Module 3: Redlining

```
library(sf) # simple features for R
library(terra) # spatial data analysis
library(tidyterra) #tidyverse methods for terra objects
library(tidyverse)
```

Use of GitHub

Link to your forked GH repository:

https://github.com/AFhmc/CLES131-module3-redlining#

Use of Quarto

Link to your .qmd file:

https://github.com/AFhmc/CLES131-module3-redlining#

Ecological consequences of redlining

In August 2020, Christopher Schell and collegues published a review in *Science* on 'The ecological and evolutionary consequences of systemic racism in urban environments' showing how systematic racism and classism has significant impacts on ecological and evolutionary processes within urban environments. Here, we combine spatial data to reproduce and extend an analysis from the paper.

The vector data

We will use a vector dataset of redlining maps from Mapping Inequality, a project led by Robert K. Nelson.

Q1 (1 point)

Click 'Explore the Maps' to look at some cities and neighborhoods you are familiar with. Who is the intended audience of this data science project, and how are the data used to communicate understanding, insight, and knowledge? Why is this effective?

The intended audience of this data science project seems to be those who want to use redlining data in other data science analysis or potentially as a teaching tool. I believe this because they make the data very accessible in multiple formats, and it isn't particularly accessible to the layman (I had trouble matching the maps to my mental images of any of these regions, they were not particularly legible). The data communicates understanding by clearly breaking the map into chunks which are colorcoded by their redlining status (which is very easy to interpret), and by showing the percentages of each category for each city, which is helpful contextual information. The large map with circles on it representing which cities did redlining and letting you eyeball the category percentages is also useful large scale information (ie I can see at a glance that Philadelphia and Essex co both did redlining but Philadelphia had a higher proportion of 'red' neighborhoods). The interactivity is also very effective, because it allows you to search for specific cities, glance over the map as a whole, and hover over regions for more information.

Q2 (1 point)

Create a `data/` folder in the root of your project and create five subfolders labeled with the city names from Fig. 2 of Schell et al. 2020. Because the spatial files will be large, add `data/` to the .gitignore file.

Then, go back to the home page of Mapping Inequality and select 'Download the Data'. Use the search bar to select and download spatial data for each city. Move the geojson file into the associated data subfolder.

Import the geojson file into your R environment with the `st_read()`' function from sf. Check the structure of this object and see that it is a special type of data frame, allowing it to be manipulated with many of the functions you already know, including ggplot.

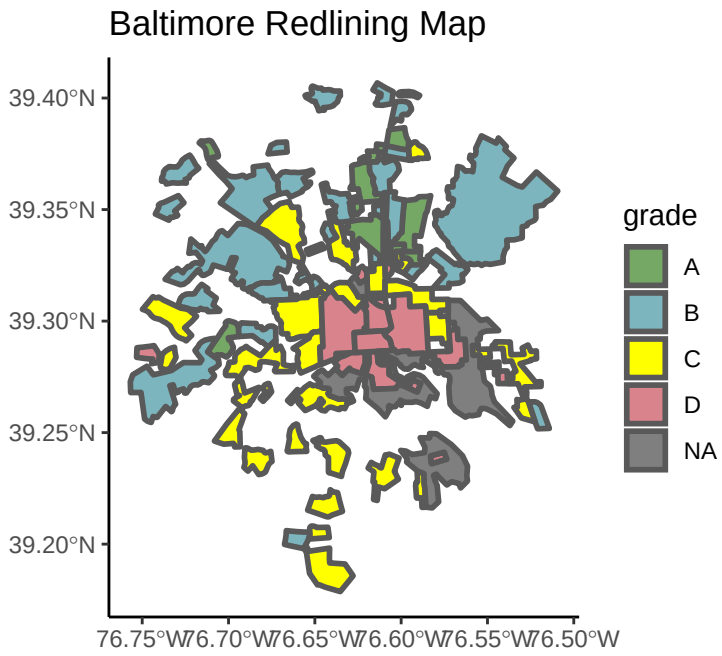
Make a quick plot of your first city showing the "grade" in color using ggplot syntax and `geom_sf()`. Select a color scheme that better comports with redlining.

```
baltimore_redlining <- st_read("data/Baltimore/geojson.json")
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Balti
  using driver `GeoJSON'
Simple feature collection with 60 features and 14 fields
```

```
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -76.7569 ymin: 39.17875 xmax: -76.50927 ymax: 39.4068
Geodetic CRS:  WGS 84
```

```
ggplot()+
  geom_sf(data = baltimore_redlining, linewidth = 1, mapping = aes(fill = grade)) + theme_cla
  scale_fill_manual(values = c("A" = "#76a865",
                        "B" =  "#7cb5bd",
                        "C" = "#ffff00",
                        "D" = "#d9838d",
                        na.value = "grey")) +
  labs(title= "Baltimore Redlining Map")
```



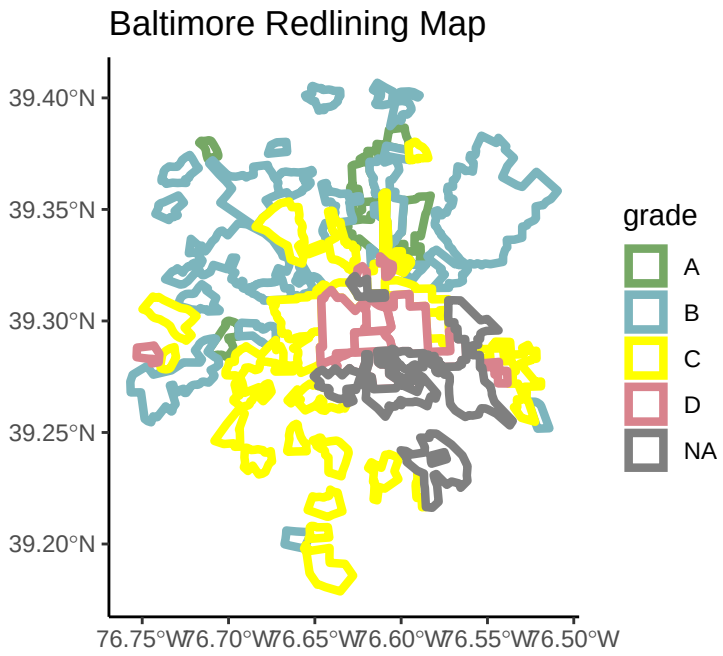
(Alternate version with no fill and colored outlines)

```
baltimore_redlining <- st_read("data/Baltimore/geojson.json")
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Balti
  using driver `GeoJSON'
Simple feature collection with 60 features and 14 fields
Geometry type: MULTIPOLYGON
```

```
Dimension:      XY
Bounding box:  xmin: -76.7569 ymin: 39.17875 xmax: -76.50927 ymax: 39.4068
Geodetic CRS:  WGS 84
```

```
ggplot()+
  geom_sf(data = baltimore_redlining, linewidth = 1.5, fill = "white", mapping = aes(color =
  scale_color_manual(values = c("A" = "#76a865",
                          "B" =  "#7cb5bd",
                          "C" = "#ffff00",
                          "D" = "#d9838d",
                          na.value = "grey")) +
  labs(title= "Baltimore Redlining Map")
```



The raster data

We will also be calculating NDVI from the European Space Agency's Sentinel-2 Mission, specifically bands B4 (red) and B8 (near infrared). There are multiple steps to importing the data, which itself takes a long time, so please get an early start.

- Click "Explore Sentinel-2 data" on this page and create an account to login
- In the explorer, make sure Sentinel-2 L2A is selected (Level 2A, atmospheric correction applied)
- Scroll and zoom to the city of choice

- Use the polygon tool (upper right corner, hover over pentagon icon and select rectangle) to draw a bounding box. Adjust until the extent approximates those in Schell et al. 2020. Try selecting the "False color" layer to help diagnose features to include or exclude
- Set a threshold for cloud cover and select a date that reasonably approximates peak greenness. You may have to test multiple options to locate it, and not all cities will have the same date
- Once the displayed images looks satisfactory, click "Find products for current view"
- Check the desired product and download. It will take a while because the files are large

The data will be packaged as a zipped SAFE file in your Downloads folder. You may need to investigate the properties of the file and click 'unblock' to give permission to open. Once unzipped, you will find: - The images are jpeg2000 files nested within the GRANULE and IMG_DATA subfolders - Multiple resolutions and bands are available - Metadata is provided in `MTD_MSIL2A.xml`

For each city, locate the 10m resolution files for the B04 and B08 bands along with associated metadata and copy them to `data/city_name/` within this project.

Q3 (1 point)

Use the terra package and the `rast()` function to import the two bands, which are reported as digital numbers. Combine to calculate NDVI ($ (NIR - R) / (NIR + R)$) and display a quick plot of your first city.

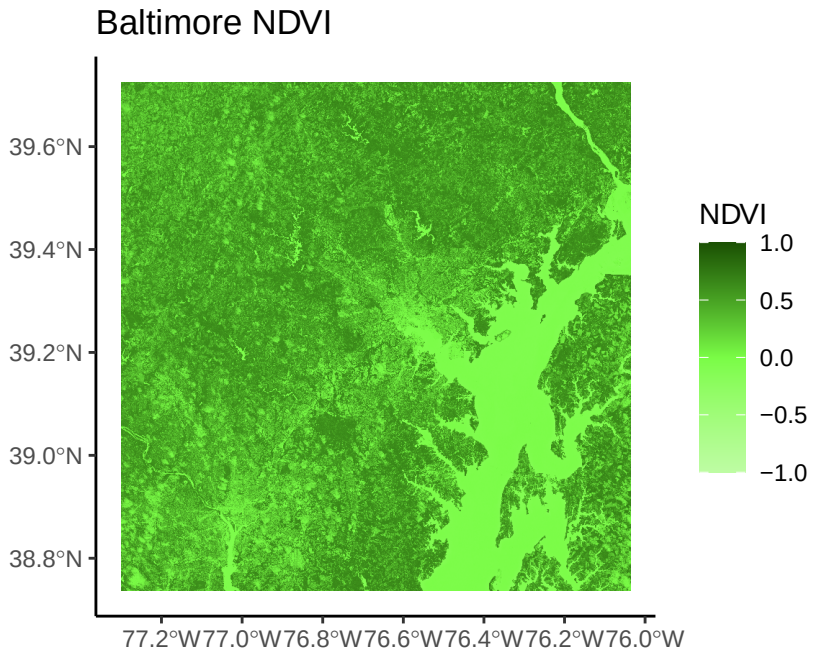
```
city <- "Baltimore"
#Note that I renamed all the files to Band04/Band08 to make this loopable
Baltimorefilepath04 <- paste("data", city, "Band04.jp2", sep = "/")
Baltimorefilepath08 <- paste("data", city, "Band08.jp2", sep = "/")


#B8 is NIR, B4 is Red

BaltimoreredBand <- terra::rast(Baltimorefilepath04)
BaltimoreNIRband <- terra::rast(Baltimorefilepath08)

BaltimoreNDVI = (BaltimoreNIRband - BaltimoreredBand)/(BaltimoreNIRband + BaltimoreredBand)
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================
```

```r
ggplot()+
  geom_spatraster(data = BaltimoreNDVI)+
  scale_fill_gradient2(low = "#befca4",
                       mid = "#7afc46",
                       high = "#1b5104",
                       na.value = NA, limits = c(-1,1)) + theme_classic() +
  labs(title = "Baltimore NDVI", fill = "NDVI")
```

```
<SpatRaster> resampled to 501264 cells.
```

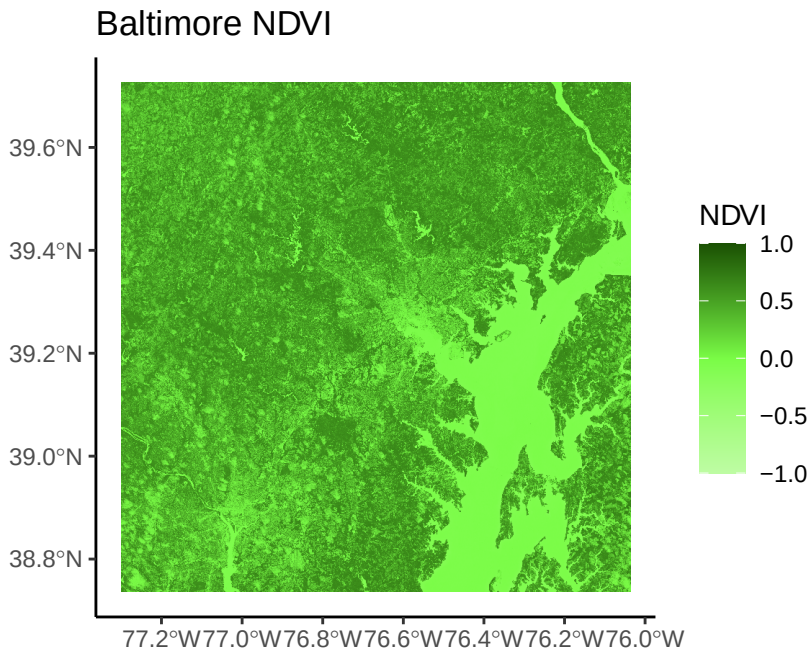


Bonus 1 (1 point)

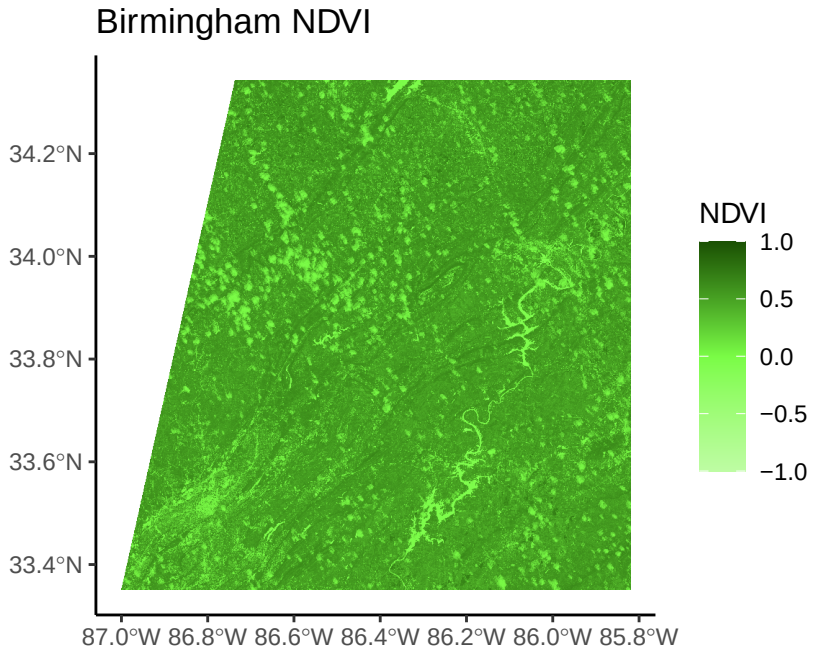
Since you have 5 cities to plot, can you optimize the same operations as Q3 with a for loop?

```
for (city in c("Baltimore", "Birmingham", "Indianapolis", "Minneapolis", "Phoenix")) {
  #Note that I renamed all the files to Band04/Band08 to make this loopable
  filepath04 <- paste("data", city, "Band04.jp2", sep = "/")
  filepath08 <- paste("data", city, "Band08.jp2", sep = "/")


  #B8 is NIR, B4 is Red

  redBand <- terra::rast(filepath04)
  NIRband <- terra::rast(filepath08)

  NDVI = (NIRband - redBand)/(NIRband + redBand)

  #plot
  plot <- ggplot()+
  geom_spatraster(data = NDVI)+
  scale_fill_gradient2(low = "#befca4",
                       mid = "#7afc46",
                       high = "#1b5104",
                       na.value = NA, limits = c(-1,1)) + theme_classic() +
  labs(title = paste(city, "NDVI"), fill = "NDVI")

  print(plot)
}
```

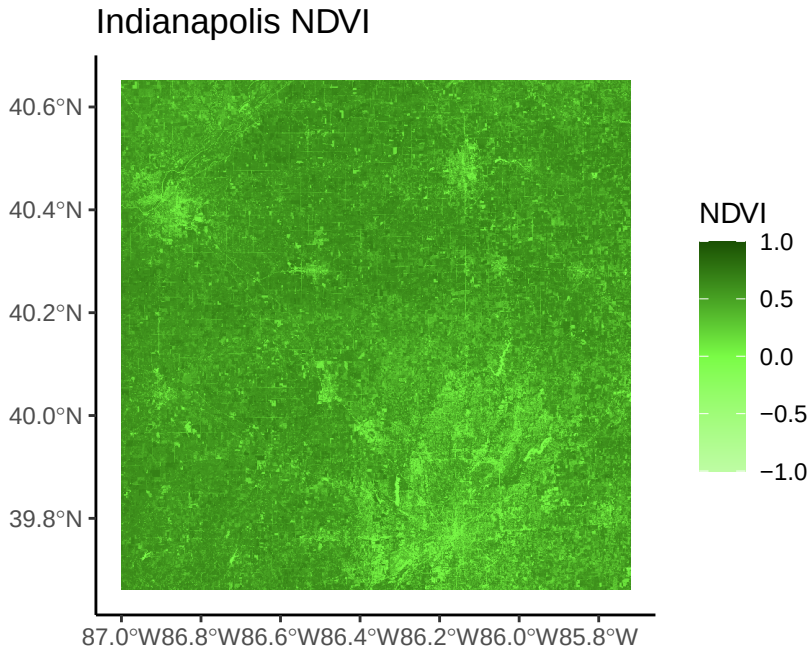
```
|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


<SpatRaster> resampled to 501264 cells.
```



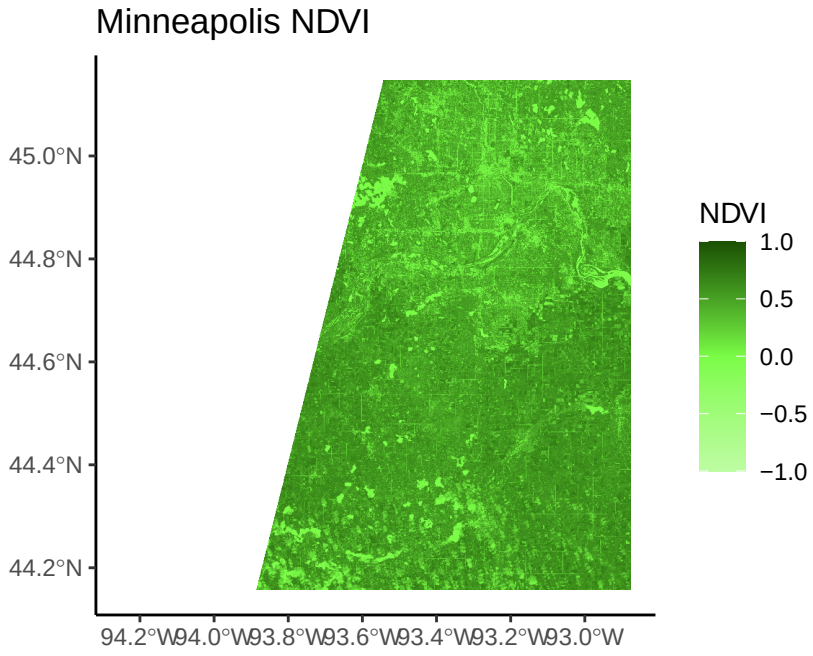

```
|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

<SpatRaster> resampled to 501264 cells.
```



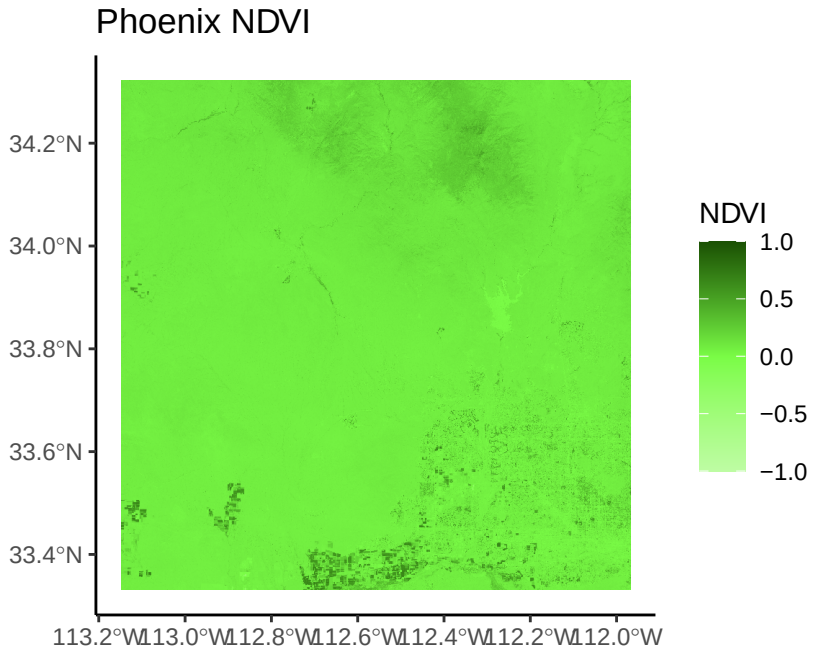

```
|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

<SpatRaster> resampled to 501264 cells.
```




```
|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

<SpatRaster> resampled to 501264 cells.
```




```
|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

|---------|---------|---------|---------|
=========================================

<SpatRaster> resampled to 501264 cells.
```



Q4 (1 point)

Do the rasters and polygons share the same coordinate reference system? If not, project both into the same CRS and justify your choice.

```
#look at the polygon coords:
st_crs(baltimore_redlining)
```

```
Coordinate Reference System:
  User input: WGS 84
  wkt:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
```

```
            ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4326]]
```

```
#look at the raster coords:
st_crs(BaltimoreNDVI)
```

```
Coordinate Reference System:
  User input: WGS 84 / UTM zone 18N
  wkt:
PROJCRS["WGS 84 / UTM zone 18N",
    BASEGEOGCRS["WGS 84",
        ENSEMBLE["World Geodetic System 1984 ensemble",
            MEMBER["World Geodetic System 1984 (Transit)"],
            MEMBER["World Geodetic System 1984 (G730)"],
            MEMBER["World Geodetic System 1984 (G873)"],
            MEMBER["World Geodetic System 1984 (G1150)"],
            MEMBER["World Geodetic System 1984 (G1674)"],
            MEMBER["World Geodetic System 1984 (G1762)"],
            MEMBER["World Geodetic System 1984 (G2139)"],
            MEMBER["World Geodetic System 1984 (G2296)"],
            ELLIPSOID["WGS 84",6378137,298.257223563,
                LENGTHUNIT["metre",1]],
            ENSEMBLEACCURACY[2.0]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4326]],
    CONVERSION["UTM zone 18N",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-75,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",0,
```

```
        LENGTHUNIT["metre",1],
        ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Navigation and medium accuracy spatial referencing."],
        AREA["Between 78°W and 72°W, northern hemisphere between equator and 84°N, onshore ar
        BBOX[0,-78,84,-72]],
    ID["EPSG",32618]]
```

They both use the WGS 84 overall system, but the NDVI uses a modification of that coordinate system to make the projection better for the particular region that it is in. So, I will convert both to use the "WGS 84 / UTM zone 18N" coordinates of the NDVI because that should be better for the particularities of the location.

```
transformed_baltimore_redlining <- st_transform(baltimore_redlining, st_crs(BaltimoreNDVI))
transformed_baltimore_redlining
```

```
Simple feature collection with 60 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 348448.2 ymin: 4337887 xmax: 369968.4 ymax: 4363164
Projected CRS: WGS 84 / UTM zone 18N
First 10 features:
   area_id city_id grade    fill label name category_id sheets         area
1     2877      68     A #76a865    A1                1      1 4.829741e-04
2     2875      68     A #76a865    A2                1      1 9.167742e-05
3     2884      68     A #76a865    A3                1      1 1.551970e-04
4     2897      68     A #76a865    A4                1      1 5.048946e-05
5     2903      68     A #76a865    A5                1      1 1.535498e-04
6     2879      68     A #76a865    A6                1      1 2.908452e-04
7     2881      68     B #7cb5bd    B1                2      1 1.687774e-04
8     2887      68     B #7cb5bd   B10                2      1 2.158681e-03
9     2880      68     B #7cb5bd   B11                2      1 2.305679e-04
10    2910      68     B #7cb5bd   B12                2      1 7.905703e-05
                                                                          boun
1  [ [ 39.327660000000002, -76.634680000000003 ], [ 39.371369999999999, -76.610209999999995 ]
```

```
2  [ [ 39.325200000000002, -76.606399999999994 ], [ 39.335410000000003, -76.584350000000001 ]
3  [ [ 39.371110000000002, -76.624009999999998 ], [ 39.386679999999998, -76.595569999999995 ]
4  [ [ 39.372239999999998, -76.717020000000005 ], [ 39.381030000000003, -76.705870000000004 ]
5  [ [ 39.283819999999999, -76.712159999999997 ], [ 39.300130000000003, -76.694029999999998 ]
6            [ [ 39.33117, -76.605279999999993 ], [ 39.356229999999996, -76.582849999999993 ]
7  [ [ 39.344180000000001, -76.646249999999995 ], [ 39.361849999999997, -76.623660000000001 ]
8            [ [ 39.324730000000002, -76.580640000000002 ], [ 39.38306, -76.509270000000001 ]
9  [ [ 39.314779999999999, -76.598529999999997 ], [ 39.331949999999999, -76.561329999999998 ]
10 [ [ 39.251710000000003, -76.526240000000001 ], [ 39.263809999999999, -76.514139999999998 ]
   residential commercial industrial    label_coords
1         TRUE      FALSE      FALSE 39.339, -76.616
2         TRUE      FALSE      FALSE 39.328, -76.592
3         TRUE      FALSE      FALSE 39.382, -76.601
4         TRUE      FALSE      FALSE 39.376, -76.710
5         TRUE      FALSE      FALSE 39.289, -76.704
6         TRUE      FALSE      FALSE 39.348, -76.593
7         TRUE      FALSE      FALSE 39.349, -76.634
8         TRUE      FALSE      FALSE 39.349, -76.552
9         TRUE      FALSE      FALSE 39.321, -76.570
10        TRUE      FALSE      FALSE 39.256, -76.519
                          geometry
1  MULTIPOLYGON (((361234.9 43...
2  MULTIPOLYGON (((363066.8 43...
3  MULTIPOLYGON (((362559.6 43...
4  MULTIPOLYGON (((352562.4 43...
5  MULTIPOLYGON (((353913.8 43...
6  MULTIPOLYGON (((363066.8 43...
7  MULTIPOLYGON (((360069 4357...
8  MULTIPOLYGON (((366205.7 43...
9  MULTIPOLYGON (((362567.2 43...
10 MULTIPOLYGON (((368643.7 43...
```

Q5 (1 points)

Overlay the projected vector file onto the projected NDVI for a single city using `tidyterra::geom_spatraster()` in addition to `geom_sf()`. Adjust the color scales and add a scale bar to approximate Fig. 2a of Schell et al. 2020.

```
baltimorePoly <- terra::project(terra::vect("data/Baltimore/geojson.json"), BaltimoreNDVI)
BaltimoreNDVI |> terra::mask(baltimorePoly)
```

```
|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


class       : SpatRaster
size        : 10980, 10980, 1  (nrow, ncol, nlyr)
resolution  : 10, 10  (x, y)
extent      : 3e+05, 409800, 4290240, 4400040  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 18N (EPSG:32618)
source      : spat_aa49686a47f0_43593_Ka4v2LrsW3RjnNX.tif
varname     : Band08
name        :     Band08
min value   : -0.3716540
max value   :  0.7403337
```

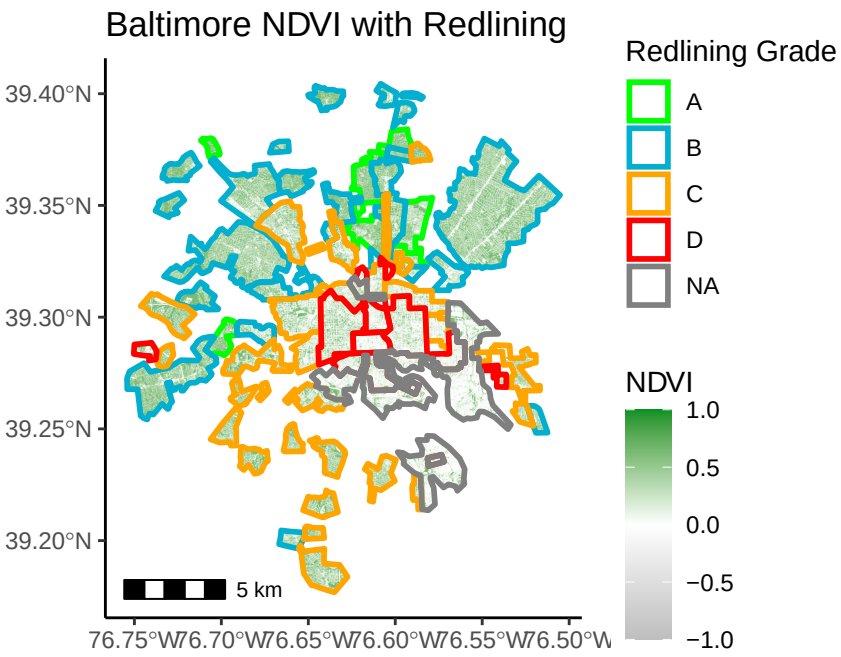
```
#install.packages("ggspatial")
library(ggspatial) # to get scalebar
```

Sadly, I cannot figure out how to to get the background landscape features (streets, waterbodies, etc) in this.

```
#cut the NDVI data down to size
cutBaltimoreNDVI <- BaltimoreNDVI |> terra::crop(transformed_baltimore_redlining)
#and then cut to only within the polygons
baltimorePoly <- terra::project(terra::vect("data/Baltimore/geojson.json"), BaltimoreNDVI)
cutBaltimoreNDVI <-cutBaltimoreNDVI |> terra::mask(baltimorePoly)

ggplot()+
  geom_spatraster(data = cutBaltimoreNDVI)+
  scale_fill_gradient2(low = "grey",
                       mid = "white",
                       high = "#019123",
                       na.value = NA, limits = c(-1,1)) + theme_classic() +
  labs(title = "Baltimore NDVI with Redlining", fill = "NDVI", color = "Redlining Grade")+
  geom_sf(data = transformed_baltimore_redlining, linewidth = 1, fill = NA, mapping = aes(co
  scale_color_manual(values = c("A" = "green",
                       "B" =  "#02afce",
                       "C" = "orange",
```

```
                        "D" = "red",
                        na.value = "grey")) +
  annotation_scale()
```

```
<SpatRaster> resampled to 500851 cells.
```



Q6 (1 point)

Repeat the above for all 5 cities and add the city name. Explore the `cowplot` or `patchwork` packages to create a multi-panel figure.

```
#Make loop, save plots
library(patchwork)
```

```
Attaching package: 'patchwork'
```

```
The following object is masked from 'package:terra':

    area
```

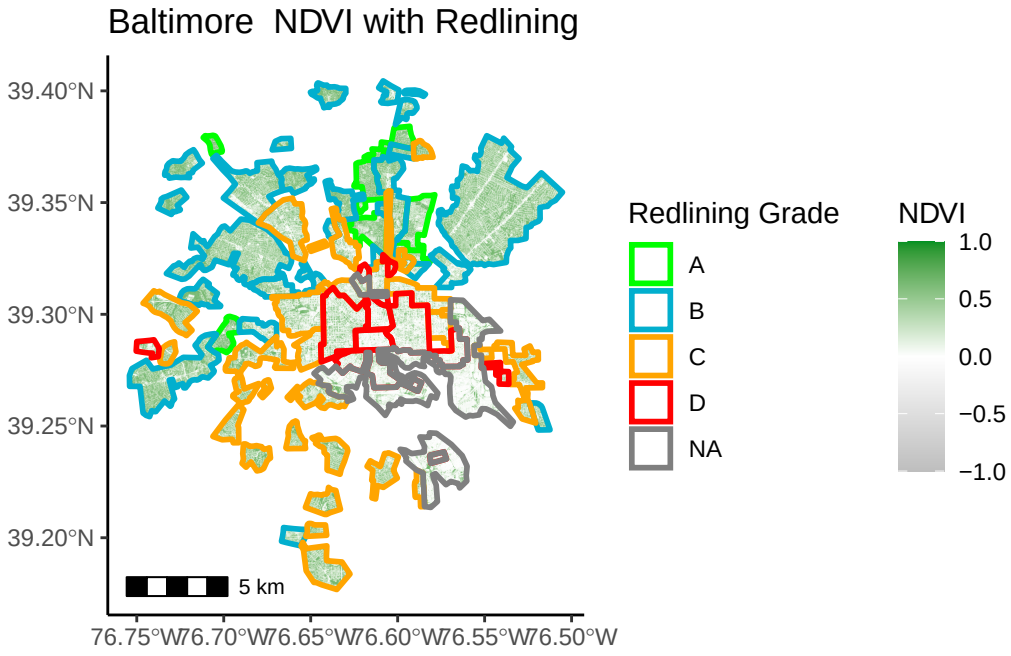
```r
complete_plot <- ggplot()

for (city in c("Baltimore", "Birmingham", "Indianapolis", "Minneapolis", "Phoenix")) {
  #Get NDVI data
  filepath04 <- paste("data", city, "Band04.jp2", sep = "/")
  filepath08 <- paste("data", city, "Band08.jp2", sep = "/")


  #B8 is NIR, B4 is Red

  redBand <- terra::rast(filepath04)
  NIRband <- terra::rast(filepath08)

  NDVI = (NIRband - redBand)/(NIRband + redBand)

  #Get Redlining Data (both poly and st)
  filepathRedlining <- paste("data", city, "geojson.json", sep = "/")
  st_redlining <- st_read(filepathRedlining) |> st_transform(st_crs(NDVI))
  poly_redlining <- terra::project(terra::vect(filepathRedlining), NDVI)
  cutNDVI <- NDVI |> terra::crop(st_redlining)
  polyNDVI <- cutNDVI |> terra::mask(poly_redlining)


  #plot
  plot <- ggplot()+
  geom_spatraster(data = polyNDVI)+
  scale_fill_gradient2(low = "grey",
                       mid = "white",
                       high = "#019123",
                       na.value = NA, limits = c(-1,1)) + theme_classic() +
  labs(title = paste(city, " NDVI with Redlining"), fill = "NDVI", color = "Redlining Grade")
  geom_sf(data = st_redlining, linewidth = 1, fill = NA, mapping = aes(color = grade)) + ther
  scale_color_manual(values = c("A" = "green",
                       "B" =  "#02afce",
                       "C" = "orange",
                       "D" = "red",
                       na.value = "grey")) +
  annotation_scale() +
    theme(legend.direction = "vertical", legend.box = "horizontal")

  complete_plot <- complete_plot + plot
  #also show each plot individually
```

```
  print(plot)
}
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================

Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Balti
  using driver `GeoJSON'
Simple feature collection with 60 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -76.7569 ymin: 39.17875 xmax: -76.50927 ymax: 39.4068
Geodetic CRS:  WGS 84

<SpatRaster> resampled to 500851 cells.
```



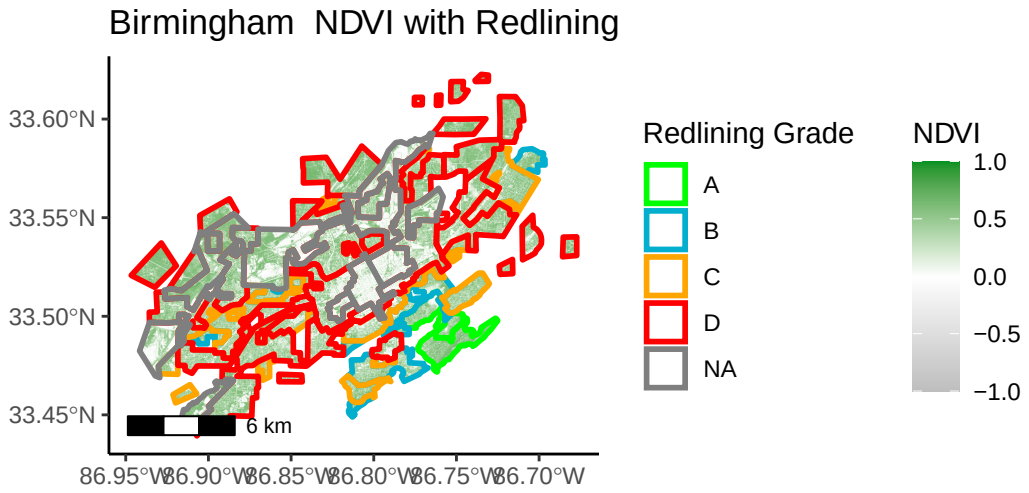

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

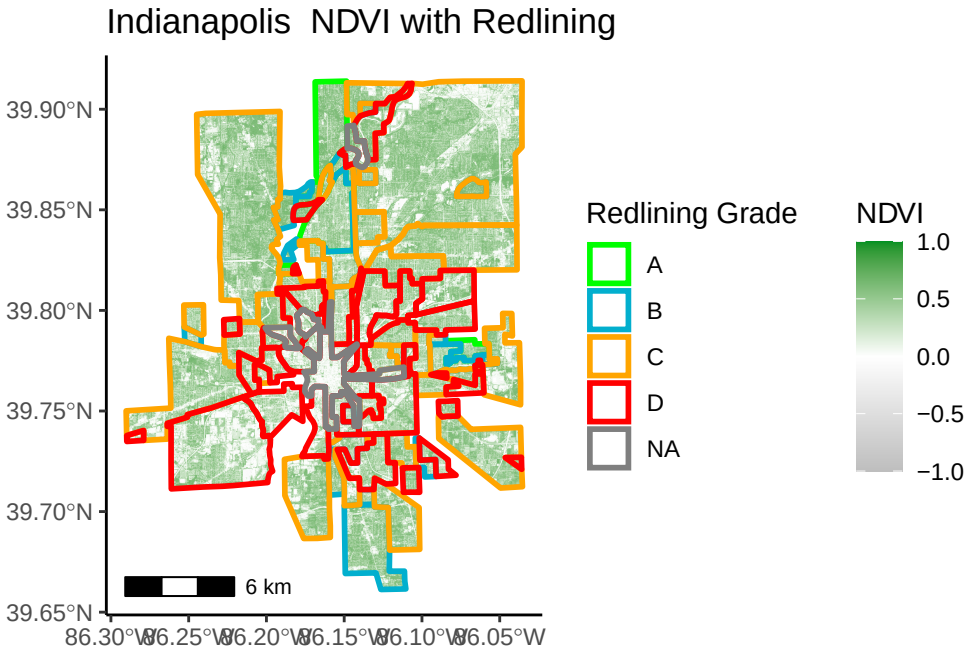
```
|---------|---------|---------|---------|
=========================================
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Birmi
  using driver `GeoJSON'
Simple feature collection with 62 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -86.94663 ymin: 33.4393 xmax: -86.67684 ymax: 33.62239
Geodetic CRS:  WGS 84
```

```
<SpatRaster> resampled to 500830 cells.
```




```
|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================

Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/India
  using driver `GeoJSON'
Simple feature collection with 74 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -86.28916 ymin: 39.66008 xmax: -86.03192 ymax: 39.91294
Geodetic CRS:  WGS 84

<SpatRaster> resampled to 501250 cells.
```



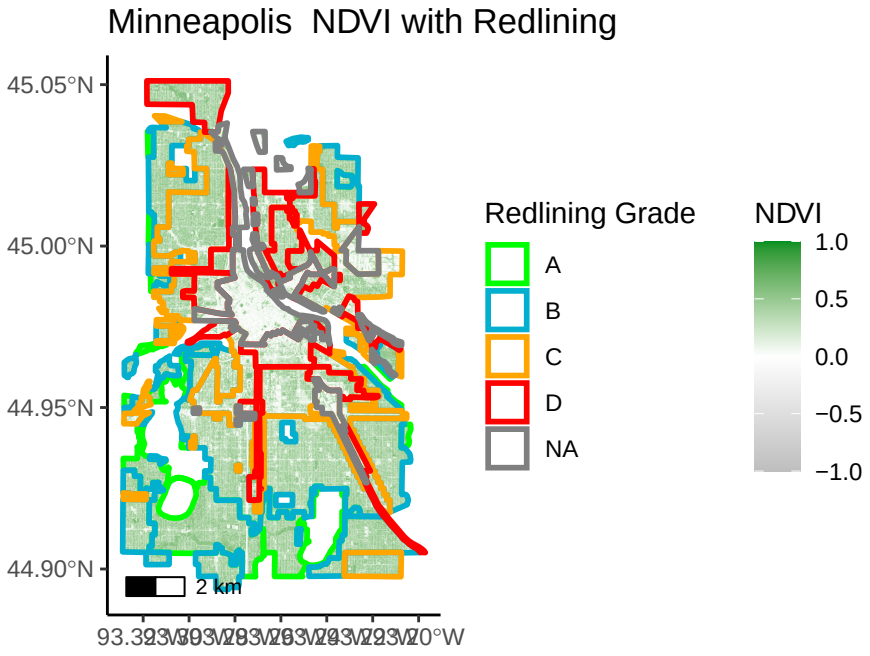

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Minne
  using driver `GeoJSON'
Simple feature collection with 71 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -93.32903 ymin: 44.89379 xmax: -93.19715 ymax: 45.0513
Geodetic CRS:  WGS 84
```

```
<SpatRaster> resampled to 500682 cells.
```



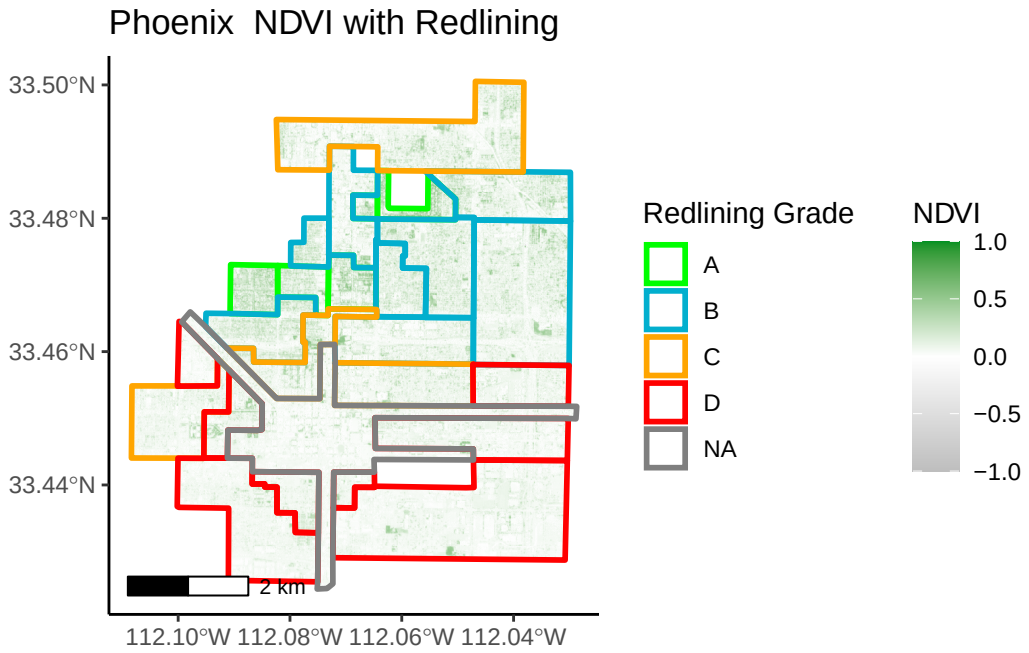

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Phoe
  using driver `GeoJSON'
Simple feature collection with 26 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -112.1087 ymin: 33.42472 xmax: -112.0291 ymax: 33.50111
Geodetic CRS:  WGS 84
```

```
<SpatRaster> resampled to 500565 cells.
```




```
#Plot everything together
# I use the png because it was the only way I could figure out to set the image size
png(filename="Q6plot.png", width=900, height=900)
complete_plot+ plot_layout(ncol = 2, widths = unit(80, "mm"), heights = unit(80, "mm"))
```

Show that png

Figure 1: Plot

Q7 (2 points)

Now, let's examine the relationship between redlining and NDVI. Temporarily re-read in your redlining polygons using `terra:::vect()`. You can use `terra:extract()` on these temporary polygons within `mutate()` on your original polygons read in with `read_sf()`. Because the output of `terra::extract()` is a data.frame, `dplyr::pull()` can be helpful.

Extract the mean, median, and central 95% quantile of NDVI from each delineated neighborhood while retaining the identity of the city. Perform your choice of at least two exploratory data visualizations utilizing different variables to evaluate this relationship and examine whether it differs between cities.

This time I use the redlining CRS because that's common between all the cities

```
#get baltimore polygons
baltimorePoly <- terra::vect("data/Baltimore/geojson.json")
BaltimoreNDVI <- terra::project(BaltimoreNDVI, baltimorePoly)
```

```
|---------|---------|---------|---------|
==========================================
```

```
#add NDVI data to redlining polygons
Baltimore_redlining_NDVI <- st_transform(transformed_baltimore_redlining, st_crs(baltimorePol
```

```
meanNDVI <- rep(0, nrow(baltimorePoly))

medianNDVI <- rep(0, nrow(baltimorePoly))

bottomQuantileNDVI <- rep(0, nrow(baltimorePoly))
topQuantileNDVI <- rep(0, nrow(baltimorePoly))

for (index in 1:nrow(baltimorePoly)) {
  neighborhood_NDVI <- pull(terra::extract(BaltimoreNDVI, baltimorePoly[index,]), Band08)
  meanNDVI[index] <- mean(neighborhood_NDVI)
  medianNDVI[index] <- median(neighborhood_NDVI)
  quantiles <- quantile(neighborhood_NDVI, probs = c(0.025, 0.975), names = FALSE)
  bottomQuantileNDVI[index] <- quantiles[1]
  topQuantileNDVI[index] <- quantiles[2]

}

Baltimore_redlining_NDVI <- Baltimore_redlining_NDVI |>
  mutate("NDVI_mean" = meanNDVI, "NDVI_median" = medianNDVI, "NDVI_2.5_Quantile"= bottomQuant
```

Now do this for all cities:

```r
all_redlining_NDVI <- Baltimore_redlining_NDVI |> mutate("City" = "Baltimore")

for (city in c("Birmingham", "Indianapolis", "Minneapolis", "Phoenix")) {
  #Get NDVI data
  filepath04 <- paste("data", city, "Band04.jp2", sep = "/")
  filepath08 <- paste("data", city, "Band08.jp2", sep = "/")


  #B8 is NIR, B4 is Red

  redBand <- terra::rast(filepath04)
  NIRband <- terra::rast(filepath08)

  NDVI = (NIRband - redBand)/(NIRband + redBand)

  #Get Redlining Data (both poly and st)
  filepathRedlining <- paste("data", city, "geojson.json", sep = "/")
  st_redlining <- st_read(filepathRedlining)
  poly_redlining <- terra::vect(filepathRedlining)

  NDVI <- terra::project(NDVI, poly_redlining)

  #Get NDVI stats by neighborhood
  meanNDVI <- rep(0, nrow(poly_redlining))

  medianNDVI <- rep(0, nrow(poly_redlining))

  bottomQuantileNDVI <- rep(0, nrow(poly_redlining))
  topQuantileNDVI <- rep(0, nrow(poly_redlining))

  for (index in 1:nrow(poly_redlining)) {
    neighborhood_NDVI <- pull(terra::extract(NDVI, poly_redlining[index,]), Band08)
    meanNDVI[index] <- mean(neighborhood_NDVI)
    medianNDVI[index] <- median(neighborhood_NDVI)
    quantiles <- quantile(neighborhood_NDVI, probs = c(0.025, 0.975), names = FALSE)
    bottomQuantileNDVI[index] <- quantiles[1]
    topQuantileNDVI[index] <- quantiles[2]

  }

  city_redlining_NDVI <- st_redlining |>
    mutate("NDVI_mean" = meanNDVI,
```

```
            "NDVI_median" = medianNDVI,
            "NDVI_2.5_Quantile"= bottomQuantileNDVI,
            "NDVI_97.5_Quantile"= topQuantileNDVI,
            "City" = city)
  all_redlining_NDVI <- rbind(all_redlining_NDVI, city_redlining_NDVI)
}
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Birmi
  using driver `GeoJSON'
Simple feature collection with 62 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -86.94663 ymin: 33.4393 xmax: -86.67684 ymax: 33.62239
Geodetic CRS:  WGS 84
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

```
|---------|---------|---------|---------|
=========================================
```

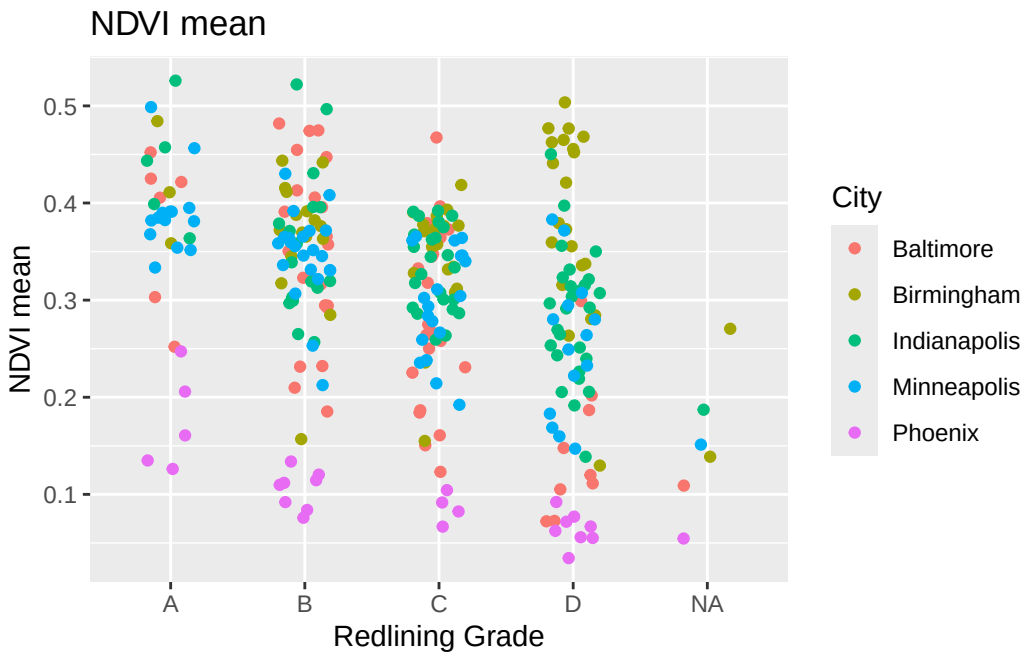
```
|---------|---------|---------|---------|
=========================================
```

```
Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/India
  using driver `GeoJSON'
Simple feature collection with 74 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -86.28916 ymin: 39.66008 xmax: -86.03192 ymax: 39.91294
Geodetic CRS:  WGS 84

|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================

Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Minne
  using driver `GeoJSON'
Simple feature collection with 71 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -93.32903 ymin: 44.89379 xmax: -93.19715 ymax: 45.0513
Geodetic CRS:  WGS 84

|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
=========================================


|---------|---------|---------|---------|
```

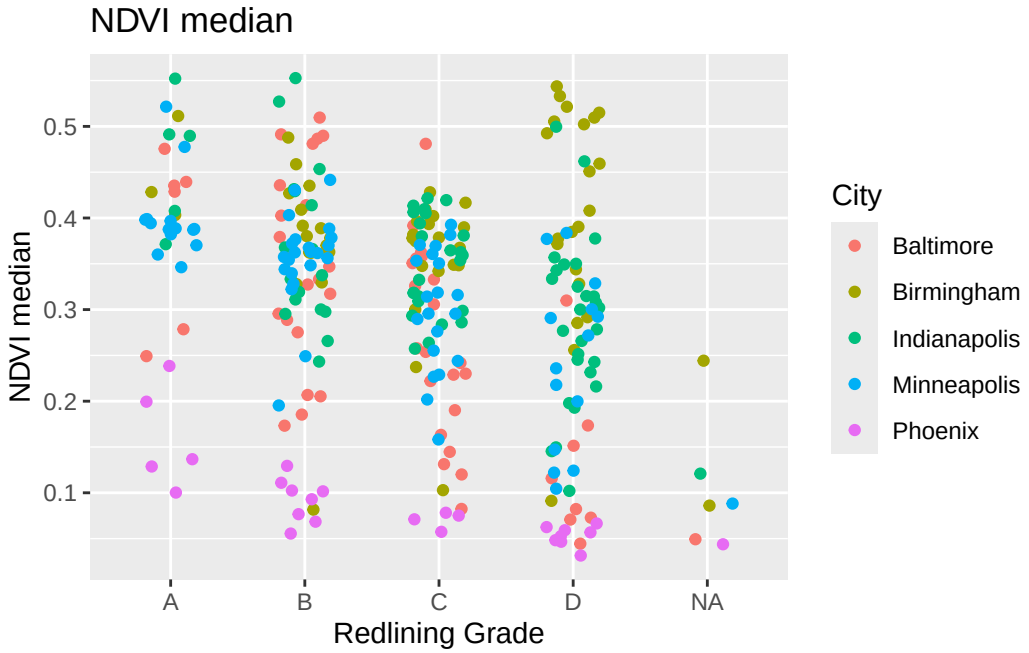
```
=========================================


|---------|---------|---------|---------|
=========================================

Reading layer `geojson' from data source
  `/Users/annafigge/Documents/DataScienceGlobalChangeBio/CLES131-module3-redlining/data/Phoe
  using driver `GeoJSON'
Simple feature collection with 26 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -112.1087 ymin: 33.42472 xmax: -112.0291 ymax: 33.50111
Geodetic CRS:  WGS 84

|---------|---------|---------|---------|
=========================================
```

For each statistic, make a scatter plot with jitter separated by redlining status and color coded by city

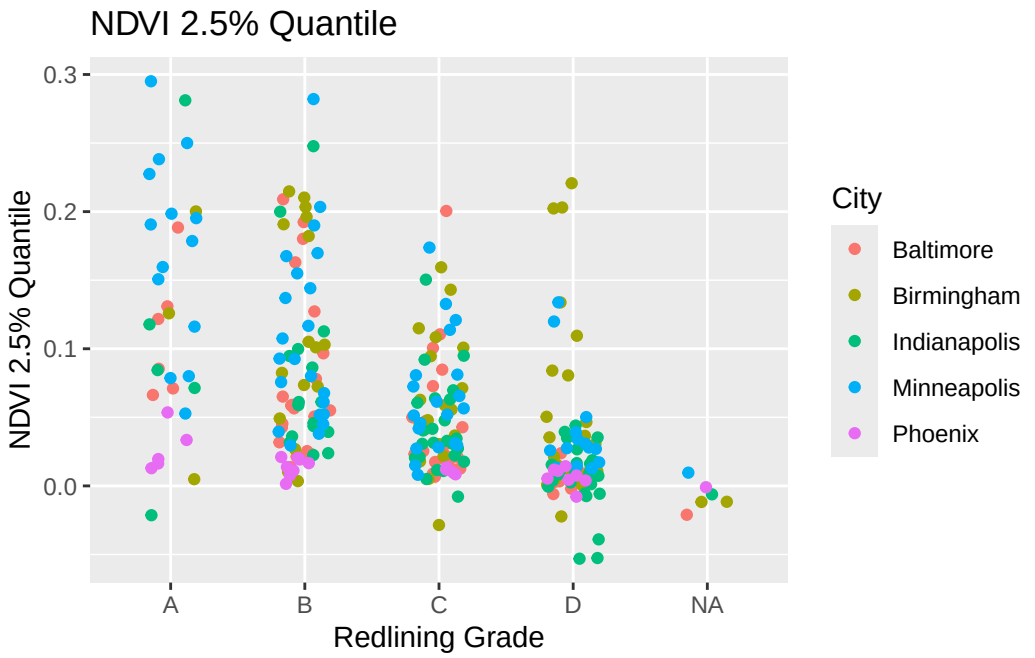
```
ggplot(data = all_redlining_NDVI, aes(x = grade, y = NDVI_mean, color= City)) + geom_jitter(w
  labs(x = "Redlining Grade", y = "NDVI mean", color = "City", title = "NDVI mean")
```



```
ggplot(data = all_redlining_NDVI, aes(x = grade, y = NDVI_median, color= City)) + geom_jitte
  labs(x = "Redlining Grade", y = "NDVI median", color = "City", title = "NDVI median")
```




```
ggplot(data = all_redlining_NDVI, aes(x = grade, y = NDVI_2.5_Quantile, color= City)) + geom_
  labs(x = "Redlining Grade", y = "NDVI 2.5% Quantile", color = "City", title = "NDVI 2.5% Qu
```




```
ggplot(data = all_redlining_NDVI, aes(x = grade, y = NDVI_97.5_Quantile, color= City)) + geo
  labs(x = "Redlining Grade", y = "NDVI 97.5% Quantile", color = "City", title = "NDVI 97.5%
```

We can see a clear relationship between redlining grade and NDVI. In general, the higher redlining grades have more NDVI, by all the metrics. This result does differ by city somewhat;

for example, Phoenix has lower NDVI everywhere but also a stronger decrease in 97.5% quantile of NDVI from A grade to D grade than any of the other cities, while Minneonapolis has a stronger decrease in 2.5% quantile of NDVI from A grade to D grade than the other cities. Birmingham seems to have the least difference between A grade and D grade in all the NDVI metrics.

Bonus 2 (1 point)

Perform a statistical test to support your visual analysis above.

Q8 (2 points)

Create a final plot and describe whether NDVI is associated with historical redlining. Does this pattern differ between the five cities examined here? If so, how?

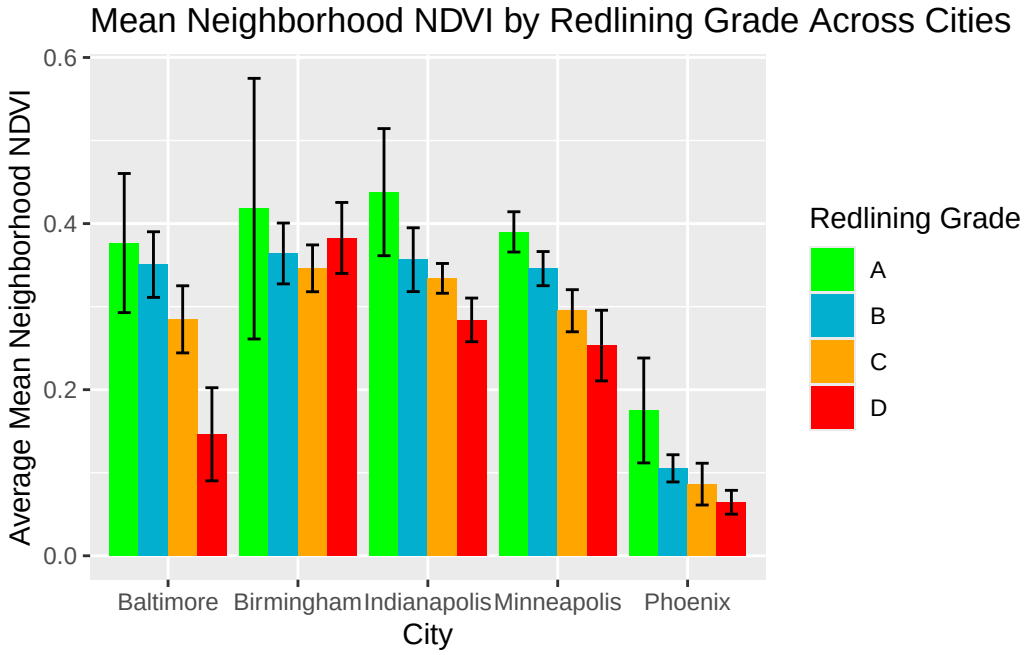
I will plot a bar chart of the average mean NDVI in each city+grade (because I think mean NDVI is the most useful summary statistic). I get and plot 95% confidence intervals on the mean by multiplying the t-score of the appropriate number of degrees of freedom (n - 1) by the standard error (sd/ sqrt(n)). I will not plot the NA grades as I do not believe that is useful for discussing redlinging effects.

```
#store all_redlining_NDVI as a tibble to make transformations easier
tibble_redlining_NDVI <- tibble(all_redlining_NDVI)
# get the means and confidence intervals
to_plot <- tibble_redlining_NDVI |> group_by(City, grade)  |> summarize("NDVIavg" = mean(NDVI
                                                              "ConfintSize" = qt(p=0.975, df =
```

```
Warning: There were 4 warnings in `summarize()`.
The first warning was:
i In argument: `ConfintSize = qt(p = 0.975, df = n() - 1) *
  sd(NDVI_mean)/sqrt(n())`.
i In group 5: `City = "Baltimore"` and `grade = NA`.
Caused by warning in `qt()`:
! NaNs produced
i Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```
`summarise()` has grouped output by 'City'. You can override using the
`.groups` argument.
```

```
#get rid of the NA redlining
to_plot <- to_plot |> filter(!is.na(grade))

#plot (I did some googling to find how to make this kind of plot)
to_plot |> ggplot(aes(x = City, y = NDVIavg, fill = grade)) + geom_bar(position = position_do
          geom_errorbar(position = position_dodge(0.9), aes(ymin=NDVIavg-ConfintSize, ymax=NI
          scale_fill_manual(values = c("A" = "green",
                                       "B" =  "#02afce",
                                       "C" = "orange",
                                       "D" = "red",
                                       na.value = "grey")) +
          labs(y = "Average Mean Neighborhood NDVI", fill = "Redlining Grade", title = "Mean
```



In general, it seems as though historical redlining is associated with current NDVI. In all cities, redlining grade A neighborhoods have the highest average mean NDVI; and in almost all cities redlining grade D neighborhoods have the lowest average mean NDVI. Birmingham is the one exception; there does not seem to be a clear relationship there between redlining and average mean neighborhood NDVI, with all the 95% confidence intervals overlapping. For all cities except Birminham, mean NDVI decreases as redlining grade worsens. The confidence intervals show that we cannot say that B and C grades are significantly different from each other in any city. The pattern is different between cities; as mentioned Birminham does not seem to have any meaningful relationship between redlining grade and mean neighborhood NDVI, but even beyond that we can see that Pheonix has less NDVI all around (in all grades)

than any of the other cities (unsurprisingly) and that Baltimore has a greater decrease (both absolute and relative to the confidence interval size) in mean NDVI between A grade and D grade neighborhoods, and is the only city where the confidence intervals for the average mean neighborhood NDVI in the C grade and D grade neighborhoods do not interact (ie we can say with confidence that the redlining grade D neighborhoods have less mean NDVI than the redlining grade C neighborhoods). All told, this data makes it clear that there is a relationship across many (but not all) cities where historically redlined neighborhoods have lower NDVI today.

Bonus 3 (1 point)

Include the results of your statistical test in the final plot and use prose to incorporate statistical output in the context of the question above.

I don't know whether the confidence intervals (derived by t-test) on the above plot count? Probably not