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 50 MB), so I have not included it in the GitHub repository (GitHub does not appreciate files over 100 MB). The raw CSV file contains 74001 records, and 166 columns. Most, but not all are numeric. The Health data is significantly smaller. smaller. The large files also poses potential data access challenges in R.

I read in just the required data columns for now.

```
the_file <- 'EJSCREEN_Full_tracts.txt'
the_path <- file.path(data_folder, the_file)
the_data <- read_csv(the_path,
                     n_max = 74001,
                     col_types = cols_only(
                       ID = col_character(),
                       LOWINCPCT = col_double(),
                       LINGISOPCT = col_double(),
                       LESSHSPCT  = col_double(),
                       UNEMPPCT = col_double()))
```

```
the_file <- 'Tract2010_LifeExpectancy.txt'
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(),
                                            State = col_character(),
                                            County = col_character(),
                                            Life_Expectancy_Standard_Error = col_double(),
                                            Shape_Length = col_double(),
                                            Shape_Area = col_double()))
```

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

```
the_data <- inner_join(life_data, the_data, by = c('GEOID10' = 'ID')) %>%
  relocate(LIFEEXP , .after = County) %>%
  rename(LIFEEXP_SE = Life_Expectancy_Standard_Error,
         Perimeter = Shape_Length,
         Area = Shape_Area) %>%
  mutate(Perimeter_km = Perimeter / 1000,
         Perimeter = Perimeter * miles_per_meter,
         Area_ha = Area / 10000,
         Area = Area * sq_miles_per_sq_meters,
         Shape_Index = Area / Perimeter^2) %>%
  mutate(NEG_LIFEEXP = 150 - LIFEEXP,              # Higher life expectancy is good
         LOWINCPCT = 100* LOWINCPCT,
         LESSHSPCT = 100* LESSHSPCT,
         LINGISOPCT =  100* LINGISOPCT,
         UNEMPPCT   = 100* UNEMPPCT) %>%
  relocate(Perimeter, Area, .after = Area_ha)

rm(life_data)
```

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_LOWINCPCT +p_LESSHSPCT +
                     p_LINGISOPCT +
                     p_UNEMPPCT) / 5)
  return(index_2)
}
```

More Calculations

```
the_data <- the_data %>%
  mutate(p_NEG_LIFEEXP = quick_percentile(NEG_LIFEEXP),
         p_LOWINCPCT   = quick_percentile(LOWINCPCT),
         p_LESSHSPCT   = quick_percentile(LESSHSPCT),
         p_LINGISOPCT  = quick_percentile(LINGISOPCT),
         p_UNEMPPCT    = quick_percentile(UNEMPPCT))
```

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

```
the_data$Index_1 <- calc_index_1(the_data)
the_data$Index_2 <- calc_index_2(the_data) * 100
the_data$p_Index_1 <- quick_percentile(the_data$Index_1)
the_data$p_Index_2 <- quick_percentile(the_data$Index_2)
```

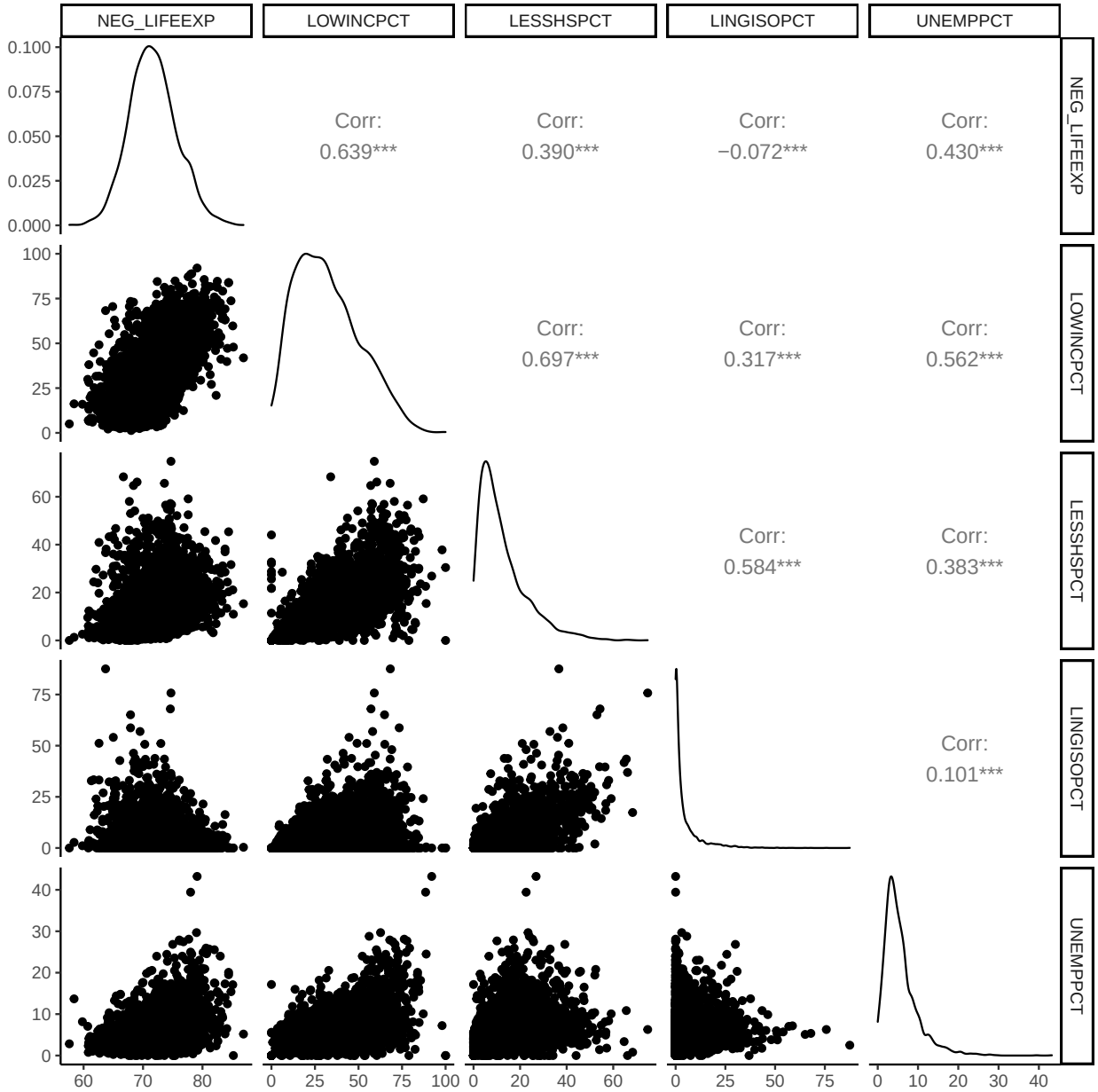
Distributions

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( "NEG_LIFEEXP", "LOWINCPCT", "LESSHSPCT", "LINGISOPCT","UNEMPPCT" ) %>%
  slice_sample(prop = 0.05, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 310 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 310 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 310 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 310 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 310 rows containing missing values
#> Warning: Removed 310 rows containing missing values (geom_point).
```

```
#> Removed 310 rows containing missing values (geom_point).
#> Removed 310 rows containing missing values (geom_point).
#> Removed 310 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( "NEG_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     71.690571  32.32590 12.536614   4.518998 5.748141
#> SD        3.990349  18.43269 10.366218   7.567037 4.434925
#> Median   71.500000  29.85428  9.667805   1.565558 4.714600
#> IQR       5.200000  26.69920 11.893758   5.258956 4.316772
```

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.

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,
                    .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

  # 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 <- ggplot(as_tibble(pca$x), aes(PC1, PC2)) +
  geom_point(alpha = 0.1, 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))
}
```

Raw Values, Unscaled

First, I run a PCA on unscaled values. This highlights the fact that the "optimal" linear combination will depend on the standard errors of the sub-indexes. That means the PCA (like the simple average proposed in the memo) will depend on the units used to express each of the sub-indexes. In this context, that is probably not ideal. IF we are trying to identify disadvantaged communities, it should not matter whether life expectancy is measured in years of months, or whether the unemployment rate is expressed as a percent or a proportion.

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

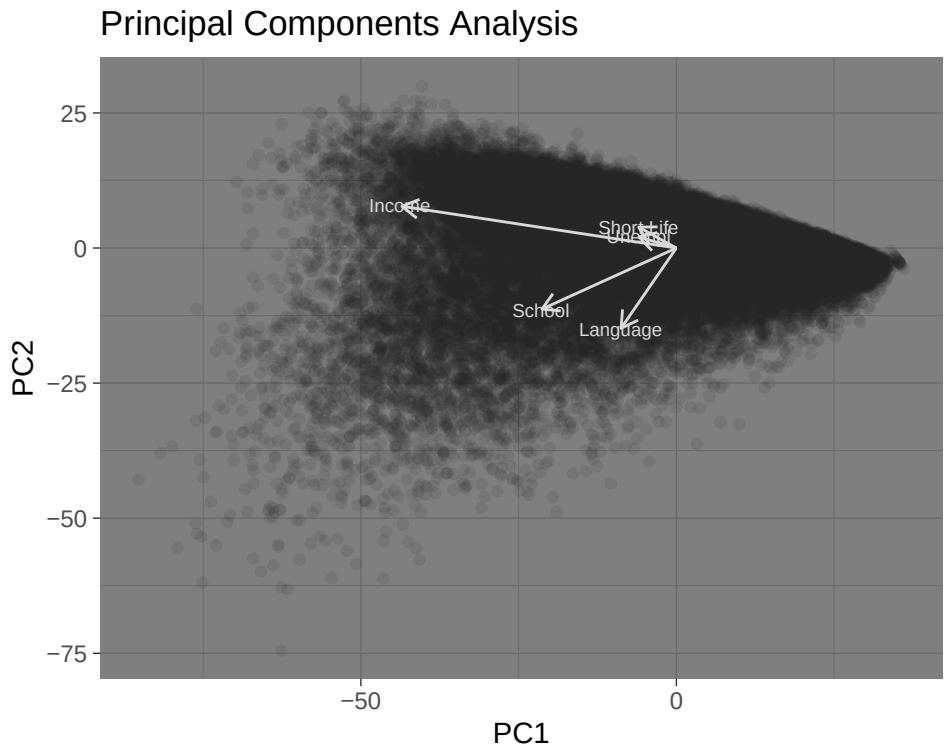
```
summary(pca)
#> Importance of components:
#>                           PC1    PC2     PC3    PC4     PC5
#> Standard deviation     19.9724 8.2667 4.71309 3.3827 2.77545
#> Proportion of Variance  0.7843 0.1344 0.04368 0.0225 0.01515
#> Cumulative Proportion   0.7843 0.9187 0.96235 0.9849 1.00000
```

The first two axes account for 92% of the variation in the sub-indexes.

```
pca$rotation
#>                    PC1         PC2         PC3         PC4         PC5
#> NEG_LIFEEXP -0.1195320  0.18648932 -0.18688676  0.06298960 -0.95500759
#> LOWINCPCT   -0.8711141  0.37363838  0.25792840 -0.14178350  0.12216788
#> LESSHSPCT   -0.4266060 -0.54701100 -0.71387546 -0.04748759  0.08314488
#> LINGISOPCT  -0.1755003 -0.71876600  0.62101006  0.11357685 -0.23242610
#> UNEMPPCT    -0.1186565  0.09884582 -0.05722422  0.98019130  0.11000251
```

The first axis is closely associated with percent low income people in each census Tract. That is because the standard deviation of the income indicator is about double the standard error of the second most variable indicator.

```
plot_pca(pca)
```



What we see is that the principal structure in the sub-indexes (when unscaled) is correlated with income and somewhat correlated with education. Linguistic isolation and education provide a fair amount of independent information via the second PCA Axis.

Scaled PCA

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

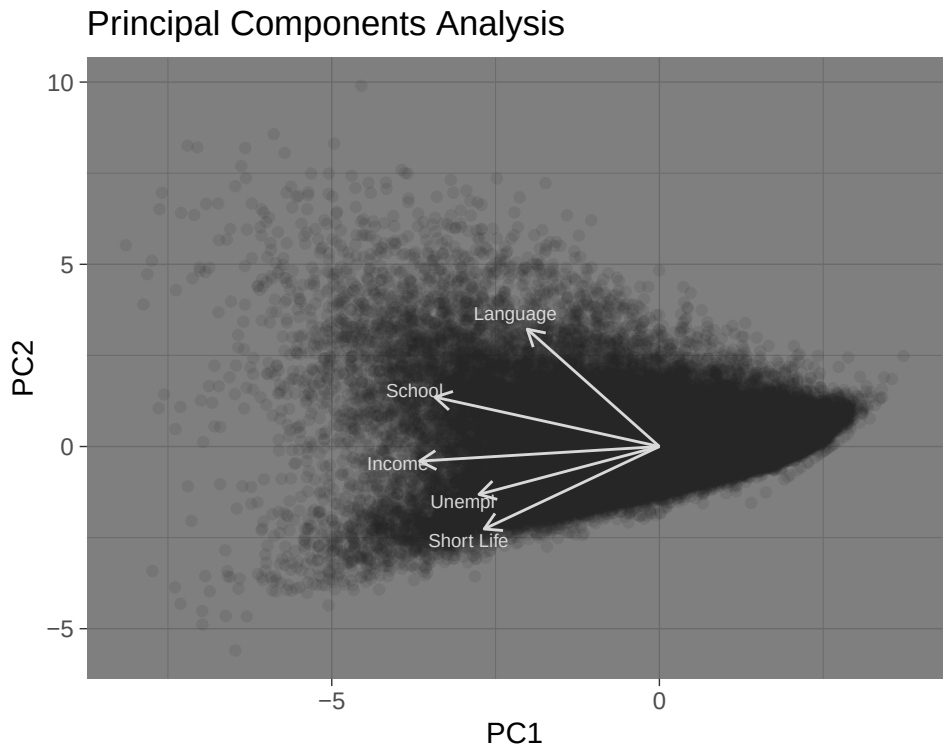
```
pca <- the_data %>%
  select( "NEG_LIFEEXP", "LOWINCPCT", "LESSHSPCT", "LINGISOPCT","UNEMPPCT" ) %>%
  filter(complete.cases(.)) %>%
prcomp(scale. = TRUE)
```

```
summary(pca)
#> Importance of components:
#>                           PC1    PC2    PC3    PC4     PC5
#> Standard deviation     1.6569 1.0958 0.7652 0.5201 0.44476
#> Proportion of Variance 0.5491 0.2402 0.1171 0.0541 0.03956
#> Cumulative Proportion  0.5491 0.7892 0.9063 0.9604 1.00000
```

This explains less of the overall variation in the first two PCA axes, confirming that some of the apparent pattern in the data was driven by the different standard deviations of the predictors. Variables with more variation (Percent low Income and percent not finishing high school) dominate the pattern.

```
pca$rotation
#>                    PC1         PC2         PC3        PC4        PC5
#> NEG_LIFEEXP -0.4033722 -0.51628930  0.50813918 -0.5273230 -0.1856375
#> LOWINCPCT   -0.5523113 -0.09137916  0.12713827  0.3789454  0.7258363
#> LESSHSPCT   -0.5168331  0.30909700  0.16033977  0.4678472 -0.6267000
#> LINGISOPCT  -0.3042858  0.73460142 -0.03571013 -0.5808921  0.1704700
#> UNEMPPCT    -0.4153755 -0.29985832 -0.83585069 -0.1483719 -0.1299521
```

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



When variables are standardized to unit variance, the dominant axis is moderately correlated with all the sub-indexes, especially income, education, and unemployment. 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,] -0.4033722
#> [2,] -0.5523113
#> [3,] -0.5168331
#> [4,] -0.3042858
#> [5,] -0.4153755
```

Percentiles

```
pca <- the_data %>%
  select( "p_NEG_LIFEEXP", "p_LOWINCPCT", "p_LESSHSPCT",
          "p_LINGISOPCT","p_UNEMPPCT" ) %>%
  filter(complete.cases(.)) %>%
prcomp(scale. = FALSE)
```

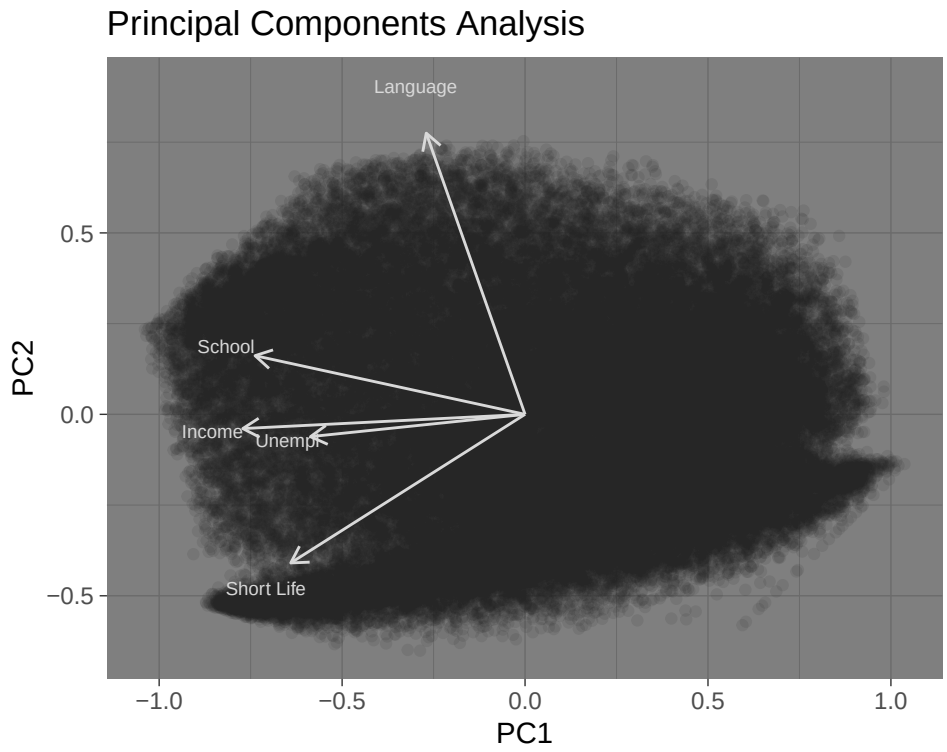
```
summary(pca)
#> Importance of components:
#>                           PC1    PC2    PC3     PC4     PC5
#> Standard deviation     0.4675 0.2981 0.2263 0.17086 0.12700
#> Proportion of Variance 0.5410 0.2200 0.1268 0.07228 0.03993
#> Cumulative Proportion  0.5410 0.7610 0.8878 0.96007 1.00000
```

The first two axes account for only 76% of the pattern in the sub-indexes.

```
pca$rotation
#>                       PC1         PC2         PC3         PC4         PC5
#> p_NEG_LIFEEXP -0.4566142 -0.45795132  0.33544091  0.66663870  0.15765884
#> p_LOWINCPCT   -0.5501492 -0.04387566  0.15713741 -0.30641563 -0.75949201
#> p_LESSHSPCT   -0.5262234  0.18074885  0.23324817 -0.50048135  0.62091265
#> p_LINGISOPCT  -0.1925582  0.86658098  0.06545998  0.44895830 -0.07816767
#> p_UNEMPPCT    -0.4181503 -0.06872241 -0.89671484  0.09827091  0.08168762
```

Here, linguistics plays little role in Axis 1, but it dominates Axis 2. It is interesting that pecentile of (negative) life expectancy decreases the axis 2 score, while linguistic isolation sharply increases it.

```
plot_pca(pca, .scale = 3, .ann_space = 0.05,
         .levels = c( "p_NEG_LIFEEXP", "p_LOWINCPCT", "p_LESSHSPCT",
                      "p_LINGISOPCT","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 lifespan.

Collectively, the first two PCA axes explain roughly three quarters of the pattern.

```
the_data$PCA_Index_V2 <- calc_scores(the_data, pca)
#>             [,1]
#> [1,] -0.4566142
#> [2,] -0.5501492
#> [3,] -0.5262234
#> [4,] -0.1925582
#> [5,] -0.4181503
```

Examining Results

Correlations

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

```
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     0.625     0.385     -0.053    0.423   0.586   0.668
```

```
#> LOWINCPCT         0.625     1.000     0.689      0.340    0.556   0.935   0.870
#> LESSHSPCT         0.385     0.689     1.000      0.594    0.374   0.883   0.790
#> LINGISOPCT       -0.053     0.340     0.594      1.000    0.113   0.578   0.454
#> UNEMPPCT          0.423     0.556     0.374      0.113    1.000   0.600   0.642
#> Index_1           0.586     0.935     0.883      0.578    0.600   1.000   0.927
#> Index_2           0.668     0.870     0.790      0.454    0.642   0.927   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     0.632     0.502     -0.077    0.382   0.642   0.686
#> LOWINCPCT         0.632     1.000     0.747      0.233    0.527   0.953   0.878
#> LESSHSPCT         0.502     0.747     1.000      0.369    0.443   0.887   0.865
#> LINGISOPCT       -0.077     0.233     0.369      1.000    0.144   0.364   0.444
#> UNEMPPCT          0.382     0.527     0.443      0.144    1.000   0.581   0.688
#> Index_1           0.642     0.953     0.887      0.364    0.581   1.000   0.962
#> Index_2           0.686     0.878     0.865      0.444    0.688   0.962   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   0.927     0.955     0.915       -0.997       -0.928
#> Index_2        0.927   1.000     0.958     0.996       -0.927       -0.987
#> p_Index_1      0.955   0.958     1.000     0.962       -0.960       -0.969
#> p_Index_2      0.915   0.996     0.962     1.000       -0.916       -0.986
#> PCA_Index_V1  -0.997  -0.927    -0.960    -0.916        1.000        0.936
#> PCA_Index_V2  -0.928  -0.987    -0.969    -0.986        0.936        1.000
```

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

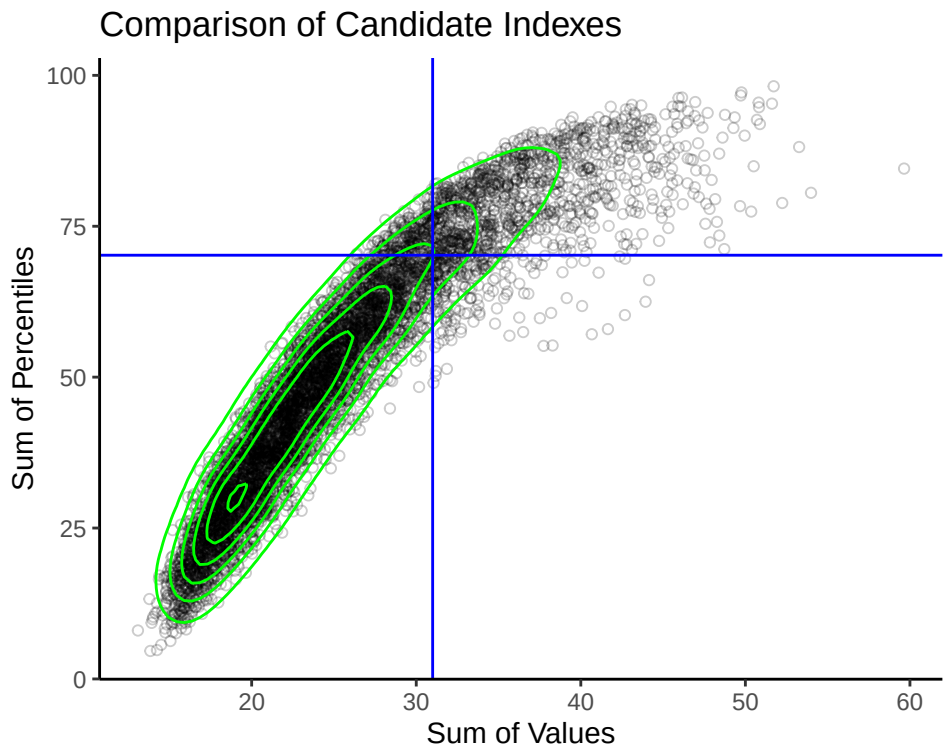
Rank correlations are even higher.

```
the_data %>%
select(c(Index_1, Index_2, PCA_Index_V1, PCA_Index_V2)) %>%
  cor(use = 'pairwise', method = 'spearman')  %>%
  round(3)
#>              Index_1 Index_2 PCA_Index_V1 PCA_Index_V2
#> Index_1        1.000   0.962       -0.998       -0.971
#> Index_2        0.962   1.000       -0.953       -0.988
#> PCA_Index_V1  -0.998  -0.953        1.000        0.969
#> PCA_Index_V2  -0.971  -0.988        0.969        1.000
```

Graphics

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

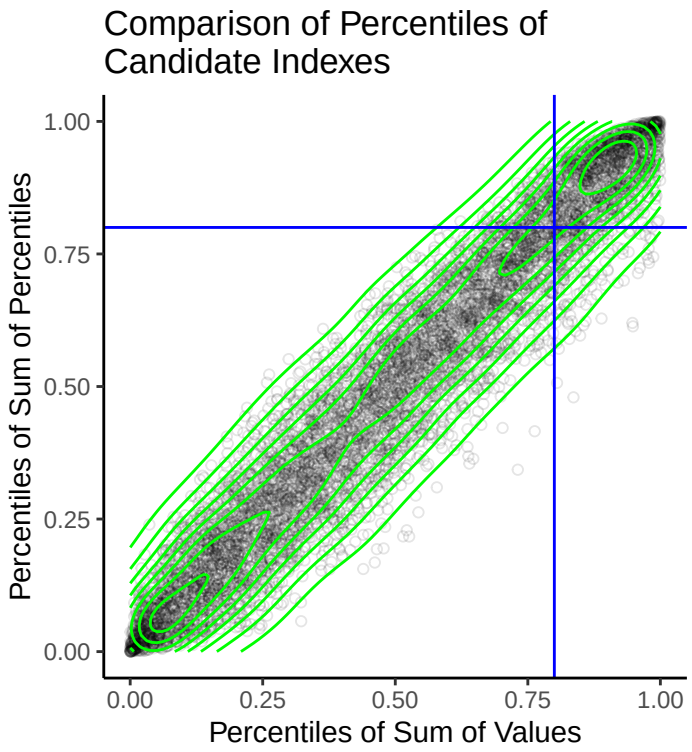
```
p80_sum <- quantile(the_data$Index_1, 0.8, na.rm = TRUE)
p80_sum_of_p  <- quantile(the_data$Index_2, 0.8, na.rm = TRUE)

plt <- the_data %>%
  slice_sample(prop = 0.1) %>%
  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( 'Sum of Values') +
  ylab( 'Sum of Percentiles') +
  ggtitle('Comparison of Candidate Indexes')
plt
#> Warning: Removed 585 rows containing non-finite values (stat_density2d).
#> Warning: Removed 585 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.

The relationship between the percentiles of 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 = 0.1) %>%
  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 Sum of Values') +
  ylab( 'Percentiles of Sum of Percentiles') +
  ggtitle('Comparison of Percentiles of\nCandidate Indexes') +
  coord_fixed()
plt
#> Warning: Removed 587 rows containing non-finite values (stat_density2d).
#> Warning: Removed 587 rows containing missing values (geom_point).
```



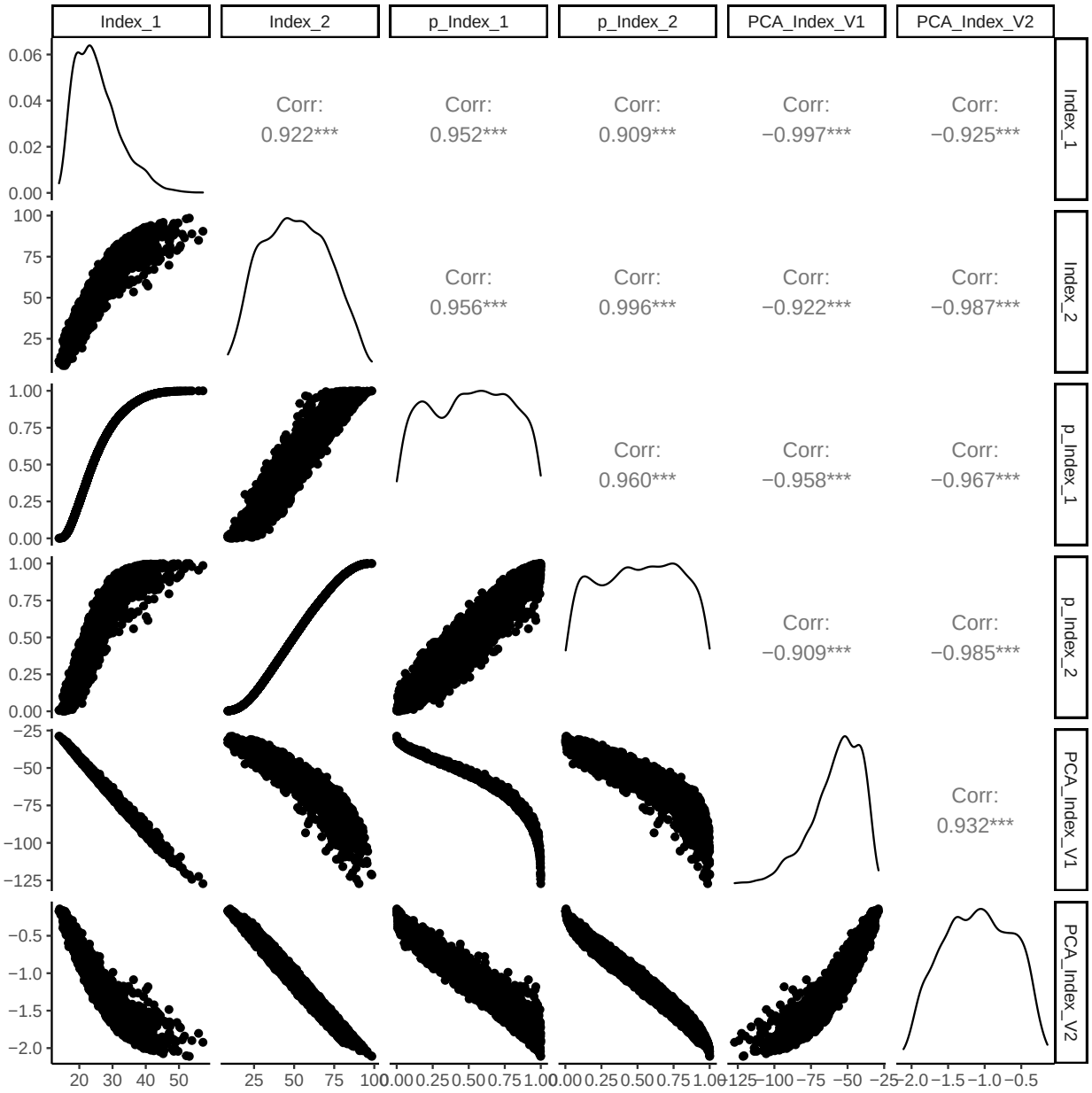
Pairs Plot of All Indexes

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

```
the_data %>%
select(c(Index_1:PCA_Index_V2)) %>%
  slice_sample(prop = 0.05, replace = FALSE) %>%
  ggpairs(progress = FALSE)
#> Warning: Removed 304 rows containing non-finite values (stat_density).
```

```
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values
#> Warning: Removed 304 rows containing missing values (geom_point).
#> Warning: Removed 304 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values
#> Warning: Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Warning: Removed 304 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values
#> Warning: Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Warning: Removed 304 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values

#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
#> Removed 304 rows containing missing values
#> Warning: Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Warning: Removed 304 rows containing non-finite values (stat_density).
#> Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
```

```
#> Removed 304 rows containing missing values
#> Warning: Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Removed 304 rows containing missing values (geom_point).
#> Warning: Removed 304 rows containing non-finite values (stat_density).
```



A few observations:

- Taking percentiles has the effect of spreading out the tails of the distributions.
- The two PCA indexes are highly correlated with their

- Generally, correlations are highest within "Group 1" (raw data) or "Group 2" (percentiles).
- Correlations are high enough between the PCA-based nd simple average based index so as to make little practical difference. There is probably no reason to continue to consider the PCA based metrics (and if there were, I'd want to explore data transformations).

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