8.6 Negative Binomial

Manual of Applied Spatial Ecology

3/11/2022

1. Exercise 8.6 - 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 NegBinomial.Rmd" and run code directly from the script
4. First we need to load the packages needed for the exercise

```
library(plyr)
library(adehabitatHR)
library(rgeos)
library(raster)
library(rgdal)
library(zoo)
library(MASS)#For nb models
```

5. Now let's 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=WGS84"
Albers.crs<-"+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96 +x_0=0 +y_0=0
  +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs"
```

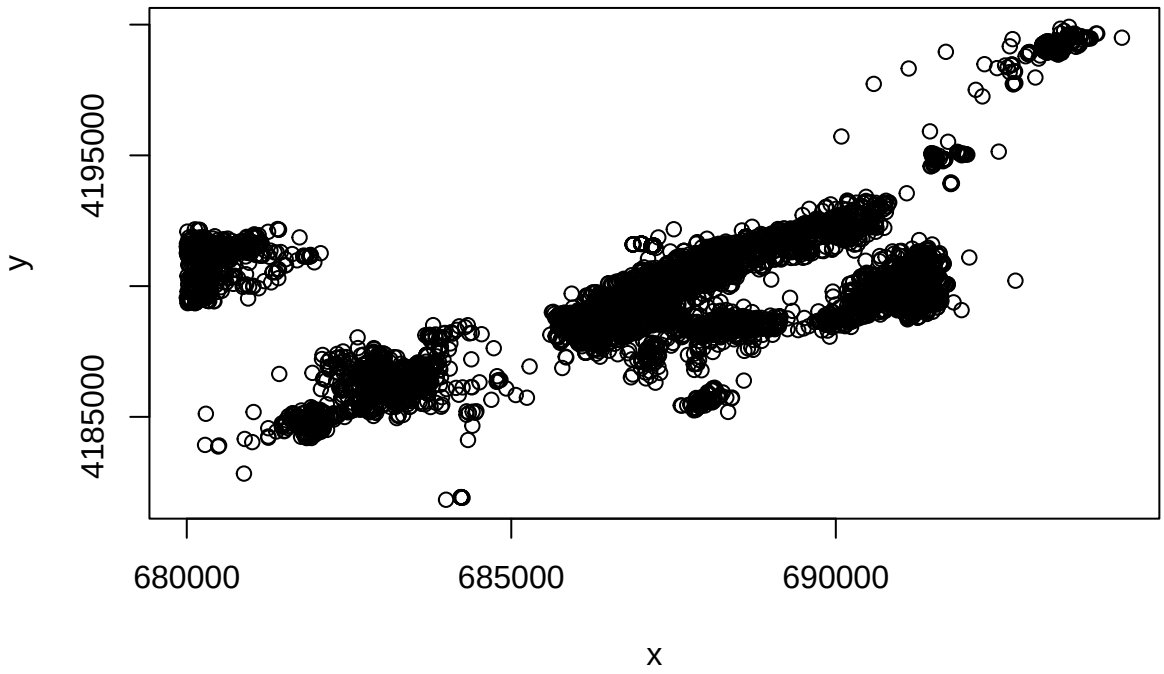
6. Load files for the mule deer dataset and clean up the dataset as we have done in previous exercises

```
#muleys<-read.csv("muleysexample.csv", header=T, sep=",")
muleys<-read.csv("DCmuleysedited.csv", header=T, sep=",")

muleys$NewDate<-as.POSIXct(muleys$GPSFixTime, format="%Y.%m.%d %H:%M:%S", origin="1970-01-01")
muleys <- subset(muleys, muleys$id != "D19")
#Dates deer GPS collars were on the air
# rangefunct <- function(muleys){
#   dates=range(muleys$NewDate)
#   print(dates)
#   days=diff(range(muleys$NewDate))
#   print(days)
#   }
# collars <- dlply(muleys, .(id), rangefunct)

##Sort Data
muleys <- muleys[order(muleys$id, muleys$NewDate),]

##TIME DIFF NECESSARY IN BBMM CODE
timediff <- diff(muleys$NewDate)*60
## remove first entry without any difference
muleys <- muleys[-1,]
```

```
muleys$timelag <-as.numeric(abs(timediff))
##Remove locations greater than 5.5 hours apart in time
muleys <- subset(muleys, muleys$timelag < 19800)
summary(muleys$timelag)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3581   10792   10800   10795   10808   14401
```

```
muleys <- subset(muleys, muleys$X > 680000)# & muleys$GPS.UTM.Easting != "NA")
muleys$id <- factor(muleys$id)

##Make a spatial data frame of locations after removing outliers
muleysSPDF<-data.frame(x = muleys$X, y = muleys$Y)
plot(muleysSPDF, axes=T)#To visualize all locations
```




```
utm.spdf <- SpatialPointsDataFrame(coords = muleysSPDF, data = muleys, proj4string =
  CRS(utm12.crs))

##change muleysSPDF from UTM to Albers
muleys.spdf <-spTransform(utm.spdf, CRS=Albers.crs)
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
## = prefer_proj): Discarded datum Unknown based on GRS80 ellipsoid in Proj4
## definition
```

```
#Make data of albers x and y and replace with original muleys dataframe
muleys <- as.data.frame(muleys.spdf)
```

```
##We can create and clip with a rectangular grid
bbox(muleys.spdf)
```

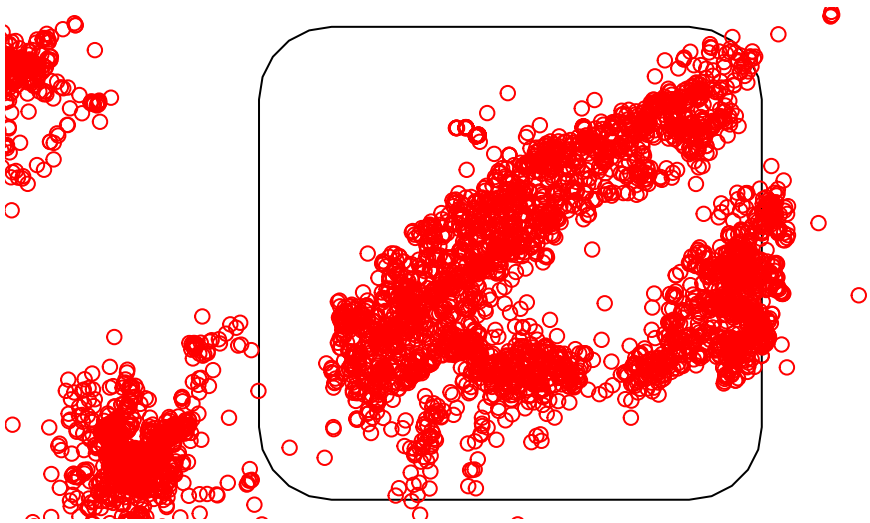
```
##        min      max
## x -1126432 -1110147
## y  1712932  1729463
```

```
bb1 <- cbind(x=c(-1120488,-1120488,-1115562,-1115562,-1120488),
   y=c(1718097, 1722611,1722611,1718097,1718097))
muleysAlbersSP <- SpatialPolygons(list(Polygons(list(Polygon(bb1)),"1")),
  proj4string=CRS(proj4string(muleys.spdf)))
```

```
## Warning in proj4string(muleys.spdf): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```
## Warning in proj4string(muleys.spdf): Discarded datum Unknown based on GRS80 ellipsoid in Proj4 defini
```

```
##Let's buffer around the bounding box to be sure it
#encompasses all locations
muleysbuffSP <- gBuffer(muleysAlbersSP,width=1000)
plot(muleysbuffSP)
points(muleys.spdf,col="red")
```



```
#Subset locations by year for season-specific RSFs if needed
winter2012 <- muleys
# winter2012 <- crop(muleys.spdf,muleysbuffSP)
# winter2012$id <- droplevels(winter2012$id)
```

7. Now we need to set up our sample circles across our study area keeping in mind our discussions on "available" habitats. For this exercise, we will keep it simple by only including a polygon around our mule deer locations. This is for demonstration only, the appropriate study area should be specific to your study design and objectives. We also need to determine what is the appropriate size of our sample circles. In this case, we will use the mean daily movement distance for mule deer we determined to be 628 meters. This will be the radius of our sample circles.

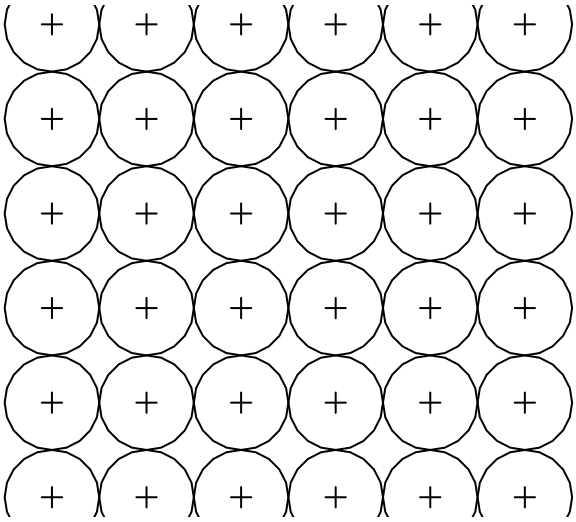
```
#Determine the boundary around our mule deer locations that we will consider to be
#available to all deer in our analysis
bbox(muleysbuffSP)
```

```
##        min      max
## x -1121488 -1114562
## y  1717097  1723611
```

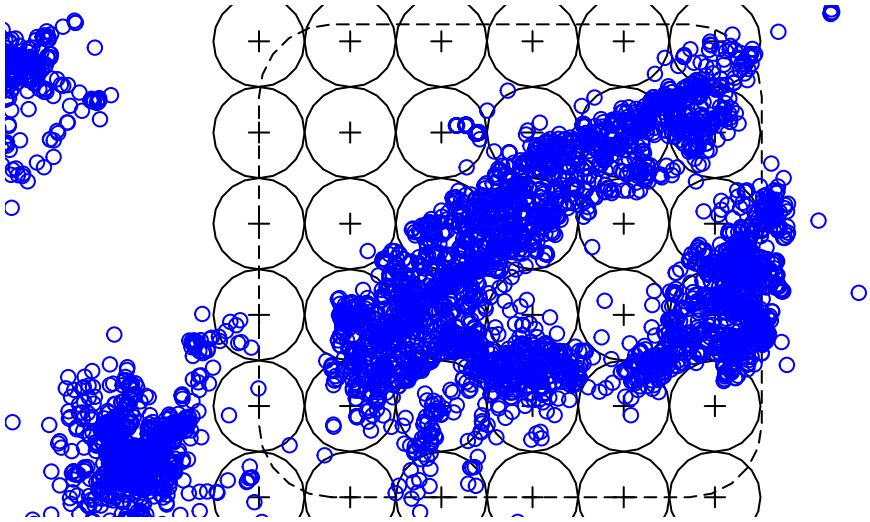
```
#  min      max
#x -1121488 -1114562
#y  1717097  1723611

#Create a square as we did in Exercise 1.9. Start by creating vectors of the x and y points
x <- seq(from = -1121488, to = -1114562, by = 1256)#628 m daily move distance times 2
y <- seq(from = 1717097, to =  1723611, by = 1256)

#Create a grid of all pairs of coordinates (as a data.frame)
#Also the restart point for later!!
xy <- expand.grid(x = x, y = y)

#Identify projection before creating Spatial Points Data Frame
grid.pts<-SpatialPointsDataFrame(coords= xy, data=xy, proj4string = CRS(Albers.crs))
plot(grid.pts)
gridded(grid.pts)
```

```
## [1] FALSE
```

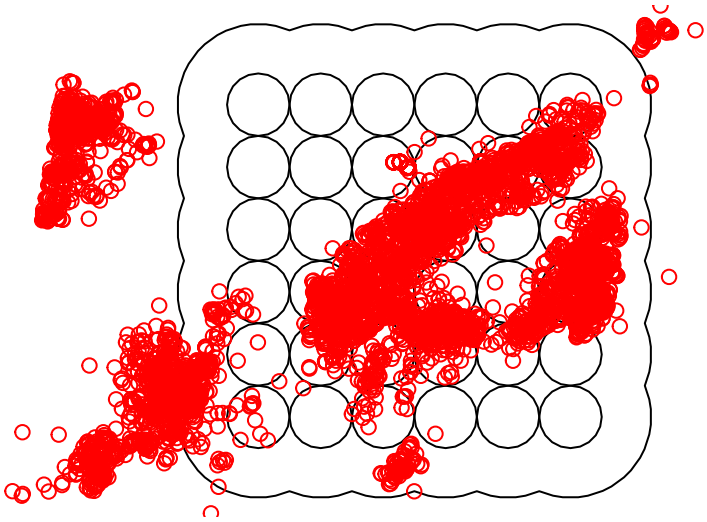
```
griddata <- as.data.frame(grid.pts)

#Create buffers around grid points
mbuffer <- gBuffer(grid.pts,width=628,byid=TRUE)
mbuff.spdf <- SpatialPolygonsDataFrame(mbuffer, data=data.frame(id=row.names(mbuffer),
  row.names=row.names(mbuffer)))
plot(grid.pts)
plot(mbuff.spdf,add=T)
```



```
plot(muleysbuffSP,lty=5)
plot(mbuff.spdf,add=T)
points(grid.pts, pch=3)
points(muleys.spdf, col="blue")
```

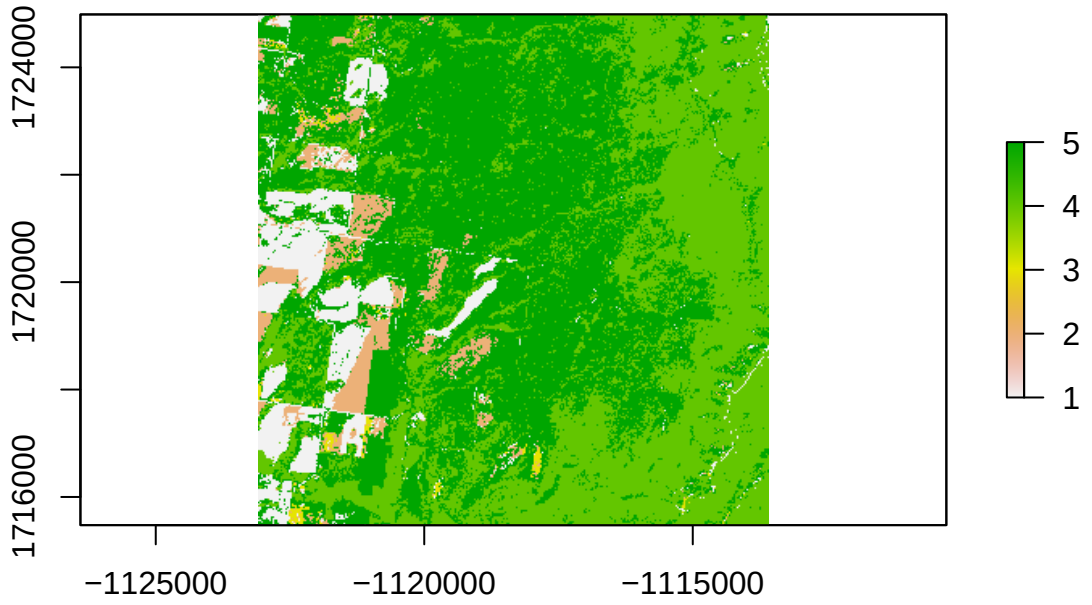


```
#encompasses all locations
muleysbuffSP2 <- gBuffer(mbuff.spdf,width=1000)
plot(muleysbuffSP2)
plot(mbuff.spdf,add=T)
points(muleys.spdf,col="red")
```

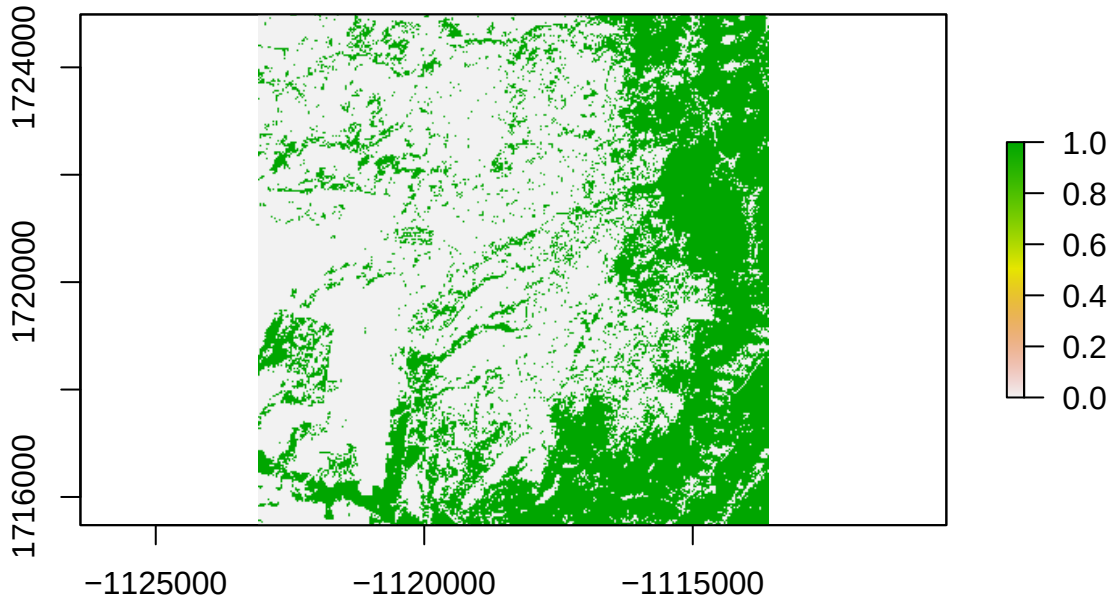


8. We will start here by creating covariate layers specifically for the year of interest, in this case crop data from NRCS for 2011

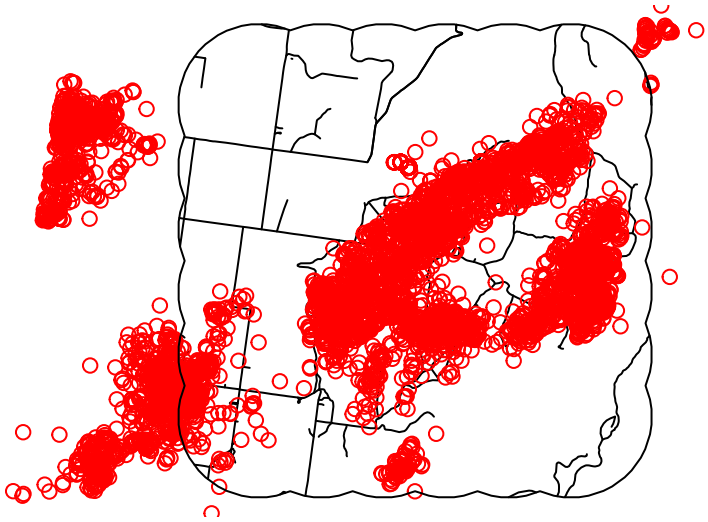
```
# Then clip buffer from crop11 layer
crop11 <- raster("crop11clip.tif")
mulcrop11clip<-crop(crop11, muleysbuffSP2)
#Crop categories
#1 = Sunflower,summer crops, random crops, grassland
#2 = Winter crops
#3 = Alfalfa
#4 = Forest
#5 = Shrubland

#Reclassify into 5 habitat categories
m11 <- c(-Inf,0,NA, 5.5, 6.5, 1, 22.5, 24.5, 2, 26.5, 27.5, 2, 29.5, 30.5, 2, 35.5, 36.5,
  3, 3.5, 4.5, 1, .5, 1.5, 1, 11.5, 12.5, 1, 27.5, 28.5, 1, 31.5, 33.5, 1,  42.5, 43.5, 1,
  47.5, 49.5, 1, 58.5, 59.5, 1, 60.5, 61.5, 1, 65.5, 69.5, 1, 76.5, 77.5, 1, 110.5,
  111.5, 1, 120.5, 124.5, 1, 130.5, 131.5, 1, 189.5, 190.5, 1, 194.5, 195.5, 1, 228.5,
  229.5, 1, 140.5, 143.5, 4, 170.5, 171.5, 1, 180.5, 181.5, 1, 36.5, 37.5, 1, 151.5,
  152.5, 5, 41.5, 42.5, 1, 204.5, 205.5, 1, 230,Inf,NA)
rclmat11 <- matrix(m11, ncol=3, byrow=TRUE)
crop11rc <- reclassify(mulcrop11clip, rclmat11)
plot(crop11rc)
```




```
##Create cover layer
cov <- c(-Inf,0,NA, 1, 140, 0, 140.5, 143.5, 1, 151.5,
  205.5, 0, 230,Inf,NA)#cover=forest only
rclmatcov <- matrix(cov, ncol=3, byrow=TRUE)
cover <- reclassify(mulcrop11clip, rclmatcov)
plot(cover)
```



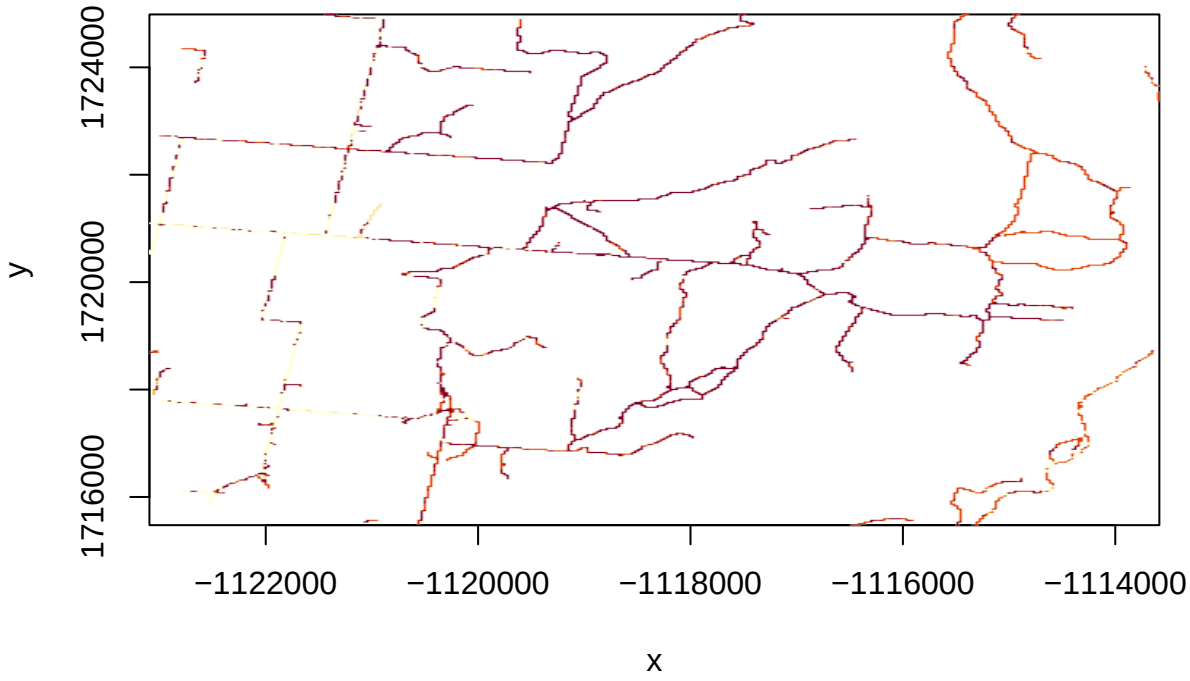

```
coverdf <- as.data.frame(as(cover,"SpatialGridDataFrame"))
coveronly <- subset(coverdf,coverdf$crop11clip=="1")
coveronly <- coveronly[c(-1)]
d_cover <- distanceFromPoints(crop11rc,coveronly)

##Bring in roads layer
roads<-readOGR(dsn=".",layer="AlbersRoads", verbose = FALSE)

#Clip using muleysbuffSP
roadclip <- crop(roads, muleysbuffSP2)
cliproads <- gIntersection(roads, muleysbuffSP2, byid=TRUE)

plot(roadclip)
points(muleys.spdf, col="red")
plot(muleysbuffSP2, add=TRUE)
```



9. The downside of creating distance to roads with spatstat in Exercise 8.2 is that it is not in a raster so we need to create a raster of distance to roads for every raster cell in layer1 before grabbing values or making our predictive surface.

We will start by using the Rasterize function to create a raster of the road shapefile with crop data used as a mask. A mask will give the spatial resolution and projection information to the raster you plan to create.

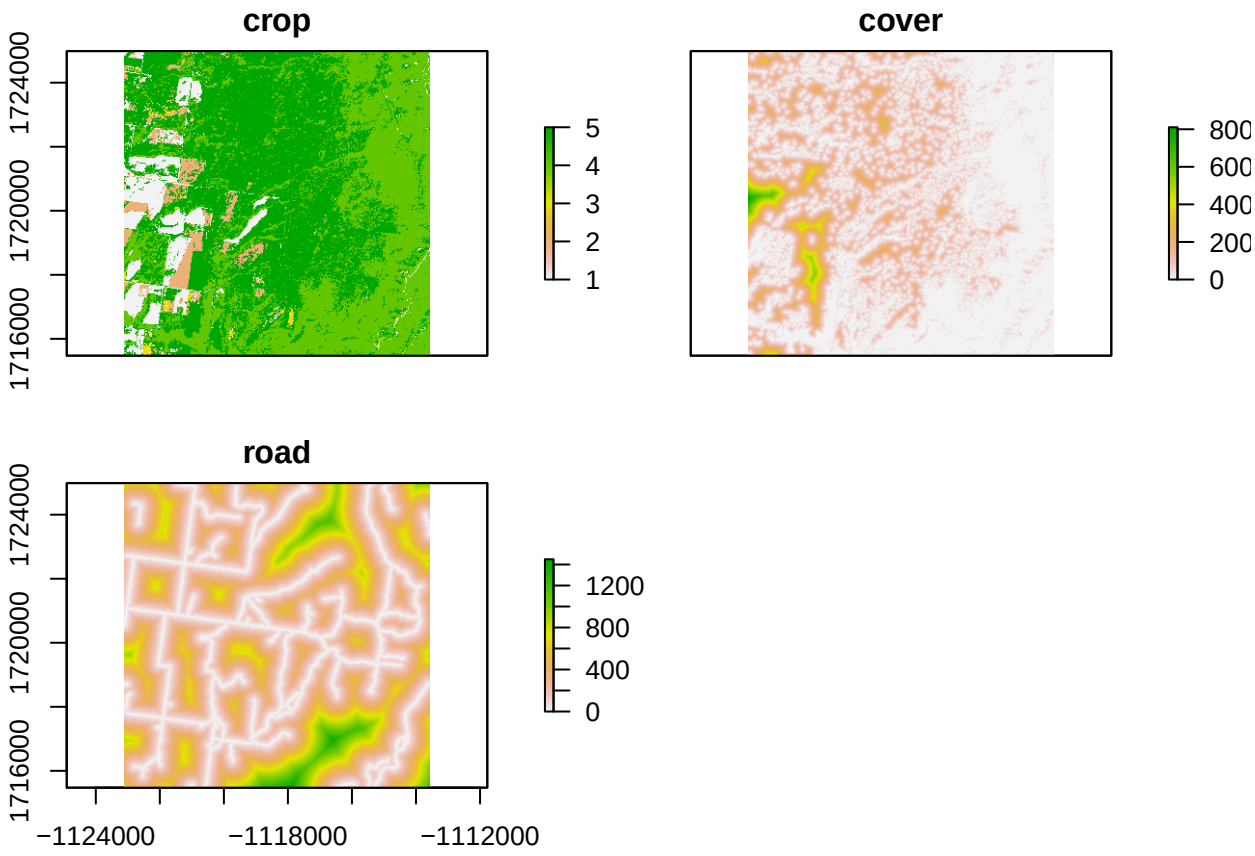
```
roadrast <- rasterize(roadclip,crop11rc, mask=TRUE)
image(roadrast)
```




```
#Now make a dataframe of all raster cell locations for each road segment
roadrastdf <- as.data.frame(as(roadrast, "SpatialGridDataFrame"))

#Remove the non-xy column of the data frame
roadrastdf <- roadrastdf[c(-1)]
#Use raster package to create distance from each raster
#cell to each road layer raster cell.
d_roadrast <- distanceFromPoints(crop11rc, roadrastdf)

##MAKE ALL RASTER LAYERS DATAFRAMES TO COMBINE LATER
crop11df <- as.data.frame(as(crop11rc, "SpatialGridDataFrame"))
#Distance to cover
d_covdf <- as.data.frame(as(d_cover, "SpatialGridDataFrame"))
#Distance to roads
final_roaddf <- as.data.frame(as(d_roadrast, "SpatialGridDataFrame"))

#Combine data frames for Crop and Distance to Cover and Roads
layers1 = cbind(crop11df, d_covdf,final_roaddf)
layers1 = layers1[,-c(2,3,5,6)]
names(layers1) = c("crop","d_cover","d_roads","x", "y")
#write.table(layers1,"layer1.txt",sep=",",col.names=TRUE, quote=FALSE)
#Now we need to extract all raster layers, grid and create a stack of all rasters
r <- stack(list(crop=crop11rc, cover=d_cover, road=d_roadrast))

nlayers(r)
```

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

```
plot(r)
```




```
names(r)
```

```
## [1] "crop"  "cover" "road"
```

```
ext <- extract(r, mbuff.spdf, weights=TRUE, fun = mean)

#NOTE above that for each grid cell in the sampling grid layer (i.e., grid), the "extract" function res
```

10. Code below extracts by land cover category and determines how many cells of each type were in each sample circle.

```
ex_crop <- (extract(crop11rc, mbuff.spdf, byid=TRUE))

#Land Cover category and number of cells (30x30m raster cells)
tab <- lapply(ex_crop, table)
tab[[1]]
```

```
##
##   1   2   3   4   5
## 364 208 100 159 519
```

```
#  1   2   3   4   5
#163  79  37   1  48
tab[[18]]
```

```
##
```

```
##   1   4   5
##   6 464 887
```

```
#1   4   5
#6 464 887

#What issue do you notice with the above habitat categories?

##############################################
#Code here thanks to Tyler Wagner, PA Coop Unit, for creating this loop to summarize
#proportions of habitat within each grid cell
##############################################
##Created land use categories
lus <- 1:5
##### Loop through and append missing land use categories to each grid cell
ex_crop_new <- list()
for(i in 1:length(mbuff.spdf)[1] ){
  # Land use cats in a given cell
  temp1 <- unique(ex_crop[[i]])
  # Give missing category 999 value
  ma1 <- match(lus, temp1, nomatch = 999, incomparables = NULL)
  # Get location (category of missing land use type)
  miss <- which(ma1%in%999)
  ex_crop_new[[i]] <- c(ex_crop[[i]], miss)
}

# New summary of land use in a grid cell
tab2 <- lapply(ex_crop_new, table)
tab2[[18]]
```

```
##
##   1   2   3   4   5
##   6   1   1 464 887
```

```
tab[[18]]
```

```
##
##   1   4   5
##   6 464 887
```

```
# Proportions of all land cover types per grid cell
prop <- list()
for(i in 1:length(mbuff.spdf)[1] ){
  prop[[i]] <- round((margin.table(tab2[[i]],1)/margin.table(tab2[[i]])),digits = 6)
}
#Function coredata is from the zoo package to convert the proportions from a list
#to a matrix
M <- coredata(do.call(cbind, lapply(prop, zoo)))
colnames(M) <- NULL
#Transpose matrix so land cover become separate columns of data
matrix <- t(M)
#Now convert the matrix to a data frame so it is easier to manipulate
dfland <- as.data.frame(matrix)
#Assing column names to land cover
colnames(dfland) <- c("sunflower","wintercrop","alfalfa","forest","shrub")
#Write out csv with new nlcd circle percents
```

```
#write.csv(dfland,paste(".", "circl_perc_nlcd.csv",sep=""))
```

11. Now that we have Land Cover in a similar format as the distance-to-derived data, we want to convert ext(the combined extracted rasters) into a data frame so it is easier to manipulate as well. The "extract" function in the raster package is supposed to be able to do this but does not work for some reason.

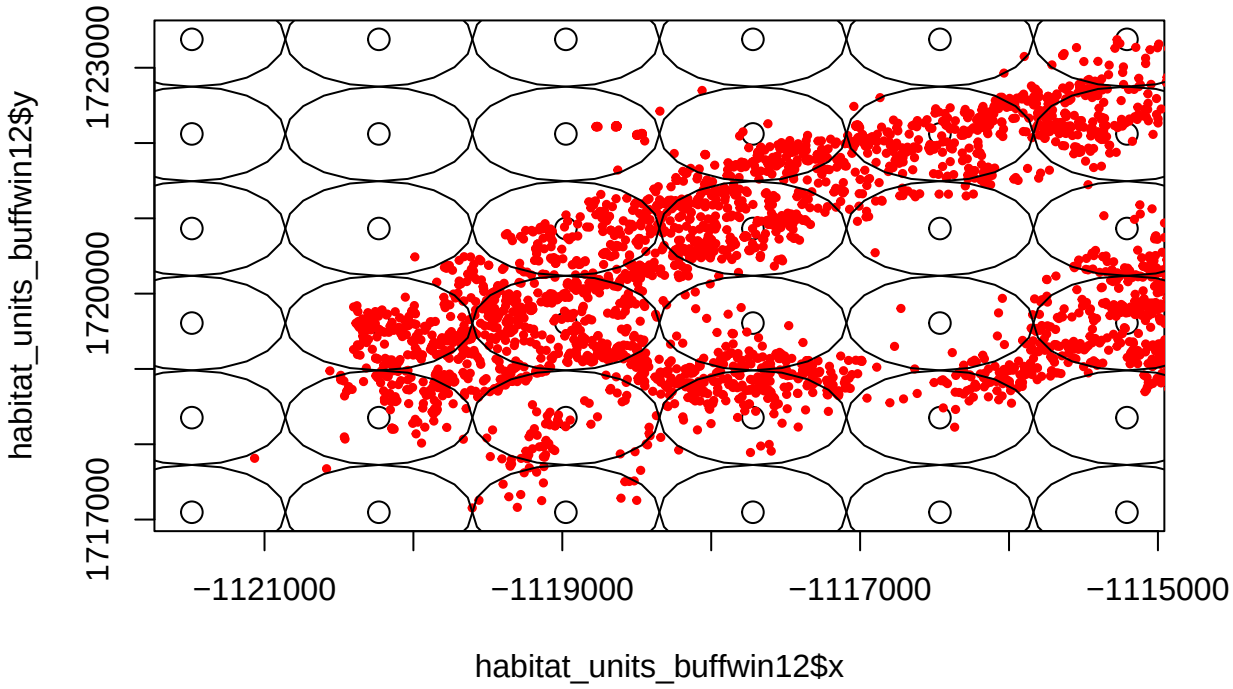
```
a <- as.data.frame(ext)
habitat_units_buffwin12 <- cbind(dfland, a)
```

12. Now we need to convert to a data frame for nb modeling. Read in animal_locations.txt or convert to data frame from above

```
#locations = read.table("deer_locations.txt", sep='\t', header=T)
locations.spdf <- crop(muleys.spdf,muleysbuffSP)
locations.df = as.data.frame(locations.spdf)
#locations.df = as.data.frame(winter2012)
locations <- locations.df[c(-1,-3:-24)]

#Add xy columns of circle centroids
mbuff.xy <- as.data.frame(grid.pts)
str(mbuff.xy)
```

```
## 'data.frame':    36 obs. of  4 variables:
##  $ x  : num  -1121488 -1120232 -1118976 -1117720 -1116464 ...
##  $ y  : num  1717097 1717097 1717097 1717097 1717097 ...
##  $ x.1: num  -1121488 -1120232 -1118976 -1117720 -1116464 ...
##  $ y.1: num  1717097 1717097 1717097 1717097 1717097 ...
```

```
habitat_units_buffwin12$x <- mbuff.xy$x
habitat_units_buffwin12$y <- mbuff.xy$y

plot(habitat_units_buffwin12$x, habitat_units_buffwin12$y, type='p', cex=1.5)
points(locations$x, locations$y, col="red", cex=0.5, pch=19)
plot(mbuff.spdf, add=T)
```



13. Calculate number of animal locations in each sampled habitat unit(see code in "count_locations.R").

```
# Source code file containing functions used below.
source("count_locations.R")

pooled.locations = locations
colnames(pooled.locations) <- c("ID","x","y")
pooled.locations$ID = 1
NB = F.count.relocations(locations.df = pooled.locations,
    habitat.units.df = habitat_units_buffwin12,
    habitat.unit.size = 628)
# List of column names:
names(NB)
```

```
##  [1] "sunflower"   "wintercrop"  "alfalfa"     "forest"      "shrub"
##  [6] "crop"        "cover"       "road"        "x"           "y"
## [11] "animal.ID"   "n.locations" "total"
```

```
#Look at the range in number of locations in our sample circles
summary(NB$n.locations)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00   18.00   82.78  131.00  521.00
```

```
#Now run a population-level model for a few covariates (forest, road). NOTE: If you run models for each
nb = glm.nb(n.locations ~ offset(log(total)) + forest + road, data=NB)
summary(nb)
```

```
##
```

```
## Call:
## glm.nb(formula = n.locations ~ offset(log(total)) + forest +
##     road, data = NB, init.theta = 0.1861383305, link = log)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.58377  -1.47378  -0.63470   0.09756   1.34567
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.698439   0.770733  -3.501 0.000463 ***
## forest      -0.170172   1.567555  -0.109 0.913552
## road        -0.003922   0.002156  -1.819 0.068947 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.1861) family taken to be 1)
##
##     Null deviance: 38.907  on 35  degrees of freedom
## Residual deviance: 37.642  on 33  degrees of freedom
## AIC: 322.13
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.1861
##           Std. Err.:  0.0462
##
##  2 x log-likelihood:  -314.1250
```

```
#-------------------------------------------------------------------------------
# Proportion of 0 counts in data
sum(NB$n.locations == 0)/nrow(NB)
```

```
## [1] 0.3888889
```

```
nb.density = structure(
function # Probability mass function for NB2

# Description: This function gives the probability that a discrete random variable, X,
#is exactly equal to some value  according to a NB2 distribution.
# Returns: Pr(X=k)

(k,
### value at which to estimate probability
### Pr(X=k)
mu,
### NB2 estimate of mu
theta
### NB2 estimate of theta
){

    (gamma(theta+k)/(gamma(theta)*factorial(k)))*
        (mu^k)*(theta^theta)/
        ((mu+theta)^(theta+k))
```

```
})
# Expected proportion under NB2 model
nb.density(k=0, mu=mean(NB$n.locations), theta=0.1861)
```

```
## [1] 0.321362
```

```
             # (Note: use estimated theta of the model output found in summary statement above)
# The value above can be interpreted as:
# "A NB2 distribution with theta=0.1861 and mu=0.9196 should have an average of 32% zero values"

#Observed
zero = NB$n.locations == 0
sum(zero)   #total number of zeros
```

```
## [1] 14
```

```
mean(zero)  #proportion that are zeros
```

```
## [1] 0.3888889
```

```
#Expected based on NB distribution and our observed over-dispersions
theta = mean(NB$n.locations)^2 / (var(NB$n.locations) - mean(NB$n.locations))
check = rnegbin(n=10000, mu=mean(NB$n.locations), theta=theta)
check.zeros = check == 0
mean(check.zeros)
```

```
## [1] 0.0966
```