# Introduction

In this lab we will be exploring some of the ways you can use R for data visualization. Specifically, we will be exploring problems associated with visualizing multi-dimensional data. After completion of the lab you will:

* Be able to plot geospatial data in R
* Be able to explain the challenges of choosing color scales for data visualizations

The data for this exercise come from the Shale Hills Critical Zone Observatory (SHCZO) in State College Pennsylvania. The SHCZO was established to understand the interactions between hydrology, ecology, geography, and climate in the north central Application mountains. For this exercise, we will be exploring the relationships between soil water content, topography, and time. In your follow-up homework assignment, you will be exploring how precipitation also interacts with these factors.

# Data sets:

* Shale\_Hills\_DEM.tiff: Digital elevation model showing the elevation (in m) of the Shale Hills watershed
* wshd\_nad83\_edit.shp: Shapefile outlining the boundary of the shale hills watershed
* SM\_10cm\_2010.csv: Text file containing soil moisture measured at 10cm depth using TDR probes, column headers indicate data of observation (YYYYMMDD)

# Getting Started

For this lab you will need to install and load several R packages. These packages are: sp, rgdal, raster, gstat, animation, RColorBrewer.

Let’s start by opening Rstudio and starting a new session.

To install a package use the command install.packages. For the sp package this would look like this:

install.packages("sp")

To load an installed package use the command library():

library(sp)

**Using install.packages() and library(), install all the required packages.**

# Loading the geospatial data

Loading geospatial data is a little different that loading text files. To load the DEM data use the command raster():

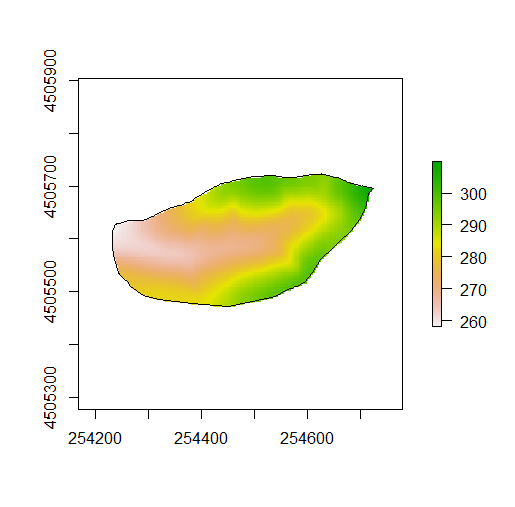
DEM <- raster('Shale\_Hills\_DEM.tif')

To load the shapefile for the watershed boundary, use the command readOGR():

ws <- readOGR(dsn = paste0(getwd(), '/Watershed\_Boundary/Watershed\_Boundary/wshd\_nad83\_edit.shp'))

Now you have the watershed boundary and DEM loaded you can plot them

Use the plot() command to plot the DEM then plot the watershed boundary on top of it (hint, to add a plot to an existing plot, use the option “add = T” in plot(). Your resulting plot should look like this:



What’s missing from this figure? (hint: the options “xlab”, “ylab” and “main” in the plot() function would be would be useful)

# Loading soil water data

The soil water data are sorted as a text file and can be loaded like any other text file using read.csv():

sm <- read.csv("SM\_10cm\_2010.csv")

Let’s take a look at the data using the head() function:

head(sm)

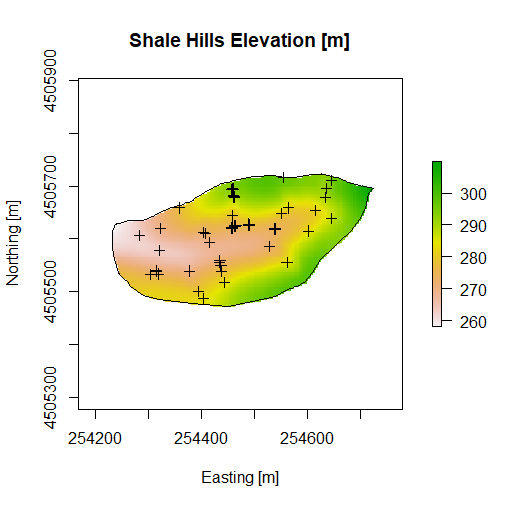
The data are arranged so that each row represents a unique location in the watershed. The first column is the station ID, the second and third column are the coordinates of the station, and the remaining columns are the soil water contents for each measurement data (denoted by the header)

We need to convert the object “sm” to a spatial object. To do this we are going to 1) remove the station ID’s from the object, and 2) convert the object to a spatial points dataframe:

sm <- data.frame(sm[, -1])

coordinates(sm) <- ~ X+Y

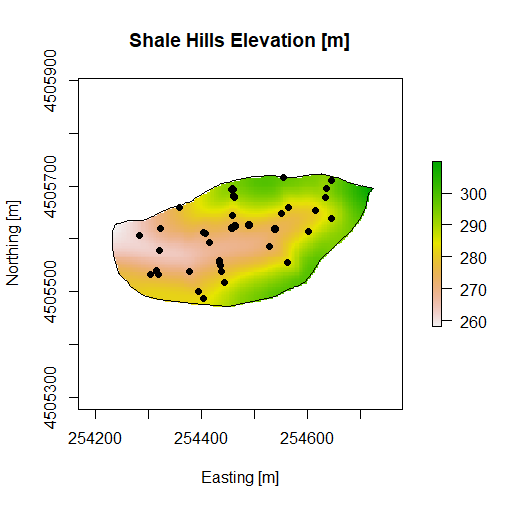
Now, plot the soil moisture stations on out watershed map using the plot command and the option “add=T”. Your plot should now look like this:



I’m not a fan of the “+” symbols so lets change them to a circle using the option “pch = 16” in the plot() function:

plot(sm, add = T, pch = 16)

How does that plot look? How would you fix it? (Hint, think of option “add”). Figure out how to make your plot look like this:



Next, let’s add a legend to denote what the circles represent. To add a legend, use the command legend():

legend('topright', legend = 'Soil Moisture Station', pch = 16)

# Visualizing soil moisture across the watershed

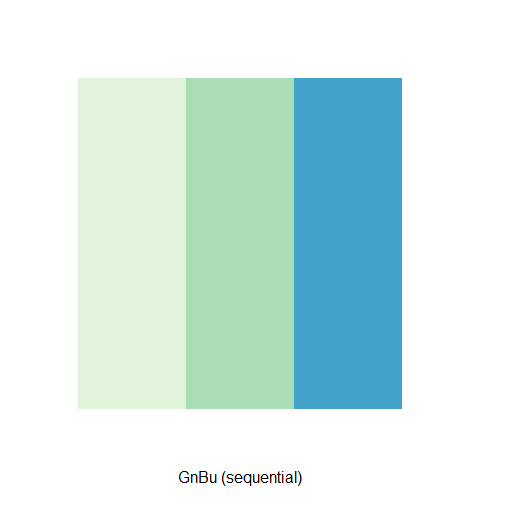
To symbolize the soil moisture across the watershed, we will be changing the color of each soil moisture points in out plot. To do this we need two things 1) a color scale and 2) a way of mapping the soil moisture values to the color scale.

For the color scale we will be using the package RColorBrewer. Color brewer has a lot of great tools for building usable color ramps. The best way to interact with the colorbrewer software is at <https://colorbrewer2.org/>.

For our purposes we want a sequential pallet the goes from light (low soil moisture) to dark (high soil moisture) to do this we will use the command display.brewer.pal():

display.brewer.pal(3, 'GnBu')

col.pal <- brewer.pal(3, 'GnBu')



In the command, the number dictates the number of classes and the string denotes the color palette. The second command, brewer.pal(), saves the pallet for use in an R plot*. Three values are probable not enough so let’s select a pallet with 7 values and the palette “YlGnBu”.*

Now we need a way to map the soil moisture values to the color pallet we will do this by assigning each range of soil moisture values a color. There are many ways to do this we will be exploring two. In the first, we will break the range of soil moistures into equal intervals from the minimum to maximum value. To do this we will use the seq() and cut() commands:

intervals <- seq(min(sm@data[,1]), max(sm@data[,1]),length=8)

intervals

idx.col <- cut(sm@data[,1], intervals, include.lowest = T)

idx.col

*Based on the outputs, what does the seq() and cut() command do?*

For now, we are only interested in the first date, so we are only accessing data for the first date stored in the first column of the soil moisture data (i.e. sm@data[, 1]).

To determine which color corresponds to each soil moisture point we can select the colors from the color pallet using the object idx.col:

col.pal[idx.col]

This will print the Hex color code for each color in the pallet that corresponds to each soil moisture range specified in “intervals”. We can then feed this data into the col() option in plot to colorize the soil moisture points:

plot(DEM, col = brewer.pal(9, 'Greys'),

xlab = 'Easting [m]', ylab = 'Northing [m]', main = 'Shale Hills Elevation [m]')

plot(sm, pch = 16, col=col.pal[idx.col], add = T)

plot(ws, add=T)

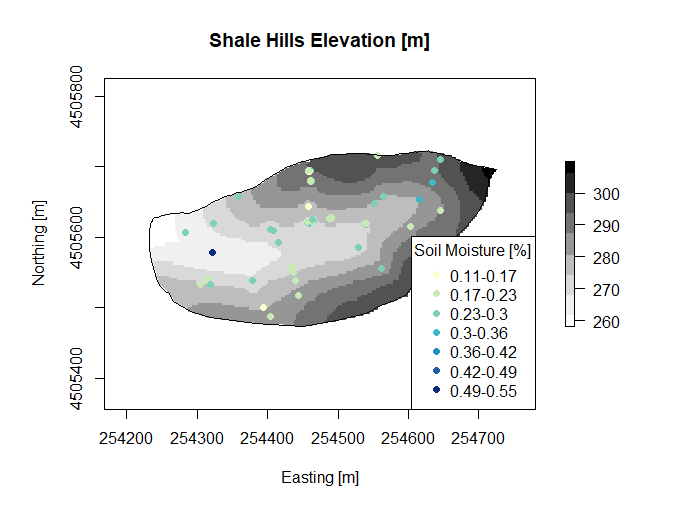
We can also add a legend:

leg <- paste0(round(intervals[-8], digits = 2), '-',

round(intervals[-1], digits = 2))

legend('bottomright', legend = leg, pch = 16, col = col.pal , title = 'Soil Moisture [%]')

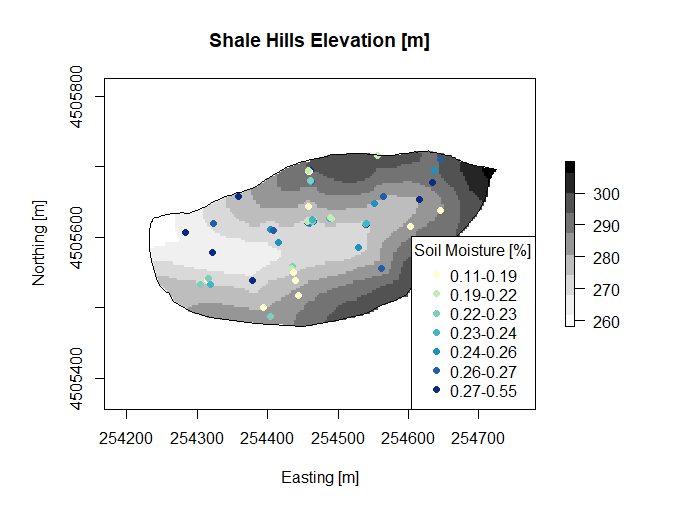
Your plot should look like this:



*What do you notice about the colors for soil moisture data? Are you using all the colors in the color ramp?* We can correct this by switching from a linear interval to a quantile interval using the quantile() command:

intervals <- quantile(sm@data[,1], seq(0,1,length=8), na.rm = T)

Your resulting graph should look like this:



*Why do these graphs look different? Which do you prefer and why?*

# Animating time series data

Ultimately, we are interested in what happens to soil moisture data over time to do this we can plot the same graph we just made but with soil moisture data from a different date. To do this we only need to change which data we are accessing from the object “sm”. The easiest way to do this is by altering the column index from sm@data[, 1] to sm@data[, 2]. The resulting code would then be (notice that the only changes are to the first three lines):

intervals <- quantile(sm@data[,2], seq(0,1,length=8), na.rm = T)

intervals

idx.col <- cut(sm@data[,2], intervals, include.lowest = T)

idx.col

col.pal[idx.col]

plot(DEM, col = brewer.pal(9, 'Greys'),

xlab = 'Easting [m]', ylab = 'Northing [m]', main = 'Shale Hills Elevation [m]')

plot(sm, pch = 16, col=col.pal[idx.col], add = T)

plot(ws, add=T)

leg <- paste0(round(intervals[-8], digits = 2), '-',

round(intervals[-1], digits = 2))

legend('bottomright', legend = leg, pch = 16, col = col.pal, title = 'Soil Moisture [%]')

To animate this plot, we need a simple way to alter the column index automatically. To do this we will use a “for loop”. In R for loops repeat the same code for prespecified amount of time for example a basic loop may look like this:

x <- 1

for(i in 1:10){

x <- x+1

}

X

In each iteration of the loop R will ad one to the value of x. After the loop, we have added 1 to x ten times resulting in a final value of x of 11.

We will use for loops to animate our data:

ani.options(interval = .3)

for(i in 1:ncol(sm@data)){

intervals <- quantile(sm@data[,i], seq(0,1,length=8), na.rm = T)

intervals

idx.col <- cut(sm@data[,i], intervals, include.lowest = T)

idx.col

col.pal[idx.col]

plot(DEM, col = brewer.pal(9, 'Greys'),

xlab = 'Easting [m]', ylab = 'Northing [m]', main = 'Shale Hills Elevation [m]')

plot(sm, pch = 16, col=col.pal[idx.col], add = T)

plot(ws, add=T)

leg <- paste0(round(intervals[-8], digits = 2), '-',

round(intervals[-1], digits = 2))

legend('bottomright', legend = leg, pch = 16, col = col.pal, title = 'Soil Moisture [%]')

ani.pause()

}

The functions ani.options() and ani.pause() control the animation. *What happens if you change the option “interval” in ani.options? Try some other values to see.*

*What does the animation show you about soil moisture changes over time?*

*Look at the color scale during the animation, what do you notice? How could you fix this problem? Does “fixing” the color scale change what you see in the animation?*