AUTOMATIC CLASSIFICATION IN R

INTRODUCTON

The objective of this exercise is to practice with the unsupervised classification (clustering), applying the general principles that have been explained in the theoretical classes to a specific case. There are many clustering or automatic classification algorithms. In this practice we will work with an algorithm that is considered one of the simplest. However, despite its simplicity, the k-means algorithm is one of the most used in multiple applications of science.

MATERIAL

We will use Sentinel Sentinel-2 MSI images for the Natural Park of "Sierra de las Nieves", which is in Malaga province (Figure 1).

Figure 1. Location of the study area.

Three images of different seasons will be used: spring, summer, and autumn for 2016. Specific dates are shown in table 1.

Table 1. Names and dates for the Sentinel-2 images

Name	Season	Date
S2A_L1C_20160329_T30SUF_.tif	Spring	25/03/2016
S2A_L1C_20160904_T30SUF_.tif	Summer	04/09/2016
S2A_L1C_20161220_T30SUF_.tif	Autumn	20/12/2016

The Sentinel 2-MSI images are made up of thirteen bands, with spatial resolutions between 10 and 60 meters, as shown in table 2. The bands at 60 meters of resolution have been excluded and the rest have been resampled to 20 meters. As a result, **for each date we have an image of 10 bands that correspond to the original bands 2, 3, 4, 5,6, 7, 8, 8A, 11 and 12, at 20 meters of spatial resolution** (table 2). Specifically, they correspond to the product at level 1C (geometric but not atmospheric correction) and being the values given in reflectivity at the top of the atmosphere.

Table2. Spectral bands and spatial resolution of the Sentinel-2A MSI system. In bold and shaded, the bands available for the exercise, resampled to 20 meters**.** Source: ESA (2015) https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument.

Band	Central wavelength	Spectral region	Original spatial resolution
1	443	Blue	60
2	**490**	**Blue-green**	**10**
3	**560**	**Green**	**10**
4	**665**	**Red**	**10**
5	**705**	**Red edge**	**20**
6	**740**	**Red edge**	**20**
7	**783**	**Near Infrared**	**20**
8	**842**	**Near Infrared**	**10**
8A	**865**	**Near Infrared**	**20**
9	945	Near Infrared	60
10	1380	Medium infrared	60
11	**1610**	**Short Wave Infrared**	**20**
12	**2190**	**Short Wave Infrared**	**20**

The three images are in tif format. Additionally, the following data are given:

- File with the spectral signatures to train the model. This file has been obtained from a random sampling form the images.

INSTRUCTIONS

1. The first step is to load the libraries that we are going to use in the exercise.

```
#load libraries
library(raster)
library(dplyr)
library(mlr)
library(randomForest)
library(ggplot2)
library(clue)
```

2. It is important to fix the “seed” to allow replicability.

```
#defining the seed
set.seed(123, "L'Ecuyer")
```

3. We will also define the working directories.

```
#defining working directories
HDFpath<- "C:/COLOUR/MAPPING" # directory with data
setwd(HDFpath)                # fixing the directory
```

4. Once we have done all the about our RStudio session is ready to start working with data. Load the .csv file with the spectral signatures of training areas.

```
#loading the csv with the training areas
training<-read.table("training.csv", header=TRUE, sep=",")
```

If you print the object training by typing “summary(training)”, you will see the information below:

```
> summary(training)
       X              BlueV           GreenV           RedV          RedEdge1V       RedEdge2V
 Min.   :   1.0   Min.   : 795.5   Min.   : 677.0   Min.   : 495.0   Min.   : 667.8   Min.   :1185
 1st Qu.:102.5   1st Qu.: 906.1   1st Qu.: 856.5   1st Qu.: 724.4   1st Qu.:1026.2   1st Qu.:1774
 Median :204.0   Median :1121.2   Median :1129.2   Median :1212.5   Median :1444.2   Median :2055
 Mean   :204.0   Mean   :1186.8   Mean   :1218.7   Mean   :1376.5   Mean   :1571.4   Mean   :2120
 3rd Qu.:305.5   3rd Qu.:1445.1   3rd Qu.:1518.8   3rd Qu.:1908.9   3rd Qu.:2036.9   3rd Qu.:2368
 Max.   :407.0   Max.   :2013.8   Max.   :2421.8   Max.   :3425.8   Max.   :3603.0   Max.   :3963
```

As you can see there is information about the basic statistics of every band (variable or feature). Note that there is a variable called X that does not represent reflectivity values, but the Identifiers or every row in the data frame (correlative integers). We need to remove this as it will bias the classification.

5. Delete the X feature with the following command.

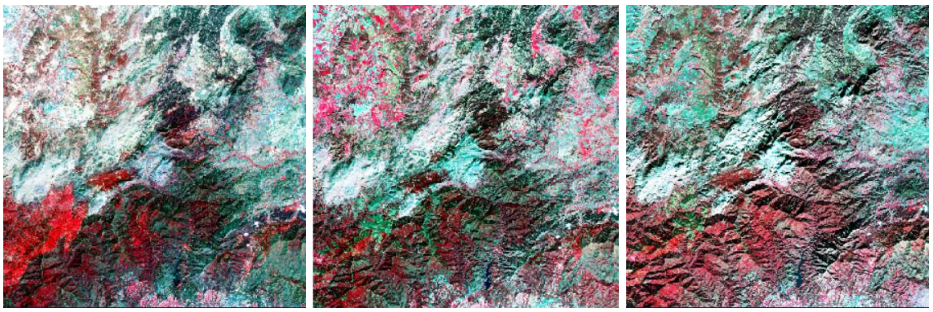
```
#removing the coloumn with row iderntifiers
training<-select(training, -X)
```

6. Now we will open the images and visualise them using different colour compositions.

```
#reading the tif images
summer<-brick("S2A_L1C_20160904_T30SUF_.tif")
spring<-brick("S2A_L1C_20160329_T30SUF_.tif")
autumn<-brick("S2A_L1C_20161220_T30SUF_.tif")

#making a true colour composition
plotRGB(summer, 3, 2, 1, stretch='hist')
plotRGB(spring, 3, 2, 1, stretch='hist')
plotRGB(autumn, 3, 2, 1, stretch='hist')
```

```
#making a false colour composition
plotRGB(summer, 7, 3, 2, stretch='hist')
plotRGB(spring, 7, 3, 2, stretch='hist')
plotRGB(autumn, 7, 3, 2, stretch='hist')
```



7. Create a stack with the three images of Summer, Spring and Autumn and save them in a multiband tiff. Thus, we will have all the features (bands) in a single image. We are also going to rename the bands of each image so that we can know which region of the spectrum and image each one corresponds to.

```
#making a stack
multiseasonal<-stack(summer, spring, autumn)
names(multiseasonal)<-c("BlueV", "GreenV", "RedV", "RedEdge1V", "RedEdge2V",
                        "RedEdge3V", "NIRV", "NIR2V", "SWIR1V", "SWIR2V",
                        "BlueP", "GreenP", "RedP", "RedEdge1P", "RedEdge2P",
                        "RedEdge3P", "NIRP", "NIR2P", "SWIR1P", "SWIR2P",
                        "BlueO", "GreenO", "RedO", "RedEdge1O", "RedEdge2O",
                        "RedEdge3O", "NIRO", "NIR2O", "SWIR1O", "SWIR2O")
```

Save the image as a multiband tiff.

```
#saving the stack Geotiff format
writeRaster(multiseasonal, filename = "C:/COLOUR/MAPPING/Results/multiseasonal.tif",
            format="GTiff", datatype = "FLT4S", overwrite = TRUE)
```

8. We are now going to try to apply the K-means algorithm. There are different versions of the algorithm. In our case we will use the Hartigan-Wong algorithm. The function to be used is called "cluster.kmeans" and it comes from the "mlr" library. We will try for example to classify with 15 spectral classes.

```
#applying the kmeans algorithm
class.task = makeClusterTask(data = training)
lrn = makeLearner("cluster.kmeans", centers = 15, iter.max=100, nstart=5)
model.kmeans = train(lrn, class.task)
```

9. You can check the information about each of the 15 models as follows:

```
#extracting information from the model
names(model.kmeans)
getLearnerModel(model.kmeans)
model.kmeans$learner.model$centers
model.kmeans$learner.model$size
model.kmeans$learner.model$tot.withinss
model.kmeans$learner.model$totss
model.kmeans$learner.model$withinss
```

10. The last step of automatic classification would be to apply the trained model to the “multi-seasonal” satellite image. To do this, we must convert the image into a data frame first, since the models obtained by the mlr library can only be applied to data frame. Then we will have to convert the prediction into a raster image.

```
#applying the model to obtain the map
new_data=as.data.frame(as.matrix(multiseasonal))
pred.kmeans<-predict(model.kmeans, newdata=new_data)
#generating a multiband image with the same number of rows and
#columns than multiseasonal image
map.kmeans15 = multiseasonal[[1]]
# replacing the values of the monoband image
#with the dataframe predicted by the model
map.kmeans15[] = pred.kmeans$data$response
map.kmeans15
```

11. Before finishing the practice do not forget to save the model and the map.

```
#saving models
save(model.kmeans, file="C:/COLOUR/MAPPING/Results/modelo_kmeans.Rdata")

#saving map in Geotiff format
writeRaster(map.kmeans15, filename = "C:/COLOUR/MAPPING/Results/kemans_15.tif",
            format="GTiff", datatype = "FLT4S", overwrite = TRUE)
```