Poverty Data - Principal Components Analysis

Michael Bemus

2024-10-01

This is the first of 3 Methods I am going to try to compare the classes of our data set. Principal components analysis is typically used to leverage the covariance structure of data to redefine new variables from a combination of those in the data set.

For our purposes, we will be creating four sets of Principal Components: 1 using the full data set, 1 using the Poor Group, 1 using the Aid Group, and 1 using the No Group. We will then use a few comparison methods to see how similar each set of components are to each other.

Load Data and Packages

```
df <- read.csv("SubsidyClass/dataset.csv")
```

First, we can remove a few extra variables that will not be helpful for our Analysis. First, the official and supplemental poverty variables can be removed because they are already represented in our group variable. Additionally, we can remove the State variable because it has too many categories to be very useful.

Also, to avoid bias, we will remove our subsidy variables like slunch and SNAP. These variables were used to define our groups of interest, and for the no group, all values are 0. They would cause errors in our analysis if we kept them, so we must remove them.

```
df <- select(df, -X, -st, -off_pov, -spm_pov, -snap, -house_sub, -slunch,
             -energy, -wic)   # Remove unnecessary variables.
```

Next, we know that PCA does not work well with categorical data that has 3 or more classes. Because of this, we need to encode dummy variables so that a correlation structure can be created for our classes.

```
# Starting with the mar variables. Our categories are:
#   Divorced
df$divorced[df$mar == "D"] <- 1
df$divorced[df$mar != "D"] <- 0

#   Married
df$married[df$mar == "M"] <- 1
df$married[df$mar != "M"] <- 0

#   Separated
df$separated[df$mar == "S"] <- 1
df$separated[df$mar != "S"] <- 0

#   Widowed
df$widowed[df$mar == "W"] <- 1
df$widowed[df$mar != "W"] <- 0

#   And, by default, an observation is Never Married.
df <- select(df, -mar)

# Next, we create variables for our Mortgage classes.
#   Owns a property with a Mortgage.
df$has_mortgage[df$mortgage == "M"] <- 1
df$has_mortgage[df$mortgage != "M"] <- 0

#   Renting a property.
df$renting[df$mortgage == "R"] <- 1
df$renting[df$mortgage != "R"] <- 0

#   By default, an observation owns a property and has no mortgage.
df <- select(df, -mortgage)

# Next, we look at our education variables. Their classes are
#   High School degree with some College.
df$less_college[df$edu == "<C"] <- 1
df$less_college[df$edu != "<C"] <- 0

#   College degree.
df$college[df$edu == "C"] <- 1
df$college[df$edu != "C"] <- 0

#   High School Degree.
df$highsch[df$edu == "HS"] <- 1
df$highsch[df$edu != "HS"] <- 0

#   By default, an observation has less than a high school education.
df <- select(df, -edu)   # Default: <HS

# Next, we look at our sex variable.
# For our purposes, a 1 represents Female.
df$sex[df$sex == "F"] <- 1
# A 0 represents a male.
```

```
df$sex[df$sex == "M"] <- 0

df$sex <- as.integer(df$sex)   # Validate data type.

# Last, we examine our Race Variable. Our classes are:
#   Asian
df$race_asian[df$race == "A"] <- 1
df$race_asian[df$race != "A"] <- 0

#   Black
df$race_black[df$race == "B"] <- 1
df$race_black[df$race != "B"] <- 0

#   Other
df$race_other[df$race == "O"] <- 1
df$race_other[df$race != "O"] <- 0

#   By default, an observation is white.
df <- select(df, -race)

# Check the end result of our process.
head(df)
```

```
##   age sex hispanic    agi hi_prem moop num_kid num_adlt spm_res spm_inc fed_tax
## 1  79   1        0  60300     558  595       0        2  106969  148000   17128
## 2  59   0        0  85000    1297 3243       0        2  106969  148000   17128
## 3  62   1        0  73550     108  700       0        2   70241   83600    5145
## 4  66   0        1  73550    5000 3000       0        2   70241   83600    5145
## 5  60   0        0 106900    3632  400       0        3   85352  126000   10878
## 6  69   1        0      0     525  500       0        2   28866   28600       0
##   fed_tax_bc eitc  fica st_tax cap_xpen wk_xpen cc_xpen spm_hi_prem med_xpen
## 1      17128    0  8033   4912     3225    3225       0        1855     7733
## 2      17128    0  8033   4912     3225    3225       0        1855     7733
## 3       5145    0     0    -81        0       0       0        2554     8295
## 4       5145    0     0    -81        0       0       0        2554     8295
## 5      10878    0 10168   4592     3411    3411       0       10241    11599
## 6          0    0     0      0        0       0       0         411     1031
##   mc_pb cohabit ui_kids group divorced married separated widowed has_mortgage
## 1  2041       0       0    no        0       0         0       1            1
## 2     0       0       0    no        0       0         0       0            1
## 3     0       0       0    no        0       1         0       0            0
## 4  2041       0       0    no        0       1         0       0            0
## 5     0       0       0    no        0       1         0       0            1
## 6     0       0       0   aid        0       1         0       0            1
##   renting less_college college highsch race_asian race_black race_other
## 1       0           1       0       0          0          1          0
## 2       0           0       1       0          0          1          0
## 3       0           0       0       1          0          0          0
## 4       0           0       0       1          0          0          1
## 5       0           0       0       1          0          0          0
## 6       0           0       0       0          0          0          0
```

Principal Components

Now that we have processed our data, we can create our Principal Components.

Full Data Components

We will start with the full data's set of Principal Components.

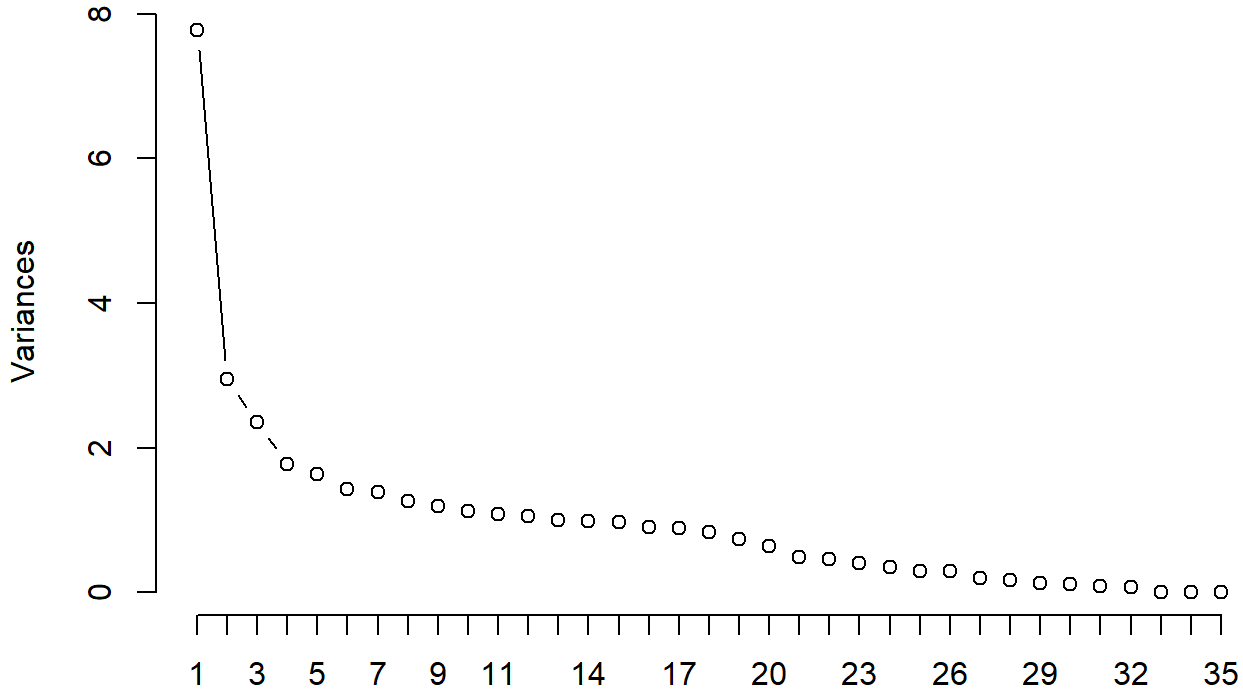
```
# Use the prcomp function from the built-in R Stats library.
#   The "scale.=TRUE" option assures we are using correlations, not covariances.
pca_full <- prcomp(select(df, -group), scale.=TRUE)

# View the relative importance of each PC we've created.
summary(pca_full)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.787 1.71677 1.53194 1.33134 1.27696 1.19592 1.17848
## Proportion of Variance 0.222 0.08421 0.06705 0.05064 0.04659 0.04086 0.03968
## Cumulative Proportion  0.222 0.30616 0.37321 0.42385 0.47044 0.51131 0.55099
##                           PC8     PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     1.12064 1.09485 1.06198 1.0383 1.02440 1.00073 0.99082
## Proportion of Variance 0.03588 0.03425 0.03222 0.0308 0.02998 0.02861 0.02805
## Cumulative Proportion  0.58687 0.62111 0.65334 0.6841 0.71412 0.74274 0.77078
##                           PC15    PC16    PC17   PC18    PC19    PC20    PC21
## Standard deviation     0.98272 0.95007 0.94436 0.9108 0.85559 0.80166 0.70071
## Proportion of Variance 0.02759 0.02579 0.02548 0.0237 0.02092 0.01836 0.01403
## Cumulative Proportion  0.79838 0.82417 0.84965 0.8733 0.89427 0.91263 0.92666
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.67645 0.63141 0.58539 0.54474 0.54158 0.44836 0.40358
## Proportion of Variance 0.01307 0.01139 0.00979 0.00848 0.00838 0.00574 0.00465
## Cumulative Proportion  0.93973 0.95112 0.96091 0.96939 0.97777 0.98351 0.98817
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.35379 0.34166 0.30220 0.27555 0.06221 0.02757 0.01986
## Proportion of Variance 0.00358 0.00334 0.00261 0.00217 0.00011 0.00002 0.00001
## Cumulative Proportion  0.99174 0.99508 0.99769 0.99986 0.99997 0.99999 1.00000
```

We see that it takes 16 components to explain 80% of the Variance present in our data. To get to 90%, we need 20 Components. Otherwise, we have a total of 35 Components to explain the full variance.

We can visualize the importance of our components using a scree plot.

```
library(graphics)

# Use the standard deviations of our model to create a scree plot.
screeplot(pca_full["sdev"], npcs=35, type="lines")
```



We see that, after about 7 components, the variance explained by our components really starts to flatten out, before another drop starting at 18 Components. In normal applications of PCA's, we might consider using only the first 4 or 5 Components as a new data set for further analysis.

However, the purpose of our analysis is not to reduce the number of variables. Instead, we want to compare the features of our components. To do this, we will first examine what features are important for our current set of components using a custom function.

```
# Initialize Function.
important_components <- function(pc) {
  # First, we need a threshold we can use to determine if a component is
  # significant enough to be observed.

  # We will use (Number of components)^-1 as our threshold.
  signf <- 1/sqrt(nrow(pc$rotation))
  # If a variable has a significance greater than signf, we will examine it.

  # Next, we want to split our data so that we are only looking at components
  # that contribute the most to explaining 80% of the variance in our data.

  # 1. Find the Variances of each Component.
  lambda <- pc$sdev**2

  # 2. Find what percent of the total variance each Component explains.
  varPct <- lambda/sum(lambda) * 100

  # 3. Create a list of cumulative variance explained by combining components.
  cumvarPct <- cumsum(varPct)

  # 4. Select indecies of all components contributing to 80% of the Variance.
  split80 <- c(1:length(lambda))[cumvarPct == cumvarPct[cumvarPct > 80][1]]

  # Whenever we run this function, we want to print some basic information.
  print("--------------------------------")
  for (i in 1:split80) {   # For each significant variable.

    # Select the indexes of those variables with significant contribution.
    signf_ind <- abs(pc$rotation[,i]) > signf
    # Select the names of those variables.
    signf_rows <- rownames(pc$rotation)[signf_ind]
    # Select the values of those variables.
    signf_vals <- pc$rotation[signf_ind, i]

    # Principal components allow variables to be positive and negative.
    # However, to compare significance, we want all values to be positive.
    signf_abs <- abs(signf_vals)

    # We create a data frame from the above to simplify a few operations.
    signf_df <- data.frame("Variable"=signf_rows,
                           "Value"=signf_vals,
                           "Abs"=signf_abs)

    # Order our variables based on their contribution to the component.
    signf_df <- signf_df[order(signf_df$Abs, decreasing=TRUE),]

    # Print which component number we are on and the variance it explains.
    print(paste("PC", i, ": ", varPct[i], sep=""))
    print("")   # Extra space.

    # Then, for each significant variable, we want to print its value.
```

```
    for (j in 1:nrow(signf_df)) {
      print(paste(signf_df$Variable[j], ": ", round(signf_df$Value[j],4),
                  sep=""))
    }
    # Move on to next component.
    print("--------------------------------")
  }

  # Once we've printed everything out, we want to be able to compare our
  # important components between sets of PCA's.

  # T/F object storing when variables are significant.
  is_signf <- abs(pc$rotation[,1:split80]) > signf

  # Create a data frame to store our outputs.
  signf_counts <- data.frame()

  # For each variable...
  for (i in 1:nrow(pc$rotation)) {
    signf_count <- sum(is_signf[i,])   # Count the number of times it's flagged.

    # Add variable to our output data frame.
    signf_counts <- rbind(signf_counts,
                          c(rownames(pc$rotation)[i], signf_count))
  }
  # Rename our columns for cleanliness.
  colnames(signf_counts) <- c("Variable", "Count")
  signf_counts$Count <- as.integer(signf_counts$Count)   # Data type validation.

  return(signf_counts)   # Output results data frame.
}
```

Now that we have this function, we can use it in our analysis of the full data principal components.

```
sig_full <- important_components(pca_full)
```

```
## [1] "------------------------------"
## [1] "PC1: 22.1948910571901"
## [1] ""
## [1] "spm_inc: 0.3378"
## [1] "spm_res: 0.3272"
## [1] "fica: 0.3224"
## [1] "fed_tax_bc: 0.316"
## [1] "agi: 0.3013"
## [1] "fed_tax: 0.2989"
## [1] "st_tax: 0.2758"
## [1] "wk_xpen: 0.2511"
## [1] "cap_xpen: 0.2497"
## [1] "------------------------------"
## [1] "PC2: 8.4208952234975"
## [1] ""
## [1] "age: -0.3777"
## [1] "mc_pb: -0.3563"
## [1] "cap_xpen: 0.2908"
## [1] "wk_xpen: 0.2842"
## [1] "hispanic: 0.2748"
## [1] "race_other: 0.2632"
## [1] "num_kid: 0.2331"
## [1] "eitc: 0.2287"
## [1] "renting: 0.2028"
## [1] "med_xpen: -0.1803"
## [1] "fed_tax: -0.1755"
## [1] "------------------------------"
## [1] "PC3: 6.70527047371638"
## [1] ""
## [1] "med_xpen: 0.4785"
## [1] "spm_hi_prem: 0.4097"
## [1] "hi_prem: 0.3192"
## [1] "num_adlt: 0.3138"
## [1] "married: 0.2121"
## [1] "moop: 0.2042"
## [1] "fed_tax: -0.1968"
## [1] "fed_tax_bc: -0.1868"
## [1] "college: -0.1781"
## [1] "renting: -0.1773"
## [1] "------------------------------"
## [1] "PC4: 5.06417263352114"
## [1] ""
## [1] "married: -0.4947"
## [1] "divorced: 0.3671"
## [1] "cohabit: 0.352"
## [1] "spm_hi_prem: 0.3413"
## [1] "hi_prem: 0.2695"
## [1] "renting: 0.225"
## [1] "med_xpen: 0.1979"
## [1] "------------------------------"
## [1] "PC5: 4.65889744025421"
## [1] ""
```

```
## [1] "race_other: 0.5522"
## [1] "hispanic: 0.5422"
## [1] "age: 0.196"
## [1] "num_kid: -0.1941"
## [1] "race_black: -0.1924"
## [1] "cc_xpen: -0.1806"
## [1] "has_mortgage: -0.1774"
## [1] "------------------------------"
## [1] "PC6: 4.08638740356846"
## [1] ""
## [1] "less_college: -0.6418"
## [1] "college: 0.5079"
## [1] "renting: 0.2204"
## [1] "has_mortgage: -0.1971"
## [1] "hi_prem: 0.1847"
## [1] "race_asian: 0.1759"
## [1] "------------------------------"
## [1] "PC7: 3.96801403056669"
## [1] ""
## [1] "highsch: 0.6629"
## [1] "less_college: -0.4277"
## [1] "num_adlt: 0.2208"
## [1] "hi_prem: -0.216"
## [1] "college: -0.2005"
## [1] "race_other: -0.1898"
## [1] "widowed: 0.1778"
## [1] "hispanic: -0.1696"
## [1] "------------------------------"
## [1] "PC8: 3.58808150146404"
## [1] ""
## [1] "has_mortgage: 0.5303"
## [1] "renting: -0.4885"
## [1] "divorced: 0.2963"
## [1] "race_black: -0.2556"
## [1] "college: 0.2362"
## [1] "less_college: -0.2214"
## [1] "------------------------------"
## [1] "PC9: 3.42487823851717"
## [1] ""
## [1] "widowed: -0.5511"
## [1] "sex: -0.5132"
## [1] "highsch: 0.2398"
## [1] "race_asian: -0.2379"
## [1] "eitc: -0.2267"
## [1] "divorced: 0.208"
## [1] "cc_xpen: -0.2079"
## [1] "num_kid: -0.1739"
## [1] "------------------------------"
## [1] "PC10: 3.22230432500555"
## [1] ""
## [1] "race_asian: 0.4033"
## [1] "num_adlt: 0.3499"
```

```
## [1] "cc_xpen: -0.348"
## [1] "cohabit: 0.2937"
## [1] "hi_prem: -0.2909"
## [1] "widowed: -0.2595"
## [1] "highsch: -0.2529"
## [1] "college: 0.2018"
## [1] "separated: -0.1875"
## [1] "race_black: -0.1855"
## [1] "------------------------------"
## [1] "PC11: 3.08012079147768"
## [1] ""
## [1] "ui_kids: 0.6384"
## [1] "num_kid: 0.3616"
## [1] "eitc: 0.3538"
## [1] "race_asian: -0.2191"
## [1] "moop: 0.1861"
## [1] "separated: 0.1745"
## [1] "widowed: -0.171"
## [1] "------------------------------"
## [1] "PC12: 2.99829715207118"
## [1] ""
## [1] "cc_xpen: 0.5905"
## [1] "separated: -0.5021"
## [1] "has_mortgage: -0.2632"
## [1] "race_black: -0.251"
## [1] "divorced: 0.2288"
## [1] "moop: 0.1944"
## [1] "renting: 0.183"
## [1] "------------------------------"
## [1] "PC13: 2.86133478877925"
## [1] ""
## [1] "race_black: -0.5648"
## [1] "race_asian: 0.484"
## [1] "ui_kids: 0.3067"
## [1] "college: -0.2635"
## [1] "highsch: 0.1777"
## [1] "hi_prem: 0.1734"
## [1] "widowed: 0.1707"
## [1] "------------------------------"
## [1] "PC14: 2.80490712171012"
## [1] ""
## [1] "separated: -0.7132"
## [1] "cc_xpen: -0.3761"
## [1] "sex: 0.2925"
## [1] "moop: -0.1781"
## [1] "cohabit: -0.1747"
## [1] "hi_prem: 0.1722"
## [1] "------------------------------"
## [1] "PC15: 2.75925443399681"
## [1] ""
## [1] "moop: 0.594"
## [1] "sex: 0.4437"
```

```
## [1] "ui_kids: -0.268"
## [1] "widowed: -0.2455"
## [1] "divorced: 0.1948"
## [1] "highsch: 0.187"
## [1] "spm_hi_prem: -0.1865"
## [1] "hi_prem: -0.1727"
## [1] "--------------------------------"
## [1] "PC16: 2.57895879014982"
## [1] ""
## [1] "eitc: -0.5686"
## [1] "ui_kids: 0.5374"
## [1] "num_kid: -0.2796"
## [1] "moop: 0.2525"
## [1] "sex: 0.2039"
## [1] "--------------------------------"
```

From this output, we can look at each component individually to create a summary of what it represents. For example, the first component appears to contain our tax and income variables, whereas our second begins to account for age and variables important to retirees.

Here, I will not go into a full analysis of our results. However, I will examine the significance counts we also calculated as outputs of our function.

```
sig_full[order(sig_full$Count, decreasing=TRUE),]
```

```
##        Variable Count
## 5       hi_prem     8
## 6          moop     6
## 27      widowed     6
## 29      renting     6
## 31      college     6
## 7       num_kid     5
## 18      cc_xpen     5
## 24     divorced     5
## 32      highsch     5
## 33   race_asian     5
## 34   race_black     5
## 2            sex     4
## 13         eitc     4
## 23      ui_kids     4
## 26    separated     4
## 28 has_mortgage     4
## 3      hispanic     3
## 8      num_adlt     3
## 11      fed_tax     3
## 19  spm_hi_prem     3
## 20     med_xpen     3
## 22      cohabit     3
## 30 less_college     3
## 35   race_other     3
## 1            age     2
## 12   fed_tax_bc     2
## 16     cap_xpen     2
## 17      wk_xpen     2
## 25      married     2
## 4            agi     1
## 9       spm_res     1
## 10      spm_inc     1
## 14         fica     1
## 15       st_tax     1
## 21        mc_pb     1
```

We see that, of our 80% Components, hi_prem is used the most frequently. However, this begs the question of whether hi_prem is contributing more or less than other variables to the components it is featured in.

In comparison, agi, spm_res, spm_inc, and fica, st_tax, and mc_pb are used the least. It appears that most of their variance is explained by the first two components, which means they are not required for the other components.

Aid Data Components

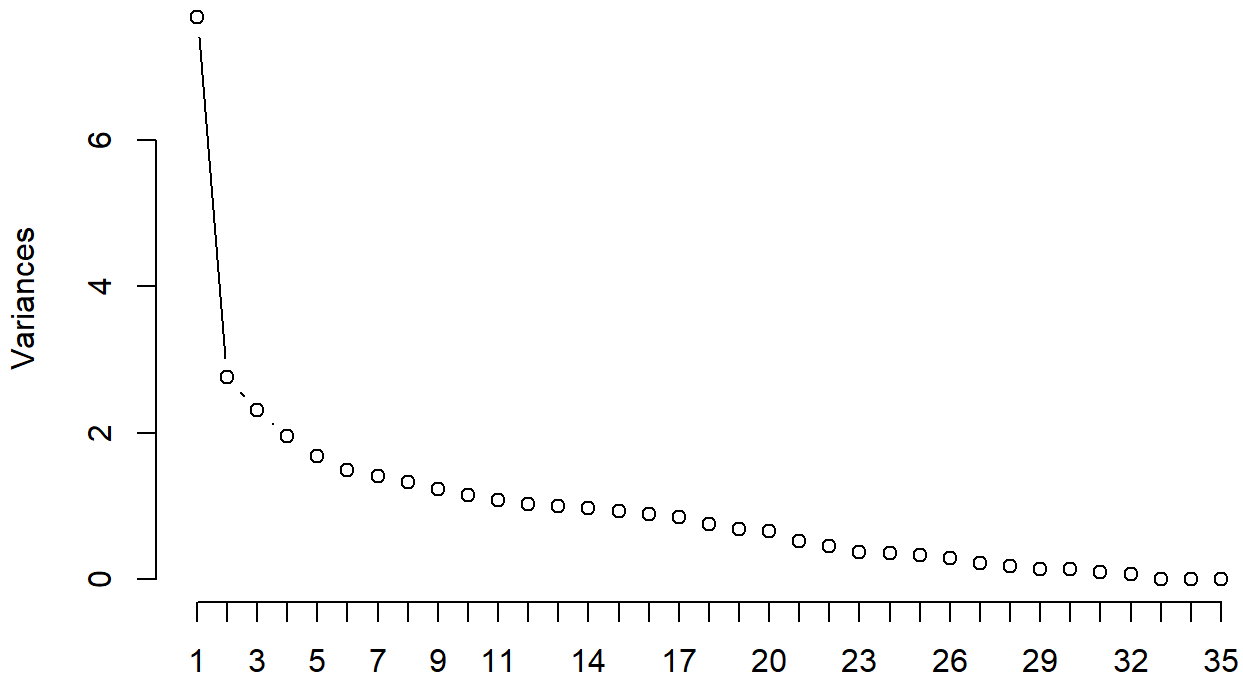
Next, we will repeat this analysis for the variables in our aid group.

```
# Define our components as before, selecting only aid group observations.
pca_aid <- prcomp(select(filter(df, group == "aid"), -group), scale.=TRUE)

# View the relative importance of our new components.
summary(pca_aid)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.7695 1.6597 1.5188 1.39827 1.29905 1.22289 1.18655
## Proportion of Variance 0.2191 0.0787 0.0659 0.05586 0.04822 0.04273 0.04023
## Cumulative Proportion  0.2191 0.2978 0.3638 0.41961 0.46783 0.51055 0.55078
##                           PC8     PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     1.15210 1.10948 1.07166 1.0383 1.01622 0.99774 0.98296
## Proportion of Variance 0.03792 0.03517 0.03281 0.0308 0.02951 0.02844 0.02761
## Cumulative Proportion  0.58870 0.62387 0.65669 0.6875 0.71699 0.74543 0.77304
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.96642 0.94188 0.92233 0.86480 0.83171 0.81215 0.71819
## Proportion of Variance 0.02668 0.02535 0.02431 0.02137 0.01976 0.01885 0.01474
## Cumulative Proportion  0.79972 0.82507 0.84938 0.87074 0.89051 0.90935 0.92409
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.67348 0.61390 0.60112 0.57800 0.53510 0.47130 0.42172
## Proportion of Variance 0.01296 0.01077 0.01032 0.00955 0.00818 0.00635 0.00508
## Cumulative Proportion  0.93705 0.94782 0.95814 0.96769 0.97587 0.98221 0.98730
##                           PC29    PC30    PC31   PC32    PC33    PC34    PC35
## Standard deviation     0.37214 0.36536 0.31011 0.2580 0.08456 0.04523 0.02698
## Proportion of Variance 0.00396 0.00381 0.00275 0.0019 0.00020 0.00006 0.00002
## Cumulative Proportion  0.99125 0.99507 0.99781 0.9997 0.99992 0.99998 1.00000
```

We see that our first component here has a slightly smaller standard deviation than it had in our first set. It still takes 16 components to explain 80% of the variance and 20 componens to explain 90%. However, our 20th component brings our cumulative variance proportion to about 91%, whereas it was 90% earlier. With this smaller sample, we see that there is a wider spread of the variance across variables.

```
# Visualizing the above results.
screeplot(pca_aid["sdev"], npcs=35, type="lines")
```



The scree plot shown here is very similar to what we saw with the full data. There are likely only small changes of degree rather than total structural changes in the overall correlation structure.

Next, we will use our function to analyze the components.

```
sig_aid <- important_components(pca_aid)
```

```
## [1] "------------------------------"
## [1] "PC1: 21.9139760048097"
## [1] ""
## [1] "spm_inc: 0.3402"
## [1] "spm_res: 0.3299"
## [1] "fica: 0.3258"
## [1] "fed_tax_bc: 0.3192"
## [1] "agi: 0.3041"
## [1] "fed_tax: 0.2924"
## [1] "st_tax: 0.2712"
## [1] "wk_xpen: 0.2135"
## [1] "cap_xpen: 0.2111"
## [1] "------------------------------"
## [1] "PC2: 7.87033392200631"
## [1] ""
## [1] "cap_xpen: -0.3547"
## [1] "wk_xpen: -0.3543"
## [1] "age: 0.3047"
## [1] "hispanic: -0.2702"
## [1] "race_other: -0.2551"
## [1] "num_adlt: -0.2542"
## [1] "mc_pb: 0.2478"
## [1] "fed_tax: 0.2453"
## [1] "eitc: -0.2194"
## [1] "num_kid: -0.2141"
## [1] "fed_tax_bc: 0.184"
## [1] "widowed: 0.1805"
## [1] "college: 0.17"
## [1] "------------------------------"
## [1] "PC3: 6.59047303203238"
## [1] ""
## [1] "med_xpen: -0.5269"
## [1] "spm_hi_prem: -0.4576"
## [1] "hi_prem: -0.2857"
## [1] "age: -0.2775"
## [1] "num_adlt: -0.2743"
## [1] "moop: -0.2343"
## [1] "mc_pb: -0.2262"
## [1] "num_kid: 0.1872"
## [1] "------------------------------"
## [1] "PC4: 5.58619192826868"
## [1] ""
## [1] "hispanic: 0.3667"
## [1] "race_other: 0.3622"
## [1] "married: -0.3294"
## [1] "hi_prem: -0.247"
## [1] "has_mortgage: -0.2421"
## [1] "renting: 0.2324"
## [1] "cohabit: 0.2279"
## [1] "fed_tax: 0.2167"
## [1] "num_kid: -0.2054"
## [1] "num_adlt: 0.1875"
```

```
## [1] "divorced: 0.1702"
## [1] "------------------------------"
## [1] "PC5: 4.82153720900471"
## [1] ""
## [1] "cohabit: 0.4011"
## [1] "divorced: 0.3678"
## [1] "married: -0.3655"
## [1] "age: -0.2849"
## [1] "mc_pb: -0.2655"
## [1] "hispanic: -0.2354"
## [1] "race_other: -0.2344"
## [1] "renting: 0.2166"
## [1] "num_adlt: -0.1888"
## [1] "widowed: -0.1876"
## [1] "race_black: 0.1871"
## [1] "------------------------------"
## [1] "PC6: 4.27275731103501"
## [1] ""
## [1] "race_other: -0.414"
## [1] "hispanic: -0.38"
## [1] "hi_prem: -0.3109"
## [1] "race_black: 0.2986"
## [1] "cap_xpen: 0.2855"
## [1] "num_adlt: 0.2577"
## [1] "wk_xpen: 0.2251"
## [1] "cc_xpen: 0.2076"
## [1] "moop: -0.2043"
## [1] "highsch: 0.1801"
## [1] "------------------------------"
## [1] "PC7: 4.02255847902982"
## [1] ""
## [1] "less_college: -0.7339"
## [1] "highsch: 0.4402"
## [1] "college: 0.3015"
## [1] "has_mortgage: -0.2014"
## [1] "renting: 0.2008"
## [1] "------------------------------"
## [1] "PC8: 3.79241588945577"
## [1] ""
## [1] "renting: -0.5194"
## [1] "has_mortgage: 0.5033"
## [1] "cohabit: 0.2779"
## [1] "divorced: 0.2695"
## [1] "highsch: 0.2434"
## [1] "less_college: -0.2392"
## [1] "ui_kids: 0.2359"
## [1] "race_black: -0.2041"
## [1] "------------------------------"
## [1] "PC9: 3.51698760059953"
## [1] ""
## [1] "college: 0.4702"
## [1] "sex: 0.418"
```

```
## [1] "highsch: -0.4081"
## [1] "cc_xpen: 0.3886"
## [1] "widowed: 0.2443"
## [1] "cap_xpen: 0.2189"
## [1] "race_asian: 0.1871"
## [1] "------------------------------"
## [1] "PC10: 3.28130118061476"
## [1] ""
## [1] "cc_xpen: -0.642"
## [1] "race_asian: 0.4106"
## [1] "num_adlt: 0.2672"
## [1] "eitc: 0.2624"
## [1] "cap_xpen: -0.2391"
## [1] "highsch: -0.219"
## [1] "sex: 0.1895"
## [1] "------------------------------"
## [1] "PC11: 3.0800076541535"
## [1] ""
## [1] "separated: 0.5201"
## [1] "race_asian: -0.4313"
## [1] "sex: 0.3743"
## [1] "race_black: 0.3561"
## [1] "divorced: -0.275"
## [1] "widowed: 0.2307"
## [1] "num_kid: 0.1711"
## [1] "------------------------------"
## [1] "PC12: 2.95059015445189"
## [1] ""
## [1] "ui_kids: -0.6959"
## [1] "num_kid: -0.3139"
## [1] "widowed: -0.256"
## [1] "divorced: 0.2557"
## [1] "has_mortgage: 0.1977"
## [1] "mc_pb: -0.1914"
## [1] "eitc: -0.1847"
## [1] "------------------------------"
## [1] "PC13: 2.8442255032864"
## [1] ""
## [1] "separated: 0.7367"
## [1] "sex: -0.3718"
## [1] "race_asian: 0.2838"
## [1] "race_black: -0.2328"
## [1] "eitc: -0.2075"
## [1] "divorced: -0.1789"
## [1] "------------------------------"
## [1] "PC14: 2.76058432600027"
## [1] ""
## [1] "moop: -0.4385"
## [1] "sex: -0.4133"
## [1] "race_black: 0.4076"
## [1] "college: 0.3349"
## [1] "highsch: -0.3336"
```

```
## [1] "race_asian: -0.3031"
## [1] "ui_kids: 0.1836"
## [1] "------------------------------"
## [1] "PC15: 2.66848550537402"
## [1] ""
## [1] "moop: 0.6103"
## [1] "widowed: -0.4215"
## [1] "hi_prem: -0.3425"
## [1] "spm_hi_prem: -0.2736"
## [1] "race_black: 0.1884"
## [1] "------------------------------"
## [1] "PC16: 2.53467861580924"
## [1] ""
## [1] "eitc: 0.5516"
## [1] "ui_kids: -0.5506"
## [1] "sex: -0.3635"
## [1] "cohabit: 0.2425"
## [1] "num_kid: 0.2065"
## [1] "mc_pb: 0.1984"
## [1] "widowed: 0.1947"
## [1] "------------------------------"
```

Our first component appears very similar to the first component of the full data as far as variables involved is concerned. However, the second component does not have mc_pb labeled as significant, instead incorporating other demographic variables such as hispanic. Looking one further, our third variable appears to contain many of our medical expense variables, as well as some details about family size.

```
sig_aid[order(sig_aid$Count, decreasing=TRUE),]
```

```
##        Variable Count
## 27      widowed     7
## 34   race_black     7
## 2          sex     6
## 7       num_kid     6
## 8      num_adlt     6
## 24     divorced     6
## 32      highsch     6
## 13         eitc     5
## 16     cap_xpen     5
## 21        mc_pb     5
## 33   race_asian     5
## 3      hispanic     4
## 5       hi_prem     4
## 6          moop     4
## 22      cohabit     4
## 23      ui_kids     4
## 28 has_mortgage     4
## 29      renting     4
## 31      college     4
## 35   race_other     4
## 1          age     3
## 11      fed_tax     3
## 17      wk_xpen     3
## 18      cc_xpen     3
## 12   fed_tax_bc     2
## 19  spm_hi_prem     2
## 25      married     2
## 26    separated     2
## 30 less_college     2
## 4          agi     1
## 9       spm_res     1
## 10      spm_inc     1
## 14         fica     1
## 15       st_tax     1
## 20     med_xpen     1
```

In our new data set, we see that hi_prem is used half as frequently, suggesting its variance is explained somewhat faster. Now, our most frequent variables are widowed and race_black. It might be interesting to see if those two variables are at all correlated. Once again, a lot of our financial variables are only used once. Here, though, it appears med_xpen has replaced mc_pb. Instead of being explaining in the second component like mc_pb, med_xpen appears to explained by the third component.

Poor Data Components

Next, we move onto the components for the variables of the poor group.

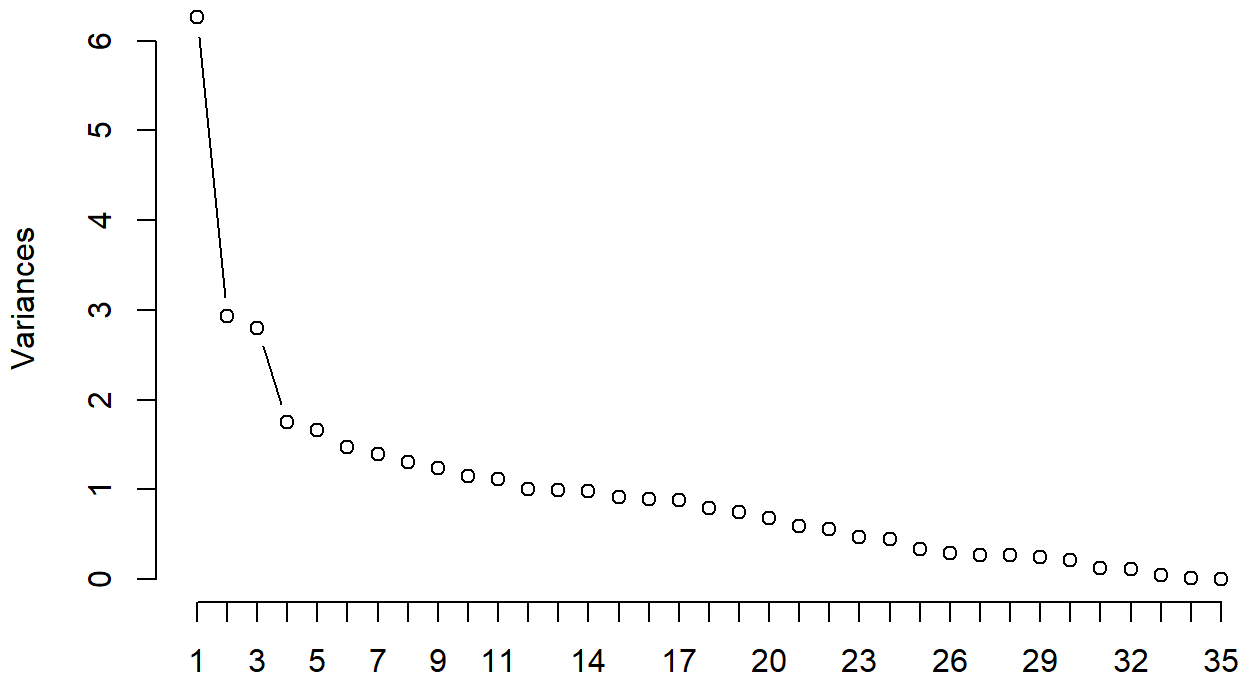
```
# Define our components as before, selecting only poor group observations.
pca_poor <- prcomp(select(filter(df, group == "poor"), -group), scale.=TRUE)

# View the relative importance of our new components.
summary(pca_poor)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.5019 1.7126 1.67448 1.32238 1.28793 1.21609 1.18289
## Proportion of Variance 0.1789 0.0838 0.08011 0.04996 0.04739 0.04225 0.03998
## Cumulative Proportion  0.1789 0.2626 0.34276 0.39272 0.44011 0.48237 0.52234
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.14125 1.11518 1.07107 1.05692 1.00493 0.99501 0.99085
## Proportion of Variance 0.03721 0.03553 0.03278 0.03192 0.02885 0.02829 0.02805
## Cumulative Proportion  0.55956 0.59509 0.62787 0.65978 0.68864 0.71692 0.74497
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.95829 0.94723 0.93745 0.89356 0.86273 0.82881 0.7692
## Proportion of Variance 0.02624 0.02564 0.02511 0.02281 0.02127 0.01963 0.0169
## Cumulative Proportion  0.77121 0.79685 0.82196 0.84477 0.86604 0.88566 0.9026
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.7460 0.68464 0.67315 0.57690 0.53863 0.52345 0.52041
## Proportion of Variance 0.0159 0.01339 0.01295 0.00951 0.00829 0.00783 0.00774
## Cumulative Proportion  0.9185 0.93186 0.94481 0.95432 0.96260 0.97043 0.97817
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.50098 0.45978 0.35834 0.33612 0.21373 0.11742 0.02765
## Proportion of Variance 0.00717 0.00604 0.00367 0.00323 0.00131 0.00039 0.00002
## Cumulative Proportion  0.98534 0.99138 0.99505 0.99828 0.99958 0.99998 1.00000
```

We see another dip in the significance of our first component. For our poor components, the drop seems to be somewhat larger than what it was for the poor group.

This drop is also reflected in our cumulative variance proportions. It now takes 17 components to explain 80% of the variance and 21 components to explain 90% of the variance. Whether this is due to a reduced sample size or an actual change in variance structure is yet to be seen. If we were to take a balanced sample of data, we could explore this further.

```
# Visualizing the above results.
screeplot(pca_poor["sdev"], npcs=35, type="lines")
```



We see a very interesting shift in correlation structure on the scree plot for the poor group. The second and third components appear to have very similar quantities of variance explained. If we compare this to our earlier components, we see that the third component has a standard deviation of only about 0.15 more. However, the 4th component is still about the same in terms of standard deviation, so it appears to be a much larger gap down to it.

If we were to select a number of components from this model, we might recommend up to 9 components. The correlation structure shown here is not nearly as smooth as what we saw earlier.

Next, we will use our function to analyze the components.

```
sig_poor <- important_components(pca_poor)
```

```
## [1] "------------------------------"
## [1] "PC1: 17.8845736182904"
## [1] ""
## [1] "spm_inc: 0.3673"
## [1] "fica: 0.3591"
## [1] "spm_res: 0.3336"
## [1] "wk_xpen: 0.3076"
## [1] "cap_xpen: 0.3067"
## [1] "fed_tax_bc: 0.2906"
## [1] "st_tax: 0.2501"
## [1] "num_adlt: 0.2233"
## [1] "fed_tax: 0.2104"
## [1] "cohabit: 0.1824"
## [1] "agi: 0.1761"
## [1] "------------------------------"
## [1] "PC2: 8.38004772400274"
## [1] ""
## [1] "med_xpen: -0.3584"
## [1] "hi_prem: -0.3145"
## [1] "spm_hi_prem: -0.3143"
## [1] "fed_tax: 0.3141"
## [1] "married: -0.2748"
## [1] "fed_tax_bc: 0.2737"
## [1] "st_tax: 0.2505"
## [1] "agi: -0.2494"
## [1] "cohabit: 0.2084"
## [1] "moop: -0.1921"
## [1] "num_adlt: -0.1698"
## [1] "------------------------------"
## [1] "PC3: 8.01105009732903"
## [1] ""
## [1] "age: 0.3073"
## [1] "med_xpen: 0.2846"
## [1] "fed_tax: 0.2843"
## [1] "eitc: -0.2679"
## [1] "hispanic: -0.2628"
## [1] "race_other: -0.2543"
## [1] "num_kid: -0.2491"
## [1] "mc_pb: 0.2298"
## [1] "spm_hi_prem: 0.2293"
## [1] "hi_prem: 0.2282"
## [1] "renting: -0.2112"
## [1] "st_tax: 0.2085"
## [1] "moop: 0.2052"
## [1] "fed_tax_bc: 0.1955"
## [1] "cap_xpen: -0.1855"
## [1] "wk_xpen: -0.1805"
## [1] "------------------------------"
## [1] "PC4: 4.99624820606259"
## [1] ""
## [1] "married: -0.4105"
## [1] "renting: 0.299"
```

```
## [1] "hispanic: -0.2681"
## [1] "race_other: -0.2661"
## [1] "divorced: 0.2588"
## [1] "race_black: 0.2578"
## [1] "hi_prem: 0.2574"
## [1] "num_adlt: -0.2496"
## [1] "spm_hi_prem: 0.2418"
## [1] "age: -0.2317"
## [1] "mc_pb: -0.2018"
## [1] "has_mortgage: -0.1996"
## [1] "-------------------------------"
## [1] "PC5: 4.73935463636226"
## [1] ""
## [1] "race_other: 0.499"
## [1] "hispanic: 0.4826"
## [1] "race_asian: -0.2782"
## [1] "married: -0.2334"
## [1] "has_mortgage: -0.2223"
## [1] "hi_prem: 0.2069"
## [1] "spm_hi_prem: 0.1991"
## [1] "renting: 0.1959"
## [1] "widowed: 0.1796"
## [1] "college: -0.1723"
## [1] "-------------------------------"
## [1] "PC6: 4.22534082687488"
## [1] ""
## [1] "highsch: -0.7092"
## [1] "less_college: 0.4421"
## [1] "college: 0.3168"
## [1] "race_black: -0.2083"
## [1] "race_other: 0.1765"
## [1] "-------------------------------"
## [1] "PC7: 3.99776619409399"
## [1] ""
## [1] "less_college: 0.5105"
## [1] "college: -0.4648"
## [1] "has_mortgage: 0.3185"
## [1] "renting: -0.3035"
## [1] "race_asian: -0.2612"
## [1] "num_kid: 0.2097"
## [1] "sex: 0.1991"
## [1] "-------------------------------"
## [1] "PC8: 3.72131401625259"
## [1] ""
## [1] "divorced: 0.4077"
## [1] "widowed: -0.3878"
## [1] "has_mortgage: 0.3857"
## [1] "renting: -0.3422"
## [1] "sex: -0.2419"
## [1] "less_college: -0.2167"
## [1] "mc_pb: -0.2134"
## [1] "age: -0.1959"
```

```
## [1] "highsch: 0.1917"
## [1] "race_black: -0.1773"
## [1] "------------------------------"
## [1] "PC9: 3.55323680324616"
## [1] ""
## [1] "widowed: 0.4963"
## [1] "sex: 0.3976"
## [1] "college: 0.3707"
## [1] "married: -0.2636"
## [1] "less_college: -0.2613"
## [1] "has_mortgage: 0.2394"
## [1] "renting: -0.215"
## [1] "race_black: -0.2121"
## [1] "------------------------------"
## [1] "PC10: 3.27769352408213"
## [1] ""
## [1] "ui_kids: -0.4548"
## [1] "num_kid: -0.4365"
## [1] "separated: -0.3824"
## [1] "divorced: 0.3186"
## [1] "cap_xpen: 0.249"
## [1] "wk_xpen: 0.2309"
## [1] "age: 0.1844"
## [1] "cc_xpen: 0.1816"
## [1] "college: -0.1783"
## [1] "------------------------------"
## [1] "PC11: 3.19163909632932"
## [1] ""
## [1] "divorced: -0.415"
## [1] "mc_pb: -0.3729"
## [1] "separated: 0.326"
## [1] "ui_kids: -0.3185"
## [1] "moop: -0.2468"
## [1] "num_kid: -0.2263"
## [1] "has_mortgage: 0.2152"
## [1] "race_black: 0.1958"
## [1] "widowed: 0.1778"
## [1] "renting: -0.1693"
## [1] "------------------------------"
## [1] "PC12: 2.88538575652755"
## [1] ""
## [1] "cc_xpen: -0.628"
## [1] "race_asian: 0.377"
## [1] "separated: -0.3212"
## [1] "moop: -0.2609"
## [1] "race_black: -0.2199"
## [1] "mc_pb: -0.2109"
## [1] "college: -0.1809"
## [1] "num_adlt: 0.1749"
## [1] "spm_hi_prem: 0.1699"
## [1] "------------------------------"
## [1] "PC13: 2.8286715753959"
```

```
## [1] ""
## [1] "separated: -0.5424"
## [1] "cc_xpen: 0.516"
## [1] "moop: -0.2822"
## [1] "ui_kids: 0.2544"
## [1] "hi_prem: 0.2425"
## [1] "mc_pb: -0.189"
## [1] "------------------------------"
## [1] "PC14: 2.8051055859399"
## [1] ""
## [1] "race_black: -0.4997"
## [1] "race_asian: 0.4676"
## [1] "separated: 0.384"
## [1] "college: -0.3164"
## [1] "cc_xpen: 0.3114"
## [1] "ui_kids: 0.2441"
## [1] "------------------------------"
## [1] "PC15: 2.62375455706431"
## [1] ""
## [1] "sex: 0.6697"
## [1] "ui_kids: -0.3879"
## [1] "widowed: -0.3137"
## [1] "divorced: 0.2215"
## [1] "married: 0.1959"
## [1] "cohabit: -0.1911"
## [1] "num_kid: 0.1742"
## [1] "------------------------------"
## [1] "PC16: 2.5635589703492"
## [1] ""
## [1] "moop: 0.4112"
## [1] "cohabit: -0.3653"
## [1] "ui_kids: 0.3541"
## [1] "agi: 0.3119"
## [1] "cc_xpen: -0.2819"
## [1] "sex: 0.2462"
## [1] "separated: -0.2227"
## [1] "mc_pb: -0.2121"
## [1] "num_adlt: -0.1903"
## [1] "spm_hi_prem: -0.1781"
## [1] "------------------------------"
## [1] "PC17: 2.5109173416435"
## [1] ""
## [1] "moop: -0.4686"
## [1] "ui_kids: 0.4683"
## [1] "num_kid: -0.313"
## [1] "age: 0.297"
## [1] "divorced: 0.2662"
## [1] "race_black: 0.2404"
## [1] "separated: 0.2191"
## [1] "cohabit: -0.1937"
## [1] "------------------------------"
```

Examining the significant variables in our components, we still see that the first component is largely composed of financial variables like income and taxes. However, we have a few additional household demographic variables concerning the number of adults in a home marked as significant, and wk_xpen seems more important than it has previously.

For our second component, we see a mix of medical and tax variables, which is slightly different than what we have seen before. For our third component, we have a real mix of medical and demographic variables that are somewhat difficult to interpret.

```
sig_poor[order(sig_poor$Count, decreasing=TRUE),]
```

```
##        Variable Count
## 34   race_black     8
## 6          moop     7
## 21        mc_pb     7
## 23      ui_kids     7
## 26    separated     7
## 29      renting     7
## 31      college     7
## 7       num_kid     6
## 19  spm_hi_prem     6
## 24     divorced     6
## 28 has_mortgage     6
## 1           age     5
## 2           sex     5
## 5       hi_prem     5
## 8      num_adlt     5
## 18      cc_xpen     5
## 22      cohabit     5
## 25      married     5
## 27      widowed     5
## 30 less_college     4
## 33   race_asian     4
## 35   race_other     4
## 3      hispanic     3
## 4           agi     3
## 11      fed_tax     3
## 12   fed_tax_bc     3
## 15       st_tax     3
## 16     cap_xpen     3
## 17      wk_xpen     3
## 20     med_xpen     2
## 32      highsch     2
## 9       spm_res     1
## 10      spm_inc     1
## 13         eitc     1
## 14         fica     1
```

If we look at how frequently each component is used, we see that race_black is significant in the most components. In comparison to earlier, we have only four variables that are significant only once, those being spm_res, spm_inc, eitc, and fica. It seems that regardless of class, income is the most important variable for

explaining the variance in our data.

No Data Components

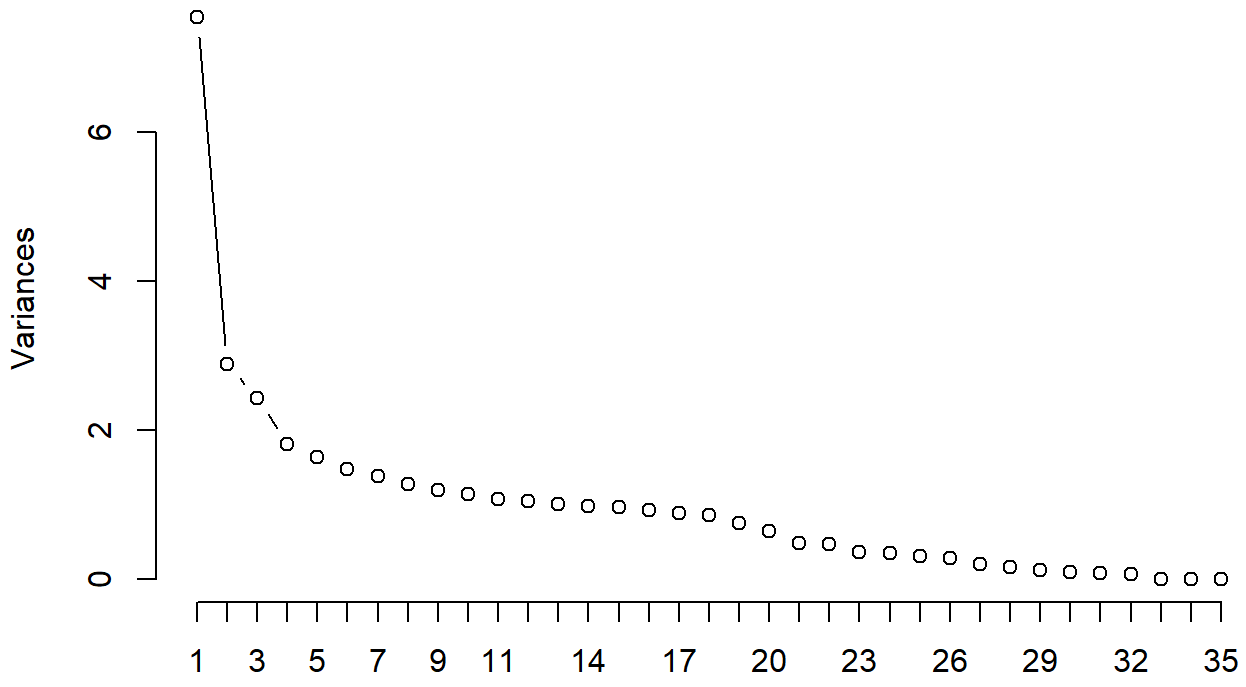
Finally, we will repeat this analysis for the variables in our no group. We will expect this group to be very similar to the full group because it composes the greatest portion of our data set.

```
# Define our components as before, selecting only no group observations.
pca_no <- prcomp(select(filter(df, group == "no"), -group), scale.=TRUE)

# View the relative importance of our new components.
summary(pca_no)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.7461 1.70059 1.55884 1.34750 1.28003 1.21746 1.17756
## Proportion of Variance 0.2155 0.08263 0.06943 0.05188 0.04681 0.04235 0.03962
## Cumulative Proportion  0.2155 0.29808 0.36751 0.41939 0.46620 0.50855 0.54817
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.13151 1.09549 1.06665 1.04049 1.02228 1.00348 0.99323
## Proportion of Variance 0.03658 0.03429 0.03251 0.03093 0.02986 0.02877 0.02819
## Cumulative Proportion  0.58475 0.61904 0.65155 0.68248 0.71234 0.74111 0.76929
##                           PC15   PC16    PC17   PC18    PC19    PC20    PC21
## Standard deviation     0.98373 0.9668 0.94613 0.9260 0.86643 0.80596 0.69679
## Proportion of Variance 0.02765 0.0267 0.02558 0.0245 0.02145 0.01856 0.01387
## Cumulative Proportion  0.79694 0.8236 0.84922 0.8737 0.89517 0.91373 0.92760
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.68759 0.60947 0.59058 0.55635 0.53669 0.44932 0.40912
## Proportion of Variance 0.01351 0.01061 0.00997 0.00884 0.00823 0.00577 0.00478
## Cumulative Proportion  0.94111 0.95172 0.96169 0.97053 0.97876 0.98453 0.98931
##                           PC29    PC30    PC31    PC32    PC33    PC34
## Standard deviation     0.35877 0.30299 0.28665 0.26401 0.03496 0.02260
## Proportion of Variance 0.00368 0.00262 0.00235 0.00199 0.00003 0.00001
## Cumulative Proportion  0.99299 0.99561 0.99796 0.99995 0.99999 1.00000
##                           PC35
## Standard deviation     6.257e-06
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

As expected, our values here appear very similar to the full data. It takes 16 components to explain 80% of the variance and 20 components to explain 90%.

```
# Visualizing the above results.
screeplot(pca_no["sdev"], npcs=35, type="lines")
```



Again, this scree plot matches very closely with the initial full-data scree plot. Of our three classes, the Poor group is the only class to break this general mold.

Next, we will use our function to analyze the components.

```
sig_no <- important_components(pca_no)
```

```
## [1] "------------------------------"
## [1] "PC1: 21.5454797829826"
## [1] ""
## [1] "spm_inc: 0.3381"
## [1] "spm_res: 0.3247"
## [1] "fica: 0.3243"
## [1] "fed_tax_bc: 0.3197"
## [1] "fed_tax: 0.3092"
## [1] "agi: 0.2954"
## [1] "st_tax: 0.2769"
## [1] "wk_xpen: 0.2501"
## [1] "cap_xpen: 0.2496"
## [1] "------------------------------"
## [1] "PC2: 8.26292342620109"
## [1] ""
## [1] "age: 0.3712"
## [1] "mc_pb: 0.3478"
## [1] "cap_xpen: -0.3176"
## [1] "wk_xpen: -0.3121"
## [1] "hispanic: -0.2494"
## [1] "race_other: -0.2377"
## [1] "renting: -0.2048"
## [1] "agi: 0.1945"
## [1] "fed_tax: 0.1932"
## [1] "med_xpen: 0.1867"
## [1] "eitc: -0.1735"
## [1] "fed_tax_bc: 0.1696"
## [1] "------------------------------"
## [1] "PC3: 6.942848489151"
## [1] ""
## [1] "med_xpen: -0.4755"
## [1] "spm_hi_prem: -0.3965"
## [1] "num_adlt: -0.3474"
## [1] "hi_prem: -0.3045"
## [1] "married: -0.2539"
## [1] "college: 0.1997"
## [1] "renting: 0.1906"
## [1] "moop: -0.1876"
## [1] "------------------------------"
## [1] "PC4: 5.18783861122157"
## [1] ""
## [1] "married: 0.4807"
## [1] "spm_hi_prem: -0.3619"
## [1] "divorced: -0.3437"
## [1] "cohabit: -0.3376"
## [1] "hi_prem: -0.2915"
## [1] "med_xpen: -0.2385"
## [1] "renting: -0.2083"
## [1] "num_kid: 0.1958"
## [1] "------------------------------"
## [1] "PC5: 4.68133103738752"
## [1] ""
```

```
## [1] "race_other: -0.5529"
## [1] "hispanic: -0.545"
## [1] "num_kid: 0.2245"
## [1] "cc_xpen: 0.1914"
## [1] "age: -0.1903"
## [1] "------------------------------"
## [1] "PC6: 4.23488288345189"
## [1] ""
## [1] "college: -0.5348"
## [1] "less_college: 0.5049"
## [1] "renting: -0.2325"
## [1] "hi_prem: -0.2065"
## [1] "has_mortgage: 0.2053"
## [1] "race_other: -0.199"
## [1] "hispanic: -0.1963"
## [1] "------------------------------"
## [1] "PC7: 3.96182639126793"
## [1] ""
## [1] "highsch: -0.6469"
## [1] "less_college: 0.5924"
## [1] "num_adlt: -0.1885"
## [1] "hi_prem: 0.1759"
## [1] "------------------------------"
## [1] "PC8: 3.65803618568073"
## [1] ""
## [1] "has_mortgage: 0.5015"
## [1] "renting: -0.4741"
## [1] "num_kid: 0.2509"
## [1] "divorced: 0.2379"
## [1] "widowed: 0.2325"
## [1] "sex: 0.2195"
## [1] "cc_xpen: 0.2084"
## [1] "married: -0.2055"
## [1] "college: 0.2005"
## [1] "less_college: -0.1749"
## [1] "------------------------------"
## [1] "PC9: 3.42884895212849"
## [1] ""
## [1] "widowed: -0.4733"
## [1] "sex: -0.4074"
## [1] "eitc: -0.2861"
## [1] "has_mortgage: 0.2767"
## [1] "race_asian: -0.2678"
## [1] "highsch: 0.24"
## [1] "divorced: 0.2312"
## [1] "cc_xpen: -0.1889"
## [1] "renting: -0.1824"
## [1] "num_adlt: -0.1758"
## [1] "hi_prem: 0.1734"
## [1] "------------------------------"
## [1] "PC10: 3.25070594290944"
## [1] ""
```

```
## [1] "cc_xpen: 0.3981"
## [1] "num_adlt: -0.3361"
## [1] "highsch: 0.316"
## [1] "college: -0.2914"
## [1] "hi_prem: 0.285"
## [1] "widowed: 0.2744"
## [1] "race_asian: -0.2653"
## [1] "cohabit: -0.257"
## [1] "num_kid: 0.2443"
## [1] "--------------------------------"
## [1] "PC11: 3.09318064788425"
## [1] ""
## [1] "ui_kids: -0.5224"
## [1] "eitc: -0.3217"
## [1] "sex: 0.306"
## [1] "widowed: 0.287"
## [1] "num_kid: -0.2825"
## [1] "divorced: -0.2517"
## [1] "cohabit: -0.2131"
## [1] "has_mortgage: 0.1986"
## [1] "cc_xpen: -0.1931"
## [1] "moop: -0.1884"
## [1] "mc_pb: -0.171"
## [1] "--------------------------------"
## [1] "PC12: 2.98586458877646"
## [1] ""
## [1] "race_black: -0.5531"
## [1] "separated: -0.5398"
## [1] "cc_xpen: 0.2996"
## [1] "race_asian: 0.2836"
## [1] "eitc: -0.2523"
## [1] "ui_kids: -0.2071"
## [1] "--------------------------------"
## [1] "PC13: 2.8770361749075"
## [1] ""
## [1] "race_black: 0.478"
## [1] "race_asian: -0.4218"
## [1] "ui_kids: -0.3868"
## [1] "separated: -0.3167"
## [1] "cc_xpen: 0.2817"
## [1] "moop: 0.2446"
## [1] "divorced: 0.1781"
## [1] "--------------------------------"
## [1] "PC14: 2.81860421010508"
## [1] ""
## [1] "separated: -0.654"
## [1] "moop: -0.4841"
## [1] "ui_kids: 0.2742"
## [1] "hi_prem: 0.234"
## [1] "cc_xpen: -0.2251"
## [1] "spm_hi_prem: 0.1765"
## [1] "race_black: 0.1746"
```

```
## [1] "-------------------------------"
## [1] "PC15: 2.76492969328173"
## [1] ""
## [1] "sex: -0.5683"
## [1] "moop: -0.4785"
## [1] "cc_xpen: 0.3646"
## [1] "ui_kids: -0.2451"
## [1] "mc_pb: 0.2243"
## [1] "cohabit: 0.1795"
## [1] "-------------------------------"
## [1] "PC16: 2.67032355556283"
## [1] ""
## [1] "eitc: 0.6034"
## [1] "ui_kids: -0.5439"
## [1] "divorced: 0.2804"
## [1] "num_kid: 0.2339"
## [1] "cohabit: -0.1856"
## [1] "-------------------------------"
```

Once again, our first component seems primarily concerned with our income and tax variables. And we also see that mc_pb is important to our second component. The third component appears to be composed of specific life expenses like medical costs, marriage, and college. The specific significance values for the third component seem somewhat different to what they were in the full data, but the variables themselves are close to the same.

```
sig_no[order(sig_no$Count, decreasing=TRUE),]
```

```
##        Variable Count
## 18      cc_xpen     9
## 5       hi_prem     7
## 7       num_kid     6
## 23      ui_kids     6
## 24     divorced     6
## 29      renting     6
## 6          moop     5
## 13         eitc     5
## 22      cohabit     5
## 2           sex     4
## 8      num_adlt     4
## 27      widowed     4
## 28 has_mortgage     4
## 31      college     4
## 33   race_asian     4
## 3      hispanic     3
## 19  spm_hi_prem     3
## 20     med_xpen     3
## 21        mc_pb     3
## 25      married     3
## 26    separated     3
## 30 less_college     3
## 32      highsch     3
## 34   race_black     3
## 35   race_other     3
## 1           age     2
## 4           agi     2
## 11      fed_tax     2
## 12   fed_tax_bc     2
## 16     cap_xpen     2
## 17      wk_xpen     2
## 9       spm_res     1
## 10      spm_inc     1
## 14         fica     1
## 15       st_tax     1
```

Interestingly, we do see a difference in the counts of significant variables. For this model, childcare expenses are the most frequently significant variable in our components, which might suggest it has strong correlations with many other variables.

Also, we see that we have only 4 variables that are used just once. spm_res, spm_inc, fica, and st_tax are all significant once in our first variable. This is a strange result because the aid model had almost the same six variables which were used only once as the full model, but this larger group that we would expect to be more similar to the full data only has 4.

Comparison

To begin our comparison, we are going to take a side-by-side look at which variables are important to our different models. To simplify our outputs, we will only consider the first 16 Principal Components of our models.

```
# Reassign signf (in case of import to a different file)
signf <- 1/sqrt(nrow(pca_full$rotation))

# Start printing out values.
print("-----------------------------------------------------------------------")
```

```
## [1] "-----------------------------------------------------------------------"
```

```
for (i in 1:16) {
  # For each component, we will basically concatenate what we would have output
  # from our earlier function. To do this, we first need to recalculate
  # everything from the earlier components.

  #   1. Select which variables are significant.
  aid_signf <- abs(pca_aid$rotation[,i]) > signf

  #   2. Find their variable names.
  aid_rows <- rownames(pca_aid$rotation)[aid_signf]

  #   3. Get their absolute values.
  aid_abs <- abs(pca_aid$rotation[aid_signf, i])

  #   4. Sort the variable names in order of significance.
  aid_rows <- aid_rows[order(aid_abs, decreasing=TRUE)]

  #   5. Count the number of variables used.
  aid_length <- length(aid_rows)

  # Repeat the above for the no group.
  no_signf <- abs(pca_no$rotation[,i]) > signf
  no_rows <- rownames(pca_no$rotation)[no_signf]
  no_abs <- abs(pca_no$rotation[no_signf, i])
  no_rows <- no_rows[order(no_abs, decreasing=TRUE)]
  no_length <- length(no_rows)

  # Repeat the above for the poor group.
  poor_signf <- abs(pca_poor$rotation[,i]) > signf
  poor_rows <- rownames(pca_poor$rotation)[poor_signf]
  poor_abs <- abs(pca_poor$rotation[poor_signf, i])
  poor_rows <- poor_rows[order(poor_abs, decreasing=TRUE)]
  poor_length <- length(poor_rows)

  # Repeat the above or the full data.
  full_signf <- abs(pca_full$rotation[,i]) > signf
  full_rows <- rownames(pca_full$rotation)[full_signf]
  full_abs <- abs(pca_full$rotation[full_signf, i])
  full_rows <- full_rows[order(full_abs, decreasing=TRUE)]
  full_length <- length(full_rows)

  # We need to know the maximum number of variables used in each component.
  max_length <- max(c(aid_length, no_length, poor_length, full_length))

  # Append NAs until the each variable list is the same length.
  #   The Aid Group.
  while (length(aid_rows) < max_length) {
    aid_rows <- append(aid_rows, NA)
  }
  #   The No Group.
  while (length(no_rows) < max_length) {
    no_rows <- append(no_rows, NA)
```

```
  }
  #   The Poor Group.
  while (length(poor_rows) < max_length) {
    poor_rows <- append(poor_rows, NA)
  }
  #   The Full Data
  while (length(full_rows) < max_length) {
    full_rows <- append(full_rows, NA)
  }

  # Concatenate our variable lists into a data frame.
  #   Columns represent groups. Rows are sorted significant variables..
  out_df <- cbind(aid_rows, no_rows, poor_rows, full_rows)

  print(paste(" PC", i, sep=""))   # Output component number.
  print(out_df)   # Print the results.
  # Move on to the next component.
  print("-----------------------------------------------------------------------")
}
```

```
## [1] " PC1"
##       aid_rows     no_rows      poor_rows    full_rows
##  [1,] "spm_inc"    "spm_inc"    "spm_inc"    "spm_inc"
##  [2,] "spm_res"    "spm_res"    "fica"       "spm_res"
##  [3,] "fica"       "fica"       "spm_res"    "fica"
##  [4,] "fed_tax_bc" "fed_tax_bc" "wk_xpen"    "fed_tax_bc"
##  [5,] "agi"        "fed_tax"    "cap_xpen"   "agi"
##  [6,] "fed_tax"    "agi"        "fed_tax_bc" "fed_tax"
##  [7,] "st_tax"     "st_tax"     "st_tax"     "st_tax"
##  [8,] "wk_xpen"    "wk_xpen"    "num_adlt"   "wk_xpen"
##  [9,] "cap_xpen"   "cap_xpen"   "fed_tax"    "cap_xpen"
## [10,] NA           NA           "cohabit"    NA
## [11,] NA           NA           "agi"        NA
## [1] "----------------------------------------------------------------------"
## [1] " PC2"
##       aid_rows     no_rows      poor_rows     full_rows
##  [1,] "cap_xpen"   "age"        "med_xpen"    "age"
##  [2,] "wk_xpen"    "mc_pb"      "hi_prem"     "mc_pb"
##  [3,] "age"        "cap_xpen"   "spm_hi_prem" "cap_xpen"
##  [4,] "hispanic"   "wk_xpen"    "fed_tax"     "wk_xpen"
##  [5,] "race_other" "hispanic"   "married"     "hispanic"
##  [6,] "num_adlt"   "race_other" "fed_tax_bc"  "race_other"
##  [7,] "mc_pb"      "renting"    "st_tax"      "num_kid"
##  [8,] "fed_tax"    "agi"        "agi"         "eitc"
##  [9,] "eitc"       "fed_tax"    "cohabit"     "renting"
## [10,] "num_kid"    "med_xpen"   "moop"        "med_xpen"
## [11,] "fed_tax_bc" "eitc"       "num_adlt"    "fed_tax"
## [12,] "widowed"    "fed_tax_bc" NA            NA
## [13,] "college"    NA           NA            NA
## [1] "----------------------------------------------------------------------"
## [1] " PC3"
##       aid_rows      no_rows       poor_rows     full_rows
##  [1,] "med_xpen"    "med_xpen"    "age"         "med_xpen"
##  [2,] "spm_hi_prem" "spm_hi_prem" "med_xpen"    "spm_hi_prem"
##  [3,] "hi_prem"     "num_adlt"    "fed_tax"     "hi_prem"
##  [4,] "age"         "hi_prem"     "eitc"        "num_adlt"
##  [5,] "num_adlt"    "married"     "hispanic"    "married"
##  [6,] "moop"        "college"     "race_other"  "moop"
##  [7,] "mc_pb"       "renting"     "num_kid"     "fed_tax"
##  [8,] "num_kid"     "moop"        "mc_pb"       "fed_tax_bc"
##  [9,] NA            NA            "spm_hi_prem" "college"
## [10,] NA            NA            "hi_prem"     "renting"
## [11,] NA            NA            "renting"     NA
## [12,] NA            NA            "st_tax"      NA
## [13,] NA            NA            "moop"        NA
## [14,] NA            NA            "fed_tax_bc"  NA
## [15,] NA            NA            "cap_xpen"    NA
## [16,] NA            NA            "wk_xpen"     NA
## [1] "----------------------------------------------------------------------"
## [1] " PC4"
##       aid_rows       no_rows       poor_rows      full_rows
##  [1,] "hispanic"     "married"     "married"      "married"
```

```
##  [2,] "race_other"   "spm_hi_prem" "renting"      "divorced"
##  [3,] "married"      "divorced"    "hispanic"     "cohabit"
##  [4,] "hi_prem"      "cohabit"     "race_other"   "spm_hi_prem"
##  [5,] "has_mortgage" "hi_prem"     "divorced"     "hi_prem"
##  [6,] "renting"      "med_xpen"    "race_black"   "renting"
##  [7,] "cohabit"      "renting"     "hi_prem"      "med_xpen"
##  [8,] "fed_tax"      "num_kid"     "num_adlt"     NA
##  [9,] "num_kid"      NA            "spm_hi_prem"  NA
## [10,] "num_adlt"     NA            "age"          NA
## [11,] "divorced"     NA            "mc_pb"        NA
## [12,] NA             NA            "has_mortgage" NA
## [1] "----------------------------------------------------------------------"
## [1] " PC5"
##       aid_rows     no_rows      poor_rows      full_rows
##  [1,] "cohabit"    "race_other" "race_other"   "race_other"
##  [2,] "divorced"   "hispanic"   "hispanic"     "hispanic"
##  [3,] "married"    "num_kid"    "race_asian"   "age"
##  [4,] "age"        "cc_xpen"    "married"      "num_kid"
##  [5,] "mc_pb"      "age"        "has_mortgage" "race_black"
##  [6,] "hispanic"   NA           "hi_prem"      "cc_xpen"
##  [7,] "race_other" NA           "spm_hi_prem"  "has_mortgage"
##  [8,] "renting"    NA           "renting"      NA
##  [9,] "num_adlt"   NA           "widowed"      NA
## [10,] "widowed"    NA           "college"      NA
## [11,] "race_black" NA           NA             NA
## [1] "----------------------------------------------------------------------"
## [1] " PC6"
##       aid_rows     no_rows        poor_rows      full_rows
##  [1,] "race_other" "college"      "highsch"      "less_college"
##  [2,] "hispanic"   "less_college" "less_college" "college"
##  [3,] "hi_prem"    "renting"      "college"      "renting"
##  [4,] "race_black" "hi_prem"      "race_black"   "has_mortgage"
##  [5,] "cap_xpen"   "has_mortgage" "race_other"   "hi_prem"
##  [6,] "num_adlt"   "race_other"   NA             "race_asian"
##  [7,] "wk_xpen"    "hispanic"     NA             NA
##  [8,] "cc_xpen"    NA             NA             NA
##  [9,] "moop"       NA             NA             NA
## [10,] "highsch"    NA             NA             NA
## [1] "----------------------------------------------------------------------"
## [1] " PC7"
##      aid_rows       no_rows        poor_rows      full_rows
## [1,] "less_college" "highsch"      "less_college" "highsch"
## [2,] "highsch"      "less_college" "college"      "less_college"
## [3,] "college"      "num_adlt"     "has_mortgage" "num_adlt"
## [4,] "has_mortgage" "hi_prem"      "renting"      "hi_prem"
## [5,] "renting"      NA             "race_asian"   "college"
## [6,] NA             NA             "num_kid"      "race_other"
## [7,] NA             NA             "sex"          "widowed"
## [8,] NA             NA             NA             "hispanic"
## [1] "----------------------------------------------------------------------"
## [1] " PC8"
##       aid_rows       no_rows        poor_rows      full_rows
```

```
##  [1,] "renting"      "has_mortgage" "divorced"     "has_mortgage"
##  [2,] "has_mortgage" "renting"      "widowed"      "renting"
##  [3,] "cohabit"      "num_kid"      "has_mortgage" "divorced"
##  [4,] "divorced"     "divorced"     "renting"      "race_black"
##  [5,] "highsch"      "widowed"      "sex"          "college"
##  [6,] "less_college" "sex"          "less_college" "less_college"
##  [7,] "ui_kids"      "cc_xpen"      "mc_pb"        NA
##  [8,] "race_black"   "married"      "age"          NA
##  [9,] NA             "college"      "highsch"      NA
## [10,] NA             "less_college" "race_black"   NA
## [1] "-----------------------------------------------------------------"
## [1] " PC9"
##       aid_rows     no_rows        poor_rows      full_rows
##  [1,] "college"    "widowed"      "widowed"      "widowed"
##  [2,] "sex"        "sex"          "sex"          "sex"
##  [3,] "highsch"    "eitc"         "college"      "highsch"
##  [4,] "cc_xpen"    "has_mortgage" "married"      "race_asian"
##  [5,] "widowed"    "race_asian"   "less_college" "eitc"
##  [6,] "cap_xpen"   "highsch"      "has_mortgage" "divorced"
##  [7,] "race_asian" "divorced"     "renting"      "cc_xpen"
##  [8,] NA           "cc_xpen"      "race_black"   "num_kid"
##  [9,] NA           "renting"      NA             NA
## [10,] NA           "num_adlt"     NA             NA
## [11,] NA           "hi_prem"      NA             NA
## [1] "-----------------------------------------------------------------"
## [1] " PC10"
##       aid_rows     no_rows      poor_rows   full_rows
##  [1,] "cc_xpen"    "cc_xpen"    "ui_kids"   "race_asian"
##  [2,] "race_asian" "num_adlt"   "num_kid"   "num_adlt"
##  [3,] "num_adlt"   "highsch"    "separated" "cc_xpen"
##  [4,] "eitc"       "college"    "divorced"  "cohabit"
##  [5,] "cap_xpen"   "hi_prem"    "cap_xpen"  "hi_prem"
##  [6,] "highsch"    "widowed"    "wk_xpen"   "widowed"
##  [7,] "sex"        "race_asian" "age"       "highsch"
##  [8,] NA           "cohabit"    "cc_xpen"   "college"
##  [9,] NA           "num_kid"    "college"   "separated"
## [10,] NA           NA           NA          "race_black"
## [1] "-----------------------------------------------------------------"
## [1] " PC11"
##       aid_rows     no_rows        poor_rows      full_rows
##  [1,] "separated"  "ui_kids"      "divorced"     "ui_kids"
##  [2,] "race_asian" "eitc"         "mc_pb"        "num_kid"
##  [3,] "sex"        "sex"          "separated"    "eitc"
##  [4,] "race_black" "widowed"      "ui_kids"      "race_asian"
##  [5,] "divorced"   "num_kid"      "moop"         "moop"
##  [6,] "widowed"    "divorced"     "num_kid"      "separated"
##  [7,] "num_kid"    "cohabit"      "has_mortgage" "widowed"
##  [8,] NA           "has_mortgage" "race_black"   NA
##  [9,] NA           "cc_xpen"      "widowed"      NA
## [10,] NA           "moop"         "renting"      NA
## [11,] NA           "mc_pb"        NA             NA
## [1] "-----------------------------------------------------------------"
```

```
## [1] " PC12"
##       aid_rows       no_rows      poor_rows     full_rows
##  [1,] "ui_kids"      "race_black" "cc_xpen"     "cc_xpen"
##  [2,] "num_kid"      "separated"  "race_asian"  "separated"
##  [3,] "widowed"      "cc_xpen"    "separated"   "has_mortgage"
##  [4,] "divorced"     "race_asian" "moop"        "race_black"
##  [5,] "has_mortgage" "eitc"       "race_black"  "divorced"
##  [6,] "mc_pb"        "ui_kids"    "mc_pb"       "moop"
##  [7,] "eitc"         NA           "college"     "renting"
##  [8,] NA             NA           "num_adlt"    NA
##  [9,] NA             NA           "spm_hi_prem" NA
## [1] "------------------------------------------------------------------------"
## [1] " PC13"
##      aid_rows     no_rows      poor_rows   full_rows
## [1,] "separated"  "race_black" "separated" "race_black"
## [2,] "sex"        "race_asian" "cc_xpen"   "race_asian"
## [3,] "race_asian" "ui_kids"    "moop"      "ui_kids"
## [4,] "race_black" "separated"  "ui_kids"   "college"
## [5,] "eitc"       "cc_xpen"    "hi_prem"   "highsch"
## [6,] "divorced"   "moop"       "mc_pb"     "hi_prem"
## [7,] NA           "divorced"   NA          "widowed"
## [1] "------------------------------------------------------------------------"
## [1] " PC14"
##      aid_rows     no_rows       poor_rows    full_rows
## [1,] "moop"       "separated"   "race_black" "separated"
## [2,] "sex"        "moop"        "race_asian" "cc_xpen"
## [3,] "race_black" "ui_kids"     "separated"  "sex"
## [4,] "college"    "hi_prem"     "college"    "moop"
## [5,] "highsch"    "cc_xpen"     "cc_xpen"    "cohabit"
## [6,] "race_asian" "spm_hi_prem" "ui_kids"    "hi_prem"
## [7,] "ui_kids"    "race_black"  NA           NA
## [1] "------------------------------------------------------------------------"
## [1] " PC15"
##      aid_rows      no_rows   poor_rows  full_rows
## [1,] "moop"        "sex"     "sex"      "moop"
## [2,] "widowed"     "moop"    "ui_kids"  "sex"
## [3,] "hi_prem"     "cc_xpen" "widowed"  "ui_kids"
## [4,] "spm_hi_prem" "ui_kids" "divorced" "widowed"
## [5,] "race_black"  "mc_pb"   "married"  "divorced"
## [6,] NA            "cohabit" "cohabit"  "highsch"
## [7,] NA            NA        "num_kid"  "spm_hi_prem"
## [8,] NA            NA        NA         "hi_prem"
## [1] "------------------------------------------------------------------------"
## [1] " PC16"
##       aid_rows  no_rows    poor_rows     full_rows
##  [1,] "eitc"    "eitc"     "moop"        "eitc"
##  [2,] "ui_kids" "ui_kids"  "cohabit"     "ui_kids"
##  [3,] "sex"     "divorced" "ui_kids"     "num_kid"
##  [4,] "cohabit" "num_kid"  "agi"         "moop"
##  [5,] "num_kid" "cohabit"  "cc_xpen"     "sex"
##  [6,] "mc_pb"   NA         "sex"         NA
##  [7,] "widowed" NA         "separated"   NA
```

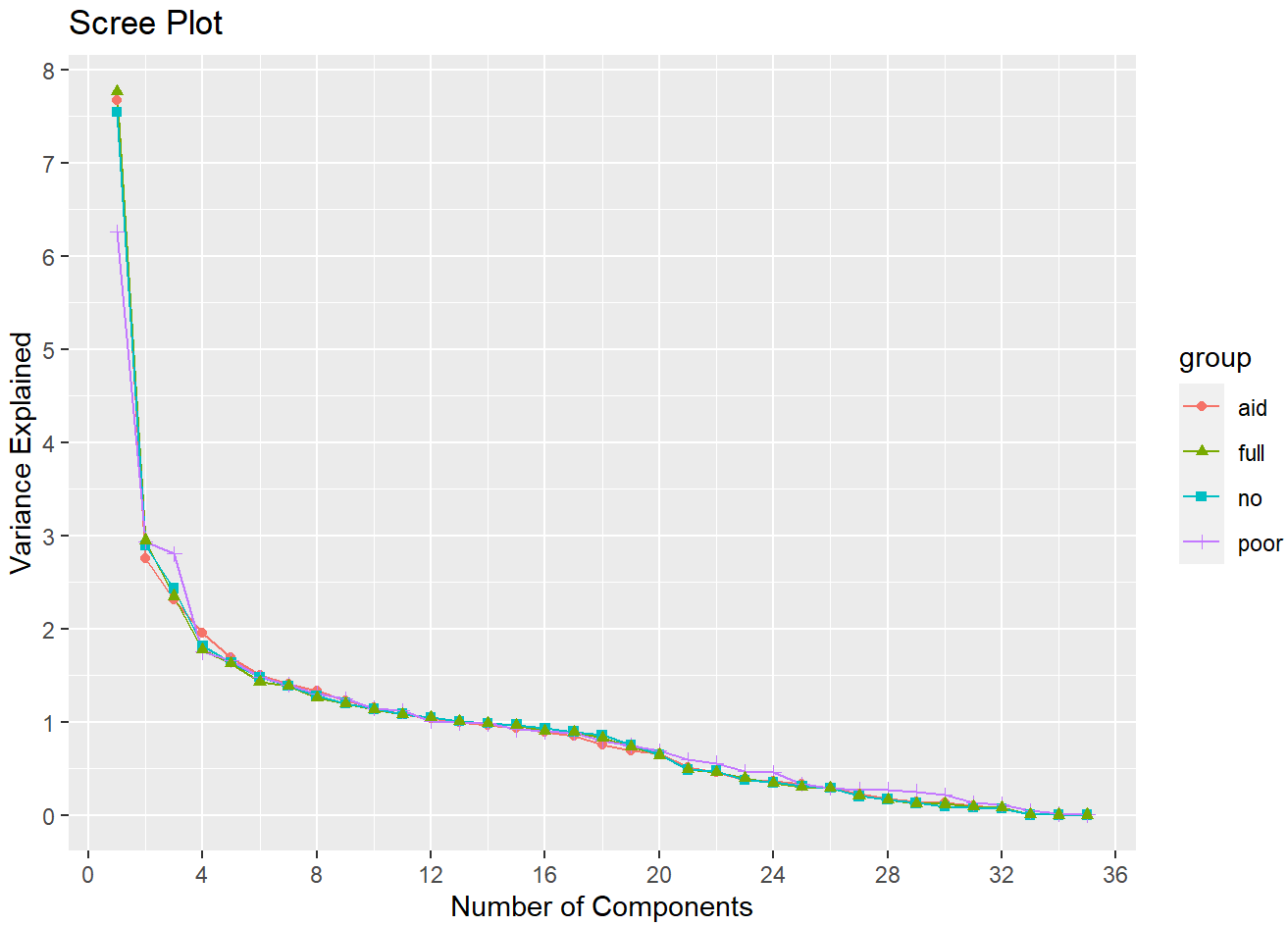
```
##  [8,] NA        NA         "mc_pb"       NA
##  [9,] NA        NA         "num_adlt"    NA
## [10,] NA        NA         "spm_hi_prem" NA
## [1] "------------------------------------------------------------------"
```

This method is a bit messy for comparing our models. However, it can provide us with a few quick insights. For example, our first component's most significant variable is spm_inc for all models. For the third component, the Poor group relies on noticeably more variables than the other groups. In the ninth component, all groups except the Aid Group have widowed as the most important variable.

The above lists can help us describe how general similarities are presented in our data and makes it easy to single out which groups are not following the trends of the others.

We can also compare our components visually by recreating our scree plots from earlier.

```
# Our basic approach is going to be to concatenate the rows of data frames
# containing data extracted from our principal components.

# From each, we will extract:
#   1. The Index of each Component (a list from 1 to 35)
#   2. The Variance expained by that Component.
#   3. Which Model the Component comes from (full, aid, poor, no)

aid_scree <- data.frame("pcs"=1:length(pca_aid$sdev),
                        "scree"=pca_aid$sdev**2,
                        "group"=rep("aid", length(pca_aid$sdev)))

no_scree <- data.frame("pcs"=1:length(pca_no$sdev),
                       "scree"=pca_no$sdev**2,
                       "group"=rep("no", length(pca_no$sdev)))

poor_scree <- data.frame("pcs"=1:length(pca_poor$sdev),
                         "scree"=pca_poor$sdev**2,
                         "group"=rep("poor", length(pca_poor$sdev)))

full_scree <- data.frame("pcs"=1:length(pca_full$sdev),
                         "scree"=pca_full$sdev**2,
                         "group"=rep("full", length(pca_full$sdev)))

# Now, we can concatenate our data frames into 1 model.
sum_scree <- rbind(aid_scree, no_scree, poor_scree, full_scree)

# Finally, we can very easily create a scree plot in ggplot2.
ggplot(sum_scree, aes(x=pcs, y=scree, color=group, shape=group)) +
  geom_line() +
  geom_point() +
  labs(title="Scree Plot", x="Number of Components", y="Variance Explained") +
  scale_x_continuous(breaks = seq(0, 36, by = 4)) +
  scale_y_continuous(breaks = seq(0, 8, by = 1))
```



With all of these put together, it is very easy to see that the main difference between the scree plots of our models is the poor group's variance explained by the first and third components. The first explains less and the third explains more. On the back half of our components, we see that the poor group has variance explained values slightly above the other in a few places, and at 18 components, there is some discrepancy in the aid group. Generally, though, these models do seem to follow a similar shape of variance explanation.

Next, we will try a few quantitative metrics to compare our components as vectors.

Significance Matching

Our first approach attempts to take the above lists of significant variables by component and count how frequently the models come to the same results. We will use two methods to do this: general matching and perfect matching. Under general matching, we will count how frequently variables are significant in the same component. Under perfect matching, we will cound how frequently a significant variable is of the same rank of importance (ie most important, second most important, etc) within the same components of different models.

```
# Again redefining signf for this chunk.
signf <- 1/sqrt(nrow(pca_full$rotation))

# A few sum variables that we want to track for final outputs.
#   Variables to store the total number of components.
total_comp_aid <- 0   # Total num components for the aid model.
total_comp_no <- 0   # Total num components for the no model.
total_comp_poor <- 0   # Total num components for the poor model.
total_comp_full <- 0   # Total num components for the full model.

#   Variables to store General and Perfect matches.
an_abs_m <- 0   # Perfect matches between aid and no.
an_ttl_m <- 0   # Component matches between aid and no.

ap_abs_m <- 0   # Perfect matches between aid and poor.
ap_ttl_m <- 0   # Component matches between aid and poor.

af_abs_m <- 0   # Perfect matches between aid and full.
af_ttl_m <- 0   # Component matches between aid and full.

np_abs_m <- 0   # Perfect matches between no and poor.
np_ttl_m <- 0   # Component matches between no and poor.

nf_abs_m <- 0   # Perfect matches between no and full.
nf_ttl_m <- 0   # Component matches between no and full.

pf_abs_m <- 0   # Perfect matches between poor and full.
pf_ttl_m <- 0   # Component matches between poor and full.

for (i in 1:35) {   # For each component.

  # "It might have been smart to make this a function."
  # Very similar to the listing process earlier.
  # The following gets the significant variables from each component.

  #   Aid Model.
  aid_signf <- abs(pca_aid$rotation[,i]) > signf
  aid_rows <- rownames(pca_aid$rotation)[aid_signf]
  aid_abs <- abs(pca_aid$rotation[aid_signf, i])
  aid_rows <- aid_rows[order(aid_abs, decreasing=TRUE)]
  aid_length <- length(aid_rows)

  #   No Model.
  no_signf <- abs(pca_no$rotation[,i]) > signf
  no_rows <- rownames(pca_no$rotation)[no_signf]
  no_abs <- abs(pca_no$rotation[no_signf, i])
  no_rows <- no_rows[order(no_abs, decreasing=TRUE)]
  no_length <- length(no_rows)

  #   Poor Model.
  poor_signf <- abs(pca_poor$rotation[,i]) > signf
  poor_rows <- rownames(pca_poor$rotation)[poor_signf]
```

```
poor_abs <- abs(pca_poor$rotation[poor_signf, i])
poor_rows <- poor_rows[order(poor_abs, decreasing=TRUE)]
poor_length <- length(poor_rows)

#   Full Model.
full_signf <- abs(pca_full$rotation[,i]) > signf
full_rows <- rownames(pca_full$rotation)[full_signf]
full_abs <- abs(pca_full$rotation[full_signf, i])
full_rows <- full_rows[order(full_abs, decreasing=TRUE)]
full_length <- length(full_rows)

# We need to incorporate NA's again for the perfect matching algorthim.
#   Find the most variables used in any model for this component..
max_length <- max(c(aid_length, no_length, poor_length, full_length))

# Verify that all vectors are of the same length.
#   Aid Vector.
while (length(aid_rows) < max_length) {
  aid_rows <- append(aid_rows, NA)
}
#   No Vector.
while (length(no_rows) < max_length) {
  no_rows <- append(no_rows, NA)
}
#   Poor Vector.
while (length(poor_rows) < max_length) {
  poor_rows <- append(poor_rows, NA)
}
#   Full Vector.
while (length(full_rows) < max_length) {
  full_rows <- append(full_rows, NA)
}

# Combine all of our vectors into columns of a data frame.
out_df <- cbind(aid_rows, no_rows, poor_rows, full_rows)
colnames(out_df) <- c("aid", "no", "poor", "full")   # Clean the column names.
out_df <- data.frame(out_df)   # Establish a full data frame.

# Count the total number of components that have been used in each model.
total_comp_aid <- total_comp_aid + aid_length
total_comp_no <- total_comp_no + no_length
total_comp_poor <- total_comp_poor + poor_length
total_comp_full <- total_comp_full + full_length

# Count how many matches exist among components.
#   For each variable marked as Significant...
for (j in 1:max_length) {
  # We're really using a bunch of if statements to enable counting.
  # First, we check if the values are all not NA's.
  #  Then we see if their values are equal (Perfect)
  #  or if 1 value is %in% the other column (General).
```

```
# Perfect matches
#   Aid and No
if ((out_df$aid[j] == out_df$no[j])&
    (is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$no[j]) == FALSE)) {
  an_abs_m <- an_abs_m + 1
}
#   Aid and Poor
if ((out_df$aid[j] == out_df$poor[j])&
    (is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$poor[j]) == FALSE)) {
  ap_abs_m <- ap_abs_m + 1
}
#   Aid and full
if ((out_df$aid[j] == out_df$full[j])&
    (is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)) {
  af_abs_m <- af_abs_m + 1
}
#   No and Poor
if ((out_df$no[j] == out_df$poor[j])&
    (is.na(out_df$no[j]) == FALSE)&(is.na(out_df$poor[j]) == FALSE)) {
  np_abs_m <- np_abs_m + 1
}
#   No and full
if ((out_df$no[j] == out_df$full[j])&
    (is.na(out_df$no[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)) {
  nf_abs_m <- nf_abs_m + 1
}
#   Poor and full
if ((out_df$poor[j] == out_df$full[j])&
    (is.na(out_df$poor[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)) {
  pf_abs_m <- pf_abs_m + 1
}


# Component Matches
#   Aid and No
if ((is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$no[j]) == FALSE)&
  (out_df$aid[j] %in% out_df$no)) {
  an_ttl_m <- an_ttl_m + 1
}
#   Aid and Poor
if ((is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$poor[j]) == FALSE)&
    (out_df$aid[j] %in% out_df$poor)) {
  ap_ttl_m <- ap_ttl_m + 1
}
#   Aid and full
if ((is.na(out_df$aid[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)&
    (out_df$aid[j] %in% out_df$full)) {
  af_ttl_m <- af_ttl_m + 1
}
#   No and Poor
if ((is.na(out_df$no[j]) == FALSE)&(is.na(out_df$poor[j]) == FALSE)&
```

```
        (out_df$no[j] %in% out_df$poor)) {
      np_ttl_m <- np_ttl_m + 1
    }
    #   No and full
    if ((is.na(out_df$no[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)&
        (out_df$no[j] %in% out_df$full)) {
      nf_ttl_m <- nf_ttl_m + 1
    }
    #   Poor and full
    if ((is.na(out_df$poor[j]) == FALSE)&(is.na(out_df$full[j]) == FALSE)&
        (out_df$poor[j] %in% out_df$full)) {
      pf_ttl_m <- pf_ttl_m + 1
    }
  }
}

# Here are a bunch of print statements to show our results.
print(paste("Num Componets in Aid:", total_comp_aid))
```

```
## [1] "Num Componets in Aid: 236"
```

```
print(paste("Num Componets in No:", total_comp_no))
```

```
## [1] "Num Componets in No: 243"
```

```
print(paste("Num Componets in Poor:", total_comp_poor))
```

```
## [1] "Num Componets in Poor: 254"
```

```
print(paste("Num Componets in full:", total_comp_full))
```

```
## [1] "Num Componets in full: 233"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (Aid & No):", an_abs_m))
```

```
## [1] "Num Perfect Matches (Aid & No): 29"
```

```
print(paste("Num Component Matches (Aid & No):", an_ttl_m))
```

```
## [1] "Num Component Matches (Aid & No): 123"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (Aid & Poor):", ap_abs_m))
```

```
## [1] "Num Perfect Matches (Aid & Poor): 19"
```

```
print(paste("Num Component Matches (Aid & Poor):", ap_ttl_m))
```

```
## [1] "Num Component Matches (Aid & Poor): 98"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (Aid & Full):", af_abs_m))
```

```
## [1] "Num Perfect Matches (Aid & Full): 42"
```

```
print(paste("Num Component Matches (Aid & Full):", af_ttl_m))
```

```
## [1] "Num Component Matches (Aid & Full): 136"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (No & Poor):", np_abs_m))
```

```
## [1] "Num Perfect Matches (No & Poor): 16"
```

```
print(paste("Num Component Matches (No & Poor):", np_ttl_m))
```

```
## [1] "Num Component Matches (No & Poor): 84"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (No & Full):", nf_abs_m))
```

```
## [1] "Num Perfect Matches (No & Full): 78"
```

```
print(paste("Num Component Matches (No & Full):", nf_ttl_m))
```

```
## [1] "Num Component Matches (No & Full): 168"
```

```
print("")
```

```
## [1] ""
```

```
print(paste("Num Perfect Matches (Poor & Full):", pf_abs_m))
```

```
## [1] "Num Perfect Matches (Poor & Full): 17"
```

```
print(paste("Num Component Matches (Poor & Full):", pf_ttl_m))
```

```
## [1] "Num Component Matches (Poor & Full): 79"
```

Unsurprisingly, we see the most matches of both kinds between the No group and the Full Group. As far as the Aid group is concerned, it has more matches with the No group than with the Poor group, suggesting those classes are more similar.

Next, we will assign a “score” to each using the number of matches and the number of total components used by each model.

```
# For our perfect values, we will define a percentage score by dividing the
# number of perfect matches by the average number of components between the
# two models.
print("Perfect Scores")
```

```
## [1] "Perfect Scores"
```

```
print(paste("-  Aid & No:", an_abs_m/((total_comp_aid+total_comp_no)/2)))
```

```
## [1] "-  Aid & No: 0.121085594989562"
```

```
print(paste("-  Aid & Poor:", ap_abs_m/((total_comp_aid+total_comp_poor)/2)))
```

```
## [1] "-  Aid & Poor: 0.0775510204081633"
```

```
print(paste("-  Aid & Full:", af_abs_m/((total_comp_aid+total_comp_full)/2)))
```

```
## [1] "-  Aid & Full: 0.17910447761194"
```

```
print(paste("-  No & Poor:", np_abs_m/((total_comp_no+total_comp_poor)/2)))
```

```
## [1] "-  No & Poor: 0.0643863179074447"
```

```
print(paste("-  No & Full:", nf_abs_m/((total_comp_no+total_comp_full)/2)))
```

```
## [1] "-  No & Full: 0.327731092436975"
```

```
print(paste("-  Poor & Full:", pf_abs_m/((total_comp_poor+total_comp_full)/2)))
```

```
## [1] "-  Poor & Full: 0.0698151950718686"
```

```
print("")
```

```
## [1] ""
```

```
# We then do the same for our general values.
print("Component Scores")
```

```
## [1] "Component Scores"
```

```
print(paste("-  Aid & No:", an_ttl_m/((total_comp_aid+total_comp_no)/2)))
```

```
## [1] "-  Aid & No: 0.51356993736952"
```

```
print(paste("-  Aid & Poor:", ap_ttl_m/((total_comp_aid+total_comp_poor)/2)))
```

```
## [1] "-  Aid & Poor: 0.4"
```

```
print(paste("-  Aid & Full:", af_ttl_m/((total_comp_aid+total_comp_full)/2)))
```

```
## [1] "-  Aid & Full: 0.579957356076759"
```

```
print(paste("-  No & Poor:", np_ttl_m/((total_comp_no+total_comp_poor)/2)))
```

```
## [1] "-  No & Poor: 0.338028169014085"
```

```
print(paste("-  No & Full:", nf_ttl_m/((total_comp_no+total_comp_full)/2)))
```

```
## [1] "-  No & Full: 0.705882352941177"
```

```
print(paste("-  Poor & Full:", pf_ttl_m/((total_comp_poor+total_comp_full)/2)))
```

```
## [1] "-  Poor & Full: 0.324435318275154"
```

For our Perfect matches, we see the Aid Group and No Groups are both most similar to the Full Group, while the Full Group itself is most similar to the No Group. Interestingly, the Poor Group is most similar to the Aid Group under the Perfect metric, suggesting there may still be some connection between the two classes.

For our General matches, we actually see the exact same results. It makes sense. We would expect more perfect matches to exist when variables align between components. It is interesting to see that the No and Full groups agree on which variables are the most Significant almost 70% of the time. The Poor Group has a much weaker connection to the other classes than they have with each other. At this stage, the Aid Group does not seem as similar to the Poor Group as it is to the No Group.

While this is a very simple way of comparing our models, it isn't the most mathematically rigorous. The difference in variable counts between components makes it so that this metric almost penalizes having many significant variables for a single component. It is a good first examination that is easier to interpret, but a more mathematical method could provide a more concrete truth.

Cosine Similarity

Our other method for comparing these components is going to be by utilizing cosine similarities. These will allow us to use the full components to compare the "directions" of our vectors.

At first, we will use two cosine similarities: Standard and Absolute. The only difference will be that we will use the absolute value of all components for the Absolute Cosine Similarity. Our reason for doing so is that we want to measure the similarity between the strengths of each variable's contribution. Signs for Principal Components are generally used only to maintain the matrix used in defining the components.

```
cos_df <- data.frame()   # Create an object to store our cosine values in.

# For each Component...
for (i in 1:35) {
  # Store the component.
  a_pc <- pca_aid$rotation[,i]
  n_pc <- pca_no$rotation[,i]
  p_pc <- pca_poor$rotation[,i]
  f_pc <- pca_full$rotation[,i]

  # Create a vector of cosine similarities within each pairing.
  cos_vec <- c(cosine(a_pc, n_pc), cosine(a_pc, p_pc), cosine(a_pc, f_pc),
               cosine(n_pc, p_pc), cosine(n_pc, f_pc), cosine(p_pc, f_pc),
               cosine(abs(a_pc), abs(n_pc)), cosine(abs(a_pc), abs(p_pc)),
               cosine(abs(a_pc), abs(f_pc)), cosine(abs(n_pc), abs(p_pc)),
               cosine(abs(n_pc), abs(f_pc)), cosine(abs(p_pc), abs(f_pc)))

  # Concatenate cosine pairings to storage data frame.
  cos_df <- rbind(cos_df, cos_vec)
}

# Rename the columns of our data frame for convenience.
colnames(cos_df) <- c("an", "ap", "af", "np", "nf", "pf",
                      "ab_an", "ab_ap", "ab_af", "ab_np", "ab_nf", "ab_pf")

# Print the mean result of both the Normal Cosine and Absolute Cosine.
print("Basic")
```

```
## [1] "Basic"
```

```
print(paste("Aid-No Mean Cosine Similarty:", mean(cos_df$an)))
```

```
## [1] "Aid-No Mean Cosine Similarty: 0.255524895130345"
```

```
print(paste("Aid-Poor Mean Cosine Similarty:", mean(cos_df$ap)))
```

```
## [1] "Aid-Poor Mean Cosine Similarty: -0.0172248111143551"
```

```
print(paste("Aid-full Mean Cosine Similarty:", mean(cos_df$af)))
```

```
## [1] "Aid-full Mean Cosine Similarty: -0.226772623263128"
```

```
print(paste("No-Poor Mean Cosine Similarty:", mean(cos_df$np)))
```

```
## [1] "No-Poor Mean Cosine Similarty: -0.00965932802203702"
```

```
print(paste("No-full Mean Cosine Similarty:", mean(cos_df$nf)))
```

```
## [1] "No-full Mean Cosine Similarty: -0.328189860895173"
```

```
print(paste("Poor-full Mean Cosine Similarty:", mean(cos_df$pf)))
```

```
## [1] "Poor-full Mean Cosine Similarty: 0.0180444014510913"
```

```
print("")
```

```
## [1] ""
```

```
print("Absolute")
```

```
## [1] "Absolute"
```

```
print(paste("Aid-No Mean Cosine Similarty:", mean(cos_df$ab_an)))
```

```
## [1] "Aid-No Mean Cosine Similarty: 0.712145031075831"
```

```
print(paste("Aid-Poor Mean Cosine Similarty:", mean(cos_df$ab_ap)))
```

```
## [1] "Aid-Poor Mean Cosine Similarty: 0.602098192180435"
```

```
print(paste("Aid-full Mean Cosine Similarty:", mean(cos_df$ab_af)))
```

```
## [1] "Aid-full Mean Cosine Similarty: 0.765335709676626"
```

```
print(paste("No-Poor Mean Cosine Similarty:", mean(cos_df$ab_np)))
```

```
## [1] "No-Poor Mean Cosine Similarty: 0.553914949343473"
```

```
print(paste("No-full Mean Cosine Similarty:", mean(cos_df$ab_nf)))
```

```
## [1] "No-full Mean Cosine Similarty: 0.8997030277178"
```

```
print(paste("Poor-full Mean Cosine Similarty:", mean(cos_df$ab_pf)))
```

```
## [1] "Poor-full Mean Cosine Similarty: 0.563744099649092"
```

Interestingly, we see that the strongest cosine similarity of the standard measure is actually a negative association between the No and Full Models. As far as magnitude is concerned, we see that the Poor Group is relatively unrelated to the other Classes for the standard measure. This continues to support the theory that the aid group is not more similar to the poor group.

For the absolute cosine metric, we see very similar results of different scale. It is interesting to see an almost 90% Cosine Similarity between the No and Full Models, but we have not gained much more information from this transformation.

As one final test, we will look at a weighted cosine similarity of our data. Instead of having each component pairing contribute equally to the mean of cosine similarities, each cosine score will contribute the average of its variance explanation between the models it is comparing. In theory, since these values should add up to 1, this is still a valid way to treat our means.

```
# Create an empty vector to store our weighted cosine scores in.
wcos_vec <- rep(0, 12)

# For each class, we first calculate the variances explained by our components.
# Then, we scale that to a percent of the total variance explained by all.
#   Aid Group.
a_lam <- pca_aid$sdev ^ 2
a_vpct <- a_lam/sum(a_lam)
#   No Group.
n_lam <- pca_no$sdev ^ 2
n_vpct <- n_lam/sum(n_lam)
#   Poor Group.
p_lam <- pca_poor$sdev ^ 2
p_vpct <- p_lam/sum(p_lam)
#   Full Group.
f_lam <- pca_full$sdev ^ 2
f_vpct <- f_lam/sum(f_lam)

# For each Component...
for (i in 1:35) {
  # Store the vector of Variables.
  a_pc <- pca_aid$rotation[,i]
  n_pc <- pca_no$rotation[,i]
  p_pc <- pca_poor$rotation[,i]
  f_pc <- pca_full$rotation[,i]

  # Calculate cosine similarities using the Absolute method.
  #   We believe the Absolute Method makes the most logical sense.
  an_cos <- cosine(abs(a_pc), abs(n_pc))
  ap_cos <- cosine(abs(a_pc), abs(p_pc))
  af_cos <- cosine(abs(a_pc), abs(f_pc))
  np_cos <- cosine(abs(n_pc), abs(p_pc))
  nf_cos <- cosine(abs(n_pc), abs(f_pc))
  pf_cos <- cosine(abs(p_pc), abs(f_pc))

  # Add our scaled values to the cosine vector.
  #   We have scales for both contribution percents so we can average them
  #   to get a final score.

  #   Aid with No.
  wcos_vec[1] <- wcos_vec[1] + an_cos*a_vpct[i]
  wcos_vec[2] <- wcos_vec[2] + an_cos*n_vpct[i]
  #   Aid with Poor.
  wcos_vec[3] <- wcos_vec[3] + ap_cos*a_vpct[i]
  wcos_vec[4] <- wcos_vec[4] + ap_cos*p_vpct[i]
  #   Aid with Full.
  wcos_vec[5] <- wcos_vec[5] + af_cos*a_vpct[i]
  wcos_vec[6] <- wcos_vec[6] + af_cos*f_vpct[i]
  #   No with Poor.
  wcos_vec[7] <- wcos_vec[7] + np_cos*n_vpct[i]
  wcos_vec[8] <- wcos_vec[8] + np_cos*p_vpct[i]
  #   No with Full.
```

```
  wcos_vec[9] <- wcos_vec[9] + nf_cos*n_vpct[i]
  wcos_vec[10] <- wcos_vec[10] + nf_cos*f_vpct[i]
  #   Poor with Full.
  wcos_vec[11] <- wcos_vec[11] + pf_cos*p_vpct[i]
  wcos_vec[12] <- wcos_vec[12] + pf_cos*f_vpct[i]
}

# Print out the means of our two weighted similarity scores for each pairing.
print("Weigthed Cosine Similarity")
```

```
## [1] "Weigthed Cosine Similarity"
```

```
print(paste("Aid-No Mean Cosine Similarty:", mean(wcos_vec[1:2])))
```

```
## [1] "Aid-No Mean Cosine Similarty: 0.794436046178525"
```

```
print(paste("Aid-Poor Mean Cosine Similarty:", mean(wcos_vec[3:4])))
```

```
## [1] "Aid-Poor Mean Cosine Similarty: 0.711177000054161"
```

```
print(paste("Aid-full Mean Cosine Similarty:", mean(wcos_vec[5:6])))
```

```
## [1] "Aid-full Mean Cosine Similarty: 0.811089061848199"
```

```
print(paste("No-Poor Mean Cosine Similarty:", mean(wcos_vec[7:8])))
```

```
## [1] "No-Poor Mean Cosine Similarty: 0.718366597556193"
```

```
print(paste("No-full Mean Cosine Similarty:", mean(wcos_vec[9:10])))
```

```
## [1] "No-full Mean Cosine Similarty: 0.945701393324834"
```

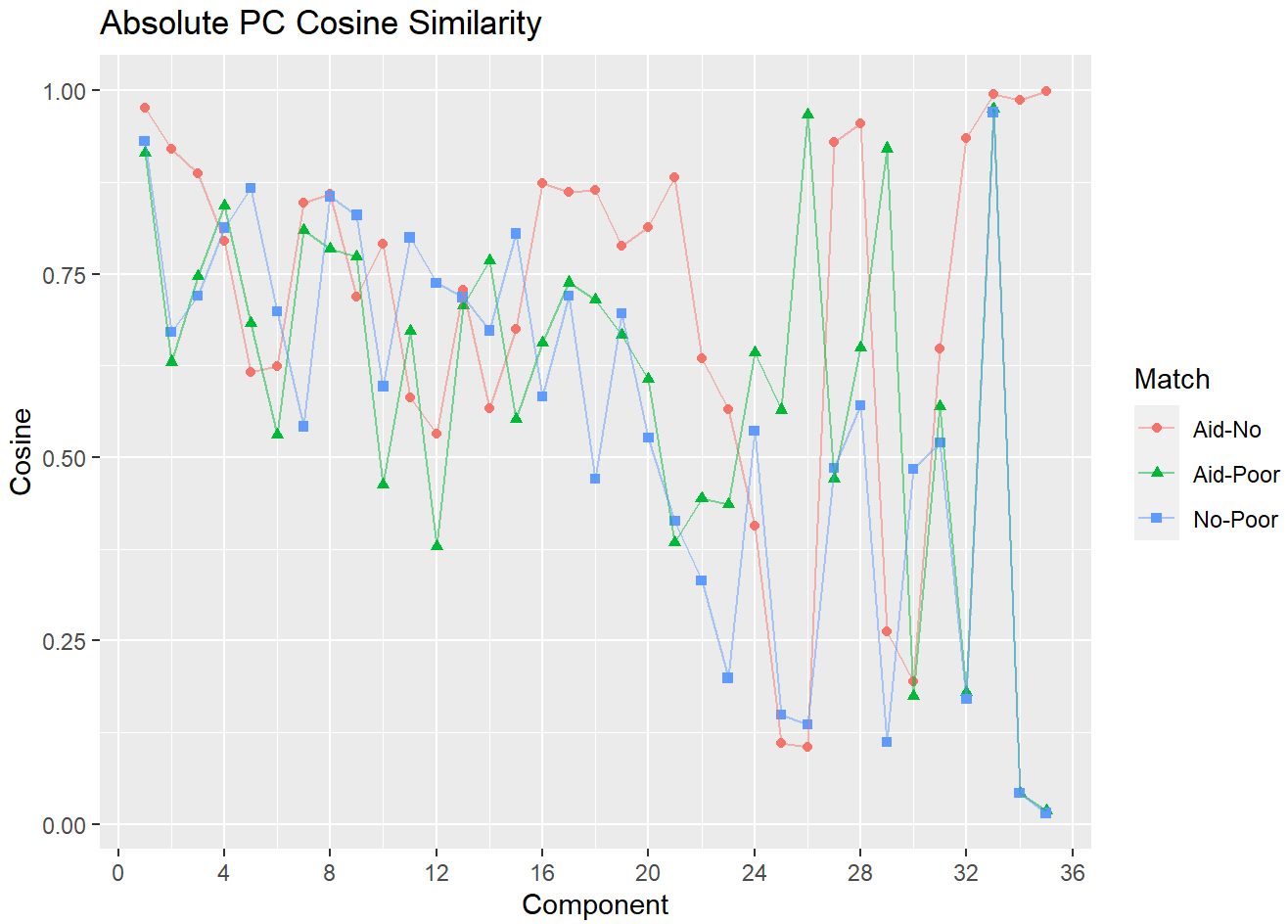
```
print(paste("Poor-full Mean Cosine Similarty:", mean(wcos_vec[11:12])))
```

```
## [1] "Poor-full Mean Cosine Similarty: 0.718634253848237"
```

Under this weighted model, we actually see that our Models are not as different as what our other metrics would suggest. We still see that the No and Full Models are the most similar, but now, all of our scores are above 70%. The lowest score shown here is actually the score between the Aid group and the Poor Group. This seems very suggestive that they are not similar enough to justify being called “the same.” And as a last note, it is interesting how similar the scores for the No-Poor and Poor-Full comparisons are.

As a final way of using these Principal components, we will attempt to graphically represent the similarities between our three primary interest Groups. We will use the absolute comparisons for this example.

```
# Create a data frame of values.
#   Component: Which number of component is it?
#   Match: Which pairing is it?
#   Cosine: What is the value?
grph_cos <- data.frame("Component" = rep(1:35, 3),
                       "Match"=c(rep("Aid-No", 35), rep("Aid-Poor", 35),
                                 rep("No-Poor", 35)),
                       "Cosine" = c(cos_df$ab_an, cos_df$ab_ap,
                                    cos_df$ab_np))

# Graph our data.
ggplot(grph_cos, aes(x=Component, y=Cosine, color=Match, shape=Match)) +
  geom_point() + geom_line(alpha=0.5) +
  labs(title="Absolute PC Cosine Similarity") +
  scale_x_continuous(breaks = seq(0, 36, by = 4))
```



This is a somewhat messy visualization, but it does show how frequently the Aid and No classes are the most similar of our Groups. It is interesting how many spikes are present in this visualization. It almost looks like the predicted values of a cyclical time series model.