1 Introduction

This script will guide you through the extraction of GIS-based covariates for Moss Sanctuary

Script created by Morgan Tingley

The ultimate goal of this exercise is to create a dataset that includes the distance from survey points within Moss Sanctuary to the nearest river and the nearest road as well as the elevation at each point. In the process, it walks you through loading spatial files (both shapefiles and rasters), dealing with projections, plotting spatial files, calculating distances, and extracting values.

```
#Load packages
library(rgdal)
library(rgeos)
library(raster)
#-------------------------------------------------------------------------------
#Load data
# getwd()
# setwd("~Documents/Rfolder/GIS_in_R/Layers")
# you need to put the files in a subdirectory of the place where this file is, or provide full paths
#We use the command readOGR() for vector data and raster() for raster data.
?readOGR
?raster
#For example:
moss <- readOGR(dsn = getwd(), layer = "MossSanctuary")
moss
#Try loading in hydrography and roads
hydro <- readOGR(dsn = getwd(), layer = "hydro")
roads <-
 #Now try a raster
 dem <- raster("DEM_mansfield.tif")
#Finally, read in the point file, which is just a .csv file
points <- read.csv("sample_points.csv")
points
#Notice that this is just a data file of points with X and Y coordinates
#We will make these points into spatial data later

#-------------------------------------------------------------------------------
#Viewing and understanding data

#Are these are all the same projection? What are their projections?
projection(moss)
projection(hydro)
projection(roads)
projection(dem)
#Moss isn't exactly the same. It's best to make sure they match. Let's set Moss the same as the othe
#Use the function spTransform()
?spTransform
moss <- spTransform(moss, CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"))
#OK, now we're good to go!
#The first step is to look at what these layers are
plot(moss)
plot(hydro)
plot(roads)
plot(dem)
#Now, let's change the look and overlay our plots
plot(moss)
lines(hydro, col="blue")
lines(roads, col="gray", lty=5) # slow
```

```
#But GIS data are *more* than just maps. They also contain data
data.frame(roads)
data.frame(hydro)
#And these things can be queried
plot(roads, col = "#00000080")
lines(roads[which(roads$FULLNAME=="State Hwy 195"), ], col = "red", lwd = 3)
#-------------------------------------------------------------------------------
#Importing point data

#Look at our point file
head(points)
plot(points[, 2:3])
#This isn't spatial data yet, although we can plot it as coordinates. We need to tell R that it's sp
pts <- points[, c(1, 4)]
#We tell it that it's spatial, by assigning coordinates
coordinates(pts) <- points[ ,c(2,3)]
plot(pts)
#But what coordinate system is it?
projection(pts)
#We need to tell R that it's lat/lon in WGS84 (which our GPS were set to)
#Luckily, our other data are already in this projection, so we can borrow the projection from those
projection(pts) <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
#Before we used sptransform to change the projection of a layer that already had a projection define
#projection() allows us to define the projection of a layer
plot(pts)
# It looks a little different, but most importantly we can now overlay it on our data
plot(moss)
lines(hydro, col = "blue")
lines(roads, col = "#00000080", lty = 2)
text(pts, labels = 1:79, cex = 0.7)
#-------------------------------------------------------------------------------
# Re-projecting data

# Our goal is to create gridded surfaces that show distance from road and distance from hydrography
# We then want to intersect these distance surfaces with our survey points
# So we need to create a gridded surface, but it's easier to do this if we change to a meter-based p
# the UTM system works great for this
# https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
# In this part of CT, we are in UTM zone 18

# We can define this projection as follows:
utm.prj <- CRS("+proj=utm +zone=18 +ellps=GRS80 +datum=NAD83 +units=m +no_defs")
# the CRS() function defines a projection in the PROJ.4 system

# Now, we can re-define out lines and polygons into the UTM system
# To do this, we use the function spTransform
?spTransform
# Create 3 new objects, that re-project our layers from lat-lon to UTM
moss.utm <- spTransform(moss, utm.prj)
roads.utm <- spTransform(roads, utm.prj)
#hydro.utm <-

# Check: If they don't all plot together, you've made an error
plot(moss.utm)
plot(hydro.utm, add = T, col = "blue", lwd = 3)
lines(roads.utm, col = "#00000080", lty = 2)
```

```
#-------------------------------------------------------------------------------
# Defining a grid

# In order to define a distance surface, we need a grid, for which to measure distance from every ce
# For our purposes, a 25-m x 25-m grid will do nicely.

# First we define the 'extent' of our grid. How big do we want it to be?
grid.ext <- extent(moss.utm)
grid.ext
plot(grid.ext)
# Next, we turn this extent into a raster layer using raster()
grid <- raster(grid.ext)
# And then we define the resolution as 25-m x 25-m
res(grid) <- c(25,25)
# What does this look like now?
grid
plot(grid)
# Our grid has no projection defined, but we know we made it in the UTM system, so we can just defin
projection(grid) <- projection(moss.utm)
# Finally, if we want to plot it to see what it looks like, we can convert to vector polygons
# use the function: rasterToPolygons()
grid.poly <- rasterToPolygons(grid)
# Check: If they don't all plot together, you've made an error
plot(moss.utm)
plot(hydro.utm, add = T, col = "blue", lwd = 3)
lines(roads.utm, col = "#00000080", lty = 2)
lines(grid.poly)
#-------------------------------------------------------------------------------
# Measuring distance to hydrography

# The time has come to measure distance. Here, we use the function gDistance from the package 'rgeos
?gDistance
# Let's start with hydrography
dist2hydro <- gDistance(spgeom1 = grid.poly, spgeom2 = hydro.utm, byid = T)
# This takes a few seconds

# Examine the output
dist2hydro
dim(dist2hydro)
# There are 1079 rows and 1406 columns. Where are these numbers coming from?
grid.poly
dim(hydro.utm)
# Our hydro layer has 1079 river segments, and the raster has 1406 cells. It calculates the distance
# EVERY piece of river in Mansfield!

# All we care about is the distance to the nearest river or pond, so we can simplify this using appl
dist2hydro <- apply(dist2hydro, 2, min)
# We now have the distance to the nearest hydrography for every cell

# The final step is inserting this into a raster layer, as these values aren't 'spatial' data yet

# First we create a copy of the raster 'grid'
dist2hydro.rast <- grid
# And then we just copy in the values we calculated
values(dist2hydro.rast) <- dist2hydro
# Did this work? Let's plot it to find out!
```

```
plot(dist2hydro.rast, col = cm.colors(50))
lines(moss.utm)
lines(hydro.utm)
# The values are from 0 to 500. What are the units? How do you know?

#-------------------------------------------------------------------------------
# Measuring distance to road

# This process can be easily repeated for the road layer Do it now and create objects "dist2road" ar
dist2road <- gDistance(spgeom1 = grid.poly, spgeom2 = roads.utm, byid = T)
dist2road <- apply(dist2road, 2, min)
dist2road.rast <- grid
values(dist2road.rast) <- dist2road
# Check!
plot(dist2road.rast)
lines(moss.utm)
lines(roads.utm, lty = 2)
#-------------------------------------------------------------------------------
# Extracting distance values for our points

# Remember that our goal is to derive these covariates for each of our points
# How do our points line up with what we've plotted?

plot(dist2road.rast)
points(pts)
# Wait, that doesn't work. What's wrong?
projection(dist2road.rast)
projection(pts)
# Remember to check for projections!  If they differ, set them equal
pts.utm <- spTransform(pts, utm.prj)
plot(dist2road.rast)
points(pts.utm)
# Extraction is pretty easy now, use the function extract()
pts.utm$dist2road <- extract(dist2road.rast, pts.utm)
pts.utm$dist2hydro <-

 # Did it work? Let's look:
 head(pts.utm)
# That was easy. Now we can explore our data:
hist(pts.utm$dist2hydro)
hist(pts.utm$dist2road)
plot(pts.utm$dist2hydro, pts.utm$dist2road, xlab = "Distance from water (m)", ylab = "Distance from
#-------------------------------------------------------------------------------
# Extracting elevation

# The final step is elevation. Let's look at the DEM again
plot(dem)
lines(moss)
# It's really large, so let's crop it to Moss Sanctuary first using crop()
dem.moss <- crop(dem, moss)
# Did that work?
plot(dem.moss, col = terrain.colors(100))
lines(moss)
points(pts)
# What is the lowest elevation? What is the highest elevation? What are the units?
```

```
# Now, all we have to do is extract these DEM values and we are done!
pts.utm$elevation <-

 # Perfect. Now get ready a data.frame for exporting
 export <- data.frame(pts.utm)
# And save it
#setwd("../")
write.csv(export, file = "Moss_gis_covars.csv")
```