8.3 Preparing Additional Covariates

Manual of Applied Spatial Ecology

3/11/2022

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 earlier chapters and extract those covariates for use in modeling all within the R environment.

1. Exercise 8.3 - Download and extract zip folder into your preferred location

2. Set working directory to the extracted folder in R under Session - Set Working Directory...

3. Now open the script MD_DataPrepscript.Rmd" and run code directly from the script

4. First we need to load the packages needed for the exercise

```
library(rgdal)
library(rgeos)#gBuffer
library(raster)#to use "raster" function
library(adehabitatHR)
library(FedData)
```

5. Now we will have a separate section of code to include projection information we will use throughout the exercise. In previous versions, these lines of code were within each block of code

```
utm12.crs<-"+proj=utm +zone=12 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0"
# #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")
Albers.crs <- CRS("+proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0
                  +datum=NAD83 +units=m +no_defs")
```

6. Load the mule deer dataset we used in the previous exercise

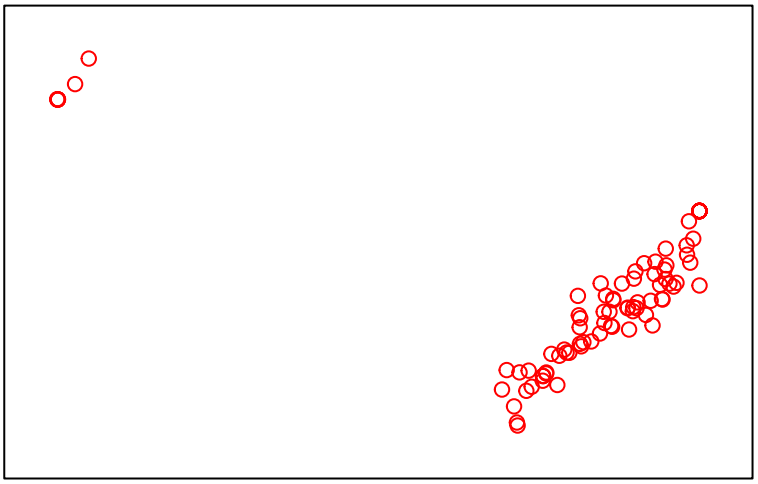
```
muleys <-read.csv("muleysexample.csv", header=T)
#Remove outlier locations
newmuleys <-subset(muleys, muleys$Long > -110.90 & muleys$Lat > 37.80)
muleys <- newmuleys
newmuleys <-subset(muleys, muleys$Long < -107)
muleys <- newmuleys

#Only use the line below for example exercise so fewer locations are used.
muleys <- muleys[sample(nrow(muleys), 100),]

#Make a spatial data frame of locations after removing outliers
coords<-data.frame(x = muleys$X, y = muleys$Y)
utm.spdf <- SpatialPointsDataFrame(coords= coords, data = muleys, proj4string = CRS(utm12.crs))
deer.spdf <-spTransform(utm.spdf, CRS=Albers.crs)
```

7. We can create a bounding box around locations and crop rasters later using coordinates of box.

```
bbox(deer.spdf)
```

```
##        min      max
## x -1127937 -1117017
## y  1718580  1724818
```

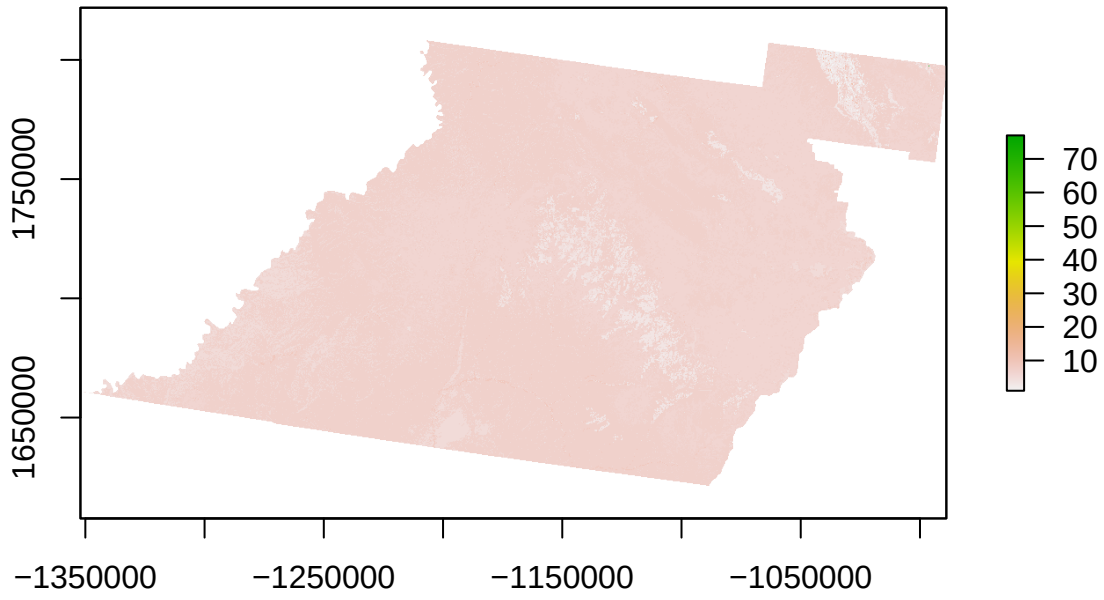
```
bb1 <- cbind(x=c(xmin(deer.spdf)-900,xmin(deer.spdf)-900,xmax(deer.spdf)+900,xmax(deer.spdf)+900,
            xmin(deer.spdf)-900), y=c(ymin(deer.spdf)-900, ymax(deer.spdf)+900,ymax(deer.spdf)+900,
                                      ymin(deer.spdf)-900,ymin(deer.spdf)-900))

AlbersSP <- SpatialPolygons(list(Polygons(list(Polygon(bb1)),"1")),
 proj4string=CRS(proj4string(deer.spdf)))

plot(AlbersSP)
points(deer.spdf, col="red")
```



8. 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("crop12clip.tif")

# Reclassify crops raster from above 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, 9,192,205,10)
```

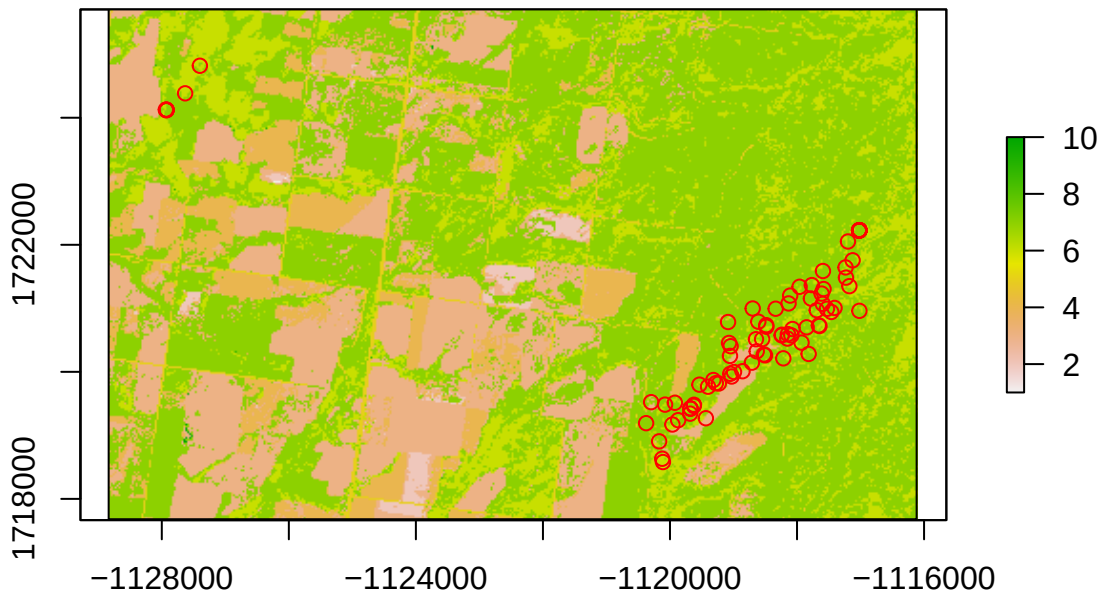
```
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc <- reclassify(crops, rclmat)
plot(rc)
```




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

as.matrix(table(values(bbclip)))#Identifies the number of cells in each category
```

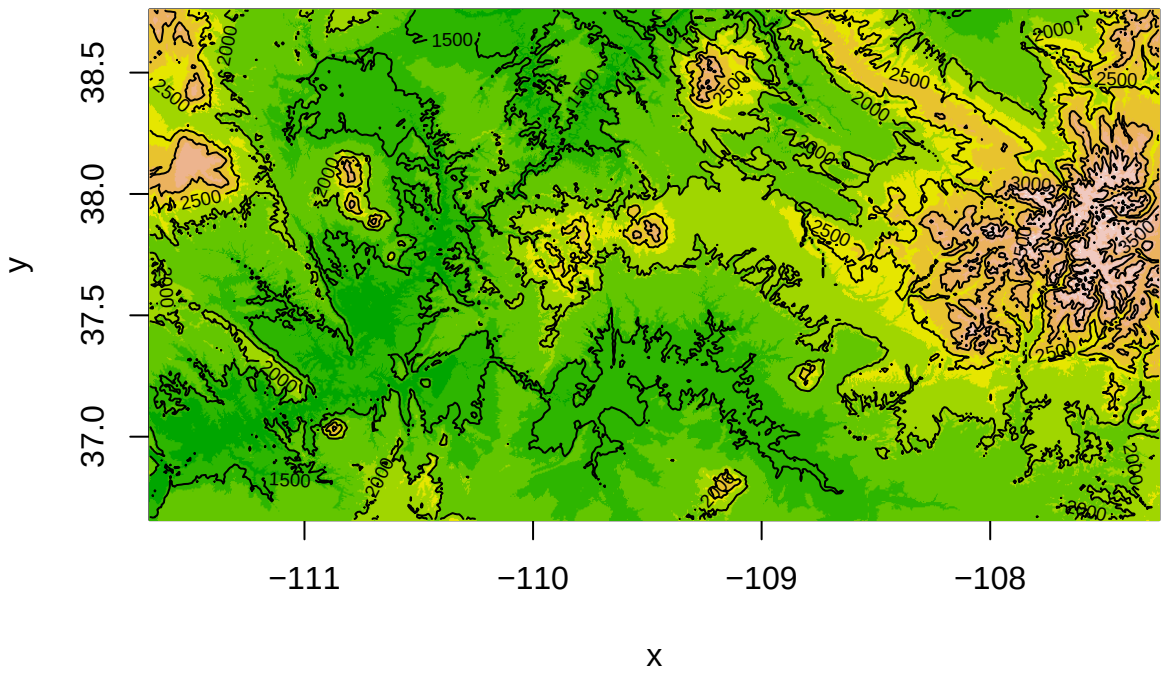
```
##      [,1]
## 1       5
## 2    1133
## 3   22571
## 4   10091
## 5    2303
## 6   16911
## 7   60595
## 9       9
## 10     14
```

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



9. We also want to look at elevation and covariates related to elevation (e.g., slope, aspect). These can be created directly in R using the terrain function in the raster package. We will use the FedData package to get Digital Elevation Models for the large study area

```
DEM <- get_ned(template=rc, label = 'PAdem',force.redo = T)
image(DEM, col=terrain.colors(10))
contour(DEM, add=TRUE)
```




```
slope = terrain(DEM,opt='slope', unit='degrees')
aspect = terrain(DEM,opt='aspect', unit='degrees')

elevation <- projectRaster(DEM, bbclip,method='ngb')
slope <- projectRaster(slope, bbclip,method='ngb')
aspect <- projectRaster(aspect, bbclip,method='ngb')

demclip <- crop(elevation, AlbersSP)
sloclip <- crop(slope, AlbersSP)
aspclip <- crop(aspect, AlbersSP)
```

10. Cast over all 4 layers to a Spatial Grid Data Frame to permit combining into one layer.

```
nlcd <- as.data.frame(as(bbclip, "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)
```

```
## 'data.frame':    113632 obs. of  3 variables:
##  $ PAdem_NED_1: num  2088 2089 2090 2087 2086 ...
##  $ s1         : num  -1128810 -1128780 -1128750 -1128720 -1128690 ...
##  $ s2         : num  1725690 1725690 1725690 1725690 1725690 ...
```

```
str(slo)
```

```
## 'data.frame':    113632 obs. of  3 variables:
##  $ slope: num  2.63 2.53 2.26 3.92 2.12 ...
##  $ s1   : num  -1128810 -1128780 -1128750 -1128720 -1128690 ...
##  $ s2   : num  1725690 1725690 1725690 1725690 1725690 ...
```

```
str(asp)
```

```
## 'data.frame':    113632 obs. of  3 variables:
##  $ aspect: num  228 216 182 114 138 ...
##  $ s1    : num  -1128810 -1128780 -1128750 -1128720 -1128690 ...
##  $ s2    : num  1725690 1725690 1725690 1725690 1725690 ...
```

```
str(nlcd)
```

```
## 'data.frame':    113632 obs. of  3 variables:
##  $ crop12clip: num  7 5 3 3 3 3 3 3 3 3 ...
##  $ s1        : num  -1128810 -1128780 -1128750 -1128720 -1128690 ...
##  $ s2        : num  1725690 1725690 1725690 1725690 1725690 ...
```

11. Combine elevation, slope, and aspect into one layer.

```
layers = cbind(nlcd, elev, asp, slo)
head(layers)
```

```
##   crop12clip       s1      s2 PAdem_NED_1       s1      s2   aspect       s1
## 1          7 -1128810 1725690    2088.444 -1128810 1725690 228.3798 -1128810
## 2          5 -1128780 1725690    2089.118 -1128780 1725690 216.0998 -1128780
## 3          3 -1128750 1725690    2089.638 -1128750 1725690 182.2455 -1128750
## 4          3 -1128720 1725690    2086.993 -1128720 1725690 113.8908 -1128720
## 5          3 -1128690 1725690    2085.756 -1128690 1725690 137.9569 -1128690
## 6          3 -1128660 1725690    2086.085 -1128660 1725690 170.2967 -1128660
##        s2    slope       s1      s2
## 1 1725690 2.629603 -1128810 1725690
## 2 1725690 2.525106 -1128780 1725690
## 3 1725690 2.262172 -1128750 1725690
## 4 1725690 3.920775 -1128720 1725690
## 5 1725690 2.123759 -1128690 1725690
## 6 1725690 1.898647 -1128660 1725690
```

```
#Remove xys that are repeated in each layer
layers = layers[,-c(2,3,5,6,8,9)]
names(layers) = c("nlcd", "elevation", "aspect", "slope","x", "y")

# turn aspect into categorical
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)
```

```
## aspect_categorical
##     E     N     S     W
##  7264 12790 29942 63636
```

```
table(is.na(aspect_categorical))
```

```
##
##  FALSE
## 113632
```

```
layers$aspect_categorical = aspect_categorical
head(layers)
```

```
##   nlcd elevation   aspect    slope        x       y aspect_categorical
## 1    7  2088.444 228.3798 2.629603 -1128810 1725690                  W
## 2    5  2089.118 216.0998 2.525106 -1128780 1725690                  S
## 3    3  2089.638 182.2455 2.262172 -1128750 1725690                  S
## 4    3  2086.993 113.8908 3.920775 -1128720 1725690                  E
## 5    3  2085.756 137.9569 2.123759 -1128690 1725690                  S
## 6    3  2086.085 170.2967 1.898647 -1128660 1725690                  S
```

```
#write.table(layers,"layer1.txt",sep=",",col.names=TRUE, quote=FALSE)

layers2 <- layers #change to layers2 simply to avoid confusion with "layer" term in function
#below
```

#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 (Line 44): #muleys <- #muleys[sample(nrow(muleys), 100),]

12. 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 from muleys for each layer in r
test = grab.values(layers2, muleys$X, muleys$Y)
head(test)
```

```
##      nlcd elevation   aspect    slope        x       y aspect_categorical
## 424     7  2389.714 258.3945 3.845844 -1116120 1725690                  W
## 4241    7  2389.714 258.3945 3.845844 -1116120 1725690                  W
## 4242    7  2389.714 258.3945 3.845844 -1116120 1725690                  W
## 4243    7  2389.714 258.3945 3.845844 -1116120 1725690                  W
## 4244    7  2389.714 258.3945 3.845844 -1116120 1725690                  W
## 4245    7  2389.714 258.3945 3.845844 -1116120 1725690                  W
```

```
##NOTE that all values are the same but this is not correct.
##What is the problem here and how do we fix it?

#Need to grab Albers XY not UTM as in muleys above
muleys <- as.data.frame(deer.spdf)

# grab all values for used and available points based on combined layer data set
# can take 5+ minutes
used = grab.values(layers2, muleys$x, muleys$y)
used$x = muleys$x
used$y = muleys$y
used$animal_id = muleys$id
used$use = 1
head(used)
```

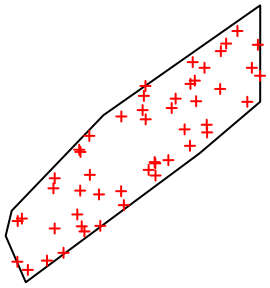
```
##       nlcd elevation   aspect    slope        x       y aspect_categorical
## 88499    7  2223.672 297.9463 8.551859 -1119625 1719464                  W
## 67371    7  2321.029 244.4324 4.261396 -1117460 1720942                  W
## 49154    7  2349.484 207.6831 2.553555 -1117021 1722228                  S
## 74526    7  2256.626 298.8952 6.089706 -1119070 1720454                  W
## 83423    6  2253.073 292.8937 8.168778 -1119278 1719821                  W
## 49578    7  2348.181 206.3074 2.839948 -1117021 1722225                  S
##       animal_id use
## 88499        D8   1
## 67371        D8   1
## 49154        D8   1
## 74526        D8   1
## 83423        D8   1
## 49578        D8   1
```

13. 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)
```

```
##      id      area
## D12 D12   3.82672
## D8   D8 435.93215
```

```
#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[[2]], col='red', pch=3, cex=.5)#The number 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))
newpts #Nice check of how many points generated per polygon
```

```
## [1] 50 50
```

```
# 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))

random.final
```

```
## class       : SpatialPointsDataFrame
## features    : 100
## extent      : -1127920, -1117020, 1718743, 1724669  (xmin, xmax, ymin, ymax)
## crs         : NA
## variables   : 1
## names       : poly_id
## min values  :     D12
## max values  :      D8
```

```
# make 'random.final' a data.frame
random.df = as.data.frame(random.final)
names(random.df) = c("ID","x","y")
# can take 5+ minutes
available = grab.values(layers2, random.df$x, random.df$y)
available$x = random.df$x
available$y = random.df$y
available$animal_id = pt_id
available$use = 0
head(available)
```

```
##       nlcd elevation    aspect    slope        x       y aspect_categorical
## 20386    7  2070.052  90.90746 6.251041 -1127817 1724252                  E
## 17848    6  2075.123 141.31828 5.862021 -1127633 1724434                  S
## 14884    6  2077.454  95.92430 5.038024 -1127519 1724650                  E
## 16153    6  2084.746 127.38931 5.314467 -1127597 1724560                  E
## 15308    6  2077.297  91.43386 5.195473 -1127516 1724604                  E
## 20387    7  2066.253  96.44460 2.293771 -1127802 1724238                  E
##       animal_id use
## 20386       D12   0
## 17848       D12   0
## 14884       D12   0
## 16153       D12   0
## 15308       D12   0
## 20387       D12   0
```

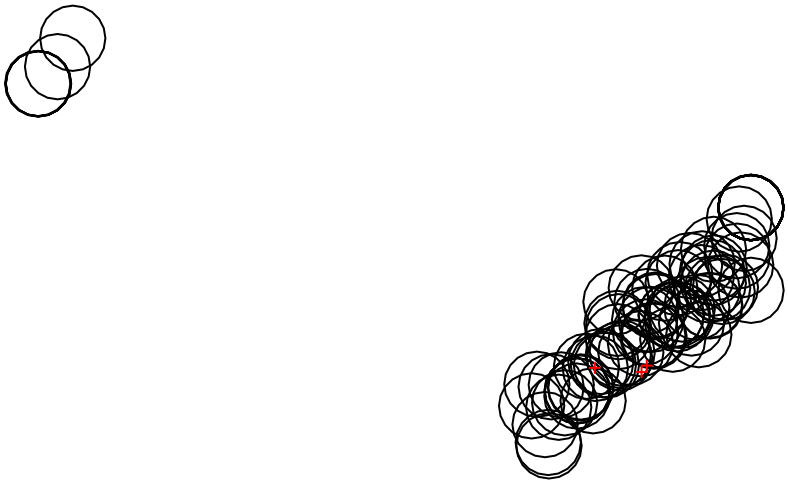
14. Bind together mule deer locations with covariates extracted (used) and random locations within each polygon by deer ID (available) into a master dataset for modeling (data). The (use) column identifies 1 as (used) and 0 as (available)

```
data = rbind(available, used)
##A quick check of the data to determine if correct number of records.
#''data.frame': 200 obs. of  9 variables:
#100 locations used +
#100 locations available (2 animals X 50 random locations)
#= 100 #Confirmed in code below
# 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 ...
```

15. 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.

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

#Determine the habitat available using all code below
#First create random sample of points in each polygon
```

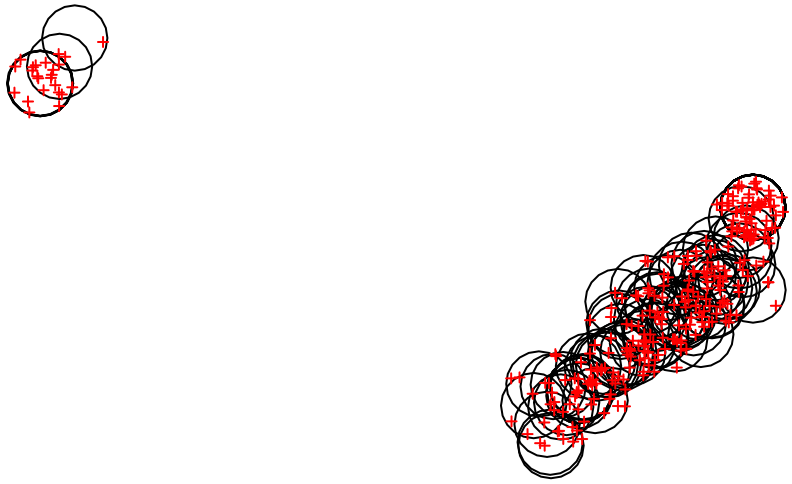
```
ranbuff <- sapply(slot(settbuff, 'polygons'), function(i) spsample(i, n=3, type='random',
  offset=c(0,0)))
plot(settbuff) ; points(ranbuff[[100]], col='red', pch=3, cex=.5)#The number in double
```



```
#brackets changes polygons
#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'))
buff_ids <- paste(settbuff$id, buff_ids, sep="_")
#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
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
# 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="red",pch=3, cex=0.5)
```



```
# make 'buff.final' a data.frame
buffer.df = as.data.frame(buff.final)
names(buffer.df) = c("ID","x", "y")
head(buffer.df)
```

```
##       ID        x       y
## 1 D8_199 -1119651 1719856
## 2 D8_199 -1120058 1719246
## 3 D8_199 -1119694 1719598
## 4 D8_587 -1117427 1721173
## 5 D8_587 -1117174 1721268
## 6 D8_587 -1117450 1721263
```

```
str(random.df)
```

```
## 'data.frame':    100 obs. of  3 variables:
##  $ ID: chr  "D12" "D12" "D12" "D12" ...
##  $ x : num  -1127817 -1127633 -1127519 -1127597 -1127516 ...
##  $ y : num  1724252 1724434 1724650 1724560 1724604 ...
```

```
str(buffer.df)
```

```
## 'data.frame':    300 obs. of  3 variables:
##  $ ID: chr  "D8_199" "D8_199" "D8_199" "D8_587" ...
##  $ x : num  -1119651 -1120058 -1119694 -1117427 -1117174 ...
##  $ y : num  1719856 1719246 1719598 1721173 1721268 ...
```

```
# can take 5+ minutes
buff_avail = grab.values(layers2, 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

data2 = rbind(buff_avail, used)

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

#Before closing, let's save the "used" and available data set to use in the next exercise
write.table(used, "MD_used.txt")
write.table(available, "MD_avail.txt")
```

16. 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 (Manly et al. 2002, Cooper and Millspaugh 2001, Erickson et al. 2001, Leban et al. 2001).