Chapter 8

Resource Selection

Contents

Figures

Resource selection is used here in reference to a suite of methods to determine resource or habitat use by an animal that can be measured in a heirarchical fashion (i.e., First order through Fourth Order; Johnson 1980). The appropriate method to use is determined by the data that was collected during the study that is often controlled by logistics, funding, and population size of the species studied. As methods to identify animal use of the landscape (i.e., GPS technology) and measures of resource or habitats (i.e., GIS layers) have improved, the methods to analyze data has evolved as well (Cooper and Millspaugh 2001, Manly et al. 2002). Several commonly used measures of habitat use or resource selection include:

1. Compositional Analysis (Aebischer et al. 1993)
2. Mahalanobis Distance (Clark et al. 1993)
3. Selection Ratios (Manly et al. 2002)
4. Resource Selection Functions (Cooper and Millspaugh 2001)

Before we begin with resource selection, we first need to prepare the data that we are interested in incorporating into our modeling efforts. Whether these are linear measures (e.g., distance to roads) or landscape/topographic characteristics (e.g., elevation, slope), summarizing variables as *used* or *available* can be completed all in R. Some of these methods were presented in Chapter 1 so we will build on that before estimating RSF/RSPF in later sections of this chapter.

8.1 Minimum Convex Polygon (MCP)

Minimum Convex Polygon (MCP) estimation was considered a home range originally described for use with identifying animals recaptured along a trapping grid (Mohr 1947). The reason we removed this from the Home Range Section is because MCP can be used to describe the extent of distribution of locations of an animal but NOT as an estimation of home range size. In fact, reporting size of home range using MCP should be avoided at all costs unless you can justify its use as opposed to the plethora of other estimators we have learned in the previous section. We may use MCP within resource selection function analysis as it has been suggested as a method to describe the extent of area occupied by a species that would be available to animals using either second or third order selection of habitat (Johnson 1980), although this should also be avoided unless specifically justified as to why MCP is better than an alternate home range estimator. The extent of an area an animal uses (i.e., habitat available) should be determined for each species and the most appropriate estimator should be used.

1. Exercise 8.1 - 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(adehabitatHR)
   ```

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

   ```
   muleys <-read.csv("muleysexample.csv", header=T)
   muleys
   str(muleys)

   newmuleys <-subset(muleys, muleys$Long > -110.90 & muleys$Lat > 37.8)
   muleys <- newmuleys
   newmuleys <-subset(muleys, muleys$Long < -107)
   muleys <- newmuleys
   ```

5. Create Spatial Points for all relocations and assign IDs to each location

   ```
   data.xy = muleys[c("X","Y")]
   #Creates class Spatial Points for all locations
   xysp <- SpatialPoints(data.xy)
   proj4string(xysp) <- CRS("+proj=utm +zone=17 +ellps=WGS84")

   #Creates a Spatial Data Frame from
   sppt<-data.frame(xysp)
   #Creates a spatial data frame of ID
   idsp<-data.frame(muleys[2])
   #Merges ID and Date into the same spatial data frame
   merge<-data.frame(idsp)
   #Adds ID and Date data frame with locations data frame
   coordinates(merge)<-sppt
   plot(merge)
   str(merge)
   ```

6. We are now ready to create MCPs for our new dataset "merge" by individual animal ID (Fig. 8.1).

```
## estimates the MCP
cp <- mcp(merge[,1], percent=95)#(95% is the default)
## The home-range size
as.data.frame(cp)
## Plot the home ranges
plot(cp)
plot(cp[2,])#only plot deer D8

## ... And the relocations
plot(merge, col=as.data.frame(merge)[,1], add=TRUE)
```

7. We can export the MCPs as shapefiles if needed for use in GIS using the maptools library

```
library(maptools)
writePolyShape(cp, "MCPhomerange")
```

8. We can also look at the area of each MCP as we exclude percentages of relocations (i.e., outliers). We can exclude 5% of the most extreme relocations or we can compute the home-range size for various choices of the number of extreme relocations to be excluded, using the function mcp.area:

```
hrs <- mcp.area(merge[,1], percent=seq(50, 100, by = 5))
hrs
```

Figure 8.1: Example of 95% estimate of home range for mule deer using Minimum Convex Polygon.

8.2 Preparing linear measures

First we will begin with determining the distance between several features. In our first example, we want to measure distance from each mule deer location to the nearest stream if it is determined *a priori* that water or riparian habitats influence mule deer distribution in our study area. While this may not seem like a very complicated process, there are numerous steps needed to achieve this feat. We will need to use the package *spatstat* that will help us in

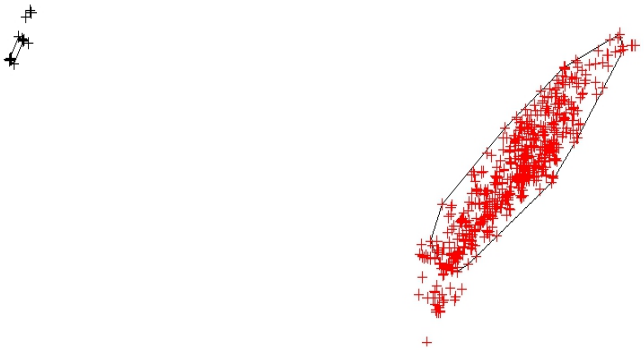

Figure 8.2: Example of 95% estimate of home range for mule deer using Minimum Convex Polygon with relocations overlayed.

creating individual segments with nodes for linear features such as roads and streams/rivers.

1. Exercise 8.2 - 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(raster)
library(sp)
library(rgdal)
library(lattice)
library(rgeos)
library(spatstat)
```

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

```
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

#Make a spatial data frame of locations after removing outliers
coords<-data.frame(x = muleys$X, y = muleys$Y)
crs<-"+proj=utm +zone=12 +datum=NAD83 +units=m +no_defs +ellps=GRS80
     +towgs84=0,0,0"

#Convert to a SpatialPointsDataFrame
deer.spdf <- SpatialPointsDataFrame(coords= coords, data = muleys,
```

```
     proj4string = CRS(crs))
deer.spdf[1:5,]
class(deer.spdf)
proj4string(deer.spdf)
```

5. Load the necessary road and rivers shapefiles already in Albers projection to match previous vegetation raster.

```
roads<-readOGR(dsn=".",layer="AlbersRoads")
rivers<-readOGR(dsn=".",layer="AlbersRivers")
plot(roads,pch=16)
points(deer.spdf, col="red")
plot(rivers,add=T, col="blue",pch=16)
```

6. We need to project the deer.spdf to Albers to match other layers

```
Albers.crs <-CRS("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96
     +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs +ellps=GRS80
     +towgs84=0,0,0")
deer.albers <-spTransform(deer.spdf, CRS=Albers.crs)
points(deer.albers, col="red")
class(deer.albers)
proj4string(deer.albers)
deer.spdf[1:5,]
deer.albers[1:5,]
```

7. Determine boundary box around mule deer locations to create a layer to clip and zoom in.

```
bbox(deer.albers)
bb1 <- cbind(x=c(-1106865,-1106865,-1145027,-1145027, -1106865),
     y=c(1695607, 1729463,1729463,1695607,1695607))
AlbersSP <- SpatialPolygons(list(Polygons(list(Polygon(bb1)),"1")),
     proj4string=CRS(proj4string(deer.albers)))
plot(AlbersSP)
```

8. Load vegetation raster layer tif that came in the Albers projection from the online source.

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

#Check to see all our layers are now in Albers projection
proj4string(crops)
proj4string(deer.albers)
proj4string(AlbersSP)

plot(crops)
points(deer.albers, col="red")
```

9. Clip the raster using the bounding box (AlbersSP) created in step 5.

```
bbclip <- crop(crops, AlbersSP) #Start re-run of code here after new clip#########
cliproads <- gIntersection(roads, AlbersSP, byid=TRUE)
cliprivers <- gIntersection(rivers, AlbersSP, byid=TRUE)

#Plot all to see if it is working for us and zoomed in to mule deer locations.
```

```
plot(bbclip)
points(deer.albers, col="red")
plot(cliproads, add=T)
plot(cliprivers, col="blue",add=T)
```

8.2.1 Formatting layers for package spatstat

Code below will be for use with the *spatstat* package to convert segments of line layers (e.g., roads, rivers) to lines to enable distance to feature from deer locations. Most calculations with spatstat require 3 new classes so most code is created to achieve this goal:

"owin" Observation windows
"ppp" Planar point patterns
"psp" Planar segment patterns

1. Let's start with the road layer by converting a single line to a set of segments packaged as a function.

```
foo <- function(cliproads){
x <- cliproads@Lines[[1]]@coords
cbind(
head(x,-1),
tail(x,-1))}

#The function can be applied successively to each line in the list we extracted
#from roads. Results are output as a list, then converted to a matrix.
segs.lst <- lapply(cliproads@lines,foo)
segs <- do.call(rbind,segs.lst)

segs.x <- c(segs[,c(1,3)])
segs.y <- c(segs[,c(2,4)])
segs.owin <- as.owin(c(range(segs.x),range(segs.y)))

#The segments as a planar segment pattern:
segs.psp <- as.psp(segs, window=segs.owin)
plot(segs.psp)
points(deer.albers)
segs.psp
lengths.psp(segs.psp)

#We can cut road segments into lengths we control as well
dist <- pointsOnLines(segs.psp, eps=1000)
```

2. We need to back up a moment to handle the mule deer locations. Need to convert deer.albers from SPDF back to a dataframe because we need xy coordinates to be in Albers. NOTE: If all data is in UTM 12N or a similar projection from the beginning then no need for the next step. Then we need to make mule deer xy coordinates a planar point patter (i.e., *ppp*) for use in package *spatstat*.

```
deer2 <-as.data.frame(deer.albers)
deer2[1:5,]
newdeer <-cbind(deer2$x,deer2$y)
newdeer[1:5,]
```

```
xy.ppp <- as.ppp(newdeer,W=bdy)
plot(xy.ppp)
```

3. Also we need to back up and make the bounding polygon (AlbersSP) a class *owin* in order to proceed with functions in package *spatstat*.

```
AlbersSPDF <- as(AlbersSP, "SpatialPolygonsDataFrame")
#poly <- AlbersSPDF@polygons[[1]]@Polygons[[1]]@coords#These 2 lines won't work
#bdy.gpc <- as(poly, "gpc.poly")                      #anymore due to gpclib issues
bdy.gpc <- as(AlbersSPDF, "gpc.poly")
bdy.owin <- gpc2owin(bdy.gpc)#new code using polyCub package
bdy.owin <- as.owin(AlbersSPDF)#NOTE:If 2 lines of new code does not work use this
                               #line with Spatial Polygons Data Frame
bdy <- as.polygonal(bdy.owin)
xy.ppp <- as.ppp(newdeer,W=bdy)
plot(xy.ppp)

#Let's check to determine if mule deer locations, bounding box, and road layer
#are in the proper classes to proceed.
is.owin(bdy.owin)
#[1] TRUE
is.ppp(xy.ppp)
#[1] TRUE
is.psp(segs.psp)
#[1] TRUE
#All is TRUE so now we can move forward with the analysis
```

4. Now we can determine the distance from mule deer locations (xy.ppp) to the nearest road

```
roaddist <- nncross(xy.ppp, segs.psp)$dist
roaddist[1:5]

#Or identify segment number closest to each point
v <- nearestsegment(xy.ppp,segs.psp)#Identifies segment number not a distance
plot(segs.psp)
plot(xy.ppp[101], add=TRUE, col="red")
plot(segs.psp[v[101]], add=TRUE, lwd=5, col="red")
```

5. Now we do the same to a river layer by converting a single line to a set of segments packaged as a function.

```
foo <- function(cliprivers){
x <- cliprivers@Lines[[1]]@coords
cbind(
head(x,-1),
tail(x,-1))}

#Again, the function is applied successively to each line in the list then
#results are output as a list, then converted to a matrix.
rivs.lst <- lapply(cliprivers@lines,foo)
rivs <- do.call(rbind,rivs.lst)

rivs.x <- c(rivs[,c(1,3)])
rivs.y <- c(rivs[,c(2,4)])
```

```
rivs.owin <- as.owin(c(range(rivs.x),range(rivs.y)))

#The segments as a planar segment pattern:
rivs.psp <- as.psp(rivs, window=rivs.owin)
plot(rivs.psp)
points(deer.albers)
is.psp(rivs.psp)
#[1] TRUE
#All is TRUE so now we can move forward with the analysis
```

6. Now we can determine the distance from mule deer locations (xy.ppp) to the nearest river.

```
rivdist <- nncross(xy.ppp, rivs.psp)$dist
#rivdist #activate this code to see all distances

#Or identify segment number closest to each point
riv <- nearestsegment(xy.ppp,rivs.psp)
plot(rivs.psp, lwd=1)
plot(xy.ppp[1], add=TRUE, col="red")
plot(rivs.psp[riv[1]], add=TRUE, lwd=5, col="red")

#This code allows us to determine the nearest river to each deer location
plot(xy.ppp[290], add=TRUE, col="blue")
plot(rivs.psp[riv[290]], add=TRUE, lwd=5, col="blue")
```

8.2.2 Summarizing linear measures as covariates

1. We can then summarize the distances in some meaningful way for analysis. Instead of representing distance to road as individual numerical values we can bin the distances in some categories we determine appropriate for our research objective.

```
br <- seq(0,4000,500)
lbl <- paste(head(br,-1),tail(br,-1),sep="-")
road.tbl <- table(cut(roaddist,breaks=br,labels=lbl))
Rdresults <- road.tbl/sum(road.tbl)
Rdresults

br1 <- seq(0,4000,500)
lbl1 <- paste(head(br1,-1),tail(br1,-1),sep="-")
river.tbl <- table(cut(rivdist,breaks=br1,labels=lbl1))
Rivresults <- river.tbl/sum(river.tbl)
Rivresults

#Or we can place each distance into a category or Bin for each deer.
library(Rcmdr)
BinRoad <- bin.var(roaddist, bins=5, method='intervals',
     labels=c('1','2','3','4','5'))
BinRoad

BinRivers <- bin.var(rivdist, bins=5, method='intervals',
     labels=c('1','2','3','4','5'))
```

```
BinRoad #end re-run here##########

#Let's try to add these distance covariates back to the original muley dataset.
Dist <- cbind(BinRoad,BinRivers)
muleys <- cbind(muleys, Dist)#Will not work because 2 locations are out of the box
```

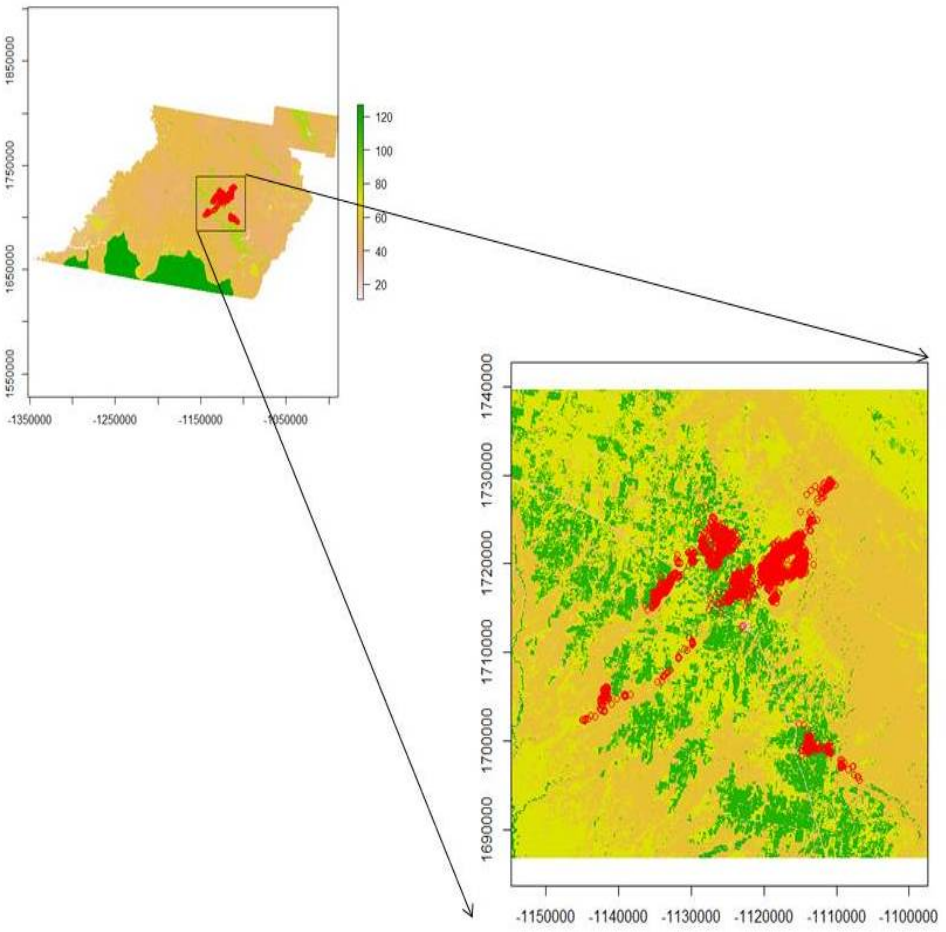

Figure 8.3: Zooming in around mule deer locations using drawExtent in the raster package in R.

8.3 Preparing additional covariates

We may often be interested in assessing various covariates that may influence resource selection of our study animals. If we have *a priori* knowledge that elevation or slope may influence selection for or use of portions of the landscape then we need to create these layers for analysis. While this may not seem like a very complicated process because it is routinely done in ArcMap, those same available layers can be used and manipulated in R as in Chapter 1. We can then create slope, aspect, hillshade or other variables within R using concepts in Chapter 6 and extract those covariates for use in modeling all within the R environment.

8.3.1 Manipulating raster layers for inclusion in modeling procedures

1. Exercise 8.3 - 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(sp)
library(lattice)
library(rgdal)#readOGR
library(rgeos)#gIntersection
library(raster)#to import rasters
library(adehabitatHR)
library(maptools)#readAsciiGrid
```

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

```
muleys <-read.csv("muleysexample.csv", header=T)
str(muleys)

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

#Make a spatial data frame of locations after removing outliers
coords<-data.frame(x = muleys$X, y = muleys$Y)
utm.crs<-"+proj=utm +zone=12 +datum=NAD83 +units=m +no_defs +ellps=GRS80
     +towgs84=0,0,0"
utm.spdf <- SpatialPointsDataFrame(coords= coords, data = muleys, proj4string =
     CRS(utm.crs))

Albers.crs <-CRS("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96 +x_0=0
     +y_0=0 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0")
deer.spdf <-spTransform(utm.spdf, CRS=Albers.crs)
```

5. We can create a bounding box around locations and clip as above or using coordinates of box.

```
bbox(deer.spdf)
bb1 <- cbind(x=c(-1127964,-1127964,-1115562,-1115562,-1127964),
   y=c(1718097,1724868,1724868,1718097,1718097))
AlbersSP <- SpatialPolygons(list(Polygons(list(Polygon(bb1)),"1")),
     proj4string=CRS(proj4string(deer.spdf)))
plot(AlbersSP)
points(deer.spdf, col="red")

#Let's buffer around the bounding box to be sure it encompasses all locations
buffSP <- gBuffer(AlbersSP,width=1000)
plot(buffSP)
points(deer.spdf,col="red")
#NOTE: Be sure that the raster layers extend beyond AlbersSP to avoid errors later.
```

6. Now it is time to import some raster layers of the covariates we are interested in for our analysis. Start with raster of vegetation from the 2012 NRCS Crop data that is a nice dataset that is crop specific for each year. Crop data can be found at the NRCS webpage Cropland Data Layer that can be accessed for each county of each state.

```
crops <-raster("crop2012albers.txt")
plot(crops)
class(crops)
as.matrix(table(values(crops)))
proj4string(crops)
crops

# Reclassify crops raster from above into 9 groups as in previous exercises.
# 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)))#Confirms new number of vegetation categories

#Clip using buffSP polygon created earlier to reduce size of raster (if needed).
bbclip <- crop(rc, buffSP)

plot(bbclip)
points(deer.spdf,col="red")
plot(buffSP, add=T)

#Be sure all are in Albers projection before moving forward.
proj4string(bbclip)
proj4string(deer.spdf)
proj4string(buffSP)
```

7. We also want to look at elevation and covariates related to elevation (e.g., slope, aspect) from Section 6.2. These can be created directly in R using the *terrain* function in the *raster* package.

```
elevation <- raster("dem_albers.txt")
image(elevation, col=terrain.colors(10))
contour(elevation, add=TRUE)

#Create slope and aspect
slope = terrain(elevation,opt='slope', unit='degrees')
aspect = terrain(elevation,opt='aspect', unit='degrees')
elevation#NOTE the number of cells for all 3 layers
slope
aspect

#Clip the 3 layers within the buffSP around locations if needed
demclip <- crop(elevation, buffSP)
sloclip <- crop(slope, buffSP)
aspclip <- crop(aspect, buffSP)
```

8. Cast over all 4 layers to a Spatial Grid Data Frame to permit combining into one layer. One very important note here is that the 3 lines of code below may not work on a standard desktop PC because the memory will be maxed out.
Fortunately, we have a Super Computer as we call it that was custom built by Jeff, our

IT guy, with specs as follows:

```
#########################################################################
Intel Core i7-3930K Sandy Bridge-E 3.2GHz LGA 2011 Six-Core Processor
G.SKILL Ripjaws Z Series 32GB (4 x 8GB) 240-Pin DDR3 SDRAM
Western Digital WD Black WD1002FAEX 1TB
7200 RPM 64MB Cache SATA 6.0Gb/s 3.5" Internal Hard Drive
#########################################################################
nlcd <- as.data.frame(as(rc, "SpatialGridDataFrame"))
elev <- as.data.frame(as(demclip, "SpatialGridDataFrame"))
slo <- as.data.frame(as(sloclip, "SpatialGridDataFrame"))
asp <- as.data.frame(as(aspclip, "SpatialGridDataFrame"))

#Now check to be sure the number of cells in each layer are the same
#before proceeding the next step of combining layers.
str(elev)
str(slo)
str(asp)

#Combine elevation, slope and aspect into one layer.
layers = cbind(nlcd,elev, asp, slo)
head(layers)
layers = layers[,-c(2,3, 5,6,8,9)]
names(layers) = c("nlcd","elevation""aspect", "slope",, "x", "y")
head(layers)

#Turn aspect into categorical variable is recommended
aspect_categorical = rep(NA, nrow(layers))
aspect_categorical[layers$aspect < 45 | layers$aspect >= 315] = "N"
aspect_categorical[layers$aspect >= 45 & layers$aspect < 135] = "E"
aspect_categorical[layers$aspect >= 135 & layers$aspect < 225] = "S"
aspect_categorical[layers$aspect >= 225 & layers$aspect < 315] = "W"
table(aspect_categorical)
table(is.na(aspect_categorical))
layers$aspect_categorical = aspect_categorical
head(layers)

write.table(layers,"layer1.txt",sep=",",col.names=TRUE, quote=FALSE)
#############################################################################
#
#
# NOTE: Script may contains Demonstration code that will subset number of locations
# to speed up processing of data during a course exercise. To prevent this, skip
# this line of code above: muleys <- muleys[sample(nrow(muleys), 100),]
#
#
#############################################################################
```

9. We can now begin the task of sampling each of our locations using the code below. This code was created by Ryan Nielsen of West Inc. and was very helpful in this exercise. Alternatively, we could have extracted each covariate layer by layer and included it in our dataset.

```
# Grab values for points created above
grab.values = function(layer, x, y){
# layer is data.frame of spatial layer, with values 'x', 'y', and ____?
# x is a vector
# y is a vector
if(length(x) != length(y)) stop("x and y lengths differ")
z = NULL
for(i in 1:length(x)){
dist = sqrt((layer$x - x[i])^2 + (layer$y-y[i])^2)
#Could adjust this line or add another line to calculate moving window or
#distance to nearest feature
z = rbind(z, layer[dist == min(dist),][1,])
}
return(z)
}

#Grab all values for used and available points based on combined layer data
#set that can take several minutes.
used = grab.values(layers, muleys$X, muleys$Y)
used$x = muleys$X
used$y = muleys$Y
used$animal_id = muleys$id
used$use = 1
used[1:10,]
```

10. We also need to get some measure of what is available for our mule deer population (2nd order selection) or for each mule deer (3rd order selection). We really do not understand the need for 2nd order selection unless you are looking at deer across different landscapes but hardly seems necessary for deer occupying similar areas such as our mule deer in southwestern Colorado. Below we will focus on 3rd order selection with *used* locations for each deer being compared to *available* locations randomly determined within each deer's MCP.

```
#Create MCP for all locations for each deer by ID (3nd order selection).
cp = mcp(deer.spdf[,2],percent=100)
as.data.frame(cp)
## Plot the home ranges and relocations.
plot(cp)
plot(deer.spdf, col=as.data.frame(deer.spdf)[,2], add=TRUE)

#Determine the habitat available using all code below
#First create random sample of points in each polygon
random <- sapply(slot(cp, 'polygons'), function(i) spsample(i, n=50,
     type='random', offset=c(0,0)))
plot(cp) ; points(random[[1]], col='red', pch=3, cex=.5)
#The number in the line of code above in double brackets changes polygons

# stack into a single SpatialPoints object
random.merged <- do.call('rbind', random)

#Extract the original IDs
ids <- sapply(slot(cp, 'polygons'), function(i) slot(i, 'ID'))
```

```
#Determine the number of ACTUAL sample points generated for each polygon
newpts <- sapply(random, function(i) nrow(i@coords))

#Nice check of how many points generated per polygon
newpts
# generate a reconstituted vector of point IDs
pt_id <- rep(ids, newpts)

# promote to SpatialPointsDataFrame
random.final <- SpatialPointsDataFrame(random.merged,
     data=data.frame(poly_id=pt_id))

#Plot random final on MCPs
plot(cp) ; points(random.final, col=random.final$poly_id, pch=3, cex=0.5)
random.final

#Make random.final a data frame to extract raster covariates for each
random.df = as.data.frame(random.final,coords=coords)
names(random.df) = c("ID", "x", "y")

#Grab covariates as we did for mule deer locations above
available = grab.values(layers, random.df$x, random.df$y)
available$x = random.df$x
available$y = random.df$y
available$animal_id = pt_id
available$use = 0
available[1:10,]
```

11. Bind together mule deer locations with covariates extracted (*used*) and random locations within each polygon by deer ID (*available*) into a master dataset for modeling (*data*). The (*use*) column identifies 1 as (*used*) and 0 as (*available*)

```
data = rbind(available, used)
str(muleys)
#A quick check of the data to determine if correct number of records.
#100 locations used +
#100 locations available (2 animals X 50 random locations)
#= 100 #Confirmed in code below
str(data)
#''data.frame': 200 obs. of  9 variables:
# nlcd              : num  7 6 7 7 7 7 7 7 7 6 ...
# elevation         : int  2058 2058 2068 2068 2070 2072 2076 2062 2071 2071 ...
# aspect            : num  105 278 105 80 135 ...
# slope             : num  2.72 3.37 4.68 4.11 6.05 ...
# x                 : num  -1127639 -1127610 -1127864 -1127805 -1127862 ...
# y                 : num  1724257 1724271 1724091 1724218 1724174 ...
# aspect_categorical: chr  "E" "W" "E" "E" ...
# animal_id         : chr  "D12" "D12" "D12" "D12" ...
# use               : num  0 0 0 0 0 0 0 0 0 0 ...
```

12. The above code is for 3rd order selection within home range of each deer. We could also look at 3rd order selection within a buffered circle around each mule deer location that is

common in Discrete Choice Models. The code is similar except the initial steps of creating buffered polygons and obviously includes a lot more polygons than simply MCPs for each deer. Determining the daily distance moved was done in Chapter 3 but new code is available to estimate for each deer or all deer combined.

13. First create buffered circles with radius of 500 m

```
settbuff=gBuffer(deer.spdf, width=500,byid=TRUE)

#Then create random sample of points in each polygon
ranbuff <- sapply(slot(settbuff, 'polygons'), function(i) spsample(i, n=5,
     type='random', offset=c(0,0)))
plot(settbuff) ; points(ranbuff[[100]], col='red', pch=3, cex=.5)

#Stack into a single SpatialPoints object
ranbuff.merged <- do.call('rbind', ranbuff)

#Extract the original IDs
buff_ids <- sapply(slot(settbuff, 'polygons'), function(i) slot(i, 'ID'))

#Determine the number of ACTUAL sample points generated for each polygon
buffpts <- sapply(ranbuff, function(i) nrow(i@coords))
buffpts[1:20] #Nice check of how many points generated per polygon

#Generate a reconstituted vector of point IDs
buffpt_id <- rep(buff_ids, buffpts)

# promote to SpatialPointsDataFrame
buff.final <- SpatialPointsDataFrame(ranbuff.merged,
     data=data.frame(poly_id=buffpt_id))

#Plot buff.final on buffered circles
plot(settbuff); points(buff.final, col=random.final$poly_id, pch=3, cex=0.5)

buff.final[1:20,]

# make 'buff.final' a data.frame
buffer.df = as.data.frame(buff.final,coords=coords)
names(buffer.df) = c("seqIDs", "x", "y")
buffer.df[1:20,]
str(random.df)
str(buffer.df)
#N = 48115 or 9623 mule deer coordinates X 5 random locations per coordinate

# The first line below required 88 minutes on the Super Computer!!
buff_avail = grab.values(layers, buffer.df$x, buffer.df$y)
buff_avail$x = buffer.df$x
buff_avail$y = buffer.df$y
buff_avail$animal_id = buffpt_id
buff_avail$use = 0
buff_avail[1:10,]
```

```
data2 = rbind(buff_avail, used)
str(data2)

#Before closing, we can save the "used" and available data set to use in
#selection ratio exercise if running full dataset
write.table(used, "MD_used.txt")
write.table(available, "MD_avail.txt")

#Save workspace so all analysis are available
save.image("RSF_dataprep.RData")
```

8.4 Selection ratios

We are going to focus the remainder of this chapter on Selection Ratios and Resource Selection Functions (RSFs) because Selection Ratios identify a general use of habitat given what is available that can be further explored and studied through use of RSFs. Resource Selection Functions are spatially-explicit models that predict the (relative) probability of use by an animal at a given area/location during a given time, based on the environmental conditions that influence or account for selection. There are numerous types of RSFs that can be performed based on the availability of data collected during the study and there are volumes of literature devoted to the topic of resource selection and sampling designs for radiotelemetry studies (Cooper and Millspaugh 2001, Erickson et al. 2001, Leban et al. 2001, Manly et al. 2002).

Selection Ratio basic functions

widesI may be used to explore resource selection by animals when designs I occur (i.e., habitat use and availability are measured at the population level because individual animals are not identified). The Manly selectivity measure (selection ratio = used/available) is computed and preference/avoidance is tested for each habitat, and the differences between selection ratios are computed and tested (Manly et al. 2002).

widesII computes the selection ratios with design II data (i.e., the same availability for all animals, but use is measured for each one). An example would be to place a minimum convex polygon around all animal locations throughout a study site and define this as "available" to all animals.

widesIII computes the selection ratios for design III data (i.e., use and the availability are measured for each animal with use and availability unique to each individuals movements and habitat use).

Note that all these methods rely on the following hypotheses: (i) independence between animals, and (ii) all animals are selecting habitat in the same way (in addition to "traditional" hypotheses in these kinds of studies: no territoriality, all animals having equal access to all available resource units, etc. (Manly et al. 2002).

1. Exercise 8.4 - 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(adehabitatHS)
   ```

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

   ```
   MDsr <- read.csv("MD_winter12.csv",header=T)
   ```

```
#Remove deer that cause errors in plot function later
MDsr <- subset(MDsr,MDsr$animal_id !="647579A")
MDsr$animal_id <- factor(MDsr$animal_id)
used <- subset(MDsr, MDsr$use == 1)
used <- used[c(-1,-3:-6,-8:-15)]
used <- xtabs(~used$animal_id + used$crop, used)
used <- data.frame(used[1:13, 1:5])
colnames(used) <- c("1","2","3","4","5")

rand <- subset(MDsr, MDsr$use == 0)
rand <- rand[c(-1,-3:-6,-8:-16)]
rand <- xtabs(~rand$animal_id + rand$crop, rand)
rand <- data.frame(rand[1:13, 1:5])
colnames(rand) <- c("1","2","3","4","5")

# PVT Code for VegRSF #
pvt.W <- widesIII(used,rand,avknown = FALSE, alpha = 0.1)
pvt.W
par(mfrow=c(1,2))
plot(pvt.W)

#Five categories
#1 = Sunflower,summer crops, random crops, grassland
#2 = Winter crops
#3 = Alfalfa
#4 = Forest
#5 = Shrubland

#Now run on distance to roads binned into 10 categories
MDsr <- read.csv("MD_winter12.csv",header=T)
#Delete deer that have limited data
MDsr <- subset(MDsr,MDsr$animal_id !="647582A" & MDsr$animal_id !="647584A")
MDsr$animal_id <- factor(MDsr$animal_id)

#Bin roads into 4 categories instead of 10
MDsr$NewRoad <- MDsr$BinRoad
levels(MDsr$NewRoad)<-list(class1=c("0-200","200-400"), class2=c("400-600",
   "600-800"), class3=c("800-1000","1000-12000","1200-1400"),class4=c("1400-1600",
   "1600-1800", "1800-2000"))

used <- subset(MDsr, MDsr$use == 1)
used <- used[c(-1:-6,-8:-15)]
used <- xtabs(~used$animal_id + used$NewRoad, used)
used <- data.frame(used[1:12, 1:4])

rand <- subset(MDsr, MDsr$use == 0)
rand <- rand[c(-1:-6,-8:-15)]
rand <- xtabs(~rand$animal_id + rand$NewRoad, rand)
rand <- data.frame(rand[1:12, 1:4])
colnames(rand) <- c('0-200','200-400','400-600','600-800','800-1000','1000-1200',
   '1200-1400','1400-1600','1600-1800','1800-2000')
```

```
pvt.road <- widesIII(used,rand,avknown = FALSE, alpha = 0.1)
pvt.road
par(mfrow=c(1,2))
plot(pvt.road)
```

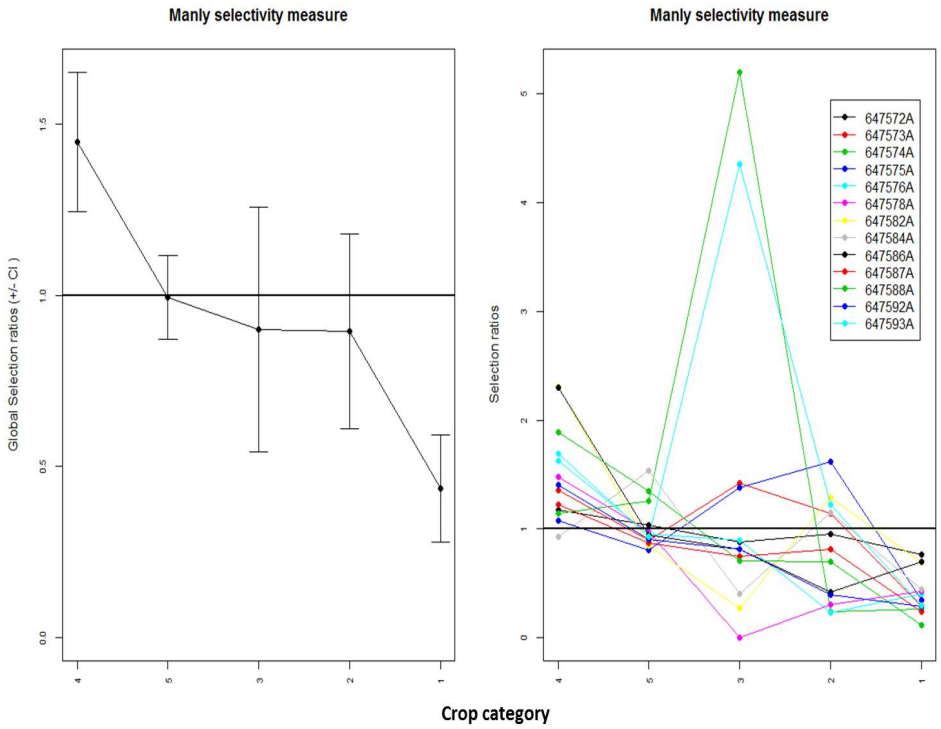

Figure 8.4: Selection ratios for mule deer for 5 habitat types (1-5) using Manly's Selectivity Measure. Habitat type is on x-axis and selectivity measure is on y-axis for a) all deer and b) each deer.

Manly's Selectivity Measure in a Design III would suggest that mule deer in southwestern Colorado are using habitat types 4 and 5 more than they are available (Figure 8.4). Use of the 5 habitat types for each mule deer identifies variability between each mule deer but the trend appears similar across all study animals. Selectivity measure is a general first step look at resource selection but does not really use all the data we have available to use in a Design III study. Although methods of resource selection can be conducted regardless of study design (i.e., I, II, III), the remaining sections of this chapter will focus on Design III because it provides the most detail for each individual animal and should be the goal of most study designs on wildlife for resource selection analysis.

8.5 Resource selection functions

Resource selection requires "used" and "available" habitats and the study designs would take up an entire course all on there own. In this section, we hope to show how we can go about this approach all in R and not need to involve excel spreadsheets with multiple columns of data. More details on methods to estimate resource selection functions (RSFs) or resource

selection probability funcitons (RSPFs) can be found in the literature (Manly et al. 2002, Johnson et al. 2006, Millspaugh et al. 2006). We do not expect you to be experts in RSFs after this section but we want you to be able to implement variious models for RSF analysis in R after determining study design, data collection protocol, and methodology to best achieve your research objectives.

8.5.1 Logistic regression

As we move forward in this section, we are going to assume that your study design and data assessment prior to this section addresses any collinearity in predictor variables and *a priori* hypothesis were used to generate your models used in logistic regression. There are several ways to to calculate RSFs in R using logistic functions that can assess population level or intra-population variation. The use of General Linear Models with various functions in the *lme4* package is often used for estimating population-level models only. Alternatively, we can assess intra-population variation using the *lmer* function. Assessing intra-population variation in a mixed-model approach that provides a powerful and flexible tool for the analysis of balanced and unbalanced grouped data that are often common in wildlife studies that have correlation between observations within the same group or variation among individuals at the same site (Gillies et al. 2006).

1. Exercise 8.4 - 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(lme4)
library(AICcmodavg)
library(adehabitatHR)
```

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

```
data_1 <- read.csv("MD_winter12.csv",header=T)
str(data_1)
```

5. We may need to identify some numerical data as factors or standardize some data prior to implementing resource selection analysis.

```
data_1$crop=as.factor(data_1$crop)
data_1[,2:3]=scale(data_1[,2:3],scale=TRUE)#standardize data to mean of zero
str(data_1)
```

6. We may need to use code that changes Reference Categories of our data. For our analysis we are going to define reference category of *used* habitat as crop= 1. Crop category 1 is sunflowere which was the crop of interest but was not selected for based on Selection Ratios in Exercise 8.4.

```
fit1 = glmer(use ~ relevel(crop,"1")+(1|animal_id), data=data_1,
   family=binomial(link="logit"),nAGQ = 0)#Sunflower and cover model
fit2 = glmer(use ~ d_cover+(1|animal_id), data=data_1,
   family=binomial(link="logit"),nAGQ = 0)#Distance to cover only model
fit3 = glmer(use ~ d_roads+(1|animal_id), data=data_1, family=binomial(link=
   "logit"),    nAGQ = 0)#Distance to roads only model
fit4 = glmer(use ~ d_cover+d_roads+(1|animal_id), data=data_1,
    family=binomial(link="logit"),nAGQ = 0)#Distance to cover and roads model
fit5 = glmer(use ~ 1|animal_id, data=data_1, family=binomial(link="logit"),
```

```
   nAGQ = 0)#Intercept model
```

7. We can view the results of our modeling procedure to select the best model using Akaike's Information Criteria (AIC; Burnham and Anderson 2002).

```
fit1
fit2
fit3
fit4
fit5

AIC(fit1,fit2,fit3,fit4,fit5)

mynames <- paste("fit", as.character(1:5), sep = "")
myaicc <- aictab(list(fit1,fit2,fit3,fit4,fit5), modnames = mynames)
print(myaicc, LL = FALSE)
```

8. Our top model (fit 1) has all the support in this case indicating that during winter 2012 the mule deer were selecting for each habitat over sunflower. Considering sunflower is not available during the winter months this makes perfect sense. Looking at parameter estimates and confidence intervals for the additional habitat categories in *fit 1* we see that forest (category 4) is most selected habitat followed by shrub (category 5). This is only a simply way to look at habitat, however, we used more animals that were on the air for several years and also could look at distance to habitat instead of representing habitat as categorical data.

```
#Get confidence intervals from the top model to interpret results
per1_se <- sqrt(diag(vcov(fit1)))
# table of estimates with 95% CI
tab_per1 <- cbind(Est = fixef(fit1), LL = fixef(fit1) - 1.96 * per1_se,
   UL = fixef(fit1) + 1.96 * per1_se)
```

9. We can then create a surface of predictions from our top model indicating where in our study site we might find the highest probability of use. To do this, we need to export a text file from our "layer" created in Exercise 8.3.

```
layer1 <- read.table("layer1.txt",sep=",")
str(layer1)
names(layer1) = c("crop", "d_cover", "d_roads","x", "y")
str(layer1)
head(layer1)
#Need to standardize the raw distance rasters first to match what we modeled
layer1[,2:3]=scale(layer1[,2:3],scale=TRUE)
head(layer1)
layer1$crop <- as.factor(layer1$crop)

# predictions based on best model
predictions = predict(fit1, newdata=layer1, re.form=NA, type="link")# based on the
  #scale of the linear predictors
predictions = exp(predictions)
range(predictions)

# create Ascii grid of raw predictions if needed
layer1$predictions = predictions
```

```
#preds = layer1
#preds = SpatialPixelsDataFrame(points=preds[c("x", "y")], data=preds)
#preds = as(preds, "SpatialGridDataFrame")
#names(preds)
#writeAsciiGrid(preds, "predictions.asc", attr=13) # attr should be column number
   for 'predictions'

# assign each cell or habitat unit to a 'prediction class'.
# classes have (nearly) equal area, if the cells or habitat units have equal areas.
# output is a vector of class assignments (higher is better).
F.prediction.classes <- function(raw.prediction, n.classes){
  # raw.prediction = vector of raw (or scaled) RSF predictions
  # n.classes = number of prediction classes.
  pred.quantiles = quantile(raw.prediction, probs=seq(1/n.classes, 1-1/n.classes,
   by=1/n.classes))
  ans = rep(n.classes, length(raw.prediction))
  for(i in (n.classes-1):1){
    ans[raw.prediction < pred.quantiles[i]] = i
  }
  return(ans)
}

str(layer1)
layer1$prediction.class = F.prediction.classes(layer1$predictions, 6)
   #attr should be column number for 'predictions'
table(layer1$prediction.class)

##############################################
# create map of RSF prediction classes in R
m = SpatialPixelsDataFrame(points = layer1[c("x", "y")], data=layer1)
names(m)
par(mar=c(0,0,0,0))
image(m, attr=7, col=c("grey90", "grey70", "grey50", "grey30", "grey10"))
par(lend=1)
legend("bottomright", col=rev(c("grey90", "grey70", "grey50", "grey30", "grey10")),
       legend=c("High", "Medium-high", "Medium", "Medium-low", "Low"),
       title="Prediction Class", pch=15, cex=1.0,bty != "n", bg="white")

# create Ascii grid of prediction classes
#m = as(m, "SpatialGridDataFrame")
#names(m)
#writeAsciiGrid(m, "PredictionClassess.asc", attr=7)

#############################################################################
#
# We are in the process of adding negative binomial, discrete choice, and
#  synoptic BBMM for resource selection analysis so check back for updates!
#
#############################################################################
```