Spatial Data Processing and Land Classification with *terra*

Ren R 690 Seminar, Winter 2023

Benjamin Panes: bpanes@ualberta.ca

Introduction

The purpose of this lab is to demonstrate some spatial processing in R using two classification techniques: LDA and Random Forest. This will be done using the "terra" package, an expanded replacement for the "raster" package that includes a wide variety of additional functions, like vector operations. You can find a list of terra functions here: https://cran.r-project.org/web/packages/terra/terra.pdf, or in the included terra.pdf.

We will be using Landsat 8 imagery from a scene of the Central Valley in California captured on the 14th of June, 2017. Specifically, we will be using surface reflectance for bands 2 through 7. You can read more about Landsat 8 spectral bands here: https://www.usgs.gov/landsat-missions/landsat-8. Landsat data were downloaded from the Earth Explorer portal under Landsat Collection 2 Level-2: https://earthexplorer.usgs.gov/. You can learn more about this specific data product here: https://www.usgs.gov/landsat-missions/landsat-collection-2-surface-reflectance. The bands were converted from 16-bit to 8-bit rasters, clipped to the study area, and renamed.

Spatial Rasters

To get started, first load each band individually. We have 6 bands which will allow us to make predictions using a wide range of the electromagnetic spectrum.

Band 2 (B2)	Blue
Band 3 (B3)	Green
Band 4 (B4)	Red
Band 5 (B5)	Near Infrared (NIR)
Band 6 (B6)	Shortwave Infrared 1 (SWIR 1)
Band 7 (B7)	Shortwave Infrared 2 (SWIR 2)

First, install and load terra.

```
install.packages("terra")
library(terra)
```

Make sure you download the data for the lab and set your working directory to the data folder (You should have the land class symbology, a folder with the 6 bands, study area, and a .csv of training points). You can then load each band as a raster.

```
setwd("[your saved directory]/terra/Data")
B <- rast("Rasters/B2.tif")
G <- rast("Rasters/B3.tif")
R <- rast("Rasters/B4.tif")
NIR <- rast("Rasters/B5.tif")
SWIR1 <- rast("Rasters/B6.tif")
SWIR2 <- rast("Rasters/B7.tif")
B #Some information about the raster
```

```
class       : SpatRaster
dimensions  : 1245, 1497, 1  (nrow, ncol, nlyr)
resolution  : 30, 30  (x, y)
extent      : 594105, 639015, 4190205, 4227555  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 10N (EPSG:32610)
source      : B2.tif
name        : LC08_L2SP_044034_20170614_20200903_02_T1_SR_B2
min value   :                                               0
max value   :                                             156
```

Raster information from blue colour band

terra allows for easy visualization of spatial data with the plot function.

```
gscale <- colorRampPalette(c("black","white")) #An easy
function to create a colour ramp for plots
plot(B,col=gscale(256)) #256 represents the number of
possible values (These are 8-bit rasters, so 0-255)
```

To visualize a composite image, first you can create a raster stack (a raster with multiple bands). This can be done by stacking the rasters that you have already loaded, or loading multiple bands at once.

Using loaded rasters:

```
B.List <- list(B,G,R,NIR,SWIR1,SWIR2)
B.Stack <- rast(B.List) #Same function as single band
B.Stack #There are now 6 layers and names
```

From file:

```
B.List <- list.files(path="./Rasters",full.names=TRUE)
B.Stack <- rast(B.List)
```

We probably want simpler variable names. The current ones came from the original files.

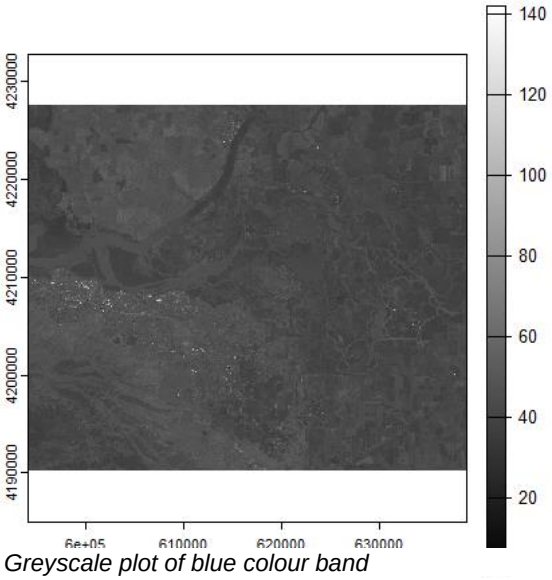
```
names(B.Stack) #Current names
names(B.Stack) <- c("B","G","R","NIR","SWIR1","SWIR2")
names(B.Stack) #Check for new names
```

You can also plot a single band from a stack.

```
plot(B.Stack,y=1,
col=gscale(256))
```

With a raster stack, you can use plotRGB to plot a coloured image using the red, green, and blue bands (or substitute other bands for different compositions).

```
data.frame(names(B.Stack))
#Check for band numbers
plotRGB(B.Stack,
r=3,g=2,b=1,stretch="lin",
smooth=TRUE)
```



Greyscale plot of blue colour band

You might notice that some of these colours do not look natural. To create a natural looking imagine, you would need another program to manually adjust the levels of each band.

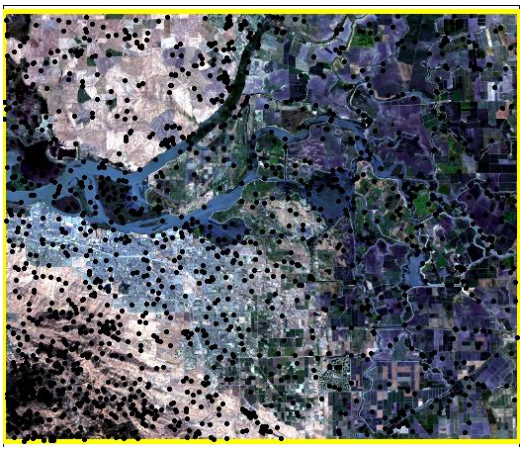
Spatial Vectors

terra can also work with vector files. Try overlaying the last plot with a polygon for the study area.

```
S.Area <- vect("./Study Area")
plot(S.Area,add=TRUE,border="yellow",lwd=5)
```

Next we can look at the training data

```
T.Points <- vect("./Training Points")
plot(T.Points,add=TRUE)
S.Area
T.Points # Compare the
projections (coord. ref.)
```



RGB plot of study area and training data

As you can see, there are no error messages if the projection is off: the data just won't be drawn.

```
T.Points <- project(
T.Points,S.Area)
plot(T.Points,add=TRUE)
```

Predictions

Now that we've confirmed the sample points and rasters match spatially, we can extract the raster values to our points.

```
T.Data <- extract(B.Stack,T.Points) #Only contains raster
information
T.LC <- as.data.frame(T.Points)[2] #Land class column
T.LC$ID <- row.names(T.LC) #ID field for merge
names(T.LC)[1] <- "LC"
T.Data <- merge(T.Data,T.LC,by="ID") #Merge land class and
extracted raster data
T.Data <- T.Data[,-1]
T.Data$LC <- as.factor(T.Data$LC) #Factors are considered as
classes for prediction
```

We will need the rest of our raster data in a format that LDA and randomForest can use to make predictions, that can also be converted back into a raster for visualization. If the prediction models were pre-generated, terra has an in-built predict function for spatial data.

```
R.Grid <- data.frame(xyFromCell(B.Stack[[1]],
1:ncell(B.Stack)[[1]])) #Creates data frame of coordinates
R.Values <- cbind(R.Grid,values(B.Stack)) #Converts layer
values into columns and binds to coordinates
```

First, the LDA model and predictions:

```
library(MASS)
LC.lda <- lda(LC~.,T.Data)
lda.P <- predict(LC.lda,R.Values)
lda.P <- lda.P$class
R.Values$lda <- lda.P
```

Then, randomForest:

```
library(randomForest)
LC.rf <- randomForest(data=T.Data,LC~.,ntree=100)
rf.P <- predict(LC.rf,R.Values)
R.Values$rf <- rf.P
```

To compare the accuracy of the two methods, we can use a confusion matrix. This is a table that compares predicted classifications to the training data. randomForest can do this by running the randomForest object.

```
LC.rf
```

MASS does not have a built-in function for this, so the process is a bit more involved.

```
T.lda.P <- predict(LC.lda,T.Data)
T.lda.P <- T.lda.P$class
library(biotools)
C.Matrix <- confusionmatrix(T.Data$LC,T.lda.P)
C.Matrix
```

We do not have error rates yet, so we need to do some calculations. This is a bit easier in excel.

```
write.csv(C.Matrix,"LDA_Confusion_Matrix.csv")
```

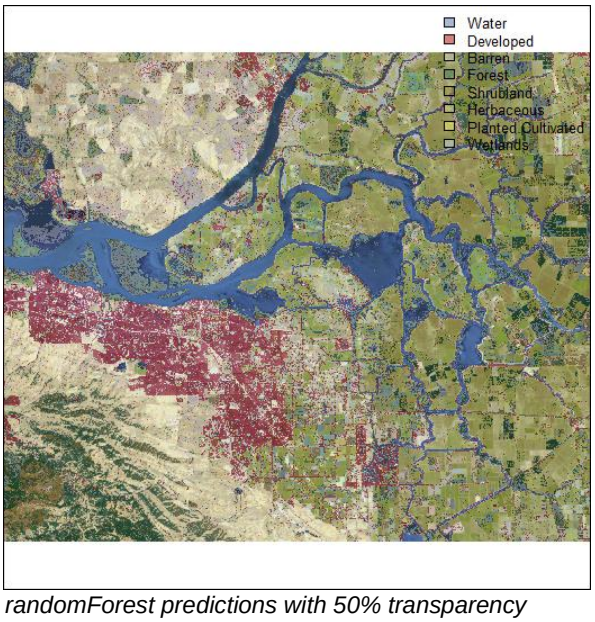
In excel, you just need to divide the correct number of classifications by the total points in each class (the diagonal cell by the sum of the row) and subtract that from 1 to get the error rate: 1-(d/total). You should get **43.69**% for the full table.

The two methods have fairly similar rates, but we should compare the rasters themselves. First create a raster using the previous data frame:

```
LC.Raster <- rast(R.Values,
type="xyz",
crs=crs(B.Stack,proj=TRUE))
names(LC.Raster) #Check for
colour bands and both land
class predictions
```

Then plot the maps. Here we plot the land classifications over the RGB plot. You can adjust the **alpha** level (**0-1**) to change transparency.

```
LC.Symbol <- read.csv(
"Landclass_Symbology.csv",
row.names=1) #Land class
labels and colours
data.frame(names(LC.Raster))
#Confirm layer numbers
```



randomForest predictions with 50% transparency

LDA map:

```
plotRGB(LC.Raster,r=3,g=2,b=1,stretch="lin",axes=TRUE,
smooth=TRUE)
plot(LC.Raster,y=7,alpha=0.5,add=TRUE,col=LC.Symbol$colors,
levels=LC.Symbol$labels,legend="topright",type="classes",
plg=list(text.col="white",border="white"))
```

randomForest map:

```
plotRGB(LC.Raster,r=3,g=2,b=1,stretch="lin",axes=TRUE,
smooth=TRUE)
plot(LC.Raster,y=8,alpha=0.5,add=TRUE,col=LC.Symbol$colors,
levels=LC.Symbol$labels,legend="topright",type="classes",
plg=list(text.col="white",border="white"))
```

Switching between the two maps, you should see that while randomForest has a better accuracy rate, LDA appears to create less noise. Try comparing plotRGB with some other band combinations to help visually identify each land class: https://www.esri.com/arcgis-blog/products/product/imagery/band-combinations-for-landsat-8/

Natural Colour	4 3 2 (R, G, B)
False Colour (urban)	7 6 4 (SWIR 2, SWIR 1, R)
Color Infrared (vegetation)	5 4 3 (NIR, R, G)
Agriculture	6 5 2 (SWIR 1, NIR, B)
Atmospheric Penetration	7 6 5 (SWIR 2, SWIR 1, NIR)
Healthy Vegetation	5 6 2 (NIR, SWIR 1, B)
Land/Water	5 6 4 (NIR, SWIR 1, R)
Natural With Atmospheric Removal	7 5 3 (SWIR 2, NIR, G)
Shortwave Infrared	7 5 4 (SWIR 2, NIR, R)
Vegetation Analysis	6 5 4 (SWIR 1, NIR, R)

You can also plot the training points with the same symbology. Try comparing them with each prediction and the base imagery.

```
LC.Points <- merge(T.Points[,2],LC.Symbol,by="id",all.x=TRUE)
#Subset to land class field and merge symbology to points
plot(LC.Points,y="id",add=TRUE,col=LC.Symbol$colors,
levels=LC.Symbol,legend="topleft",plg=list(text.col="white"))
```

For example, note that there are no wetlands in the bottom left.