# Assignment 9: Spatial Analysis in R

Nancy Bao

## OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics (ENV872L) on spatial analysis.

### Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Use the lesson as a guide. It contains code that can be modified to complete the assignment.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document. Space for your answers is provided in this document and is indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio.
5. When you have completed the assignment, **Knit** the text and code into a single HTML file.
6. After Knitting, please submit the completed exercise (PDF file) in Sakai. Please add your last name into the file name (e.g., "Fay_A10_SpatialAnalysis.pdf") prior to submission.

## DATA WRANGLING

### Set up your session

1. Check your working directory
2. Import libraries: tidyverse, sf, leaflet, and mapview

```
#1. Check working directory
getwd()
```

```
## [1] "/Users/Nancy/Desktop/Semester 4/ENV 872L/Environmental_Data_Analytics_2021"
```

```
#2. Import libraries
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------------------------------

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.1.4, PROJ 6.3.1
```

```
library(leaflet)
library(mapview)
```

```
## GDAL version >= 3.1.0 | setting mapviewOptions(fgb = TRUE)
```

### Read (and filter) county features into an sf dataframe and plot

In this exercise, we will be exploring stream gage height data in Nebraska corresponding to floods occurring there in 2019. First, we will import from the US Counties shapefile we've used in lab lessons, filtering it this time for just Nebraska counties. Nebraska's state FIPS code is `31` (as North Carolina's was `37`).

3. Read the `cb_2018_us_county_20m.shp` shapefile into an sf dataframe, filtering records for Nebraska counties (State FIPS = 31)
4. Reveal the dataset's coordinate reference system
5. Plot the records as a map (using `mapview` or `ggplot`)

```r
#3. Read in Counties shapefile into an sf dataframe, filtering for just NE counties
NE_counties_sf <- st_read("./Data/Spatial/cb_2018_us_county_20m.shp") %>%
                  filter(STATEFP == 31)
```

```
## Reading layer `cb_2018_us_county_20m' from data source `/Users/Nancy/Desktop/Semester 4/ENV 872L/Envi
## Simple feature collection with 3220 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
## Geodetic CRS:  NAD83
```

```r
#4. Reveal the CRS of the counties features
st_crs(NE_counties_sf)
```

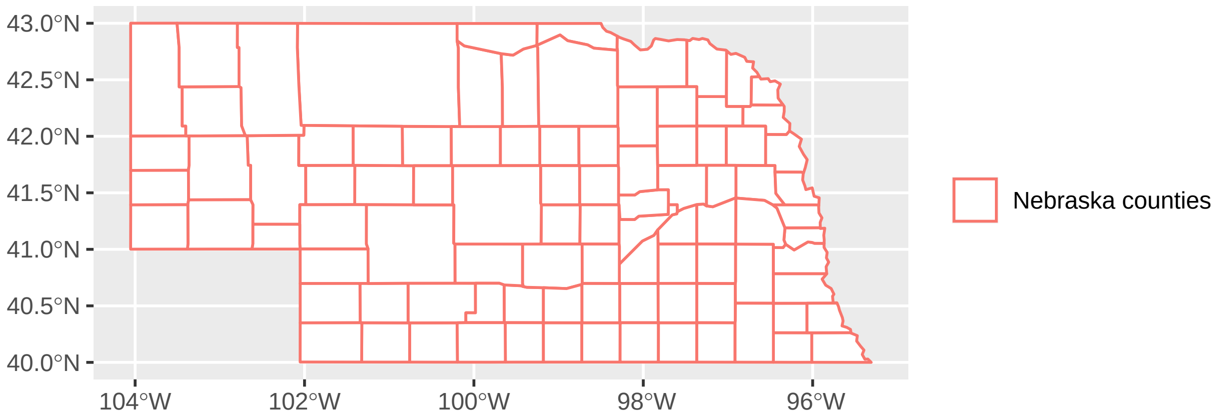
```
## Coordinate Reference System:
##   User input: NAD83
##   wkt:
## GEOGCRS["NAD83",
##     DATUM["North American Datum 1983",
##         ELLIPSOID["GRS 1980",6378137,298.257222101,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["latitude",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["longitude",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     ID["EPSG",4269]]
```

```r
st_crs(NE_counties_sf)$epsg
```

```
## [1] 4269
```

```r
#The CRS is NAD83 (North American Datum 1983); epsg code: 4269

#5. Plot the data
NEcounties_plot<-ggplot()+
                geom_sf(data=NE_counties_sf, aes(color="NAME"),fill="white")+
                scale_color_discrete(name=NULL,labels=c("Nebraska counties"))
print(NEcounties_plot)
```




```
#I used scale_color_discrete to rename the legend to "Nebraska counties".
```

6. What is the EPSG code of the Counties dataset? Is this a geographic or a projected coordinate reference system? (Or, does this CRS use angular or planar coordinate units?) To what datum is this CRS associated? (Tip: look the EPSG code on https://spatialreference.org)

   ANSWER: The EPSG code of the Counties dataset is 4269. This is a geographic coordinate reference system. This CRS uses angular coordinate units (longitude/latitude). The CRS is associated with the North American Datum 1983.

### Read in gage locations csv as a dataframe, then display the column names it contains

Next we'll read in some USGS/NWIS gage location data added to the `Data/Raw` folder. These are in the `NWIS_SiteInfo_NE_RAW.csv` file.(See `NWIS_SiteInfo_NE_RAW.README.txt` for more info on this dataset.)

7. Read the NWIS_SiteInfo_NE_RAW.csv file into a standard dataframe.
8. Display the column names of this dataset.

```
#7. Read in gage locations csv as a dataframe
usgs_gage_locations <-read.csv("./Data/Raw/NWIS_SiteInfo_NE_RAW.csv")

#8. Reveal the names of the columns
colnames(usgs_gage_locations)
```

```
## [1] "site_no"             "station_nm"          "site_tp_cd"
## [4] "dec_lat_va"          "dec_long_va"         "coord_acy_cd"
## [7] "dec_coord_datum_cd"
```

9. What columns in the dataset contain the x and y coordinate values, respectively?
   > ANSWER: The x coordinate values are in the dec_long_va column. The y coordinate values are in the dec_lat_va column. >

### Convert the dataframe to a spatial features ("sf") dataframe

10. Convert the dataframe to an sf dataframe.

- Note: These data use the same coordinate reference system as the counties dataset

11. Display the column names of the resulting sf dataframe

```
#10. Convert to an sf object
usgs_gage_sf<- st_as_sf(usgs_gage_locations,
                        coords = c('dec_long_va','dec_lat_va'),
                        crs=4269)
```

```r
#11. Re-examine the column names
colnames(usgs_gage_sf)
```

```
## [1] "site_no"            "station_nm"         "site_tp_cd"
## [4] "coord_acy_cd"       "dec_coord_datum_cd" "geometry"
```

12. What new field(s) appear in the sf dataframe created? What field(s), if any, disappeared?

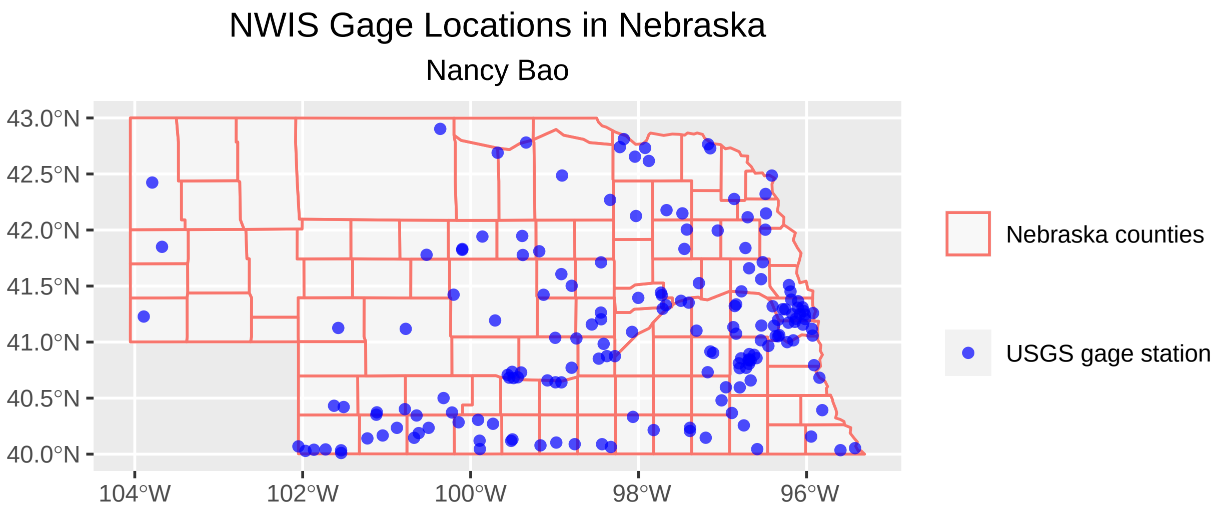
    ANSWER: The new field that appeared in the sf dataframe was the "geometry" column. The fields that disappeared included "dec_lat_va" and "dec_long_va".

### Plot the gage locations on top of the counties

13. Use `ggplot` to plot the county and gage location datasets.

- Be sure the datasets are displayed in different colors
- Title your plot "NWIS Gage Locations in Nebraska"
- Subtitle your plot with your name

```r
#13. Plot the gage locations atop the county features
NWIS_Gage_Location_NE_plot<- ggplot()+
                              geom_sf(data= NE_counties_sf,aes(color="NAME"),
                                      alpha=0.5, fill="white") +
                              scale_color_discrete(name = NULL,
                                    labels = c("Nebraska counties"))+
                              geom_sf(data= usgs_gage_sf,color="blue",
                                     aes(fill="station_nm"), alpha=0.7)+
                              scale_fill_discrete(name = NULL, labels = c("USGS gage station"))+
                              labs(title="NWIS Gage Locations in Nebraska",
                                   subtitle="Nancy Bao",
                              caption="Blue dots represent USGS gage stations. County borders are outli
                              theme(plot.title = element_text(hjust = 0.5),
                              plot.subtitle = element_text(hjust = 0.5),
                              plot.caption=element_text(hjust=0.25))
print(NWIS_Gage_Location_NE_plot)
```




Blue dots represent USGS gage stations. County borders are outlined in gray.

```r
# I manually picked orange for the county layer and blue for the gage locations.
#I used labs(title="",subtitle="",caption="") to label the map.
```

```r
# I renamed legend titles with scale_color_discrete() and scale_fill_discrete()
#I added a caption to describe the colors and markers.
```

### Read in the gage height data and join the site location data to it.

Lastly, we want to attach some gage height data to our site locations. I've constructed a csv file listing many of the Nebraska gage sites, by station name and site number along with stream gage heights (in meters) recorded during the recent flood event. This file is titled `NWIS_SiteFlowData_NE_RAW.csv` and is found in the Data/Raw folder.

14. Read the `NWIS_SiteFlowData_NE_RAW.csv` dataset in as a dataframe.
15. Show the column names .
16. Join our site information (already imported above) to these gage height data.

- The `site_no` and `station_nm` can both/either serve as joining attributes.
- Construct this join so that the result only includes spatial features where both tables have data.

17. Show the column names in this resulting spatial features object
18. Show the dimensions of the resulting joined dataframe

```r
#14. Read the site flow data into a data frame
nwis_site_flow <-read.csv("./Data/Raw/NWIS_SiteFlowData_NE_RAW.csv")

#15. Show the column names
colnames(nwis_site_flow)
```

```
## [1] "site_no"    "station_nm" "date"       "gage_ht"
```

```r
#16. Join location data to it
site_flow_location_join<-usgs_gage_sf %>%
              inner_join(nwis_site_flow, by=c('station_nm','site_no'))
#I did an inner_join, so results only had spatial features where both tables have data

#17. Show the column names of the joined dataset
colnames(site_flow_location_join)
```

```
## [1] "site_no"            "station_nm"         "site_tp_cd"
## [4] "coord_acy_cd"       "dec_coord_datum_cd" "date"
## [7] "gage_ht"            "geometry"
```

```r
#18. Show the dimensions of this joined dataset
dim(site_flow_location_join)
```

```
## [1] 136   8
```

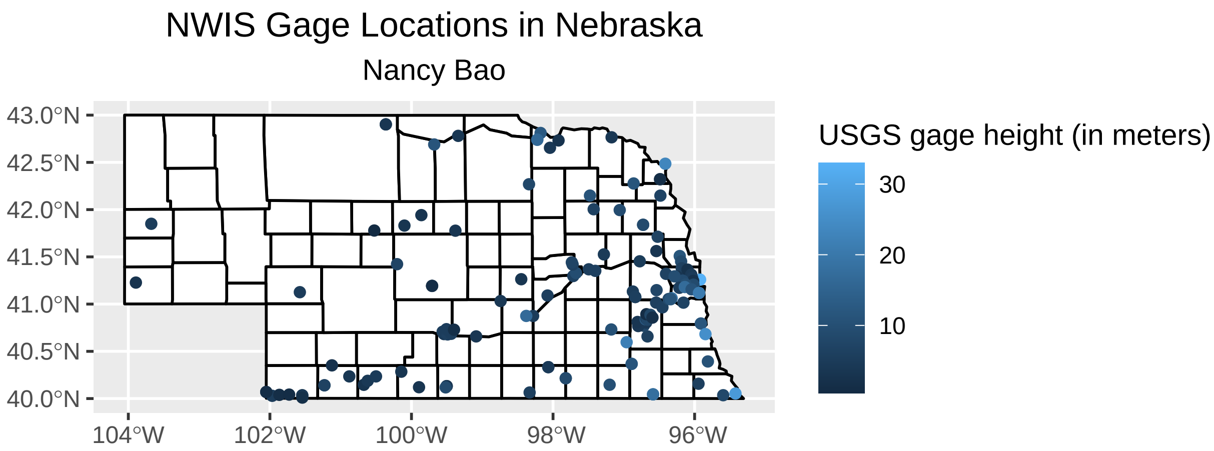
### Map the pattern of gage height data

Now we can examine where the flooding appears most acute by visualizing gage heights spatially. 19. Plot the gage sites on top of counties (using `mapview`, `ggplot`, or `leaflet`) * Show the magnitude of gage height by color, shape, other visualization technique.

```r
#Map the points, sized by gage height
gage_heights_map<-ggplot()+
                    geom_sf(data=NE_counties_sf,color="black",
                          fill="white")+
                    geom_sf(data=site_flow_location_join,
                                aes(color= gage_ht),size=1.5,alpha=1)+
                    scale_color_continuous(name="USGS gage height (in meters)")+
```

```
                    labs(title="NWIS Gage Locations in Nebraska",
                               subtitle="Nancy Bao")+
                    theme(plot.title = element_text(hjust = 0.5),
                          plot.subtitle = element_text(hjust = 0.5))

print(gage_heights_map)
```




```
#I distinguished gage height by a blue gradient:
#with the lowest heights the darkest and the highest heights the lightest blues.
#I used hjust=0.5 to center the title
#I renamed the legend with the scale_color_continuous()
```

---

## SPATIAL ANALYSIS

Up next we will do some spatial analysis with our data. To prepare for this, we should transform our data into a projected coordinate system. We'll choose UTM Zone 14N (EPGS = 32614).

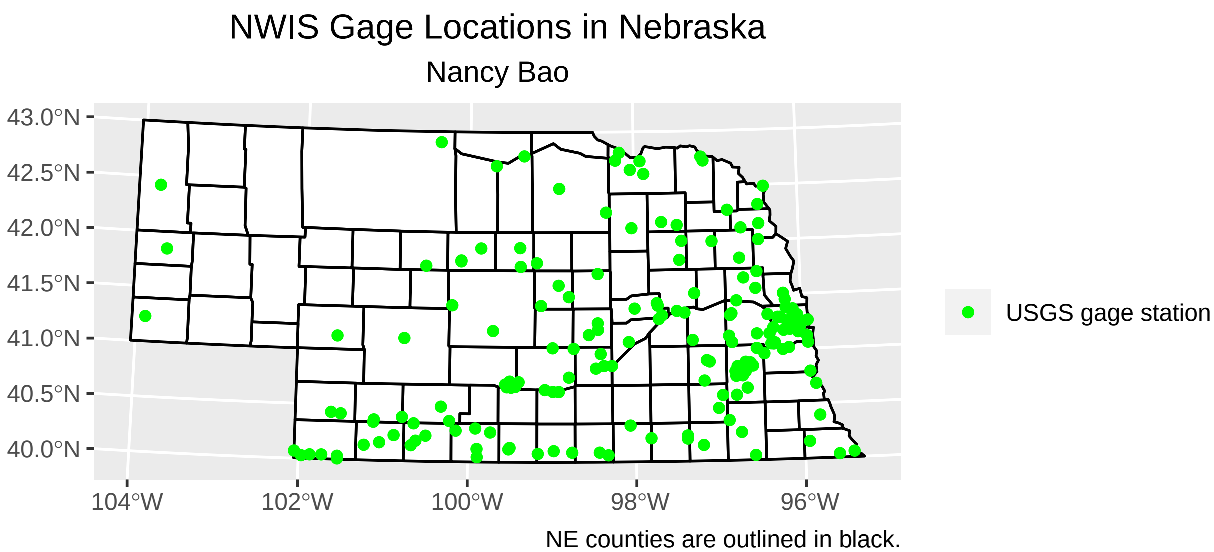
### Transform the counties and gage site datasets to UTM Zone 14N

20. Transform the counties and gage sf datasets to UTM Zone 14N (EPGS = 32614).
21. Using `mapview` or `ggplot`, plot the data so that each can be seen as different colors

```
#20 Transform the counties and gage location datasets to UTM Zone 14
NE_counties_sf_utm <-st_transform(NE_counties_sf, crs =32614)
nwis_gage_location_sf_utm <-st_transform(usgs_gage_sf , crs = 32614)

#21 Plot the data
#I used ggplot to plot the transformed datasets and used scale_fill_discrete to rename legend
gage_county_map<-ggplot()+
                  geom_sf(data=NE_counties_sf_utm,
                         color="black",fill="white")+
                 geom_sf(data=nwis_gage_location_sf_utm,
                         color="green",size=1.5,aes(fill="USGS gage station"))+
                 scale_fill_discrete(name="")+
                 labs(title="NWIS Gage Locations in Nebraska",
                            subtitle="Nancy Bao",
                      caption="NE counties are outlined in black.")+
                theme(plot.title = element_text(hjust = 0.5),
```

```
                              plot.subtitle = element_text(hjust = 0.5))
print(gage_county_map)
```




**Select the gages falling within a given county**

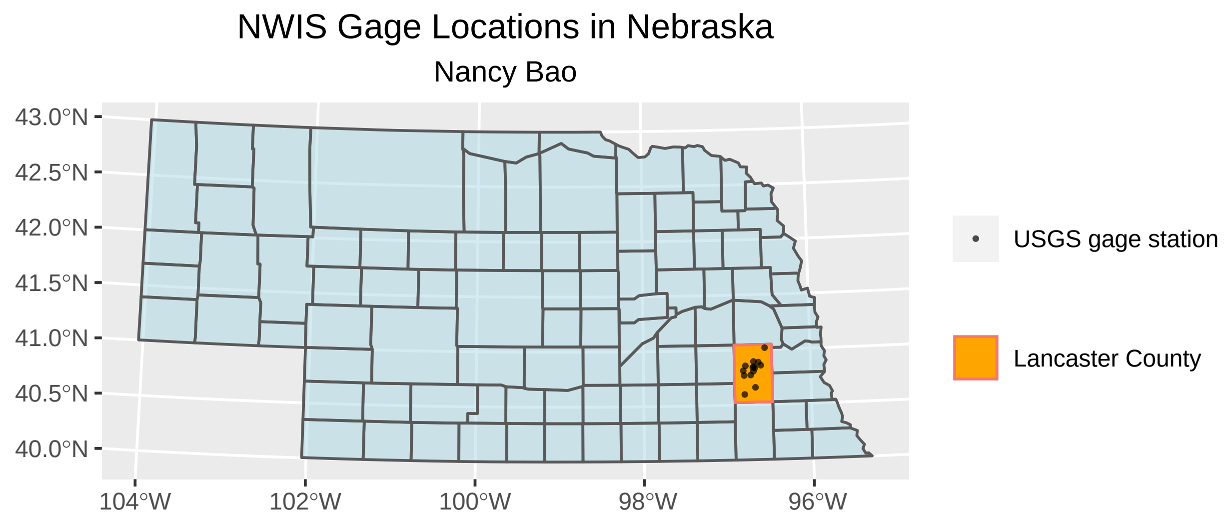
Now let's zoom into a particular county and examine the gages located there. 22. Select Lancaster county from your county sf dataframe 23. Select the gage sites falling `within` that county * Use either matrix subsetting or tidy filtering 24. Create a plot showing: * all Nebraska counties, * the selected county, * and the gage sites in that county

```
#22 Select the county
#I used the UTM transformed datasets
Lancaster_co_sf_utm<- NE_counties_sf_utm %>%
                    filter(NAME=="Lancaster")

#23 Select gages within the selected county
selected_gages_sf_utm<- nwis_gage_location_sf_utm[Lancaster_co_sf_utm,]
#I used the matrix subsetting to get the gage locations within Lancaster Co.

#24 Plot
#I used the UTM transformed datasets to plot
Lancaster_plot <- ggplot()+
                  geom_sf(data=NE_counties_sf_utm,
                          fill="light blue",alpha=0.5)+
                  geom_sf(data=Lancaster_co_sf_utm,
                          fill="orange",aes(color="NAME"))+
                  scale_color_discrete(name = NULL,
                                       labels = c("Lancaster County"))+
                  geom_sf(data=selected_gages_sf_utm,
                          aes(fill='station_nm'),size=0.5, alpha=0.7)+
                  scale_fill_discrete(name = NULL, labels = c("USGS gage station"))+
                 labs(title="NWIS Gage Locations in Nebraska",
                                  subtitle="Nancy Bao",
                      caption="All county borders in Nebraska are outlined in gray and filled in light 
                              theme(plot.title = element_text(hjust = 0.5),
                              plot.subtitle = element_text(hjust = 0.5),
```

```
                                    plot.caption=element_text(hjust=0.25))
print(Lancaster_plot)
```




nty borders in Nebraska are outlined in gray and filled in light blue except Lancaster County, which is outlined in pink ar

```
#I layered each sf dataframe to show all the NE counties,
# distinguish Lancaster Co., and show the gages within Lancaster Co.

#I used scale_color_discrete and scale_fill_discrete to rename the legends
#to show Lancaster Co. and the gage stations in that county.

#I adjusted the labels so that they would be centered and
#I added a caption to the bottom of the graph to explain what the blue blocks mean.
```