Data Pre-processing

Data Loading

To load the extracts created on the https://cps.ipums.org/cps/index.shtml platform in RStudio, we require the ".XML" file and ".DAT" file corresponding to the extract as well as the library "ipumsr" which will enable RStudio to read the xml and load the data accordingly. We also use the "openxlsx" library to store the cleaned data in an XLSX format for ease of exploration in Excel.

```
list.of.packages <- c("ipumsr", "openxlsx", "tidyverse", "tidycensus", "tigris")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library('ipumsr')
library('tidyverse')
library('openxlsx')

set.seed(123) # Setting a seed for reproducibility

ddi <- read_ipums_ddi("cps_00004.xml")
data <- read_ipums_micro(ddi)
```

Data Cleaning and Filtering

In this pre-processing step, we apply various rules to clean and filter our extract. Below we describe the rules applied as well as the rationale behind the rule:

1. `YEAR == 2023`: Selects data only for the year 2023, focusing on a specific year for analysis
2. `INCTOT != 999999999`: Excludes cases where total income is coded as 999999999, which represents missing or invalid values.
3. `AGE != 0`: Filters out cases where age is coded as 0 (invalid/missing value)
4. `!is.na(CITIZEN)`: Removes cases where citizenship status is missing, ensuring all selected cases have a valid citizenship status.

   1- Born in U.S, 2- Born in U.S. outlying, 3- Born abroad of American parents, 4- Naturalized citizen, 5- Not a citizen
5. `!is.na(IND)`: Eliminates cases where industry information is missing, ensuring all selected cases have a valid industry code.
6. `IND != 0`: Excludes cases where industry code is 0 (invalid/missing value)
7. `METFIPS < 99998`: Excludes cases where the metropolitan area code is 99999 and 99998 (invalid/missing value)

8. `RACE != 999`: Excludes cases where race is coded as 999 (invalid/missing value)
9. `!is.na(RACE)`: Removes cases where race information is missing, ensuring all selected cases have a valid race code.
10. `EDUC != 999`: Excludes cases where education level is coded as 999, assuming it's an invalid or missing value.
11. `INCTOT > 0`: Filters out cases where total income is less than or equal to 0
12. `EMPSTAT == 10`: Selects cases where employment status is coded as 10, indicating being employed.
13. `CLASSWKR %in% c(21,22,25,27,28)`: Selects cases where class of worker is in the specified values (21, 25, 27, 28), which represent specific categories of worker status.

    21- Wage/salary, private, 22- Private, for profit, 24- Wage/salary, government, 25- Federal government employee, 27- State government employee, 28- Local government employee
14. `WKSTAT %in% c(11,12,14,15)`: Selects cases where work status is in the specified values (11, 12, 14, 15), representing specific categories of work status.

    11 Full-time hours (35+), usually full-time 12 Part-time for non-economic reasons, usually full-time 14 Full-time hours, usually part-time for economic reasons 15 Full-time hours, usually part-time for non-economic reasons
15. `UHRSWORKT < 997`: Excludes cases where usual hours worked per week are greater than or equal to 997 (invalid/missing value)

Further, post cleaning of the data extract, we drop the columns which would not be useful in modelling/cannot be used as features such as YEAR, ID, and certain variables used for filtering

```r
# Data Cleaning Process
data = data %>% filter(
  YEAR == 2023,
  INCTOT != 999999999,
  AGE != 0,
  !is.na(CITIZEN),
  !is.na(IND),
  IND != 0,
  METFIPS < 99998,
  RACE != 999,
  !is.na(RACE),
  EDUC != 999,
  INCTOT > 0,
  EMPSTAT == 10,
  CLASSWKR %in% c(21,22,24,25,27,28),
  WKSTAT %in% c(11,12,14,15),
  UHRSWORKT < 997
)

data <- data %>%
  select(-YEAR, -SERIAL, -CPSID,
         -PERNUM, -COUNTY, -YRIMMIG,
         -NATIVITY, -EMPSTAT, -LABFORCE,
         -CLASSWKR, -DURUNEMP, -WKSTAT,
         -FTOTVAL, -INCRENT, -EITCRED,
         -MARGTAX, -STATETAX, -STATAXAC,
         -TAXINC, -REGION)
```

Feature Engineering

Based on our literature review, we also implement a log transform of our target variable which is "INCTOT" which denotes the Total personal income of the individual. This is done primarily due to a few reasons:

1. **Reduce Skewness:** Income data often exhibits skewness, ie, there are a few very high earners and a larger number of people clustered near the median income. Log transformation compresses the scale, bringing those high earners closer to the rest of the data and creating a more symmetrical distribution. This is important because many statistical methods assume normality in the data
2. **Linearize Relationship:** In our model, we are also interested in observing the **percentage change** in income rather than the absolute difference. Logarithms have a convenient property where a change in the log of income reflects a percentage change in the original income. This can make it easier to interpret the relationship between income and other economic factors in the model
3. **Improved Performance:** Log-transformed income may improve the performance of our models by making the relationship between income and other variables more linear

```
# Performing Log transformation on target variable for future analysis

data$log_INCTOT <- log(data$INCTOT)
```

Further, we also explore a few categorical columns which have an exceptionally large number of categories. The columns are namely: 1. OCC (Occupation Code): ~**520 Categories** 2. IND (Industry Code): ~**260 Categories** 3. METFIPS (Metropolitan Area FIPS Code): ~**300 Categories**

For OCC and IND, we were able to locate Mapping files `Mapping.xlsx` which enable us to group various OCC codes and IND codes into their respective classes, vastly reducing the number of categories to <50. In `Mapping.xlsx`, we have used separate sheets for each important feature.

For METFIPS, we used `tidycensus` and `tigris` packages, using which we mapped the Metropolitan areas to their respective populations, and ranked them according to their sizes to create a numerical feature. The underlying assumption for this is, not considering cost-of-living and taxes, larger cities tend to be more lucrative in terms of opportunity, and consequently would have higher wages and would attract more people.

Further, we use additional mapping files to map codes OCC, IND, STATEFIPS, and METFIPS with their respective Text Values so that we can interpret the Exploratory Data Analysis results

```
wb <- loadWorkbook("Mapping.xlsx")
occ <- read.xlsx(wb, sheet = "OCC")
ind <- read.xlsx(wb, sheet = "IND")
state <- read.xlsx(wb, sheet = "STATEFIPS")
met <- read.xlsx(wb, sheet = "METFIPS")

library(tidycensus)
library(tigris)
options(tigris_use_cache = TRUE)

pop <- get_estimates(geography = "metropolitan statistical area/micropolitan statistical area", variable
pop <- pop %>% filter(variable == "POPESTIMATE")
pop <- pop %>% select(c(GEOID, value))

ctc <- c("STATEFIP", "METFIPS", "AGE", "SEX", "RACE", "CITIZEN", "OCC", "IND", "UHRSWORKT", "EDUC", "IN
data <- data %>% mutate_at(vars(ctc), as.integer)
```

```r
data <- left_join(data, occ, by = "OCC")
data <- left_join(data, ind, by = "IND")
data <- left_join(data, state, by = "STATEFIP")
data <- left_join(data, met, by = "METFIPS")

data <- merge(data, pop, by.x = "METFIPS", by.y = "GEOID", all.x = TRUE)

# Checking for unique Metropolitan areas where population is missing
unique(subset(data, is.na(value))$MET)

# Unique Met Areas are :"California-Lexington Park, MD", "Cleveland-Elyria, OH", "Dayton, OH", "Prescot
# We manually find the population of these 4 areas and add to the data

data$value[which(data$MET == "California-Lexington Park, MD")] = 113000
data$value[which(data$MET == "Cleveland-Elyria, OH")] = 2063132
data$value[which(data$MET == "Dayton, OH")] = 135944
data$value[which(data$MET == "Prescott, AZ")] = 47603

# Create a Rank using "value" (population) column
uni_pop <- unique(data[, c("MET", "value")])
uni_pop$MET_CODE <- rank(uni_pop$value, ties.method = "min")

data <- merge(data, uni_pop, by.x = c("MET", "value"), by.y = c("MET", "value"), all.x = TRUE)

# Drop population value since we obtained RANK for each Metro Area
data <- data %>% select(-value)

# Ordering Columns
data <- data[, c("METFIPS", "MET", "MET_CODE", "STATEFIP", "STATE", "AGE", "SEX", "RACE", "CITIZEN", "E
```

Saving the Dataset

Finally, we save the data. We use this cleaned, processed data for exploratory Data Analysis

```r
write.csv(data, "eda_data_clean.csv", row.names = FALSE)
```