R Notebook

Code

The SLIC and SLICO algorithms

Each pixel in an image contains two types of information: its color, represented by a vector in a three dimensional space such as the RGN or CIELAB color space, and its location, represented by a vector in two dimensional space (its *x* and *y* coordinates). Thus the totality of information in a pixel is a vector in a five dimensional space.

The SLIC and SLICO algorithms perform a clustering operation similar to K-means clustering on the collection of pixels as represented in this five-dimensional space.

Hide

```
library(sp)
library(raster)
```

Step 1.0: Read the sattelite images for the blue, green, red, and near infrared bands as RasterLayer objects.

Hide

```
# step 1.0
# path to images
droneImagesPath <- "MachineLearningClass/DroneImages/"
droneImageName <- "NobalWheat_NUE_7-6-2021_20m_transparent_reflectance"
#print(getwd())
#print(sprintf("%s%s%s", droneImagesPath, droneImageName, "_blue.tif"))
setwd('..')
```

```
Warning: The working directory was changed to /Volumes/Data Science 214386/DataScienc
e214386/MTAT.03.227 - Machine Learning/Project/P08 - Differentiation of weeds and cro
p plants from drone imagery, Computer vision inside a notebook chunk. The working dir
ectory will be reset when the chunk is finished running. Use the knitr root.dir optio
n in the setup chunk to change the working directory for notebook chunks.
```

Hide

```
#print(getwd())
b2 <- raster(sprintf("%s%s%s", droneImagesPath, droneImageName, "_blue.tif")) # Blue
```

```
Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_proj = pref
er_proj) :
  Discarded datum Estonia 1997 in Proj4 definition
```

Hide

```
b3 <- raster(sprintf("%s%s%s", droneImagesPath, droneImageName, "_green.tif")) # Gree
n
```

```
Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_proj = pref
er_proj) :
  Discarded datum Estonia 1997 in Proj4 definition
```

Hide

```
b4 <- raster(sprintf("%s%s%s", droneImagesPath, droneImageName, "_red.tif")) # Red
```

```
Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_proj = pref
er_proj) :
  Discarded datum Estonia 1997 in Proj4 definition
```

Hide

```
b5 <- raster(sprintf("%s%s%s", droneImagesPath, droneImageName, "_nir.tif")) # NIR
```

```
Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_proj = pref
er_proj) :
  Discarded datum Estonia 1997 in Proj4 definition
```

Hide

```
b2[is.na(b2)] <- 0
b3[is.na(b3)] <- 0
b4[is.na(b4)] <- 0
b5[is.na(b5)] <- 0
#b6 <- raster(sprintf("%s%s%s", droneImagesPath, droneImageName, "_red edge.tif")) #
 Red Edge
```

Step 1.1: Create a RasterBrick object.

Hide

```
# step 1.1
full.brick <- brick(b2, b3, b4, b5)
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
names(full.brick) <- c("blue", "green", "red", "IR")
writeRaster(full.brick, filename = "regions/region.grd", bylayer = TRUE, suffix = nam
es(full.brick), overwrite=TRUE)
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Step 2.1: Read RasterBrick object from disk.

Step 2.2: Plot the region.brick object and wrote it to disk. As a result we get the False color image of the region.

Hide

```
# step 2.1
full.brick <- brick(b2, b3, b4, b5)
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
names(full.brick) <- c("blue", "green", "red", "IR")
writeRaster(full.brick, filename = "regions/region.grd", bylayer = TRUE, suffix = nam
es(full.brick), overwrite=TRUE)
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
b2 <- raster("regions/region_blue.grd")
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
b3 <- raster("regions/region_green.grd")
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
b4 <- raster("regions/region_red.grd")
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
b5 <- raster("regions/region_IR.grd")
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

Hide

```
region.brick <- brick(b2, b3, b4, b5)
```

```
Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefe
r_proj) :
  Discarded datum Unknown based on GRS80 ellipsoid in Proj4 definition
```

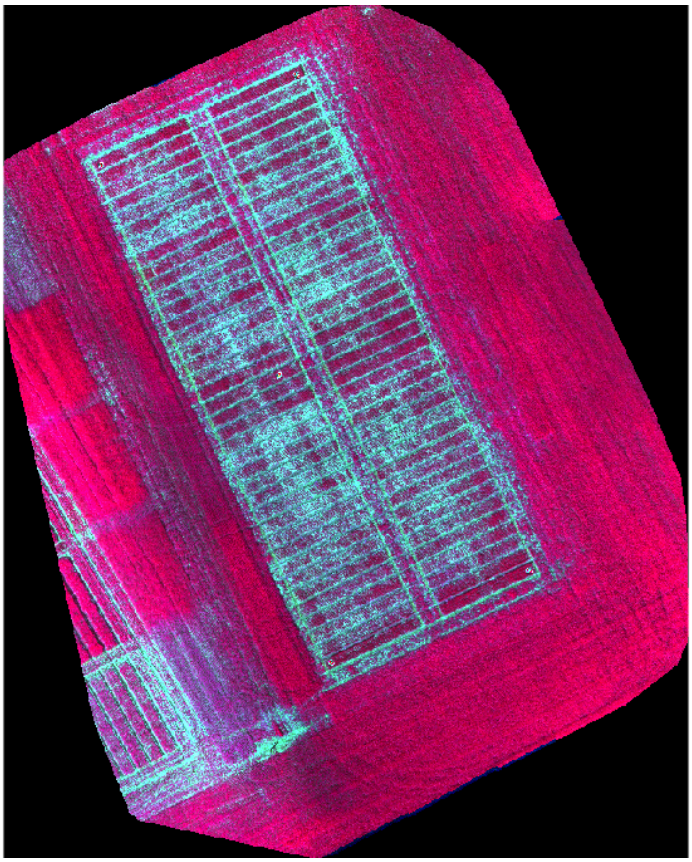
Hide

```
#print(nrows <- region.brick@nrows)
#print(ncols <- region.brick@ncols)

# step 2.2
# plot the region/brick object and write it to disk
plotRGB(region.brick, r = 4, g = 3, b = 2, stretch = "lin")
# write the image to disk
jpeg("FalseColor.jpg", width = ncols, height = nrows)
```

Hide

```
plotRGB(region.brick, r = 4, g = 3, b = 2, stretch = "lin")
dev.off()
```

```
quartz_off_screen
                2
```



Step 3.1: Use SLIC for False Color image segmentation.

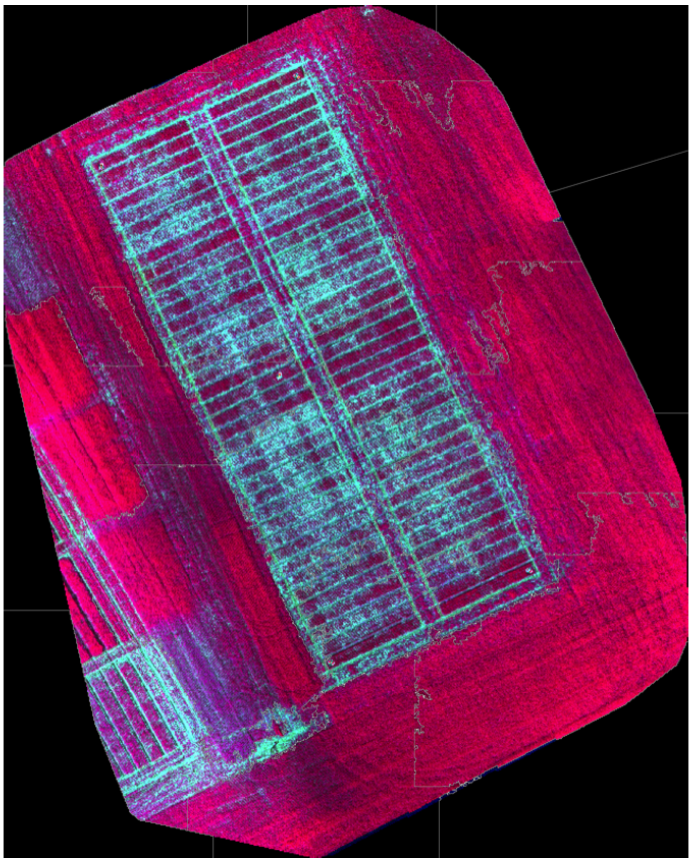
Hide

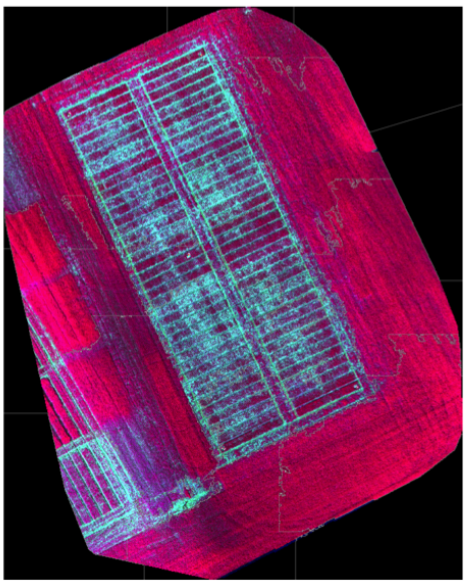
```
# step 3.1
# use SLIC for image segmentation
library(OpenImageR)
False.Color <- readImage("FalseColor.jpg")
Region.slic = superpixels(input_image = False.Color,
                          method = "slic",  #method = "slico",
                          superpixel = 27,  #superpixel = 200,
                          compactness = 3,
                          return_slic_data = TRUE,
                          return_labels = TRUE,
                          write_slic = "",
                          verbose = FALSE)
```

```
Warning in interface_superpixels(input_image, method, superpixel, compactness,  :
  The input data has values between 0.000000 and 1.000000. The image-data will be mul
tiplied by the value: 255!
```

Hide

```
OpenImageR::imageShow(Region.slic$slic_data) #Fig. 5
plot_slic = OpenImageR::NormalizeObject(Region.slic$slic_data)
plot_slic = grDevices::as.raster(plot_slic)
graphics::plot(plot_slic)
```





Step 4.1: Computing NDVI and plot NDVI.region. The darkest areas have the highest NDVI.

Step 4.2: Since NDVI is a ratio scale quantity, the theoretically best practice is to plot it using a monochromatic representation in which the brightness of the color (i.e., the color value) represents the value.

Step 4.2.1: Convert the raster data to a matrix that can be imported into our image segmentation machinery. Remember that by default R constructs matrices by columns. The data in Raster* objects such as NDVI.region are stored by rows, so in converting these data to a matrix we must specify byrow=TRUE.

Step 4.2.2: The function imageShow() works with data that are either in the eight bit 0 - 255 range or in the

$$0, 1$$

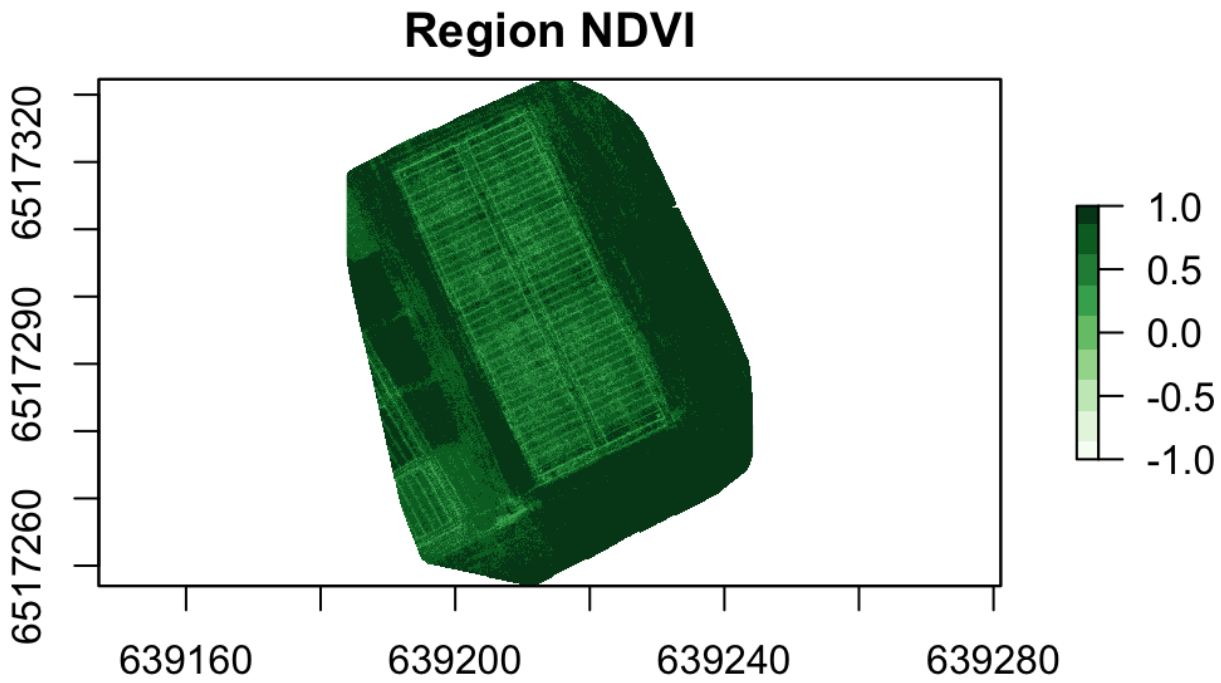
range (i.e., the range of x such that 0<=x<=1). It does not, however, work with NDVI values if these values are negative. Therefor, we will scale NDVI values to

$$0, 1$$

.

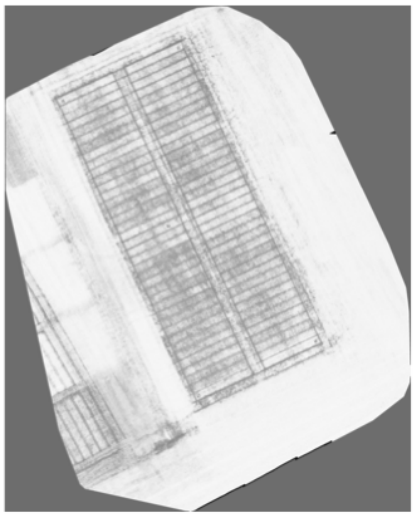
Hide

```
# Step 4.1
library(RColorBrewer)
# compute NDVI
NDVI.region <- (b5 - b4) / (b5 + b4)
plot(NDVI.region, col = brewer.pal(9, "Greens"), axes = TRUE, main = "Region NDVI")
```



Hide

```
# Step 4.2
# Step 4.2.1
NDVI.region[is.na(NDVI.region)] <- 0
NDVI.mat <- matrix(NDVI.region@data@values, nrow = NDVI.region@nrows, ncol = NDVI.reg
ion@ncols, byrow = TRUE)
# Step 4.2.2
# Scale the NDVI to [0,1]
m0 <- min(NDVI.mat)
m1 <- max(NDVI.mat)
NDVI.mat1 <- (NDVI.mat - m0) / (m1 - m0)

#OpenImageR::imageShow(NDVI.mat1)  # Fig. 9
plot_slic = OpenImageR::NormalizeObject(NDVI.mat1)
plot_slic = grDevices::as.raster(plot_slic)
graphics::plot(plot_slic)
```



Step 5.1: We are now ready to carry out the segmentation of the NDVI data. Since the structure of the NDVI image to be analyzed is the same as that of the false color image, we can simply create a copy of this image, fill it with the NDVI data, and run it through the superpixel segmentation function.

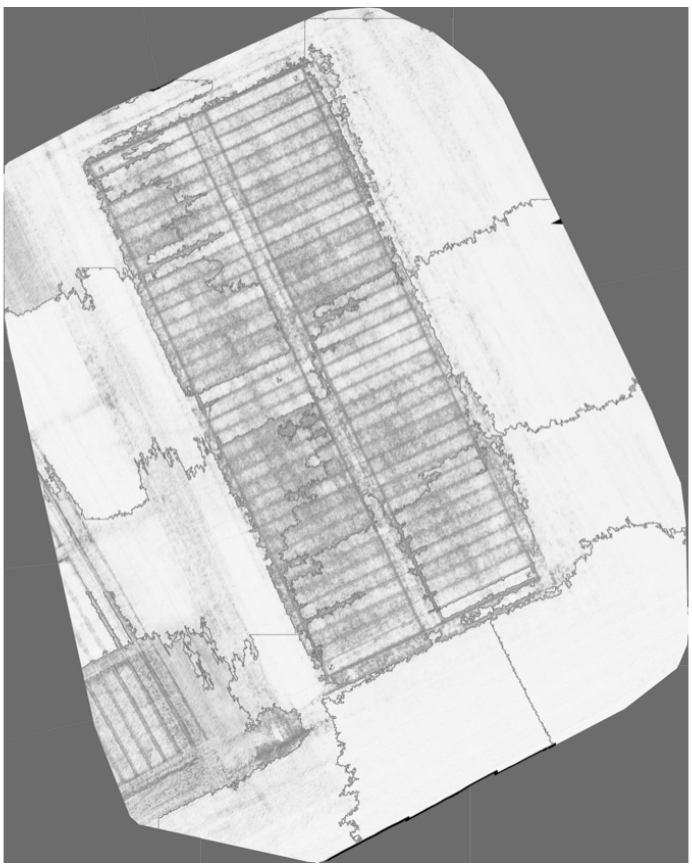
Hide

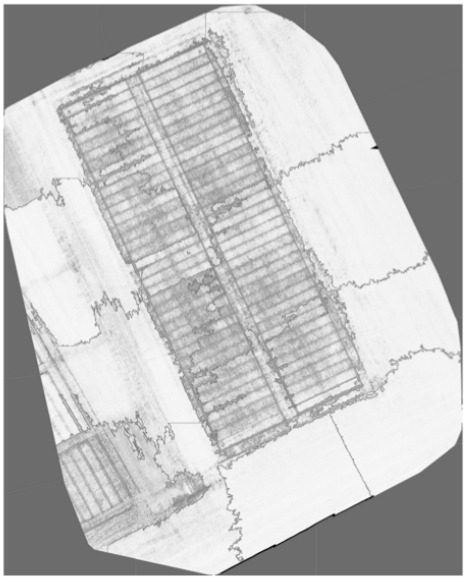
```
# Step 5.1
False.Color <- readImage("FalseColor.jpg")
NDVI.data <- False.Color
NDVI.data[,,1] <- NDVI.mat1
NDVI.data[,,2] <- NDVI.mat1
NDVI.data[,,3] <- NDVI.mat1

NDVI.80 = superpixels(input_image = NDVI.data,
                      method = "slic",  #method = "slico",
                      superpixel = 37,   #superpixel = 200,
                      compactness = 3,
                      return_slic_data = TRUE,
                      return_labels = TRUE,
                      write_slic = "",
                      verbose = FALSE)
```

```
Warning in interface_superpixels(input_image, method, superpixel, compactness,  :
  The input data has values between 0.000000 and 1.000000. The image-data will be mul
tiplied by the value: 255!
```

Hide

```
OpenImageR::imageShow(NDVI.80$slic_data)
plot_slic = OpenImageR::NormalizeObject(NDVI.80$slic_data)
plot_slic = grDevices::as.raster(plot_slic)
graphics::plot(plot_slic)
```





The segmentation process is now can be organized into two steps.

The first step will be to replace each of the superpixels generated by the OpenImageR function superpixel() with one in which each pixel has the same value, corresponding to a measure of central tendency (e.g. mean) of the original superpixel.

Function: Identify a measure of central tendency of each superpixel

Hide

```
# Identify a measure of central tendency of each superpixel
make.segments <- function(x, ftn){
# The argument ftn is any functional measure of central tendency
   z <- x
# For each identified superpixel, compute measure of central tendency
   for (k in unique(as.vector(x$labels))){
# Identify members of the superpixel having the given label
      in.super <- matrix(0, nrow(x$label), ncol(x$label))
      for (i in 1:nrow(x$label))
         for (j in 1:ncol(x$label))
            if (x$label[i,j] == k)
               in.super[i,j] <- 1
#Identify the boundary cells as having all values 0
      on.bound <- matrix(0, nrow(x$label), ncol(x$label))
      for (i in 1:nrow(x$label))
         for (j in 1:ncol(x$label))
            if (in.super[i,j] == 1){
               if (x$slic_data[i,j,1] == 0 & x$slic_data[i,j,2] == 0
                  & x$slic_data[i,j,3] == 0)
                     on.bound[i,j] <- 1
         }
#Identify the superpixel cells not on the boundary
      sup.data <- matrix(0, nrow(x$label), ncol(x$label))
         for (i in 1:nrow(x$label))
         for (j in 1:ncol(x$label))
            if (in.super[i,j] == 1 & on.bound[i,j] == 0)
               sup.data[i,j] <- 1
# Compute the measure of central tendency of the cells in R, G, B
      for (n in 1:3){
# Create a matrix M of the same size as the matrix of superpixel values
         M <- matrix(0, dim(x$slic_data)[1], dim(x$slic_data)[2])
         for (i in 1:nrow(x$label))
            for (j in 1:ncol(x$label))
# Assign to M the values in the superpixel
               if (sup.data[i,j] == 1) M[i,j] <- x$slic_data[i,j,n]
          if (length(M[which(M > 0 & M < 255)]) > 0)
# Compute the measure of central tendency
            ftn.n <- round(ftn(M[which(M > 0 & M < 255)]), 0)
         else
            ftn.n <- 0
         for (i in 1:nrow(x$label))
            for (j in 1:ncol(x$label))
               if (in.super[i,j] == 1) z$slic_data[i,j,n] <- ftn.n
           }
      }
   return(z)
   }
```

Hide

```
NDVI.means <-  make.segments(NDVI.80, mean)
```

The second step will be to use the k-means unsupervised clustering procedure to organize the superpixels from step 1 into a set of clusters and give each cluster a value corresponding to a measure of central tendency of the cluster.

The land is ubdivided into five types: 1, 2, 3, 4, 5.

The raster function ratify() is used to assign descriptive factor levels to the clusters.

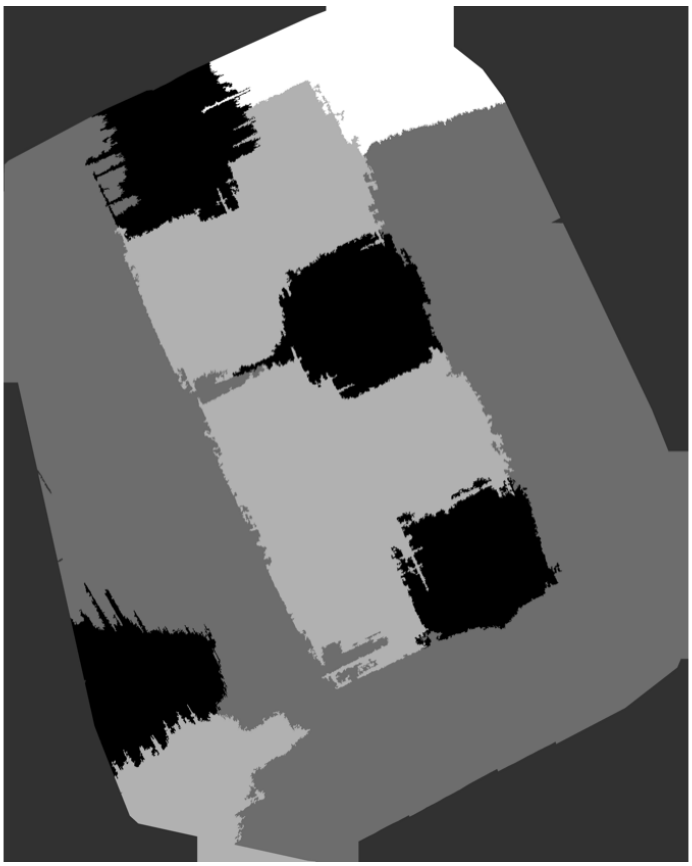
Hide

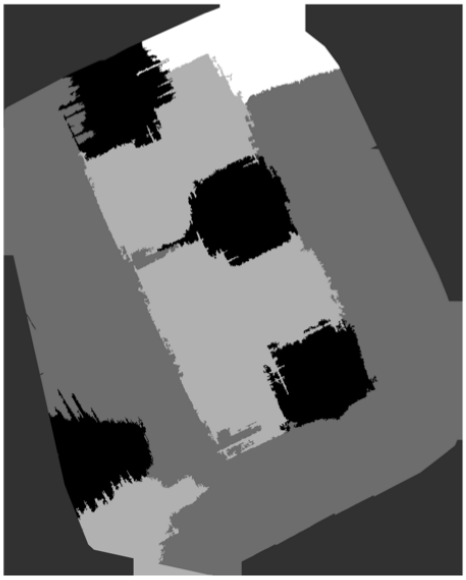
```
library(rasterVis)
# Look at the individual values
sort(unique(as.vector(NDVI.means$slic_data[,,1])))
```

```
 [1] 127 202 203 205 206 207 212 215 217 228 230 236 238 239 240 241 243 244 246
```

Hide

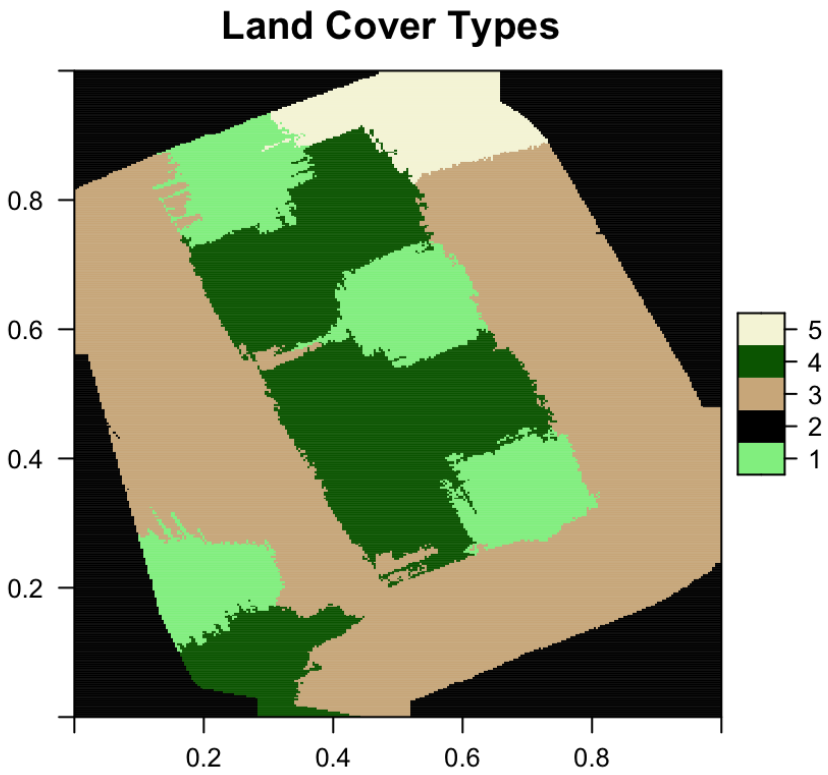
```
# Group into five clusters
set.seed(123)
NDVI.clus <- kmeans(as.vector(NDVI.means$slic_data[,,1]), 5)
vege.class <- matrix(NDVI.clus$cluster, nrow = NDVI.region@nrows,
   ncol = NDVI.region@ncols, byrow = FALSE)
imageShow(vege.class) # Just a check not shown as a figure
plot_slic = OpenImageR::NormalizeObject(vege.class)
plot_slic = grDevices::as.raster(plot_slic)
graphics::plot(plot_slic)
```





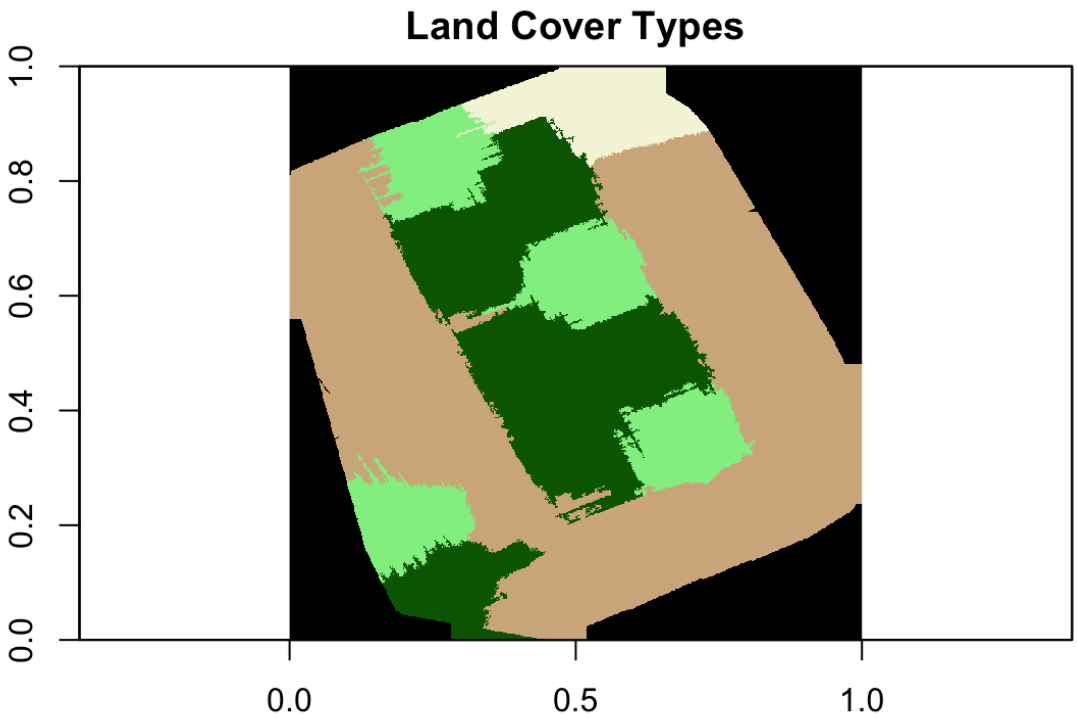
Hide

```
class.ras <- raster(vege.class)
class.ras <- ratify(class.ras)
rat.class <- levels(class.ras)[[1]]
rat.class$landcover <- c("1", "2", "3", "4", "5")
levels(class.ras) <- rat.class
levelplot(class.ras, margin=FALSE, col.regions= c("lightgreen", "black",
  "tan", "darkgreen", "beige"), main = "Land Cover Types") # Fig. 19
```



Hide

```
plot(class.ras, col = c("lightgreen", "black",
  "tan", "darkgreen", "beige"), main = "Land Cover Types",
   legend = FALSE) # Fig. 19
```



Hide

```
#legend("right", legend = c("1", "2", "3", "4",
#  "5"), fill = c("darkgreen", "tan", "lightgreen", "green",
#  "black"))
#plot(NDVI.polymns, add = TRUE)
```