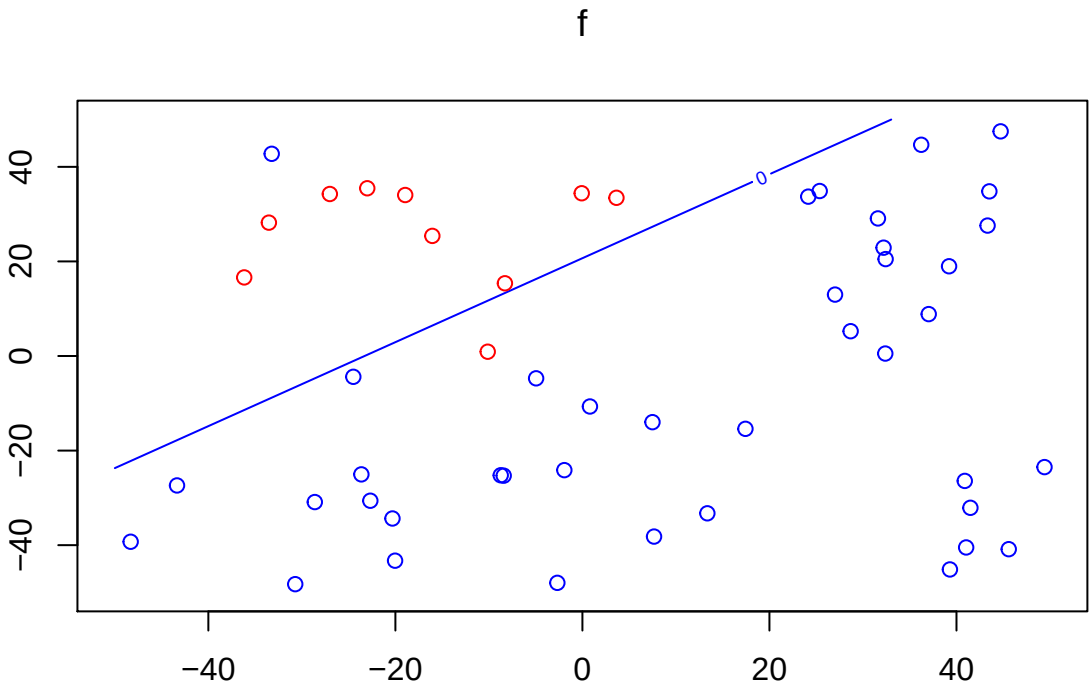
Evaluación funciones cuadráticas

```
sol_cuadratic_1 <- ajusta_PLA_MOD(lista_unif, etiqueta_1, 1000, rep(0,3))$hiperplane

draw_function(c(-50,50), c(-50,50), function(x,y)
```

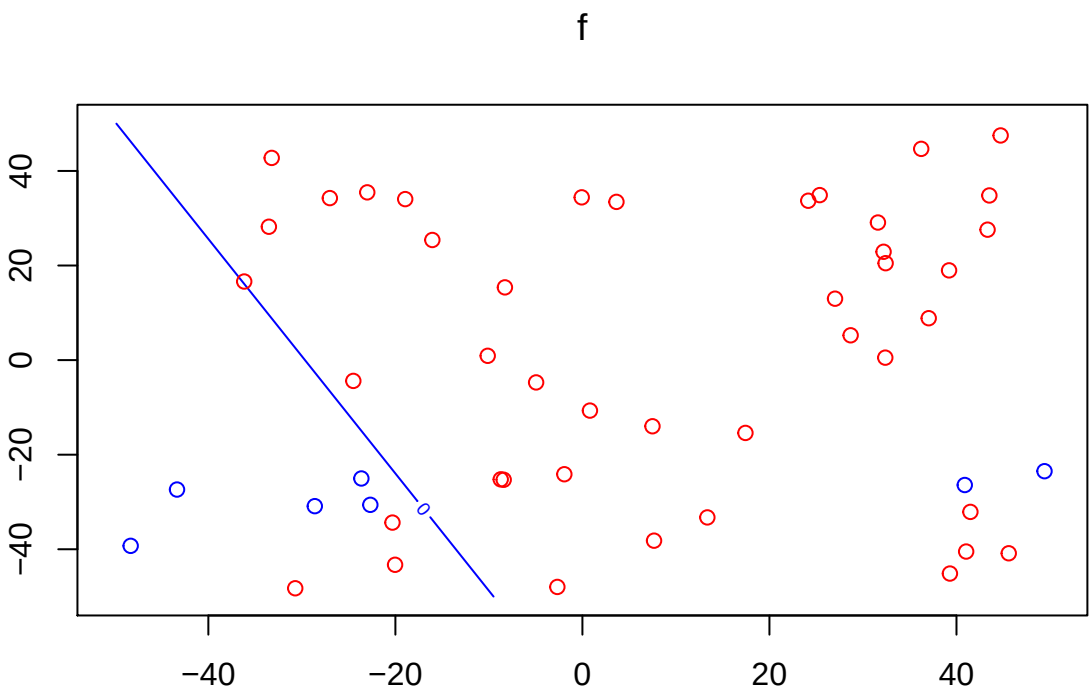
```
  y +sol_cuadratic_1[1]*x +sol_cuadratic_1[3], col = 4)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_1+3))
```



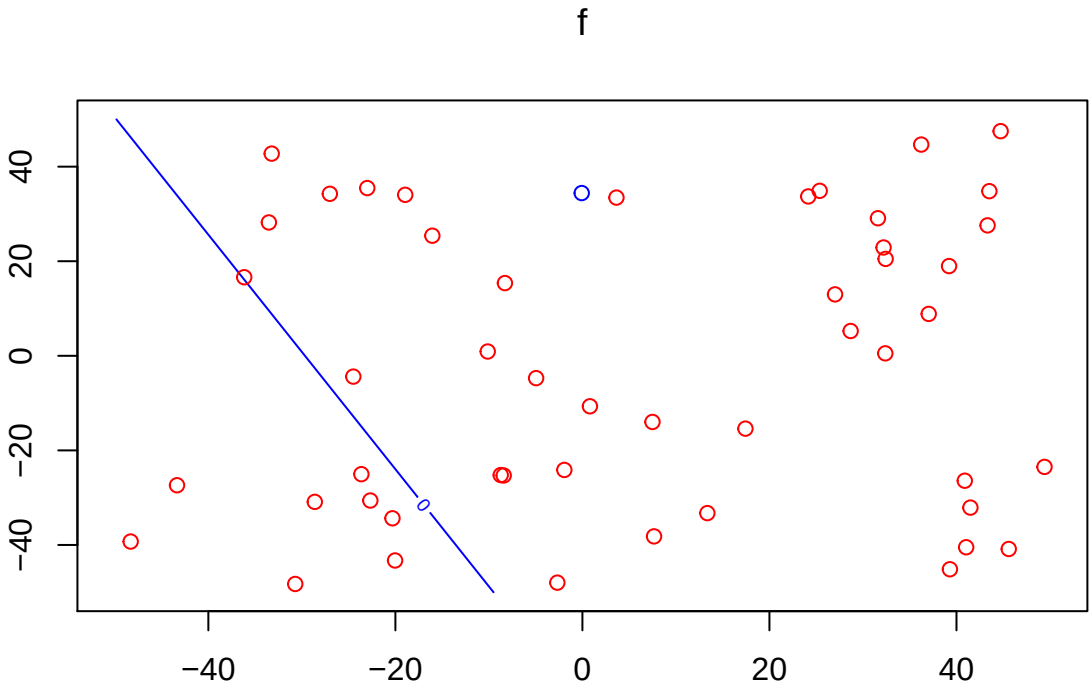

```
sol_cuadratic_2 <- ajusta_PLA_MOD(lista_unif, etiqueta_2, 1000, rep(0,3))$hiperplane

draw_function(c(-50,50), c(-50,50), function(x,y)
  y +sol_cuadratic_2[1]*x +sol_cuadratic_2[3], col = 4)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_2+3))
```



```
sol_cuadratic_3 <- ajusta_PLA_MOD(lista_unif, etiqueta_3, 1000, rep(0,3))$hiperplane

draw_function(c(-50,50), c(-50,50), function(x,y)
  y +sol_cuadratic_3[1]*x +sol_cuadratic_3[3], col = 4)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_3+3))
```




```
sol_cuadratic_4 <- ajusta_PLA_MOD(lista_unif, etiqueta_3, 1000, rep(0,3))$hiperplane

draw_function(c(-50,50), c(-50,50), function(x,y)
  y +sol_cuadratic_4[1]*x +sol_cuadratic_4[3], col = 4)
points(lista_unif[1,], lista_unif[2,], col = (etiqueta_4+3))
```



Drawing iterations

```
#sol <- ajusta_PLA(lista_unif,etiqueta, 1000,rep(0,3), TRUE)$hiperplane
#sol <- ajusta_PLA(lista_unif,etiqueta_mod, 1000,rep(0,3), TRUE)$hiperplane
#sol <- ajusta_PLA_MOD(lista_unif,etiqueta_mod, 1000,rep(0,3), TRUE)$hiperplane
```

Regresión lineal

Lectura de datos y dotación de forma

```
datos = scan("datos/zip.train",sep=" ")
datos = matrix(datos,ncol=257,byrow=T)
datos <- datos[-(which(datos[,1] != 1 & datos[,1]!=5)),]

number <- datos[,1]
pixels <- aperm(array(t(datos[,2:257]), dim=c(16,16,nrow(datos))), perm=c(2,1,3))
```

Cálculo de media y simetría

```
means <- apply(pixels, 3, mean)

simetria_vertical <- function(M){
  -sum(apply(M, 2, function(x) sum(abs(x-x[length(x):1]))))
}
```

```
sim_vertical <- apply(pixels,3,simetria_vertical)
```

Representación

```
number_colors = numeric(length(number))
number_colors[which(number==1)] <- 4
number_colors[which(number==5)] <- 3
data_list <- list('number'=number, 'pixels'=pixels, 'mean'=means,
                  'v_symmetry'=sim_vertical, 'colors'=number_colors )
plot(data_list$mean, data_list$v_symmetry, col = data_list$colors )
```