Calculating Demographic Indexes for the Casco Bay Watershed

July 27, 2022

Contents

Introduction

CBEP, like other National Estuary Programs will receive additional funding to support our programs via the "Bipartisan Infrastructure Law" signed into law last December.

EPA has recently released guidance for applying for those funds. A core component of the guidance is that overall, the NEP program should comply with the White House's "Justice 40" initiative, which requires that "at least 40% of the benefits and investments from BIL funding flow to disadvantaged communities."

EPA suggested that we use the National-scale EJSCREEN tools to help identify "disadvantaged communities" in our region. The EPA guidance goes on to suggest we focus on five demographic indicators:

- Percent low-income;
- Percent linguistically isolated;
- Percent less than high school education;
- Percent unemployed; and
- Low life expectancy.

This notebook builds on the work in "Calc_Indexes.pdf" to calculate data for the Casco Bay Watershed Census Tracts, and calculates how Casco Bay Census tracts compare at National, Statewide, and Regional scales.

Load Libraries

```
library(tidyverse)
#> -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
#> v ggplot2 3.3.6     v purrr   0.3.4
#> v tibble  3.1.7     v dplyr   1.0.9
#> v tidyr   1.2.0     v stringr 1.4.0
#> v readr   2.1.2     v forcats 0.5.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
library(GGally)
#> Registered S3 method overwritten by 'GGally':
#>   method from
#>   +.gg   ggplot2
library(readr)
```

Set Graphics Theme

This sets `ggplot()`graphics for no background, no grid lines, etc. in a clean format suitable for (some) publications.

```
theme_set(theme_classic())
```

Load Data

Folder References

I use folder references to allow limited indirection, thus making code from GitHub repositories more likely to run "out of the box".

```
data_folder <- "Original_Data"
dir.create(file.path(getwd(), 'figures'), showWarnings = FALSE)
```

I use the "Original_Data" folder to retain data in the form originally downloaded. That minimizes the chances of inadvertently modifying the source data. All data was accessed via EJScreen. The 2021 EJSCREEN Data was accessed on July 26, 2022, at https://gaftp.epa.gov/EJSCREEN/2021/. I downloaded geodatabases, and open the geospatial data they contained in ArcGIS and exported the tabular attribute data to CSV files. That tabular CSV data is provided in the "Original Data" folder here.

The "figures" folder isolates "final" versions of any graphics I produce. That just makes it a bit easier to find final products in what can sometimes be fairly large GitHub Repositories (although not here).

Load Data

The Tabular data is quite extensive (over 165 MB), which poses some data access challenges. The raw CSV file contains 74001 records, and 166 columns. Most, but not all are numeric. The Health data is slightly smaller in length, and has only a handful of relevant data columns, but it DOES include State ad County names, which are more convenient that the GEOID10 values to search.

```
the_file <- "Casco_Tracts.txt"
the_path <- file.path(data_folder, the_file)

cb_data <- read_csv(the_path, n_max = 81,
                    col_types = c(paste(c(rep('-',2),  'c-',  rep('d',2),
                                          rep('-',7), 'd'), collapse = '')))

the_file <- "National_Draft_Indexes.csv"
the_data <- read_csv(the_file,
                     n_max = 74001,
                     col_types = c(rep(col_character(),3),
                                   rep(col_double(), 23)))
```

```
cb_data <- cb_data %>%
  inner_join(the_data, by = 'GEOID10')
```

Save Data

```
write_csv(cb_data, "cb_tracts_indexes.csv")
```

Calculate Thresholds

We have six different indexes, and we want threshold values for each at National, State, and Casco Bay Region levels. It's convenient to automate the calculations using a small function and the `map()` function.

```
(nms <- names(cb_data)[23:28])
#> [1] "Index_1"      "Index_2"      "P_Index_1"    "P_Index_2"    "PCA_Index_V1"
#> [6] "PCA_Index_V2"
```

Utility Function

This function calculates the 80th percentile (by default, anyway) of a named data column from a data frame. There is no error checking, so this is NOT appropriate for programming with out more work.

```
quantile_select_col <- function(.data, .col, .q = 0.8) {
  return(quantile(.data[,.col], .q, na.rm = TRUE))
}

the_data %>%
  quantile_select_col( "Index_1")
#>      80%
#> 30.99741
```

Calculations

I calculate a named vector of threshold values at NAtional, State and Casco Bay Regional levels.

```
National <- map(nms, function(x) quantile_select_col(the_data, x))
National <- unlist(National) # Flatten List to numeric vector
names(National) <- nms       # Add Names
National
#>      Index_1      Index_2    P_Index_1    P_Index_2 PCA_Index_V1 PCA_Index_V2
#>    30.997409    70.839103     0.800003     0.800003    71.304144   157.020785
```

```
Maine <- map(nms,
             function(x) quantile_select_col(the_data[the_data$State == 'Maine',], x))
Maine <- unlist(Maine) # Flatten List to numeric vector
names(Maine) <- nms       # Add Names
Maine
#>      Index_1      Index_2    P_Index_1    P_Index_2 PCA_Index_V1 PCA_Index_V2
#>   26.6214141   55.5550440    0.6352263    0.5798520   61.3592947  126.1009802
```

```
Region <- map(nms, function(x) quantile_select_col(cb_data, x))
Region <- unlist(Region) # Flatten List to numeric vector
names(Region) <- nms       # Add Names
Region
#>      Index_1      Index_2    P_Index_1    P_Index_2 PCA_Index_V1 PCA_Index_V2
#>   22.6837646   43.7415479    0.4168774    0.3937596   51.6064856   99.3023085
```

```
thresholds <- bind_rows(National, Maine, Region, .id = 'Scale') %>%
  mutate(Scale = c('National', 'Maine', 'Region'))
```

Calculate ThresholdExceedences

I use a similar functional programming approach for calculating whether specific Casco Bay Census Tracts exceed each threshold. Here I pass both a dataframe and a names list or vector containing the thresholds.

Utility Function

```
threshold_compare <- function(.data, .thresholds, .col) {
  return(.data[.col] > .thresholds[.col])
}
```

```
National["Index_1"]
#>  Index_1
#> 30.99741
```

And we can now quickly demonstrate that no Casco Bay census Block exceeds that value.

```
sum(threshold_compare(cb_data, National, "Index_1"), na.rm = TRUE)
#> [1] 0
```

And not =ne exceed the 80th percentile for Maine either.

```
sum(threshold_compare(cb_data, Maine, "Index_1"), na.rm = TRUE)
#> [1] 0
```

Calculations

```
National_Exceeds <- map(as.list(nms),
                        function(x) threshold_compare(cb_data, National, x))
National_Exceeds <- as.data.frame(National_Exceeds)
names(National_Exceeds) <- paste0('Ex_Nat_', nms)
unlist(map(National_Exceeds, sum, na.rm = TRUE))
#>      Ex_Nat_Index_1      Ex_Nat_Index_2    Ex_Nat_P_Index_1    Ex_Nat_P_Index_2
#>                   0                   0                   0                   0
#> Ex_Nat_PCA_Index_V1 Ex_Nat_PCA_Index_V2
#>                   0                   0
```

```
Maine_Exceeds <- map(as.list(nms),
                     function(x) threshold_compare(cb_data, Maine, x))
Maine_Exceeds <- as.data.frame(Maine_Exceeds)
names(Maine_Exceeds) <- paste0('Ex_Maine_', nms)
unlist(map(Maine_Exceeds, sum, na.rm = TRUE))
#>      Ex_Maine_Index_1      Ex_Maine_Index_2    Ex_Maine_P_Index_1
#>                     0                     0                     0
#>    Ex_Maine_P_Index_2 Ex_Maine_PCA_Index_V1 Ex_Maine_PCA_Index_V2
#>                     0                     0                     0
```

test <- map(as.list(nms), function(x) threshold_compare(cb_data, Region, x)) names(test) <- nms test[[1]][1:5] test <- as.data.frame(test) test

```
CB_Exceeds <- map(as.list(nms),
                  function(x) threshold_compare(cb_data, Region, x))
CB_Exceeds <- as.data.frame(CB_Exceeds)
names(CB_Exceeds) <- paste0('Ex_CB_', nms)
unlist(map(CB_Exceeds, sum, na.rm = TRUE))
#>      Ex_CB_Index_1      Ex_CB_Index_2    Ex_CB_P_Index_1    Ex_CB_P_Index_2
#>                 14                 14                 14                 14
#> Ex_CB_PCA_Index_V1 Ex_CB_PCA_Index_V2
#>                 14                 14
```

Combine Data

```
cb_data <- cb_data %>%
  bind_cols(National_Exceeds, Maine_Exceeds, CB_Exceeds)
```

I don't save that data, because it is actually easier to manage map colors without the hard thresholds.