sewageandrain

Loading the libraries needed for the analysis.

```
library(tidycensus)
library(tidyverse)
library(dplyr)
library(units)
library(sf)
census_api_key("125fdf4db95fe2b4e4e00e1253ab9e87d2864a58", overwrite = TRUE, install = TRUE)
```

```
## [1] "125fdf4db95fe2b4e4e00e1253ab9e87d2864a58"
```

```
library(tigris)
options(tigris_class = "sf")
options(tigris_use_cache = TRUE)
library(ggplot2)
library(mapbaltimore)
library(tidyr)
library(purrr)
library(xts)
library(lubridate)
library(mapview)
library(plotly)
library(tmap)
library(gifski) #used to make gifs
library(raster) #used for the area
library(RColorBrewer) #used for color palettes
library(areal)
library(biscale)
library(cowplot)
library(magick)
library(spdep)
library(RColorBrewer)
```

Calling in population and income for Baltimore City based on census tracts.

```
#pulling in the income and population information per census tract in 2019
bmore_data <- get_acs(
  geography = "tract",
  variables = c("income" = "B19013_001",
```

```
                "population" = "B01001_001"),
  state = "MD", #FIPS code 24
  county = "Baltimore City", #FIPS code 510
   year = 2019,
  geometry = TRUE,
   output = "wide"
) %>%
  st_transform(3857) #Changing to Web Mercator

# Download the water for Baltimore city and county
bmore_water <- area_water("MD", c(510,005), class = "sf") %>% filter(AWATER > 20000) #calls water for b
bmore_water <- st_transform(bmore_water, 3857) #Transform to Web Mercator

bmore_water <- st_make_valid(st_buffer(bmore_water, 0)) #Fix topology

st_erase <- function(x, y) {
  st_difference(x, st_make_valid(st_union(st_combine(y)))) # clipping x intersection (y)^(complement)
}

#Erase water from joined variable
bmore_minuswater <- st_erase(bmore_data,bmore_water)
# plot(st_geometry(bmore_minuswater))
```

This is reading in the csv file that have the sewage overflow information, and adjusting it in a way that is most useful to what I am trying to do here.

```
#pulling in the sewage overflow data downloaded from Maryland Dept of the Environment filtering out by
overflows_total <- read.csv(file = "/Users/Tyrah/adv GIS classwork/finalproject_687/overflows_from_MDE.
overflows19 <- subset(overflows_total, select = c(Date_discovered,Time_discovered, Latitude, Longitude)

overflows_points = st_as_sf(overflows19, coords = c("Longitude", "Latitude"), crs = 4326) %>% st_transf

# plot(st_geometry(overflows_points), pch=16, col="navy") #pch denotes the shape of a circle, this is m
```

This is reading in the rain data files, one with the lat/long info per pixel and the other with the data at 15 min intervals everyday from April to September.

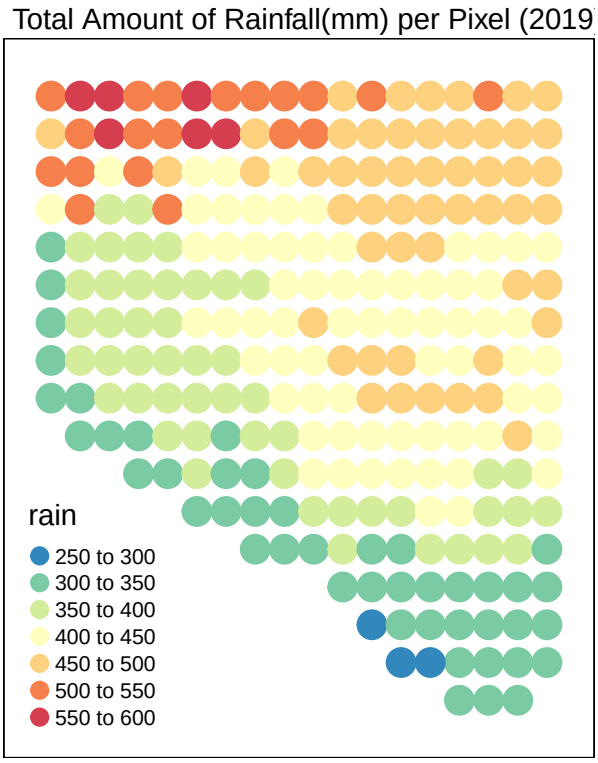
```
## Uploading rainfall data and the corresponding lat/long locations from another data file
#this is pulling in the rainfall data for 2019, information collected every 15min
rainfall_data <- read_csv("/Users/Tyrah/adv GIS classwork/finalproject_687/BaltCity2019_Tyrah_finalproj

#this is the location of the pixels collecting the rainfall data
pixels_latlong <- read_csv("/Users/Tyrah/adv GIS classwork/finalproject_687/Balt_latlong.csv")
```

```r
# this is getting the geometries of the pixels
# pixel_location <- st_as_sf(pixels_latlong, coords = c("longitude", "latitude"),  crs = 4326)
# plot(pixel_location$geometry)
```

```r
#trying to pivot the information to make the data in a long format rather than wide...
rainfall_data_transposed <- rainfall_data %>% pivot_longer(cols = 3:244, names_to = "gridnum", values_t

#had to change this column to be of character type (original a double), so that it could join with the
pixels_latlong$PixelNumber <- as.character(pixels_latlong$PixelNumber)

rainwithlatlong <- left_join(rainfall_data_transposed, pixels_latlong, by = c("gridnum" = "PixelNumber"

rainwithlatlong$gridnum <- as.vector(rainwithlatlong$gridnum)

#if i dont use the removed zeros in the group by can also use rainwithlatlong
removed_zeros <- filter(rainwithlatlong, rainmm > 0)
dayandpixel <- removed_zeros %>% group_by(Date, gridnum, latitude, longitude)
# sumbyday_rain <- dayandpixel %>% summarise(
#   sum = sum(rainmm)
# )

#this is grouping by grid number and summarizing the rain information by total amount and average amoun
by_gridnum <- rainwithlatlong %>% group_by(gridnum, latitude, longitude)
sum_rain <- by_gridnum %>% summarise(
  rain = sum(rainmm),
  mean = mean(rainmm)
)
```

```
## `summarise()` has grouped output by 'gridnum', 'latitude'. You can override
## using the `.groups` argument.
```

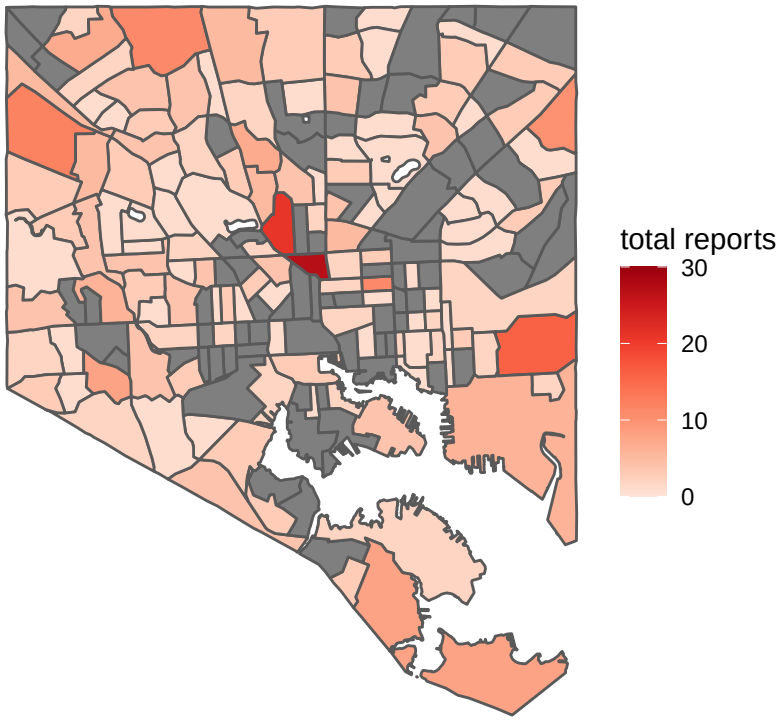
```r
rain_sf <- st_as_sf(sum_rain, coords = c("longitude", "latitude"),  crs = 4326)
rain_proj <- rain_sf %>% st_transform(3857)

totalrain_plot <- tm_shape(rain_sf) +
  tm_dots(group ="rain", col = "rain", size = 1, palette = "-Spectral") +
  tm_layout( outer.margins = rep(0.06, 6),inner.margins = rep(0.08, 8), main.title = "Total Amount of Ra
totalrain_plot
```



This is showing the number of sewage over flow points in each tract based on the lat/long of the sewage overflow occurrence.

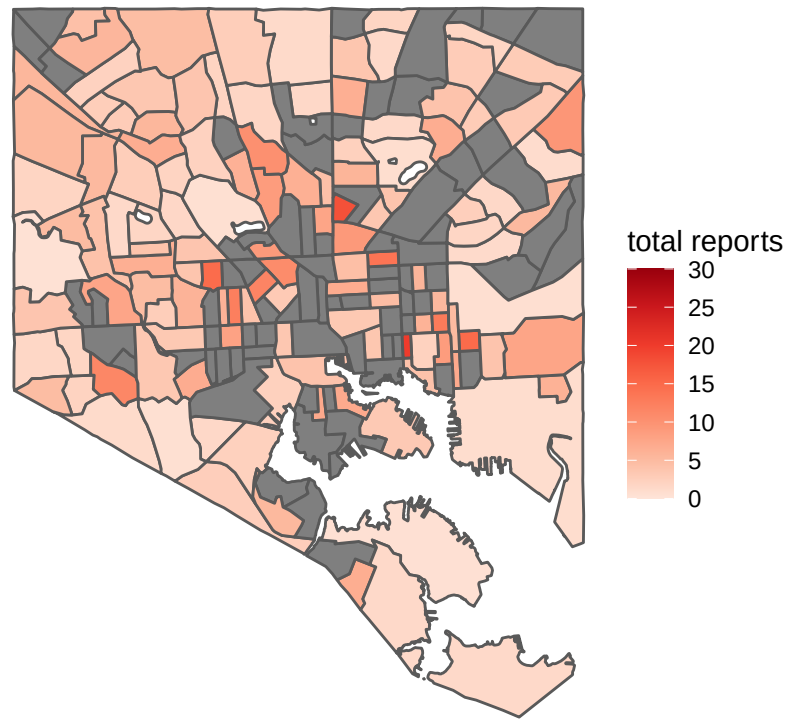
```
#this is joining datasets to show the points found within each tract
overflows_tracts_join <- st_join(overflows_points, bmore_minuswater)

#this is counting the number of reports per tract
reports_intracts_count <- count(as_tibble(overflows_tracts_join), GEOID)

info2019 <- left_join(bmore_minuswater, reports_intracts_count, by=c("GEOID")) #this joins the info so

reportsbytract <- ggplot(data = info2019, aes(fill = n)) +
  geom_sf() +
  scale_fill_distiller(palette = "Reds",
                       direction = 1,
                       limits = c(0,30)) +
  labs(title = "2019 Total Sewage Overflow Reports",
       caption = "Overflow information provided by MDE",
       fill = "total reports") +
  theme_void()

plot(reportsbytract)
```

2019 Total Sewage Overflow Reports



Overflow information provided by MDE

```
info2019$area <- st_area(info2019$geometry) #calculating area via geometry, outputin units m^2
info2019$area <- drop_units(info2019$area) #this is dropping the units so that the dataframe is easier

info2019$sqmi <- (info2019$area * 0.00000038610) #converting to miles^2

info2019$reportspersqmi <- (info2019$n / info2019$sqmi) #this is calculating density
# info2019$reportspersqmi <- drop_units(info2019$reportspersqmi) #again, dropping the units

reports_sqmi<- ggplot(data = info2019, aes(fill = reportspersqmi)) +
  geom_sf() +
  scale_fill_distiller(palette = "Reds",
                       direction = 1,
                       limits = c(0,30),
                       breaks=c(0,5,10, 15, 20, 25,30)) +
  labs(title = "Sewage Overflow Report Density (per Sq Mi)",
       caption = "Overflow information provided by MDE",
       fill = "total reports") +
  theme_void()

plot(reports_sqmi)
```



Overflow information provided by MDE

```
ggsave(filename = "reports_persqmi.png", plot=reports_sqmi,width=4,height=4,units="in",scale=1)
```

#This is breaking Baltimore City up as a series of hexagons for future comparisons.

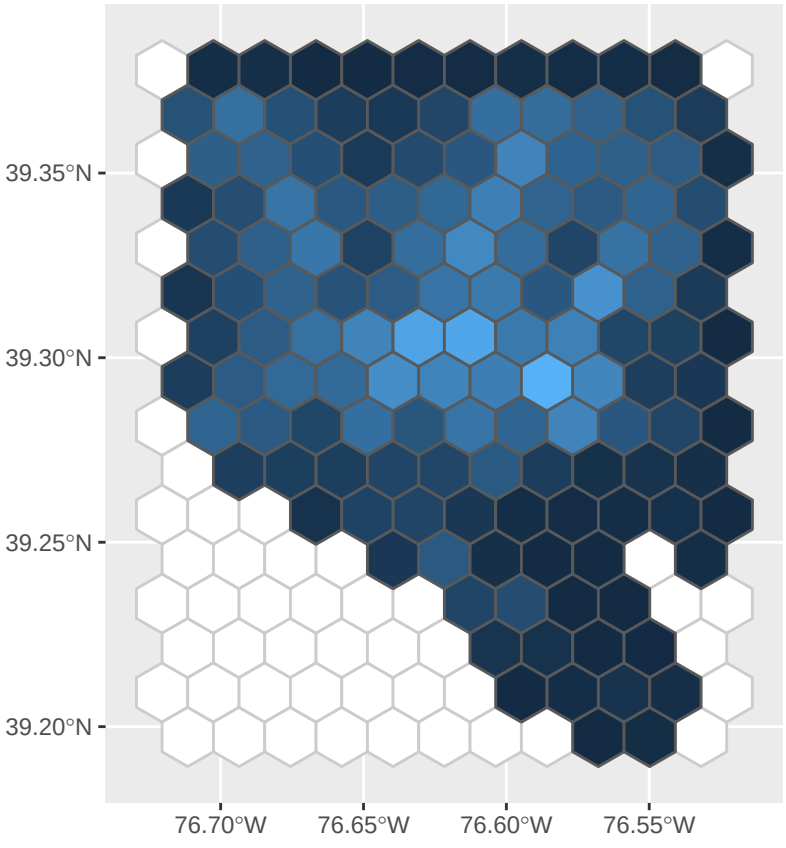
```
## Make and subset grid
# Make a grid
bmore_grid_2 <- st_make_grid(bmore_minuswater,
  2 * 1000, # Kms
  crs = 3857,
  what = "polygons", #you can also create lines
  square = FALSE # hexagons , knows it is a hexagon if squares is set to false
)

# To sf
bmore_grid_2 <- st_sf(index = 1:length(lengths(bmore_grid_2)), bmore_grid_2) # Add index , #making a sf

#plot(st_geometry(bmore_grid), border="#aaaaaa", lwd = .1)
#plot(st_geometry(bmore_bg_income), add=T, lwd = .1)

bmore_shape_2 <- st_union(bmore_minuswater) #stunion dissolves it and makes it one polygon shape that i

bmore_grid2.intersects <- st_intersects(bmore_shape_2, bmore_grid_2)
bmore_grid2.subset <- bmore_grid_2[bmore_grid2.intersects[[1]],]

# plot(st_geometry(bmore_grid2.subset), col="blue")
```

#This is doing the areal interpolation of the data.

```
ar_validate(source = info2019, target = bmore_grid2.subset, varList = "populationE", method = "aw")
```

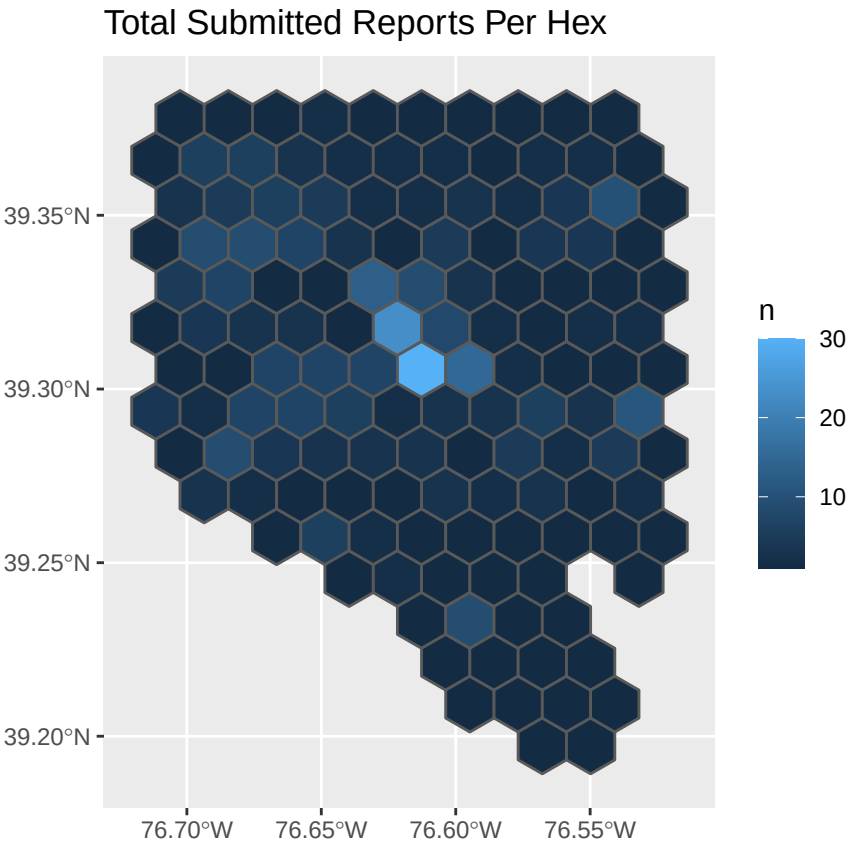
```
## [1] TRUE
```

```
bmore_interpolate <- aw_interpolate(bmore_grid2.subset, tid = index, source = info2019, sid = "GEOID", 


# plot(st_geometry(bmore_interpolate))

ggplot() +
  geom_sf(
    data = bmore_grid_2,
    fill = "white", colour = "gray80"
  ) +
  geom_sf(
    data = bmore_interpolate,
    mapping = aes(fill = populationE), show.legend = FALSE
  ) +
  coord_sf()
```



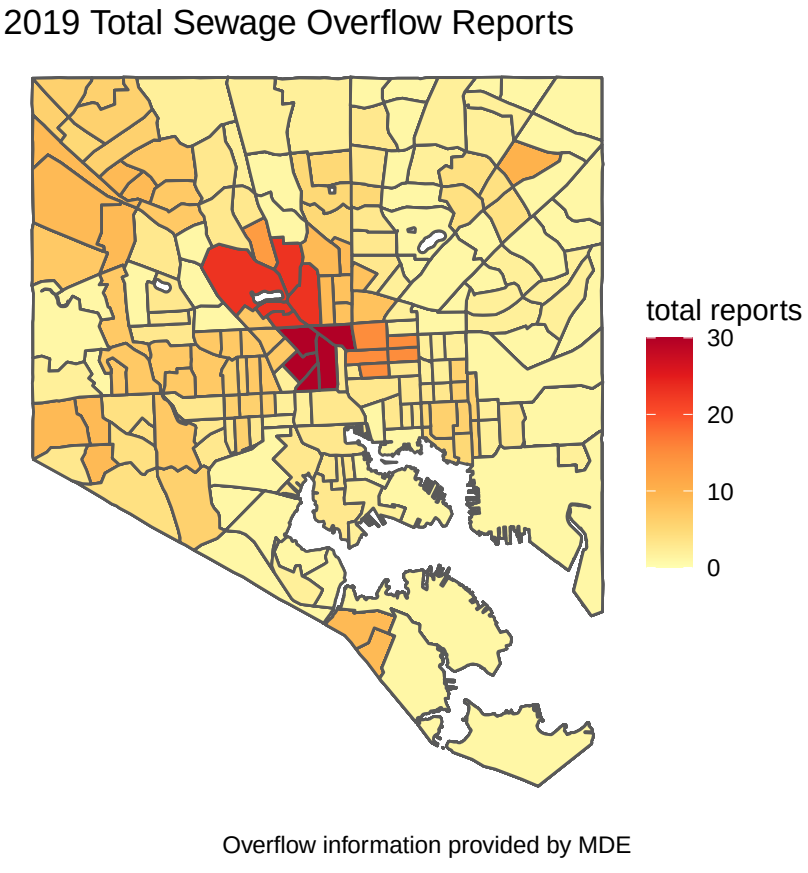
This is grouping the hexagons by index and counting how many times an index occurs because that tells us the number of reports per hex.

```
reports_hex_join <- st_join(bmore_interpolate, overflows_points) #this joins the Baltimore hex grid to
reportsbyindex <- reports_hex_join %>% group_by(index) %>% count(index) #this is grouping the data by i

reportsbyhex_map <- ggplot() +
  geom_sf(
    data = reports_hex_join,
    fill = "white", colour = "gray80"
  ) +
  geom_sf(
    data = reportsbyindex,
    mapping = aes(fill = n), show.legend = TRUE
  ) +
  coord_sf() +
    labs(title="Total Submitted Reports Per Hex")

reportsbyhex_map
```



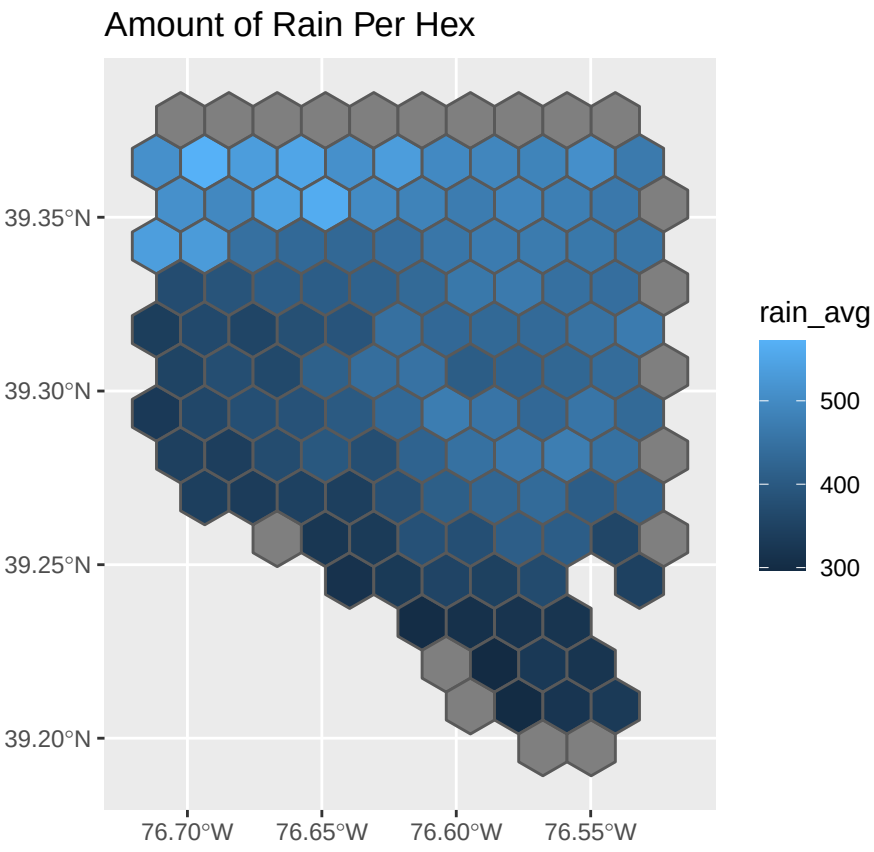
This is joining the hexagon data for overflow reports and relating it to the geometry of Baltimore tracts. This offers a better understanding of how the reports are distributed when compared to the initial map that only based on lat/long of the reports.

```
hexreports_tracts_join <- st_join(bmore_minuswater, reportsbyindex)

# plot(hexreports_tracts_join$geometry)

hexreports_tracts_join_map <- ggplot(data = hexreports_tracts_join, aes(fill = n)) +
  geom_sf() +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1,
                       limits = c(0,30)) +
  labs(title = "2019 Total Sewage Overflow Reports",
       caption = "Overflow information provided by MDE",
       fill = "total reports") +
  theme_void()

plot(hexreports_tracts_join_map)
```



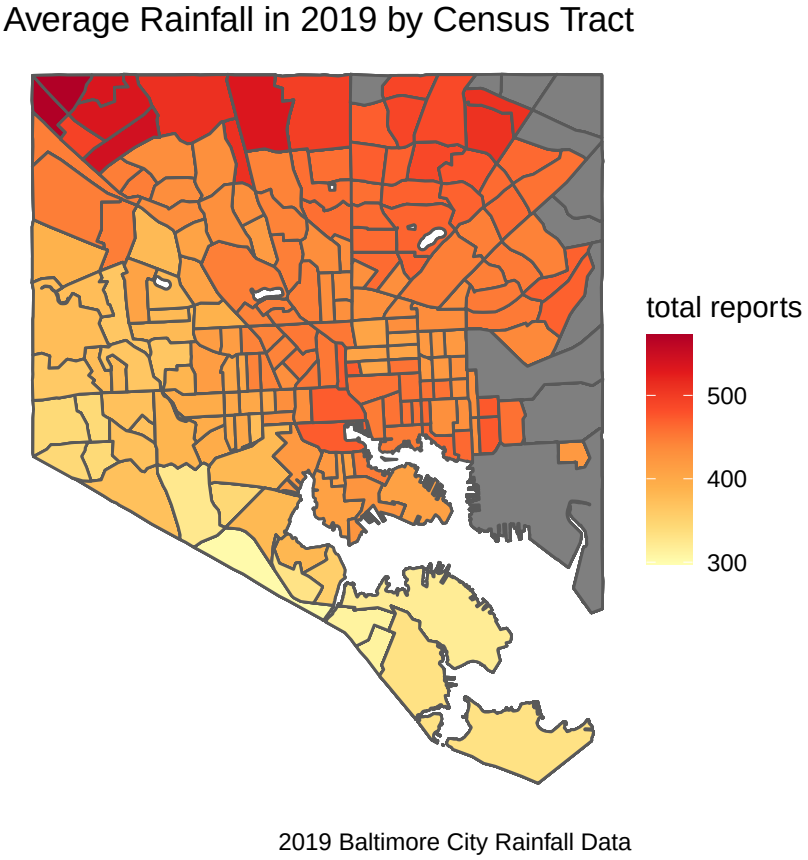
This is breaking down the city by a hexagon grid and then relating the rainfall data per hexagon cell.

```
rain_hex_join <- st_join(bmore_interpolate, rain_proj) #this joins the Baltimore hex grid to the rainfa
rainbyindex <- rain_hex_join %>% group_by(index) %>% summarize(rain_avg  = mean(rain)) #this is groupin

rainbyhex_map <- ggplot() +
  geom_sf(
    data = rain_hex_join,
    fill = "white", colour = "gray80"
  ) +
  geom_sf(
    data = rainbyindex,
    mapping = aes(fill = rain_avg), show.legend = TRUE
  ) +
  coord_sf() +
    labs(title="Amount of Rain Per Hex")

rainbyhex_map
```



This is joining the rain information from hexagons to Baltimore Geometry to get an understanding of how rain is distrubuted on average across each tract in a 6 month time span in 2019 (April to September)

```
hexrain_tracts_join <- st_join(bmore_minuswater, rainbyindex)

#plot(hexrain_tracts_join$geometry)

hexreports_tracts_join_map <- ggplot(data = hexrain_tracts_join, aes(fill = rain_avg)) +
  geom_sf() +
  scale_fill_distiller(palette = "YlOrRd",
                       direction = 1) +
  labs(title = "Average Rainfall in 2019 by Census Tract",
       caption = "2019 Baltimore City Rainfall Data",
       fill = "total reports") +
  theme_void()

plot(hexreports_tracts_join_map)
```



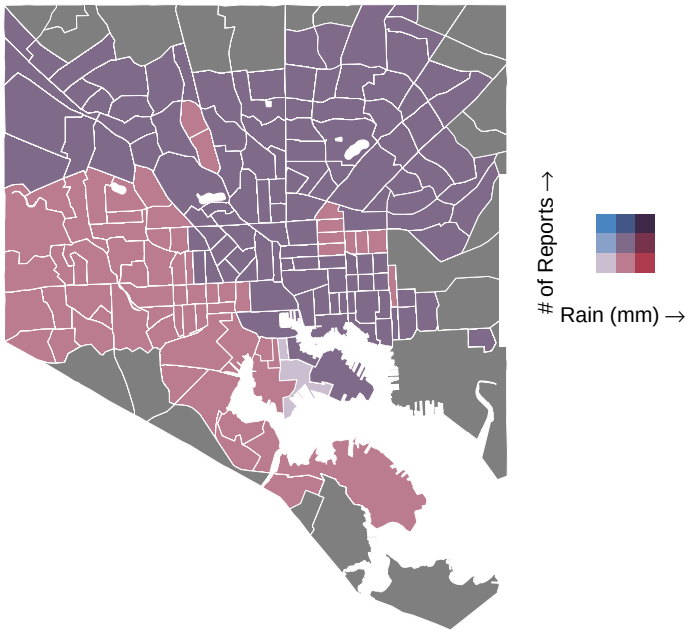
Trying to relate hexagons to census tracts via a spatial join.

```
#did this join so that total number of reports could be joined together in the same dataframe as Baltim
reportsanderain_hex<- st_join(rainbyindex,hexreports_tracts_join)

reportsandrain_baltimore <- st_join(bmore_minuswater,reportsanderain_hex, by=c("GEOID"))

totalreportsandrainjoin <- st_join(reportsandrain_baltimore,rainbyindex, by=c("index"))

rain_reports <- bi_class(reportsandrain_baltimore, x = n, y = rain_avg, style = "quantile", dim = 2)
```

```
## Warning in classInt::classIntervals(bins_y, n = dim, style = "quantile"): var
## has missing values, omitted in finding classes
```

```
# create map
rain_reports_map <- ggplot() +
  geom_sf(data = rain_reports, mapping = aes(fill = bi_class), color = "white", size = 0.1, show.legend
  bi_scale_fill(pal = "DkViolet", dim = 3) +
  labs(title = "Sewage Reports and Rain") +
  bi_theme()

# rain_reports_map

rain_reports_legend <- bi_legend(pal = "DkViolet",
                    dim = 3,
                    xlab = "Rain (mm)",
                    ylab = " # of Reports",
                    size = 8)

rain_reports_final <- ggdraw() +
  draw_plot(rain_reports_map, 0, 0, 1, 1) +
  draw_plot(rain_reports_legend, 0.68, .4, 0.2, 0.2)

rain_reports_final
```

Sewage Reports and Rain



This is finding the number (n) of submitted sewage overflow reports from 2019 per hex

Comparing Income and the number of reports per Census Tract

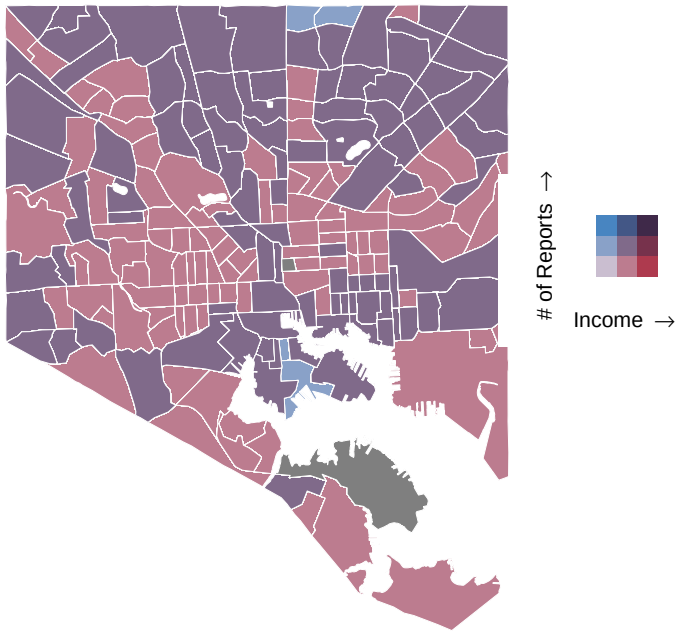
```
income_reports <- bi_class(totalreportsandrainjoin, x = n, y = incomeE.x, style = "quantile", dim = 2)
```

```
## Warning in classInt::classIntervals(bins_y, n = dim, style = "quantile"): var
## has missing values, omitted in finding classes
```

```
# create map
income_reports_map <- ggplot() +
  geom_sf(data = income_reports, mapping = aes(fill = bi_class), color = "white", size = 0.1, show.lege
  bi_scale_fill(pal = "DkViolet", dim = 3) +
  labs(title = "Sewage Reports and Income") +
  bi_theme()

# income_reports_map

income_reports_legend <- bi_legend(pal = "DkViolet",
                    dim = 3,
                    xlab = "Income ",
                    ylab = " # of Reports ",
                    size = 8)

income_reports_final <- ggdraw() +
  draw_plot(income_reports_map, 0, 0, 1, 1) +
```

```
  draw_plot(income_reports_legend, 0.68, .4, 0.2, 0.2)

income_reports_final
```

Sewage Reports and Income




```
pop_reports <- bi_class(totalreportsandrainjoin, x = n, y = populationE.x, style = "quantile", dim = 2)

# create map
pop_reports_map <- ggplot() +
  geom_sf(data = pop_reports, mapping = aes(fill = bi_class), color = "white", size = 0.1, show.legend =
  bi_scale_fill(pal = "DkViolet", dim = 3) +
  labs(
    title = " Sewage Reports and Population") +
  bi_theme()
# pop_reports_map

pop_reports_legend <- bi_legend(pal = "DkViolet",
                    dim = 3,
                    xlab = "Population ",
                    ylab = " # of Reports ",
                    size = 8)

pop_reports_final <- ggdraw() +
  draw_plot(pop_reports_map, 0, 0, 1, 1) +
  draw_plot(pop_reports_legend, 0.68, .4, 0.2, 0.2)

pop_reports_final
```

Sewage Reports and Population