Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

---

USDA Data Carpentry - Geospatial

Instructors and helpers:

- Jon Wheeler (UNM)
- Kelly Meehan (USFS)
- Heather Savoy (USDA-ARS-SCINet) - Helper
- Amisha Poret-Peterson (USDA-ARS Crops Pathology and Genetics Research Unit) - Helper
- Ryan Lucas (USDA-ARS-SCINet) - Helper
- Kelly Thorp (USDA-ARS, Maricopa AZ) - Helper

Please add your name below:

- Pat Clark (USDA-ARS, Northwest Watershed Research Center, Boise)
- Mahesh Lal Maskey (USDA-ARS, Sustainable Water Management Research Unit) ORISE Postdoctoral Hydrological Modeler Fellow at SEA, Stoneville, MS
- Peter Olsoy (Boise State University / ARS Boise)
- Kate White (USDA-ARS Adaptive Cropping Systems Lab)
- Ann Bybee-Finley (USDA-ARS Sustainable Agricultural Systems Lab)
- Maja Subelj, UA, UL

Useful links:
Workshop website: https://eniac-6.github.io/2022-11-29-usda-online/
Data Carpentry Geospatial lesson: https://datacarpentry.org/geospatial-workshop/
Data download link: https://ndownloader.figshare.com/articles/2009586/versions/10
(could be slow)

Project management with RStudio

https://datacarpentry.org/r-intro-geospatial/02-project-intro/index.html

Intro to Raster Data

https://datacarpentry.org/r-raster-vector-geospatial/01-raster-structure/index.html

```
# If you need to install packages:
install.packages('raster')
install.packages('rgdal')
install.packages('ggplot2')
install.packages('dplyr')

Those 4 packages will be the main ones we use today.
#Note: package 'renv' for keeping track of package versions to not break code later
install.packages('renv')

# Load packages for the lesson
library(raster)
library(rgdal)
library(ggplot2)
library(dplyr)

# read a DSM (digital surface model) raster
# use GDALinfo() function to view info about a raster
GDALinfo("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")

# Read raster data into memory
DSM_HARV <- raster(
"data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif"
)

# View summary statistics of the da
summary(DSM_HARV)

# In order to plot raster data, we need to convert to a data frame
DSM_HARV_df <- as.data.frame(DSM_HARV, xy = TRUE)

# Plot the data
ggplot() +
   geom_raster(data = DSM_HARV_df , aes(x = x, y = y, fill = HARV_dsmCrop)) +
   scale_fill_viridis_c() +
   coord_quickmap()

# View raster metadata and attributes
# view the CRS info
crs(DSM_HARV)

# get the minimum value
minValue(DSM_HARV)
```

```
# get the number of layers
nlayers(DSM_HARV)

# Explore the data
# make a histogram of the raster data values
ggplot() +
  geom_histogram(data = DSM_HARV_df, aes(HARV_dsmCrop))
```

Plot Raster Data

https://datacarpentry.org/r-raster-vector-geospatial/02-raster-plot/index.html

```
# Working with categorical data values
# create a new column to sort elevation values into 3 bins
DSM_HARV_df <- DSM_HARV_df %>%
  mutate(fct_elevation = cut(HARV_dsmCrop, breaks = 3))

# plot the distribution of the new column
ggplot() +
  geom_bar(data = DSM_HARV_df, aes(fct_elevation))

# view the levels for the fct_elevation factor
unique(DSM_HARV_df$fct_elevation)

# plot the categorical elevation values
ggplot() +
  geom_raster(data = DSM_HARV_df,
                      aes(x = x, y = y, fill = fct_elevation)) +
   coord_quickmap()                    # can use this method as an alternative to coord_sf()

# customize the color palette
# view 3 hex colors from a "terrain colors" palette
terrain.colors(3)

# use those colors in a plot
ggplot() +
  geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation)) +
  scale_fill_manual(values = terrain.colors(3)) +
  coord_quickmap()

# change legend name
ggplot() +
  geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation)) +
  scale_fill_manual(values = terrain.colors(3), name = "Elevation") +
  coord_quickmap()

# Layer rasters in a single plot
# Read in a new raster
```

```
DSM_hill_HARV <- raster(
"data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_DSMhill.tif"
)

# convert to a data frame
DSM_hill_HARV_df <- as.data.frame(DSM_hill_HARV, xy = TRUE)

# plot the hillshade raster by itself
ggplot() +
  geom_raster(data = DSM_hill_HARV_df, aes(x = x, y = y, alpha =HARV_dsmHill )) +
  scale_alpha(range = c(0.15, 0.65), guide = "none") +
  coord_quickmap()

# Create a plot with elevation and hillshade rasters
ggplot() +
 geom_raster(data = DSM_HARV_df ,
           aes(x = x, y = y,
               fill = HARV_dsmCrop)) +
 geom_raster(data = DSM_hill_HARV_df,
           aes(x = x, y = y,
               alpha = HARV_DSMhill)) +
 scale_fill_viridis_c() +
 scale_alpha(range = c(0.15, 0.65), guide = "none") +
 ggtitle("Elevation with hillshade") +
 coord_quickmap()
```

Reproject raster data

https://datacarpentry.org/r-raster-vector-geospatial/03-raster-reproject-in-r/index.html

```
# read DTM (digital terrain model) raster
DTM_HARV <-  raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_dtmCrop.tif")

# read in hillshade raster
DTM_hill_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_DTMhill_WGS84.tif")

# convert both to data frames
DTM_HARV_df <- as.data.frame(DTM_HARV, xy = TRUE)
DTM_hill_HARV_df <- as.data.frame(DTM_hill_HARV, xy = TRUE)

# plot the rasters together
# note: this plot doesn't render correctly!
ggplot() +
  # Add elevation layer
  geom_raster(data = DTM_HARV_df, aes(x = x, y = y, fill = HARV_dtmCrop)) +
  # Add hillshade layer
```

```
  geom_raster(data = DTM_hill_HARV_df, aes(x = x, y = y, alpha = HARV_DTMhill_WGS84)) +
  scale_fill_gradientn(name = "Elevation",  colors = terrain.colors(10)) +
  coord_quickmap()

# try plotting the layers separately to see what the error is
# elevation data first
ggplot() +
  geom_raster(data = DTM_HARV_df, aes(x = x, y = y, fill = HARV_dtmCrop)) +
  scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10)) +
  coord_quickmap()

# now render hillshade plot
ggplot() +
  geom_raster(data = DTM_hill_HARV_df, aes(x = x , y = y, alpha = HARV_DTMhill_WGS84)) +
  coord_quickmap()

# both render fine by themselves - check the crs for each raster
crs(DTM_HARV)

crs(DTM_hill_HARV)

# CRSs of the two rasters are different
# reproject the hillshade raster
# use projectRaster() function
DTM_hillUTMZ18N_HARV <- projectRaster(DTM_hill_HARV, crs = crs(DTM_HARV))

# Useful alternatives to the above:
# specify the raster package
DTM_hillUTMZ18N_HARV <- raster::projectRaster(DTM_hill_HARV, crs = crs(DTM_HARV))
# Harmonize rasters (all attributes)
DTM_hillUTMZ18N_HARV <- projectRaster(DTM_hill_HARV, to = DTM_HARV)

# check crs and extent of reprojected raster
crs(DTM_hillUTMZ18N_HARV)

extent(DTM_hillUTMZ18N_HARV)

# also check the resolution
res(DTM_hillUTMZ18N_HARV)

# in this case the resolution is off so we need to also reproject the resolution
DTM_hillUTMZ18N_HARV <- projectRaster(DTM_hill_HARV, crs = crs(DTM_HARV), res =
res(DTM_HARV))

# now we can plot those rasters together
DTM_hill_HARV_2_df <- as.data.frame(DTM_hillUTMZ18N_HARV, xy = TRUE)

ggplot() +
    geom_raster(data = DTM_HARV_df ,
```

```
            aes(x = x, y = y,
             fill = HARV_dtmCrop)) +
    geom_raster(data = DTM_hill_HARV_2_df,
            aes(x = x, y = y,
              alpha = HARV_DTMhill_WGS84)) +
    scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10)) +
    coord_quickmap()
```

Challenge: Reproject, then Plot a Digital Terrain Model

Create a map of the San Joaquin Experimental Range field site using the SJER_DSMhill_WGS84.tif and SJER_dsmCrop.tif files.

Reproject the data as necessary to make things line up!

Solutions:

Raster Calculations

https://datacarpentry.org/r-raster-vector-geospatial/04-raster-calculations-in-r/index.html

```
# We can do math against raster values
# In this case, Canopy Height Model (CHM) is the difference between the pixel values in the
# digital surface model (DSM) - digital terrain model (DTM)

# First plot the rasters
# DTM
ggplot() +
     geom_raster(data = DTM_HARV_df ,
            aes(x = x, y = y, fill = HARV_dtmCrop)) +
    scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10)) +
    coord_quickmap()

# DSM
ggplot() +
     geom_raster(data = DSM_HARV_df ,
            aes(x = x, y = y, fill = HARV_dsmCrop)) +
    scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10)) +
    coord_quickmap()

# One way to do raster math
# Canopy height model (CHM)
CHM_HARV <- DSM_HARV - DTM_HARV

# convert to a data frame
CHM_HARV_df <- as.data.frame(CHM_HARV, xy = TRUE)

# plot the distribution of values in the CHM
```

```
ggplot(CHM_HARV_df) +
   geom_histogram(aes(layer))
```

Anonymous feedback: https://forms.gle/knQkAxUZ4vKM5TKh7

Raster calculations continued

```
# There is a second way to do raster calculations
# using the overlay() function
# in this case we are defining the funtion within the "fun" argument to the overlay() function
CHM_ov_HARV <- overlay(DSM_HARV, DTM_HARV, fun = function(r1, r2) { return(r1 - r2)})

# alternatively, we can define the function separately
calculate_CHM <- function(r1, r2) {
  return(r1 - r2)
}

# and then call the function by itself
CHM_fun <- calculate_CHM(r1 = DSM_HARV, r2 = DTM_HARV)

# OR call it within the overlay() function
CHM_ov_HARV_2 <- overlay(DSM_HARV, DTM_HARV, fun = calculate_CHM)

# convert to dataframe
CHM_ov_HARV_df <- as.data.frame(CHM_ov_HARV, xy = TRUE)

# quick plot to view
plot(CHM_ov_HARV_2)

# plot the CHM
ggplot() +
  geom_raster(data = CHM_ov_HARV_df, aes(x = x, y = y, fill = layer)) +
  scale_fill_gradientn(name = "Canopy Height", colors = terrain.colors(10)) +
  coord_quickmap()

# Since we created a new raster (CHM), we can save it to a tif
writeRaster(x = CHM_ov_HARV,
            filename = "CHM_HARV.tiff",
            format = "GTiff",
            overwrite = TRUE,
            NAflag = -9999)
```

Challenge: Explore the NEON San Joaquin Experimental Range Field Site

Data are often more interesting and powerful when we compare them across various locations. Let’s compare some data collected over Harvard Forest to data collected in Southern California. The NEON San Joaquin Experimental Range (SJER) field site located in Southern California has a very different ecosystem and climate than the NEON Harvard Forest Field Site in Massachusetts.

Import the SJER DSM and DTM raster files and create a Canopy Height Model. Then compare the two sites. Be sure to name your R objects and outputs carefully, as follows: objectType_SJER (e.g. DSM_SJER). This will help you keep track of data from different sites!

1. You should have the DSM and DTM data for the SJER site already loaded from the Plot Raster Data in R episode.) Don't forget to check the CRSs and units of the data.
2. Create a CHM from the two raster layers and check to make sure the data are what you expect.
3. Plot the CHM from SJER.
4. Export the SJER CHM as a GeoTIFF.
5. Compare the vegetation structure of the Harvard Forest and San Joaquin Experimental Range.

Solutions:
read SJER rasters:

```
DSM_SJER <- raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DSM/SJER_dsmCrop.tif")
DSM_SJER_df <- as.data.frame(DSM_SJER, xy = TRUE)
DTM_SJER<-raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DTM/SJER_dtmCrop.tif")
DTM_SJER_df<-as.data.frame(DTM_SJER, xy=TRUE)
CHM_ov_SJER <- overlay(DSM_SJER, DTM_SJER, fun = function(r1, r2){return(r1-r2)})
plot(CHM_ov_SJER)
```

Work with multi-band rasters

https://datacarpentry.org/r-raster-vector-geospatial/05-raster-multi-band-in-r/index.html

```
# read a layer from an RGB raster
RGB_band1_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")

# by default raster() will import the first band of a multiband raster
# convert to df
RGB_band1_HARV_df <- as.data.frame(RGB_band1_HARV, xy = TRUE)

# plot the raster
ggplot() +
  geom_raster(data = RGB_band1_HARV_df, aes(x = x, y = y, alpha = layer)) +
  coord_quickmap()

# view the CRS of the raster
crs(RGB_band1_HARV)

# specify which band to read when reading a multi band raster
RGB_band2_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif", band
```

```
= 2)

# convert to dataframe
RGB_band2_HARV_df <- as.data.frame(RGB_band2_HARV, xy = TRUE)

# plot and compare with band1
ggplot() +
  geom_raster(data = RGB_band2_HARV_df, aes(x = x, y = y, alpha = layer)) +
  coord_quickmap()

# We can also read in all bands/layers at once using the stack() method
RGB_stack_HARV <-
stack("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")

# Inspect the attributes of the bands
# All info at once
RGB_stack_HARV

# info about attributes for each layer
RGB_stack_HARV@layers

# info about a specific band - in this case the second band; note 2 square brackets for indexing
RGB_stack_HARV[[2]]

# convert entire stack to a dataframe
RGB_stack_HARV_df <- as.data.frame(RGB_stack_HARV, xy = TRUE)

# inspect the df
# note in the output that the bands have been converted to columns - 1 column for each band
str(RGB_stack_HARV_df)

# plot the dataframe
# histogram of the first band
ggplot() +
  geom_histogram(data = RGB_stack_HARV_df, aes(HARV_RGB_Ortho_1))

# plot of the second band
ggplot() +
  geom_raster(data = RGB_stack_HARV, aes(x = x, y = y, alpha = HARV_RGB_Ortho_2)) +
  coord_quickmap()

# we can access any band in a stack using the syntax above
# we can also plot all of the bands together, in this case using an RGB color map
# note the first argument is the raster, not the dataframe
plotRGB(RGB_stack_HARV, r = 1, g = 2, b = 3)

# In some cases we may want to stretch a plot to improve contrast or clarity by using the full range
# of values (0-255 in the case of RGB)
```

```
# apply a linear stretch
plotRGB(RGB_stack_HARV, r =1, g = 2, b = 3, scale = 800, stretch = "lin")

# apply a histogram stretch
plotRGB(RGB_stack_HARV, r =1, g = 2, b = 3, scale = 800, stretch = "hist")

# raster brick vs raster stack
# depending on use case, bricks can be faster
RGB_brick_HARV <- brick(RGB_stack_HARV)

# check the size of the brick
object.size(RGB_brick_HARV)
```

Open and plot shapefiles

https://datacarpentry.org/r-raster-vector-geospatial/06-vector-open-shapefile-in-r/index.html

```
# Shapefiles contain vector data, not raster data
# We are using a new package - sf
library(sf)

# create a vector - "area of interest" or AOI
aoi_boundary_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HarClip_UTMZ18.shp")

# since shapefile data is already in a vector, shapefile objects are similar to dataframes
# as we can note from the class() output
class(aoi_boundary_HARV)

# view attributes
aoi_boundary_HARV

# use a function to get the geometry
st_geometry_type(aoi_boundary_HARV)

# Or get the CRS
st_crs(aoi_boundary_HARV)

# get the bounding box
st_bbox(aoi_boundary_HARV)

# Plot the shapefile - no need to convert to df
# NOTE: depending on the version of sf you're using, try "linewidth" in place of the "size" argument
ggplot() +
  geom_sf(data = aoi_boundary_HARV, size = 3, color = "black", fill = "cyan1") +
  ggtitle("AOI Boundary Plot")
```

Challenge: Import Line and Point Shapefiles

Using the steps above, import the HARV_roads and HARVtower_UTM18N layers into R. Call the HARV_roads object lines_HARV and the HARVtower_UTM18N point_HARV.

Answer the following questions:

1.
2. What type of R spatial object is created when you import each layer?
3.
4. What is the CRS and extent for each object?
5.
6. Do the files contain points, lines, or polygons?
7.
8. How many spatial objects are in each file?

Solutions

```
# roads as lines (lines)
lines_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")

# flux tower (point)
point_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")

# get info
# class (data frame)
class(lines_HARV)
class(point_HARV)

# bounding box
st_bbox(lines_HARV)
st_bbox(point_HARV)
```

Explore and plot by vector layer attributes

https://datacarpentry.org/r-raster-vector-geospatial/07-vector-shapefile-attributes-in-r/index.html

```
# Using shaepfile objects created above
# number of columns
ncol(lines_HARV)

# first six rows (spatial features in this case)
head(lines_HARV)

# values from a specific column
lines_HARV$TYPE

# create a subset of just footpaths
footpath_HARV <- lines_HARV %>%
  filter(TYPE == "footpath")
```

```
# how many obervations match query/filter
nrow(footpath_HARV)

# Plot the footpaths
ggplot() +
   geom_sf(data = footpath_HARV) +
   ggtitle("NEON Harvard Forest Field Site", subtitle = "Footpaths")

# Plot looks like one footpath, we know we have two
# we can use aesthetics to show the different paths
# NOTE: use "size" or "linewidth" depending on your version of sf
ggplot() +
   geom_sf(data = footpath_HARV, aes(color = factor(OBJECTID)), size = 1.5) +
   labs(color = "Footpath ID") +
   ggtitle("NEON Harvard Forest Field Site", subtitle = "Footpaths")
```

Challenge: Subset Spatial Line Objects Part 1

Subset out all boardwalk from the lines layer and plot it.

Solutions:

```
boardwalk_HARV<-lines_HARV%>%filter(TYPE=="boardwalk")

ggplot()+
  geom_sf(data = boardwalk_HARV,
          aes(color = factor(OBJECTID)), size = 1.5)+
  labs(color = "boardwalk ID")+
  ggtitle("NEON Harvard Forest Field Site",
          subtitle = "boardwalk")
```

Challenge: Subset Spatial Line Objects Part 2

Subset out all stone wall features from the lines layer and plot it. For each plot, color each feature using a unique color.

Solutions:

Customize Plots

```
# Create custom color palette
road_colors <- c('blue', 'green', 'navy', 'purple')

# Plot all roads and specify color for each TYPE
```

```
ggplot() +
  geom_sf(data = lines_HARV, aes(color = TYPE)) +
  scale_color_manual(values = road_colors) +
  labs(color = 'Road Type') +
  ggtitle('NEON Harvard Forest Field Site', subtitle = 'Roads & Trails') +

# Plot all roads and also specify line thickness for each TYPE
ggplot() +
  geom_sf(data = lines_HARV, aes(color = TYPE, linewidth = TYPE)) +
  scale_color_manual(values = road_colors) +
  labs(color = 'Road Type') +
  ggtitle('NEON Harvard Forest Field Site', subtitle = 'Roads & Trails - Line width varies')
```

Do challenge
Save challenge +1+1+1

Anonymous feedback: https://forms.gle/xowYVXfyMpjsKBhc6

Day 2

```
# We may need to reload packages and re-create objects.
# The below code can be pasted into an R script and run to accomplish this

# load_proj_env.R
# Load packages
library(sf)
library(ggplot2)
library(raster)
library(dplyr)
library(rgdal)

## Episode 1 objects
# Digital Surface Model, Harvard
DSM_HARV <-
  raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")
DSM_HARV_df <- as.data.frame(DSM_HARV, xy = TRUE)

## Episode 2
# mutate DSM_HARV
DSM_HARV_df <- DSM_HARV_df %>%
  mutate(fct_elevation = cut(HARV_dsmCrop, breaks = 3))

custom_bins <- c(300, 350, 400, 450)
DSM_HARV_df <- DSM_HARV_df %>%
  mutate(fct_elevation_2 = cut(HARV_dsmCrop, breaks = custom_bins))

# DSM hillshade
```

```
DSM_hill_HARV <-
  raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_DSMhill.tif")
DSM_hill_HARV_df <- as.data.frame(DSM_hill_HARV, xy = TRUE)

## Episode 3
# Digital Terrain Model, Harvard
DTM_HARV <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_dtmCrop.tif")
DTM_HARV_df <- as.data.frame(DTM_HARV, xy = TRUE)

# DTM hillshade
DTM_hill_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_DTMhill_WGS84.tif")
DTM_hill_HARV_df <- as.data.frame(DTM_hill_HARV, xy = TRUE)

# Reproject hillshade
DTM_hill_UTMZ18N_HARV <- projectRaster(DTM_hill_HARV,
                                       crs = crs(DTM_HARV),
                                       res = res(DTM_HARV))
DTM_hill_HARV_2_df <- as.data.frame(DTM_hill_UTMZ18N_HARV, xy = TRUE)


## Episode 4
# Canopy Height Model, Harvard
CHM_HARV <- DSM_HARV - DTM_HARV
CHM_HARV_df <- as.data.frame(CHM_HARV, xy = TRUE)


## Episode 5
# RGB band 1
RGB_band1_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")
RGB_band1_HARV_df  <- as.data.frame(RGB_band1_HARV, xy = TRUE)

# RGB band 2
RGB_band2_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif", band
= 2)
RGB_band2_HARV_df <- as.data.frame(RGB_band2_HARV, xy = TRUE)

# Raster stack
RGB_stack_HARV <-
stack("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")
RGB_stack_HARV_df  <- as.data.frame(RGB_stack_HARV, xy = TRUE)


## Episode 6
# AOI (area of interest) boundary HARV (polygon)
aoi_boundary_HARV <- st_read(
  "data/NEON-DS-Site-Layout-Files/HARV/HarClip_UTMZ18.shp")
```

```
# roads as lines (lines)
lines_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")

# flux tower (point)
point_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")
```

Challenge: Plot Line Width by Attribute

In the example above, we set the line widths to be 1, 2, 3, and 4. Because R orders factor levels alphabetically by default, this gave us a plot where woods roads (the last factor level) were the thickest and boardwalks were the thinnest.

Let's create another plot where we show the different line types with the following thicknesses:

1. woods road size = 6
2. boardwalks size = 1
3. footpath size = 3
4. stone wall size = 2

Hint:

```
line_widths <- c(1, 2, 3, 4)
ggplot() +
  geom_sf(data = lines_HARV, aes(color = TYPE, size = TYPE)) +
  scale_color_manual(values = road_colors) +
  labs(color = 'Road Type') +
  scale_size_manual(values = line_widths) +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Roads & Trails - Line width
varies") +
  coord_sf()
```

Solutions

```
line_widths <- c(1, 3, 2, 6)

# Plot
ggplot() +
  geom_sf(data = lines_HARV, aes(size = TYPE)) +
  scale_size_manual(values = line_width) +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Roads & Trails - Line width
varies") +
  coord_sf()
```

Breakout rooms

On our own+1+1+1 +1+1+1

Challenge: Plot Polygon by Attribute

1. Create a map of the state boundaries in the United States using the data located in your downloaded data folder: NEON-DS-Site-Layout-Files/US-Boundary-Layers\US-State-Boundaries-Census-2014.
2. Apply a line color to each state using its region value. Add a legend.

Solutions
st
str(states)

```
ggplot() +
  geom_sf(data=states, aes(color=region)) +
  ggtitle("United States", subtitle="Colored by Region") +
  theme(legend.text = )
  coord_sf()
```

```
# Map of state boundaries with regions reordered
state_boundary_US <- sf::st_read('data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-State-Boundaries-Census-2014.shp', type = 7)

levels(factor(state_boundary_US$region))

# Re-order levels
state_boundary_US$region <- factor(state_boundary_US$region, levels = c('West', 'Southwest', 'Midwest', 'Southeast', 'Northeast'))

# create color scheme
colors <- c('purple', 'springgreen', 'yellow', 'brown', 'navy')

# plot
ggplot() +
  geom_sf(data = state_boundary_US, aes(color = region), linewidth = 1) +
  scale_color_manual(values = colors) + ggtitle('Contiguous US State Boundaries') + coord_sf()
```

Challenge: Plot Polygon by Attribute

1. Using the NEON-DS-Site-Layout-Files/HARV/PlotLocations_HARV.shp shapefile, create a map of study plot locations, with each point colored by the soil type (soilTypeOr). How many different soil types are there at this particular field site? Overlay this layer on top of the lines_HARV layer (the roads). Create a custom legend that applies line symbols to lines and point symbols to the points.

Hint: Refer to help info on scale fill and scale color:
?scale_color_manual
?scale_fill_manual

```
# Example from earlier:
# plot all of our sf
ggplot() +
  # site boundary first
  geom_sf(data = aoi_boundary_HARV, fill = "grey", color = "grey") +
  # add lines sf
  geom_sf(data = lines_HARV,
          aes(color = TYPE),
          show.legend = "line",
          linewidth = 1) +
  # add flux tower point
  geom_sf(data = point_HARV, aes(fill = Sub_Type),
          color = "black",
          shape = 15) +
  # use custom road colors
  scale_color_manual(values = road_colors, name = "Line Type") +
  # symbology for points
  scale_fill_manual(values = "black", name = "Tower Location") +
  ggtitle("NHFFS") +
  coord_sf()


# Solution to challenge
# read sf
plot_locations <- st_read(
  "data/NEON-DS-Site-Layout-Files/HARV/PlotLocations_HARV.shp",
  type = 7
)
View(plot_locations)
# create color scheme
blue_orange <- c("cornflowerblue", "darkorange")

# plot- use an guide_legend() to override defaults
ggplot() +
  geom_sf(data = lines_HARV, aes(color = TYPE), show.legend = "line") +
  geom_sf(data = plot_locations, aes(fill = soilTypeOr),
          shape = 21, show.legend = 'point') +
  scale_color_manual(name = "Line Type", values = road_colors,
                     guide = guide_legend(override.aes = list(linetype = "solid",
                                                              shape = NA))) +
  scale_fill_manual(name = "Soil Type", values = blue_orange,
                    guide = guide_legend(override.aes = list(linetype = "blank",
                                                             shape = 21,
                                                             color = NA))) +
  ggtitle("NHFFS") +
  coord_sf()
```

Challenge: Plot Raster & Vector Data Together

You can plot vector data layered on top of raster data using the + to add a layer in ggplot. Create a plot that uses the NEON AOI Canopy Height Model data/NEON-DS-Airborne-Remote-Sensing/HARV/CHM/HARV_chmCrop.tif as a base layer. On top of the CHM, please add:

- The study site AOI.
- Roads.
- The tower location.

Be sure to give your plot a meaningful title.

Hint: make sure you have the canopy height model data frame:
str(CHM_HARV_df)

Solutions

```
ggplot()+
  geom_raster(data = CHM_HARV_df,
              aes(x=x,
                  y=y,
                  alpha=layer), #legend could be improved--make into factor....
              show.legend = "point")+
  geom_sf(data = aoi_boundary_HARV,
          fill = "blue",
          color = "blue",
          alpha = 0.5) +
  geom_sf(data = lines_HARV, aes(color = TYPE), show.legend = "line")+
  geom_sf(data = plot_locations, aes(fill = soilTypeOr),
          shape = 21, show.legend = "point")+
  scale_color_manual(name = "Line Type", values = road_colors,
                     guide = guide_legend(override.aes = list(linetype = "solid",
                                                              shape = NA))) +
  scale_fill_manual(name = "Soil Type", values = blue_orange,
                    guide = guide_legend(override.aes = list(linetype = "blank",
                                                             shape = 21,
                                                             color = NA))) +
  ggtitle("NHFFS") +
  coord_sf()

# Solution - requires CHM_HARV_df!
ggplot() +
  geom_raster(data = CHM_HARV_df, aes(x = x, y = y, fill = HARV_chmCrop)) +
  geom_sf(data = lines_HARV, color = "black") +
  geom_sf(data = aoi_boundary_HARV, color = "grey20", size = 1) +
  geom_sf(data = point_HARV, pch = 8) +
  ggtitle("NEON Harvard Forest Field Site w/ Canopy Height Model") +
  coord_sf()
```

Lesson 8: Handling Spatial Projection & CRS

```
# Spatial projection and CRS in shapefiles
state_boundary_US <- st_read('data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-State-Boundaries-Census-2014.shp', type = 7)

# plot
ggplot() +
  geom_sf(data = country_boundary_US, color = 'gray 40', linewidth = 2) +
  geom_sf(data = state_boundary_US, color = 'gray18') +
  ggtitle('US States')

# Us Boundary Layer

country_boundary_US <- st_read('data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-Boundary-Dissolved-States.shp', type = 7)
```

Challenge - Import & Plot Additional Points

We want to add two phenology plots to our existing map of vegetation plot locations.

Import the .csv: HARV/HARV_2NewPhenPlots.csv into R and do the following:

1. Find the X and Y coordinate locations. Which value is X and which value is Y?
2. These data were collected in a geographic coordinate system (WGS84). Convert the dataframe into an sf object.
3. Plot the new points with the plot location points from above. Be sure to add a legend. Use a different symbol for the 2 new points!

If you have extra time, feel free to add roads and other layers to your map!

```
# Plot starter code
# plot the data
ggplot() +
  geom_sf(data = aoi_boundary_HARV) +
  geom_sf(data = plot_locations_sp_HARV) +
  ggtitle("AOI Boundary Plot") +
  coord_sf()
```

Troubleshooting (paste error messages here)

Solutions:

```
newphenplots <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_2NewPhenPlots.csv")
str(newphenplots)

crs_wgs84 <- crs(country_boundary_US)

newphenplots_sf <- st_as_sf(newphenplots,
```

```
                coords=c("decimalLon", "decimalLat"), #whoops, check the right order
                crs=crs_wgs84)

ggplot() +
  geom_sf(data = aoi_boundary_HARV) +
  geom_sf(data = plot_locations_sp_HARV) +
  geom_sf(data = newphenplots_sf, color="red") +
  geom_sf(data=lines_HARV, aes(color=TYPE), show.legend="line") +
  scale_color_manual(name="Line Type", values=road_colors) +
  ggtitle("New Phenology Plots") +
  coord_sf()

# solution
pheno_locations_HARV <- read.csv(
  "data/NEON-DS-Site-Layout-Files/HARV/HARV_2NewPhenPlots.csv"
)
str(pheno_locations_HARV)

# set CRS to WGS84
st_crs(country_boundary_US)
geogCRS <- st_crs(country_boundary_US)
class(geogCRS)

# convert csv to sp
pheno_locations_sp_HARV <- st_as_sf(pheno_locations_HARV,
                            coords = c("decimalLon", "decimalLat"),
                            crs = geogCRS)

st_crs(pheno_locations_sp_HARV)
```

Extra challenge:
Save the phenology plot SF object to a shapefile.
Example:

```
# write to sf
st_write(plot_locations_sp_HARV,
        "data/PlotLocations_HARV.shp",
        driver = "ESRI Shapefile")

# Solution
# write phenology plot sf to file
st_write(pheno_locations_sp_HARV,
        "data/PhenologyPlots_HARV.shp",
        driver = "ESRI Shapefile")
```

Anonymous feedback: https://forms.gle/phGPRX2hoBiUvWyC9

Challenge: Crop to Vector Points Extent

1. Crop the Canopy Height Model to the extent of the study plot locations.
2. Plot the vegetation plot location points on top of the Canopy Height Model.

Solutions:

Challenge: Extract Raster Height Values For Plot Locations

1) Use the plot locations object (plot_locations_sp_HARV) to extract an average tree height for the area within 20m of each vegetation plot location in the study area. Because there are multiple plot locations, there will be multiple averages returned, so the df = TRUE argument should be used.

2) Create a plot showing the mean tree height of each area.

Troubleshooting (paste errors)

Solutions

```
mean_tree_height_plots <- extract(x = CHM_HARV,
                                  y = plot_locations_sp_HARV,
                                  buffer = 20,
                                  df = TRUE,
                                  fun = mean)
  mean_tree_height_plots

# Install a couple more packages
install.packages("reshape")
install.packages("scales")
```

Challenge: Raster Metadata

Investigate the metadata for our RasterStack and answer the following questions.

1. What are the x and y resolution of the data?
2. What units are the above resolution in?

Solutions:

```
extent(NDVI_HARV_stack)

harv_met_daily <- read.csv(
  "data/NEON-DS-Met-Time-Series/HARV/FisherTower-Met/hf001-06-daily-m.csv"
)
```

Examine RGB Raster Files

Plot the RGB images for the Julian days 277 and 293. Compare those with the RGB plots for Julian days 133 and 197 (shown above). Does the RGB imagery from these two days explain the low NDVI values observed on these days?

Do together+1+1+1+1+1
Do on our own

Post workshop survey https://carpentries.typeform.com/to/UgVdRQ?slug=2022-11-29-usda-online