Control de calidad de clasificacion

Viacheslav Shalisko

14 de diciembre de 2016

```
verbose = 0

control_parcels_path <- "../Sampling/Sampling_ponits_1km_UTM_circles.shp"

raster_path <- "../Landsat/LC2016_entregables/L2016_clasificado_simplificado.tif"

class_codes <- c(100,103,105,110,115,120,130,135,150,160,161,190,192,195,197,200)
class_legend <- c("Urbano",
                  "Infraestructura",
                  "SVA",
                  "Agr. riego",
                  "Agr. temporal",
                  "Pastizal",
                  "Matorral",
                  "BEsp",
                  "BTC",
                  "BTemp disp.",
                  "BTemp dens.",
                  "Golf",
                  "AVU",
                  "VAS",
                  "BGal",
                  "Agua")
class_col <- c("gray20",
               "gray40",
               "lightgray",
               "yellow",
               "lightyellow",
               "palegreen",
               "orange",
               "coral",
               "indianred3",
               "forestgreen",
               "darkgreen",
               "green",
               "seagreen3",
               "blue",
               "darkgreen",
               "navy")

class_simplification <- c(100,100,105,110,110,120,130,150,150,160,160,190,190,200,160,200)
```

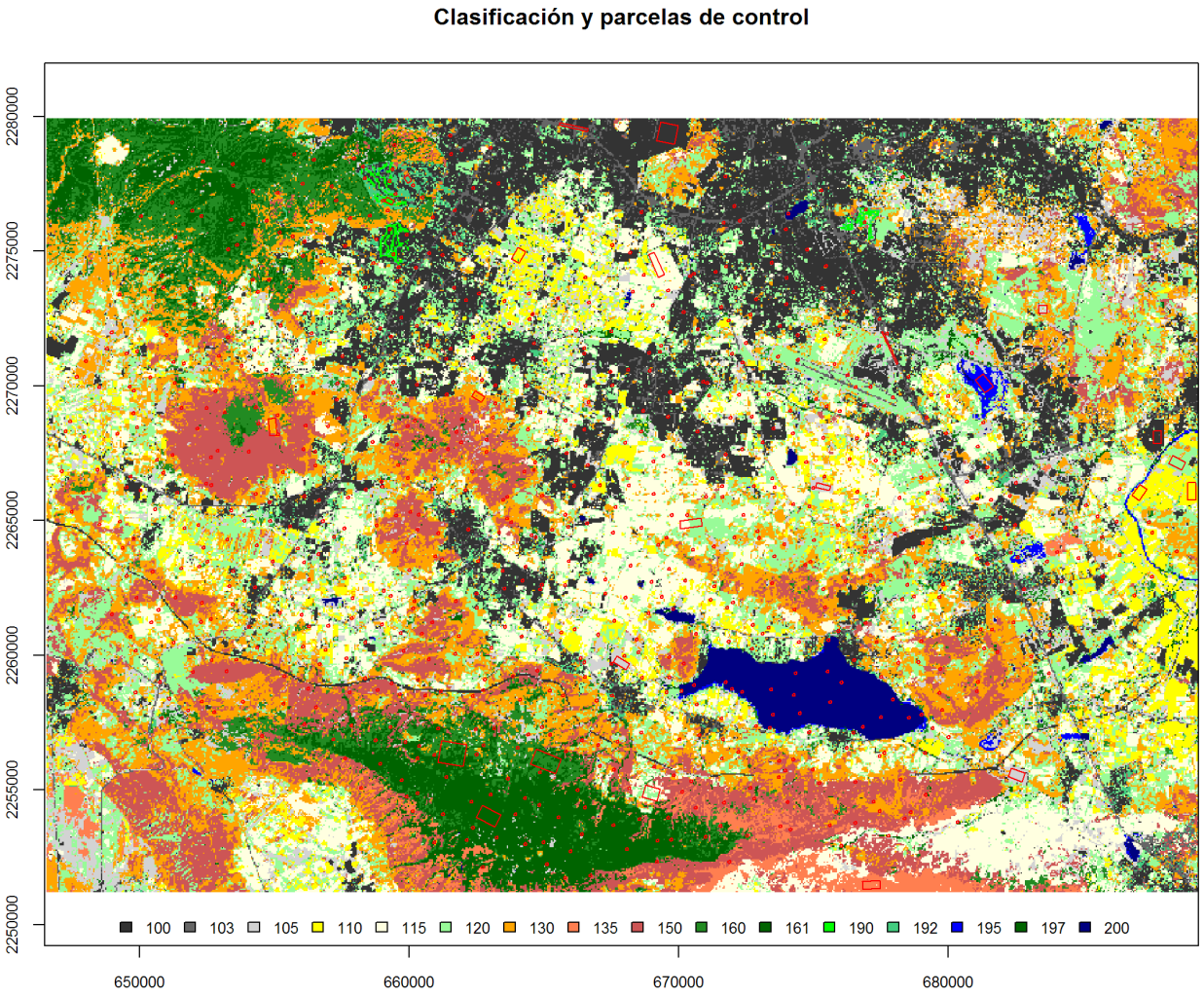
Visualización declos datos espaciales

```
control_parcels <- readShapePoly(control_parcels_path)

classification_raster <- raster(raster_path)
dim(classification_raster)
```

```
## [1]  955 1420    1
```

```
par(cex.axis = 0.8)
plot(classification_raster, axes = TRUE, legend = FALSE,
     col = class_col, breaks = c(class_codes - 1,201),
     main="Clasificación y parcelas de control")
plot(control_parcels, border = "red", add = TRUE)
legend("bottom", horiz = TRUE,
       legend = class_codes, fill = class_col,
       bty = "n", cex = 0.8)
```



```
## generar matriz de errores en blanco
var_num <- length(class_codes)
error_matrix <- matrix(rep(0,var_num * var_num), nrow = var_num, ncol = var_num)
rownames(error_matrix) <- class_codes
colnames(error_matrix) <- class_codes
```

Muestreo del raster clasificado

```
control_sampling <- extract(classification_raster, control_parcels)
```

Generaión de matriz de confusión

```
for (i in 1:length(control_sampling)) {
  primary_class <- control_parcels@data$STDID1[i]
  secondary_class <- control_parcels@data$STDID2[i]
  antropic <- control_parcels@data$ANTRO[i]

  if (verbose) {
    cat("Parcela de control: ",as.character(control_parcels@data$GRID_ID[i]),"\n")
    cat("Clase primario = ",primary_class,"\n")
    cat("Clase secundario = ",secondary_class,"\n")
    cat("Antrópico = ",antropic,"\n")
    cat("Pixeles observados = ",control_sampling[[i]],"\n")
    cat("Número de pixeles = ",length(control_sampling[[i]]),"\n\n")
  }

  for (j in 1:length(control_sampling[[i]])) {
    if (control_sampling[[i]][j] == primary_class) {
      # coincidencia de clase primario
      ind_1 <- as.character(primary_class)
      error_matrix[ind_1,ind_1] <- error_matrix[ind_1,ind_1] + 1
    } else {
      if (control_sampling[[i]][j] == secondary_class) {
          # coincidencia de clase secundario
          ind_1 <- as.character(secondary_class)
          error_matrix[ind_1,ind_1] <- error_matrix[ind_1,ind_1] + 1
      } else {
          # valor primario esperado (de control) - filas del matriz
          ind_1A <- as.character(primary_class)
          # valor secundario esperado (de control) - filas del matriz
          ind_1B <- as.character(secondary_class)
          # valor observado (en clasificación) - columnas del matriz
          ind_2 <- as.character(control_sampling[[i]][j])
          error_matrix[ind_1A,ind_2] <- error_matrix[ind_1A,ind_2] + 0.5
          error_matrix[ind_1B,ind_2] <- error_matrix[ind_1B,ind_2] + 0.5
      }
    }
  }
}

# representar matriz de confusión
cat("Matriz de confusión:\n")
```

```
## Matriz de confusión:
```

```
round(error_matrix, digits = 0)
```

```
##      100 103 105 110  115 120 130 135 150  160  161 190 192 195 197 200
## 100 1381  28  14   1    6  62  16   0   0    2    0   0   4   0   0   0
## 103   25 300  16   1    4  18   8   0   0    1    0   0   1   0   0   0
## 105   24   5 646  17   16  44  41   4  10   10    2   0   6   1   3   8
## 110    8   2  26 676    6  59   8   4   0   18    0   0   2   0   2   0
## 115   55  10  92  68 1306 192  81   0  18   28   11   0   4   1   1   0
## 120   41  15  70  58   40 873  60   0  15   12    1   0   4   2   0   4
## 130   34   8  50  27   34 138 927   1  49   20   10   0   1   0   1   0
## 135    5   0   8   0    2   6  62 246  24    5    6   0   0   0   0   0
## 150    9   0  24   2    7  16  62   2 643   28   11   0   2   0   0   0
## 160    5   0  26   2    6  19  50   2  37 1066   16   0   2   0   0   8
## 161    3   0  10   1    0   1  11   1  12   34 1463   0   2   0   0   0
## 190    0   0   1   0    0   0   0   0   0    0    0  97   0   0   0   0
## 192   12   6   2   0    2  14   3   0   0    0    0   0 108   0   0   0
## 195    3   0   4   0    0  14   0   0   0    2    0   0   0 259   2   4
## 197    0   0   4   1    1   0   0   0   0    0    0   0   0   0  26   0
## 200    2   0   5   0    0  10   2   0   0    2    0   0   1   0   0 152
```

Estimación del error de clasificación

```
error_stat <- function(m, legend) {
  n <- sum(m)
  cat("Número de elementos: ",n,"\n")
  d <- diag(m)

  rowsums <- apply(m, 1, sum)
  colsums <- apply(m, 2, sum)
  p <- rowsums / n
  q <- colsums / n
  cat("\nSumas normalizadas en filas (p) y en columnas (q):","\n")
  print(data.frame(legend,p,q))

  # respuesta por clase
  recall <- d / colsums
  # precición por clase
  precision <- d / rowsums
  # Métrica F1 (media harmónica de precición y respuesta)
  f1 <- 2 * precision * recall / (precision + recall)
  cat("\nPrecición y respuesta por clase:","\n")
  print(data.frame(legend,precision, recall, f1))

  # precición general
  accuracy <- sum(d) / n
  cat("\nPrecición general: ",accuracy,"\n")

  # métrica de precición kappa
  expaccuracy = sum(p * q)
  kappa = (accuracy - expaccuracy) / (1 - expaccuracy)
  cat("\nKappa de Cohen: ",kappa,"\n")

}

error_stat(error_matrix, class_legend)
```

```
## Número de elementos:  12589
##
## Sumas normalizadas en filas (p) y en columnas (q):
##              legend           p           q
## 100          Urbano 0.120224005 0.127651124
## 103 Infraestructura 0.029748193 0.029787910
## 105             SVA 0.066526333 0.079275558
## 110      Agr. riego 0.064381603 0.067916435
## 115   Agr. temporal 0.148383509 0.113591230
## 120        Pastizal 0.095043292 0.116530304
## 130        Matorral 0.103185321 0.105727222
## 135            BEsp 0.028874414 0.020652951
## 150             BTC 0.064103582 0.064262451
## 160     BTemp disp. 0.098458972 0.097466042
## 161     BTemp dens. 0.122090714 0.120899198
## 190            Golf 0.007903725 0.007705139
## 192             AVU 0.011716578 0.010803082
## 195             VAS 0.022916832 0.020970689
## 197            BGal 0.002581619 0.002700771
## 200            Agua 0.013861307 0.014059894
##
## Precición y respuesta por clase:
##              legend precision    recall        f1
## 100          Urbano 0.9124546 0.8593653 0.8851146
## 103 Infraestructura 0.8010681 0.8000000 0.8005337
## 105             SVA 0.7713433 0.6472946 0.7038954
## 110      Agr. riego 0.8340531 0.7906433 0.8117682
## 115   Agr. temporal 0.6991435 0.9132867 0.7919951
## 120        Pastizal 0.7296281 0.5950920 0.6555284
## 130        Matorral 0.7136259 0.6964688 0.7049430
## 135            BEsp 0.6767538 0.9461538 0.7890938
## 150             BTC 0.7967782 0.7948084 0.7957921
## 160     BTemp disp. 0.8600242 0.8687857 0.8643827
## 161     BTemp dens. 0.9518543 0.9612352 0.9565217
## 190            Golf 0.9748744 1.0000000 0.9872774
## 192             AVU 0.7322034 0.7941176 0.7619048
## 195             VAS 0.8977470 0.9810606 0.9375566
## 197            BGal 0.8000000 0.7647059 0.7819549
## 200            Agua 0.8710602 0.8587571 0.8648649
##
## Precición general:  0.8077687
##
## Kappa de Cohen:  0.787669
```

```
## obtener la estructura de la capa control_parcels
#str(control_parcels)
#as.character(control_parcels@data$GRID_ID)
#as.character(control_parcels@data$STDID1)
#as.character(control_parcels@data$STDID2)
#as.character(control_parcels@data$ANTRO)
```