Control de calidad de clasificacion

Viacheslav Shalisko

12 de junio de 2017

```
verbose = 0

control_parcels_path <- "Control_zones.shp"

raster_path <- "../2016_classif/2016_habitats.tif"

class_codes <- c(100,120,150,200)
class_legend <- c("Artificial",
                  "Inducido",
                  "Natural",
                  "Acuatico")
class_col <- c("lightgray",
               "orange",
               "darkgreen",
               "navy")
```

Visualización de datos

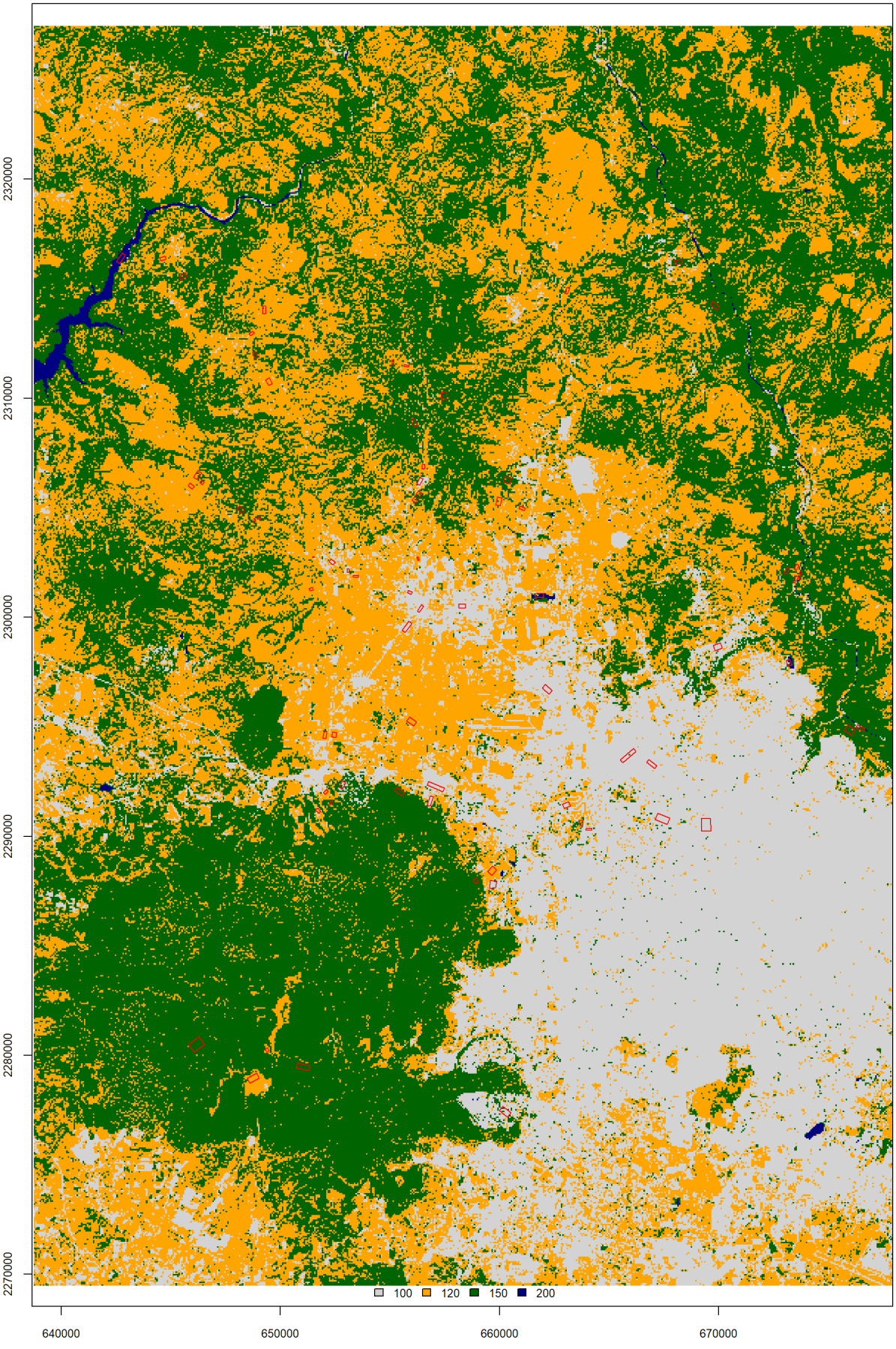
```
control_parcels <- readShapePoly(control_parcels_path)

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

```
## [1] 3840 2618    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)
```

Clasificación y parcelas de control



```
## 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$Habitat[i]
  secondary_class <- control_parcels@data$Habitat2[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("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  120  150 200
## 100 7169  521  206   0
## 120    7 3835   30   0
## 150   42  642 6085   4
## 200   24   46  193 769
```

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:  19572
##
## Sumas normalizadas en filas (p) y en columnas (q):
##         legend          p         q
## 100 Artificial 0.40345902 0.3700184
## 120   Inducido 0.19780809 0.2576640
## 150    Natural 0.34603004 0.3328224
## 200   Acuatico 0.05270284 0.0394952
##
## Precición y respuesta por clase:
##         legend precision    recall        f1
## 100 Artificial 0.9078706 0.9899199 0.9471216
## 120   Inducido 0.9905721 0.7604600 0.8603960
## 150    Natural 0.8984865 0.9341418 0.9159673
## 200   Acuatico 0.7455162 0.9948254 0.8523137
##
## Precición general:  0.9124259
##
## Kappa de Cohen:  0.8716857
```

Estimación del Kappa de Cohen

```
library(fmsb)
```

```
## Warning: package 'fmsb' was built under R version 3.3.2
```

```
Kappa.test(error_matrix, conf.level=0.95)
```

```
## $Result
##
##  Estimate Cohen's kappa statistics and test the null hypothesis
##  that the extent of agreement is same as random (kappa=0)
##
## data:  error_matrix
## Z = 178.79, p-value < 2.2e-16
## 95 percent confidence interval:
##  0.8658832 0.8774882
## sample estimates:
## [1] 0.8716857
##
##
## $Judgement
## [1] "Almost perfect agreement"
```

```
## 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)
```