Poverty Data - Load And Cleanse

Michael Bemus

2024-09-25

Loading Data

For this project, we will use open-source data from the United States Census Bureau at census.gov. The file is formatted as either a SAS or STATA file. All data related to this file can be found at:

https://www.census.gov/data/datasets/time-series/demo/supplemental-poverty-measure/acs-research-files.html (https://www.census.gov/data/datasets/time-series/demo/supplemental-poverty-measure/acs-research-files.html)

For our purposes, we will read in the STATA file and complete transformations to use for later steps.

```r
# We need to use the "haven" R package to read in our data.

#install.packages("haven")
library(haven)
```

```
## Warning: package 'haven' was built under R version 4.1.3
```

```r
# The "read_stata" function will create a tibble data frame from the data.
df <- read_stata("https://www2.census.gov/programs-surveys/supplemental-poverty-measure/dataset
s/spm/spm_2022_pu.dta")

head(df)   # View the first 6 rows of the data frame.
```

```
## # A tibble: 6 x 46
##   FILEDATE serialno sporder    st    PUMA OFFPoor    wt   Age   Mar   Sex
##      <dbl>    <dbl>   <dbl> <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 20240126        1       1    17 1703167       0    43    79     2     2
## 2 20240126        1       2    17 1703167       0    45    59     5     1
## 3 20240126        2       1     8  801101       0   151    62     1     2
## 4 20240126        2       2     8  801101       0   157    66     1     1
## 5 20240126        3       1    24 2401202       0   101    60     1     1
## 6 20240126        3       2    24 2401202       0   117    50     1     2
## # i 36 more variables: Education <dbl>, Race <dbl>, Hispanic <dbl>, AGI <dbl>,
## #   Tax_unit <dbl>, HI_premium <dbl>, Moop_other <dbl>, SPM_ID <dbl>,
## #   SPM_Poor <dbl>, SPM_PovThreshold <dbl>, SPM_EquivScale <dbl>,
## #   SPM_GeoAdj <dbl>, SPM_NumPer <dbl>, SPM_NumKids <dbl>, SPM_NumAdults <dbl>,
## #   SPM_TenMortStatus <dbl>, SPM_Resources <dbl>, SPM_Totval <dbl>,
## #   SPM_SNAPSub <dbl>, SPM_CapHouseSub <dbl>, SPM_SchLunch <dbl>,
## #   SPM_EngVal <dbl>, SPM_WICval <dbl>, SPM_FedTax <dbl>, ...
```

For specifics on the endpoints related to this data set, please see the data dictionary found at:

https://www2.census.gov/programs-surveys/supplemental-poverty-measure/datasets/spm/spm-asc-data-dictionary.pdf (https://www2.census.gov/programs-surveys/supplemental-poverty-measure/datasets/spm/spm-asc-data-dictionary.pdf)

Data Processing

The main data that concerns us for this analysis is the demographic/household data, tax/income data, insurance/healthcare data, and subsidy data. In the followings steps, we will remove and reclassify much of our data to simplify future processes.

Create Target Group

First, we need to create our target labels. An observation is classified as "poor" if they fall under either the Federal Poverty Metric or the Supplemental Poverty Metric. An observation is classified as "aid" if they are earning subsidies without but not classified under the "poor" group. And an observation is classified as "no" if they are not impoverished and not earning subsidies.

```
df$group[(df$OFFPoor == 1)|
           (df$SPM_Poor == 1)] <- "poor"
```

```
## Warning: Unknown or uninitialised column: `group`.
```

```
df$group[(df$OFFPoor==0)&
           (df$SPM_Poor==0)&
           ((df$SPM_SNAPSub>0)
            |(df$SPM_CapHouseSub>0)
            |(df$SPM_SchLunch>0)
            |(df$SPM_EngVal>0)
            |(df$SPM_WICval>0))] <- "aid"

df$group[(df$OFFPoor==0)&
           (df$SPM_Poor==0)&
           (df$SPM_SNAPSub==0)&
           (df$SPM_CapHouseSub==0)&
           (df$SPM_SchLunch==0)&
           (df$SPM_EngVal==0)&
           (df$SPM_WICval==0)] <- "no"

table(df$group)
```

```
##
##     aid      no    poor
##  831803 1860706  499699
```

Remove Excess Columns

We see that the majority of observations are of non-impoverished, non-subsidy earning individuals. We have about 60% more observations in the aid group than in the impoverished group.

Next, we remove some extra variables that aren't necessary for our analysis.

```
# Removing extra columns primarily related to poverty calculations.
df <- df |>
  select(-SPM_EquivScale, -wt, -Tax_unit, -SPM_ID, -PUMA, -SPM_NumPer,
         -SPM_GeoAdj, -serialno, -FILEDATE, -sporder, -SPM_PovThreshold)

head(df)   # Show new data frame.
```

```
## # A tibble: 6 x 36
##      st OFFPoor   Age   Mar   Sex Education  Race Hispanic    AGI HI_premium
##   <dbl>   <dbl> <dbl> <dbl> <dbl>     <dbl> <dbl>    <dbl>  <dbl>      <dbl>
## 1    17       0    79     2     2         3     2        0  60300        558
## 2    17       0    59     5     1         4     2        0  85000       1297
## 3     8       0    62     1     2         2     1        0  73550        108
## 4     8       0    66     1     1         2     4        1  73550       5000
## 5    24       0    60     1     1         2     1        0 106900       3632
## 6    24       0    50     1     2         3     1        0 106900      15600
## # i 26 more variables: Moop_other <dbl>, SPM_Poor <dbl>, SPM_NumKids <dbl>,
## #   SPM_NumAdults <dbl>, SPM_TenMortStatus <dbl>, SPM_Resources <dbl>,
## #   SPM_Totval <dbl>, SPM_SNAPSub <dbl>, SPM_CapHouseSub <dbl>,
## #   SPM_SchLunch <dbl>, SPM_EngVal <dbl>, SPM_WICval <dbl>, SPM_FedTax <dbl>,
## #   SPM_FedTaxBC <dbl>, SPM_EITC <dbl>, SPM_FICA <dbl>, SPM_StTax <dbl>,
## #   SPM_CapWkCCXpns <dbl>, SPM_WkXpns <dbl>, SPM_ChildcareXpns <dbl>,
## #   SPM_Premium <dbl>, SPM_MedXpns <dbl>, Medicare_PartB <dbl>, ...
```

Rename Columns

Now, with our columns of interest selected, I renamed them to make them a bit easier to use.

```
print(colnames(df))
```

```
##  [1] "st"                "OFFPoor"           "Age"
##  [4] "Mar"               "Sex"               "Education"
##  [7] "Race"              "Hispanic"          "AGI"
## [10] "HI_premium"        "Moop_other"        "SPM_Poor"
## [13] "SPM_NumKids"       "SPM_NumAdults"     "SPM_TenMortStatus"
## [16] "SPM_Resources"     "SPM_Totval"        "SPM_SNAPSub"
## [19] "SPM_CapHouseSub"   "SPM_SchLunch"      "SPM_EngVal"
## [22] "SPM_WICval"        "SPM_FedTax"        "SPM_FedTaxBC"
## [25] "SPM_EITC"          "SPM_FICA"          "SPM_StTax"
## [28] "SPM_CapWkCCXpns"   "SPM_WkXpns"        "SPM_ChildcareXpns"
## [31] "SPM_Premium"       "SPM_MedXpns"       "Medicare_PartB"
## [34] "SPM_wCohabit"      "SPM_wUI_LT15"      "group"
```

```
colnames(df) <- c("st", "off_pov", "age", "mar", "sex", "edu", "race",
                         "hispanic", "agi", "hi_prem", "moop", "spm_pov",
                         "num_kid", "num_adlt", "mortgage", "spm_res",
                         "spm_inc", "snap", "house_sub", "slunch", "energy",
                         "wic", "fed_tax", "fed_tax_bc", "eitc", "fica",
                         "st_tax", "cap_xpen", "wk_xpen", "cc_xpen",
                         "spm_hi_prem", "med_xpen", "mc_pb", "cohabit",
                         "ui_kids", "group")

print(colnames(df))   # Show results of change.
```

```
##  [1] "st"          "off_pov"     "age"         "mar"         "sex"
##  [6] "edu"         "race"        "hispanic"    "agi"         "hi_prem"
## [11] "moop"        "spm_pov"     "num_kid"     "num_adlt"    "mortgage"
## [16] "spm_res"     "spm_inc"     "snap"        "house_sub"   "slunch"
## [21] "energy"      "wic"         "fed_tax"     "fed_tax_bc"  "eitc"
## [26] "fica"        "st_tax"      "cap_xpen"    "wk_xpen"     "cc_xpen"
## [31] "spm_hi_prem" "med_xpen"    "mc_pb"       "cohabit"     "ui_kids"
## [36] "group"
```

Reclassify Categorical Variables

Next, we want to examine some of our categorical variables and redefine them as factors.

```r
# Change Sex from binary to a character factor.
df$sex <- factor(df$sex)
levels(df$sex) <- c("M", "F")

# Change Marital Status from numeric to character factor.
df$mar <- factor(df$mar)
levels(df$mar) <- c("M", "W", "D", "S", "NM")

# Assign Hispanic as a factor rather than numeric.
df$hispanic <- factor(df$hispanic)

# Change Race from numeric to character factor.
df$race <-factor(df$race)
levels(df$race) <- c("W", "B", "A", "O")

# Change Education Level from numeric to character factor.
df$edu <- factor(df$edu)
levels(df$edu) <- c("<25", "<HS", "HS", "<C", "C")

# Assign Official and Supplemental Poverty variables as binary factors.
df$off_pov <- factor(df$off_pov)
df$spm_pov <- factor(df$spm_pov)

# Assign Unidentified Kids as a binary factor
df$ui_kids <- factor(df$ui_kids)

# Change Mortgage from numeric to a character factor.
df$mortgage <- factor(df$mortgage)
levels(df$mortgage) <- c("M", "N", "R")

# Change State from numeric to a character factor.
df$st <- factor(df$st)
levels(df$st) <- c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC",
                          "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY",
                          "LA", "ME", "MD", "MA", "MI", "MN", "MS", "MO", "MT",
                          "NE", "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "OH",
                          "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT",
                          "VT", "VA", "WA", "WV", "WI", "WY")

# Set our variable of interest, Group, to a 3-class factor.
df$group <- factor(df$group)
```

Next, we want to remove unnecessary data in our examination. Because the data set already has three million observations, we felt free to remove many high-leverage values that were irregular from the data as a whole.

Data Filtering

The first change we made was removing individuals younger than 25. This way, we will be dealing with adults who should have unique values for all of our financial variables.

```
# Remove all observations with an age less than 26.
df <- df |>
  filter(age > 25)

print(nrow(df))   # Check to see how much data remains.
```

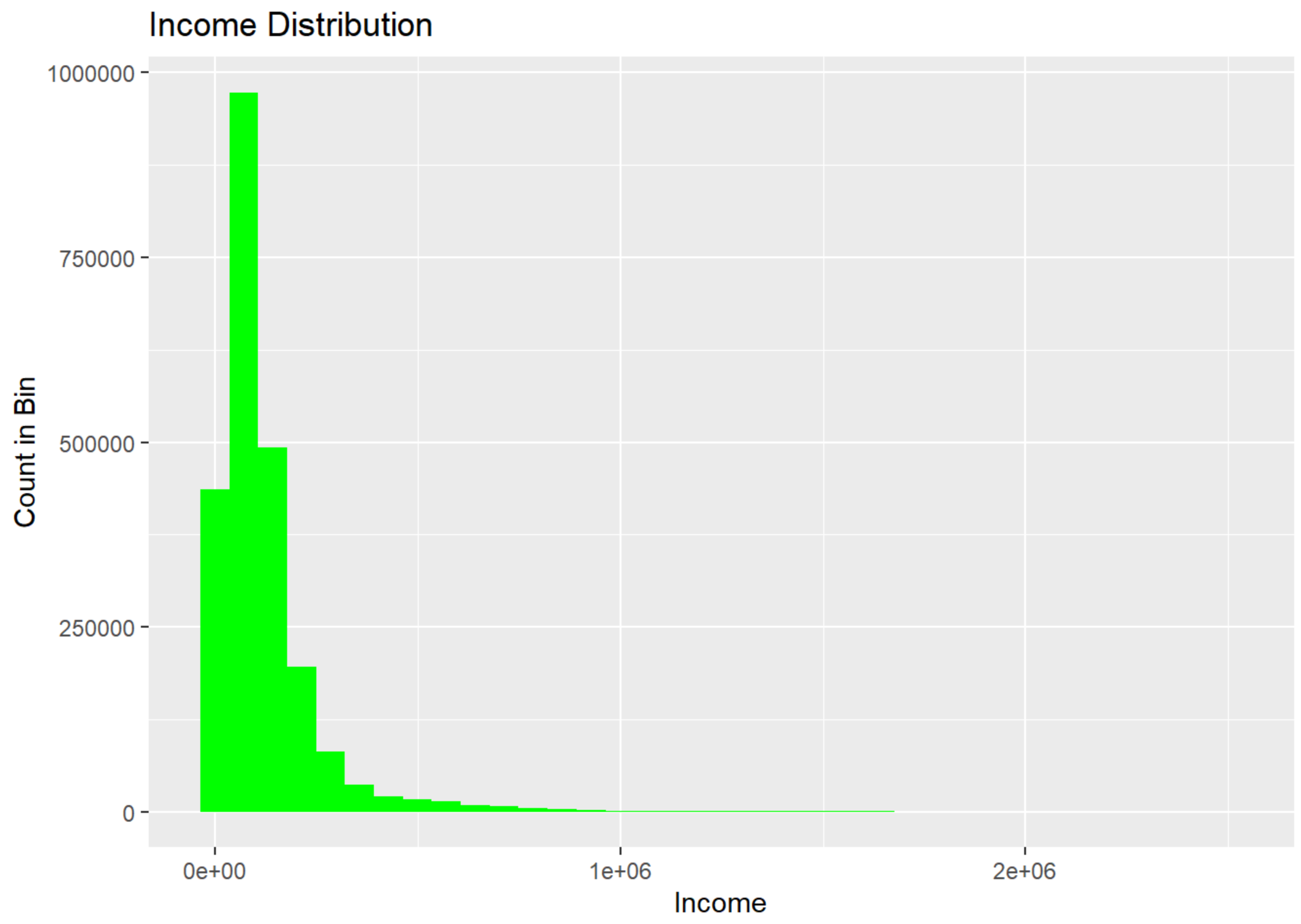
```
## [1] 2294991
```

Next, we need to begin visualizing our data to better understand where outliers may be present.

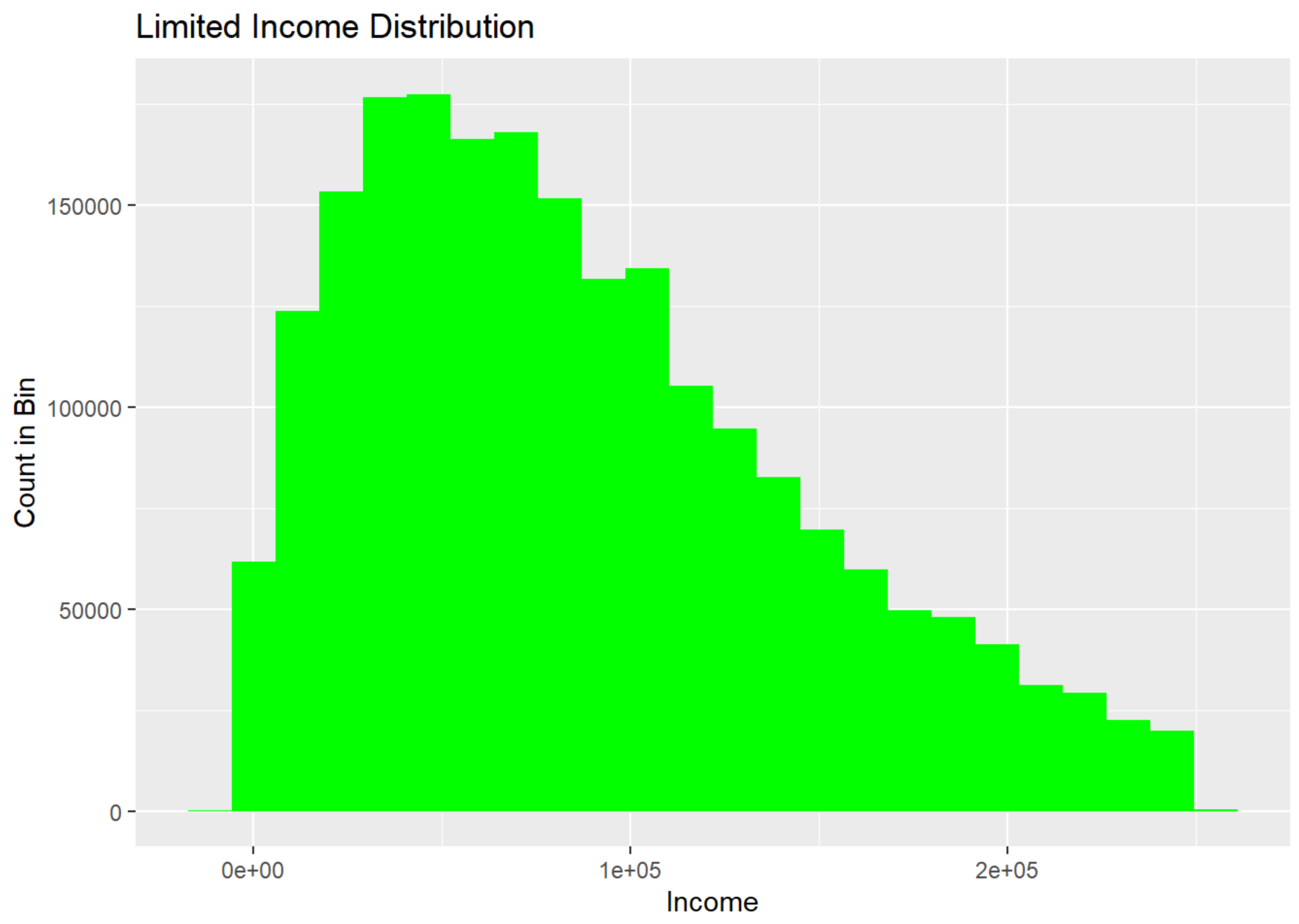
Univariate Visualization

Income

Income is a fairly simply variable that is easy to begin with.



We see that most observations are within the first three bins, revealing a right skew to the data. Next, we see what happens if we cut off that right tail.



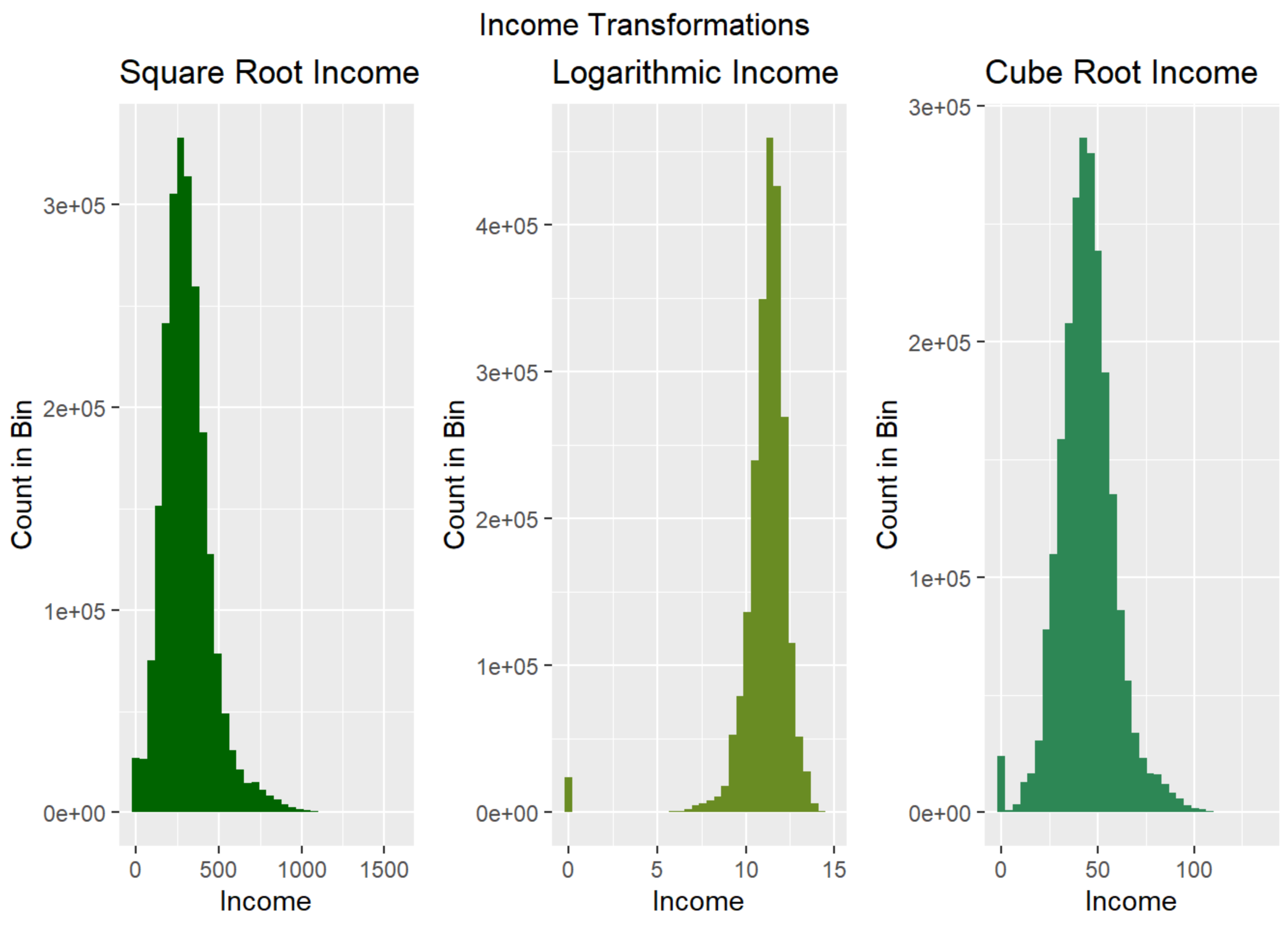
In this model, we do not see as much fall-off as we would perhaps like for a normal model. To fix this, we could try statistical transformations for our data.

```
## Warning in sqrt(spm_inc): NaNs produced

## Warning in sqrt(spm_inc): NaNs produced
```

```
## Warning: Removed 978 rows containing non-finite values (`stat_bin()`).
```

```
## Warning in log(spm_inc + 1): NaNs produced

## Warning in log(spm_inc + 1): NaNs produced
```

```
## Warning: Removed 978 rows containing non-finite values (`stat_bin()`).
## Removed 978 rows containing non-finite values (`stat_bin()`).
```



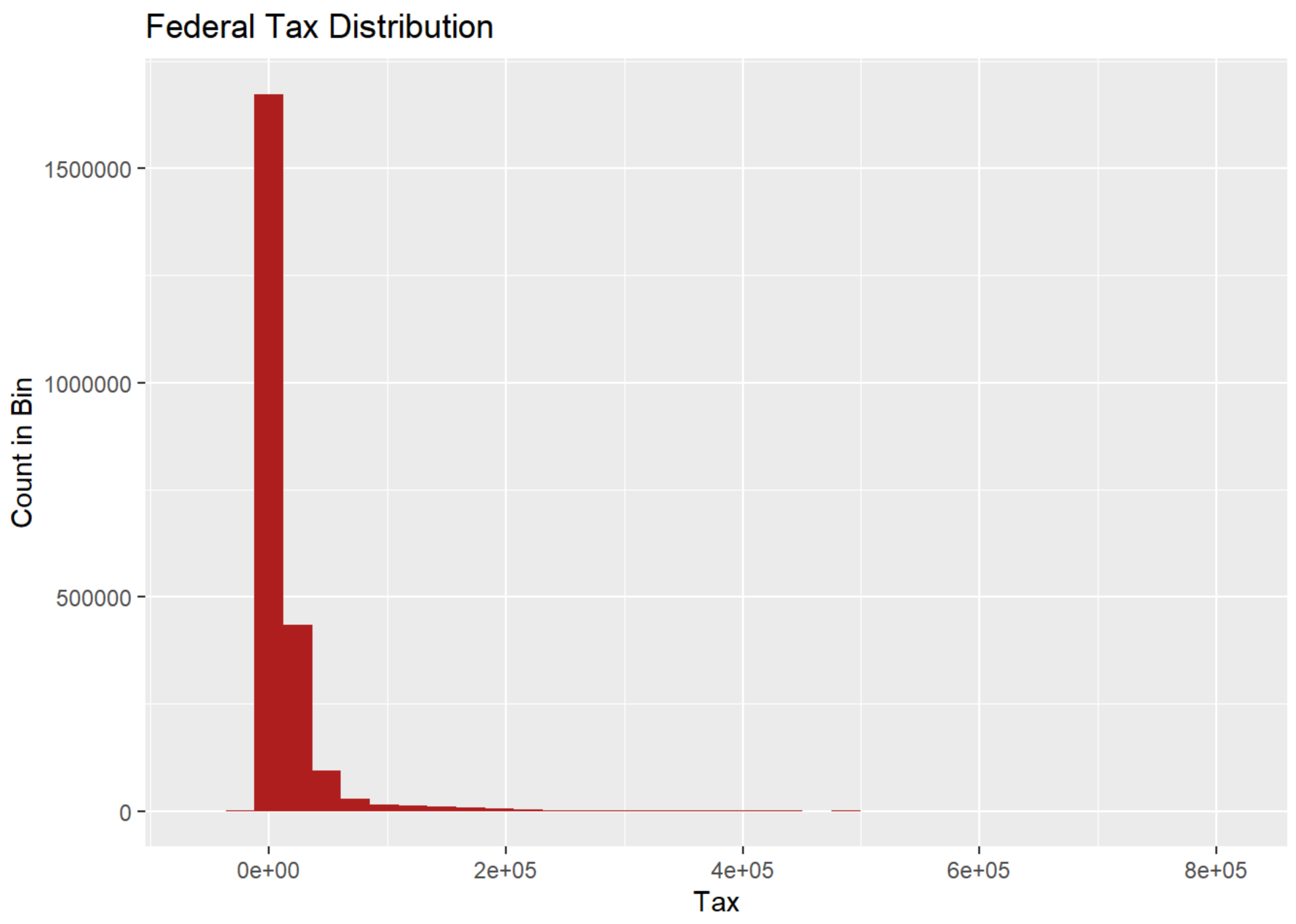
We see that the cube-root does the best job of reducing the skew of our data. The square-root still has quite a bit of skew present, and the logarithm actually goes too far and gives us some left skewing.

However, for our purposes, we want to reduce skewing, so we will prefer data removal over data transformation. Since we are using such a large dataset, there should not be any concerns relating to underfitting our models.

Tax

Federal Tax is another fairly easy variable that should be fairly similar to income. Its distribution is shown next.

```
# Plotting the distribution of our tax variable.
ggplot(df, aes(x=fed_tax)) + geom_histogram(bins = 36, fill="firebrick") +
  labs(title="Federal Tax Distribution", x ="Tax", y = "Count in Bin")
```



We see that we are finding observations with negative tax, which might make sense if an individual receives more credits than they pay in actual taxes. however, we have a massive right skew similar to what we saw in the distribution of income.

In later stages, we will consider only those observations paying less than $100,000 in taxes, which should closer represent the general population.

While I was playing with the data, I found an interesting result relating to the individuals paying greater than $100,000 in taxes.

```
print(paste("Count of Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)])))
```

```
## [1] "Count of Individuals Paying > $100,000 in Federal Taxes: 52698"
```

```
print(paste("Count of \"No\" Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)&(df$group == "no")])))
```

```
## [1] "Count of \"No\" Individuals Paying > $100,000 in Federal Taxes: 41491"
```

We have more than 10,000 individuals paying more than $100,000 who fall into one of our aid or poverty group. If well examine further, we find the following.

```
print(paste("Count of \"Poor\" Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)&(df$group == "poor")])))
```

```
## [1] "Count of \"Poor\" Individuals Paying > $100,000 in Federal Taxes: 172"
```

```
print(paste("Count of \"Aid\" Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)&(df$group == "aid")])))
```

```
## [1] "Count of \"Aid\" Individuals Paying > $100,000 in Federal Taxes: 11035"
```

It is interesting that we see individuals who count as impoverished having to pay more than $100,000 in taxes. If we explore this further…

```
print(paste("Count of Federally Poor Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)&(df$off_pov == 1)])))
```

```
## [1] "Count of Federally Poor Individuals Paying > $100,000 in Federal Taxes: 171"
```

```
print(paste("Count of Supplementally Poor Individuals Paying > $100,000 in Federal Taxes:",
            length(df$fed_tax[(df$fed_tax > 100000)&(df$spm_pov == 1)])))
```

```
## [1] "Count of Supplementally Poor Individuals Paying > $100,000 in Federal Taxes: 1"
```

These observations might need to be examined further to explain why these individuals are impoverished, but having to pay so much in taxes. Is it an error, or is the definition of poverty faulty?

Outlier Results

We can do the same general analysis with all of our numeric variables. However, what we are really looking to do is remove high leverage values to reduce bias in our model. Because of this, we will rely on statistical methods to remove values greater than 3 Standard Deviations from the Mean of our centerpoints.

```
# Loop through our columns to print out their data types.
for (i in (colnames(df))) {
  print(paste(i, "-", class(df[[i]])))
}
```

```
## [1] "st - factor"
## [1] "off_pov - factor"
## [1] "age - numeric"
## [1] "mar - factor"
## [1] "sex - factor"
## [1] "edu - factor"
## [1] "race - factor"
## [1] "hispanic - factor"
## [1] "agi - numeric"
## [1] "hi_prem - numeric"
## [1] "moop - numeric"
## [1] "spm_pov - factor"
## [1] "num_kid - numeric"
## [1] "num_adlt - numeric"
## [1] "mortgage - factor"
## [1] "spm_res - numeric"
## [1] "spm_inc - numeric"
## [1] "snap - numeric"
## [1] "house_sub - numeric"
## [1] "slunch - numeric"
## [1] "energy - numeric"
## [1] "wic - numeric"
## [1] "fed_tax - numeric"
## [1] "fed_tax_bc - numeric"
## [1] "eitc - numeric"
## [1] "fica - numeric"
## [1] "st_tax - numeric"
## [1] "cap_xpen - numeric"
## [1] "wk_xpen - numeric"
## [1] "cc_xpen - numeric"
## [1] "spm_hi_prem - numeric"
## [1] "med_xpen - numeric"
## [1] "mc_pb - numeric"
## [1] "cohabit - numeric"
## [1] "ui_kids - factor"
## [1] "group - factor"
```

Because all of our variables are either factors or numeric, we can use a simple if statement to find the mean and standard deviations of all of our variables.

```r
# Loop through our columns to print out their data types.
for (i in (colnames(df))) {
  if(class(df[[i]]) == "numeric") {   # Select only numeric variables.
    x_bar <- mean(df[[i]])   # Calculate mean of column.
    x_dev <- sd(df[[i]])   # Calculate standard deviation of column.
    x_top <- x_bar + 3*x_dev   # Find upper bound of confidence interval.
    x_bot <- x_bar - 3*x_dev   # Find lower bound of confidence interval.

    # Print results.
    print(paste(i, ": ", x_bar, ", ", x_dev, ", ", x_top, ", ", x_bot, sep=""))
  }
}
```

```
## [1] "age: 54.5697730405043, 16.6224603303289, 104.437154031491, 4.70239204951756"
## [1] "agi: 87769.2387412412, 113772.998055272, 429088.232907058, -253549.755424576"
## [1] "hi_prem: 1771.39339195666, 4325.00200919614, 14746.3994195451, -11203.6126356318"
## [1] "moop: 1293.99747798575, 2917.60446385797, 10046.8108695597, -7458.81591358817"
## [1] "num_kid: 0.546720662521117, 1.01100618971044, 3.57973923165243, -2.4862979066102"
## [1] "num_adlt: 2.14937008467571, 0.950965918814728, 5.0022678411199, -0.703527671768472"
## [1] "spm_res: 86352.4304470039, 83431.5038289906, 336646.941933976, -163942.081039968"
## [1] "spm_inc: 116439.6197606, 123169.115536781, 485946.966370943, -253067.726849742"
## [1] "snap: 358.864797365543, 1471.46252437278, 4773.25237048387, -4055.52277575278"
## [1] "house_sub: 114.074299637776, 1089.35628118378, 3382.14314318911, -3153.99454391355"
## [1] "slunch: 198.416400761484, 529.895583019515, 1788.10314982003, -1391.27034829706"
## [1] "energy: 20.6699930865998, 198.97634017502, 617.599013611661, -576.259027438461"
## [1] "wic: 9.92522802921667, 102.380611147699, 317.067061472315, -297.216605413882"
## [1] "fed_tax: 12823.0127721634, 29972.0330312651, 102739.111865959, -77093.0863216319"
## [1] "fed_tax_bc: 14153.6773965562, 29182.9317891878, 101702.47276412, -73395.1179710072"
## [1] "eitc: 268.913152164867, 1002.1792903051, 3275.45102308016, -2737.62471875042"
## [1] "fica: 6318.97627572396, 6643.13455905522, 26248.3799528896, -13610.4274014417"
## [1] "st_tax: 3290.51622337517, 6963.0628426073, 24179.7047511971, -17598.6723044467"
## [1] "cap_xpen: 2277.59408773281, 2658.65768222533, 10253.5671344088, -5698.37895894319"
## [1] "wk_xpen: 1984.81325024804, 1562.03243907823, 6670.91056748274, -2701.28406698666"
## [1] "cc_xpen: 303.232413191466, 2018.18142914045, 6357.77670061281, -5751.31187422988"
## [1] "spm_hi_prem: 2370.07515933614, 4502.67074062825, 15878.0873812209, -11137.9370625486"
## [1] "med_xpen: 6079.04336444021, 6916.75253815037, 26829.3009788913, -14671.2142500109"
## [1] "mc_pb: 519.658319792975, 1076.43788652614, 3748.9719793714, -2709.65533978545"
## [1] "cohabit: 0.0676390452075847, 0.251125530860523, 0.821015637789152, -0.685737547373983"
```

From this list, we will ignore cohabit, num_kid, num_adlt, age, and our subsidy variables. We really want to focus on our earning variables. Additionally, since the bottom of all of our ranges is below 0, which should not be too common in these variables, we will exclusively filter the top ranges.

```r
# How much data we had before.
print(paste("Num Rows Before Filtering:", nrow(df)))
```

```
## [1] "Num Rows Before Filtering: 2294991"
```

```r
# Filters according to what is shown above.
df <- df |>
  filter((df$agi < 430000)&(df$hi_prem < 15000)&(df$moop < 10000)&
           (df$spm_res < 350000)&(df$spm_inc < 500000)&(df$fed_tax < 100000)&
           (df$fed_tax_bc < 100000)&(df$eitc < 3300)&(df$fica < 27500)&
           (df$st_tax < 25000)&(df$cap_xpen < 10500)&(df$wk_xpen < 7000)&
           (df$cc_xpen < 6500)&(df$spm_hi_prem < 16000)&(df$med_xpen < 27500)&
           (df$mc_pb < 4000))

# How much data we have after.
print(paste("Num Rows After Filtering:", nrow(df)))
```

```
## [1] "Num Rows After Filtering: 2010975"
```

Since most of these rules have to do with financial variables, there is a lot of overlap in the cases that they cover. However, this full transformation should make our sample better reflect the population we are interested in, removing primarily “no-group” observations that we already have plenty of.

Save Changes