

# Calculating Demographic Indexes for the Casco Bay Watershed

July 27, 2022

## Contents

<b>Introduction</b>	<b>1</b>
<b>Load Libraries</b>	<b>2</b>
<b>Set Graphics Theme</b>	<b>2</b>
<b>Load Data</b>	<b>2</b>
Folder References . . . . .	2
Load Data . . . . .	2
<b>Calculate Thresholds</b>	<b>3</b>
Calculate Thresholds . . . . .	4
<b>Combine Data</b>	<b>8</b>

## 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 and County names, which are more convenient than 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), rep('c',2), rep('d',2),
                                         rep('-',6), '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))) %>%
  mutate(PCA_Index_V1 = -PCA_Index_V1,
         PCA_Index_V2 = -PCA_Index_V2)

cb_data <- cb_data %>%
  inner_join(the_data, by = 'GE0ID10')

```

## 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)[25:30])
#> [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 without 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    70.201615    0.800003    0.800003    71.304144    1.545725
```

```
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    54.8230773    0.6352263    0.5794647    61.3592947    1.2463658
```

```
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    41.4129108    0.4168774    0.3662750    51.6064856    0.9755620
```

## Calculate Thresholds

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.

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

But at least one sample does exceed the 80th percentile of the regional thresholds.

```
threshold_compare(cb_data, Region, "Index_1")
#>      Index_1
#> [1,] FALSE
#> [2,] FALSE
#> [3,] TRUE
#> [4,] TRUE
#> [5,] FALSE
#> [6,] FALSE
#> [7,] TRUE
```

```
#> [8,] TRUE
#> [9,] TRUE
#> [10,] FALSE
#> [11,] TRUE
#> [12,] FALSE
#> [13,] FALSE
#> [14,] FALSE
#> [15,] FALSE
#> [16,] FALSE
#> [17,] FALSE
#> [18,] FALSE
#> [19,] FALSE
#> [20,] FALSE
#> [21,] FALSE
#> [22,] FALSE
#> [23,] FALSE
#> [24,] FALSE
#> [25,] TRUE
#> [26,] TRUE
#> [27,] FALSE
#> [28,] FALSE
#> [29,] FALSE
#> [30,] FALSE
#> [31,] FALSE
#> [32,] FALSE
#> [33,] FALSE
#> [34,] FALSE
#> [35,] FALSE
#> [36,] FALSE
#> [37,] FALSE
#> [38,] FALSE
#> [39,] TRUE
#> [40,] TRUE
#> [41,] TRUE
#> [42,] FALSE
#> [43,] FALSE
#> [44,] FALSE
#> [45,] FALSE
#> [46,] FALSE
#> [47,] TRUE
#> [48,] FALSE
#> [49,] FALSE
#> [50,] FALSE
#> [51,] FALSE
#> [52,] FALSE
#> [53,] FALSE
#> [54,] FALSE
#> [55,] FALSE
#> [56,] FALSE
#> [57,] FALSE
#> [58,] FALSE
#> [59,] TRUE
#> [60,] FALSE
```

```

#> [61,] FALSE
#> [62,] FALSE
#> [63,] FALSE
#> [64,] FALSE
#> [65,] FALSE
#> [66,] FALSE
#> [67,] TRUE
#> [68,] FALSE
#> [69,] NA
#> [70,] NA
#> [71,] NA
#> [72,] NA
#> [73,] NA
#> [74,] NA
#> [75,] NA
#> [76,] NA
#> [77,] NA
#> [78,] NA
#> [79,] NA
#> [80,] NA
#> [81,] NA

```

```

test <- map(as.list(nms), function(x) threshold_compare(cb_data, Region, x))
test <- as.data.frame(test)
test

```

```

#>   Index_1 Index_2 p_Index_1 p_Index_2 PCA_Index_V1 PCA_Index_V2
#> 1  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 2  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 3   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 4   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 5  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 6  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 7   TRUE  FALSE   TRUE  FALSE   TRUE  FALSE
#> 8   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 9   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 10 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 11  TRUE  FALSE   TRUE  FALSE  FALSE  FALSE
#> 12 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 13 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 14 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 15 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 16 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 17 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 18 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 19 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 20 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 21 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 22 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 23 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
#> 24 FALSE   TRUE  FALSE   TRUE  FALSE   TRUE
#> 25  TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 26  TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
#> 27 FALSE  FALSE  FALSE  FALSE  FALSE  FALSE

```

#> 28	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 29	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 30	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 31	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 32	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 33	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 34	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 35	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
#> 36	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 37	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 38	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 39	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
#> 40	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
#> 41	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
#> 42	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 43	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 44	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 45	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 46	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 47	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
#> 48	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 49	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 50	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 51	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 52	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 53	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 54	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 55	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 56	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 57	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 58	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 59	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
#> 60	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 61	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 62	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
#> 63	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 64	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 65	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 66	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
#> 67	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
#> 68	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
#> 69	NA	NA	NA	NA	NA	NA
#> 70	NA	NA	NA	NA	NA	NA
#> 71	NA	NA	NA	NA	NA	NA
#> 72	NA	NA	NA	NA	NA	NA
#> 73	NA	NA	NA	NA	NA	NA
#> 74	NA	NA	NA	NA	NA	NA
#> 75	NA	NA	NA	NA	NA	NA
#> 76	NA	NA	NA	NA	NA	NA
#> 77	NA	NA	NA	NA	NA	NA
#> 78	NA	NA	NA	NA	NA	NA
#> 79	NA	NA	NA	NA	NA	NA
#> 80	NA	NA	NA	NA	NA	NA

```
#> 81      NA      NA      NA      NA      NA      NA
```

## 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              2              0
#>      Ex_Maine_p_Index_2 Ex_Maine_PCA_Index_V1 Ex_Maine_PCA_Index_V2
#>              2              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)
```

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