Assignment: Spatial Analysis in R

Sebastian Bognar

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

```
# import tidyverse, sf, and leaflet
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------
## v ggplot2 3.1.0        v purrr   0.3.2
## v tibble  2.1.1        v dplyr   0.8.0.1
## v tidyr   0.8.3        v stringr 1.4.0
## v readr   1.3.1        v forcats 0.4.0
## -- Conflicts ------------------------------------------------------------ tidyve:
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sf)
```

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

```
library(leaflet)
library(mapview)
library(ggplot2)
```

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
counties<- st_read('./Data/Spatial/cb_2017_us_county_20m.shp') %>%
  filter(STATEFP == 31)
```

```
## Reading layer `cb_2017_us_county_20m' from data source `/Users/Seabass/Documents/Duke/spring_2019/env
## 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(counties)
```

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

```
#Plot the data
plot(counties['COUNTYNS'])
```

COUNTYNS
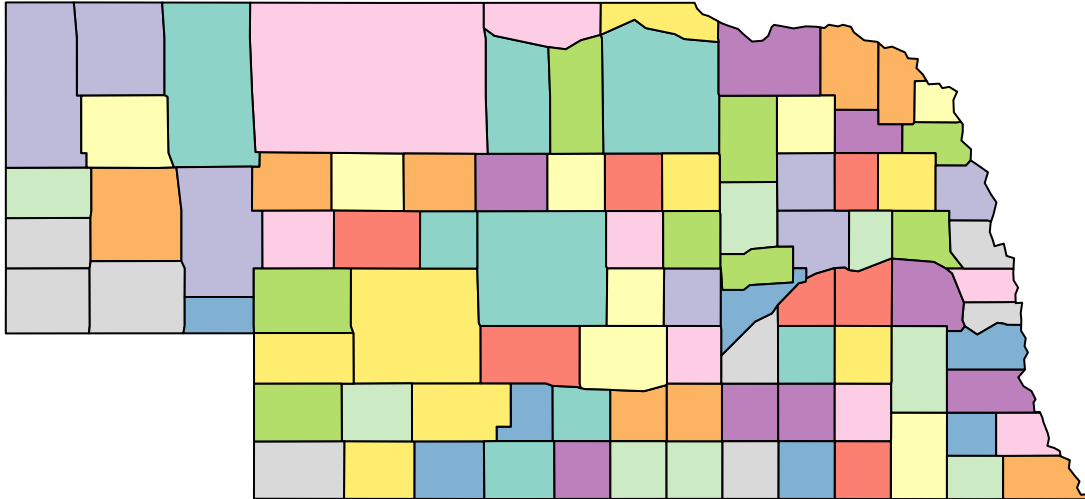
> QUESTION: What is the EPSG code of the Counties dataset? Using http://spatialreference.org, is this a geographic or a projected coordinate system? (Or, does this CRS use angular or planar coordinate units?) To what datum is this CRS associated?
> ANSWER: The EPSG code of the Counties dataset is 4269 and it is a geographic coordinate system. The CRS uses angular coordinate units. The data that the CRS is associated with 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

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

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

```
## [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: "dec_lat_va" (latitude) contain the y coordinate and the "dec_long_va" (longitude) contain the x coordinate.

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

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

```
#Convert to an sf object

NWIS_GAGE_SF <- st_as_sf(NWIS_RAW,coords = c('dec_long_va','dec_lat_va'),crs=4269)

#Reveal the structure

colnames(NWIS_GAGE_SF)
```

```
## [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: The new field that appears is "geometry". "dec_lat_va" and "dec_long_va" fields disappeared.

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

- Plot the different datasets in different colors

```
ggplot() +
  geom_sf(data = counties,col='black') +
  geom_sf(data = NWIS_GAGE_SF,col='red')+
  ggtitle("Gage locations in the Counties of Nebraska")
```

Gage locations in the Counties of Nebraska
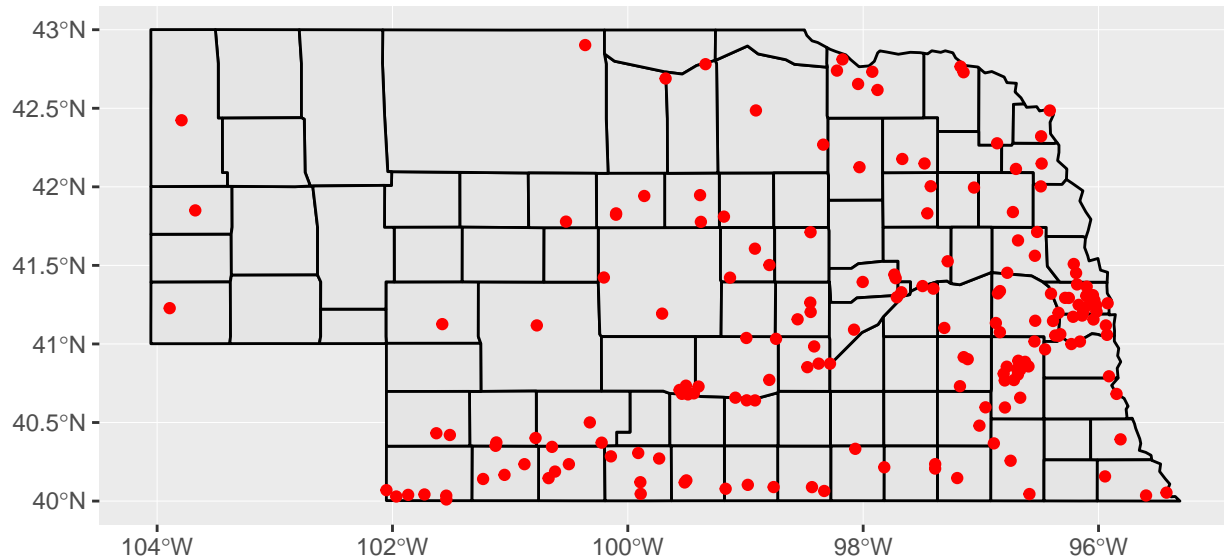
6. Read in the gage height data and join the site location data to it.

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

GAGE_HEIGHT_DATA_RAW <- read.csv("./Data/Raw/NWIS_SiteFlowData_NE_RAW.csv")

#Show the column names

colnames(GAGE_HEIGHT_DATA_RAW)
```

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

```r
#Join location data to it

GAGE_HEIGHT_SF_JOIN<-drop_na(left_join(NWIS_GAGE_SF,GAGE_HEIGHT_DATA_RAW, by = "site_no" ))
```

```
#colnames of join dataframe

colnames(GAGE_HEIGHT_SF_JOIN)
```

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

```
#Convert back to sf dataframe
GAGE_HEIGHT_SF <- st_as_sf(GAGE_HEIGHT_SF_JOIN, coords = "geometry", crs=4269 )
```

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.

```
#Plot the values
ggplot() +
  geom_sf(data=counties, fill ="grey")+
  geom_sf(data=GAGE_HEIGHT_SF, aes(color=gage_ht)) +
 scale_color_distiller("Gage Height (m)", palette = "Blues", direction = -1)+
  ggtitle("Variability of Gage Height at Gage Locations in Nebraska")
```
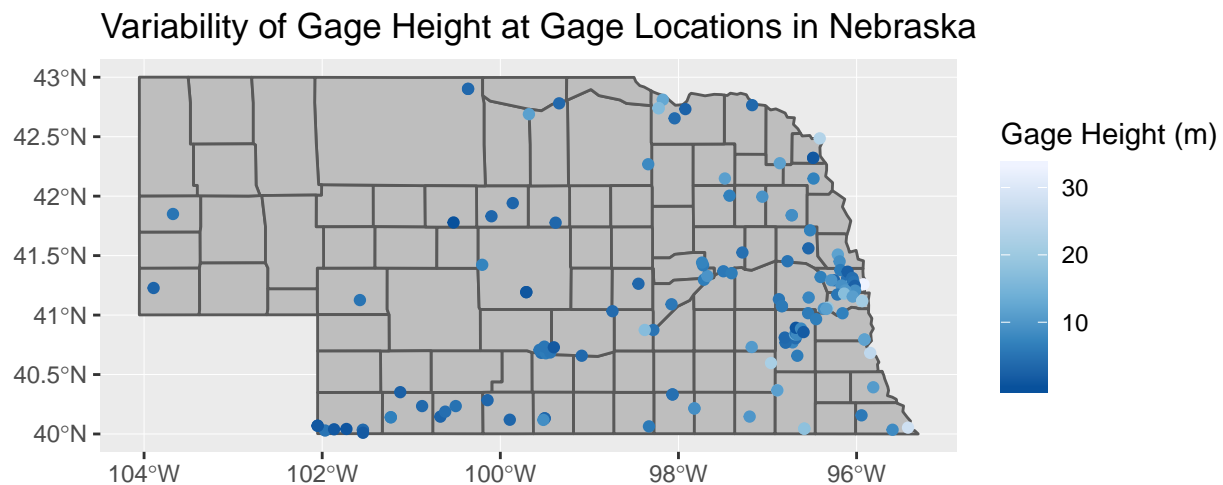
SPATIAL ANALYSIS

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
counties_sf_utm <- st_transform(counties, c=32614)
GAGE_HEIGHT_UTM <-  st_transform(GAGE_HEIGHT_SF, c=32614)
#Plot the data
ggplot() +
  geom_sf(data = counties_sf_utm,aes(fill=COUNTYNS)) +
  guides(fill=FALSE)+
   geom_sf(data = GAGE_HEIGHT_UTM,aes(color=gage_ht))+
   scale_color_distiller("Gage Height (m)", palette = "Blues", direction = -1)+
  ggtitle("Variability of Gage Height at Gage Locations in Nebraska")
```
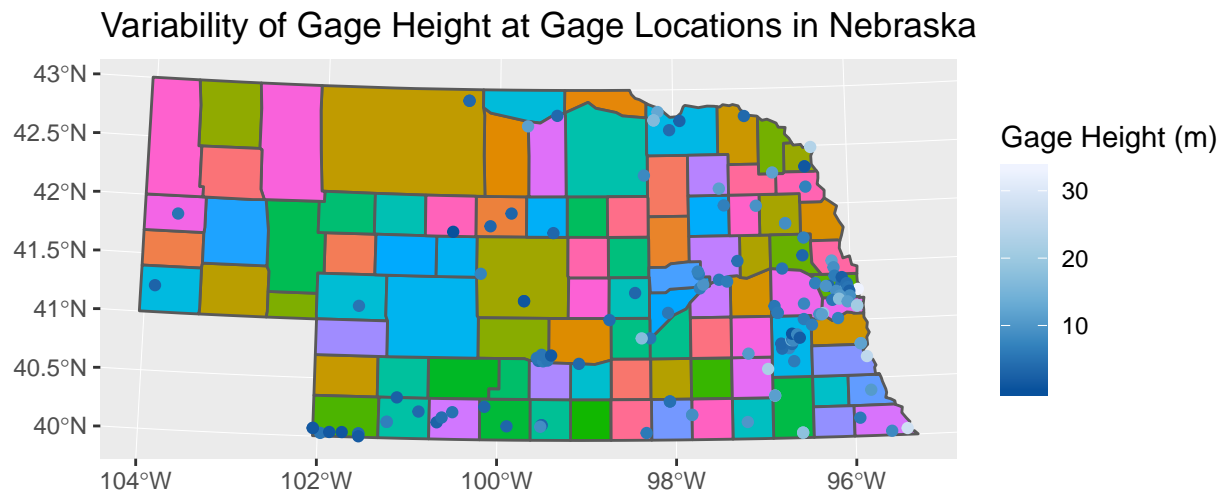

> QUESTION: The shape of Nebraska should look a bit different than the one created in Step 5? Why? > ANSWER: The shape if Nebraska looks a little bit different than the one created in step 5 because UTM is a projected coordinate system, which is planar coordinate system. This in turn leads to stretching due to trying to convert a circular object to a planar surface.

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<- counties_sf_utm %>%
  filter(NAME == "Lancaster")

#Create a simple features point geometry from the point

lancaster_pt<- st_as_sf(Lancaster, coords = "geometry", crs=32614 )


#Transform the mask to match the CRS of the counties dataset
lancaster_transformed <- st_transform(lancaster_pt, crs = st_crs(GAGE_HEIGHT_UTM))

#Create a boolean mask
resultMask <- st_intersects(lancaster_transformed,
                            GAGE_HEIGHT_UTM,
                            sparse = FALSE)

#use mask to select certain county

selected_gages <- GAGE_HEIGHT_UTM[resultMask,]




#Plot

ggplot()+
  geom_sf(data = lancaster_transformed) +
  geom_sf(data = selected_gages, color = "magenta", size =2)+
ggtitle("Gage Sites in the County of Lancaster, Nebraska")
```
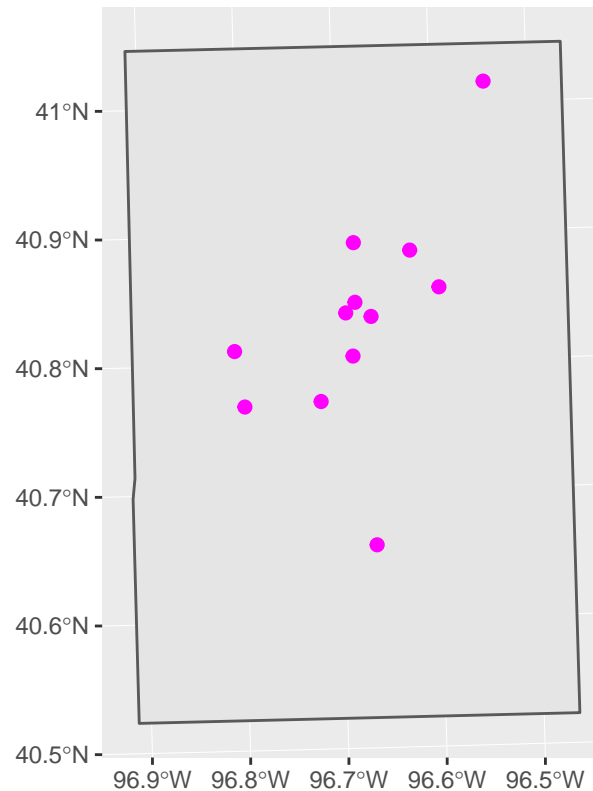
Gage Sites in the County of Lancaster, Nebraska

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 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

```
#Join features

counties_gage_join <- GAGE_HEIGHT_UTM %>%
  st_join(y = counties_sf_utm)

#Show column names

colnames(counties_gage_join)

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

#Plot
```
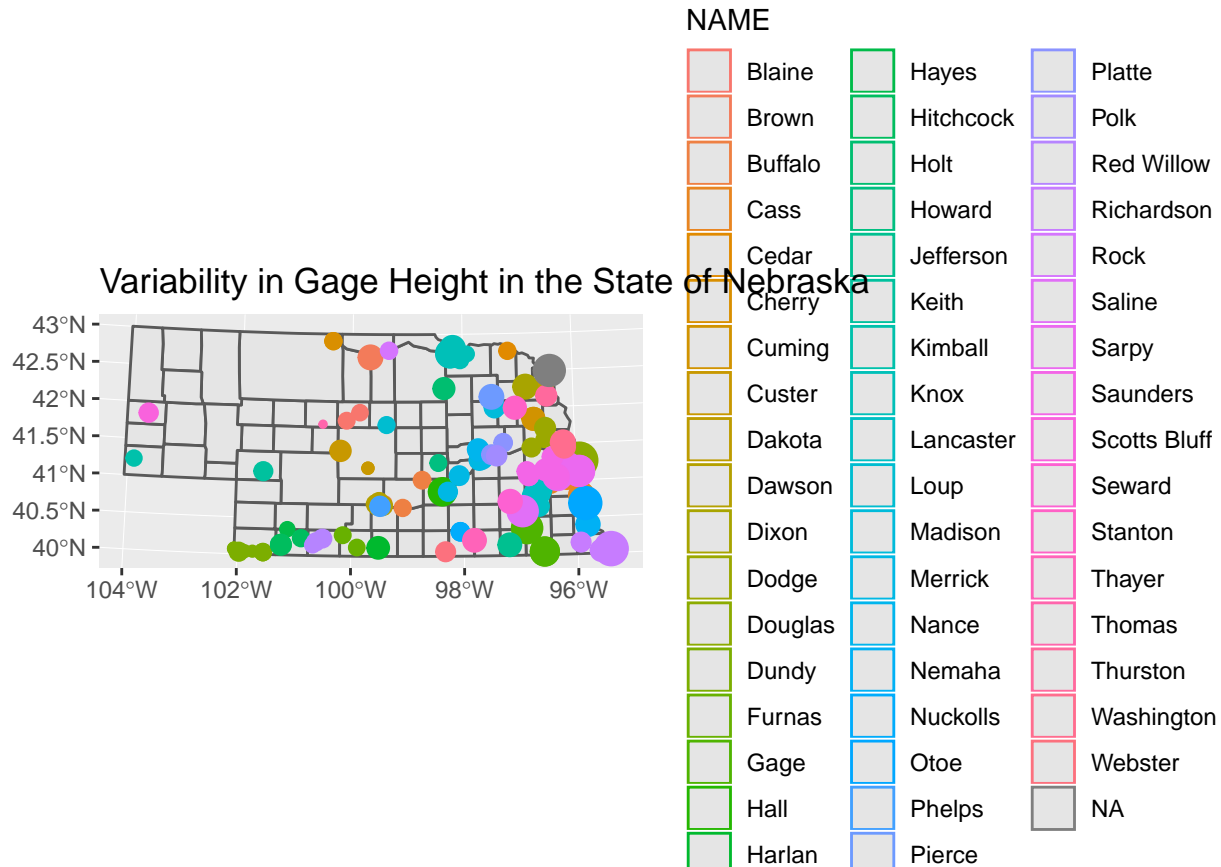
```
ggplot()+
  geom_sf(data = counties_sf_utm) +
  geom_sf(data = counties_gage_join,aes(color=NAME, size = gage_ht))+
  guides(size=FALSE)+
  ggtitle("Variability in Gage Height in the State of Nebraska")
```

NAME

Blaine	Hayes	Platte
Brown	Hitchcock	Polk
Buffalo	Holt	Red Willow
Cass	Howard	Richardson
Cedar	Jefferson	Rock
Cherry	Keith	Saline
Cuming	Kimball	Sarpy
Custer	Knox	Saunders
Dakota	Lancaster	Scotts Bluff
Dawson	Loup	Seward
Dixon	Madison	Stanton
Dodge	Merrick	Thayer
Douglas	Nance	Thomas
Dundy	Nemaha	Thurston
Furnas	Nuckolls	Washington
Gage	Otoe	Webster
Hall	Phelps	NA
Harlan	Pierce	

Variability in Gage Height in the State of Nebraska

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

#drop na
counties_gage_join_no_NA<-drop_na(counties_gage_join)

#find mean of gage height
mean_gage_county<-counties_gage_join_no_NA %>%
  group_by(NAME)%>%
  summarize(MEAN_GAGE_HEIGHT = mean(gage_ht))
```

```
#Convert result to a simple dataframe
mean_gage_county_simple <- st_drop_geometry(mean_gage_county)

#Join summary to County fc
counties_summary_join <- counties_sf_utm %>%
  left_join(y = mean_gage_county_simple, by = "NAME")

#Plot

ggplot()+
  geom_sf(data = counties_summary_join) +
  geom_sf(data = counties_summary_join,aes(fill=MEAN_GAGE_HEIGHT))+
  scale_fill_distiller("Mean Gage Height (m)", palette = "Blues", direction = -1)+
  ggtitle("Mean Gage Height for the Counties of Nebraska")
```