Chapter 7

Landscape Metrics

Contents

Spatial pattern analysis is a large and expanding field that permits the assessment of various landscape metrics to describe a defined region. Methods were originally introduced by Kevin McGarigal and were made available on various platforms through Fragstats (McGarigal and Marks 1995). Various landscape metrics can be estimated in Fragstats that include derivations of measures ranging from patch size and area to edge density and spatial configuration of habitat. At the current time, the only package availabe in R is the *SDMToolsl* package. Although the *SDMTools* package is limited, it provides a great deal of landscape metrics and will be the focus of the code that follows.

7.1 Landscape metrics for a single area

Some research designs may just need landscape metrics for a single area or several study areas and that is what the *SDMToolsl* package is able to estimate in the code that follows. While the single area can be defined by the extent of the raster we imported as in previous chapters, the ability of the *SDMToolsl* package to determine patch and class statistics depends on the area defined by the user from that could be study site, within polygons such as counties or townships, or within buffers around locations.

1. Exercise 7.1_7.2- Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(SDMTools)
library(raster)
library(plyr)
library(maptools)
library(rgdal)
```

4. Now open the script "Frag_Auto.R" and run code directly from the script

```
#Load vegetation raster layer textfile clipped in ArcMap
crops <-raster("crop2012utm12.tif")
plot(crops)
class(crops)
```

```
as.matrix(table(values(crops)))
proj4string(crops)
crops

# reclassify the values into 9 groups # all values between 0 and 20 equal 1, etc.
m <- c(-Inf,0,NA,2, 7, 2, 20, 60, 3, 60, 70, 4, 110, 132, 5, 133, 150, 6,
    151, 191, 7, 192,Inf,NA)
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc <- reclassify(crops, rclmat)
plot(rc)
rc
as.matrix(table(values(rc)))#Look at the resulting vegetation categories

#Now we get into Landscape Metrics with the SDTM Tool
#Calculate the Patch statistics
ps.data = PatchStat(rc)
ps.data

str(ps.data)
#data.frame':   7 obs. of  12 variables:
#$ patchID          : int  1 2 3 4 5 6 7
# $ n.cell           : int  16 7260 370104 258106 42429 1016835 726939
# $ n.core.cell      : int  0 2844 222965 107046 1241 699160 279134
# $ n.edges.perimeter: int  50 7902 206012 261672 98766 406726 662034
# $ n.edges.internal : int  14 21138 1274404 770752 70950 3660614 2245722
# $ area             : num  16 7260 370104 258106 42429 ...
# $ core.area        : num  0 2844 222965 107046 1241 ...
# $ perimeter        : num  50 7902 206012 261672 98766 ...
# $ perim.area.ratio : num  3.125 1.088 0.557 1.014 2.328 ...
# $ shape.index      : num  3.12 23.11 84.64 128.65 119.86 ...
# $ frac.dim.index   : num  1.82 1.71 1.69 1.78 1.9 ...
# $ core.area.index  : num  0 0.3917 0.6024 0.4147 0.0292 ...

#Calculate the Class statistics
cl.data = ClassStat(rc)
cl.data

str(cl.data)
#'data.frame':   7 obs. of  38 variables:

NOTE the difference in the output from function PatchStat and ClassStat.
```

7.2 Landscape metrics within polygons

Some research designs may just need landscape metrics for a single area or several study areas and that is what the *SDMToolsl* package is able to estimate in the code that follows. While the single area can be defined by the extent of the raster we imported as in previous chapters, the ability of the *SDMToolsl* package to determine patch and class statistics depends on the area defined by the user from that could be study site, within polygons such as counties or townships, or within buffers around locations.

1. Exercise 7.1_7.2- Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
#Clear workspace from first run
rm(list=ls())

library(SDMTools)
library(raster)
library(rgdal)
library(maptools)
library(sp)
library(adehabitatHR)
library(rgeos)
library(plyr)
```

4. Now continuing along within the script "Frag_Auto.R", run code below metrics within a single area

```
### load raster file into R
raster <- raster("county_hab")
raster

### load PA shapefile into R
HareCounties <- readOGR(dsn=".", layer="Hare_Counties")
HareCounties
plot(HareCounties)
proj4string(HareCounties)
proj4string(raster)
image(raster)
plot(HareCounties, add=T)

#Let's project Counties to habitat just to be safe
new.crs <-CRS("+proj=lcc +lat_1=41.95 +lat_2=40.88333333333333
     +lat_0=40.16666666666666 + lon_0=-77.75 +x_0=600000 +y_0=0
     +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs")
county <- spTransform(HareCounties, CRS=new.crs)
HareCounties <- county
proj4string(HareCounties)
proj4string(raster)
#Matching projections successful!
row.names(HareCounties)<-as.character(HareCounties$COUNTY_NAM)
names.polygons<-sapply(HareCounties@polygons, function(x) slot(x,"ID"))
text(coordinates(HareCounties), labels=sapply(slot(HareCounties, "polygons"),
     function(i) slot(i, "ID")), cex=0.8)
```

5. Now we want to export individual counties by County name (i.e., COUNTY_NAM) as individual shapefiles

```
indata <- HareCounties
innames <- unique(HareCounties@data$COUNTY_NAM)
innames <- innames[1:2]
outnames <- innames
```

```
# begin loop to create separate county shapefiles
for (i in 1:length(innames)){
  data <- indata[which(indata$COUNTY_NAM==innames[i]),]
  if(dim(data)[1] != 0){
    writePolyShape(data,fn=paste(outnames[i],sep="/"),factor2char=TRUE)
    write.table(innames, "List.txt", eol=".shp\n", col.names=FALSE, quote=FALSE,
     row.names=FALSE)
}
}
```

6. We then need to read a list of the counties we want to use in our analysis. Multiple text files with shapefile names can be saved if you want to run separately across states or study regions.

```
#Read in a list of shapefiles files from above
Listshps<-read.table("List.txt",sep="\t",header=F)
Listshps
```

7. Now we use the *plyr* package to create a single function that runs multiple functions in a single statement.

```
shape <- function(Listshps) {
file <- as.character(Listshps[1,])
shape <-readShapeSpatial(file)
mask <- mask(raster,shape)
### Calculate the Class statistics in each county
cl.data <- ClassStat(mask)
}
```

8. The line below also uses the *plyr* package to run the newly created function (shape) over the list of shapefiles called in with the List statement (Listshps) by each ID ([.(id)]) in the Listshps

```
results <- ddply(Listshps, .(id), shape)
write.table(results, "FragCounty.txt")
```

7.3 Landscape Metrics within buffers

1. Exercise 7.3 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(SDMTools)
library(raster)
library(rgeos)
library(plyr)
```

4. Now open the script "PatchAnalystScript.R" and run code directly from the script
5. Load vegetation raster layer textfile clipped in ArcMap

```
crops <-raster("crop2012utm12.tif")
plot(crops)
```

```
class(crops)
as.matrix(table(values(crops)))
proj4string(crops)

# reclassify the values into 9 groups
# all values between 0 and 20 equal 1, etc.
m <- c(-Inf,0,NA,2, 7, 2, 20, 60, 3, 60, 70, 4, 110, 132, 5, 133, 150, 6,
     151, 172, 7, 180, 183, 8, 189, 191, 9,192,Inf,NA)
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc <- reclassify(crops, rclmat)
plot(rc)
rc
as.matrix(table(values(rc)))
```

6. The code above looks at patch and class metrics for the entire study area or within large polygons. However, what if we wanted to compare difference in landscape metrics among all deer? The question is how do you we want to go about doing this? We could run metrics within the home range of each animal or within a buffered circle for each location. To begin this, let's return to our mule deer dataset and import the locations, cleanup, and make buffers around each location.

```
muleys <-read.csv("muleysexample.csv", header=T)
summary(muleys$id)

#Remove outlier locations
newmuleys <-subset(muleys, muleys$Long > -110.90 & muleys$Lat > 37.80)
muleys <- newmuleys
newmuleys <-subset(muleys, muleys$Long < -107)
muleys <- newmuleys

muleys$GPSFixTime<-as.POSIXct(muleys$GPSFixTime, format="%Y.%m.%d%H:%M:%S")
```

7. Here we can use code to create a function using the using the *plyr* package that will let us select all location for each deer, create buffers around each deer, and then run landscape metrics within these merged buffers.

```
buff3rd <- function(muleys) {
  coords<-data.frame(x = muleys$X, y = muleys$Y)
  deer.spdf <- SpatialPointsDataFrame(coords=coords, data = muleys,
   proj4string = CRS("+proj=utm +zone=12 +datum=NAD83 +units=m +no_defs
  +datum=GRS80 +towgs84=0,0,0"))
  settbuff <- gBuffer(deer.spdf, width=1000, byid=FALSE)
  buffclip <- mask(rc, settbuff)
  buff.data <- PatchStat(buffclip)
  newline <- muleys$id
  bind <-cbind(newline[1], buff.data)
}

results <- dlply(muleys, .(id), buff3rd)
results
class(results)

#Now convert results of class "List" to class "data frame"
```

```
df <- do.call(rbind.data.frame, results)
df

"Export to a table if needed"
write.table(df, "FragCombined.txt")

"Read back into R if needed
data <- read.table("FragCombined.txt", sep="", header=T)
data
```

8. The code above looks at patch and class metrics for each deer by combining all buffers into one polygon for each deer (i.e., to define available habitat in 3rd order selection). However, what if we wanted to compare difference in patch statistics among all deer by averaging metrics across buffers?

```
#First we need to create buffers then re-assign a new ID for each deer and
#each buffer generated. We then can apply the function to our dataset.
coords<-data.frame(x = muleys$X, y = muleys$Y)
deer.spdf <- SpatialPointsDataFrame(coords=coords, data = muleys,
     proj4string = CRS("+proj=utm +zone=12 +datum=NAD83 +units=m +no_defs
    +datum=GRS80 +towgs84=0,0,0"))
setbuff <- gBuffer(deer.spdf, width=1000, byid=TRUE)
setbuff
muleys$newID <- paste(muleys$id, setbuff@plotOrder, sep="_")

buff3rdA <- function(muleys) {
  bufclip <- mask(rc, setbuff)
  buf.data <- PatchStat(bufclip)
}

results2 <- ddply(muleys, .(newID), buff3rdA)
results2
```