
Working with Geospatial Data in R

10, 11, 15, & 16 November, 2022 - online workshop with Jornada LTER, Sevilleta LTER, and NM EPSCoR

Instructors: Todd Quinn, Alesia Hallmark, Darren James, Jon Wheeler, Greg Maurer
Helpers: John Ragosta & any instructor not currently instructing
****
Workshop website: https://jornada-im.github.io/2022-11-10-jornada-geo-online/
Code of Conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

All lesson content: https://datacarpentry.org/lessons/#geospatial-curriculum
Of potential interest, the NEON Tutorials (covering many of the same topics using NEON data): https://www.neonscience.org/resources/learning-hub/tutorials

Part 1: Introduction to Geospatial Concepts

Slides for this part: https://docs.google.com/presentation/d/1ft_Pau2HWs-_r4vGZsetbxzOjIjnBH4aGPu0u1oR25M/edit?usp=sharing

Introduction to raster data

https://datacarpentry.org/organization-geospatial/01-intro-raster-data/index.html

Exercise:

Advantages and Disadvantages

With your neighbor, brainstorm potential advantages and disadvantages of storing data in raster format. Add your ideas to the Etherpad. The Instructor will discuss and add any points that weren’t brought up in the small group discussions.

Share your thoughts here!

advantage: pixels can be as large or as small as the study needs
disadvantages: larger pixels can't be made smaller to improve presicion

Disadvantage: Low picture quality, permits of where pictures can be taken
Advantage: pictures of big landscapes/areas
  Tradeoffs high resolution: larger area cover with more detail, more data accrued
  low resolution: smaller files but not as much detail
  same extent
  polygons and points

Disadvantages: Low resolution, limited temporal coverage, accuracy, storage
Disadvantages: Alternatively, too high resolution may lead to constraints in data processing speeds or computational ability needed to process

Disadvantage: file size
Advantantage: Often easily accessible (open source) and lots of available data

Maybe pixels could be to broad? Difficult to appropriately classify types (disadvantage). Data is freely available for use from sattelite imagery (advantage)

Advantages: adjustable resolution to some extent, shareable across platforms, etc., open access software for manipulation
Disadvantages: computationally clunky
1. high res are more precise but computationally challenging/intensive
2. yes they do have the same extent

CONS: Different data types (int, float, mix precision, complex
 numbers); tech failure -- lost pixels in images; need analysis pipeines for image processing; categorical binning will reflect limitations / dated-ness in classification;

 PROS: Good for static images at a particular moment, possible to view many frames in a "stream"; can do array / matrix math on rasters

TRADEOFFS: hi res -- more data / larger data sets; low res -- less data;

1. higher resolution = bigger file size
2. no
poligons and points

raster data is wayyy prettier than a boring excel file of numbers!

lower resolution - faster processing times. Higher resolutions - higher level of detail in spatial information.
Yes, same extent.

Higher res = more detail = heftier file
same extent for all datasets
- Ach, trick question, Greg!

1) Higher resolution, more computational power needed. 2) Wait no! Point extent didn't reach as far north.

Lower resolution data = smaller file sizes, potential faster processing time with analysis
Yes all same extent
POINTS LINES POLYGONS - same extent?

Question - which of the 3 types are here

Polygon and points
points, polygons
Points and polygons
points and polygons
Polygon and points
^^

Question - What is the PROJ4 String?

Projection

projection - utm, zone - 18, datum - WGS84, ellipsoid - WGS84, units are in meters, eastern united states
projection = UTM, zone = 18, datum = WGS84, ellipsoid = WGS83, units = meter, eastern part of the US
Projection= UTM, zone = 18, datum = WGS84, ellipsoid = WGS84. Units = meters. Data is from eastern US
projection= utm, zone = 18, datum = WGS84; ellipsoid= WGS84; units = m;   between 78 and 72 degrees W (NE US ish)
projections=UTM, zone=18, datum=WGS84, ellipso=WGS84. Measured in meters. generally the eastern

part of north america

- pro - utm, zone 18, datum- wgs84, ellips - wgs 84, meters

Part 2: Introduction to R (a refresher)

Introduction to R

Challenge 1:

What will be the value of each variable after each statement in the following program?

mass <- 47.5
age <- 122
mass <- mass * 2.3
age <- age - 20

mass: 109.25
age: 102

mass= 109.25, age = 102
age = 102, mass 109.25
mass=109.25, age=102
mass = 109.25, age = 102

Challenge 2:

Run the code from the previous challenge, and write a command to compare mass to age. Is mass larger than age?

mass > age TRUE
mass >= age TRUE

mass > age TRUE

mass>= age true

> mass > age
[1] TRUE

Use the paste() function to concatenate strings or values of objects into a print output
result <- (1 + 100)
paste0("result 1: ", result) will output "result 1: 101"

Challenge 3

Which of the following are valid R variable names?

min_height - Valid
max.height - valid
_age - not valid
.mass -not valid
MaxLength valid
min-length not valid
2widths - not valid
celsius2kelvinvalid

#min_height yes
#max.height no
#_age yes ("_" connotes private in other languages)
#.mass no
#MaxLength yes
#min-length no
#2widths no
#celsius2kelvin yes

All valid except _age, min-length, 2widths

all valid except _age, min-length, 2widths,

Challenge 4

What code would we use to install the ggplot2 package?

```
install.packages("ggplot2")
```

install.packages("ggplot2")
install.packages("ggplot2")
^^

RStudio Cloud: https://rstudio.cloud/
The free tier is fine for the workshop!

Code to install necessary packages in an RStudio cloud environment
install.packages("dplyr")
install.packages("ggplot2")

Project management with RStudio

Challenge: Creating a self-contained project

Challenge 1: Download files

1. Download each of the data files listed below (Ctrl+S, right mouse click -> "Save as", or File -> "Save page as")

- nordic country data (https://raw.githubusercontent.com/datacarpentry/r-intro-geospatial/master/_episodes_rmd/data/nordic-data.csv)
- nordic country data (version 2) (https://raw.githubusercontent.com/datacarpentry/r-intro-geospatial/master/_episodes_rmd/data/nordic-data-2.csv)
- gapminder data (https://raw.githubusercontent.com/datacarpentry/r-intro-geospatial/master/_episodes_rmd/data/gapminder_data.csv)

2. Make sure the files have the following names:

- `nordic-data.csv`
- `nordic-data-2.csv`
- `gapminder_data.csv`

3. Save the files in the data/ folder within your project.

We will load and inspect these data later.

Challenge 2

We also want to move the data that we downloaded from the data page (http://datacarpentry.org/geospatial-workshop/data/) into a subdirectory inside r-geospatial.

If you haven’t already downloaded the data, you can do so by clicking this download link: https://ndownloader.figshare.com/articles/2009586/versions/10

1. Move the downloaded zip file to the data directory.
2. Once the data have been moved, unzip all files.

NOTE: If you are using the RStudio Cloud, you can use the following code to download and save the files to your data directory:

```
download.file(url = "https://raw.githubusercontent.com/datacarpentry/r-intro-geospatial/master/
_episodes_rmd/data/nordic-data.csv",
              destfile = "data/nordic-data.csv")


download.file(url = "https://raw.githubusercontent.com/datacarpentry/r-intro-geospatial/master/
_episodes_rmd/data/nordic-data-2.csv",
              destfile = "data/nordic-data-2.csv")
```

Manipulating data with dplyr

Executed code:

```
# We recommend creating a new R script to save your code (CONTROL+SHIFT+N in Windows)
(MAC...?)
# Read a file
gapminder <- read.csv("data/gapminder.csv")

# View the data as a table
View(gapminder)

# Get some statistics
mean(gapminder[gapminder$continent == "Africa", "gdpPercap"])
mean(gapminder[gapminder$continent == "Americas", "gdpPercap"])

# Using dplyr
library("dplyr")
year_country_gdp <- select(gapminder, year, country, gdpPercap)

# This uses the "pipe" operator (%>%), but same result as above
year_country_gdb <- gapminder %>%
  select(year, country, gdpPercap)
```

```
# Move forward with above using only countries on the European continent
year_country_gdp_euro <- gapminder %>%
  filter(continent == "Europe") %>%
  select(year, country, gdpPercap)
```

Write a single command (which can span multiple lines and includes pipes) that will produce a dataframe that has the African values for lifeExp, country and year, but not for other Continents.

```
year_country_lifeexp_Africa = gapminder %>%  # would be better with " <- " assignment arrow
  filter(continent == "Africa") %>%
  select (year, country, lifeExp)

year_country_life_africa <- gapminder %>% filter(continent == "Africa")  %>%
  select(year, country, lifeExp)

yr_ctry_lifeExp_Africa <- gapminder %>%
  filter(continent == "Africa") %>%
  select(lifeExp, country, year)

afr_life <- gapminder %>%
  filter(continent == "Africa") %>%
  select (year, country, lifeExp)


# Groupby and summarize

# Create a grouped dataframe - continent is the grouping variable
gapminder %>%
  group_by(continent) %>%
  str()

# Get the GDP per continent
gdp_bycontinents <- gapminder >%>
   group_by(continent) >%>
   summarize(mean_gdpPercap = mean(gdpPercap))

# View the aggregated data in the new dataframe, 'gdp_bycontinents'
gpd_bycontinents
```

Challenge 2

Calculate the average life expectancy per country. Which has the longest average life expectancy and which has the shortest average life expectancy?

Solutions:

```
avg_lifeExp <- gapminder %>%
  group_by(country) %>%
  summarise(mean_lifeExp = mean(lifeExp))

avg_lifeExp %>%
+   filter(mean_lifeExp == min(mean_lifeExp))
# A tibble: 1 × 2
  country      mean_lifeExp
  <chr>               <dbl>
1 Sierra Leone         36.8
>
> avg_lifeExp %>%
+   filter(mean_lifeExp == max(mean_lifeExp))
# A tibble: 1 × 2
  country mean_lifeExp
  <chr>          <dbl>
1 Iceland         76.5
>
```

```
lifeExp_percountry <- gapminder %>%
  group_by(country) %>%
  summarise(mean_lifeExp = mean(lifeExp))
#the shortest average life expectancy = Sierra Leone, 36.76917
#the longest = 76.51142
```

```
meanLifeExp <- gapminder %>%
  group_by(country) %>%
  summarise(mean_lifeExp = mean(lifeExp))
  sierra leone 36.8, iceland 76.6
  76.5
```

```
 lifeExp_by_Country <- gapminder %>%
  group_by(country) %>%
  summarize(mean_lifeExp_by_Country = mean(lifeExp))
lifeExp_by_Country<- data.frame(lifeExp_by_Country)
View(lifeExp_by_Country) #to open data file and manually click and sort on mean column to sort by smallest and largest
## shortest = Sierra Leone ~37 (36.76917)
## longest =  Iceland  ~77 (76.51142)
```

ANONYMOUS feedback for today:
https://forms.gle/u7RytzZoR7Tx6ydy9

Day 2

Introduction to Visualization (ggplot)

Challenge 1

Modify the example so that the figure shows the distribution of gdp per capita, rather than life expectancy:

```
ggplot(data = gapminder, aes(x = gdpPercap))+
  geom_histogram()
^^
ggplot(gapminder, aes(x = gdpPercap)) +
  geom_histogram()

ggplot(gapminder, aes(x = gdpPercap)) +
  geom_histogram()


ggplot(gapminder, aes(x = gdpPercap)) +
  geom_histogram()


ggplot (gapminder, aes(x=gdpPercap)) +
  geom_histogram()
```

Challenge 2

In the previous examples and challenge we've used the aes function to tell the geom_histogram() and geom_col() functions which columns of the data set to plot. Another aesthetic property we can modify is the color. Create a new bar (column) plot showing the gdp per capita of all countries in the Americas for the years 1952 and 2007, color coded by year.

```
gapminder_small <- gapminder %>%
filter(year == 2007 | year == 1952, continent == "Americas") %>%
mutate(year = factor(year))

ggplot(data = gapminder_small, aes(x = country, y=gdpPercap))+
  geom_col(aes(fill = year)) +
  coord_flip()


gapminder_small2 <- gapminder %>%
  filter(year %in% c(2007, 1952),
         continent == "Americas") %>%
  mutate(year = factor(year))

ggplot(data = gapminder_small2, aes(x = country, y=gdpPercap))+
  geom_col(aes(fill = year), position = "dodge") +
  coord_flip()
ggplot(data = gapminder_small2, aes(x = country, y=gdpPercap))+
  geom_col(aes(fill = year), position = "dodge") +
  coord_flip()
```

Intro to Raster Data

```
# file attributes
GDALinfo("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")

# save attribute data
HARV_dsmCrop_info <- capture.output(
  GDALinfo("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")
)

# open raster in R
DSM_HARV <-
  raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")
DSM_HARV

# data summary
summary(DSM_HARV)

# lil data summary
summary(DSM_HARV, maxsamp = ncell(DSM_HARV))

# save data in data.frame
# this format plays nicely with ggplot
DSM_HARV_df <- as.data.frame(DSM_HARV, xy = TRUE)

# look at structure of this data frame
str(DSM_HARV_df)

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

# quickplot
plot(DSM_HARV)
```

Challenge

Look at output from crs(DSM_HARV). What units are our data in? What UTM Zone are we in?

meters
meters
meters UTM zone 18N
meters, degree, 18N
meters, 18
meters, zone 18N
zone 18
18
18N

```
# Calculate Raster Min and Max Values
# "interrogate the raster"
minValue(DSM_HARV)
maxValue(DSM_HARV)
```

Challenge

Use the output from the GDALinfo() function to find out what NoDataValue is used for our DSM_HARV dataset.

```
# find "missing data" values
GDALinfo("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")
> ...
> apparent band summary:
> GDType hasNoDataValue NoDataValue blockSize1 blockSize2
>  1 Float64  TRUE                 -9999              1                   1697
```

-9999
^^
-9999
-9999

```
# Create A Histogram of Raster Values
ggplot() +
  geom_histogram(data = DSM_HARV_df, aes(HARV_dsmCrop))

# change width of bins
ggplot() +
  geom_histogram(data = DSM_HARV_df, aes(HARV_dsmCrop), binwidth = 3)
```

CHALLENGE

Use GDALinfo() to determine the following about the NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_DSMhill.tif file:

1. Does this file have the same CRS as DSM_HARV?

2.What is the NoDataValue?
3. What is resolution of the raster data?
4. How large would a 5x5 pixel area be on the Earth's surface?
5.Is the file a multi- or single-band raster?
Notice: this file is a hillshade. We will learn about hillshades in the Working with Multi-band Rasters in R episode.

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

1. yes
2. -9999
3. 1x1 meters
4. 25 sq meters
5. single

  1. yes
  2. -9999
  3. 1x1
  4. 5x5 meters
  5. single band raster

1. yes, 2. -9999, 3. 1 m, 4. 25 meter square, 5. single-band

Si
-9999
1 m
25 sq m
single-band

1)yes 2)-9999 3) 1x1 4)25 m^2 5)single

Plot Raster Data

```
#Making a new column using dplyr::mutate
DSM_HARV_df <- DSM_HARV_df %>%
          mutate(fct_elevation = cut(HARV_dsmCrop, breaks = 3))
#Then plot a bar chart
ggplot() +
  geom_bar(data = DSM_HARV_df, aes(fct_elevation))


# use dplyr to count the number of rows in our dataframe in each elevation cluster
DSM_HARV_df %>%
  group_by(fct_elevation) %>%
  count()

# Create custom bin sizes
custom_bins <- c(300, 350, 400, 450)

DSM_HARV_df <- DSM_HARV_df %>%
   mutate(fact_elevation2 = cut(HARV_dsmCrop, breaks = custom_bins))

# Look at unique values in fct_elevation2 column
unique(DSM_HARV_df$fct_elevation2)

# Use ggplot to plot bin sizes
ggplot() +
   geom_bar(data = DSM_HARV_df, aes(fct_elevation2))

# use dplyr to count the number of pixels in each bin
DSM_HARV_df %>%
   group_by(fact_elevation2) %>%
   count()

# map the data using the new elevation bands
ggplot() +
   geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation2)) +
   coord_quickmap()

# look at a pre-made color palette for plots
# the 3 argument returns 3 colors
terrain.colors(3)

# redraw our previous plot using terrain.colors()
ggplot() +
   geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation2)) +
   scale_fill_manual(values = terrain.colors(3)) +
  coord_quickmap()

# clearly define a color palette
my_col <- terrain.colors(3)

# redraw the same plot
# add a better legend title
ggplot() +
  geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation2)) +
  scale_fill_manual(values = my_col, name = "Elevation") +
  coord_quickmap()

# same plot again, this time with axis titles
ggplot() +
  geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation2)) +
  scale_fill_manual(values = my_col, name = "Elevation") +
  theme(axis.title = element_blank()) +
  coord_quickmap()
```

CHALLENGE

Create a plot of the Harvard Forest Digital Surface Model (DSM) that has:

1. Six classified ranges of values (break points) that are evenly divided among the range of pixel values.
2. Axis labels.
3. A plot title.

```
DSM_HARV_df <- DSM_HARV_df %>%
 mutate(fct_elevation6 = cut(HARV_dsmCrop, breaks = 6))

my.col6 = terrain.colors(6)

ggplot() +
 geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation6))+
 scale_fill_manual(values = my.col6, name = "Elevation") +
 ggtitle("Elevation map") +
 xlab('UTM E (m)') +
 ylab('UTM N (m)')+
 coord_quickmap()


DSM_HARV_df <- DSM_HARV_df %>%
 mutate(fct_elev_3 = cut(HARV_dsmCrop, breaks = 6))
DSM_HARV_df

ggplot() +
 geom_bar(data = DSM_HARV_df, aes(fct_elev_3))
```

```
my_col <- terrain.colors(6)

ggplot()+
  geom_raster(data = DSM_HARV_df, aes(x=x, y=y, fill=fct_elev_3))+
  scale_fill_manual(values = my_col, name = "elevation") +
  coord_quickmap()+
  ggtitle("Elevation plot w/ 6 classes")

DSM_HARV_df <- DSM_HARV_df  %>%
  mutate(fct_elevation_6 = cut(HARV_dsmCrop, breaks = 6))

ggplot() +
  geom_raster(data = DSM_HARV_df , aes(x = x, y = y,
                                       fill = fct_elevation_6)) +
  scale_fill_manual(values = terrain.colors(6), name = "Elevation") +
  ggtitle("Classified Elevation of Harvard Forest NEON Site") +
  xlab("UTM Easting (m)") +
  ylab("UTM Northing (m)") +
  coord_quickmap()




  DSM_HARV_df <- DSM_HARV_df %>%
  mutate(fct_elevation_3 = cut(HARV_dsmCrop, breaks = 6))
  my_col <- terrain.colors(6)
ggplot() +
  geom_raster(data = DSM_HARV_df , aes(x = x, y = y,
                                       fill = fct_elevation_3)) +
  scale_fill_manual(values = my_col, name = "Elevation") +
  xlab("UTM Easting Coordinate (m)")+
  ylab("UTM Northing Coordinate (m)")+
  ggtitle("Harvard Forest Digital Surface Model") +
  coord_quickmap()


  custom_bins <- c(300, 320, 340, 360, 380, 400, 420)
DSM_HARV_df <- DSM_HARV_df %>%
  mutate(fct_elevation3 = cut(HARV_dsmCrop, breaks = custom_bins))
my_col <- terrain.colors(6)
ggplot() +
  geom_raster(data = DSM_HARV_df, aes(x = x, y = y, fill = fct_elevation3)) +
  scale_fill_manual(values = my_col, name = "Elevation") +
  ggtitle("Elevation Map") +
  xlab("UTM E (m)") +
  ylab("UTM N (m)") +
  coord_quickmap()
```

Layering plots

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

# view info about the object in the console
DSM_hill_HARV

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

# inspect the dataframe (output to console)
str(DSM_hill_HARV_df)

# plot the dataframe
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()

# layering two rasters on top of one another
# Order matters
ggplot() +
  # 1st layer
  geom_raster(data = DSM_HARV_df, aes(x = x. y = y, fill = HARV_dsmCrop)) +
  # 2nd layer
  geom_raster(data = DSM_hill_HARV_df, aes(x = x, y = y, alpha = HARV_DSMhill)) +
  # add a scale
  scale_fill_viridis_c() +
  scale_alpha(range = c(0.15, 0.65), guide = "none") +
  ggtitle("Elevation with hillshade") +
  coord_quickmap()
```

CHALLENGE

Setup code:

```
# CREATE DTM MAP
# import DTM
DTM_SJER <-
raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DTM/SJER_dtmCrop.tif")DTM_SJER_df <-
as.data.frame(DTM_SJER, xy = TRUE)
# DTM Hillshade
DTM_hill_SJER <-
raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DTM/SJER_dtmHill.tif")DTM_hill_SJER_df <-
as.data.frame(DTM_hill_SJER, xy = TRUE)

# plot the data
ggplot() +
 geom_raster(data = DTM_SJER_df, aes(x = x, y = y, fill = SJER_dtmCrop)) +
 geom_raster(data = DTM_hill_SJER_df, aes(x = x, y = y, alpha = SJER_dtmHill)) +
 scale_fill_viridis_c() +
 scale_alpha(range = c(0.4, 0.7), guide = "none") +
 theme_bw() +
 theme(panel.grid.major = element_blank(),
       panel.grid.minor = element_blank(),
       axis.title = element_blank()) +
 ggtitle("DTM with Hillshade - SJER site")
```

Mariah's answer:

```
# CREATE DTM MAP
# import DTM
DTM_SJER <- raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DTM/SJER_dtmCrop.tif")
DTM_SJER_df <- as.data.frame(DTM_SJER, xy = TRUE)

# DTM Hillshade
DTM_hill_SJER <- raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DTM/SJER_dtmHill.tif")
DTM_hill_SJER_df <- as.data.frame(DTM_hill_SJER, xy = TRUE)

ggplot() +
    geom_raster(data = DTM_SJER_df ,
                aes(x = x, y = y,
                    fill = SJER_dtmCrop,
                    alpha = 2.0)
    ) +
    geom_raster(data = DTM_hill_SJER_df,
               aes(x = x, y = y,
                   alpha = SJER_dtmHill)
    ) +
    scale_fill_viridis_c() +
    guides(fill = guide_colorbar()) +
    scale_alpha(range = c(0.4, 0.7), guide = "none") +
    theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank()) +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank()) +
    ggtitle("DTM with Hillshade") +
    coord_quickmap()
```

Reproject Raster Data

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

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

```
# also load the Harvard hillshade data
DTM_hill_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_DTMhill_WGS84.tif")

# convert raster-type data to data.frames
DTM_HARV_df <- as.data.frame(DTM_HARV, xy = TRUE)
str(DTM_HARV_df)

DTM_hill_HARV_df <- as.data.frame(DTM_hill_HARV, xy = TRUE)
str(DTM_hill_HARV_df)

# make a map of DTM and hillshade data (same process as previous episode)
ggplot() +
  geom_raster(data = DTM_HARV_df, aes(x = x, y = y, fill = HARV_dtmCrop)) +
  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()

# these results are "curious"

# plotting both layers didn't work well
# let's just plot the DTM data alone
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()

# look at only the hillshade data
ggplot() +geom_raster(data = DTM_hill_HARV_df,
    aes(x = x, y = y,
    alpha = HARV_DTMhill_WGS84)) +
    coord_quickmap()

# the axes don't match!
# we must reproject...
```

End of day 2 feedback

Please fill out this ANONYMOUS feedback form about the workshop today:
https://forms.gle/Gf8zEdQCqT4VvNnD7

Day 3 :

Reproject Raster Data

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

If you need to re-load your data:

```
# load required libraries
library(raster)
library(sp)
library(rgdal)
library(dplyr)
library(ggplot2)

# load Harvard terrain model
DTM_HARV <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_dtmCrop.tif")
# also load the Harvard hillshade data
DTM_hill_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_DTMhill_WGS84.tif")

# convert raster-type data 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)

# plotting both layers didn't work well
# let's just plot the DTM data alone
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()

# look at only the hillshade data
ggplot() +
  geom_raster(data = DTM_hill_HARV_df,
                    aes(x = x, y = y,
                        alpha = HARV_DTMhill_WGS84)) +
  coord_quickmap()

# The units are different?!
```

Challenge

View the CRS for each of these two datasets.(share the command)

What projection does each use?

```
crs(DTM_HARV)
projection:  UTM in meters
crs(DTM_hill_HARV)
projection: WGS84 lat and long

crs(DTM_HARV) #utm zone 18n
crs(DTM_hill_HARV) #WGS84


crs(DTM_HARV) UTM 18N
crs(DTM_hill_HARV) longlat

crs(DTM_HARV) #UTM Zone 18N
crs(DTM_hill_HARV)#WGS84


# reproject our DTM_hill_HARV raster data to match the DTM_HARV raster CRS
DTM_hill_UTMZ18N_HARV <- projectRaster(DTM_hill_HARV, crs = crs(DTM_HARV))
# I get a lot of errors with this function. Run it again and they disappear. Success?!
# This has happened to most of the instructors too. It doesn't seem to affect the outcome - the data are
# correctly reprojected. The error is a known issue that can be ignored.

# compare each CRS again
crs(DTM_hill_UTMZ18N_HARV)
crs(DTM_hill_HARV)

# Find the extent of the reprojected raster and compare to the first one
extent(DTM_hill_HARV)
extent(DTM_hill_UTMZ18N_HARV)

# Also check the resolution
res(DTM_hill_HARV)
res(DTM_hill_UTMZ18N_HARV)

# ensure a resolution match by using res(DTM_HARV) as a variable
DTM_hill_UTMZ18N_HARV <- projectRaster(DTM_hill_HARV,
                                crs = crs(DTM_HARV),
                                res = res(DTM_HARV))
# double-check our resolution to be sure it worked
res(DTM_hill_UTMZ18N_HARV)
res(DTM_HARV)
# success!
```

Notes on the `terra` package:

Most of what we just used in raster are the same or similar in terra

raster::extent() -> terra::ext()
raster::res() -> terra::res()
raster::crs() -> terra::crs()
raster::projectRaster -> terra::project() which can take SpatRaster or SpatVector objects and convert them

```
# create a dataframe from our newly reprojected raster.
DTM_hill_HARV_2_df <- as.data.frame(DTM_hill_UTMZ18N_HARV, xy = TRUE)

# create a plot of this data
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()

# NEON dataset metadata: https://www.neonscience.org/data-collection/airborne-remote-sensing
```

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:

```
SJ_DSM <- raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DSM/SJER_dsmCrop.tif")
SJER_hill_DTM <-
raster("data/NEON-DS-Airborne-Remote-Sensing/SJER/DSM/SJER_DSMhill_WGS84.tif")

extent(SJ_DSM)
extent(SJER_hill_DTM)    #whoops, used the terra command for a raster obj
res(SJ_DSM)
res(SJER_hill_DTM)

SJER_hill_UTM11N_DTM <- projectRaster(SJER_hill_DTM,
                                      crs = crs(SJ_DSM),
                                      res = res(SJ_DSM))

SJ_DSM_df <- as.data.frame(SJ_DSM, xy=T)
SJER_hill_UTM11N_DTM_df <- as.data.frame(SJER_hill_UTM11N_DTM, xy=T)
as_tibble(SJ_DSM_df)

ggplot()+
  geom_raster(data = SJ_DSM_df, aes(x=x,y=y, fill = SJER_dsmCrop))+
  geom_raster(data = SJER_hill_UTM11N_DTM_df, aes(x=x, y=y, alpha = SJER_DSMhill_WGS84))+
  scale_fill_gradientn(name = "elevation", colors = terrain.colors(10))+
  coord_quickmap()


# import DSM
DSM_SJER <- raster("Data/NEON-DS-Airborne-Remote-Sensing/SJER/DSM/SJER_dsmCrop.tif")
# import DSM hillshade
DSM_hill_SJER_WGS <-
  raster("Data/NEON-DS-Airborne-Remote-Sensing/SJER/DSM/SJER_DSMhill_WGS84.tif")

#check CRS
crs(DSM_SJER)#utm zone 11 wgs84 m
crs(DSM_hill_SJER_WGS)#long lat wgs84

# reproject raster
DTM_hill_UTMZ18N_SJER <- projectRaster(DSM_hill_SJER_WGS,
                                       crs = crs(DSM_SJER))

#check that the resolution matches
res(DSM_SJER)#1 1
res(DSM_hill_SJER_WGS)# 1.091943e-05 9.264392e-06
#the resolution is not the same!

#specify resolution on re-projection
DTM_hill_UTMZ18N_SJER = projectRaster(DSM_hill_SJER_WGS, crs = crs(DSM_SJER), res =
res(DSM_hill_SJER_WGS))


# convert to data.frames
DSM_SJER_df <- as.data.frame(DSM_SJER, xy = TRUE)

DSM_hill_SJER_df <- as.data.frame(DTM_hill_UTMZ18N_SJER, xy = TRUE)

ggplot() +
  geom_raster(data = DSM_hill_SJER_df,
              aes(x = x, y = y,
                  alpha = SJER_DSMhill_WGS84)) +
  geom_raster(data = DSM_SJER_df,
              aes(x = x, y = y,
                  fill = SJER_dsmCrop,
                  alpha=0.8)) +
  scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10)) +
  coord_quickmap()
```

Raster Calculations

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

(If necessary) reload libraries and data from Day 2

```
library(raster)
library(rgdal)
library(dplyr)
library(ggplot2)

DTM_HARV <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_dtmCrop.tif")
DSM_HARV <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")
```

Exercise

Use the GDALinfo() function to view information about the DTM and DSM data files. Do the two rasters have the same or different CRSs and resolutions? Do they both have defined minimum and maximum values?

Solutions

```
GDALinfo("Data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_dtmCrop.tif")#UTM, 18, WGS84, M
GDALinfo("Data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")#SAME AS ABOVE
#BOTH HAVE MINS AND MAXS
```

Both have defined mins and maxs

```
#plot the DTM elevation data alone
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()

# plot the DSM elevation data alone
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()

# create a dataframe so we can plot with ggplot.
CHM_HARV <- DSM_HARV - DTM_HARV
```

```
CHM_HARV_df <- as.data.frame(CHM_HARV, xy = TRUE)

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

# look at the distribution of pixel values in our newly created Canopy Height Model (CHM).
ggplot(CHM_HARV_df) +
  geom_histogram(aes(layer))
```

Challenge: Explore CHM Raster Values

It's often a good idea to explore the range of values in a raster dataset just like we might explore a dataset that we collected in the field.

1. What is the min and maximum value for the Harvard Forest Canopy Height Model (CHM_HARV) that we just created?
2. What are two ways you can check this range of data for CHM_HARV?
3. What is the distribution of all the pixel values in the CHM?
4. Plot a histogram with 6 bins instead of the default and change the color of the histogram.
5. Plot the CHM_HARV raster using breaks that make sense for the data. Include an appropriate color palette for the data, plot title and no axes ticks / labels.

>> Try this by yourself later if you'd like!

Efficient Raster Calculations: Overlay Function

```
# format: function_name <- function(variable1, variable2) { WhatYouWantDone, WhatToReturn}

# Try this with HARV data
CHM_ov_HARV <- overlay(DSM_HARV,
                  DTM_HARV,
                  fun = function(r1, r2) { return( r1 - r2) })

# convert our new object to a data frame for plotting with ggplot.
CHM_ov_HARV_df <- as.data.frame(CHM_ov_HARV, xy = TRUE)

# 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()
```

Export a GeoTIFF

```
# write this raster object to a GeoTIFF file
# name it CHM_HARV.tiff
# specify the output format ("GTiff")
# the no data value NAflag = -9999
# also tell R to overwrite any data that is already in a file of the same name

writeRaster(CHM_ov_HARV, "CHM_HARV.tiff",
        format="GTiff",
        overwrite=TRUE,
        NAflag=-9999)
```

(Skipped) Challenge: Explore CHM Raster Values

It's often a good idea to explore the range of values in a raster dataset just like we might explore a dataset that we collected in the field.

1. What is the min and maximum value for the Harvard Forest Canopy Height Model (CHM_HARV) that we just created?
2. What are two ways you can check this range of data for CHM_HARV?
3. What is the distribution of all the pixel values in the CHM?
4. Plot a histogram with 6 bins instead of the default and change the color of the histogram.
5. Plot the CHM_HARV raster using breaks that make sense for the data. Include an appropriate color palette for the data, plot title and no axes ticks / labels.

Solution is available at
https://datacarpentry.org/r-raster-vector-geospatial/04-raster-calculations-in-r/index.html#answers
(click the down arrow next to "Answers")

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.

>>> Try this by yourself if you'd like!

Work With Multi-Band Rasters

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

```
# using the raster() function, R only reads first band of RasterStack object
RGB_band1_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")
# :(

# convert this data to a data frame in order to plot it with ggplot.
RGB_band1_HARV_df  <- as.data.frame(RGB_band1_HARV, xy = TRUE)

# look at top of data frame
# data column is named "layer"
head(RGB_band1_HARV_df)

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

# look at raster object
RGB_band1_HARV

# notice it says that there is 1 (of 3) bands
```

Notes on the `terra` package:

- The `terra` package handles multiband rasters a little differently than `raster`. Terra loads all bands of a multiband raster by default, not just the first (raster only loads the first). Terra uses the "rast" function to load a raster, and to get just the first band, you would use terra::rast(<filename>, lyrs=1)

- Terra doesn't need to stack bands since it loads them all by default. But, if you did want to "stack" individual raster layers in terra you would to that by concatenating them (c() function).
- plotRGB is (essentially) the same in terra and raster

```
# import the second band
RGB_band2_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif", band
= 2)

#convert this band to a data frame and plot
RGB_band2_HARV_df <- as.data.frame(RGB_band2_HARV, xy = T)
head(RGB_band2_HARV_df)
ggplot() +
  geom_raster(data = RGB_band2_HARV_df,
              aes(x = x, y = y, alpha = layer)) +
  coord_equal()
```

Raster Stacks in R

```
# bring in all bands of a multi-band raster, we use thestack() function.
RGB_stack_HARV <-
stack("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_RGB_Ortho.tif")

# preview the attributes of our stack object:
RGB_stack_HARV

#view the attributes of each band (slots) in the stack in a single output
RGB_stack_HARV@layers

# only look at second band
RGB_stack_HARV[[2]]

# Create a dataframe
RGB_stack_HARV_df  <- as.data.frame(RGB_stack_HARV, xy = TRUE)

# plot the 2nd band as a raster
ggplot() +
  geom_raster(data = RGB_stack_HARV_df,
              aes(x = x, y = y, alpha = HARV_RGB_Ortho_2)) +
  coord_quickmap()
```

```
## Create A Three Band Image
# this function makes something that looks more like an RGB photograph
plotRGB(RGB_stack_HARV, #this is the rasterStack object, not a df
        r = 1, g = 2, b = 3)

# stretch the values to extend to the full 0-255 range of potential values
# this will increase the visual contrast of the image.
plotRGB(RGB_stack_HARV,
        r = 1, g = 2, b = 3,
        scale = 800,
        stretch = "lin")
# notice that this image is a lot "brighter" We stretched the pixel values to their max (255)!

# stretch it again, using  different (hist) method
plotRGB(RGB_stack_HARV,
        r = 1, g = 2, b = 3,
        scale = 800,
        stretch = "hist")
# also has bright spots, but also some dark regions. The pixels are prob closer to a "bell curve"
```

Challenge - NoData Values

Let’s explore what happens with NoData values when working with RasterStack objects and using the plotRGB() function. We will use the HARV_Ortho_wNA.tif GeoTIFF file in the NEON-DS-Airborne-Remote-Sensing/HARVRGB_Imagery/ directory.

1. View the files attributes. Are there NoData values assigned for this file?
2. If so, what is the NoData Value?
3. How many bands does it have?
4. Load the multi-band raster file into R.
5. Plot the object as a true color image.
6. What happened to the black edges in the data?
7. What does this tell us about the difference in the data structure between HARV_Ortho_wNA.tif and HARV_RGB_Ortho.tif (R object RGB_stack). How can you check?

```
# import data
RGB_stack_HARV_na <-
stack("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_Ortho_wNA.tif")
# lookee
GDALinfo("data/NEON-DS-Airborne-Remote-Sensing/HARV/RGB_Imagery/HARV_Ortho_wNA.tif")
# view the attributes of each band (slots) in the stack
RGB_stack_HARV_na@layers
# plot
plotRGB(RGB_stack_HARV_na,
        r = 1, g = 2, b = 3)
plotRGB(RGB_stack_HARV_na,
        r = 1, g = 2, b = 3, colNA='red') # make the NA values RED


# brick it
object.size(RGB_stack_HARV)
RGB_brick_HARV <- brick(RGB_stack_HARV)
object.size(RGB_brick_HARV)
```

Open and Plot Shapefiles

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

```
library(raster)
#install.packages("sf")
library(sf)

#import our AOI:
aoi_boundary_HARV <- st_read(
  "data/NEON-DS-Site-Layout-Files/HARV/HarClip_UTMZ18.shp")

# some basic information about the data automatically prints

#view the geometry type for our AOI shapefile
st_geometry_type(aoi_boundary_HARV)

#check what CRS this file data is in:
st_crs(aoi_boundary_HARV)

#find the extent of our AOI
st_bbox(aoi_boundary_HARV)

#view all of the metadata and attributes for this shapefile
#similar to output when object was imported
aoi_boundary_HARV

# no need to convert data
# plot
ggplot() +
  geom_sf(data = aoi_boundary_HARV, size = 3, color = "black", fill = "cyan1") +
  ggtitle("AOI Boundary Plot") +
  coord_sf()
# wow, that's a nice square!


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

Solutions:

```
lines_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")
point_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")
class(lines_HARV)
class(point_HARV)

st_crs(lines_HARV)
st_bbox(lines_HARV)

st_crs(point_HARV)
st_bbox(point_HARV)

lines_HARV
point_HARV

# Answer the following questions:
# 1. What type of R spatial object is created when you import each layer?
# Multiline string for lines_HARV
# point for point_HARV
#
# 2. What is the CRS and extent for each object?
# CRS: WGS 84 / UTM zone 18N - both
# xmin: 730741.2 ymin: 4711942 xmax: 733295.5 ymax: 4714260 - lines_HARV
# xmin: 732183.2 ymin: 4713265 xmax: 732183.2 ymax: 4713265 - point_HARV
#
# 3. Do the files contain points, lines, or polygons?
# lines_HARV
# point_HARV
#
# 4. How many spatial objects are in each file?
# 13 feat in lines_HARV
# 1 feat in point_HARV
# 1. roads = MULTILINESTRING, flux tower = points 2) roads =WGS 84 / UTM zone 18N; flux tower =WGS 84 / UTM zone 18N  3)  roads = lines, flux tower = points 4. How many spatial objects are in each file?
```

Explore and Plot by Vector Layer Attributes

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

```
# Make sure we have everything for the lesson
library(sf)
library(ggplot2)
library(dplyr)

lines_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")
point_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")

#import more layers
lines_HARV <- st_read("Data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")
point_HARV <- st_read("Data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")

class(lines_HARV)
#sf - data.frame
class(point_HARV)
#sf - data.frame

#CRS and extent
st_crs(lines_HARV)
#WGS 84, UTM ZONE 18N
st_crs(point_HARV)
#WGS 84, UTM ZONE 18N

#extent
st_bbox(lines_HARV)
#xmin      ymin      xmax      ymax
#730741.2 4711942.0  733295.5 4714260.0
st_bbox(point_HARV)
#xmin      ymin      xmax      ymax
#732183.2 4713265.0  732183.2 4713265.0
lines contains lines (13), and points contains 1 point

# how many columns/attributes
ncol(lines_HARV)

#view the individual name of each attribute
names(lines_HARV)
```

#view just the first 6 rows of attribute values
head(lines_HARV)

Challenge: Attributes for Different Spatial Classes

Explore the attributes associated with the point_HARV and aoi_boundary_HARV spatial objects.

1. How many attributes does each have?
2. Who owns the site in the point_HARV data object?
3. Which of the following is NOT an attribute of the point_HARV data object?

A) Latitude B) County C) Country

1. point_HARV:15, 2)Harvard University, LTER 3) C) Country

#1. 15, 2. Harvard University, LTER, 3. C) Country

### Explore Values within One Attribute

```
# see the contents of the TYPE field of our lines feature
lines_HARV$TYPE
unique(lines_HARV$TYPE)


# subset data
# we might be interested only in features that are of TYPE “footpath”
footpath_HARV <- lines_HARV %>%
  filter(TYPE == "footpath")
footpath_HARV
#how many footpaths?
nrow(footpath_HARV)

# do the same, but specify that we want filter() function from dplyr
footpath_HARV <- lines_HARV %>%
  dplyr::filter(TYPE == "footpath")
footpath_HARV


# plot only the footpath lines:
ggplot() +
  geom_sf(data = footpath_HARV) +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Footpaths") +
  coord_sf()
```

```
# Why does the plot look like there is only one feature?
# adjust the colors used in our plot.
# convert OBJECTID column to categorical data (factor)
# use size argument outside of aesthetic to universally make the lines bigger
# change legend title using labs()
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") +
  coord_sf()
```

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.

```
stonewall_HARV <- lines_HARV %>%
  filter(TYPE == "stone wall")

ggplot()+
  geom_sf(data = stonewall_HARV,
          aes(color = factor(OBJECTID)),
          size = 1.5)+
  labs(color = "Stone Wall ID")+
  ggtitle("NEON Harvard Forest Field Site",
          subtitle = "Stone Wall")+
  coord_sf()

  stonewall_HARV <- lines_HARV %>%
  dplyr::filter(TYPE == "stone wall")
ggplot() +
  geom_sf(data = stonewall_HARV, aes(color = factor(OBJECTID)), size = 1.5) +
  labs(color = 'Stone Wall ID') +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Stone walls") +
  coord_sf()



unique(lines_HARV$TYPE)
stone_wall_HARV <- lines_HARV %>%
  dplyr::filter(TYPE == "stone wall")
ggplot() +
  geom_sf(data = stone_wall_HARV, aes(color = factor(OBJECTID)), size=1.5) +
  labs(color = "Stone Wall ID") +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Stone Wall") +
  coord_sf()

  stonewall_HARV <- lines_HARV %>%
  dplyr::filter(TYPE == "stone wall")

ggplot() +
  geom_sf(data = stonewall_HARV, aes(color = factor(OBJECTID)), size = 1.5) +
  labs(color = 'stone wall ID') +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "stone wall") +
  coord_sf()



unique(lines_HARV$TYPE)
stonewalls_HARV <- lines_HARV %>%
  filter (TYPE =="stone wall")
ggplot() +
  geom_sf(data = stonewalls_HARV, aes(color = factor(OBJECTID)),size=1.5) +
  labs(color = 'Footpath ID') +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Stone Walls") +
  coord_sf()




## Customize plots

# look at data types
unique(lines_HARV$TYPE)

# create a palette of four colors, one for each type
road_colors <- c("blue", "green", "navy", "purple")

# tell ggplot to use these colors when we plot the data.
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") +
  coord_sf()
  # sorry, foot paths


# same plot, different line widths
# set four different line widths
line_widths <- c(1, 2, 3, 4)

# use those line widths when we plot the data.
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()
```

Challenge: Plot Lines by Attribute

Create a plot that emphasizes only roads where bicycles and horses are allowed. To emphasize this, make the lines where bicycles are not allowed THINNER than the roads where bicycles are allowed. NOTE: this attribute information is located in the lines_HARV$BicyclesHo attribute.

Be sure to add a title and legend to your map. You might consider a color palette that has all bike/horse-friendly roads displayed in a bright color. All other lines can be black.

Anonymous feedback for today:
https://forms.gle/5he4SNXP9iktu9eA9

Day 4:

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

```
# Get ready for today!
library(dplyr)
library(ggplot2)
library(sf)

aoi_boundary_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HarClip_UTMZ18.shp")
lines_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARV_roads.shp")
point_HARV <- st_read("data/NEON-DS-Site-Layout-Files/HARV/HARVtower_UTM18N.shp")

road_colors <- c("blue", "green", "navy", "purple")
```

Challenge: Plot Lines by Attribute

Create a plot that emphasizes only roads where bicycles and horses are allowed. To emphasize this, make the lines where bicycles are not allowed THINNER than the roads where bicycles are allowed. NOTE: this attribute information is located in the lines_HARV$BicyclesHo attribute.

Be sure to add a title and legend to your map. You might consider a color palette that has all bike/horse-friendly roads displayed in a bright color. All other lines can be black.

```
# emphasize paths where horses and bicycles are allowed
ggplot() +
  geom_sf(data = lines_HARV) +
  geom_sf(data = lines_HARV[lines_HARV$BicyclesHo %in% "Bicycles and Horses Allowed",],
linewidth = 3, colour = "magenta") +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Roads Where Bikes and Horses Are Allowed") +
  coord_sf()


# Greg's solution:
# look at uniques values in the BicyclesHo column
unique(lines_HARV$BicyclesHo)

# make a dataframe of the roads where lines_HARV$BicyclesHo is not NA
lines_removeNA <- lines_HARV[!is.na(lines_HARV$BicyclesHo),]
# then filter for rows where "Bicycles and Horses Allowed"
lines_showHarv <- lines_removeNA %>% filter(BicyclesHo == "Bicycles and Horses Allowed")

#plot the data. first plot all the roads
ggplot() +
  geom_sf(data = lines_HARV) +
  #then overlay the lines_showHarv data in a bigger size and brighter color
  ###if "size" doesn't work for you, use "linewidth"
geom_sf(data = lines_showHarv, aes(color = BicyclesHo), size = 2) +
  scale_color_manual(values = "magenta") +
  ggtitle("NEON Harvard Forest Field Site", subtitle = "Roads Where Bikes and Horses Are Allowed") +
  coord_sf()
```

Plot Multiple Shapefiles

```
# create a plot with the site boundary as the first layer
# then layer the tower location and road data on top using +.
ggplot() +
  geom_sf(data = aoi_boundary_HARV, fill = "grey", color = "grey") +
  geom_sf(data = lines_HARV, aes(color = TYPE), size = 1, linewidth = 1) +
  geom_sf(data = point_HARV) +
  ggtitle("NEON Harvard Forest Field Site") +
  coord_sf()

# add the tower to the map
ggplot() +
  geom_sf(data = aoi_boundary_HARV, fill = "grey", color = "grey") +
  geom_sf(data = lines_HARV, aes(color = TYPE),
          show.legend = "line", size = 1, linewidth = 1) +
  geom_sf(data = point_HARV, aes(fill = Sub_Type), color = "black") +
  scale_color_manual(values = road_colors) +
  scale_fill_manual(values = "black") +
  ggtitle("NEON Harvard Forest Field Site") +
  coord_sf()
```

Challenge: Plot Polygon by Attribute

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.

Modify the plot above. Tell R to plot each point, using a different symbol of shape value.

Answers:

```
#read in the data and see how many unique soils are represented in the soilTypeOr attribute.
plot_locations <-
st_read("data/NEON-DS-Site-Layout-Files/HARV/PlotLocations_HARV.shp")

# look at attributes
names(plot_locations)

# look at unique vals in soilTypeOr column
unique(plot_locations$soilTypeOr)

# create a new color palette
blue_orange <- c("cornflowerblue", "tomato")

# plot! that! dirt!
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, colour = NA))) +
  ggtitle("NEON Harvard Forest Field Site") +
  coord_sf()

# now alter the symbols for each and override the aesthetics for the legends/guides
ggplot() +
  geom_sf(data = lines_HARV, aes(color = TYPE), show.legend = "line", size = 1) +
  geom_sf(data = plot_locations, aes(fill = soilTypeOr, shape = soilTypeOr),
          show.legend = 'point', size = 3) +
  scale_shape_manual(name = "Soil Type", values = c(21, 22)) +
  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 = c(21, 22),
     color = blue_orange))) +
  ggtitle("NEON Harvard Forest Field Site") +
  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.

Solution:

```
NEON_canopy<-
raster("Data/NEON-DS-Airborne-Remote-Sensing/HARV/CHM/HARV_chmCrop.tif")
NEON_canopy_df=as.data.frame(NEON_canopy, xy = TRUE)

ggplot() +
  geom_raster(data = NEON_canopy_df, aes(x = x, y = y, fill = HARV_chmCrop)) +
  geom_sf(data = lines_HARV, color = "black") +
  geom_sf(data = aoi_boundary_HARV, color = "red", linewidth = 1) +
  geom_sf(data = point_HARV, pch = 8) +
  ggtitle("NEON Harvard Site and Canopy Height Model") +
  coord_sf()


r <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/CHM/HARV_chmCrop.tif")
r_df <- as.data.frame(r, xy = T)
head(r_df)

ggplot() +
  geom_raster(data = r_df, aes(x=x, y=y, fill=HARV_chmCrop))+
  scale_fill_gradientn(colors = terrain.colors(10))+
  geom_sf(data = aoi_boundary_HARV)+
  geom_sf(data = lines_HARV)+
  geom_sf(data = point_HARV)+
  ggtitle("NEON AOI CHM with AOI, roads, and tower location")+
  coord_sf()
```

Handling Spatial Projection

```
# libraries
library(raster)
library(ggplot2)
library(dplyr)
library(sf)

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

# inspect shapefile metadata/attributes
st_crs(point_HARV)                    # CRS info

# Import US state boundary layer
# TRY:
state_boundary_US <- st_read("data/NEON-DS-Site-Layout-Files/US-Boundary-
Layers/US-State-Boundaries-Census-2014.shp")

# If you get an error, try:
state_boundary_US <- st_read("data/NEON-DS-Site-Layout-Files/US-Boundary-
Layers/US-State-Boundaries-Census-2014.shp", type=7)

# plot boundary layer
ggplot() +
   geom_sf(data = state_boundary_US) +
   ggtitle("Map of Contiguous US State Boundaries") +
   coord_sf()

# open US boundary layer
country_boundary_US <- st_read(
"data/NEON-DS-Site-Layout-Files/US-Boundary-Layers/US-Boundary-Dissolved-
States.shp"
)

# Plot the country boundary with the state boundaries
# change "size" argument to "linewidth" as needed depending on package versions/updates
ggplot() +
  geom_sf(data = country_boundary_US, color = "gray18", size = 2) +   # size OR linewidth
  geom_sf(data = state_boundary_US, color = "gray40") +
  ggtitle("Map of Contiguous US State Boundaries") +
  coord_sf()

# Check CRS of different layers
st_crs(point_HARV)
st_crs(country_boundary_US)
st_crs(state_boundary_US)

# Bounding boxes
st_bbox(point_HARV)
```

```
# The CRS for point_HARV is different from the boundary vectors
# ggplot can reproject on the fly

ggplot() +
  geom_sf(data = country_boundary_US, color = "gray18", size = 2) +   # size OR linewidth
  geom_sf(data = state_boundary_US, color = "gray40") +
  geom_sf(data = point_HARV, shape = 19, color = "purple") +
  ggtitle("Map of Contiguous US State Boundaries") +
  coord_sf()
```

Convert from .csv to a Shapefile

```
# Load a csv
plot_locations_HARV <-
read.csv("data/NEON-DS-Site-Layout-Files/HARV/HARV_PlotLocations.csv")

# inspect dataframe
str(plot_locations_HARV)
head(plot_locations_HARV$easting)
head(plot_locations_HARV$northing)
head(plot_locations_HARV$geodeticDa)
head(plot_locations_HARV$utmZone)

# get CRS from a different shapefile/vector
st_crs(point_HARV)

# Apply CRS from shapefile vector to the dataframe loaded from CSV
utm18nCRS <- st_crs(point_HARV)

# Convert CSV data to sf object with a CRS
plot_locations_sp_HARV <- st_as_sf(plot_locations_HARV,
                                   coords = c("easting", "northing"),
                                   crs = utm18nCRS)

# Check CRS of result object
st_crs(plot_locations_sp_HARV)


# Plot the data
ggplot() +
   geom_sf(data = plot_locations_sp_HARV) +
   ggtitle("Map of Plot Locations")

 # Redraw the plot with area of interest boundary

ggplot() +
   geom_sf(data = aoi_boundary_HARV) +
   geom_sf(data = plot_locations_sp_HARV) +
   ggtitle("Map of Plot Locations with AOI")
```

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_locations_HARV_2new <-
  read.csv("data/NEON-DS-Site-Layout-Files/HARV/HARV_2NewPhenPlots.csv")

DTM_hill_HARV <-
raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/DTM/HARV_DTMhill_WGS84.tif")
latlonWGS84 <- crs(DTM_hill_HARV)

plot_locations_sp_HARV_2new <- st_as_sf(plot_locations_HARV_2new, coords = c("decimalLon",
"decimalLat"), crs = latlonWGS84) #I'm not sure this object name is long enough
st_crs(plot_locations_sp_HARV_2new)

ggplot() +
  geom_sf(data = plot_locations_sp_HARV_2new) +
  ggtitle("Map of Plot Locations")
```

Below did not plot correctly------> Yes! I see the suggestions fixed it. Thanks Darren!

```
phenology <- read.csv("data/NEON-DS-Site-Layout-Files/HARV/HARV_2NewPhenPlots.csv")
#st_crs(phenology)
str(phenology)   #x = 42.5 (lat), y= -72.2 (long)
##The location is still in Harvard, so we can still use (UTM Zone 18N CRS)
#utm18nCRS_phen <- st_crs(phenology)
# The error could be the position of the coordinate columns in the next line
# Try coords = c("decimalLon", "decimalLat")
# Also, the utm18nCRS seems to be an issue - try: crs = st_crs(country_boundary_US)
plot_locations_sp_HARV_phen <- st_as_sf(phenology, coords = c("decimalLat","decimalLon"),
                                        crs = utm18nCRS)

#Plot the data
ggplot() +
  geom_sf(data = plot_locations_sp_HARV) +
  geom_sf(data = plot_locations_sp_HARV_phen) +
  ggtitle ("Map of Plot Locations and Phenology") +
  coord_sf()
```

Manipulate Raster Data

```
# libraries
library(raster)
library(ggplot2)
library(dplyr)
library(sf)

# if needed, read the HARV canopy height model
CHM_HARV <- raster("data/NEON-DS-Airborne-Remote-Sensing/HARV/CHM/HARV_chmCrop.tif")

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

# if needed, also reimport the area of interest
aoi_boundary_HARV <- st_read(
"data/NEON-DS-Site-Layout-Files/HARV/HarClip_UTMZ18.shp"
)

# Also add plot locations shapefile/vector as needed
plot_locations_sp_HARV <-
st_read("data/NEON-DS-Site-Layout-Files/HARV/PlotLocations_HARV.shp")

ggplot() +
  geom_raster(data = CHM_HARV_df, aes(x = x, y = y, fill = HARV_chmCrop)) +
  scale_fill_gradientn(name = "Canopy Height", colors = terrain.colors(10)) +
  geom_sf(data = aoi_boundary_HARV, color = "blue", fill = NA) +
   ggtitle("Lets crop this raster with the AOI box", subtitle="same as 'clip' if you are an ESRI user"") +
  coord_sf()

# Crop the raster to the bounding box
CHM_HARV_Cropped <- crop(x = CHM_HARV, y = aoi_boundary_HARV)
# Make data.frame
CHM_HARV_Cropped_df <- as.data.frame(CHM_HARV_Cropped, xy = TRUE)

# Plot the cropped area relative to the full extent of the CHM raster
ggplot() +
  geom_sf(data = st_as_sfc(st_bbox(CHM_HARV)), fill = "green",
          color = "green", alpha = .2) +
  geom_raster(data = CHM_HARV_Cropped_df,
              aes(x = x, y = y, fill = HARV_chmCrop)) +
  scale_fill_gradientn(name = "Canopy Height", colors = terrain.colors(10)) +
  coord_sf()


# Plot of just the zoomed in cropped section:
ggplot() +
  geom_raster(data = CHM_HARV_Cropped_df,
            aes(x = x, y = y, fill = HARV_chmCrop)) +
  geom_sf(data = aoi_boundary_HARV, color = "blue", fill = NA) +
  scale_fill_gradientn(name = "Canopy Height", colors = terrain.colors(10)) +
  coord_sf()

# Get the bounding box of the cropped extent for comparison
st_bbox(CHM_HARV_Cropped)
st_bbox(CHM_HARV)

# Extract tree height values in the area of interest or aoi
tree_height <- extract(x = CHM_HARV, y = aoi_boundary_HARV, df = TRUE)

# View desc stats of the extracted data
summary(tree_height$HARV_chmCrop)

# Get a specific statistic - for example the mean
mean_tree_height_AOI <- extract(x = CHM_HARV, y = aoi_boundary_HARV, fun = mean)
mean_tree_height_AOI # output = 22.43018

# Extract with a buffer
mean_tree_height_tower <- extract(x = CHM_HARV,
                                  y = aoi_boundary_HARV,
                                  buffer = 20,
                                  fun = mean)
```

Notes on Terra R package

Terra also has an extract function (terra::extract) that is quite similar to raster. It doesn't have an option to return a dataframe though.

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.

Solutions:

```
# extract data at each plot locationmean_tree_height_plots_HARV <- extract(
                                             x = CHM_HARV,
                          y = plot_locations_sp_HARV,
                                        buffer = 20,
                                        fun = mean,
                                        df = TRUE)

# view data
mean_tree_height_plots_HARV

# plot the result
# plot data
ggplot(data = mean_tree_height_plots_HARV, aes(ID, HARV_chmCrop, fill=factor(ID))) +
  geom_bar(stat = "identity") +
  ggtitle("Mean Tree Height at each Plot") +
  xlab("Plot ID") +
  ylab("Tree Height (m)")
```

Post workshop survey: https://carpentries.typeform.com/to/UgVdRQ?slug=2022-11-10-jornada-geo-online