Module 3: Redlining

```
library(sf) # simple features for R
library(terra) # spatial data analysis
library(tidyterra) #tidyverse methods for terra objects
library(ggplot2)
library(tidyverse)
library(ggspatial)
library(cowplot)
```

Use of GitHub

Link to your forked GH repository: https://github.com/nikhilshah7/CLES131-module3-redlining

Use of Quarto

Link to your .qmd file: https://github.com/nikhilshah7/CLES131-module3-redlining/blob/main/module3.qmd

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?

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.

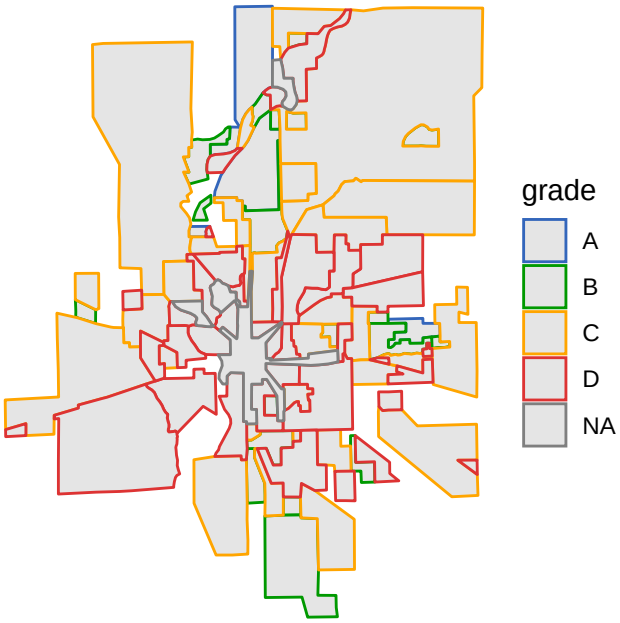
Import the geojson file into your R environment with the `read_sf()`' function from sf. Check the structure of this object and see that it is a special type of data frame, allowing it to be manipulated with many of the functions you already know, including ggplot.

```
indianapolis <- read_sf('data/indianapolis/geojson.json') |>
  mutate(city = 'Indianapolis')
baltimore <- read_sf('data/baltimore/geojson.json') |>
  mutate(city = 'Baltimore')
phoenix <- read_sf('data/phoenix/geojson.json') |>
  mutate(city = 'Phoenix')
birmingham <- read_sf('data/birmingham/geojson.json') |>
  mutate(city = 'Birmingham')
minneapolis <- read_sf('data/minneapolis/geojson.json') |>
  mutate(city = 'Minneapolis')

cities <- indianapolis |>
  bind_rows(baltimore) |>
  bind_rows(phoenix) |>
  bind_rows(birmingham) |>
  bind_rows(minneapolis)
```

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.

```
indianapolis |>
  ggplot() +
  geom_sf(aes(color = grade),
          linewidth = 0.5) +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  theme_void()
```




```
#cities |>
#  ggplot() +
#  geom_sf(aes(color = grade),
#          linewidth = 0.5) +
#  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
#  facet_wrap(~city, scales = 'free') +
#  theme_void()
```

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 avalailable - Metadata is provided in `MTD_MSIL2A.xml`

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

Q3 (1 point)

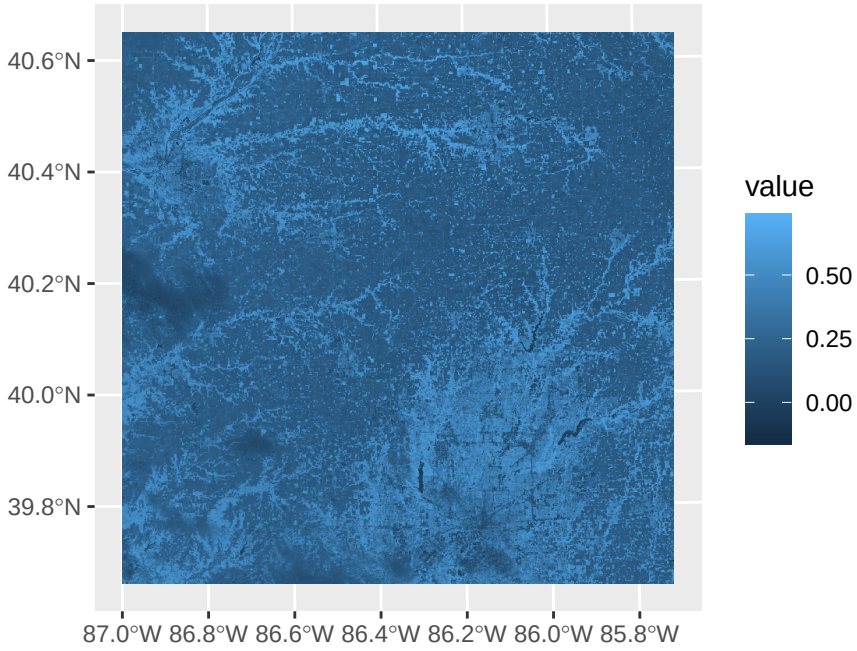
Use the terra package and the `rast()` function to import the two bands, which are reported as digital numbers. Combine to calculate NDVI and display a quick plot of your first city. Since you have 5 cities to plot, how can you optimize these operations with a for loop? (Not necessary to actually do so.)

```
indianapolis_raster_B04 <- rast('data/indianapolis/T16TEK_20250531T162829_B04_60m.jp2')
indianapolis_raster_B08 <- rast('data/indianapolis/T16TEK_20250531T162829_B8A_60m.jp2')

indianapolis_ndvi <- (indianapolis_raster_B08 - indianapolis_raster_B04) / (indianapolis_rast

indianapolis_ndvi |>
  ggplot() +
  geom_spatraster(data = indianapolis_ndvi)
```

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




```
baltimore_raster_B04 <- rast('data/baltimore/T18SUJ_20250603T155001_B04_60m.jp2')
baltimore_raster_B08 <- rast('data/baltimore/T18SUJ_20250603T155001_B8A_60m.jp2')

baltimore_ndvi <- (baltimore_raster_B08 - baltimore_raster_B04) / (baltimore_raster_B08 + bal

phoenix_raster_B04 <- rast('data/phoenix/T12SUC_20250329T181751_B04_60m.jp2')
phoenix_raster_B08 <- rast('data/phoenix/T12SUC_20250329T181751_B8A_60m.jp2')

phoenix_ndvi <- (phoenix_raster_B08 - phoenix_raster_B04) / (phoenix_raster_B08 + phoenix_ras

birmingham_raster_B04 <- rast('data/birmingham/T16SEC_20250521T162829_B04_60m.jp2')
birmingham_raster_B08 <- rast('data/birmingham/T16SEC_20250521T162829_B8A_60m.jp2')

birmingham_ndvi <- (birmingham_raster_B08 - birmingham_raster_B04) / (birmingham_raster_B08 -

minneapolis_raster_B04 <- rast('data/minneapolis/T15TVK_20250714T165921_B04_60m.jp2')
minneapolis_raster_B08 <- rast('data/minneapolis/T15TVK_20250714T165921_B8A_60m.jp2')

minneapolis_ndvi <- (minneapolis_raster_B08 - minneapolis_raster_B04) / (minneapolis_raster_
```

Q4 (1 point)

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

```
#st_crs(baltimore)
#st_crs(baltimore_raster_B04)

indianapolis_ndvi_4326 <- project(indianapolis_ndvi, 'epsg:4326')
baltimore_ndvi_4326 <- project(baltimore_ndvi, 'epsg:4326')
phoenix_ndvi_4326 <- project(phoenix_ndvi, 'epsg:4326')
birmingham_ndvi_4326 <- project(birmingham_ndvi, 'epsg:4326')
minneapolis_ndvi_4326 <- project(minneapolis_ndvi, 'epsg:4326')
```

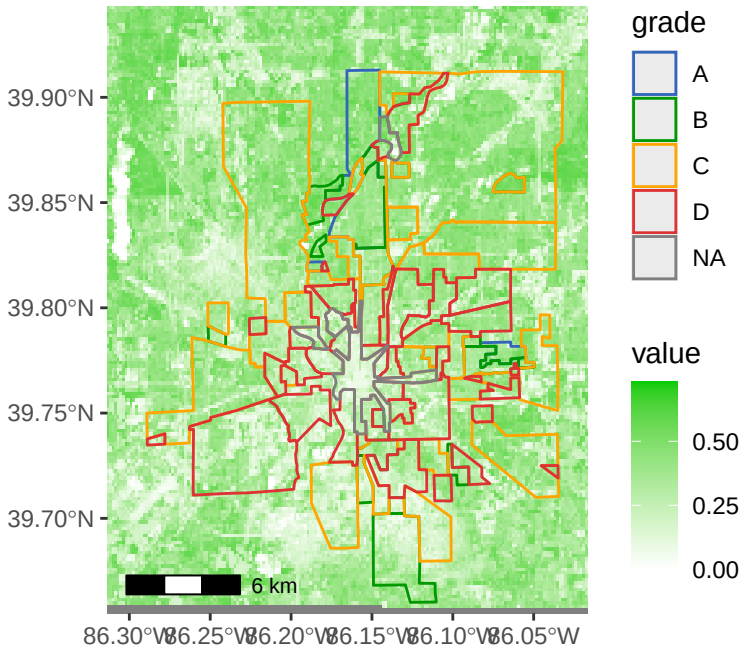
- They are not, the vector data uses a WGS84 GCS and the raster data uses a WGS84 PCS. I am putting both into the geographic coordinate system, because the PCS distorts angles and offsets the maps from being aligned, which is unintuitive and doesn't match Schell et al.

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.

```
ggplot() +
  geom_spatraster(data = indianapolis_ndvi_4326) +
  geom_sf(data = indianapolis,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-86.3, -86.03), ylim = c(39.93, 39.668)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333"))
```

```
<SpatRaster> resampled to 500960 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.

```
indianapolis_plot <- ggplot() +
  geom_spatraster(data = indianapolis_ndvi_4326) +
  geom_sf(data = indianapolis,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-86.3, -86.03), ylim = c(39.93, 39.668)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  labs(title = 'Indianapolis',
       color = 'Redlining',
       fill = 'NDVI Index') +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
```

```
    axis.line = element_blank(),
    legend.box = "horizontal"
  )
```

```
<SpatRaster> resampled to 500960 cells.
```

```
baltimore_plot <- ggplot() +
  geom_spatraster(data = baltimore_ndvi_4326) +
  geom_sf(data = baltimore,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-76.76, -76.5), ylim = c(39.175, 39.41)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  labs(title = 'Baltimore') +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.line = element_blank()
  )
```

```
<SpatRaster> resampled to 500448 cells.
```

```
phoenix_plot <- ggplot() +
  geom_spatraster(data = phoenix_ndvi_4326) +
  geom_sf(data = phoenix,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-112.11, -112.02), ylim = c(33.42, 33.5)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  labs(title = 'Phoenix') +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
```

```
    axis.line = element_blank()
  )
```

```
<SpatRaster> resampled to 500650 cells.
```

```
birmingham_plot <- ggplot() +
  geom_spatraster(data =  birmingham_ndvi_4326) +
  geom_sf(data = birmingham,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-86.95, -86.67), ylim = c(33.42, 33.65)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  labs(title = 'Birmingham') +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.line = element_blank()
  )
```

```
<SpatRaster> resampled to 500650 cells.
```

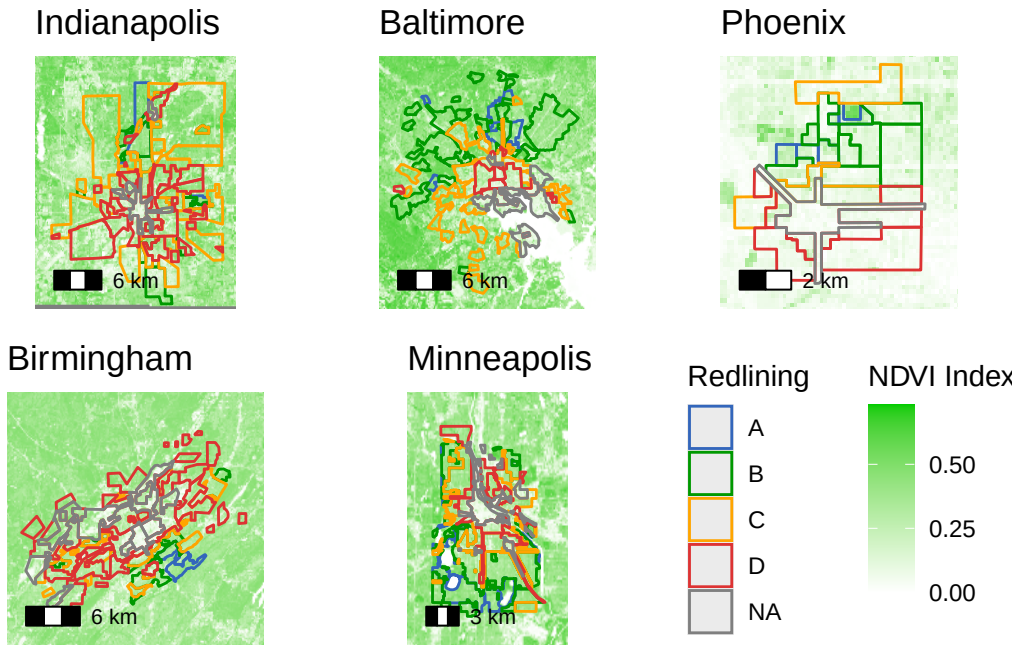
```
minneapolis_plot <- ggplot() +
  geom_spatraster(data = minneapolis_ndvi_4326) +
  geom_sf(data = minneapolis,
          aes(color = grade),
          linewidth = 0.5,
          fill = 'transparent') +
  coord_sf(xlim = c(-93.35, -93.18), ylim = c(44.88, 45.07)) +
  annotation_scale() +
  scale_fill_gradient2(low = '#ffffff', high = '#00cc00') +
  scale_color_manual(values = c("#3366bb", "#009900", "orange", "#dd3333")) +
  labs(title = 'Minneapolis') +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    axis.line = element_blank()
  )
```

```
<SpatRaster> resampled to 500286 cells.
```

```
legend <- get_legend(indianapolis_plot + theme(legend.box.margin = margin(0, 0, 0, 12)))
```

```
Warning in get_plot_component(plot, "guide-box"): Multiple components found;
returning the first one. To return all, use `return_all = TRUE`.
```

```
plot_grid(indianapolis_plot + theme(legend.position="none"),
          baltimore_plot + theme(legend.position="none"),
          phoenix_plot + theme(legend.position="none"),
          birmingham_plot + theme(legend.position="none"),
          minneapolis_plot + theme(legend.position="none"),
          legend)
```



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.

Bonus 1 (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?

Bonus 2 (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.