Calculating Composite Demographic Indexes

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 examines the distributions of EPA's suggested demographic indicators and calculates relevant composite indexes a couple of different ways, and calculates how Casco Bay Census tracts compare at national, Statewide, and Local 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 (National) source data is quite extensive (over 100 MB), so I have not included it in the GitHub repository (GitHub does not appreciate files over 100 MB). The large files also poses potential data access challenges in R.

I read just the required data columns for now.

```
the_file <- 'EJSCREEN_Full.csv'
the_path <- file.path(data_folder, the_file)
the_data <- read_csv(the_path, n_max = 220333,
                     col_types = cols_only(
                       ID = col_character(),
                       #ID_CHAR = col_character(),
                       #ID_CHAR_2 = col_character(),
                       STATE_NAME = col_character(),
                       ST_ABBREV = col_character(),
                       MINORPCT = col_double(),
                       LOWINCPCT = col_double(),
                       LESSHSPCT  = col_double(),
                       LINGISOPCT = col_double(),
                       UNEMPPCT = col_double(),
                       P_MINORPCT = col_double(),
                       P_LWINCPCT = col_double(),
                       P_LNGISPCT = col_double(),
                       P_LESHSPCT  = col_double(),
                       P_UNEMPPCT = col_double()))
```

Check for values loaded inadvertently as abbreviations in scientific notation.

```
any(grepl('E', the_data$ID))
#> [1] FALSE
```

```
the_file <- 'Tract2010_LifeExpectancy.csv'
the_path <- file.path(data_folder, the_file)
life_data <- read_csv(the_path,
                      col_types = cols_only(GEOID10 = col_character(),
                                            State = col_character(),
                                            LIFEEXP = col_double(),
                                            P_LIFEEXP = col_double(),
                                            Life_Expectancy_Standard_Error = col_double())) %>%
  select (-State)
```

Merge Data

The Life Expectancy data is only available at the Tract level, so to incorporate that into the data, we need to merge the data, but the ID for the block group has one more digit that the GEOID at the Tract level.

```
the_data %>%
  filter(STATE_NAME == 'Maine') %>%
  mutate(tract_geoid10 =  substr(ID, 1, 11)) %>%
  select(ID, tract_geoid10)
#> # A tibble: 1,086 x 2
#>    ID           tract_geoid10
#>    <chr>        <chr>
#>  1 230010101001 23001010100
#>  2 230010101002 23001010100
#>  3 230010102001 23001010200
#>  4 230010102002 23001010200
#>  5 230010102003 23001010200
#>  6 230010103001 23001010300
#>  7 230010103002 23001010300
#>  8 230010104001 23001010400
#>  9 230010104002 23001010400
#> 10 230010105001 23001010500
#> # ... with 1,076 more rows
```

The match rate is not great. But there are missing values in the life expectancy data.

```
the_data_2 <- the_data %>%
  mutate(tract_geoid10 =  substr(ID, 1, 11)) %>%
  left_join(life_data, by = c('tract_geoid10' = 'GEOID10')) # %>%
  #select(-tract_geoid10)


cat('Percentage Unmatched\n')
#> Percentage Unmatched
round(sum(is.na(the_data_2$LIFEEXP))/length(the_data$ID)*100,2)
#> [1] 23.97
cat('\nPercentage Matched\n')
#>
#> Percentage Matched
round(sum(! is.na(the_data_2$LIFEEXP))/length(the_data$ID)*100,2)
#> [1] 76.03
```

```
the_data <- the_data_2 %>%
  rename(LIFEEXP_SE = Life_Expectancy_Standard_Error) %>%
  mutate(NEG_LIFEEXP = 150 - LIFEEXP,             # Higher life expectancy is good
         LOWINCPCT = 100* LOWINCPCT,
         LESSHSPCT = 100* LESSHSPCT,
         LINGISOPCT =  100* LINGISOPCT,
         UNEMPPCT   = 100* UNEMPPCT) %>%
  relocate(STATE_NAME, ST_ABBREV,  .after = ID) %>%
  relocate(LIFEEXP, .after = MINORPCT) %>%
  relocate(P_LIFEEXP, .after = P_MINORPCT) %>%
  relocate(NEG_LIFEEXP, .after = LIFEEXP_SE)

rm(the_data_2, life_data)
```

Functions for Calculating Indexes

The first indexis calculated based on raw scores. It tends to emphasize income, which has higher variance than the ofther metrics.

```
calc_index_raw <- function(.data) {
  index_1 <- with(.data,
    (NEG_LIFEEXP + LOWINCPCT + LESSHSPCT + LINGISOPCT + UNEMPPCT) / 5)
  return(index_1)
}
```

The primary alternative is to calculate an average of the percentiles.That makes the composite index approximately scale-free in each sub-index.

```
calc_index_pcntiles_5 <- function(.data) {
  index_2 <- with(.data,
                  (P_LIFEEXP + P_LWINCPCT + P_LESHSPCT +
                     P_LNGISPCT +
                     P_UNEMPPCT) / 5)
  return(index_2)
}
```

Unfortunately, we are missing data on life expectancy at many locations, so it is convenient to calculate an alternative index that omits that variable.

```
calc_index_pcntiles_4 <- function(.data) {
  index_3 <- with(.data,
                  (P_LWINCPCT + P_LESHSPCT +
                     P_LNGISPCT +
                     P_UNEMPPCT) / 4)
  return(index_3)
}
```

Or perhaps we use all five metrics when available, but only four when life expectancy data is missing.

```
calc_index_pcntiles_best <- function(.data) {
  index_4 <- with(.data,
                  if_else(is.na(P_LIFEEXP),
                          (P_LWINCPCT + P_LESHSPCT +
                             P_LNGISPCT +
                             P_UNEMPPCT) / 4,
                          (P_LIFEEXP + P_LWINCPCT + P_LESHSPCT +
                             P_LNGISPCT +
                             P_UNEMPPCT) / 5))
  return(index_4)
}
```

More Calculations

The following depends on having columns with the correct names, and there is no error checking. . . .

- Index_Raw is based on averaging the VALUES
- Index_5_Ptiles is based on averaging the PERCENTILES
- Index_4_Ptiles omits life expectancy
- Index_best_Ptiles averaged all five metrics if life expectancy is available, otherwise averaged s the other four.

```
the_data$Index_Raw <- calc_index_raw(the_data)
the_data$Index_5_Ptiles <- calc_index_pcntiles_5(the_data)
the_data$Index_4_Ptiles <- calc_index_pcntiles_4(the_data)
the_data$Index_best_Ptiles <- calc_index_pcntiles_best(the_data)
```

Distributions

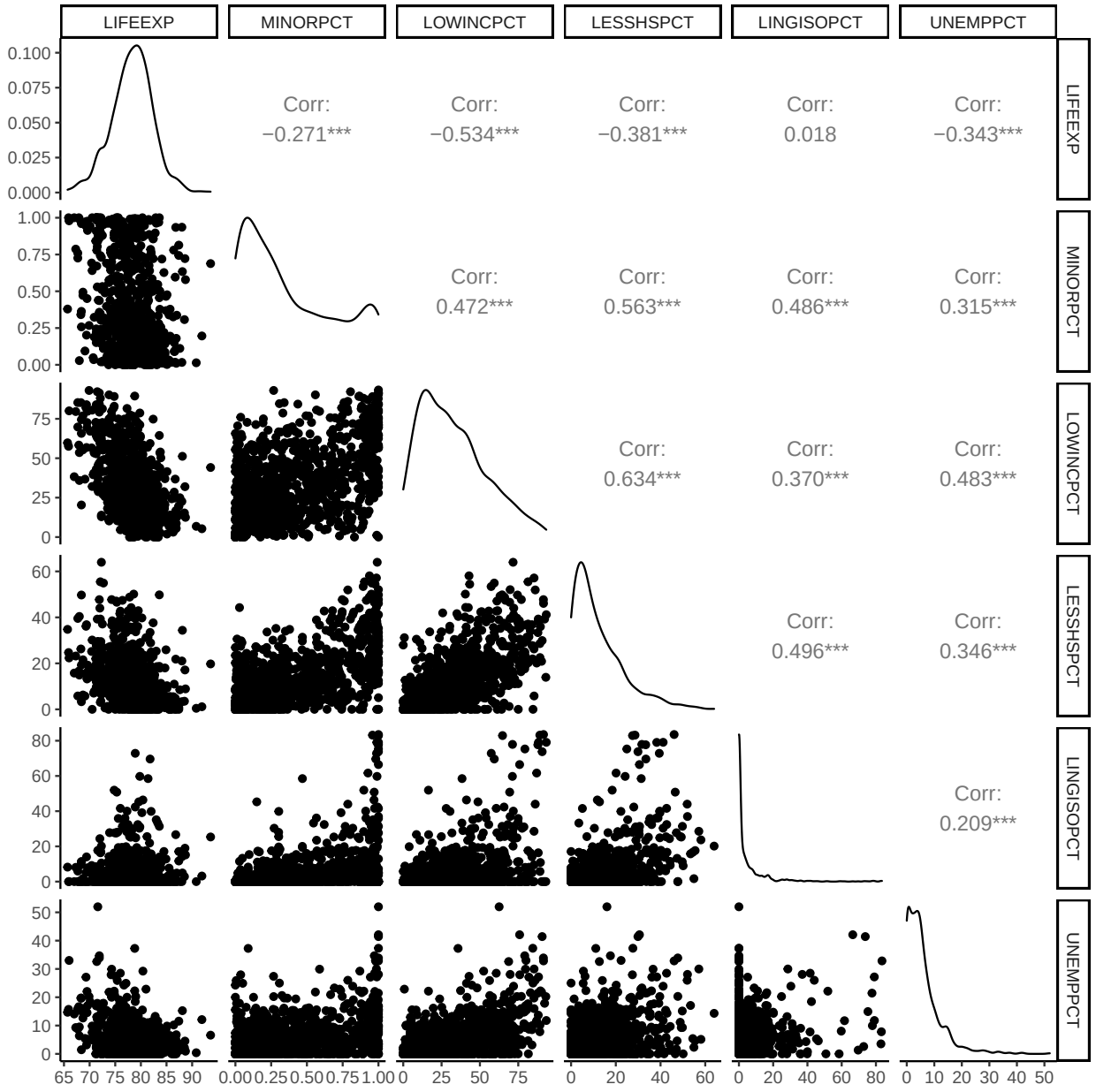
Metrics

Pairs Plot

`GGPairs` runs slowly because of the amount of data involved. In addition, this graphic ends up taking a huge amount of space in the final PDF. We reduce plot complexity by plotting only a a 0.5% sample of the data. That makes no difference for qualitative assessmsnt of a data set as large as this one.

```
the_data %>%
  select("LIFEEXP", "MINORPCT", "LOWINCPCT", "LESSHSPCT", "LINGISOPCT","UNEMPPCT" ) %>%
  slice_sample(prop = 0.005, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 255 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 255 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
```

```
#> Removed 255 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 255 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 255 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 255 rows containing missing values
#> Warning: Removed 255 rows containing missing values (geom_point).
#> Removed 255 rows containing missing values (geom_point).
#> Removed 255 rows containing missing values (geom_point).
#> Removed 255 rows containing missing values (geom_point).
#> Removed 255 rows containing missing values (geom_point).
```



Data (except life expectancy) is not normally distributed, especially for those sub-indexes that have mostly low values. That is not unexpected for percents. which are bounded below by zero, and are a transformation of count data.

Adding or averaging raw values will lead to indexes dominated by the sub-indexes with the largest variance. For some analyses, I would consider data transformations, but that will not be needed here, since I will work with percentiles instead.

Means, SD, Medians and IQR

```
quick_sum <- function(.dat)
  return(list(Mean = mean(.dat, na.rm = TRUE),
```

```
              SD = sd(.dat, na.rm = TRUE),
              Median = median(.dat, na.rm = TRUE),
              IQR = IQR(.dat, na.rm = TRUE)
              ))
```

```
the_data %>%
  select( "LIFEEXP", "MINORPCT", "LOWINCPCT", "LESSHSPCT",
          "LINGISOPCT","UNEMPPCT" ) %>%
  map(quick_sum) %>%
  unlist() %>%
  array(dim = c(4,5),
        dimnames = list(c('Mean', 'SD', 'Median', 'IQR'),
                        c("NEG_LIFEEXP", "LOWINCPCT", "LESSHSPCT",
                          "LINGISOPCT","UNEMPPCT" )))
#>        NEG_LIFEEXP LOWINCPCT LESSHSPCT LINGISOPCT  UNEMPPCT
#> Mean      78.12814 0.3798798  32.89986  12.736512  5.134305
#> SD         3.96802 0.3187029  21.38511  11.913489 11.010335
#> Median    78.30000 0.2842270  29.45455   9.269663  0.000000
#> IQR        5.20000 0.5267171  31.17712  13.740259  5.313351
```

Utility Function Simply adding these metrics together (as in "Index_Raw") will, roughly speaking, end up with an index that will emphasize poverty about twice as much as the lack of high school education and about four times as much as the other indicators. Moderate to high correlations among predictors will affect that somewhat, but the general idea is sound.

Correlations

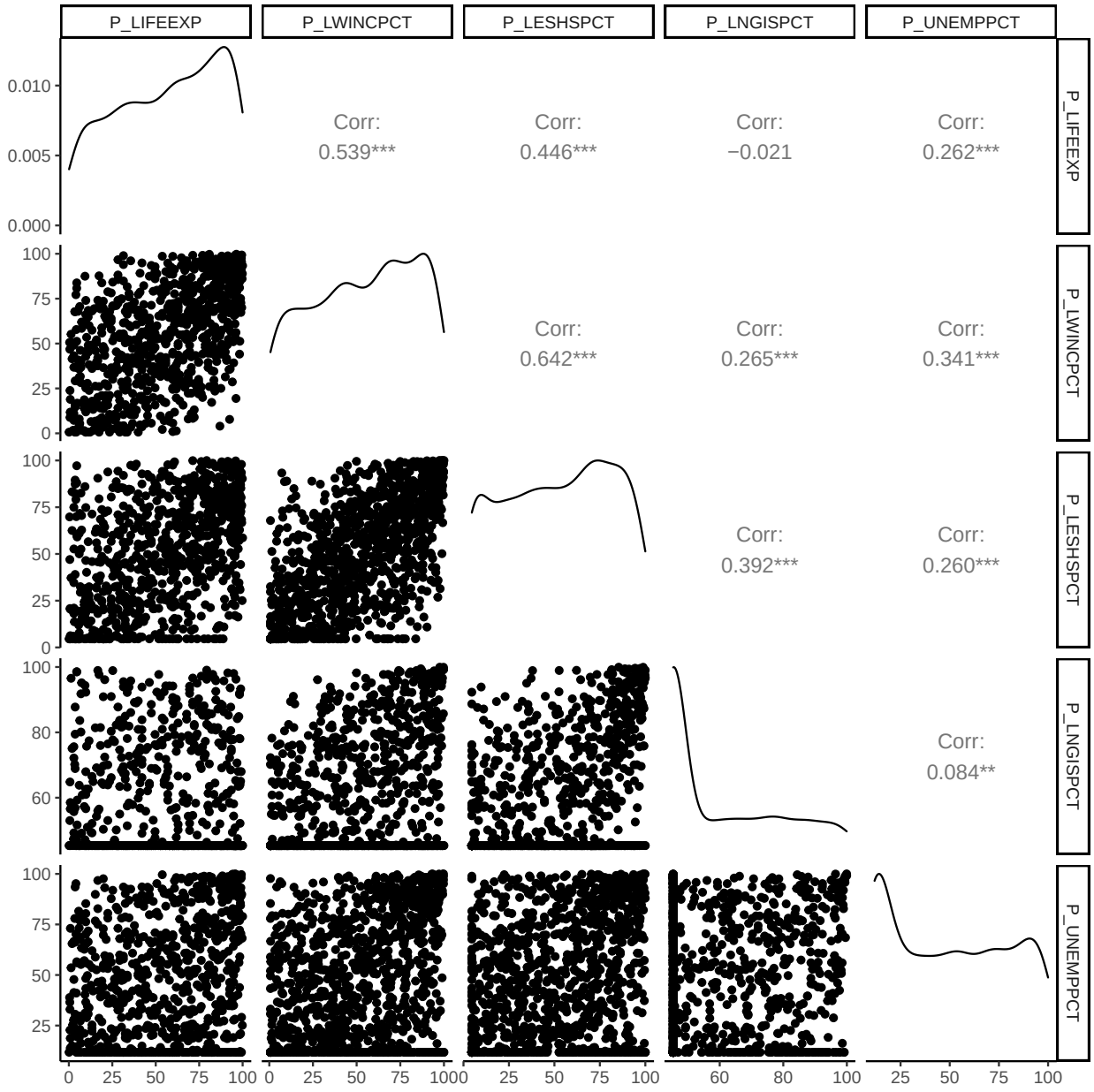
The Raw indexes are highly correlated with each of the sub-indexes, but especially with income and education. Rank correlations are roughly scale-free, so provide a more robust alternative where some metrics (as here) are not normally distributed.

```
the_data %>%
select(MINORPCT, NEG_LIFEEXP, LOWINCPCT, LESSHSPCT, LINGISOPCT, UNEMPPCT,
       Index_best_Ptiles) %>%
  cor(method = 'spearman', use = 'pairwise') %>%
  round(3)
#>                  MINORPCT NEG_LIFEEXP LOWINCPCT LESSHSPCT LINGISOPCT UNEMPPCT
#> MINORPCT            1.000       0.220     0.404     0.445      0.524    0.266
#> NEG_LIFEEXP         0.220       1.000     0.541     0.438     -0.054    0.243
#> LOWINCPCT           0.404       0.541     1.000     0.638      0.233    0.354
#> LESSHSPCT           0.445       0.438     0.638     1.000      0.346    0.277
#> LINGISOPCT          0.524      -0.054     0.233     0.346      1.000    0.113
#> UNEMPPCT            0.266       0.243     0.354     0.277      0.113    1.000
#> Index_best_Ptiles   0.518       0.701     0.838     0.811      0.397    0.605
#>                  Index_best_Ptiles
#> MINORPCT                     0.518
#> NEG_LIFEEXP                  0.701
#> LOWINCPCT                    0.838
#> LESSHSPCT                    0.811
#> LINGISOPCT                   0.397
```

```
#> UNEMPPCT                    0.605
#> Index_best_Ptiles           1.000
```

Percentiles

```
the_data %>%
  select( "P_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT", "P_LNGISPCT","P_UNEMPPCT" ) %>%
  slice_sample(prop = 0.005, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 275 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 275 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 275 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 275 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 275 rows containing missing values
#> Warning: Removed 275 rows containing missing values (geom_point).
#> Removed 275 rows containing missing values (geom_point).
#> Removed 275 rows containing missing values (geom_point).
#> Removed 275 rows containing missing values (geom_point).
```



Even the percentiles don't provide a true uniform distribution because of zero inflation in the data. This is most pronounced for the `P_LNGISPCT` and `P_UNEMPPCT` variables, where a substantial fraction of census block groups have zero linguistic isolation (at roughly 40th percentile) and very low unemployment(at around 10th percentile).

```
the_data %>%
  select( "P_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT",
          "P_LNGISPCT","P_UNEMPPCT" ) %>%
  map(quick_sum) %>%
  unlist() %>%
  array(dim = c(4,5),
        dimnames = list(c('Mean', 'SD', 'Median', 'IQR'),
                        c( "P_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT",
          "P_LNGISPCT","P_UNEMPPCT" )))
```

```
#>        P_LIFEEXP P_LWINCPCT P_LESHSPCT P_LNGISPCT P_UNEMPPCT
#> Mean    54.12220   51.91407   50.75473   58.96775   50.47131
#> SD      29.21829   29.28085   29.02588   17.85648   29.30012
#> Median  55.78092   53.18598   51.69256   45.35320   49.59784
#> IQR     50.88127   50.52023   50.65150   27.56211   55.23926
```

The Raw indexes are highly correlated with each of the sub-indexes, but especially with income and education. Rank correlations are roughly scale-free, so provide a more robust alternative where some metrics (as here) are not normally distributed.

```
the_data %>%
select(P_MINORPCT, P_LIFEEXP, P_LWINCPCT, P_LESHSPCT, P_LNGISPCT, P_UNEMPPCT) %>%
  cor(method = 'spearman', use = 'pairwise') %>%
  round(3)
#>             P_MINORPCT P_LIFEEXP P_LWINCPCT P_LESHSPCT P_LNGISPCT P_UNEMPPCT
#> P_MINORPCT       1.000     0.220      0.404      0.445      0.524      0.266
#> P_LIFEEXP        0.220     1.000      0.541      0.438     -0.054      0.243
#> P_LWINCPCT       0.404     0.541      1.000      0.638      0.233      0.354
#> P_LESHSPCT       0.445     0.438      0.638      1.000      0.346      0.277
#> P_LNGISPCT       0.524    -0.054      0.233      0.346      1.000      0.113
#> P_UNEMPPCT       0.266     0.243      0.354      0.277      0.113      1.000
```

PCA Analysis of Sub-indexes

We can get more formal about the relationships between the different sub-indexes using Principal Components Analysis. The first PCA axis shows the "best fit" line through the multi-dimensional cloud of points defined by the set of sub-indexes. The second PCA axis defines the "best fit" line through the remaining variation, and so on.

Each PCA axis is a linear combination of the metrics, and thus can be thought of as a weighted average. The average of the metrics (as suggested in EPA's funding memo) is another linear combination of the sub-indexes. The first PCA axis is the optimal linear combination of the sub-indexes for summarizing the multidimensional data with just a single value, and thus should outperform the mean as a single measure of disadvantage.

Function for Plotting PCAs

I encapsulate the logic of plotting the PCA results just to simplify later code. This code provides no error checking.

```
plot_pca<- function(.pca, .scale = 2.5, .ann_space = 0.15,
                    .sample = 1.0,
                    .levels = c('NEG_LIFEEXP', 'LOWINCPCT', 'LESSHSPCT',
                                'LINGISOPCT', 'UNEMPPCT'),
                    .labels = c('Short Life', 'Income', 'School',
                                'Language', 'Unempl'),
                    .title = 'Principal Components Analysis')  {
  # .scale:  how much to expand the arrows to make them fit well against the plot
  # .ann_space: How far past the end of the arrow to place annotations
  # .sample -- what fraction of raw observations to show.
  # Gather the first two PCA axes as unit length vectors
```

```
  arrows <- as_tibble(.pca$rotation[,1:2]) %>%
     rename(PC1_Raw = PC1,
            PC2_Raw = PC2)

  # Scale length of each vector according to the standard deviations of
  # the relevant principal components (actually the square root of the
  # eigenvalues of the covariance matrix).

  scaled_arrows <-  pca$rotation %*% diag(pca$sdev) * .scale

  # Build the tibble containing  data to plot
  scaled_arrows <- as_tibble(scaled_arrows,
                             rownames = 'Variable',
                             .name_repair = ~paste0('PC', 1:5)) %>%
    select(Variable, PC1, PC2) %>%
    bind_cols(arrows) %>%
    mutate(ann1 = PC1 + .scale * .ann_space * PC1_Raw,
           ann2 = PC2 + .scale * .ann_space * PC2_Raw) %>%
    select(-PC1_Raw, -PC2_Raw) %>%
    mutate(Variable = factor(Variable,
                             levels = .levels,
                             labels = .labels ))

  plt <- as_tibble(pca$x) %>%
    slice_sample(prop = .sample) %>%
    ggplot(aes(PC1, PC2)) +
    geom_point(alpha = 0.25, color = 'grey15') +
    #geom_density_2d(color = 'grey65') +
    geom_segment(data = scaled_arrows,
                 mapping = aes(x = 0, y = 0, xend = PC1, yend = PC2),
                 arrow = arrow(length = unit(0.25, 'cm'), type = 'open'),
                 color = 'grey85') +
    geom_text(data = scaled_arrows,
              mapping = aes(x = ann1, y = ann2, label = Variable),
              color = 'grey85', size = 2.5) +
    ggtitle(.title) +
    theme_dark()

  return(plt)
}
```

Function to Calculate First PCA Axis Scores

I calculate scores anew to avoid alignment problems caused by missing data. As before, this function makes assumptions about the parameters passed to the function and provides no error checking.

```
calc_scores <- function(.dat, .pca) {
  names <- rownames(.pca$rotation)
  vals <- map(names, ~.dat[,.])
  vals <- do.call(cbind, vals)  # converts list of vectors to data frame
  vals <- as.matrix(vals)

  mults <- matrix(.pca$rotation[,1], nrow = length(.pca$rotation[,1]))
```

```
  print(mults)
  res <- vals %*% mults
  return(as.vector(res))
}
```

Scaled PCA

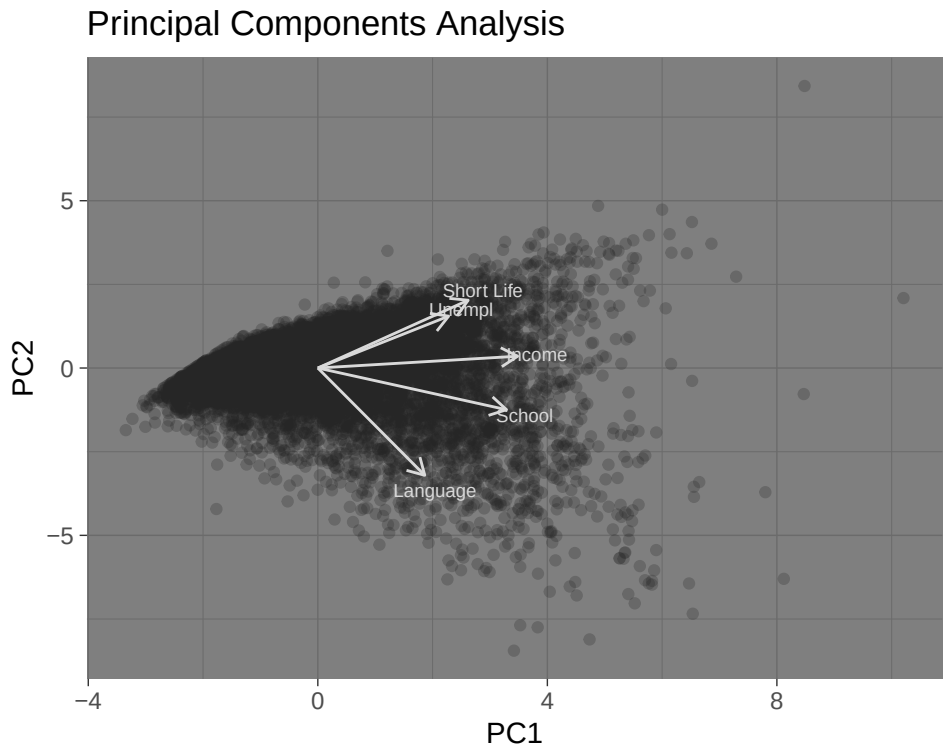
A scaled PCA first standardizes all variables to unit variance before conducting the PCA, thus making the PCA independent of scale (although not of distribution). that ensures they changing units won't change the results of the PCA, which (here) is an advantage.

```
pca <- the_data %>%
  select( "NEG_LIFEEXP", "LOWINCPCT", "LESSHSPCT", "LINGISOPCT","UNEMPPCT" ) %>%
  filter(complete.cases(.)) %>%
prcomp(scale. = TRUE)
```

```
summary(pca)
#> Importance of components:
#>                           PC1    PC2    PC3     PC4     PC5
#> Standard deviation     1.5507 1.0739 0.8549 0.62126 0.57018
#> Proportion of Variance 0.4809 0.2307 0.1462 0.07719 0.06502
#> Cumulative Proportion  0.4809 0.7116 0.8578 0.93498 1.00000
```

```
pca$rotation
#>                   PC1         PC2        PC3        PC4        PC5
#> NEG_LIFEEXP 0.4223973  0.47155085 -0.4960526  0.5709731  0.1647479
#> LOWINCPCT   0.5613756  0.08206038 -0.1153772 -0.3742976 -0.7243707
#> LESSHSPCT   0.5302285 -0.28978725 -0.1395412 -0.4579423  0.6369444
#> LINGISOPCT  0.3004420 -0.74615751  0.1335353  0.5562610 -0.1603919
#> UNEMPPCT    0.3674646  0.36080218  0.8386397  0.1214645  0.1293115
```

```
plot_pca(pca, .scale = 4, .sample = 0.1)
```

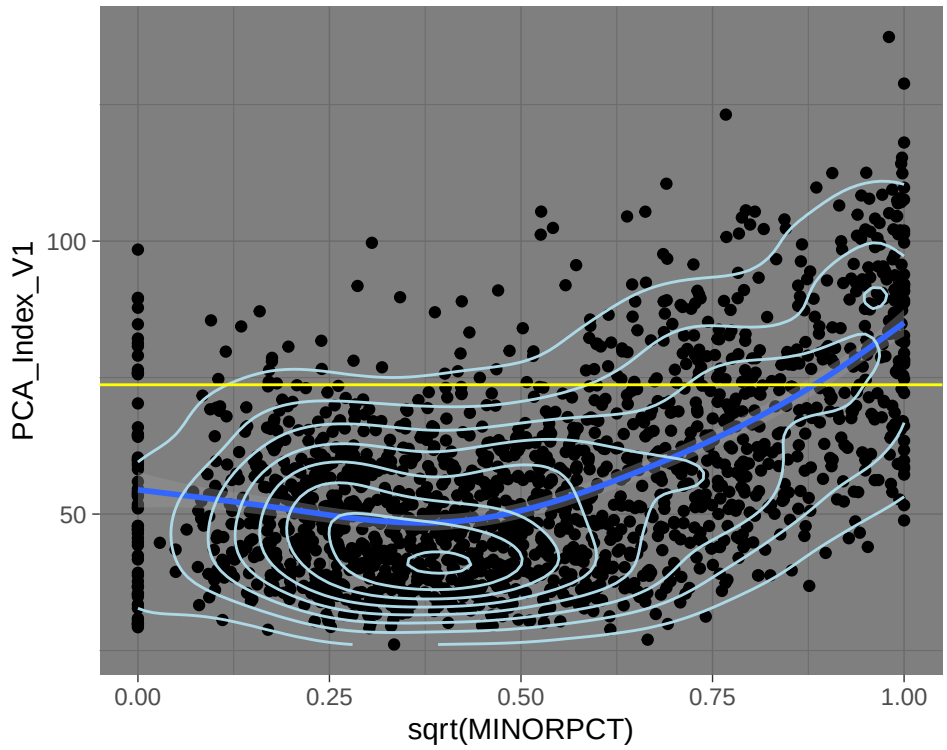


When variables are standardized to unit variance, the dominant axis is moderately correlated with all the sub-indexes, especially income and education. That suggests a common structure of community vulnerability. The first axis captures about half the structure of vulnerability in these five metrics.

```
the_data$PCA_Index_V1 <- calc_scores(the_data, pca)
#>           [,1]
#> [1,] 0.4223973
#> [2,] 0.5613756
#> [3,] 0.5302285
#> [4,] 0.3004420
#> [5,] 0.3674646
```

Minority communities are widely scattered along that first PCA axis. Many disadvantaged communities (judged by the first PCA Axis) have littel minority population. But overall, communities with a low percentage of minorities tend to have low PCA vulnerability scores.

```
the_data %>%
  slice_sample(prop = 0.01) %>%
ggplot(aes(sqrt(MINORPCT), PCA_Index_V1)) +
  geom_point() +
  geom_smooth() +
  geom_density_2d(color = 'lightblue') +
  geom_hline(yintercept = quantile(the_data$PCA_Index_V1, 0.8, na.rm = TRUE),
             color = 'yellow') +
  theme_dark()
#> `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
#> Warning: Removed 507 rows containing non-finite values (stat_smooth).
#> Warning: Removed 507 rows containing non-finite values (stat_density2d).
#> Warning: Removed 507 rows containing missing values (geom_point).
```



Percentiles

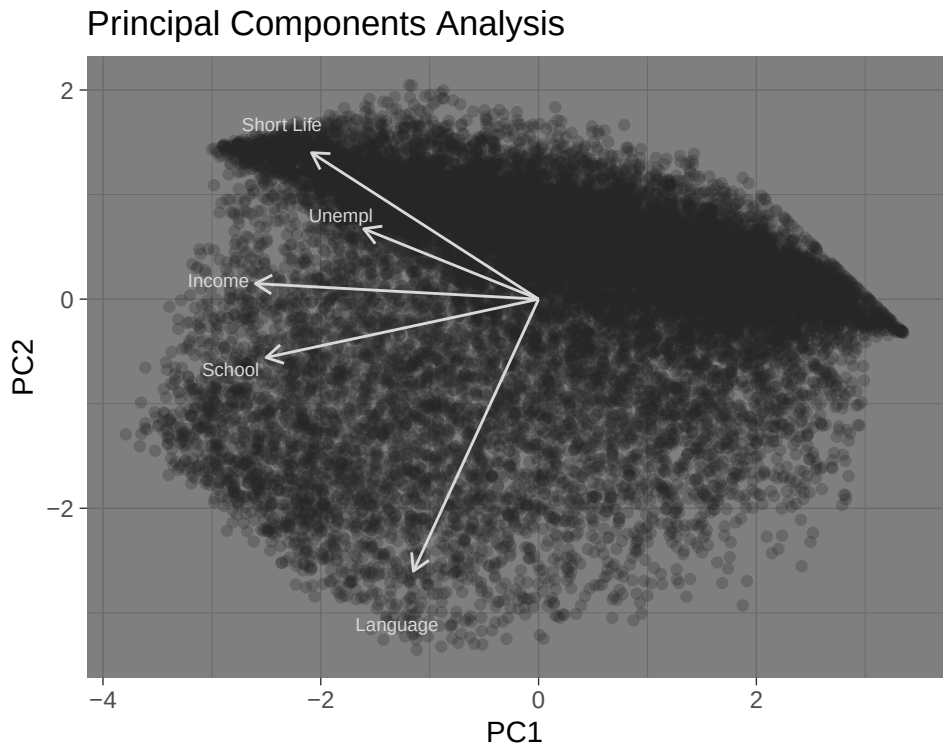
```
pca <- the_data %>%
  select( "P_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT",
          "P_LNGISPCT","P_UNEMPPCT" ) %>%
  filter(complete.cases(.)) %>%
prcomp(scale. = TRUE)
```

```
summary(pca)
#> Importance of components:
#>                           PC1    PC2    PC3     PC4     PC5
#> Standard deviation     1.5373 1.0282 0.8966 0.66156 0.58123
#> Proportion of Variance 0.4727 0.2114 0.1608 0.08753 0.06757
#> Cumulative Proportion  0.4727 0.6841 0.8449 0.93243 1.00000
```

The first two axes account for only 68% of the pattern in the sub-indexes.

```
pca$rotation
#>                   PC1         PC2         PC3         PC4         PC5
#> P_LIFEEXP  -0.4520868  0.45400782  0.34713064 -0.66191110 -0.17569406
#> P_LWINCPCT -0.5638361  0.04823933  0.13392266  0.28629844  0.76148525
#> P_LESHSPCT -0.5424473 -0.18094031  0.18948111  0.50896349 -0.61486962
#> P_LNGISPCT -0.2492848 -0.84337052 -0.08011539 -0.46561920  0.05799612
#> P_UNEMPPCT -0.3483023  0.21802899 -0.90512070 -0.06373244 -0.08856413
```

```
plot_pca(pca, .scale = 3, .ann_space = .2, .sample = 0.1,
         .levels = c(  "P_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT",
          "P_LNGISPCT","P_UNEMPPCT" ))
```



So, an index based on the (national) percentiles of the scores produces a PCA with less structure, as expected. Here axis 1 is a composite of all the sub-indexes, with the strongest association with Schooling, Income and Life Expectancy. Axis 2 is principally linguistic isolation, but also has moderate loading for life expectancy.

```
the_data$PCA_Index_V2 <- calc_scores(the_data, pca)
#>             [,1]
#> [1,] -0.4520868
#> [2,] -0.5638361
#> [3,] -0.5424473
#> [4,] -0.2492848
#> [5,] -0.3483023
```

Examining Results

The composite indexes are highly correlated, as expected.

```
the_data %>%
select(c(Index_Raw:PCA_Index_V2)) %>%
  cor(use = 'pairwise', method = 'spearman')  %>%
  round(3)
#>                 Index_Raw Index_5_Ptiles Index_4_Ptiles Index_best_Ptiles
#> Index_Raw           1.000          0.954          0.944              0.954
```

```
#> Index_5_Ptiles        0.954          1.000          0.959          1.000
#> Index_4_Ptiles        0.944          0.959          1.000          0.970
#> Index_best_Ptiles     0.954          1.000          0.970          1.000
#> PCA_Index_V1          0.997          0.943          0.930          0.943
#> PCA_Index_V2         -0.965         -0.993         -0.944         -0.993
#>                   PCA_Index_V1 PCA_Index_V2
#> Index_Raw                0.997       -0.965
#> Index_5_Ptiles           0.943       -0.993
#> Index_4_Ptiles           0.930       -0.944
#> Index_best_Ptiles        0.943       -0.993
#> PCA_Index_V1             1.000       -0.961
#> PCA_Index_V2            -0.961        1.000
```

Note that the `PCA_Index_V2` scores are **negatively** correlated with the other metrics. PCA, like most ordinations, is defined only to reflections. We swap the sign here. (I do not understand how the direction of the PCA is determined. In some runs of these calculations one PCA index is reversed, sometimes both are, so this needs to be checked manually).

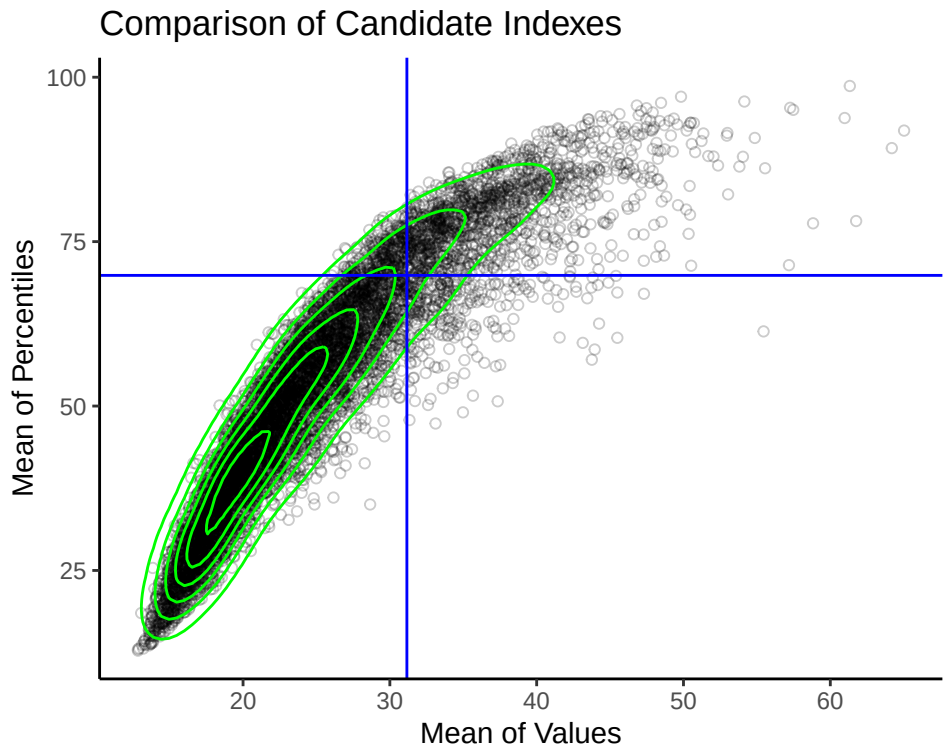
At first it may appear odd that Index_5_Ptiles and Index_best_Ptiles show up as perfently correlated, as the two contain different values. But recal that the calculation is based on pairwise comparisons, and the two **are** identical, except when `P_LIFEEXP` is `NA`, so the pairwise correlation is blind to the presence of different data in the places where the first is itlsef `NA`.

```
#the_data$PCA_Index_V1 <- -the_data$PCA_Index_V1
the_data$PCA_Index_V2 <- -the_data$PCA_Index_V2
```

Graphics

I plot only 5% of the data, to reduce the size of the PDF file....

```
p80_sum <- quantile(the_data$Index_Raw, 0.8, na.rm = TRUE)
p80_sum_of_p  <- quantile(the_data$Index_5_Ptiles, 0.8, na.rm = TRUE)

plt <- the_data %>%
  slice_sample(prop = 0.05) %>%
  ggplot(aes(Index_Raw, Index_5_Ptiles)) +
  geom_point(alpha = 0.2, shape = 21) +
  geom_density_2d(color = 'green') +
  geom_vline(xintercept = p80_sum, color = 'blue') +
  geom_hline(yintercept = p80_sum_of_p, color = 'blue') +

  xlab( 'Mean of Values') +
  ylab( 'Mean of Percentiles') +
  ggtitle('Comparison of Candidate Indexes')
plt
#> Warning: Removed 2637 rows containing non-finite values (stat_density2d).
#> Warning: Removed 2637 rows containing missing values (geom_point).
```



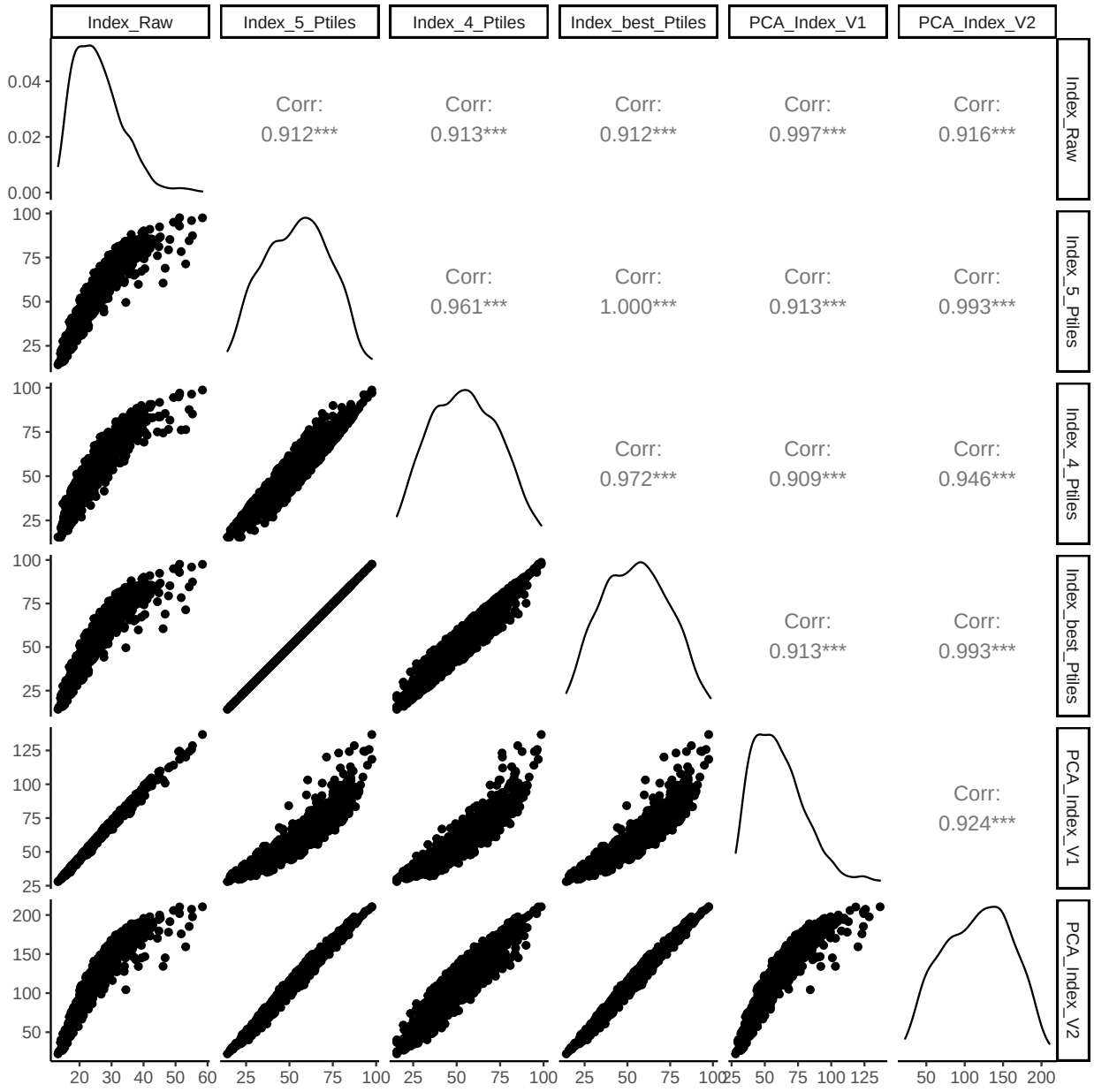
The relationship between the indexes is not linear, but correlations are likely to be high over any finite range. Unfortunately, the correlations appear less robust at higher index values, exactly where we may want the most precision. The Blue lines represent the 80th percentiles in each axis. Note that the 80th percentile of the mean of five percentiles lies well below 80.

Pairs Plot of All Indexes

Again, I reduce plot complexity by plotting only 1% of the data.

```
the_data %>%
select(c(Index_Raw:PCA_Index_V2)) %>%
  slice_sample(prop = 0.005, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 258 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values
```

```
#> Warning: Removed 258 rows containing missing values (geom_point).
#> Warning: Removed 258 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values
#> Warning: Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values
#> Warning: Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values
#> Warning: Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Warning: Removed 258 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 258 rows containing missing values
#> Warning: Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Removed 258 rows containing missing values (geom_point).
#> Warning: Removed 258 rows containing non-finite values (stat_density).
```



Note that both index 1 and PCA based on the raw (scaled) data have a fairly long tail. In contrast, Index 2 and the PCA V2 (based on national percentiles) are highly correlated, and more evenly spread over the range.

One of the statistical features this reveals is that the sum (or average) of percentiles is not distributed according to a uniform distribution. The eightieth percentile of the sum of five percentiles is substantially below 80, as the separate percentiles are only some what correlated, because it is unlikely that all five component metrics are high for a single location.

Upon reflection, we see that:

1. The indexes based on national percentiles have nice statistical properties.
2. The difference between a PCA score and a simple unweighted mean is relatively minor, presumably because the primary PCA axis is correlated with all five metrics, with weightings not to far from equal.

Output Results

```
write_csv(the_data, 'National_Draft_Indexes.csv')
```