Extracción de información temática de imágenes usando *R* - V.0.3

Ivan Lizarazo

Abril de 2013

Resumen

Este documento muestra las posibilidades de realizar análisis de imagenes usando el software estadístico *R*. Se ilustra, paso a paso, el uso de tres técnicas diferentes, *distancia Mahalanobis*, *árboles de decisión* y *máquinas de soporte vectorial*, para realizar una clasificación supervisada de la cobertura del suelo. En su desarrollo, se utiliza una imagen de juguete (144 pixeles) que consta de cuatro bandas espectrales. Se simula un mapa raster con las clases de cobertura existentes en el terreno con el objeto de obtener muestras de entrenamiento y de validación. La exactitud tematica de los resultados se evalua usando la matriz de error. El lector puede usar este tutorial como referencia para realizar sus propios análisis basados en imágenes reales.

1. Introducción

Este tutorial se realizó utilizando la versión 2.15.3 del software *R* instalada en una máquina con sistema operativo Linux (Ubuntu 12.04). Para el desarrollo de los ejercicios prácticos se instalaron previamente las siguientes librerías:

- rgdal
- sp
- raster
- scatterplot3d

- mda
- vcd

Este tutorial no describe conceptos básicos de procesamiento de imágenes ni explica los fundamentos teóricos de las diferentes técnicas de clasificación utilizadas, es decir *distancia Mahalanobis*, *árboles de decision* ni *máquinas de soporte vectorial*. El autor asume que el lector ha revisado previamente dichos conceptos y técnicas y que, por tanto, entiende de manera general los supuestos y algoritmos asociados a cada técnica y está listo para realizar actividades prácticas.

2. Imagen de trabajo

Este tutorial utiliza como datos básicos una imagen obtenida con un sensor de juguete, con cuatro bandas espectrales. Cada una de las bandas comprende 12 filas y 12 columnas.

Las bandas inviduales se pueden descargar de los siguientes enlaces a su directorio de trabajo:
https://docs.google.com/file/d/0BzEwvK1H17qeVE5QTkdaSGhLU00/edit?usp=sharing
https://docs.google.com/file/d/0BzEwvK1H17qeMFVNNU9pMWk3WE0/edit?usp=sharing
https://docs.google.com/file/d/0BzEwvK1H17qeakRrd1lFeHk4dkk/edit?usp=sharing
https://docs.google.com/file/d/0BzEwvK1H17qeSHdmSl9Td2JHY1E/edit?usp=sharing

Inicie una sesion en R. Defina como su directorio de trabajo el sitio en donde estan las imagenes descargadas.Luego, lea las bandas individuales usando las siguientes instrucciones:

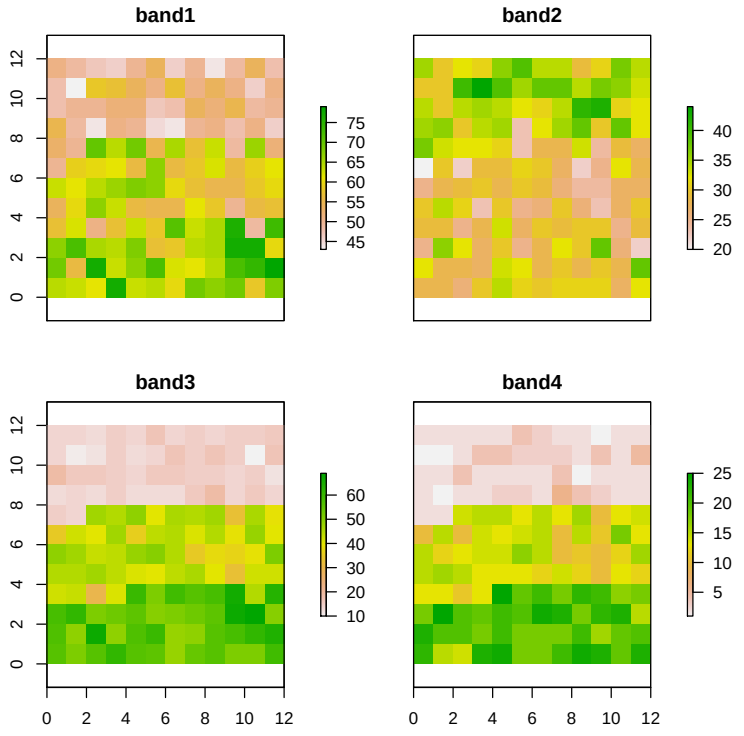
```
> library(raster)
> # lectura de banda 1
> b1 <- raster("band1.tif")
> # descripcion del objeto b1
> b1

class       : RasterLayer
dimensions  : 12, 12, 144  (nrow, ncol, ncell)
resolution  : 1, 1  (x, y)
extent      : 0, 12, 0, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : /home/ivan/ud/pdi avanzado/R/band1.tif
names       : band1
values      : 43, 79  (min, max)
```

```
> # lectura de las tres bandas adicionales
> b2 <- raster("band2.tif")
> b3 <- raster("band3.tif")
> b4 <- raster("band4.tif")
> # creacion de una imagen que agrupa las cuatro bandas
> toy <- stack(b1,b2,b3,b4)
> # asignacion de nombres especificos a cada banda
> names(toy) <- c("band1","band2","band3","band4")
> toy
```

```
class       : RasterStack
dimensions  : 12, 12, 144, 4  (nrow, ncol, ncell, nlayers)
resolution  : 1, 1  (x, y)
extent      : 0, 12, 0, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
names       : band1, band2, band3, band4
min values  :    43,    20,    10,     1
max values  :    79,    44,    69,    25
```

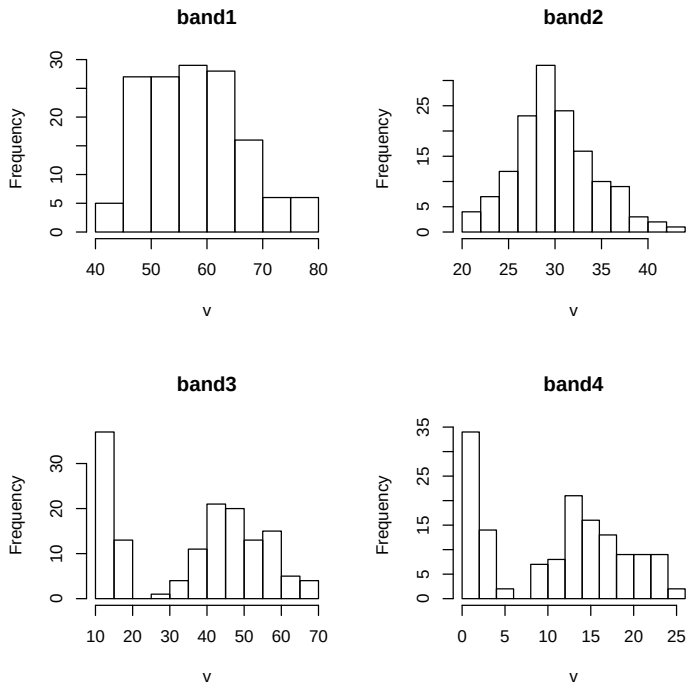
Visualice las cuatro bandas de la imagen usando la siguiente instruccion:

```
> # ploteo de bandas individuales usando la tabla de color default
> plot(toy)
```



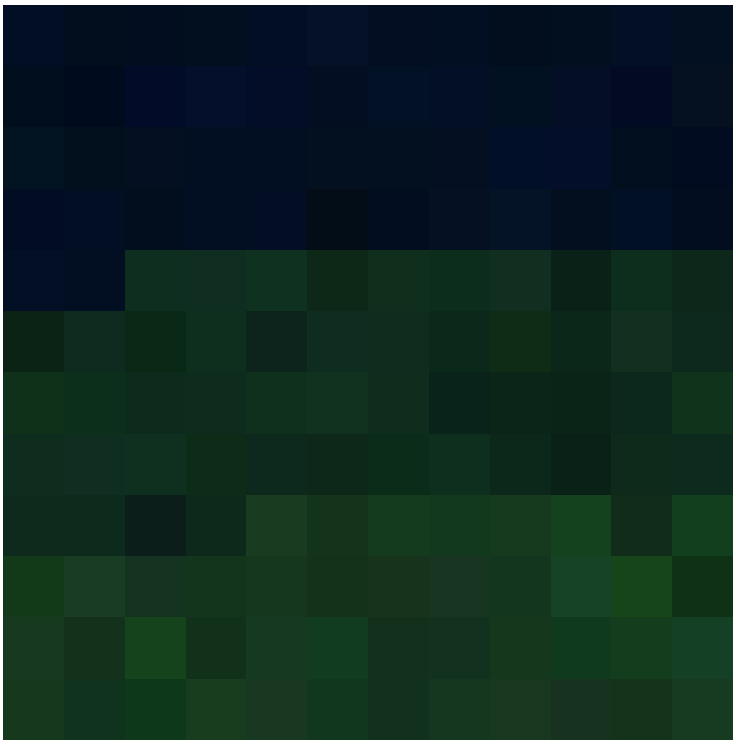
Visualice los histogramas de cada banda de la imagen usando la siguiente instruccion:

```
> hist(toy)
```



Visualice una composicion a color de la imagen usando la siguiente instruccion:

```
> # definicion de la ventana de graficos
> par(mfrow=c(1,1))
> # composicion a color RGB432
> plotRGB(toy, r=4, g=3, b=2)
```



Las estadísticas de la imagen se pueden obtener usando las siguientes instrucciones:

```
> # estadisticas unibanda
> resumen <- summary(toy)
> resumen
```

```
        band1 band2 band3 band4
Min.       43 20.00    10     1
1st Qu.    51 28.00    15     3
Median     57 30.00    42    13
3rd Qu.    64 33.25    51    18
Max.       79 44.00    69    25
NA's        0  0.00     0     0
```

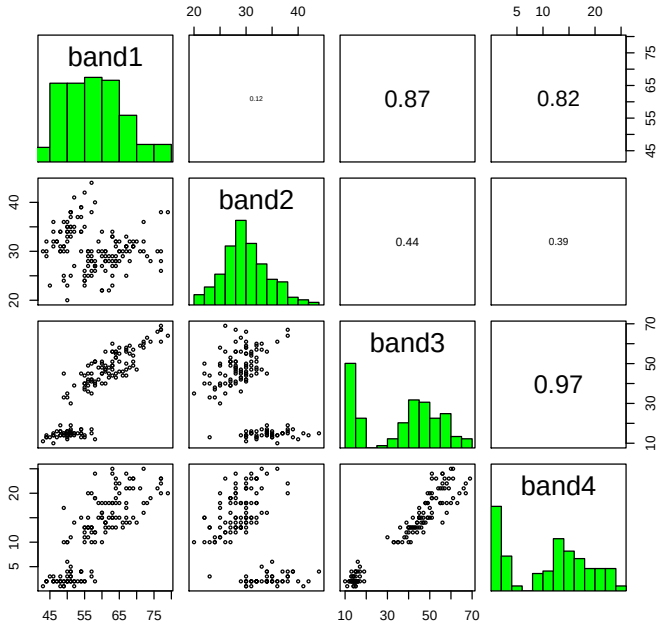
```
> # matriz de covarianza
> covar <- cov(as.matrix(toy))
> covar
```

```
           band1      band2    band3     band4
band1  69.830420  -4.409091 129.0871  51.37675
band2  -4.409091  19.447552 -34.4120 -12.78147
band3 129.087121 -34.412005 314.5361 128.63097
band4  51.376748 -12.781469 128.6310  56.46460
```

```
> # matriz de correlacion
> corr <- cor(as.matrix(toy))
> corr

           band1      band2      band3      band4
band1  1.0000000 -0.1196449  0.8710144  0.8181938
band2 -0.1196449  1.0000000 -0.4399894 -0.3857093
band3  0.8710144 -0.4399894  1.0000000  0.9652109
band4  0.8181938 -0.3857093  0.9652109  1.0000000
```

Visualice ploteos en dispersión entre cada par de bandas usando la siguiente instruccion:

```
> pairs(toy)
```



3. Datos de referencia

La zona cubierta por la imagen es una zona rural en la cual existen tres clases de cobertura vegetal: pasto (1), bosque (2) y cultivo (3). En los dias pasados se realizó una visita de campo en la cual se obtuvo, para cada pixel de la imagen, la clase de cobertura existente. Luego, en la oficina, se elaboró un mapa raster con las clases de cobertura existentes. Dicho mapa se puede descargar del siguiente

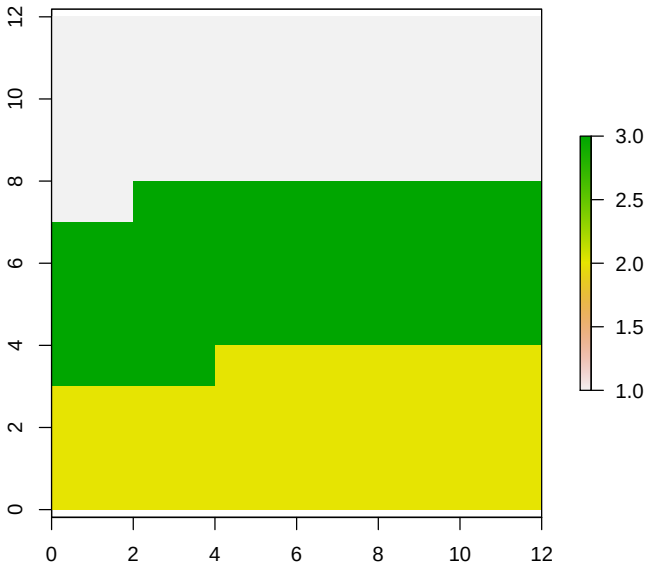
enlace:
https://docs.google.com/file/d/0BzEwvK1H17qeVEJZTk1DWjEwMGM/edit?usp=sharing

Las siguientes instrucciones permiten leer el mapa raster y convertirlo a una estructura vectorial con geometría de puntos o de polígonos:

```
> terreno <- raster("terreno.tif")
> terreno

class       : RasterLayer
dimensions  : 12, 12, 144  (nrow, ncol, ncell)
resolution  : 1, 1  (x, y)
extent      : 0, 12, 0, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
data source : /home/ivan/ud/pdi avanzado/R/terreno.tif
names       : terreno
values      : 1, 3  (min, max)

> # conversion de raster a puntos
> pterreno <- rasterToPoints(terreno)
> # conversion de raster a poligonos
> spterreno <- rasterToPolygons(terreno)
```

La visualización del mapa raster se puede obtener utilizando la siguiente expresión:

```
> plot(terreno)
```



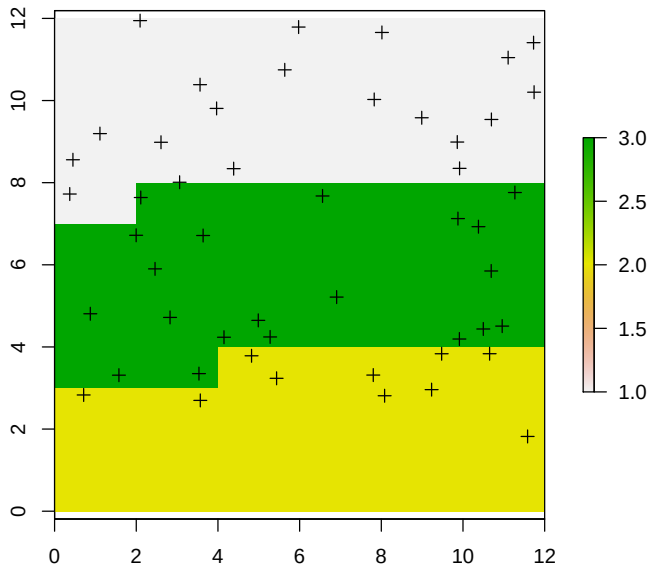
4. Muestra de entrenamiento

La creación de la muestra de entrenamiento se puede realizar usando los siguientes comandos:

```
> # definicion aleatoria de una muestra de 150 puntos
> set.seed <- 7435
> p.sample <- spsample(spterreno,150,"random")
> # seleccion aleatoria de 50 puntos para entrenamiento
> train <- sort(sample(1:150, floor(50)))
> p.train <- p.sample[train,]
> # obtencion de clases existentes en los sitios de muestreo
> temp1 <- overlay(spterreno,p.train)
> # creacion de la respuesta del modelo
> resp <- temp1$value
> # Obtencion de ND de la imagen en cada punto de muestreo
> trainvals <- extract(toy, p.train)
> #trainvals
> # Adicion de clases de cobertura a los puntos de entrenamiento
> sp.train = SpatialPointsDataFrame(p.train, temp1)
> #sp.train
```

La visualización de los puntos de entrenamiento superpuestos sobre el raster de terreno se puede realizar usando la siguiente expresión:

```
> plot(terreno)
> plot(sp.train, add=TRUE)
```



5. Análisis de separabilidad

Para realizar el análisis visual de separabilidad es conveniente crear un nuevo raster que integre las cuatro bandas espectrales y la clase de cobertura existente en el terreno. Las siguientes instrucciones permiten obtener ese raster:

```
> newtoy <- stack(toy, terreno)
> names(newtoy) <- c("band1","band2","band3","band4","clase")
> newtoy

class       : RasterStack
dimensions  : 12, 12, 144, 5  (nrow, ncol, ncell, nlayers)
resolution  : 1, 1  (x, y)
extent      : 0, 12, 0, 12  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
names       : band1, band2, band3, band4, clase
```

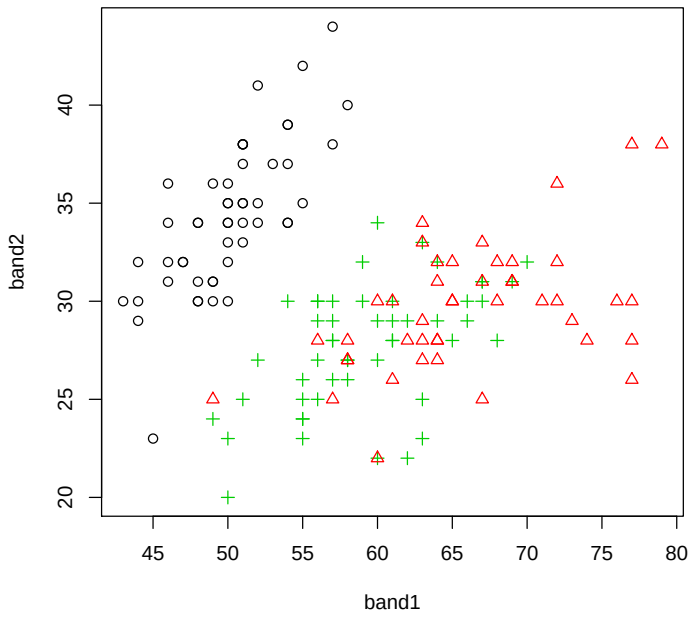
```
min values  :    43,    20,    10,     1,     1
max values  :    79,    44,    69,    25,     3
```

```
> ntoy <- as.data.frame(newtoy)
> str(ntoy)
```

```
'data.frame':        144 obs. of  5 variables:
 $ band1: num  51 49 47 46 50 54 46 50 44 49 ...
 $ band2: num  35 30 32 31 36 39 34 34 29 31 ...
 $ band3: num  14 14 13 15 14 17 14 15 14 15 ...
 $ band4: num  2 2 2 2 2 4 3 2 2 1 ...
 $ clase: num  1 1 1 1 1 1 1 1 1 1 ...
```

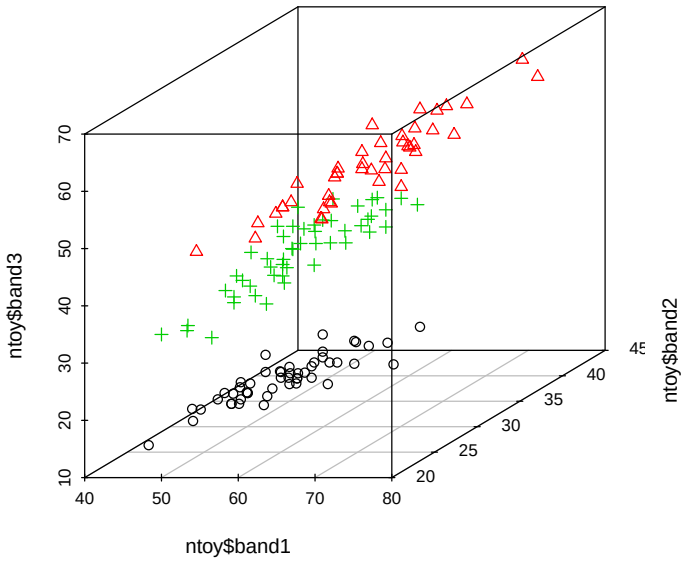
La siguiente instrucción permite obtener un gráfico de separabilidad de las diferentes clases en función de la respuesta espectral de las bandas 1 y 2:

```
> with(ntoy, plot(band1, band2, col=clase, pch=as.numeric(clase)))
```



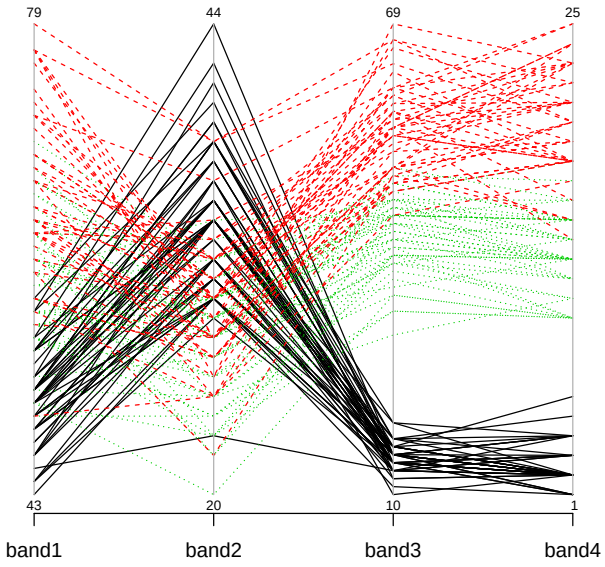
Observe que, de acuerdo a la figura anterior, existe confusión espectral entre dos de las clases de interes. Por eso es recomendable tener un ploteo de dispersión usando tres bandas, en lugar de dos. Para el efecto, use la siguiente instrucción:

```
> library(scatterplot3d)
> scatterplot3d(ntoy$band1, ntoy$band2, ntoy$band3, color= ntoy$clase, pch=ntoy$clase)
```



Otra alternativa útil de visualización de las características espectrales de las clases es mediante un perfil espectral de cada uno de los pixeles, simbolizando cada clase con un color diferente. Para el efecto se puede aplicar la siguiente expresión:

```
> library(MASS)
> parcoord(ntoy[1:4], col = ntoy$clase , lty = ntoy$clase, var.label = TRUE)
```



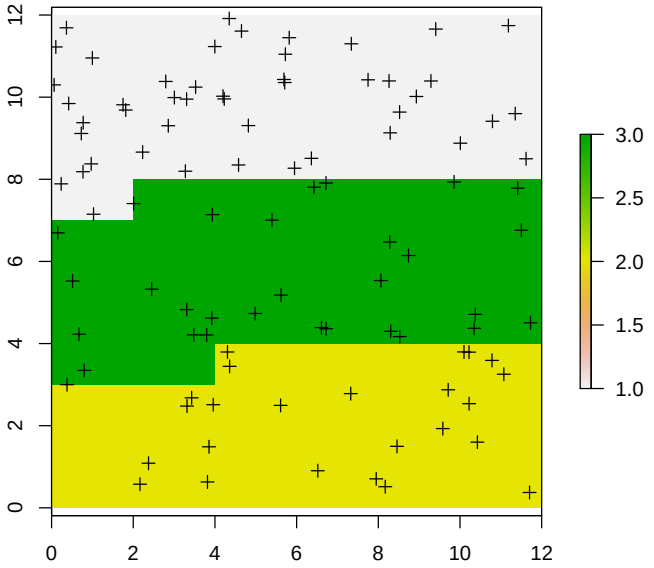
6. Muestra de validación

La creacion de la muestra de validación se puede realizar usando la siguiente expresión:

```
> # definicion aleatoria de 100 puntos de validacion
> # seleccionados tomando, en la muestra de 150 puntos, previamente obtenida,
> # aquellos puntos que no corresponden a sitios de entrenamiento
> p.test <- p.sample[-train,]
> # obtencion de clases existentes en los sitios de validacion
> temp <- overlay(spterreno,p.test)
> # creacion de la respuesta del modelo
> response <- temp$value
> # Obtencion de ND de la imagen en cada punto de muestreo
> testvals <- extract(toy, p.test)
> #testvals
> # Adicion de clases de cobertura a los puntos de entrenamiento
> sp.test = SpatialPointsDataFrame(p.test, temp)
> #sp.test
```

La visualización de los puntos de entrenamiento superpuestos sobre el raster de terreno se puede realizar usando la siguiente expresión:

```
> plot(terreno)
> plot(sp.test, add=TRUE)
```



7. Clasificación de la cobertura del suelo usando Distancia Mahalanobis

El metodo de distancia Mahalanobis es bastante conocido en la literatura de percepción remota. Se puede entender como un clasificador de 'máxima probabilidad'. La distancia de Mahalanobis es una medida estadística de distancia que se basa fuertemente en las correlaciones entre las variables involucradas.

La distancia de Mahalanobis entre un pixel $x = (x_1, x_2, ...x_n)^T$ y una clase de valores con media muestral $\mu = (\mu_1, \mu_2, ..., \mu_n)^T$ se define como:

$$D_M(x) = \sqrt{(x-\mu)^T COV^{-1}(x-\mu)}$$

siendo COV^{-1} la inversa de la matriz de covarianza.

Un pixel se clasifica en aquella clase cuyo centro de clase (es decir, el valor promedio de las muestras de entrenamiento que pertenecen a dicha clase) esté mas cerca de dicho pixel en términos de la distancia Mahalanobis.

Las siguientes instrucciones permiten obtener la matriz de covarianza de cada una de las clases existentes:

```
> #  conversion de tipos de objetos
> train <- cbind(trainvals, resp)
> df.train <- as.data.frame(train)
> # Clase 1: Pasto
> # recuperacion de las 4 bandas para todos los pixeles de esta clase
> pasto <- df.train[df.train$resp==1,1:4]
> # valor medio de clase 1
> mean1 <- colMeans( pasto )
> mean1

band1 band2 band3 band4
50.15 34.30 14.75  2.55

> # matriz de covarianza de clase 1
> var1<-var( pasto )
> var1

          band1      band2     band3     band4
band1 10.976316 10.6368421 1.2500000 1.0184211
band2 10.636842 16.0105263 1.3421053 0.7736842
band3  1.250000  1.3421053 1.5657895 0.4078947
band4  1.018421  0.7736842 0.4078947 0.8921053

> #
> # Clase 2: Bosque
> bosque <- df.train[df.train$resp==2,1:4]
> # valor medio de clase 2
> mean2 <- colMeans( bosque )
> mean2

band1 band2 band3 band4
 66.1  30.2  57.5  19.7

> # matriz de covarianza clase 2
> var2<-var( bosque )
> var2

          band1     band2    band3     band4
band1 85.211111 32.866667 60.83333  9.366667
band2 32.866667 22.844444 25.55556  7.177778
band3 60.833333 25.555556 48.72222 10.611111
band4  9.366667  7.177778 10.61111  5.788889

> #
> # Clase 3: Cultivo
> cultivo <- df.train[df.train$resp==3,1:4]
> # valor medio de clase 3
> mean3 <- colMeans( cultivo )
> mean3
```

```
band1 band2 band3 band4
58.30 28.35 42.20 13.55
```

```
> # matriz de covarianza clase 3
> var3<-var(cultivo)
> var3
```

```
          band1    band2     band3    band4
band1 29.589474 9.100000 17.831579 4.826316
band2  9.100000 5.923684  8.557895 3.376316
band3 17.831579 8.557895 17.536842 6.410526
band4  4.826316 3.376316  6.410526 3.207895
```

Las siguientes instrucciones permiten obtener un vector que contiene la distancia Mahalanobis de cada pixel a cada una de las clases:

```
> #
> val <- getValues(toy)
> nval <- nrow(val)
> seq <- 1: nval
> dm <- data.frame(dm1=numeric(),dm2=numeric(),dm3=numeric(),clase=numeric())
> #
> for (i in seq)
+ {
+ new<-c(val[i,1],val[i,2],val[i,3],val[i,4])
+ dm[i,1] <- mahalanobis(new, mean1, var1)
+ dm[i,2] <- mahalanobis(new, mean2, var2)
+ dm[i,3] <- mahalanobis(new, mean3, var3)
+ }
> #dm
```

Las siguientes instrucciones permiten obtener la clase mas cercana a cada uno de los pixeles de la imagen:

```
> for (i in seq)
+ {
+ dm[i,4] <- which.min(dm[i,])
+ }
> dm[c(1:5,50:55,120:125),]
```

```
             dm1        dm2        dm3 clase
1      0.8571630 271.329896 312.741010     1
2      2.3860901 233.993626 205.142970     1
3      2.4057878 238.200842 252.617262     1
4      2.1233662 196.840010 206.982315     1
5      1.1502262 266.049164 335.238919     1
50     0.8432394 253.739068 265.685443     1
51   779.5519591  41.319409   6.456891     3
```

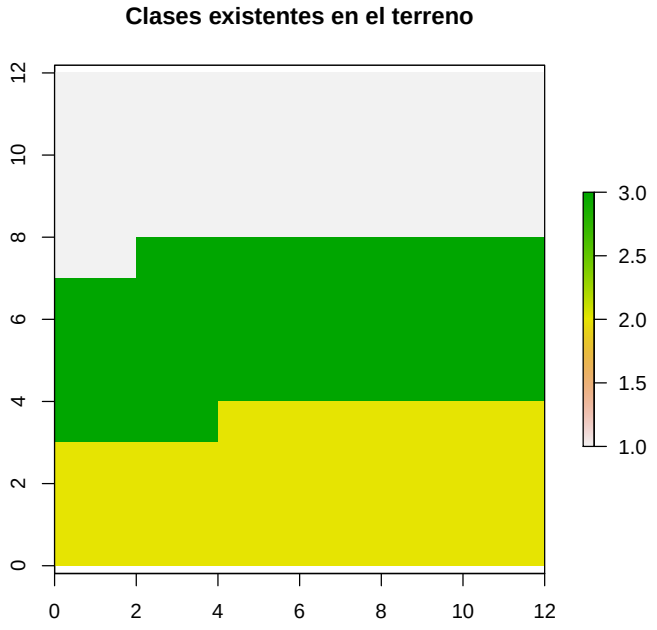
```
52   682.2080775  31.479371   4.289352     3
53   875.0944757  25.592307   4.457043     3
54   513.1488745  20.302291  12.023673     3
55   752.4189947  28.032215   5.989533     3
120  953.4314878   4.926042  64.411595     2
121 1388.4029702  16.096046  49.489052     2
122  956.3467447   4.557744  34.741806     2
123 2019.2991388   7.617391 128.563540     2
124  920.6859931  14.833726  25.934763     2
125 1330.3871180   2.587669  23.313295     2
```

La siguiente instruccion permite crear un raster con las clases de cobertura obtenidas usando el metodo Distancia de Mahalanobis (DM):

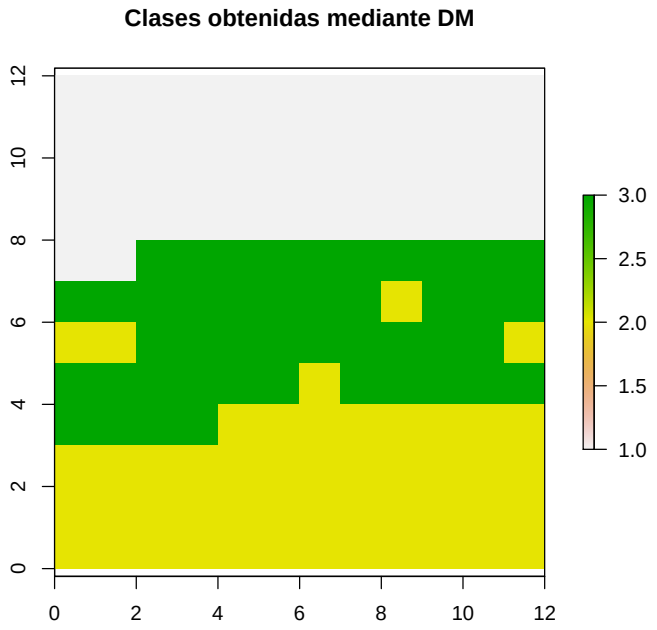
```
> pred_dm <- toy[[1]]
> pred_dm[] <- dm[,4]
```

Las siguientes instrucciones permiten visualizar el raster de clases existentes en el terreno y el raster de clases obtenidas mediante DM:

```
> plot(terreno, main="Clases existentes en el terreno")
```




```
> plot(pred_dm, main="Clases obtenidas mediante DM")
```



La evaluación de la exactitud temática de la clasificación obtenida se puede realizar usando las siguientes instrucciones:

```
> # Evaluacion de exactitud tematica
> # obtencion de clase predicha en cada punto de validacion
> temp$clas <- extract(pred_dm, p.test)
> # derivacion de la matriz de confusion
> library(mda)
> conf <- confusion(temp$value, temp$clas)
> # impresion de la matriz de confusion
> conf

          true
predicted  1  2  3
        1 46  0  0
        2  0 25  0
        3  0  5 24
attr(,"error")
[1] 0.05

> # porcentaje correctamente clasificado
> pcc <- 100 * sum(diag(conf))/nrow(temp)
> pcc
```

```
[1] 95

> #obtencion del valor kappa
> library(vcd)
> k1 = Kappa(conf)$Unweighted[[1]]
> #
> k1

[1] 0.9223361
```

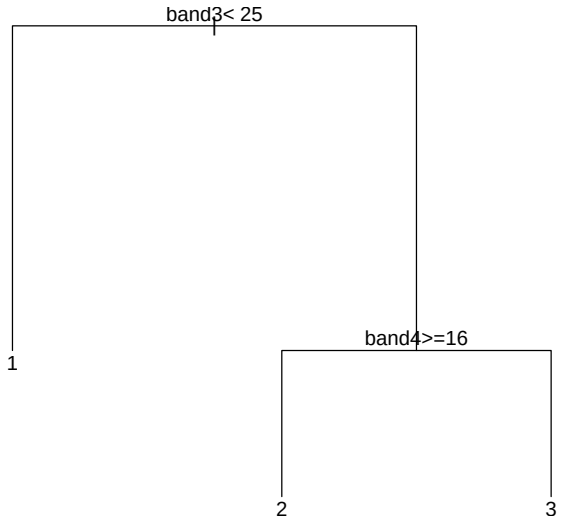
8. Clasificación de la cobertura del suelo usando árboles de decisión

Los metodos conocidos como *árboles de decisión* construyen una serie de reglas, basadas en los valores de los atributos de la muestra de entrenamiento, para asignar una clase a cada uno de los individuos de interes. Existen diversos algoritmos que implementan arboles de decision, tambien conocidos como 'classification and regression trees', entre ellos los siguientes: $ID3$, $C4{,}5$, $C5{,}0$.

En R existen diversas librerías que implementan árboles de decisión; las más conocidas son *tree* y *rpart*.

La siguiente instrucción usa la librería *rpart* para realizar la inducción de un arbol para clasificar cobertura del suelo a partir de las muestras de entrenamiento que se están usando en este tutorial:

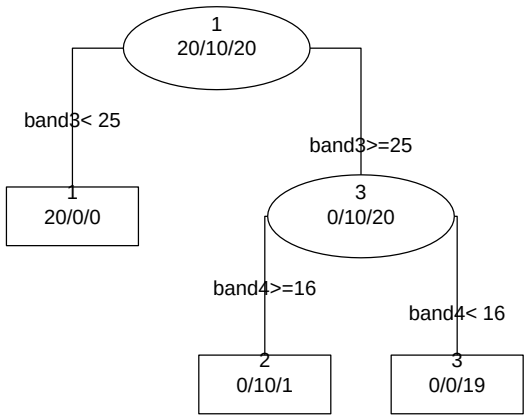
```
> library(rpart)
> # creacion de un data frame con los datos requeridos
> train <- cbind(trainvals, resp)
> df.train <- as.data.frame(train)
> # creacion de un modelo de decision sin indicar parametros
> # por default, se usa la metrica Gini
> rp1 <- rpart(resp ~ .,dat=df.train, method="class")
> # ploteo default del arbol de clasificacion
> plot(rp1)
> text(rp1)
```




```
> # creacion de un modelo de decision aumentando la complejidad del arbol
> # para el efecto se cambia el valor del parametro cp que, por default, es 0.01
> rp2 <- rpart(resp ~ .,dat=df.train, method="class", control=rpart.control(cp=0.005))
> # detalles del arbol
> summary(rp2)

Call:
rpart(formula = resp ~ ., data = df.train, method = "class",
    control = rpart.control(cp = 0.005))
  n= 50

         CP nsplit  rel error    xerror       xstd
1 0.6666667      0 1.00000000 1.3000000 0.09763879
2 0.3000000      1 0.33333333 0.3333333 0.09428090
3 0.0050000      2 0.03333333 0.2333333 0.08178563

Variable importance
band4 band3 band1 band2
   34    33    19    14

Node number 1: 50 observations,    complexity param=0.6666667
  predicted class=1  expected loss=0.6  P(node) =1
```

```
    class counts:    20    10    20
   probabilities: 0.400 0.200 0.400
  left son=2 (20 obs) right son=3 (30 obs)
  Primary splits:
      band3 < 25   to the left,  improve=18.666670, (0 missing)
      band4 < 7.5  to the left,  improve=18.666670, (0 missing)
      band1 < 55.5 to the left,  improve= 9.760000, (0 missing)
      band2 < 32.5 to the right, improve= 7.772059, (0 missing)
  Surrogate splits:
      band4 < 7.5  to the left,  agree=1.00, adj=1.00, (0 split)
      band1 < 53.5 to the left,  agree=0.86, adj=0.65, (0 split)
      band2 < 32.5 to the right, agree=0.80, adj=0.50, (0 split)

Node number 2: 20 observations
  predicted class=1  expected loss=0  P(node) =0.4
    class counts:    20     0     0
   probabilities: 1.000 0.000 0.000

Node number 3: 30 observations,    complexity param=0.3
  predicted class=3  expected loss=0.3333333  P(node) =0.6
    class counts:     0    10    20
   probabilities: 0.000 0.333 0.667
  left son=6 (11 obs) right son=7 (19 obs)
  Primary splits:
      band4 < 16   to the right, improve=11.5151500, (0 missing)
      band3 < 49.5 to the right, improve=11.4285700, (0 missing)
      band1 < 62.5 to the right, improve= 4.4444440, (0 missing)
      band2 < 29.5 to the right, improve= 0.2777778, (0 missing)
  Surrogate splits:
      band3 < 47.5 to the right, agree=0.967, adj=0.909, (0 split)
      band1 < 62.5 to the right, agree=0.767, adj=0.364, (0 split)
      band2 < 31.5 to the right, agree=0.733, adj=0.273, (0 split)

Node number 6: 11 observations
  predicted class=2  expected loss=0.09090909  P(node) =0.22
    class counts:     0    10     1
   probabilities: 0.000 0.909 0.091

Node number 7: 19 observations
  predicted class=3  expected loss=0  P(node) =0.38
    class counts:     0     0    19
   probabilities: 0.000 0.000 1.000

> # un mejor ploteo del arbol de clasificacion
> plot(rp2,
+      uniform=TRUE,
```

```
+      compress=TRUE,
+      margin = .2)
> text(rp2,
+      use.n=TRUE,
+      all = TRUE,
+      fancy = TRUE)
```



La siguiente instrucción permite usar el modelo *rpart* para predecir la clase de cobertura en toda la imagen:

```
> # prediccion
> dfval <- as.data.frame(getValues(toy))
> clasepred <- predict(rp2,dfval,type="class")
> clasepred
```

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  1   1   1   1   1   1   1   1   1   1   3   3   3   3   3   3   2   3   3   3
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   2   3   3
```

```
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
  3   3   3   2   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   3
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  2   2   2   2   2   2   2   2   2   2   2   2   2   3   3   2   2   2   2   2
141 142 143 144
  2   2   2   2
Levels: 1 2 3
```

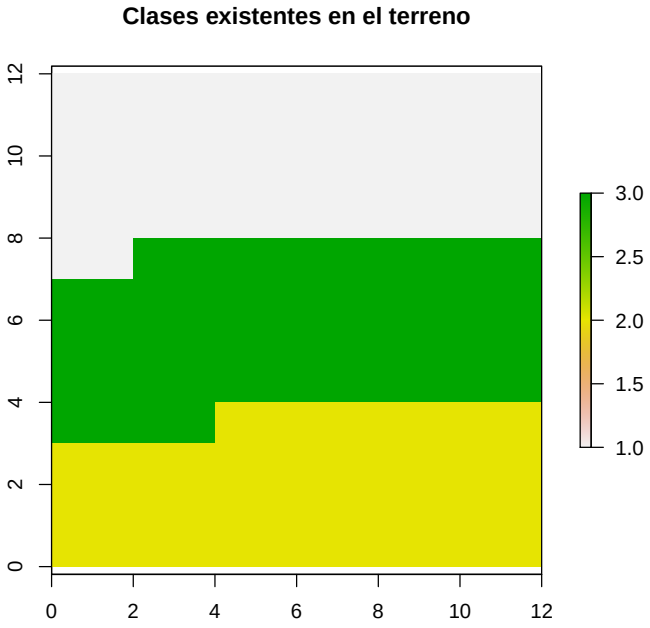
La siguiente instrucción permite crear un raster con las clases de cobertura obtenidas por el algoritmo *rpart*:

```
> pred_dt <- toy[[1]]
> pred_dt[] <- as.numeric(clasepred)
> pred_dt[]
```

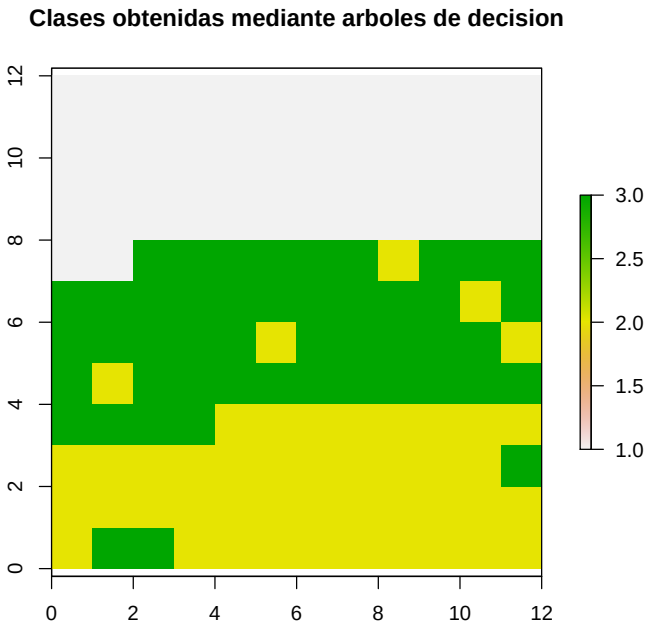
```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
 [75] 3 3 3 2 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2
```

Las siguientes instrucciones permiten visualizar el raster de clases existentes en el terreno y el raster de clases obtenidas mediante arboles de decision:

```
> plot(terreno, main="Clases existentes en el terreno")
```




```
> plot(pred_dt, main="Clases obtenidas mediante arboles de decision")
```


Clases obtenidas mediante arboles de decision
0
2
4
6
8
10
12
1.0
1.5
2.0
2.5
3.0

Para realizar la evaluacion de la exactitud tematica de la clasificacion obtenida se pueden usar las siguientes instrucciones:

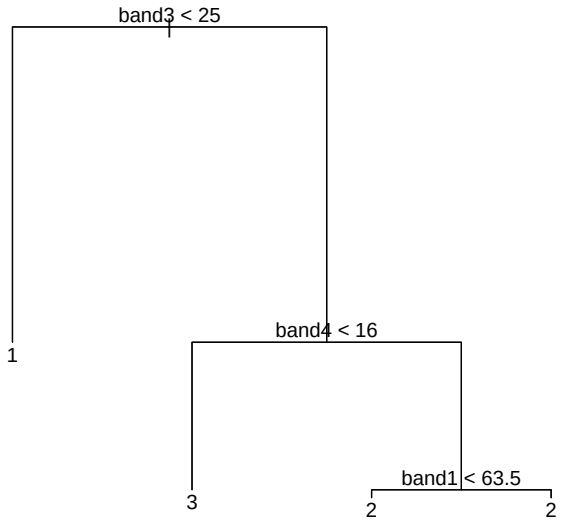
```
> # Evaluacion de exactitud tematica
> # obtencion de clase predicha en cada punto de validacion
> temp$clas <- extract(pred_dt, p.test)
> # derivacion de la matriz de confusion
> conf3 <- confusion(temp$value, temp$clas)
> # impresion de la matriz de confusion
> conf3

         true
predicted  1  2  3
        1 46  0  0
        2  0 24  1
        3  0  1 28
attr(,"error")
[1] 0.02

> # porcentaje correctamente clasificado
> pcc3 <- 100 * sum(diag(conf3))/nrow(temp)
> pcc3

[1] 98

> # obtencion del valor kappa
> k3 = Kappa(conf3)$Unweighted[[1]]
> #
> k3

[1] 0.9688376
```

Otra alternativa para crear arboles de clasificacion en R es usar la libreria *tree*. La siguiente expresion permite realizar la induccion de un arbol de clasificacion usando los datos de este tutorial:

```
> library(tree)
> tree1 <- tree(as.factor(resp) ~ .,data=df.train)
> # descripcion del arbol de clasificacion
> tree1

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 50 105.500 1 ( 0.40000 0.20000 0.40000 )
   2) band3 < 25 20   0.000 1 ( 1.00000 0.00000 0.00000 ) *
   3) band3 > 25 30  38.190 3 ( 0.00000 0.33333 0.66667 )
     6) band4 < 16 19   0.000 3 ( 0.00000 0.00000 1.00000 ) *
```

```
     7) band4 > 16 11   6.702 2 ( 0.00000 0.90909 0.09091 )
      14) band1 < 63.5 5   5.004 2 ( 0.00000 0.80000 0.20000 ) *
      15) band1 > 63.5 6   0.000 2 ( 0.00000 1.00000 0.00000 ) *
```

```
> # ploteo default del arbol de clasificacion
> plot(tree1)
> text(tree1)
```

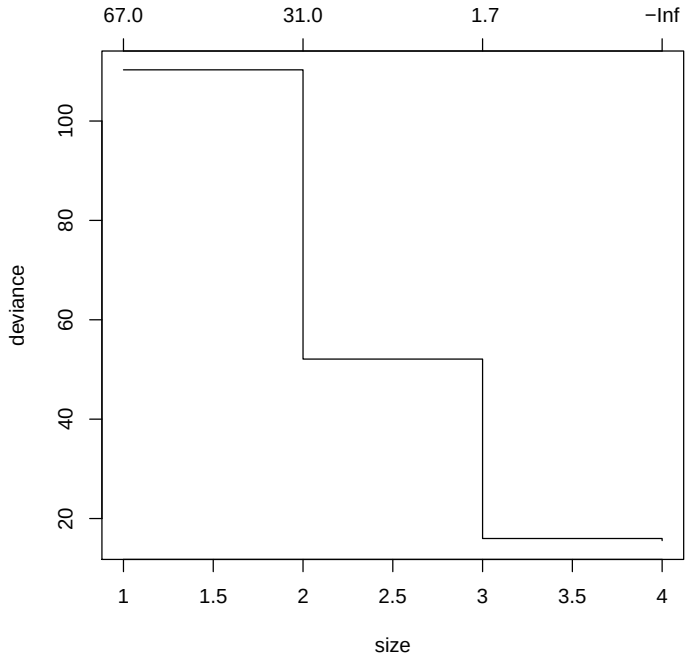


Es posible examinar el error de clasificacion usando la siguiente instruccion:

```
> summary(tree1)

Classification tree:
tree(formula = as.factor(resp) ~ ., data = df.train)
Variables actually used in tree construction:
[1] "band3" "band4" "band1"
Number of terminal nodes:  4
Residual mean deviance:  0.1088 = 5.004 / 46
Misclassification error rate: 0.02 = 1 / 50
```

Observe que el error de clasificacion es pequeño. Sin embargo, para evitar 'overfitting' se puede realizar validacion cruzada:

```
> cvtree1 <- cv.tree(tree1, FUN=prune.tree)
> plot(cvtree1)
```

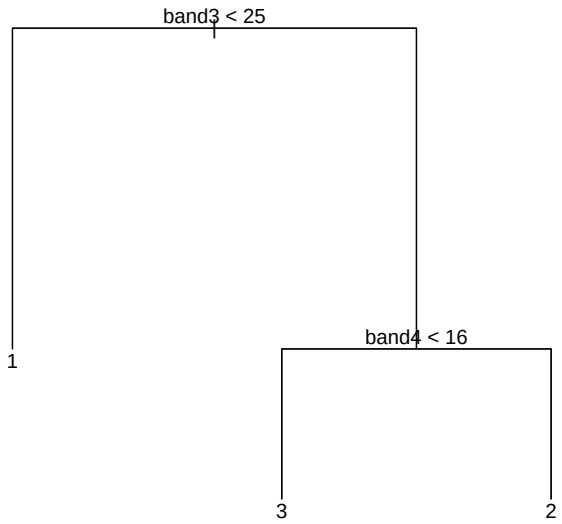


Observe que la desviación (medida de impureza) es pequeña cuando el número de nodos terminales es mayor que tres. Se puede realizar una poda del arbol de manera que el numero de nodos terminales sea de 3:

```
> poda1 <- prune.tree(tree1, best=3)
> poda1

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 50 105.500 1 ( 0.40000 0.20000 0.40000 )
  2) band3 < 25 20   0.000 1 ( 1.00000 0.00000 0.00000 ) *
  3) band3 > 25 30  38.190 3 ( 0.00000 0.33333 0.66667 )
    6) band4 < 16 19   0.000 3 ( 0.00000 0.00000 1.00000 ) *
    7) band4 > 16 11   6.702 2 ( 0.00000 0.90909 0.09091 ) *

> summary(poda1)

Classification tree:
snip.tree(tree = tree1, nodes = 7L)
Variables actually used in tree construction:
[1] "band3" "band4"
Number of terminal nodes:  3
Residual mean deviance:  0.1426 = 6.702 / 47
Misclassification error rate: 0.02 = 1 / 50
```

Observe si el error de clasificación es mas alto que el obtenido usando el árbol original. Enseguida se puede realizar el ploteo del árbol podado:

```
> plot(poda1)
> text(poda1)
```



El modelo del árbol podado se puede utilizar para predecir la clase de cobertura en toda la imagen:

```
> # prediccion
> clasepred <- predict(poda1,dfval, type="class")
> clasepred
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
 [75] 3 3 3 2 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2
Levels: 1 2 3
```

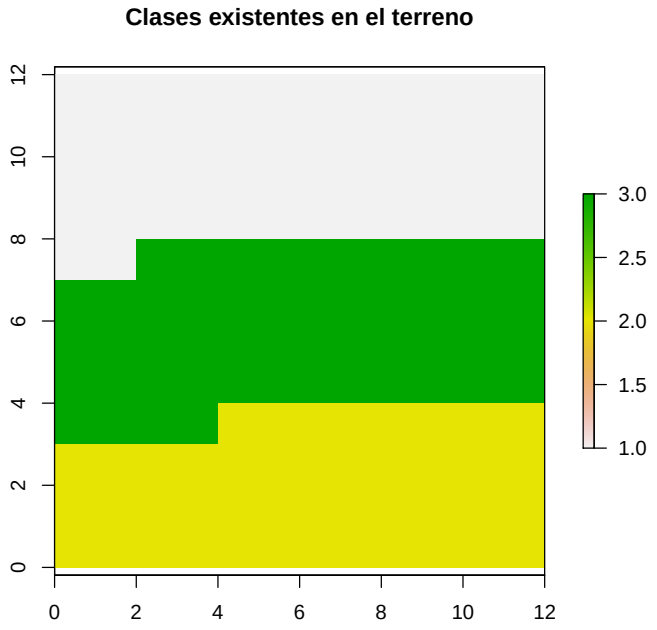
El raster con las clases de cobertura obtenidas se obtiene con la siguiente expresión:

```
> pred_tree <- toy[[1]]
> pred_tree[] <- as.numeric(clasepred)
> pred_tree[]
```

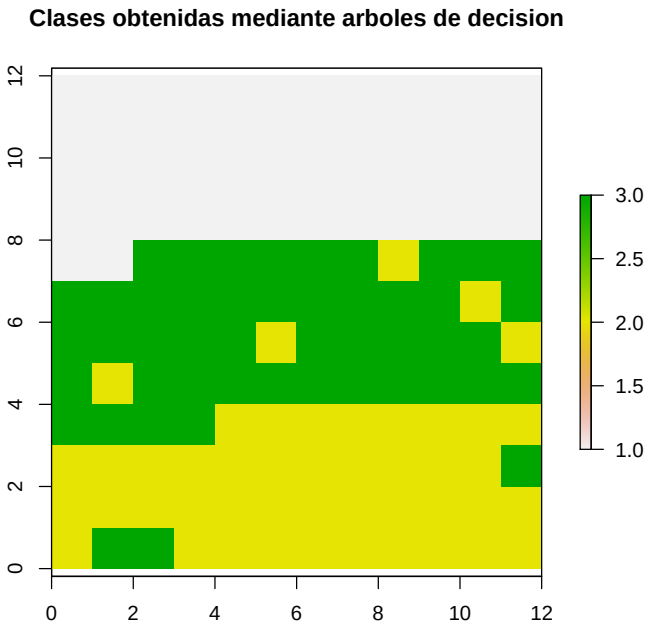
```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
 [75] 3 3 3 2 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2
```

La visualización del raster de clases existentes en el terreno y el raster de clases obtenidas se puede obtener así:

```
> plot(terreno, main="Clases existentes en el terreno")
```




```
> plot(pred_tree, main="Clases obtenidas mediante arboles de decision")
```



Enseguida se realiza la evaluación de la exactitud temática:

```
> # Evaluacion de exactitud tematica
> # obtencion de clase predicha en cada punto de validacion
> temp$clas <- extract(pred_tree, p.test)
> # derivacion de la matriz de confusion
> conf4 <- confusion(temp$value, temp$clas)
> # impresion de la matriz de confusion
> conf4

          true
predicted  1  2  3
        1 46  0  0
        2  0 24  1
        3  0  1 28
attr(,"error")
[1] 0.02

> # porcentaje correctamente clasificado
> pcc4 <- 100 * sum(diag(conf4))/nrow(temp)
> pcc4

[1] 98
```

```
> # obtencion del valor kappa
> k4 = Kappa(conf4)$Unweighted[[1]]
> #
> k4

[1] 0.9688376
```

9. Clasificación de la cobertura del suelo usando máquinas de soporte vectorial

El metodo denominado *Support Vector Machine* (SVM) busca encontrar un hiperplano de separacion optima entre las clases existentes utilizando una funcion kernel, basada en el producto punto entre vectores, que permita obtener los mismos resultados que una transformacion de las muestras de entrenamiento en un espacio de mayor dimensionalidad que el espacio original.

En este tutorial, se utilizara la libreria *kernlab* que permite aplicar eficientemente el algoritmo SVM.

La siguiente instruccion permite entrenar el algoritmo SVM a partir de la muestra de entrenamiento obtenida anteriormente usando unos parametros arbitrarios:

```
> # cargue de la libreria requerida
> library(kernlab)
> # entrenamiento de SVM
> svp <- ksvm(trainvals,resp, type="C-svc", kernel='rbf',kpar=list(sigma=5.0),C=12.5)
> # resumen general
> svp

Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 12.5

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  5

Number of Support Vectors : 42

Objective Function Value : -8.0913 -8.5097 -8.6476
Training error : 0

> # vectores de soporte
> alpha(svp)

[[1]]
 [1] 0.36456035 0.98485726 0.94782691 0.73933375 0.73947463 0.48966957
```

```
 [7] 0.75608153 0.70427414 0.67213954 0.29464407 1.00846097 0.80332531
[13] 0.50334661 0.70580415 0.66029905 0.87909791 1.00821980 0.51866163
[19] 0.30160166 0.67897137 0.09518044 1.00806030 0.39385538 0.92672284

[[2]]
 [1] 0.686779377 1.036186976 0.704788401 0.755405994 0.777404648 0.353107982
 [7] 0.514597291 0.795720246 0.740793320 0.706523425 0.310435699 0.428548183
[13] 0.844651383 0.544201576 0.582514173 0.021849332 0.003427867 0.528996683
[19] 0.134228106 0.898589362 0.693821678 0.025590719 0.924764222 0.661695649
[25] 0.462930122 0.545449144 0.317650289 0.553843913 0.322615142 0.099621849
[31] 0.629295055 0.414380741

[[3]]
 [1] 0.68616853 0.38473915 1.00542228 0.69912609 0.87275512 0.79711784
 [7] 0.30769575 0.70791266 0.73567309 1.05199721 0.37331071 0.53976709
[13] 0.65984281 0.01333983 0.22235711 0.73607448 0.90688484 0.01656446
[19] 0.65679412 1.05138270 0.46361421 0.54842803 0.75203397 0.30884941
[25] 1.09281192 0.73948217 0.96436575

> alphaindex(svp)

[[1]]
 [1]  4  5  6  8 10 13 14 16 18 19 20 22 28 30 32 35 38 41 42 44 46 47 49 50

[[2]]
 [1]  1  5  7  9 10 12 13 14 15 18 19 21 22 23 24 25 26 28 29 31 32 33 35 37 40
[26] 41 42 43 45 46 48 49

[[3]]
 [1]  1  4  6  7  8  9 12 15 16 20 21 23 24 25 29 30 31 33 37 38 40 43 44 45 47
[26] 48 50

> b(svp)

[1]  0.008156897 -0.043091858 -0.052032376
```

La siguiente instruccion permite usar el modelo SVM para predecir la clase de cobertura en toda la imagen:

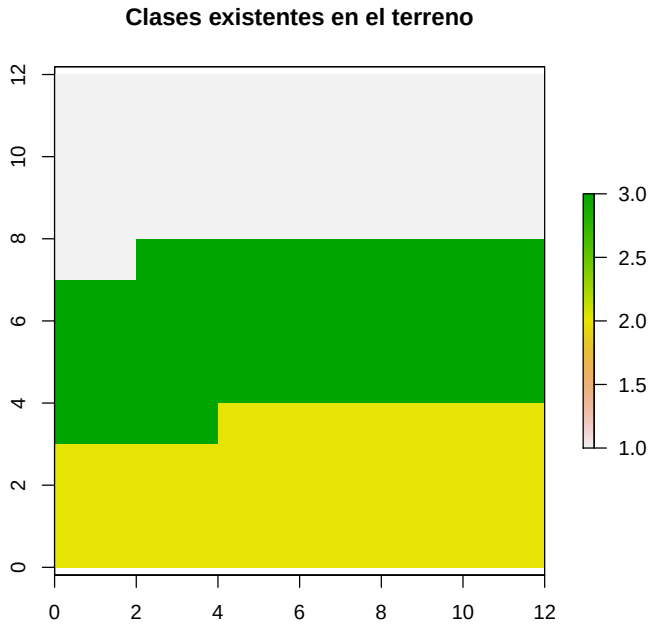
```
> # prediccion
> clasepred <- predict(svp,getValues(toy))
```

La siguiente instruccion permite crear un raster con las clases de cobertura obtenidas por el algoritmo SVM:

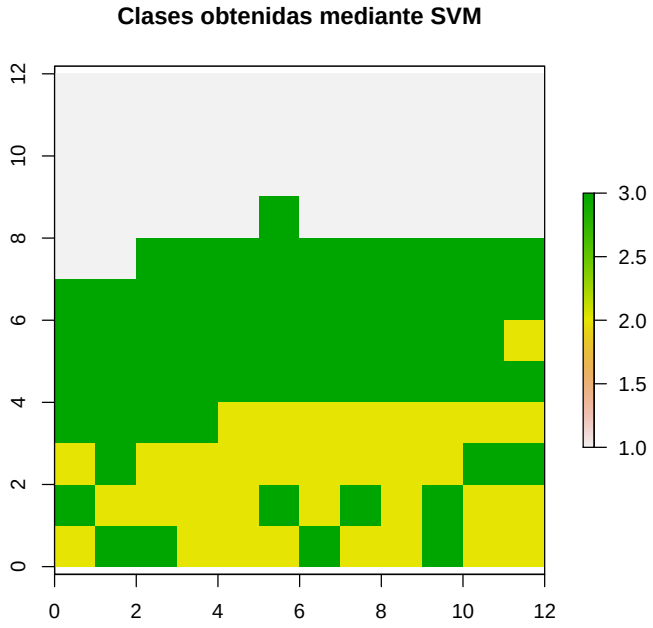
```
> pred_svm <- toy[[1]]
> pred_svm[] <- clasepred
```

Las siguientes instrucciones permiten visualizar el raster de clases existentes en el terreno y el raster de clases obtenidas mediante SVM:

```
> plot(terreno, main="Clases existentes en el terreno")
```




```
> plot(pred_svm, main="Clases obtenidas mediante SVM")
```



Observe que la clases obtenidas no coinciden completamente con la referencia obtenida en el terreno. Esta situacion sugiere la necesidad de 'afinar' los parametros C y *sigma* requeridos por el algoritmo *SVM*.

Tal vez sea mejor encontrar un procedimiento automatizado que permite buscar los parametros optimos del algoritmo. Si esta interesado en ello, tal vez le convenga explorar las funcionalidades de la libreria *e1071* que el autor ha utilizado previamente con muy buenos resultados.

En cualquier caso, es importante realizar la evaluacion de la exactitud tematica de la clasificacion obtenida. Para el efecto, se pueden usar las siguientes instrucciones:

```
> # Evaluacion de exactitud tematica
> # obtencion de clase predicha en cada punto de validacion
> temp$clas <- extract(pred_svm, p.test)
> # derivacion de la matriz de confusion
> conf2 <- confusion(temp$value, temp$clas)
> # impresion de la matriz de confusion
> conf2

          true
predicted  1  2  3
        1 45  0  1
```

```
        2  0 21  4
        3  0  0 29
attr(,"error")
[1] 0.05

> # porcentaje correctamente clasificado
> pcc2 <- 100 * sum(diag(conf2))/nrow(temp)
> pcc2

[1] 95

> # obtencion del valor kappa
> k2 = Kappa(conf2)$Unweighted[[1]]
> #
> k2

[1] 0.9221062
```

10. Que sigue?

El lector puede usar sus propios datos para realizar procedimientos similares a los indicados en este tutorial y obtener informacion tematica de interes. No olvide que las librerias de *R* utilizadas tienen muchas opciones adicionales a las indicadas en este documento. Muchos exitos!!!