Masri et al. (2020) Pb Levels and Census Tracts

Thi Truong

Introduction

"Despite efforts to coalesce around the use of these terms, lack of consensus persists across disciplines. The resulting confusion is an obstacle in moving forward to improve reproducibility and replicability" (Barba et al., 2018; NASM, 2019)

This is a Quarto document to show processing supplementary data in order to **reproduce**, to my best ability, the Pb concentration heat map plots from Masri et al. (2020). Why reproduce the data? Because it is an important component of science, and I am unable to **replicate** the data because I do not have the resources to collect thousands of natural samples and run Pb concentrations on them. However, I am also limited in a 1:1 reproduction, because I do not have the model for data interpolation and k-fitting, which is used in the well-known figure, which itself has some interesting quirks:

Load Packages

I did not use all of these packages, but I tend to load these for most of my data processing:

```r
library(tidyverse)
library(dplyr) #data functions
library(readr) #for reading csv files
#library(knitr) #makes cool tables
#library(stringr) #string extraction functions

## Visualization
library(ggplot2)
library(mapview)
library(maps) # for city names
```

```r
## Interactive viz
library(plotly)
library(leaflet)

## Census
library(tigris)
```

Load Data

Tigris Census Tracts Data

Experimenting with Tigris package. Tigris simplifies the process for R users of obtaining and using Census geographic datasets.

`states()` function can be run without arguments to download a boundary file of US states and state equivalents. We get a message letting us know that data for the year 2021 are returned; tigris typically defaults to the most recent year for which a complete set of Census shapefiles are available, which at the time of this writing is 2021.

Plotting California counties.

Plotting Orange County, CA tracts

City Names Data

This is to get city names in California. This is performed in two steps,.

```r
## Data on 1005 US cities
data(us.cities)
```

Once the city data is loaded (check global environment to make sure there are rows), save specific city names from California, and remove redundant information.

```r
cities_data <- us.cities |>
  filter(country.etc %in% "CA") |>
# This will drop the state name from the name column
  mutate(fixed_name = str_replace(name,country.etc,''))
```

Masri et al. (2020) Data

Supplementary data tables from Masri et al. (2020). I renamed the columns already for easier data processing. I confess I did this in Excel because it is simply easier to do this in a WYSIWYG editor.

There are two data tables:

1. `table_SI.csv` is table 1. Store the raw data as `masri_table_SI_raw`. We are using this now.
2. `table_SII.csv` is table 2. Store the raw data as `masri_table_SII_raw`. I am not processing that today, but FYI, it contains zip code average data, which appears to be an average of census tract data…

```
masri_table_SI_raw <- read_csv("table_SI.csv", col_names = TRUE)

masri_table_SII_raw <- read_csv("table_SII.csv", col_names = TRUE)
```

We are working with the first table today. It has 61 rows and 4 columns (census tract, number of samples analyzed, Pb (ppm) averaged, and cumulative risk index. Each row has data for each census tract. See below:

```
print(masri_table_SI_raw)
```

```
# A tibble: 61 x 4
   census_tract_number n_number pb_ppm_average cumulative_risk_index
                 <dbl>    <dbl>          <dbl>                 <dbl>
 1                750.       18           611.                  1   
 2                750.       27           356.                  1   
 3                749.       20           129.                  1   
 4                749.       31           373.                  0.97
 5                750.       35           169.                  0.94
 6                744.       40           130.                  0.92
 7                746.       62           217.                  0.89
 8                744.       40           145.                  0.89
 9                748.       27           136.                  0.81
10                748.       42           134.                  0.81
# i 51 more rows
```

Now, the next code chunk is processing this raw data `masri_table_SI_raw` into a processed data tibble named `masri_census_lead`. The changes I am making are adding (mutating) additional columns:

1. Census tract ID, or `census_ID`. Note that the order is straight from the supplementary data and I am unsure of the logic of the original sorting.
2. Adding a categorical level, from 1-10, called `pb_ppm_average_level`, which will match the color scheme of the original Masri et al. (2020) figure. E.g., 1 is the lowest level (values less than 42.4 ppm) and 10 is the highest (values above 229.6 ppm).
3. Census tract number `NAME` that matches the Tigris census tract column. This is stored as character instead of number to also match the Tigris data.

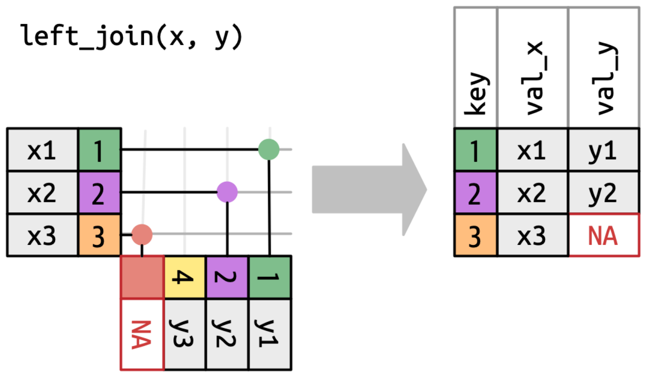
```
masri_census_lead <- masri_table_SI_raw |>
  mutate(census_ID = row_number(),
         pb_ppm_average_level = case_when(
           pb_ppm_average >= 21.8 & pb_ppm_average < 42.4 ~ '1',
           pb_ppm_average >= 42.4 & pb_ppm_average < 52.7 ~ '2',
           pb_ppm_average >= 52.7 & pb_ppm_average < 61.2 ~ '3',
           pb_ppm_average >= 61.2 & pb_ppm_average < 69.9 ~ '4',
           pb_ppm_average >= 69.9 & pb_ppm_average < 81.4 ~ '5',
           pb_ppm_average >= 81.4 & pb_ppm_average < 91.3 ~ '6',
           pb_ppm_average >= 91.3 & pb_ppm_average < 105.2 ~ '7',
           pb_ppm_average >= 105.2 & pb_ppm_average < 142.0 ~ '8',
           pb_ppm_average >= 142.0 & pb_ppm_average < 229.6 ~ '9',
           pb_ppm_average >= 229.6 ~ '10',
           TRUE ~ 'NA'
           ),
         NAME = as.character(census_tract_number)
         )
```

Finally, for some reason, one census tract just kept getting stored as just "741.1" instead of "741.10" because of some rounding issue, which ends up messing the join later. So, I have to add a "0" at the end to this one value (which is the last one, or number 61):

```
masri_census_lead$NAME[61] <- "741.10"
```

Perform data join

To be able to plot the Masri et al. (2020) data with census tracts, I am performing a left join. A left join is visualized in the figure below:



In my case, the left table (x) is the Masri et al. (2020) data and the right is the California census data. This means I only keep matches from the census data, and drop any unmatched columns (basically, anything that isn't in the Orange County area of focus).

```r
masri_census_join <- masri_census_lead |>
  left_join(OC_tracts, by = "NAME")
```

Print out the column names of the joined data, and we see that there are far more than the original 7 columns. All of the column names in all capital letters are from the census data. The final column, geometry, has the polygons or shape data to be able to map the census tracts.

```r
colnames(masri_census_join)
```

```
 [1] "census_tract_number"   "n_number"               "pb_ppm_average"
 [4] "cumulative_risk_index" "census_ID"              "pb_ppm_average_level"
 [7] "NAME"                  "STATEFP"                "COUNTYFP"
[10] "TRACTCE"               "GEOID"                  "NAMELSAD"
[13] "MTFCC"                 "FUNCSTAT"               "ALAND"
[16] "AWATER"                "INTPTLAT"               "INTPTLON"
[19] "geometry"
```

Note that city names do not have to be joined here, they can be called upon when making each plot.

Make Plots

First, I am going to get the bounding box information to get the limits/coordinates for any plots i make (xmin, ymin, xmax, ymax), where x values are longitude and y values are latitude.

According to some tutorials, I should be able to use the `st_bbox` function to get these values, but I cannot find it! :(

```r
masri_census_join$geometry
```

```
Geometry set for 61 features  (with 1 geometry empty)
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -117.9549 ymin: 33.69178 xmax: -117.8074 ymax: 33.78819
Geodetic CRS:  NAD83
First 5 geometries:

MULTIPOLYGON (((-117.8676 33.75431, -117.8676 3...

MULTIPOLYGON (((-117.8764 33.75474, -117.8764 3...

MULTIPOLYGON (((-117.8851 33.73995, -117.885 33...

MULTIPOLYGON (((-117.8851 33.74794, -117.885 33...

MULTIPOLYGON (((-117.8765 33.74796, -117.8765 3...
```

After running this, I get this result (copied and pasted the bounding box result row).

```
xmin: -117.9549 ymin: 33.69178 xmax: -117.8074 ymax: 33.78819
```

I will store these to make plotting easier:

```r
xmin <- -117.9549
xmax <- -117.8074
ymin <- 33.69178
ymax <-  33.78819
```

Second, I am going to make the color scheme for the target plot (Figure at the top). The color scale from Masri et al. (2020) figure is described in this color scale (breaks, labels, names). These were interpolated Pb concentrations, but here they represent the range of averaged data.

```r
# To match the factor levels in the data that i made
masri_pb_breaks <- c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10")

# For color fill of census tracts
masri_pb_colors <- c("10" = "#DB0018",
                     "9" = "#D55C23",
                     "8" = "#F18133",
                     "7" = "#F5AC46",
                     "6" = "#F7E74C",
                     "5" = "#E3ED6E",
                     "4" = "#BFD389",
                     "3" = "#9ABDA1",
                     "2" = "#73A7B5",
                     "1" = "#4993CC")

# The interpolated Pb levels from Masri et al. (2020)
masri_pb_labels <- c("21.8-42.4",
                     "42.4-52.7",
                     "52.7-61.2",
                     "61.2-69.9",
                     "69.9-81.4",
                     "81.4-91.3",
                     "91.3-105.1",
                     "105.2-142.0",
                     "142.0-229.6",
                     ">229.6")
```

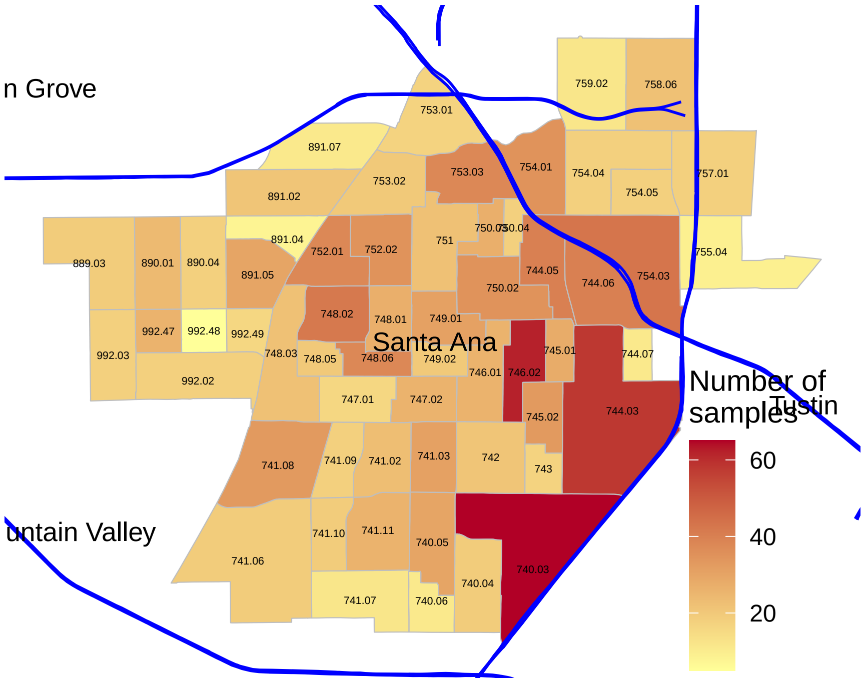
For each plot, a lot of this information seems redundant and makes for long code chunks, but not going to try to store the info in a single chunk to recall because I'm not sure if that's possible for ggplot (or, it would require some knowledge about order and sequence that is easy to forget).

Number of samples collected per census tract

This is using the number of samples as the fill. The gradient is defined by two colors: light yellow for low, and dark red for high.

```r
# Save with unique name
gg_masri_number_samples <-

# Base join data in ggplot
```

```r
  ggplot(data = masri_census_join) +

# Census tract borders with grey borders
# Fill color depending on number of samples collected
  geom_sf(color = "grey",
          aes(geometry = geometry, fill = n_number)) +

# Custom gradient palette
  scale_fill_gradient(low = '#ffff99',
                      high = '#b10026',
                      na.value = 'white' )+
# Add roads in blue
  geom_sf(data = CA_roads, color = 'blue',
          aes(geometry = geometry)) +

# City Names
  geom_text(color = 'black',
            data = cities_data, check_overlap = TRUE, size = 3.5,
            aes(x = long, y = lat, label = fixed_name)) +

# Census Tract Names
  geom_sf_text(size = 1.5,
               aes(geometry = geometry, label = NAME)) +

# Minimal theme gets rid of x and y-axis labels and grid
  theme_void() +

# White background so it is not transparent
  theme(panel.background = element_rect(fill = 'white', color = NA)) +

# Coordinate limits, defined earlier with bounding box
  coord_sf(xlim = c(xmin, xmax), ylim = c(ymin, ymax)) +

# Label legend
  labs(fill = 'Number of \nsamples') +

# Move legend to bottom right
  theme(legend.justification = c(0,0), legend.position = c(0.8,0.01))

# Print plot to preview
gg_masri_number_samples
```



```
# Save plot (commented until I make changes to save)
#ggsave('figures/gg_masri_number_samples.png', dpi=400, units='in')
#ggsave('figures/gg_masri_number_samples.png')
```

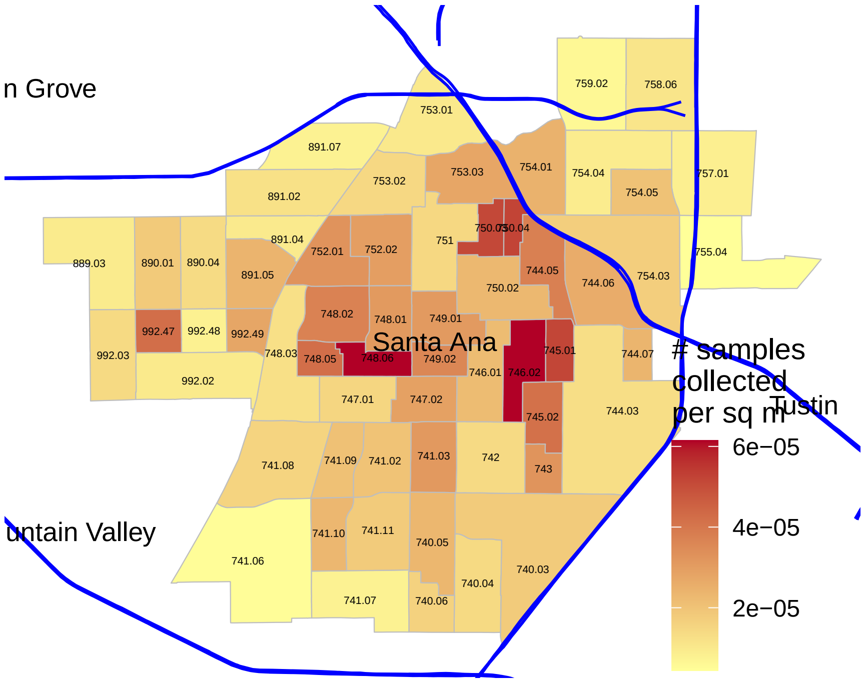
Sample density: Number of samples collected per square meter per census tract

`n_number` divided by `ALAND` value in the census tracts dataset can be used to visualize samples collected per square meter.

```
gg_masri_number_samples_per_m_sq <-

# Base join data
  ggplot(masri_census_join) +

# Census tract borders with grey borders
# Fill color for number of samples divided by land area in square meters
  geom_sf(color = "grey",
          aes(geometry = geometry, fill = n_number/ALAND)) + #census tract borders

# Custom gradient palette
  scale_fill_gradient(low = '#ffff99',
                      high = '#b10026',
                      na.value = 'white' )  + # custom fill palette

# Add roads in blue
  geom_sf(data = CA_roads, color = 'blue',
          aes(geometry = geometry)) +

# City Names
  geom_text(color = 'black', data = cities_data, check_overlap = TRUE, size = 3.5,
            aes(x = long, y = lat, label = fixed_name)) +

# Census Tract Names
  geom_sf_text(size = 1.5,
               aes(geometry = geometry, label = NAME)) +

# Minimal theme gets rid of x and y-axis labels and grid
  theme_void() +

# White background so it is not transparent
  theme(panel.background = element_rect(fill = 'white', color = NA)) +

# Coordinate limits, defined earlier with bounding box
  coord_sf(xlim = c(xmin, xmax), ylim = c(ymin, ymax)) +

# Label legend
  labs(fill = '# samples \ncollected  \nper sq m') +

# Move legend to bottom right
  theme(legend.justification = c(0,0), legend.position = c(0.78,0.01))

# Print plot to preview
gg_masri_number_samples_per_m_sq
```

```
Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
give correct results for longitude/latitude data
```

```
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_text()`).
```



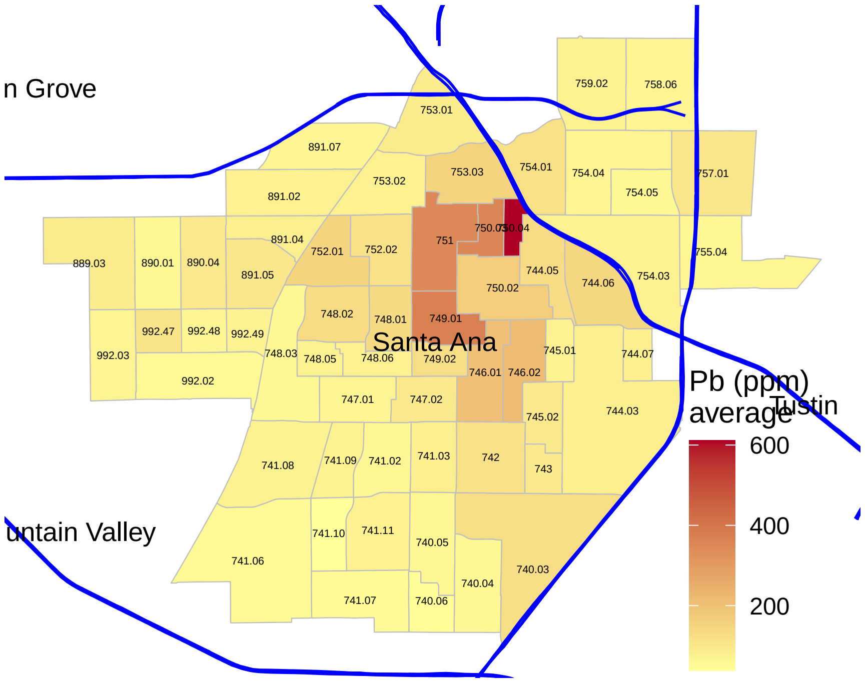
```
# Save plot (commented until I make changes to save)
#ggsave('figures/gg_masri_number_samples_per_m_sq.png', dpi=400, units='in')
#ggsave('figures/gg_masri_number_samples_per_m_sq.png')
```

Pb average (ppm) concentrations

Using a very basic gradient color scale.

```
gg_masri_pb_average <-

# Base join data
  ggplot(masri_census_join) +

# Census tract borders with grey borders
# Fill color Pb average concentrations
  geom_sf(color = "grey",
          aes(geometry = geometry, fill = pb_ppm_average)) +

# Custom gradient palette
  scale_fill_gradient(low = '#ffff99',
                      high = '#b10026',
                      na.value = 'white' )  + # custom fill palette
```

```
# Add roads in blue
  geom_sf(data = CA_roads, color = 'blue',
          aes(geometry = geometry)) +

# City Names
  geom_text(color = 'black', data = cities_data, check_overlap = TRUE, size = 3.5,
            aes(x = long, y = lat, label = fixed_name)) +

# Census Tract Names
  geom_sf_text(size = 1.5,
               aes(geometry = geometry, label = NAME)) +

# Minimal theme gets rid of x and y-axis labels and grid
  theme_void() +

# White background so it is not transparent
  theme(panel.background = element_rect(fill = 'white', color = NA)) +

# Coordinate limits, defined earlier with bounding box
  coord_sf(xlim = c(xmin, xmax), ylim = c(ymin, ymax)) +

# Label legend
  labs(fill = 'Pb (ppm) \naverage') +

# Move legend to bottom right
  theme(legend.justification = c(0,0), legend.position = c(0.80,0.01))

# Print plot to preview
gg_masri_pb_average
```

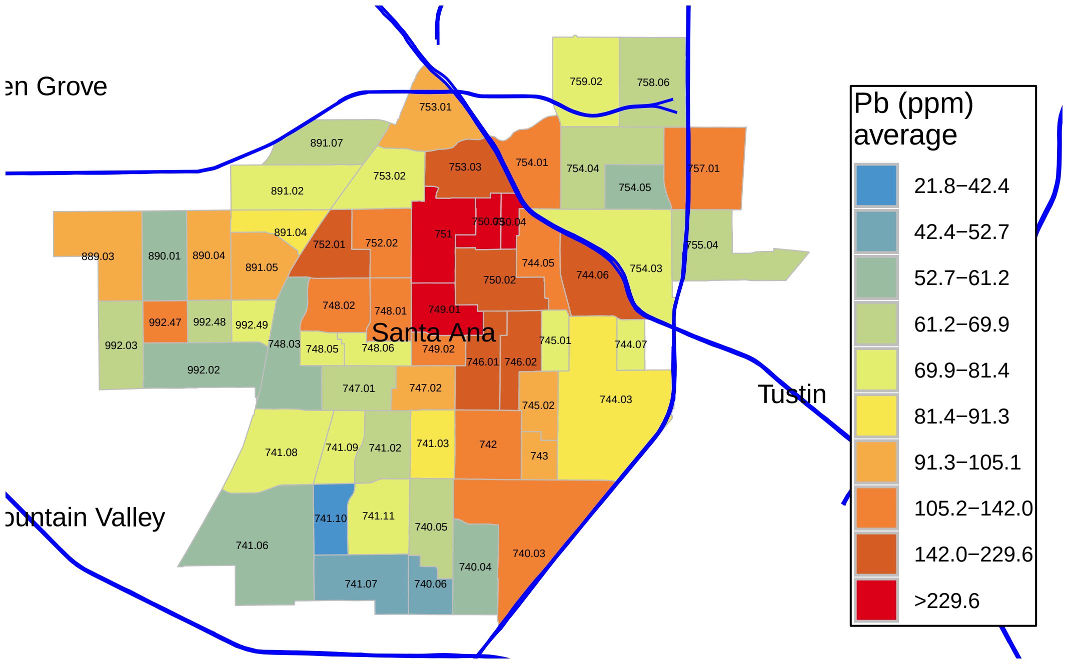


```
# Save plot (commented until I make changes to save)
#ggsave('figures/gg_masri_pb_average.png', dpi=400, units='in')
#ggsave('figures/gg_masri_pb_average.png')
```

Pb average (ppm) concentrations with original Masri color scale

```
gg_masri_pb_average_OG_colors <-

# Base join data
  ggplot(masri_census_join) +

# Census tract borders with grey borders
# Fill color Pb average concentrations
  geom_sf(color = "grey",
          aes(geometry = geometry, fill = pb_ppm_average_level)) +

# Add roads in blue
  geom_sf(data = CA_roads, color = 'blue', aes(geometry = geometry)) +

# City Names
  geom_text(color = 'black', data = cities_data, check_overlap = TRUE, size = 3.5,
```

```r
            aes(x = long, y = lat, label = fixed_name)) +

# Census Tract Names
  geom_sf_text(size = 1.5,
               aes(geometry = geometry, label = NAME)) +

# Minimal theme gets rid of x and y-axis labels and grid
  theme_void() +

# White background so it is not transparent
  theme(panel.background = element_rect(fill = 'white', color = NA)) +

# Coordinate limits plus more x-axis to make room for the legend
    coord_sf(xlim = c(xmin, xmax+0.04), ylim = c(ymin, ymax)) +

# Label legend
  labs(fill = 'Pb (ppm) \naverage') + #label legend

# Custom color palette, legend labels
  scale_fill_manual(breaks = masri_pb_breaks,
                    values = masri_pb_colors,
                    labels = masri_pb_labels) +

# Move legend to bottom right
  theme(legend.justification = c(0,0), legend.position = c(0.80,0.05),
        legend.background = element_rect(fill = "white"),
        legend.margin = margin(1, 1, 1, 1, "pt"),
        legend.text = element_text(size = rel(0.75)))

# Print plot to preview
gg_masri_pb_average_OG_colors
```



```
# Save plot (commented until I make changes to save)
#ggsave('figures/gg_masri_pb_average_OG_colors.png', dpi=400, units='in')
#ggsave('figures/gg_masri_pb_average_OG_colors.png')
```

Example interactive plots

Still in development!

```
#plot(OC_block_groups$geometry)

#leaflet(OC_tracts) %>%
#   addTiles() %>%
#   addPolygons(popup = ~NAME)

#mapview(OC_tracts)
```

This was ran in RStudio and documented on a Quarto (.qmd) file. Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Version information:

> RStudio 2024.04.2+764 "Chocolate Cosmos" Release (e4392fc9ddc21961fd1d0efd47484b43f07a4177, 2024-06-05) for windows Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) RStudio/2024.04.2+764 Chrome/120.0.6099.291 Electron/28.3.1 Safari/537.36, Quarto 1.4.555

Further Reading and Sources:

- Barba, L. A. (2018). Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311*. https://arxiv.org/abs/1802.03311
- Masri, S., LeBrón, A., Logue, M., Valencia, E., Ruiz, A., Reyes, A., Lawrence, J.M., & Wu, J. (2020). Social and spatial distribution of soil lead concentrations in the City of Santa Ana, California: Implications for health inequities. *Science of the Total Environment*,*743*, 140764. https://doi.org/10.1016/j.scitotenv.2020.140764
- National Academies of Sciences, Policy, Global Affairs, Board on Research Data, Information, Division on Engineering, … & Replicability in Science. (2019). *Reproducibility and Replicability in Science*. National Academies Press. https://www.ncbi.nlm.nih.gov/books/NBK547546/