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(),
                       STATE_NAME = col_character(),
                       ST_ABBREV = col_character(),
                       LOWINCPCT = col_double(),
                       LESSHSPCT  = col_double(),
                       LINGISOPCT = col_double(),
                       UNEMPPCT = col_double(),
                       P_LWINCPCT = col_double(),
                       P_LNGISPCT = col_double(),
                       P_LESHSPCT  = col_double(),
                       P_UNEMPPCT = col_double()))
```

```
the_file <- 'Tract2010_LifeExpectancy.csv'
the_path <- file.path(data_folder, the_file)
life_data <- read_csv(the_path,
                      n_max = 73057,
                      col_types = cols_only(LIFEEXP = col_double(),
                                            GEOID10 = col_character(),
                                            Life_Expectancy_Standard_Error = col_double()))
```

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 <- the_data %>%
  mutate(tract_geoid10 = paste0('0', substr(ID, 1, 10))) %>%
  left_join(life_data, by = c('tract_geoid10' = 'GEOID10')) %>%
  select(-tract_geoid10)
```

```
rm(life_data)
```

```
miles_per_meter <- 0.000621371
sq_miles_per_sq_meters <- miles_per_meter^2
```

```
the_data <- the_data %>%
  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, LIFEEXP,  .after = ID) %>%
  relocate(NEG_LIFEEXP, .after = LIFEEXP_SE)
```

Utility Functions

```
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)
              ))
```

```
quick_percentile <- function(.dat) {
  L <- sum(! is.na(.dat))
  val <- rank(.dat) / L
  val[is.na(.dat)] <- NA
  return(val)
  }
```

Functions for Calculating Indexes

```
calc_index_1 <- function(.data) {
  index_1 <- with(.data,
    (NEG_LIFEEXP + LOWINCPCT + LESSHSPCT +LINGISOPCT + UNEMPPCT) / 5)
  return(index_1)
}
```

The primary alternative is to calculate percentiles within each sub-index, and sum those. That makes the composite index approximately scale-free in each sub-index. Again, because of correlations among sub-indexes, that won't be quite correct, but it will be close.

```
calc_index_2 <- function(.data) {
  index_2 <- with(.data,
                  (P_NEG_LIFEEXP + P_LWINCPCT + P_LESHSPCT +
                     P_LNGISPCT +
                     P_UNEMPPCT) / 5)
  return(index_2)
}
```

More Calculations

```
the_data <- the_data %>%
  mutate(P_NEG_LIFEEXP = quick_percentile(NEG_LIFEEXP) * 100) %>%
  relocate(P_NEG_LIFEEXP, .before = P_LWINCPCT)
```

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

- Index_1 is based on averaging the VALUES
- Index_2 is based on averaging the PERCENTILES
- P_Index_1 records the percentiles of Index 1, and is thus scale-free.
- P_Index_2 shows percentiles of Index 2.

```
the_data$Index_1 <- calc_index_1(the_data)
the_data$Index_2 <- calc_index_2(the_data)
the_data$P_Index_1 <- quick_percentile(the_data$Index_1)
the_data$P_Index_2 <- quick_percentile(the_data$Index_2)
```

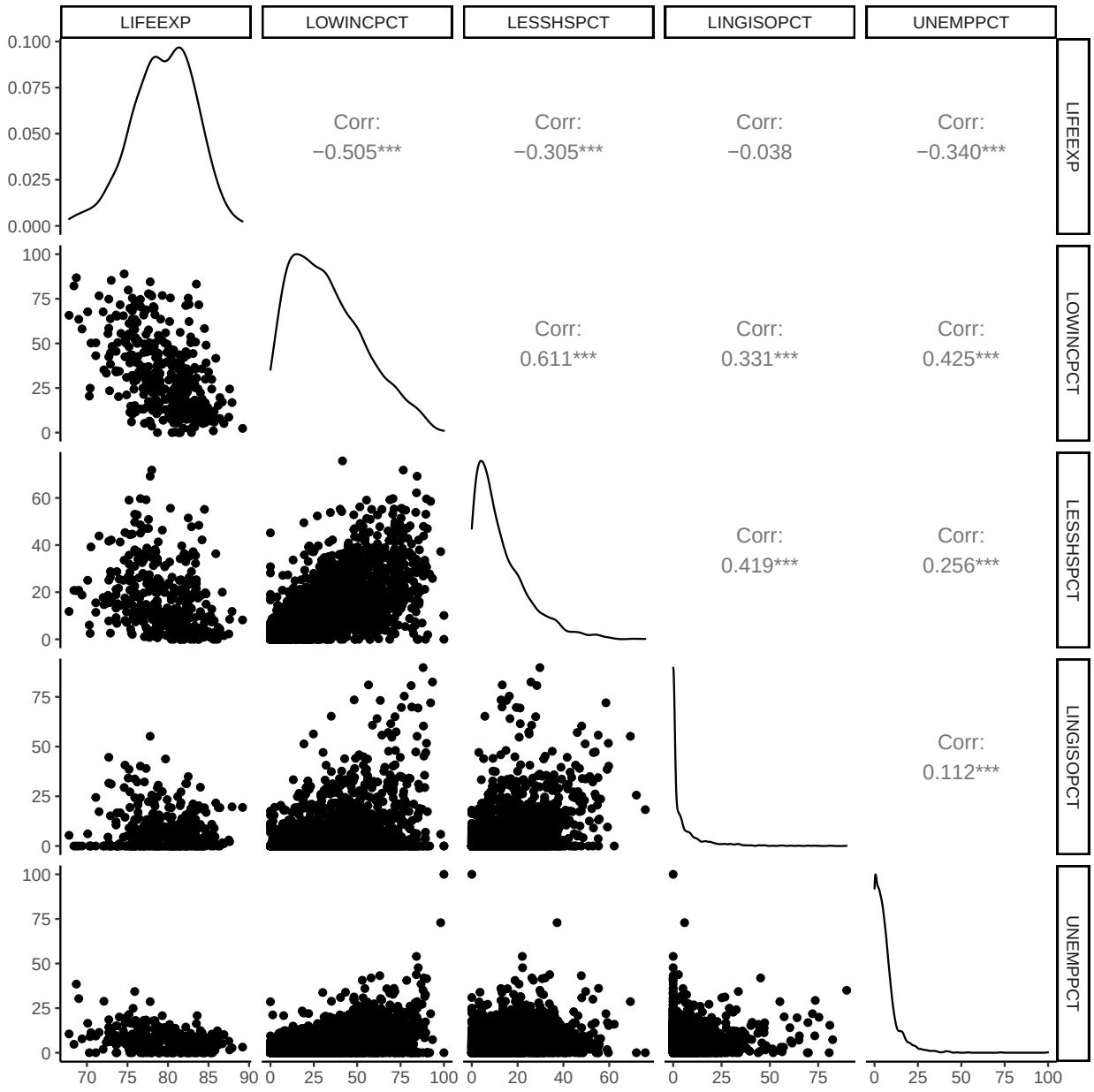
Distributions

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 5% sample of the data.

```
the_data %>%
  select( "LIFEEXP", "LOWINCPCT", "LESSHSPCT", "LINGISOPCT","UNEMPPCT" ) %>%
  slice_sample(prop = 0.01, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 1830 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1830 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1830 rows containing missing values
```

```
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1830 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1830 rows containing missing values
#> Warning: Removed 1830 rows containing missing values (geom_point).
#> Removed 1830 rows containing missing values (geom_point).
#> Removed 1830 rows containing missing values (geom_point).
#> Removed 1830 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.

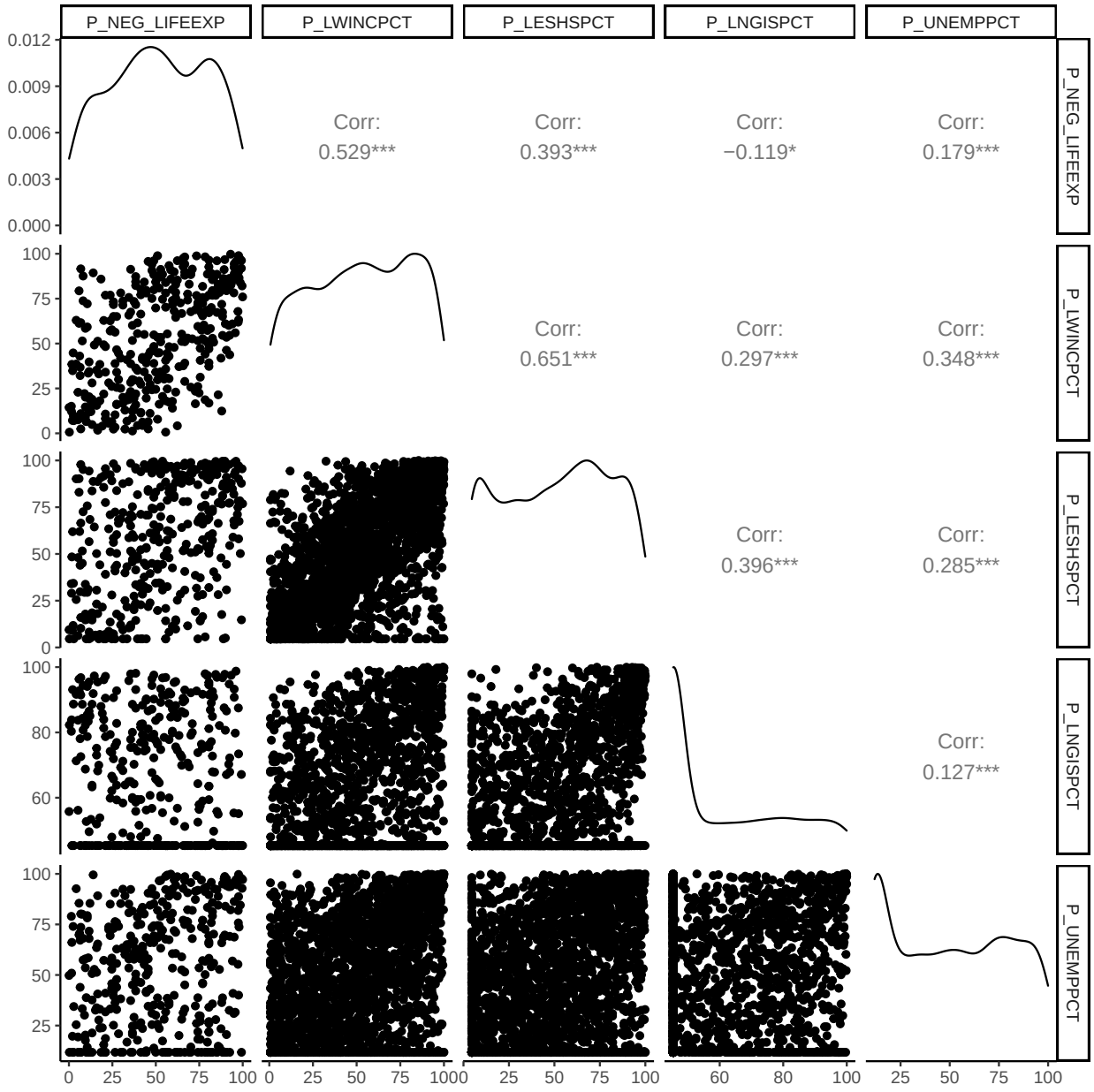
Adding or averaging raw values will lead to indexes dominated by the sub-indexes with the largest variance. For some analyses, I would consider data transformations, but that will not be needed here, since I will work with percentiles instead.

Means, SD, Medians and IGR

```
the_data %>%
  select( "LIFEEXP", "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     79.229939  32.89986 12.736512   5.134305 5.887947
#> SD        3.869197  21.38511 11.913489  11.010335 6.530343
#> Median   79.500000  29.45455  9.269663   0.000000 4.187817
#> IQR       5.200000  31.17712 13.740259   5.313351 6.306004
```

Simply adding these indexes together 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.

```
the_data %>%
  select( "P_NEG_LIFEEXP", "P_LWINCPCT", "P_LESHSPCT", "P_LNGISPCT","P_UNEMPPCT" ) %>%
  slice_sample(prop = 0.01, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 1827 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1827 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1827 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1827 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1827 rows containing missing values
#> Warning: Removed 1827 rows containing missing values (geom_point).
#> Removed 1827 rows containing missing values (geom_point).
#> Removed 1827 rows containing missing values (geom_point).
#> Removed 1827 rows containing missing values (geom_point).
```



PCA Analysis of Sub-indexes

We can get more formal about the relationships between the different sub-indexes by calculating a 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.

The first PCA axis can be thought of as a linear combination of the sub-indexes. The average of the sub-indexes (as suggested in the funding memo) is another linear combination of the sub-indexes. In a specific sense, the first PCA axis is the optimal linear combination of the sub-indexes for summarizing the multidimensional data with just a single value.

Function for Plotting PCAs

I encapsulate the logic of plotting the PCA results just to simplify later code

```
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.

```
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

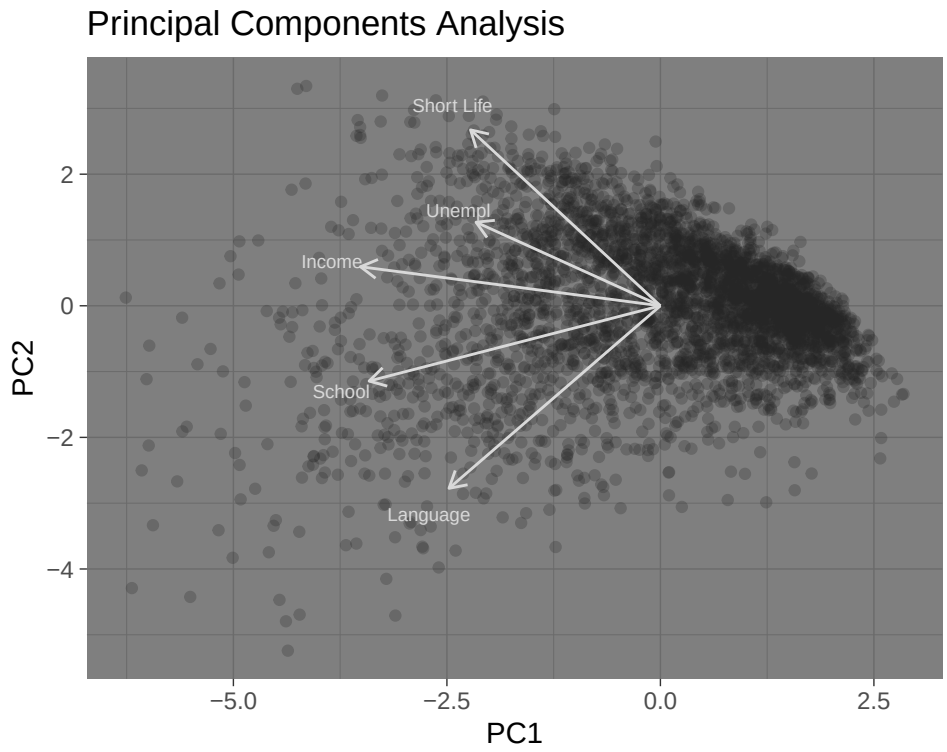
A scaled PCA first standardizes all variables to unit variance before conducting the PCA, thus making sure that changing units won't change results.

```
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     1.5752 1.0642 0.8827 0.57740 0.52300
#> Proportion of Variance 0.4963 0.2265 0.1558 0.06668 0.05471
#> Cumulative Proportion  0.4963 0.7228 0.8786 0.94529 1.00000
```

```
pca$rotation
#>                    PC1        PC2         PC3        PC4         PC5
#> NEG_LIFEEXP -0.3528600  0.6281350 -0.43585178 -0.5260733  0.11923213
#> LOWINCPCT   -0.5573362  0.1393137 -0.13116435  0.5061037 -0.62978019
#> LESSHSPCT   -0.5413920 -0.2687817 -0.14728642  0.3256054  0.71199642
#> LINGISOPCT  -0.3928658 -0.6517124  0.01469373 -0.5885824 -0.27254767
#> UNEMPPCT    -0.3426621  0.2984383  0.87802047 -0.1210695  0.08910381
```

```
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. Language and to a lesser extent life expectancy are most heavily loaded on the second PCA axis.

This suggests that an index that is NOT based on scaled values of percentiles will largely function as a surrogate for income, while if the index is based on scaled values or percentiles, the index will reflect the effects of several different sources of disadvantage.

```
the_data$PCA_Index_V1 <- calc_scores(the_data, pca)
#>             [,1]
#> [1,] -0.3528600
#> [2,] -0.5573362
#> [3,] -0.5413920
#> [4,] -0.3928658
#> [5,] -0.3426621
```

Percentiles

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

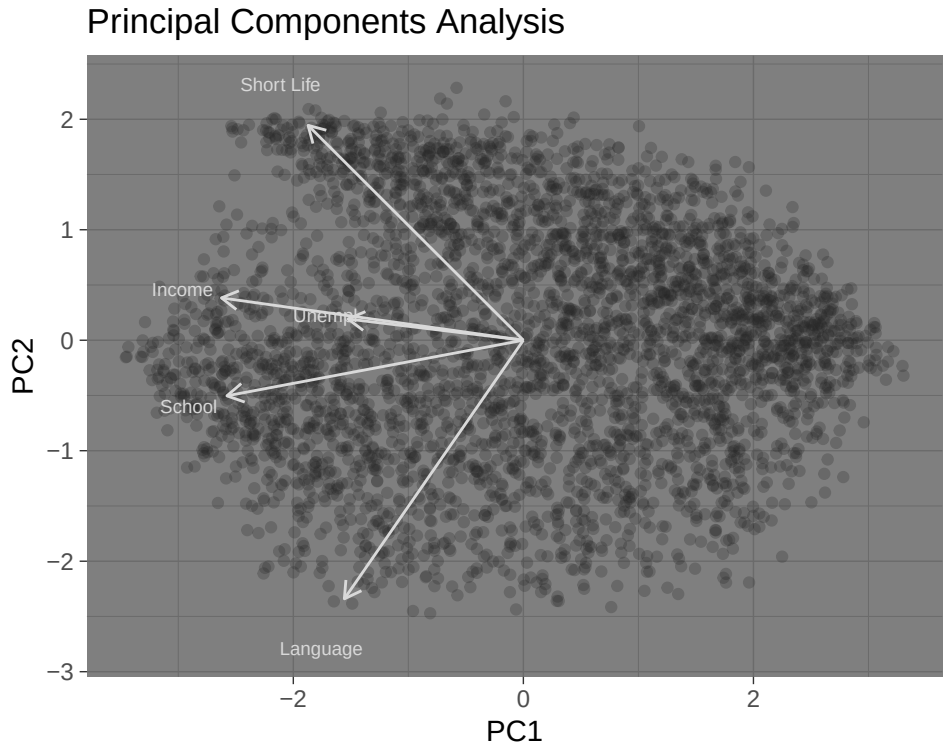
```
summary(pca)
#> Importance of components:
```

```
#>                          PC1    PC2    PC3     PC4     PC5
#> Standard deviation     1.556 1.0380 0.9139 0.60526 0.54839
#> Proportion of Variance 0.484 0.2155 0.1671 0.07327 0.06015
#> Cumulative Proportion  0.484 0.6995 0.8666 0.93985 1.00000
```

The first two axes account for only 70% of the pattern in the sub-indexes.

```
pca$rotation
#>                       PC1         PC2        PC3         PC4         PC5
#> P_NEG_LIFEEXP -0.4010435  0.62458131  0.2412909 -0.60813732 -0.14494827
#> P_LWINCPCT    -0.5624525  0.12306640  0.1208611  0.37443468  0.71672385
#> P_LESHSPCT    -0.5522164 -0.16161435  0.1948937  0.43406722 -0.66523679
#> P_LNGISPCT    -0.3334511 -0.75155060  0.1014576 -0.54373378  0.13432051
#> P_UNEMPPCT    -0.3266171  0.06168752 -0.9374935 -0.07685641 -0.06866533
```

```
plot_pca(pca, .scale = 3, .ann_space = .2, .sample = 0.1,
         .levels = c(  "P_NEG_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 Unemployment. Axis 2 is principally linguistic isolation, but also has moderate loading for unemployment.

```
the_data$PCA_Index_V2 <- calc_scores(the_data, pca)
#>            [,1]
#> [1,] -0.4010435
#> [2,] -0.5624525
```

```
#> [3,] -0.5522164
#> [4,] -0.3334511
#> [5,] -0.3266171
```

Examining Results

Correlations

The Raw indexes are highly correlated with each of the sub-indexes, but especially with income and education.

```
the_data %>%
select(NEG_LIFEEXP, LOWINCPCT, LESSHSPCT, LINGISOPCT, UNEMPPCT,
       Index_1, Index_2) %>%
  cor(use = 'pairwise')  %>%
  round(3)
#>             NEG_LIFEEXP LOWINCPCT LESSHSPCT LINGISOPCT UNEMPPCT Index_1 Index_2
#> NEG_LIFEEXP       1.000     0.522     0.299     -0.030    0.238   0.477   0.628
#> LOWINCPCT         0.522     1.000     0.609      0.359    0.425   0.907   0.837
#> LESSHSPCT         0.299     0.609     1.000      0.469    0.274   0.867   0.762
#> LINGISOPCT       -0.030     0.359     0.469      1.000    0.180   0.642   0.478
#> UNEMPPCT          0.238     0.425     0.274      0.180    1.000   0.481   0.553
#> Index_1           0.477     0.907     0.867      0.642    0.481   1.000   0.916
#> Index_2           0.628     0.837     0.762      0.478    0.553   0.916   1.000
```

Rank correlations are roughly scale-free, so provide a more robust alternative where some metrics (as here) are not normally distributed.

```
the_data %>%
select(NEG_LIFEEXP, LOWINCPCT, LESSHSPCT, LINGISOPCT, UNEMPPCT,
       Index_1, Index_2) %>%
  cor(method = 'spearman', use = 'pairwise') %>%
  round(3)
#>             NEG_LIFEEXP LOWINCPCT LESSHSPCT LINGISOPCT UNEMPPCT Index_1 Index_2
#> NEG_LIFEEXP       1.000     0.538     0.390     -0.057    0.194   0.538   0.645
#> LOWINCPCT         0.538     1.000     0.638      0.233    0.354   0.926   0.854
#> LESSHSPCT         0.390     0.638     1.000      0.346    0.277   0.850   0.834
#> LINGISOPCT       -0.057     0.233     0.346      1.000    0.113   0.501   0.463
#> UNEMPPCT          0.194     0.354     0.277      0.113    1.000   0.437   0.574
#> Index_1           0.538     0.926     0.850      0.501    0.437   1.000   0.959
#> Index_2           0.645     0.854     0.834      0.463    0.574   0.959   1.000
```

The composite indexes are highly correlated, as expected.

```
the_data %>%
select(c(Index_1:PCA_Index_V2)) %>%
  cor(use = 'pairwise', method = 'pearson')  %>%
  round(3)
#>             Index_1 Index_2 P_Index_1 P_Index_2 PCA_Index_V1 PCA_Index_V2
#> Index_1       1.000   0.916     0.955     0.908       -0.998       -0.927
#> Index_2       0.916   1.000     0.956     0.997       -0.909       -0.994
```

```
#> P_Index_1      0.955   0.956     1.000     0.959       -0.957       -0.972
#> P_Index_2      0.908   0.997     0.959     1.000       -0.902       -0.993
#> PCA_Index_V1  -0.998  -0.909    -0.957    -0.902        1.000        0.926
#> PCA_Index_V2  -0.927  -0.994    -0.972    -0.993        0.926        1.000
```

Note that the PCA Index scores are **negatively** correlated with the other metrics. PCA, like most ordinations, is defined only to reflections. We swap the sign here.

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

Rank correlations are even higher.

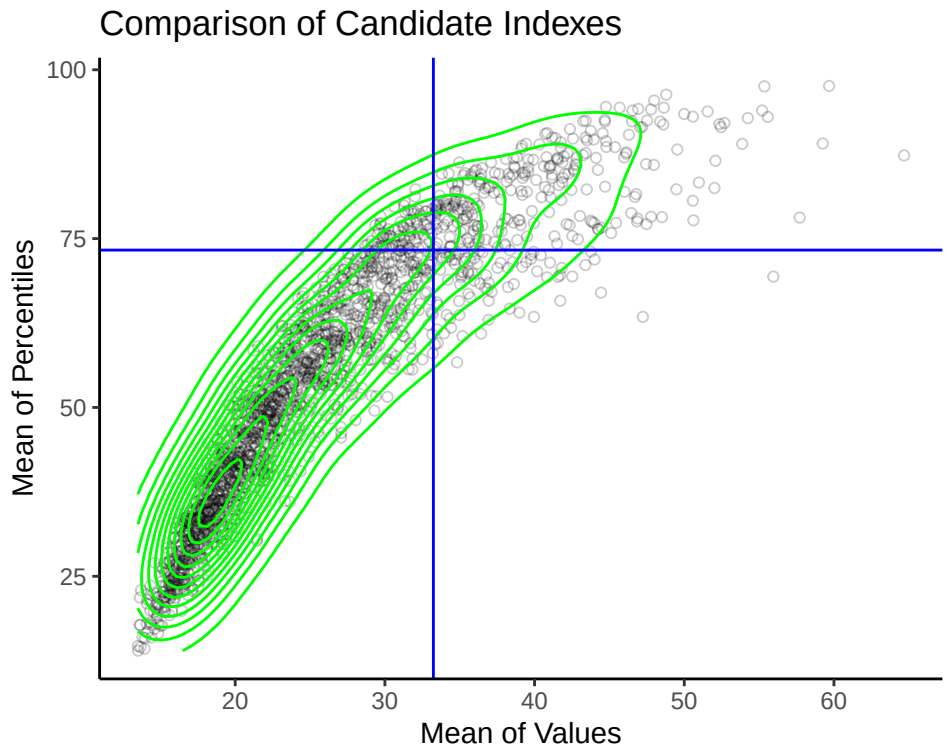
```
the_data %>%
select(c(Index_1, Index_2, P_Index_1, P_Index_2,  PCA_Index_V1, PCA_Index_V2)) %>%
  cor(use = 'pairwise', method = 'spearman')  %>%
  round(3)
#>              Index_1 Index_2 P_Index_1 P_Index_2 PCA_Index_V1 PCA_Index_V2
#> Index_1        1.000   0.959     1.000     0.959        0.998        0.973
#> Index_2        0.959   1.000     0.959     1.000        0.948        0.994
#> P_Index_1      1.000   0.959     1.000     0.959        0.998        0.973
#> P_Index_2      0.959   1.000     0.959     1.000        0.948        0.994
#> PCA_Index_V1   0.998   0.948     0.998     0.948        1.000        0.968
#> PCA_Index_V2   0.973   0.994     0.973     0.994        0.968        1.000
```

So, the raw index and the PCA indexes based on the same measurements are highly correlated, as one might expect. one is a simple average, the other a weighted average of the same five basic metrics. Note that the Percentile of scores and the scores are are perfectly rank correlated. That's expected.

Graphics

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

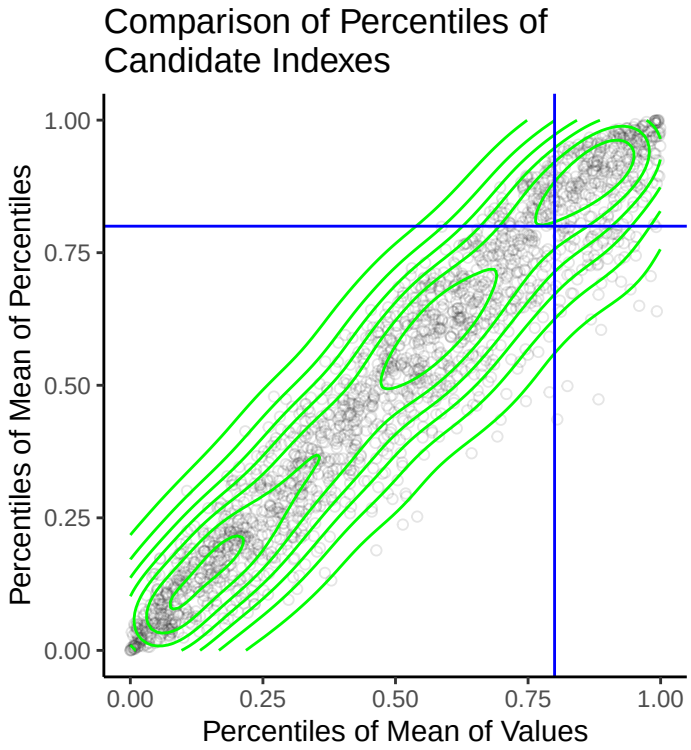
```
p80_sum <- quantile(the_data$Index_1, 0.8, na.rm = TRUE)
p80_sum_of_p  <- quantile(the_data$Index_2, 0.8, na.rm = TRUE)

plt <- the_data %>%
  slice_sample(prop = 0.05) %>%
  ggplot(aes(Index_1, Index_2)) +
  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 9133 rows containing non-finite values (stat_density2d).
#> Warning: Removed 9133 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.

The relationship between the percentiles are close to linear, but each index is much closer – although not identical.

```
p80_sum <- quantile(the_data$P_Index_1, 0.8, na.rm = TRUE)
p80_sum_of_p  <- quantile(the_data$P_Index_2, 0.8, na.rm = TRUE)

plt <- the_data %>%
  slice_sample(prop = 0.05) %>%
  ggplot(aes(P_Index_1, P_Index_2)) +
  geom_point(alpha = 0.1, shape = 21) +
  geom_density_2d(color = 'green') +
  geom_vline(xintercept = p80_sum, color = 'blue') +
  geom_hline(yintercept = p80_sum_of_p, color = 'blue') +
  xlab( 'Percentiles of Mean of Values') +
  ylab( 'Percentiles of Mean of Percentiles') +
  ggtitle('Comparison of Percentiles of\nCandidate Indexes') +
  coord_fixed()
plt
#> Warning: Removed 9132 rows containing non-finite values (stat_density2d).
#> Warning: Removed 9132 rows containing missing values (geom_point).
```



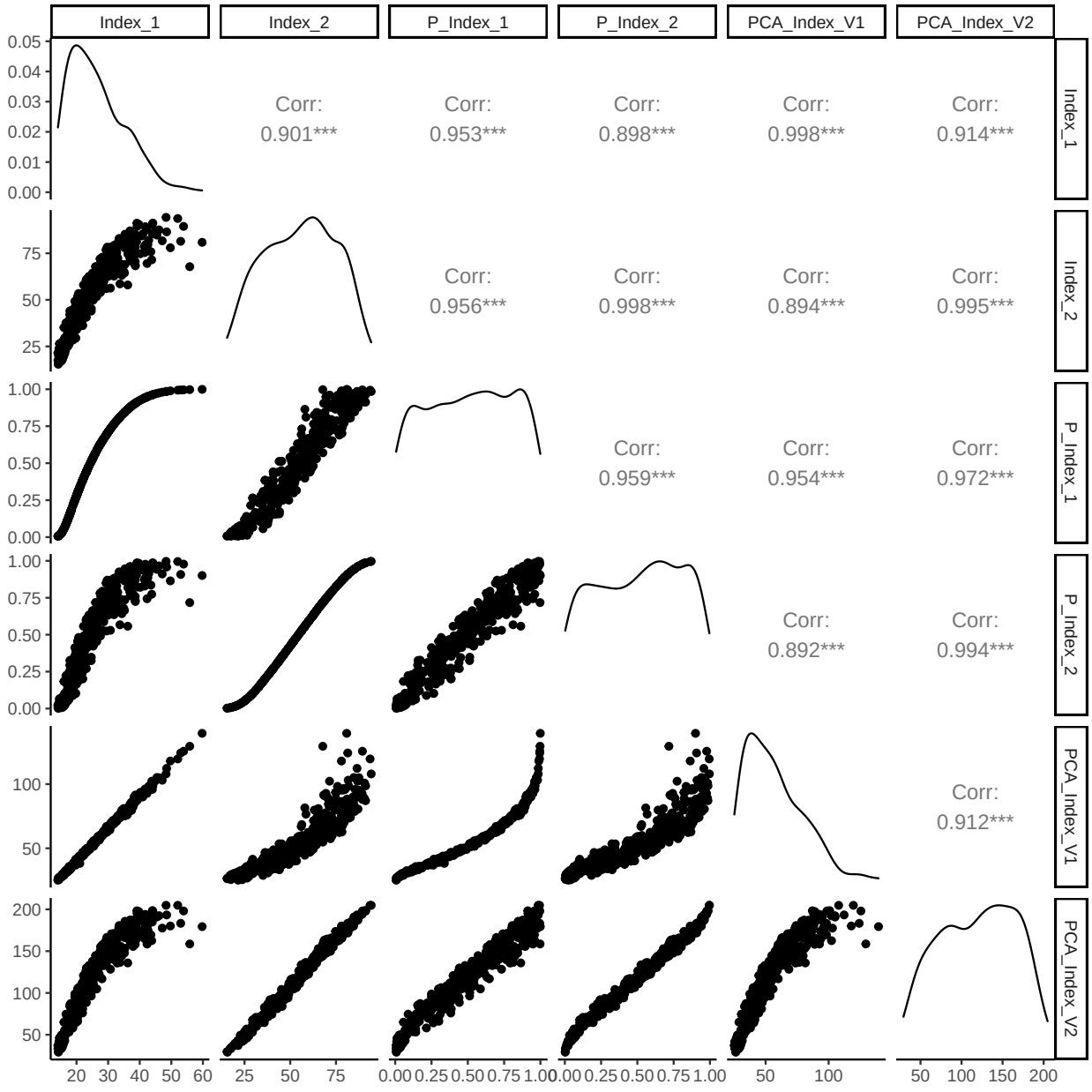
Here the percentiles DO fall

Pairs Plot of All Indexes

Again, I reduce plot complexity by plotting only 5% of the data.

```
the_data %>%
select(c(Index_1:PCA_Index_V2)) %>%
  slice_sample(prop = 0.01, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 1828 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values
#> Warning: Removed 1828 rows containing missing values (geom_point).
#> Warning: Removed 1828 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
```

```
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values
#> Warning: Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Warning: Removed 1828 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values
#> Warning: Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Warning: Removed 1828 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values
#> Warning: Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Warning: Removed 1828 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 1828 rows containing missing values
#> Warning: Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Removed 1828 rows containing missing values (geom_point).
#> Warning: Removed 1828 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, thus 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. That because the primary PCA axis for bot hPCAs is moderately correlated with all five metrics, the difference between a PCA scor and a simple unweighted) mean is relatively minor

Output Results

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