Assignment: Spatial Analysis in R

Laurie Muzzy

Contents

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 (HTML file) to the dropbox in Sakai. Please add your last name into the file name (e.g., "Fay_A09_SpatialAnalysis.pdf") prior to submission.

DATA WRANGLING

1. Prepare the workspace

- Import: tidyverse, sf, and leaflet

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2

## -- Attaching packages --------- tidyverse 1.2.1 --

## v ggplot2 3.1.0       v purrr   0.3.0
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.2       v stringr 1.3.1
## v readr   1.3.1       v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'readr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4
## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'stringr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.3

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

```
library(tidyr)
library(leaflet)
```

```
## Warning: package 'leaflet' was built under R version 3.4.4
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 3.4.4

## Linking to GEOS 3.6.1, GDAL 2.1.3, PROJ 4.9.3
```

```
library(mapview)
```

```
## Warning: package 'mapview' was built under R version 3.4.4
```

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

```
## Warning: package 'raster' was built under R version 3.4.4

## Loading required package: sp

## Warning: package 'sp' was built under R version 3.4.4

##
## Attaching package: 'raster'

## The following object is masked from 'package:dplyr':
##
##     select

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
getwd()
```

```
## [1] "/Users/laurie/Desktop/Envtl_Data_Analytics/MuzzyGitFile"
```

2. Read filtered county features into an sf dataframe and plot

In this exercise, we will be exploring stream gage height data in Nebraska, as there's been recent floods there. 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).

- Read the cb_2017_us_county_20m.shp shapefile into an sf dataframe
- Filter for Nebraska counties (State FIPS = 31)
- Show the dataset's coordinate reference system
- Plot the records as a map (in any format)

```
#Read in Counties shapefile into an sf dataframe, filtering for just NC counties
NEcounties <- st_read("./Data/Spatial/cb_2017_us_county_20m.shp") %>%
  filter(STATEFP == 31)
```

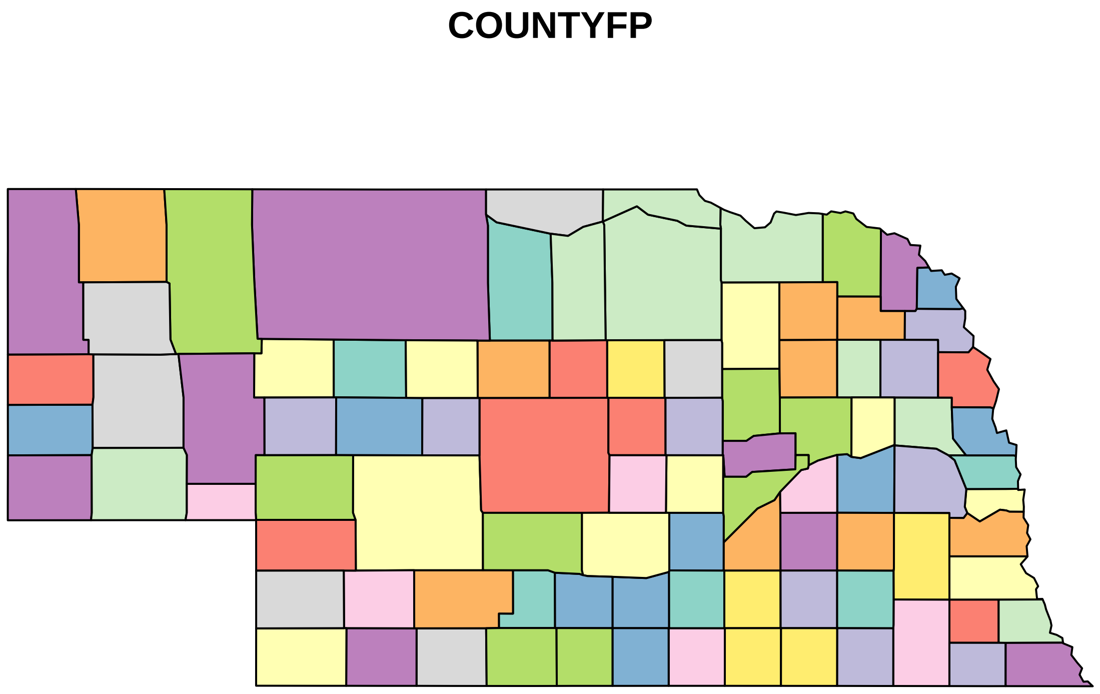
```
## Reading layer `cb_2017_us_county_20m' from data source `/Users/laurie/Desktop/Envtl_Data_Analytics/Mu
## Simple feature collection with 3220 features and 9 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
## epsg (SRID):    4269
## proj4string:    +proj=longlat +datum=NAD83 +no_defs
```

```
#Reveal the CRS of the counties features
st_crs(NEcounties)
```

```
## Coordinate Reference System:
##   EPSG: 4269
##   proj4string: "+proj=longlat +datum=NAD83 +no_defs"
```

```
#Plot the data

plot(NEcounties["COUNTYFP"])
```



```
# could also use mapView(NEcounties)
```

QUESTION: What is the EPSG code of the Counties dataset? Using http://spatialreference.org, is this a geographic or a projected coordinate system? (Or, does this CRS use angular or planar coordinate units?) To what datum is this CRS associated?
ANSWER: 4269 is the GCS, the datum is NAD83.

3. 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 I've 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 datset.) * Read the NWIS_SiteInfo_NE_RAW.csv file into a standard dataframe * Display the column names of this dataset

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

#Reveal the names of the columns
colnames(NEgage)
```

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

> QUESTION: What columns in the dataset contain the x and y coordinate values, respectively?
> ANSWER: x = dec_long_va and y = dec_lat_va

4. Convert the gage locations dataframe to an sf dataframe of points

- These data use the same coordnate reference system as the counties dataset
- Display the column names of the resulting sf dataframe

```r
#Convert to an sf object
NEgagepoints <- st_as_sf(NEgage, coords = c("dec_long_va","dec_lat_va"),
                        crs = 4269)
#Reveal the structure
colnames(NEgagepoints)
```

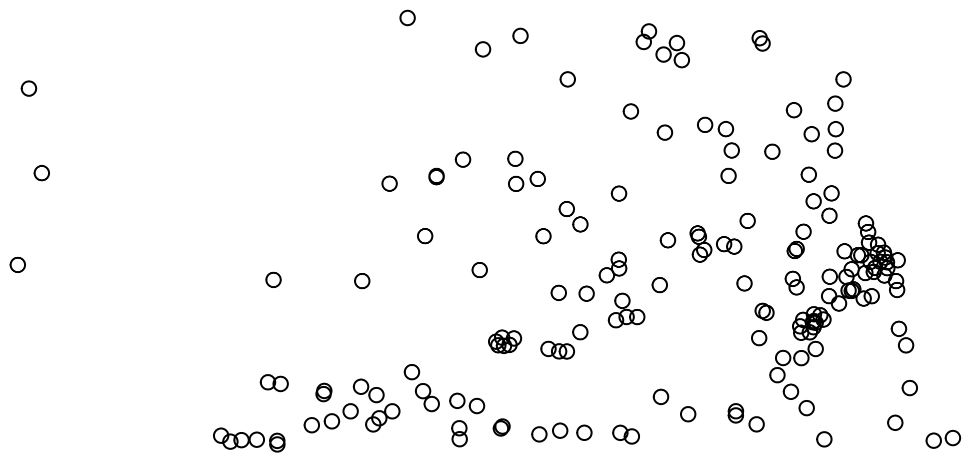
```
## [1] "site_no"            "station_nm"         "site_tp_cd"
## [4] "coord_acy_cd"       "dec_coord_datum_cd" "geometry"
```

> QUESTION: What new field(s) appear in the sf dataframe created? What field(s), if any, disappeared? ANSWER: It added a geometry column, with x and y coordinates; the latitude and longitude columns disappeared/ were transformed into geometry

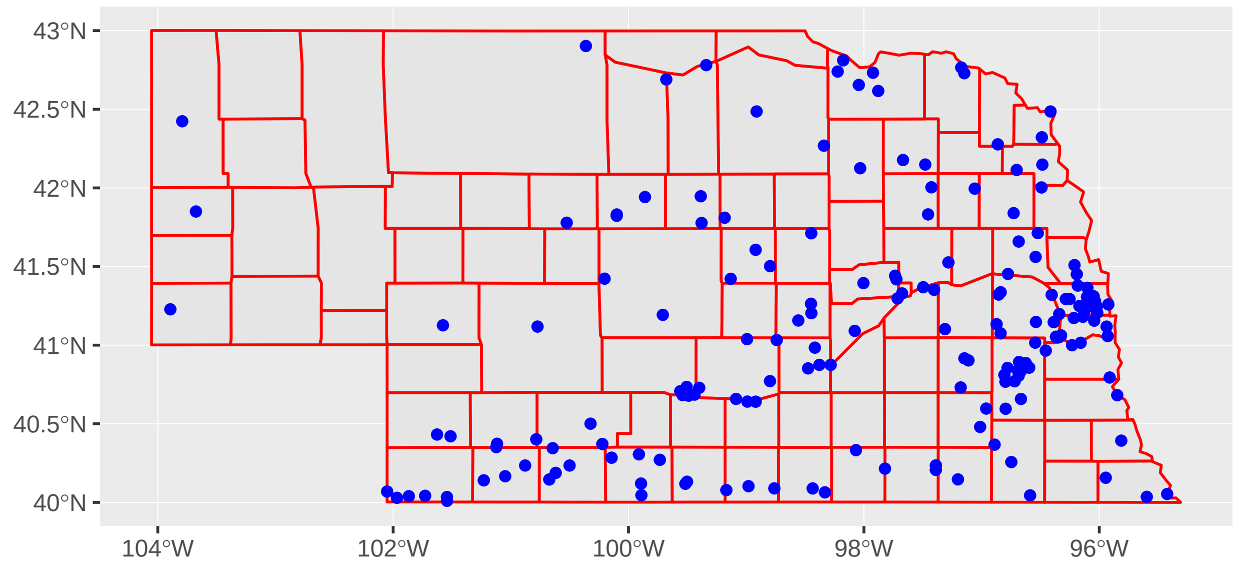
5. Use `ggplot` to plot the gage locations on top of the counties

- Plot the different datasets in different colors

```r
NEgagemap <-
  ggplot() +
  geom_sf(data = NEcounties, col = "red") +
  geom_sf(data = NEgagepoints, col = "blue") +
  plot(NEgagepoints$geometry)
```



```r
plot(NEgagemap)
```



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

And finally, 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.

- Read this dataset in as a dataframe.
- Join our site information (already imported above) to these gage height data.
- The `site_no` and `station_nm` can both serve as joining attributes.
- Construct this join so that the result only includes records where both tables have data.
- Show the column names in this resulting dataframe
- Once joined, we will again have to convert this product (a dataframe) into a spatial dataframe. Do that.

```r
#Read in the data
NWISgage_ht <- read.csv("./Data/Raw/NWIS_SiteFlowData_NE_RAW.csv")

#Show the column names
colnames(NWISgage_ht)
```

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

```r
#colnames(NWISflow)
#class(NWISflow$site_no)
#class(NEgage$site_no)

#Join location data to it
NWISjoin <-  NWISgage_ht %>%
  left_join(y = NEgagepoints, by = c('site_no', 'station_nm')) %>%
  na.omit() #if there's na in a row, it'll omit that (!is.na will work on col)
```

```
## Warning: Column `station_nm` joining factors with different levels,
## coercing to character vector
```

```r
colnames(NWISjoin)
```

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

```r
#[1] "site_no"            "station_nm"         "date"               "gage_ht"
#[5] "site_tp_cd"         "coord_acy_cd"       "dec_coord_datum_cd" "geometry"

#Convert back to sf dataframe

NWISjoin_sf <- st_as_sf(NWISjoin, crs = 4269)
class(NWISjoin_sf)
```

```
## [1] "sf"         "data.frame"
```

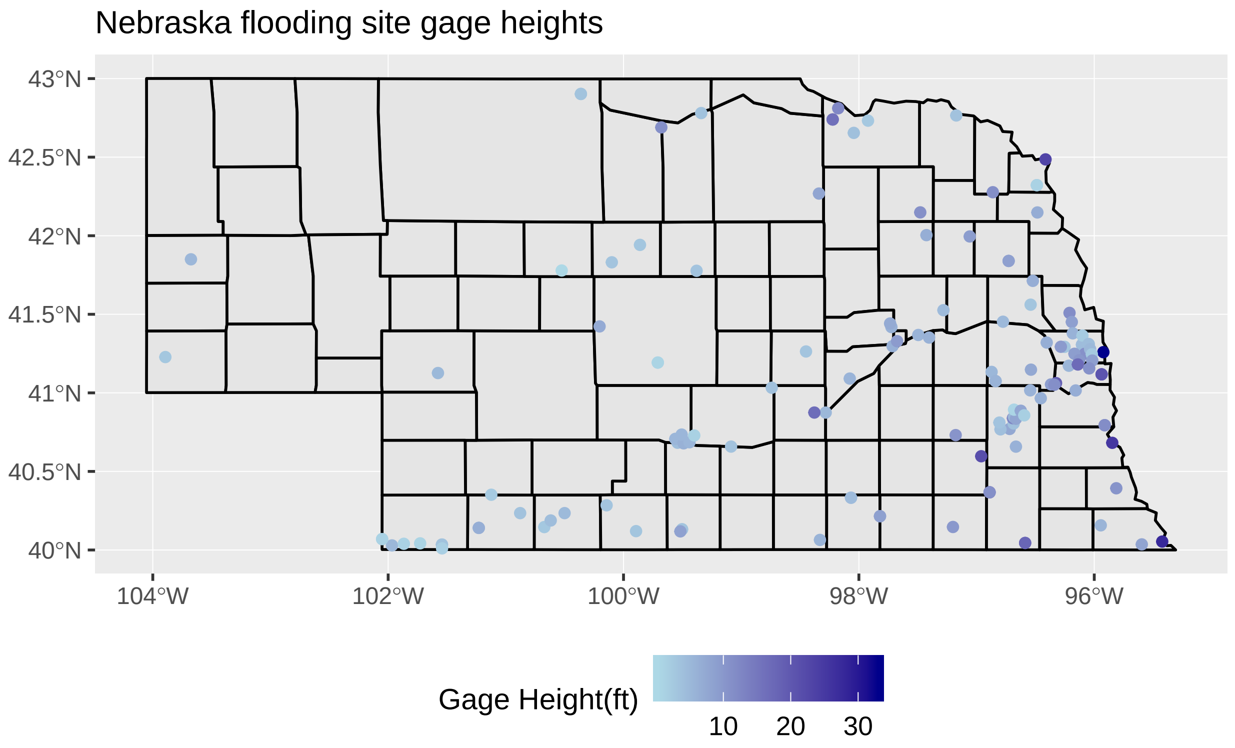
7. Map the pattern of gage height data

Now we can examine where the flooding appears most acute by visualizing gage heights spatially. * Plot the gage sites on top of counties * Show the magnitude of gage height by color, shape, other visualization technique.

```r
#Plot the values
legend_title_1 <- "Gage Height(ft)"

NWISviz <- ggplot() +
  geom_sf(data = NEcounties, color = "black") +
  geom_sf(data = NWISjoin_sf, aes(color = gage_ht)) +
  scale_color_gradient(legend_title_1, low = "lightblue", high = "darkblue") +
  ggtitle("Nebraska flooding site gage heights") +
  theme(plot.title = element_text(size = 12, color = "black"),
        legend.position = "bottom",
        legend.text = element_text(size = 10, color = "black"))

plot(NWISviz)
```



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).

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

- Transform each dataset to crs 32614
- Using ggplot, plot the data so that each can be seen as different colors

```
#Transform the counties and gage location datasets to UTM Zone 14

NEcounties_utm <- st_transform(NEcounties, crs = 32614)
st_crs(NEcounties_utm)
```

```
## Coordinate Reference System:
##   EPSG: 32614
##   proj4string: "+proj=utm +zone=14 +datum=WGS84 +units=m +no_defs"
```

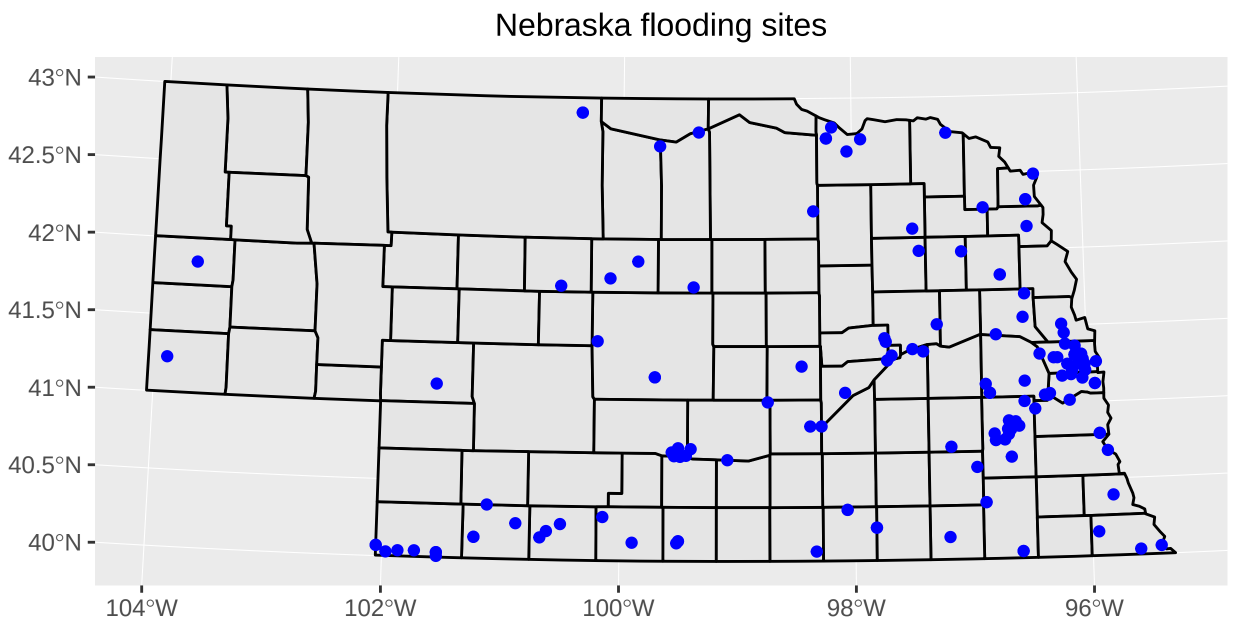
```
class(NEcounties_utm)
```

```
## [1] "sf"         "data.frame"
```

```
NWISjoin_utm <- st_transform(NWISjoin_sf, c = 32614)
st_crs(NWISjoin_utm)
```

```
## Coordinate Reference System:
##   EPSG: 32614
##   proj4string: "+proj=utm +zone=14 +datum=WGS84 +units=m +no_defs"
```

```r
#Plot the data
ggplot() +
geom_sf(data = NEcounties_utm, col = "black") +
  geom_sf(data = NWISjoin_utm, col = "blue") +
  ggtitle("Nebraska flooding sites") +
  theme(plot.title = element_text(size = 12, color = "black", hjust = .5))
```



> QUESTION: The shape of Nebraska should look a bit different than the one created in Step 5? Why? > ANSWER: The projection is applied to this map, so it's a bit cured, like the earth.

9. Select the gages falling within a given county

Now let's zoom into a particular county and examine the gages located there.

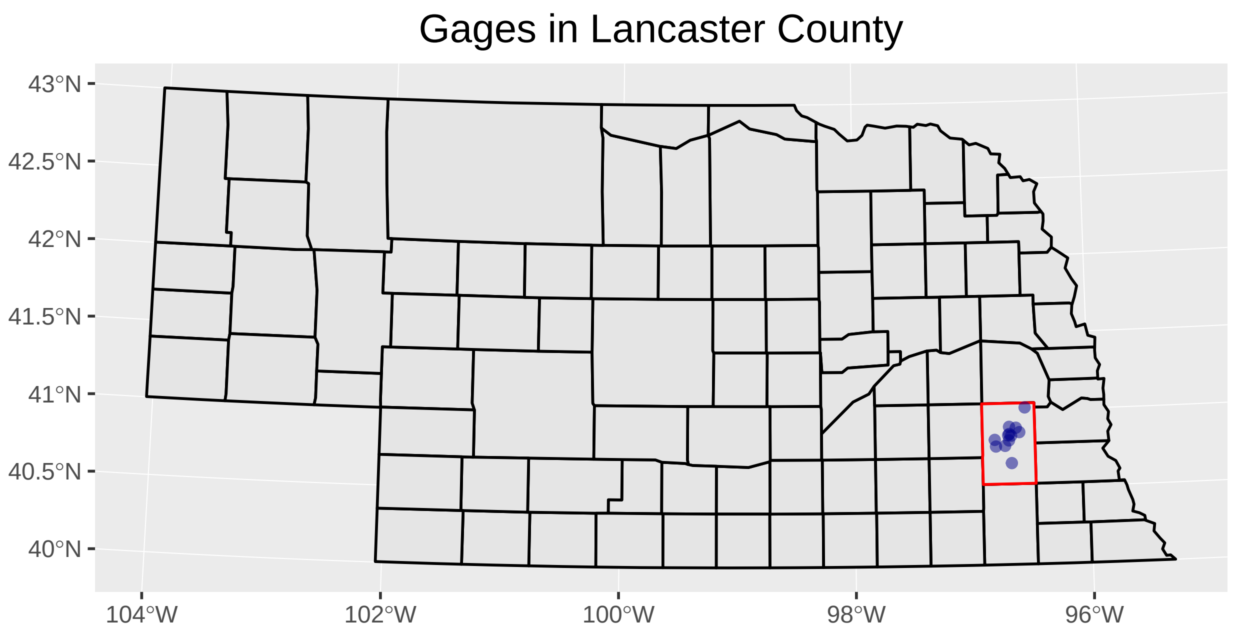
- Select Lancaster county from your county sf dataframe
- Select the gage sites falling `within` that county
- Remember you'll have to create a mask and then apply that mask
- Create a plot showing:
- all Nebraska counties,
- the selected county,
- and the gage sites in that county

```r
#Select the county Lancaster
Lancaster <- NEcounties_utm %>%
  filter(NAME == "Lancaster")

#Select gages within
gages_Lancaster_int <- st_intersects(Lancaster, NWISjoin_utm, sparse = FALSE)

gages_Lancaster <- NWISjoin_utm[gages_Lancaster_int,]

#Plot
ggplot() +
  geom_sf(data = NEcounties_utm, col = "black") +
  geom_sf(data = Lancaster, col = "red") +
  geom_sf(data = gages_Lancaster, col = "darkblue", alpha = 0.5) +
  ggtitle("Gages in Lancaster County") +
  theme(plot.title = element_text(size = 15, color = "black", hjust = 0.5))
```



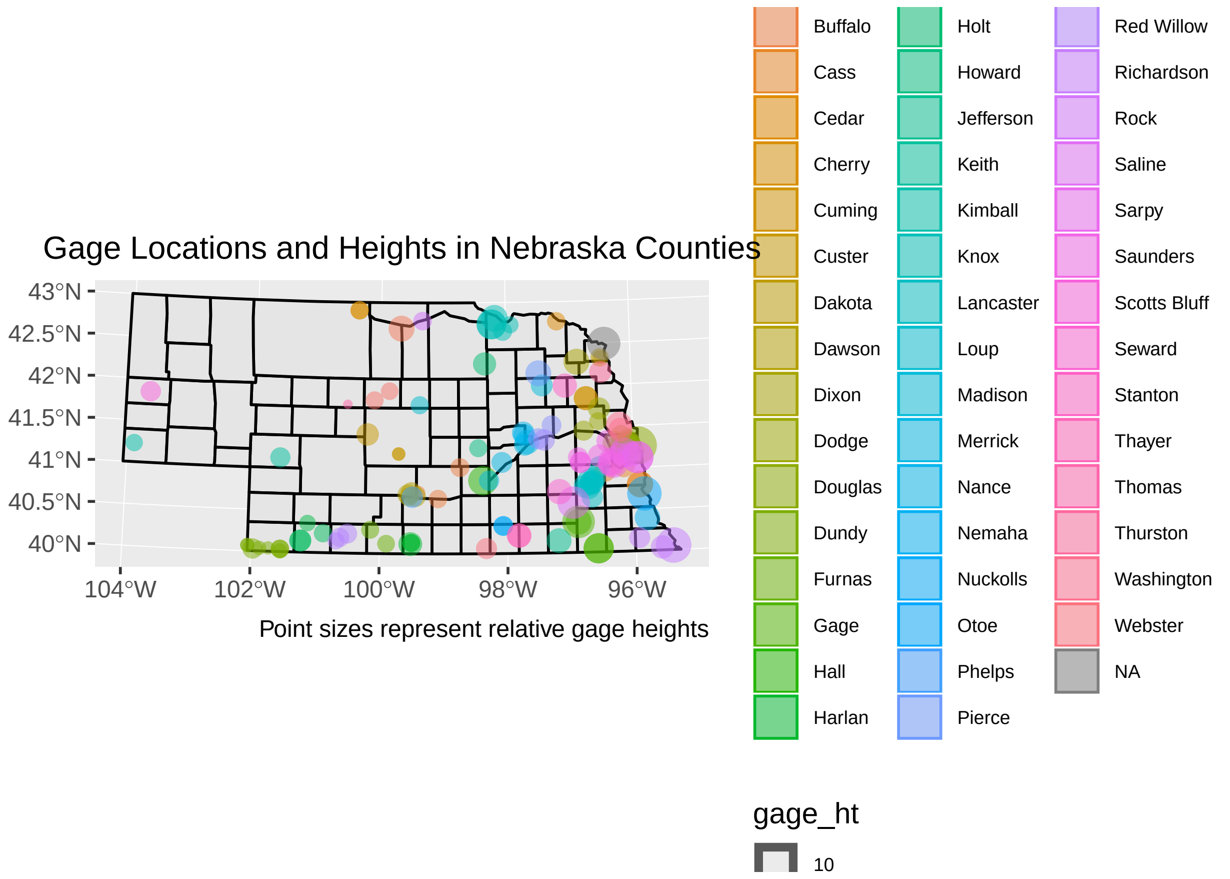
10. Tag each gage site with the name of the county in which it falls

A spatial join (`st_join`) allows us to assign the attributes of an overlapping feature onto a another feature. We will use to to assign each gage location the attributes of the county in which it is located. * Spatially join the county features to the gage height features * Display the list of fields in the resulting dataset * Map the gage locations, * Include county boundaries * Displaying each gage locations county "NAME" as a different color. * Display each gage size proportional to its "gage_ht" value

```r
#Join features
counties_gages_join <- st_join(NWISjoin_utm, NEcounties_utm)
#Show column names
colnames(counties_gages_join)
```

```
##  [1] "site_no"            "station_nm"         "date"
##  [4] "gage_ht"            "site_tp_cd"         "coord_acy_cd"
##  [7] "dec_coord_datum_cd" "STATEFP"            "COUNTYFP"
## [10] "COUNTYNS"           "AFFGEOID"           "GEOID"
## [13] "NAME"               "LSAD"               "ALAND"
## [16] "AWATER"             "geometry"
```

```r
#Plot
ggplot() +
  geom_sf(data = NEcounties_utm, col = "black") +
  geom_sf(data = counties_gages_join,
          aes(size = gage_ht,fill = NAME, color = NAME), alpha = 0.5) +
  theme(legend.position = "right",
        legend.text = element_text(size = 7),
        plot.title = element_text(size = 12, color = "black", hjust = .5)) +
  ggtitle("Gage Locations and Heights in Nebraska Counties") +
  labs(caption = "Point sizes represent relative gage heights")
```



11. Summarize data by county

Finally, we'll summarize our gage height data by county and then display each county by it's mean gage height. * Group the spatially joined gage location/county dataset on the county name * Compute mean gage height * Join (non-spatially) this result to our county sf dataframe * Prior to joining, you'll need to drop the geometry column from the gage locations * To do this, see the `st_drop_geometry` function * Plot the counties showing mean gage heights for each county * Not all counties will have data

```
#Group and summarize
grp_counties_gages_join <- counties_gages_join %>%
group_by(NAME) %>%
summarize(mean_gage_ht = mean(gage_ht))
```

```
## Warning: Factor `NAME` contains implicit NA, consider using
## `forcats::fct_explicit_na`

## Warning: Factor `NAME` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
#Convert result to a simple dataframe
samplegrp_counties_gages_join <- st_as_sf(grp_counties_gages_join)
class(samplegrp_counties_gages_join)
```

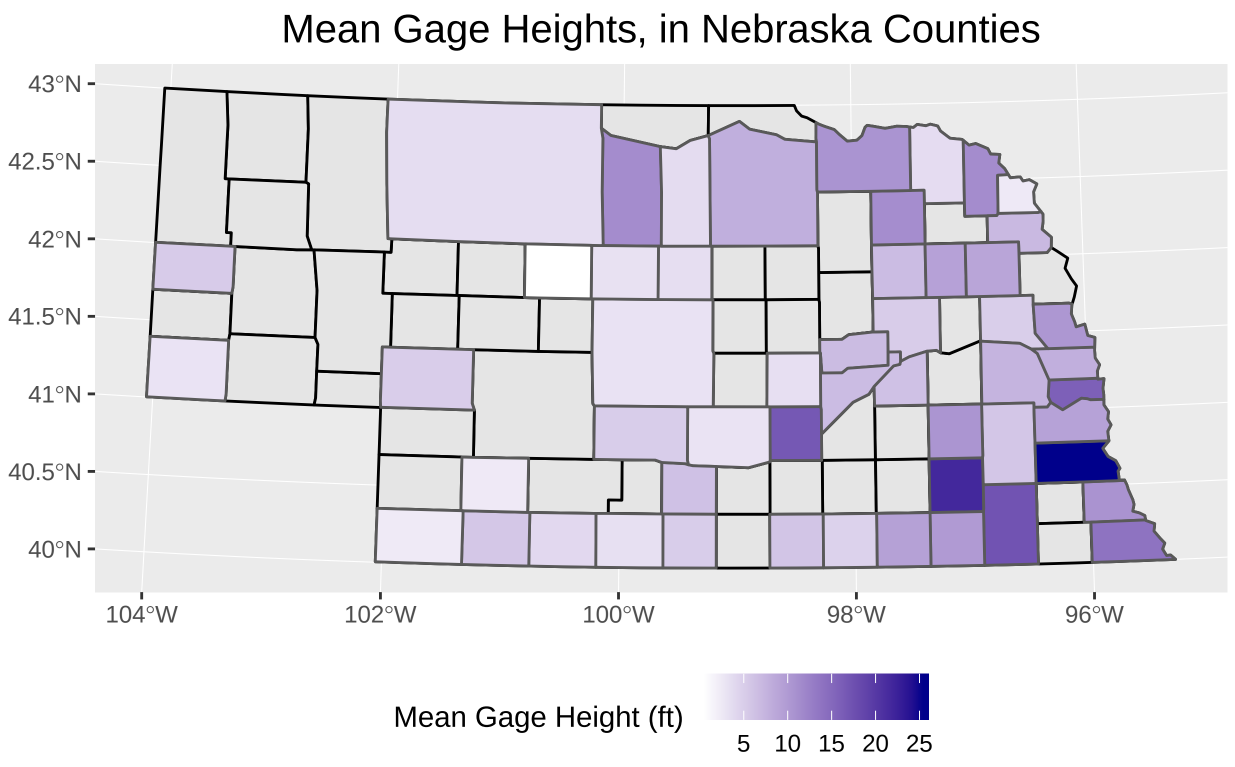
```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
#Join summary to County fc
drop_sample <- st_drop_geometry(samplegrp_counties_gages_join)
```

```r
countyavg <- NEcounties_utm %>%
inner_join(y = drop_sample, by = c("NAME" = "NAME"))

#Plot
legend_title_2 <- "Mean Gage Height (ft)"

ggplot() +
  geom_sf(data = NEcounties_utm, col = "black") +
  geom_sf(data = countyavg, aes (fill = mean_gage_ht)) +
  scale_fill_gradient(legend_title_2, low = "white", high = "darkblue") +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 9),
        plot.title = element_text(size = 15, color = "black", hjust = .5)) +
  ggtitle("Mean Gage Heights, in Nebraska Counties")
```


Mean Gage Heights, in Nebraska Counties
43°N
42.5°N
42°N
41.5°N
41°N
40.5°N
40°N
104°W
102°W
100°W
98°W
96°W
Mean Gage Height (ft)
5
10
15
20
25