1.7 Manipulate Raster Data Layer

Manual of Applied Spatial Ecology

3/11/202

1. Exercise 1.7 - 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 "RasterScript.R" in that folder and run code directly from the script. Create an Ascii file from your raster grid in ArcMap 10.X if experimenting with your own raster using the following Toolbox in ArcMap 10.X:

ArcToolbox - Conversion Tools - From Raster - Raster to Ascii

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

```
#install.packages(c("adehabitatHR","maptools","raster", "rgdal"))

library(adehabitatHR)
library(raster)
library(rgdal)
library(maptools)
```

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

```
#CRS of shapefile layers
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")
#CRS of raster layers
crs2 <- 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")
utm.crs<-"+proj=utm +zone=17N +ellps=WGS84"
```

6. Alternatively, you can set your working directory by opening a previously saved workspace by double clicking it in that folder which will also serve to set that folder as the working directory. All of the raster layers we are going to use will be located here

Note: Most of the exercises that follow are exploratory in nature and not the recommended way to bring raster data into R. I include these exercises only for those that may perhaps only have alternate rasters available (e.g., ASCII). I would recommend using .tif and not ASCII or .txt files as in the first few exercises below.

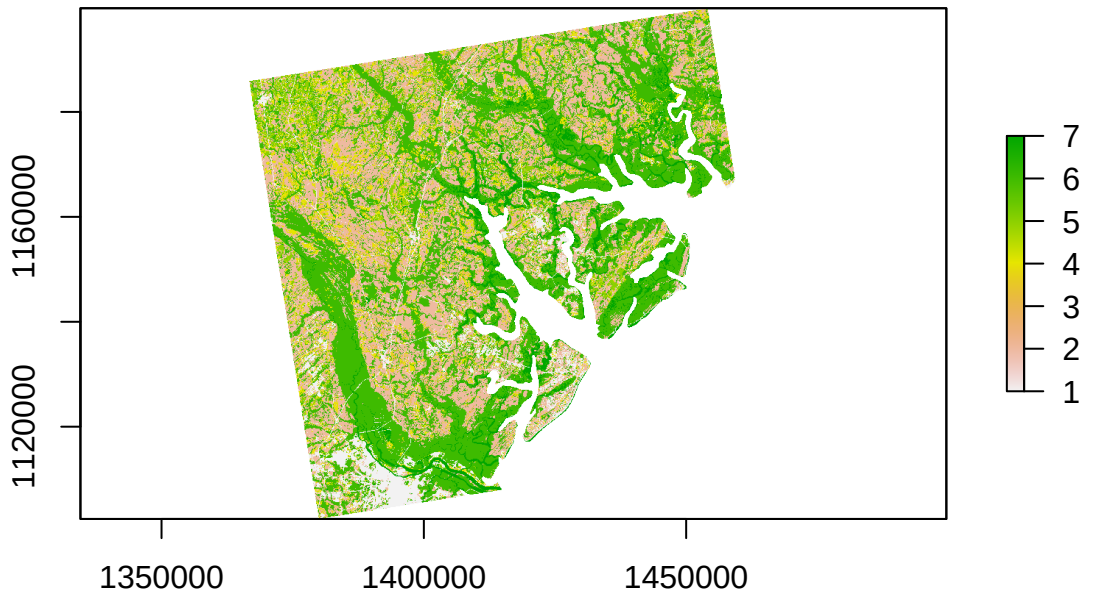
7. If you have troubles getting a raster to Ascii in ArcMap to actually show up as an Ascii file there is good reason. We need to rename the text file by replacing ".txt" with ".asc" in Windows Explorer. ArcMap will not save as an ".asc" file and I have no idea why!

8. Here is some code to import Ascii files (i.e., rasters) from ArcMap into R using one of several packages. Ascii files can be categorical (Vegetation/Habitat categories) or numeric (DEMs). Because adehabitat package has been discontinued, we will skip this section and move on to step 9.

```
#Import raster as text using the "raster" package
r <-raster("polyascii2.txt")
proj4string(r)#Note: No projection information was imported with the raster
```

```
## [1] NA
```

```
plot(r)
```




```
#Assign projection information for imported text file
proj4string(r) <-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")
r
```

```
## class      : RasterLayer
## dimensions : 3245, 3353, 10880485  (nrow, ncol, ncell)
## resolution : 30, 30  (x, y)
## extent     : 1366773, 1467363, 1102414, 1199764  (xmin, xmax, ymin, ymax)
## crs        : +proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0 +ellps=GRS80 +units
## source     : polyascii2.txt
## names      : polyascii2
## values     : -2147483648, 2147483647  (min, max)
```

```
proj4string(r)
```

```
## [1] "+proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0 +ellps=GRS80 +units=m +no_
```

```
#Import raster as an Ascii file (factor) using "adehabitat" package if available for
#file1, file2, and file3 in the 3 sections of code below:
```

```
#Path of the file to be imported
# file1 <-  paste("polyextract2.asc", sep = "\\")
# levelfile <- paste("TableExtract.txt", sep = "\\")
# asp <- import.asc(file1, lev = levelfile, type = "factor")
# image(asp)
# asp
#
# #Now let's look at the vegetation categories of the file
# ta <- table(as.vector(asp))
# names(ta) <- levels(asp)[as.numeric(names(ta))]
# ta
#
# file2 <-  paste("polyclip.asc", sep = "\\")
# levelfile2 <- paste("TableClip.txt", sep = "\\")
# asp2 <- import.asc(file2, lev = levelfile2, type = "factor")
# image(asp2)
# asp2
#
# #Shows 7 vegetation categories
#
# #Now let's look at the vegetation categories of the file
# ta2 <- table(as.vector(asp2))
# names(ta2) <- levels(asp2)[as.numeric(names(ta2))]
# ta2

#NOTE: R won't recognize double digit veg categories
#Import raster as an Ascii files (factor) using:
#Path of the file to be imported
# file3 <-  paste("polyascii2.asc")
# levelfile3 <- paste("TableCode.txt")
# asp3 <- import.asc(file3, lev = levelfile3, type = "factor")
# image(asp3)
# asp3

#Now let's look at the vegetation categories of the file
# ta3 <- table(as.vector(asp3))
# names(ta3) <- levels(asp3)[as.numeric(names(ta3))]
# ta3

#Or we can also use the "adehabitat" package to import DEM
# fileElev <-  paste("demascii.asc")
# elev <- import.asc(fileElev)
# image(elev)
# elev
```

9. We will primarily be using the "rgdal" package to import an ascii grid as a Spatial Grid Data Frame

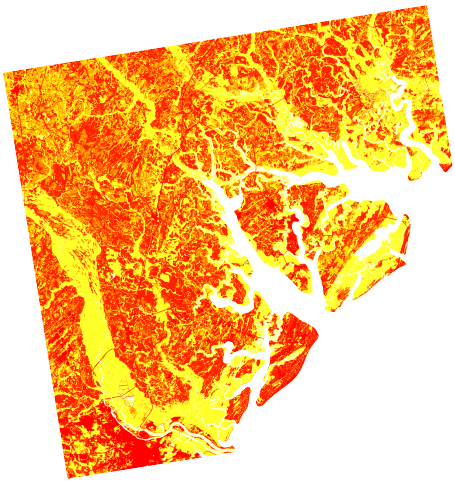
```
habitat <- readGDAL("polyascii2.asc")
```

```
## polyascii2.asc has GDAL driver AAIGrid
## and has 3245 rows and 3353 columns
```

```
proj4string(habitat)
```

```
## [1] NA
```

```
proj4string(habitat) <-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")
image(habitat)
```



10. Now let's add some shapefiles to our raster

```
#Load county shapefile
county<-readOGR(dsn=".",layer="BeaufortCoAlbers", verbose = FALSE)
proj4string(county)
```

```
## [1] "+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_
```

```
#Now let's make county a SpatialPolygon class to simply remove data contained within it
polys <- as(county, "SpatialPolygons")
plot(polys,lwd=2)
text(coordinates(polys), labels="Beaufort")

#Load airport runway shapefile
run<-readOGR(dsn=".",layer="RunwayAlbers", verbose = FALSE)
proj4string(run)
```

```
## [1] "+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_
```

```
polys2 <- as(run, "SpatialPolygons")
plot(polys2,add=T,lwd=2)
text(coordinates(polys2), labels="Runway")

#Load aircraft flight pattern shapefile
```

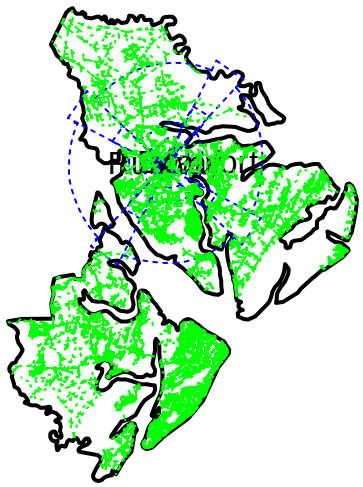
```
path<-readOGR(dsn=".",layer="FlightImage", verbose = FALSE)
proj4string(path)
```

```
## [1] "+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_
```

```
polys3 <- as(path, "SpatialLines")
plot(polys3,add=T,lty="32", col="blue")

#Load aircraft flight pattern shapefile
road<-readOGR(dsn=".",layer="CountyRoadAlbers", verbose = FALSE)
proj4string(road)
```

```
## [1] "+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_
```

```
polys4 <- as(road, "SpatialLines")
plot(polys4,add=T,lty="22", col="green")
```

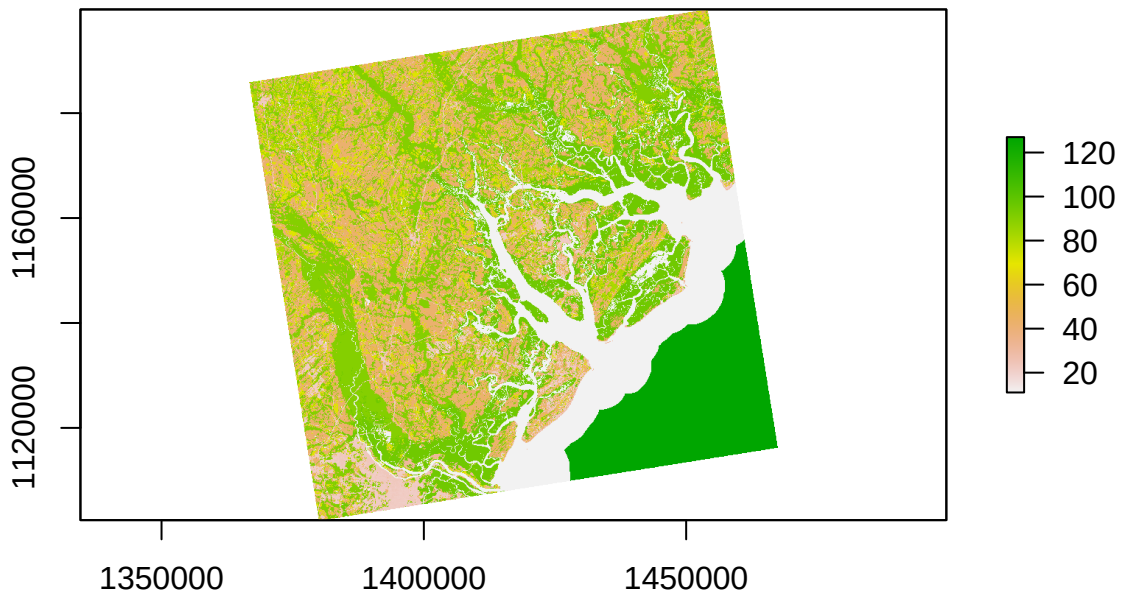


11. Plot out all the shapefiles overlayed on each other with and without the raster.

```
plot(county)
plot(road, add=T)
plot(run, col="red",add=T)
plot(path, col="blue",add=T)
```

12. Let's reclassify layer to get fewer vegetation categories to make it easier to work with the raster.

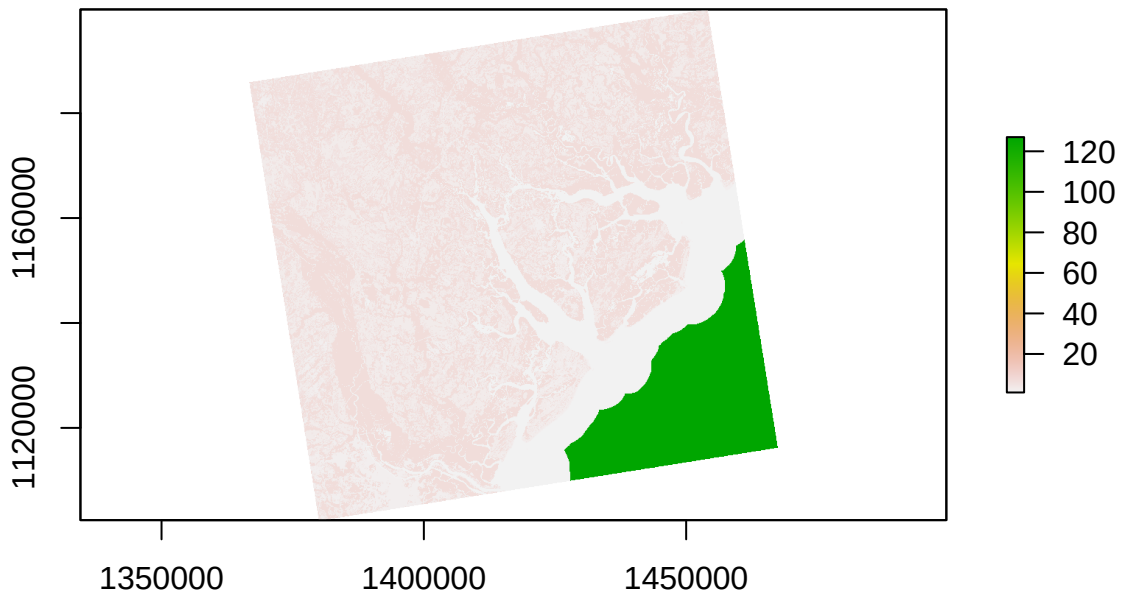
```
veg <-raster("polydouble.txt")
plot(veg)
```



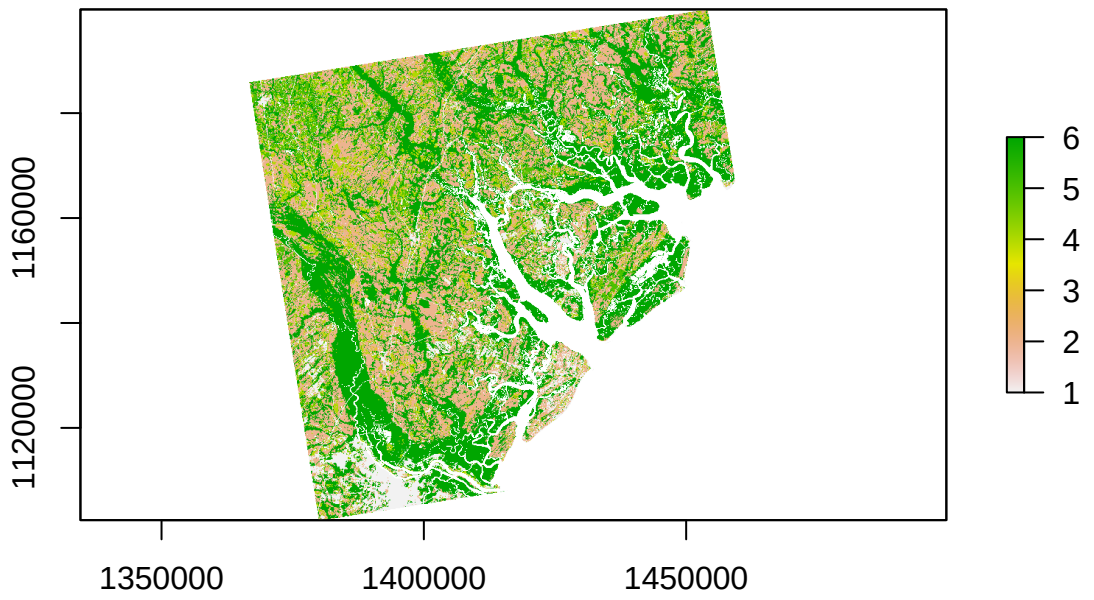

```
veg
```

```
## class      : RasterLayer
## dimensions : 3245, 3353, 10880485  (nrow, ncol, ncell)
## resolution : 30, 30  (x, y)
## extent     : 1366773, 1467363, 1102414, 1199764  (xmin, xmax, ymin, ymax)
## crs        : NA
## source     : polydouble.txt
## names      : polydouble
## values     : -2147483648, 2147483647  (min, max)
```

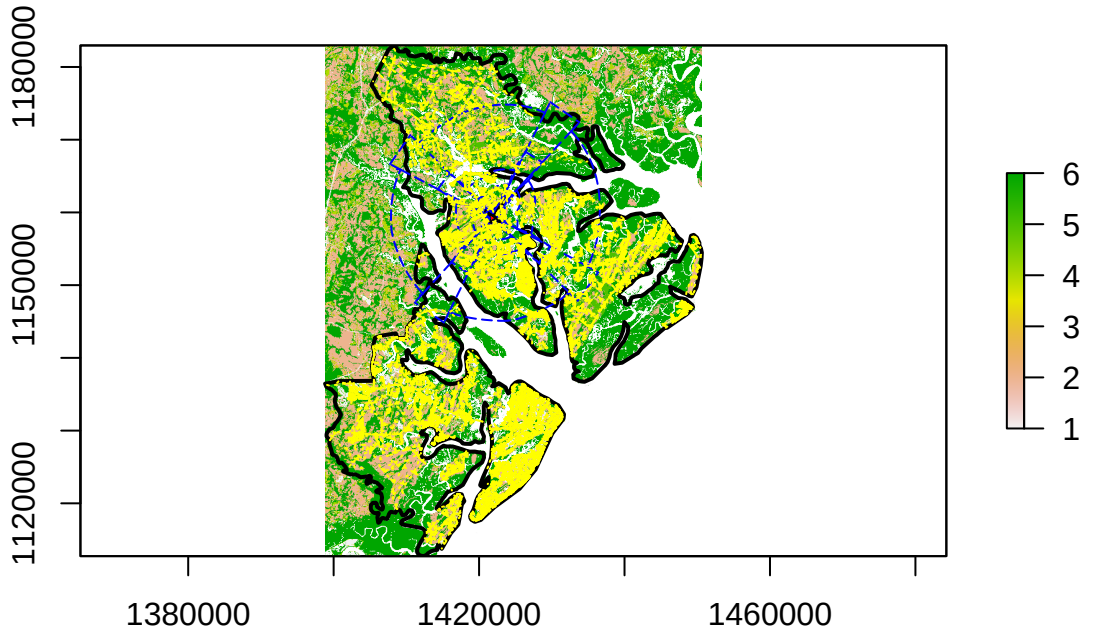
```
# reclassify the values into 7 groups
# all values between 0 and 20 equal 1, etc.
m <- c(0, 19, 1, 20, 39, 2, 40, 50, 3, 51, 68, 4, 69,79, 5, 80, 88, 6, 89, 99, 7)
rclmat <- matrix(m, ncol=3, byrow=TRUE)
rc <- reclassify(veg, rclmat)
plot(rc)
```




```
#Now, let's remove water that is coded 11 and No Data that is coded as 127
m1 <- c(0, 19, NA, 20, 39, 1, 40, 50, 2, 51,68, 3, 69, 79, 4, 80, 88, 5, 89, 99, 6, 100, 150, NA)
rclmat1 <- matrix(m1, ncol=3, byrow=TRUE)
rc1 <- reclassify(veg, rclmat1)
plot(rc1)
```




```
#Clip the raster within the county polygon for a zoomed in view then plot
clip <- crop(rc1, polys)
plot(clip)
plot(polys,add=T,lwd=2)
plot(polys2,add=T,lwd=2, col="red")
plot(polys3,add=T,lty="62", col="blue")
plot(polys4,add=T,lty="22", col="yellow")
```



13. We can load some vulture locations to extract landcover that each location occurs in that will be considered "used" habitat in resource selection analysis.

```
#Import bird 49 locations to R
bv49 <-read.csv("Bird49.csv", header=T)#How many bird locations?

#Make a spatial points data frame of locations and convert to Albers
coords<-data.frame(x = bv49$x, y = bv49$y)
bvspdf <- SpatialPointsDataFrame(coords= coords, data = bv49, proj4string = CRS(utm.crs))
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
## = prefer_proj): Discarded datum Unknown based on WGS84 ellipsoid in Proj4
## definition
```

```
plot(bvspdf, col="red")
#NOTE: Locations must be assigned to the UTM coordinate system prior to projection to Albers
#so won't overlay on veg layer at this point because veg is in Albers
bv49Albers <-spTransform(bvspdf, CRS=crs2)
class(bv49Albers)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
proj4string(bv49Albers)
```

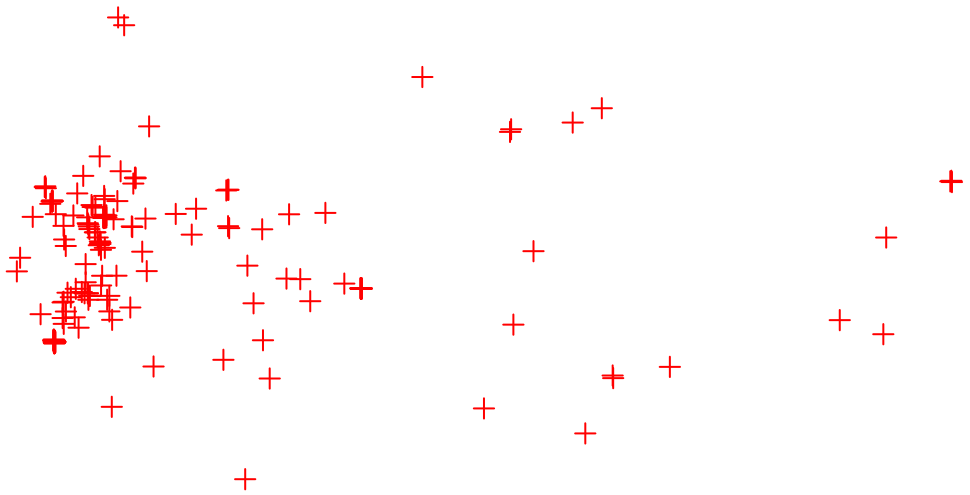
```
## Warning in proj4string(bv49Albers): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```
## [1] "+proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0 +ellps=GRS80 +units=m +no_
```

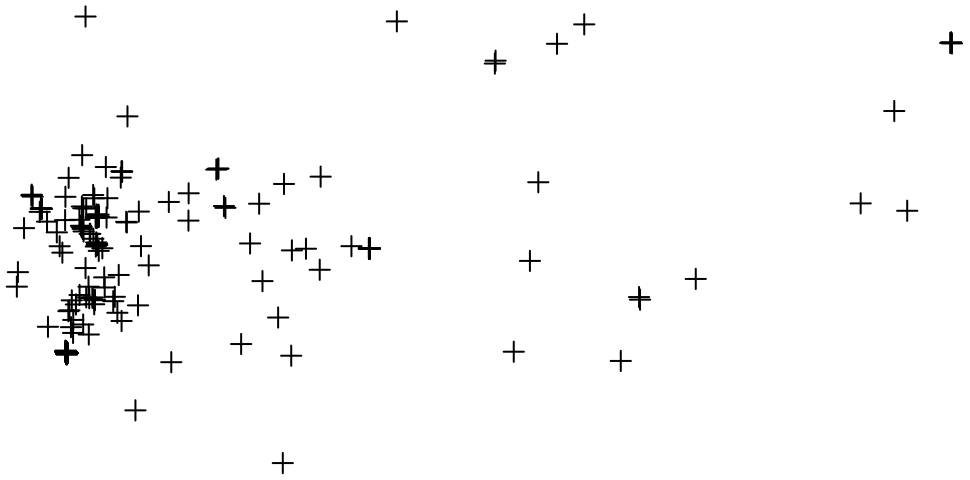
```
bv49Albers[1:5,]
```

```
## class       : SpatialPointsDataFrame
## features    : 5
## extent      : 1415830, 1418301, 1152353, 1156708  (xmin, xmax, ymin, ymax)
## crs         : +proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0 +ellps=GRS80 +uni
## variables   : 4
## names       :     Date, id,      x,       y
## min values  : 20061001, 49, 519067, 3587085
## max values  : 20061030, 49, 521821, 3591245
```

```
points(bv49Albers, col="red")
```



```
#Determine which of those points lie within a cell that contains data by using the extract function.
#The extract function will extract covariate information from the raster at a particular the xy coordin
veg.survey<-extract(clip, bv49Albers)
veg.survey<-subset(bv49Albers,!is.na(veg.survey))
plot(veg.survey, col="black")
```

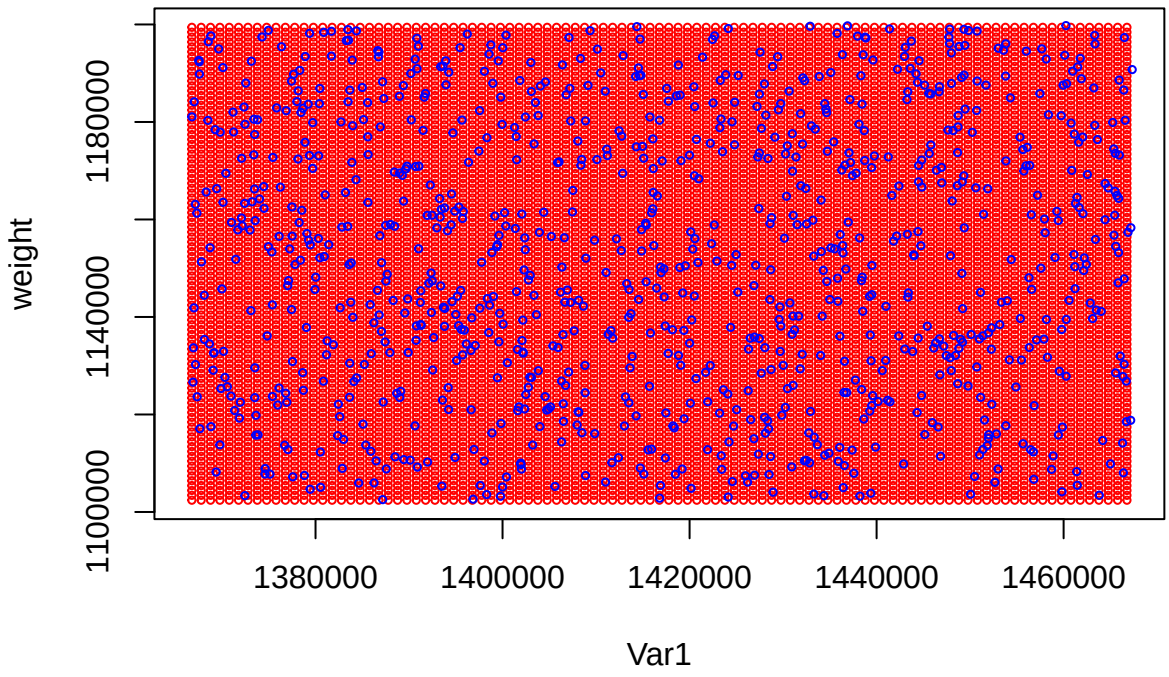


14. We can also create some random points within the extent of the area to be considered as "available" habitat.

```
##First we need to create a grid across the study site with sample points
Sample.points<-expand.grid(seq(veg@extent@xmin, veg@extent@xmax, by=1000),
  weight = seq(veg@extent@ymin, veg@extent@ymax, by=1000))
plot(Sample.points, bg="red", cex=.5,col="red")

#Now create some random points using the minimum and maximum coordinates of the raster to
#determine the range of points from which to select x and y
x.pts<-sample(seq(veg@extent@xmin, veg@extent@xmax, by=10),1000) ##generate x coordinates for random po
y.pts<-sample(seq(veg@extent@ymin, veg@extent@ymax, by=10),1000)

#Now create a spatial points file from the randomly generated points
coords2<-data.frame(x = x.pts, y = y.pts)
head(coords2)
```

```
##         x       y
## 1 1404583 1188204
## 2 1428163 1181394
## 3 1464773 1166484
## 4 1380883 1198384
## 5 1452813 1116004
## 6 1395923 1137284
```

```
points(coords2, bg="red", cex=.5,col="blue")
```



```
#Determine which of those points lie within a cell that contains data by using the extract
#function agin.
veg.sample<-extract(rc1, coords2)
head(veg.sample)
```

```
## [1]  5  2 NA NA NA  2
```

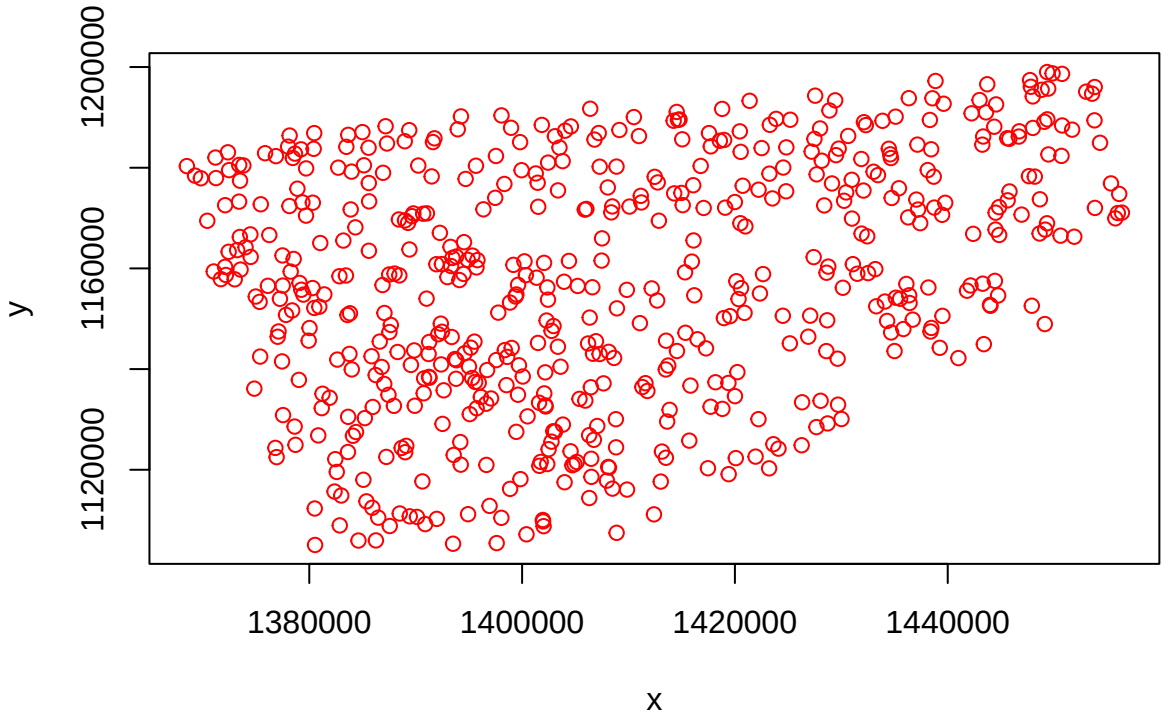
```
veg.sample<-subset(coords2,!is.na(veg.sample))
str(veg.sample)
```

```
## 'data.frame':    576 obs. of  2 variables:
##  $ x: num  1404583 1428163 1395923 1434093 1444833 ...
##  $ y: num  1188204 1181394 1137284 1153414 1166644 ...
```

```
head(veg.sample)
```

```
##          x       y
## 1  1404583 1188204
## 2  1428163 1181394
## 6  1395923 1137284
## 8  1434093 1153414
## 9  1444833 1166644
## 10 1438383 1147464
```

```
plot(veg.sample,col="red")
```



15. We can also do the same using the clipped vegetation raster to be more in line with vulture locations or using the reclassified vegetation categories. For each locations, we can determine if a locations lies within a cell that contains data by using the extract function and this will extract covariate information from the

raster at a each location.

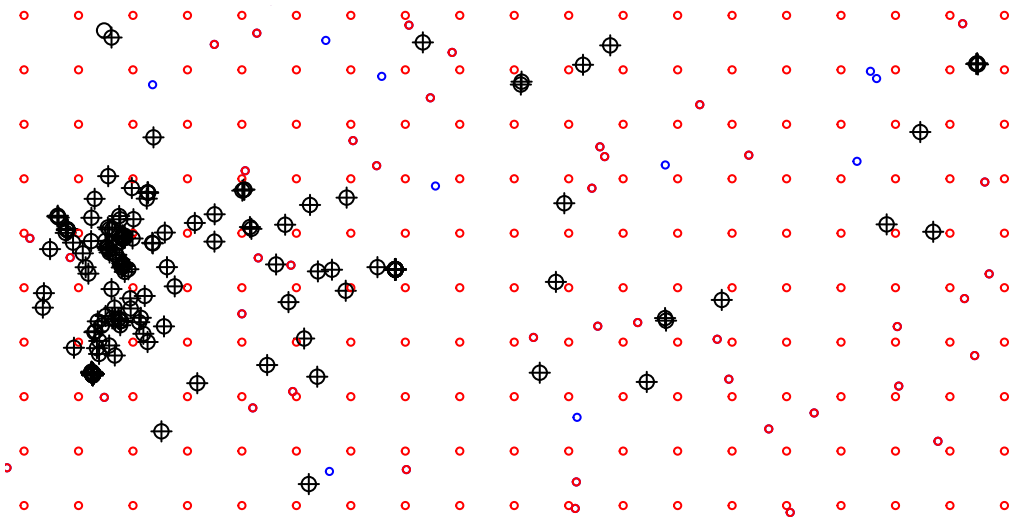
```
clip.survey<-extract(clip, bv49Albers)
clip.survey
```

```
##   [1]  1  2  2  6  6  6  6  6  6  6  6  6  6  6  6  6  6  4  6  6  6  6  2  6  6
##  [26]  6  6  6  6  6  6  6  6  2  2  6  6  6  4  2  6  6  6  6  6  6  6  6  6  2
##  [51]  6  6  6  2  2  6  6  6  6  1  6  6  6  6  6  6  6  6  6  6  6  6  6  6  2
##  [76]  6  6  6  6  6  6  6  6  3  6  6  2  2  6  6  6  6  6  6  3  6  2  2  6  6
## [101]  6  6  6  6  6  6  6  6  2  2  4  2  3  6  6  6  6  4  2  2  2  6  6  6  6
## [126]  1  2  6  1  2  6  6  6  6  6  6  6  6  6  6  6  6  4  3  2  6  2  6  6  2  4
## [151]  6  6  1  1  5  4  2  3  3  2  2  2  1  2  2  2  1  1  6  1  2  1  2  2  2
## [176]  2  2  6  2  6  2  6  2  1  2  2  2  2  2  1  1  4  4  4  4  4  4  1  4  6
## [201]  2  4  1  4  4  4  5  4  2 NA  4  3  3  6  6  6  2  6  2  2  5  2  4  5  2
## [226]  3  6  6  6  6  6  6  2  2  6  4  6  4  1  1  1  2  1  6  6  4  5  4  4  2
## [251]  2  2  1  2  4  4  2  4  2  2
```

```
clip.survey<-subset(bv49Albers,!is.na(clip.survey))
plot(clip.survey, col="black")

#Create a regular grid and do the same thing
Sample.points2<-expand.grid(seq(clip@extent@xmin, clip@extent@xmax, by=1500),
  weight = seq(clip@extent@ymin, clip@extent@ymax, by=1500))
points(Sample.points2, bg="red", cex=.5,col="red")

#Create random points using the minimum and maximum coordinates of the raster
x.pts2<-sample(seq(clip@extent@xmin, clip@extent@xmax, by=10),500)
y.pts2<-sample(seq(clip@extent@ymin, clip@extent@ymax, by=10),500)

#Now create a spatial points file from the randomly generated points
coords3<-data.frame(x = x.pts2, y = y.pts2)
points(coords3, bg="red", cex=.5,col="blue")

#Determine which of those points lie within a cell that contains data by using the extract
#function.
clip.sample<-extract(clip, coords3)
head(clip.sample)
```

```
## [1] NA  6 NA  6 NA  3
```

```
clip.sample<-subset(coords3,!is.na(clip.sample))
points(clip.sample, cex=.5, col="red")
points(bv49Albers)
```



New code addition below (Update 1/7/2021) to use mapview package to treat plotting data layers as done with GIS software. This package enables user to zoom in and out and scroll around layer after calling a plot function

```
library(mapview)
mapview(clip)
mapview(polys) + mapview(bv49Albers, color="yellow")
mapview(polys) + bv49Albers + polys4
```