Comparative Analysis of US Personal Earned Income Across Presidential Terms: A Sample Survey Study

STAT344 Group
P.R.I.M.E.
"Population Research Institute for Methodological Excellence"

Contents

Team Members & Contributions:

- **Jiangzhe Liu** (*Group Leader*) - ***33914714***:
  - **Project Design**
  - **Data Fetching**
  - **Exploratory Data Analysis**
  - **Sampling Code**
  - **Report Modification**
- **Siyi Man** - ***67937664***:
  - **Target Parameter Determination**
  - **Report Introduction Part**
- **Yucheng Gong** - ***96710447***:
  - **Data Cleaning & Preprocessing**
  - **Report Main Analysis Part**
- **Jinyuan Zhou** - ***41159278***:
  - **Sampling Simulation Code**
  - **Report Discussion Part**

1 Project Overview

This study aims to conduct an objective analysis of economic well-being among **US citizens** by comparing personal income levels across different presidential administrations, specifically focusing on the **Trump** (2016-2020) and **Biden** (2021-2022) presidencies. Through systematic sampling and statistical analysis of IPUMS USA microdata, our **primary objective** is: to *evaluate whether there are significant differences in citizen economic outcomes(i.e. change in* ***personal yearly total income*** *& change in* ***proportion of people whose income is above a certain threshold****) between these administrative periods.*

1.1 Data Source

The sampling population is derived from The Integrated Public Use Microdata Series (**IPUMS USA**), focusing on US citizens' income, gender, age as well as other 21 demographic variables(excluding sampling ID & sampling weights & dummy variables) in **3 pivotal years**:

- **2016**: Pre-Trump presidency baseline
- **2020**: End of Trump's term
- **2022**: Mid-term of Biden's presidency

1.2 Target Parameters

In our dataset, there're 6 personal income types:

1. **INCTOT**: total personal income

- It reports each respondent's total pre-tax personal income or losses from all sources for the previous year

2. **INCWAGE**: wage and salary income

- It reports each respondent's total pre-tax wage and salary income, that is, money received as an employee for work or services performed

3. **INCWELFR**: welfare(public assistance) income

- It reports how much pre-tax income (if any) the respondent received during the previous year from various public assistance programs

4. **INCRETIR**: retirement income

- It reports how much pre-tax retirement, survivor, and disability pension income, other than Social Security, the respondent received during the previous year

5. **INCSUPP**: supplementary security income

- It reports how much pre-tax income (if any) the respondent received from Supplemental Security Income (SSI) during the previous year

6. **INCEARN**: total personal earned income

- It reports income earned from wages or a person's own business or farm for the previous year

However, we wish to only focus on 1 specific type of income that is **most relevant to people's well-being**, which we here define as ***how fair the income is compatible to the efforts taken***.

So we conducted a correlation analysis, the results are shown in **Figure 1, 2 & 3**. And we noticed that ***INCEARN*** has the highest correlation with ***UHRSWORK***[1] in all 3 years. Thus it is safe to assume that it is a good indicator of the effort taken.

[1]Usual number of hours a person worked per week

Moreover, in *economic research*, ***INCEARN*** is also a critical metric for assessing the economic well-being of citizens. Thus, using this metric as the research subject will allow us to **better analyze the income distribution* of U.S. residents during the terms of two different presidents *with higher credibility*.

Thus, our research will mainly focus on this type of income.

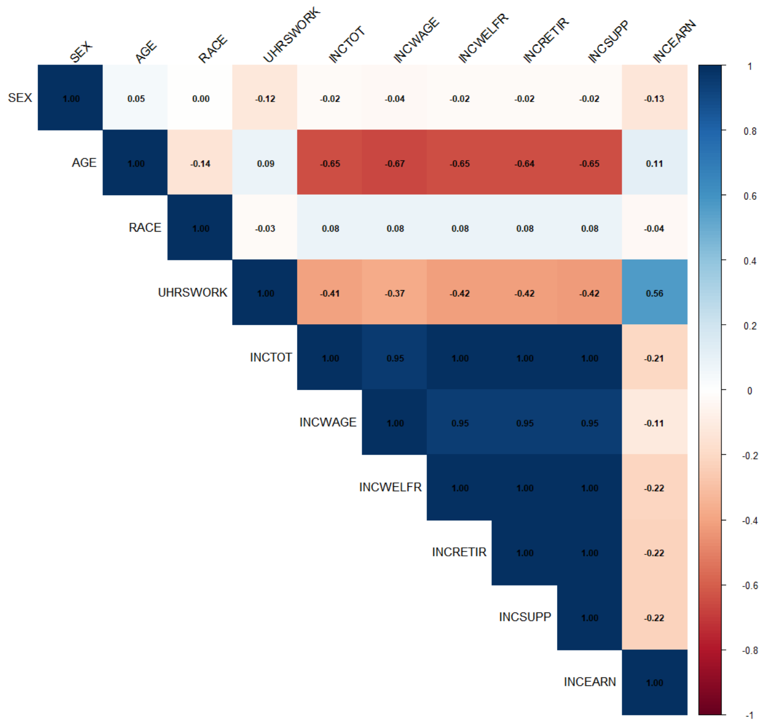


Figure 1: Correlation between consecutive variables in 2016

1.2.1 Determining Income Threshold

After determining the *1st target parameter* —— ***average yearly earned income of all US citizens***, we are also interested in a *2nd target parameter* —— ***the proportion of people who can earn higher than a certain amount in a single year***

The **reason** we are interested in this parameter is that:

- Through analyzing the above-medium income earners, this study can capture *a critical segment of the population* whose financial performance is often **indicative of larger macroeconomic trends**.
- So by adding a comparison between the proportion of people with high INCEARN levels across president administrations, we can make a **more objective evaluation** of the economic trajectories shaped by the two presidents and their different policy priorities.

Moreover, according to *Economic Policy Institute's research*, the top 5% in the United States has an income of **$308,487**, **$322,349**, and **$335,891** in **2019**, **2020**, and **2021** respectively.

Since our report mainly focused on the Trump (2016-2020) and Biden (2021-2022) presidencies, here we will adopt ***$30,000*** as the threshold for indicating **"above medium income individuals"**.

1.2.2 Other considerations

And finally, to eliminate the impact of **inflation** on our analysis, all the income data we use in this project are U.S. CPI-adjusted dollar volumes (adjusted to *2010*).

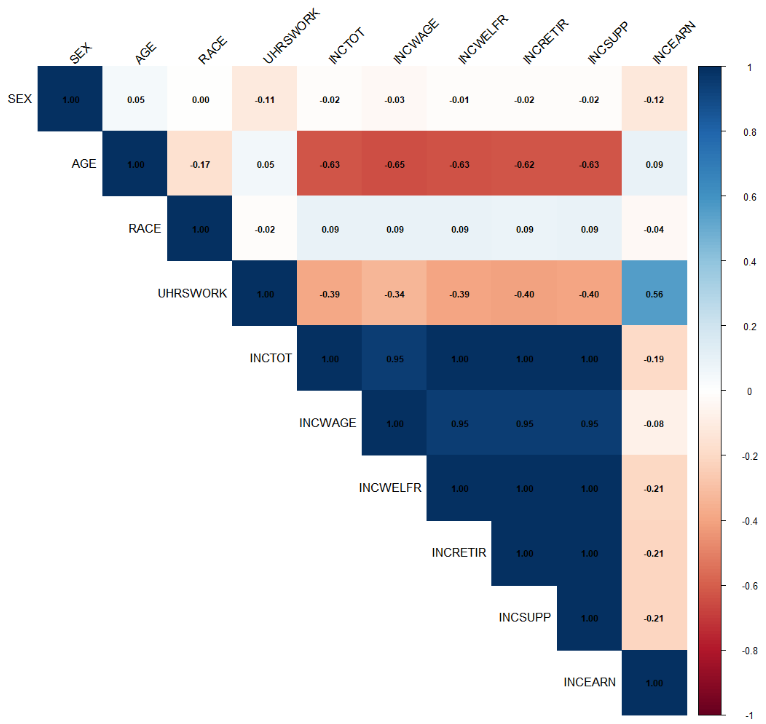


Figure 2: Correlation between consecutive variables in 2020

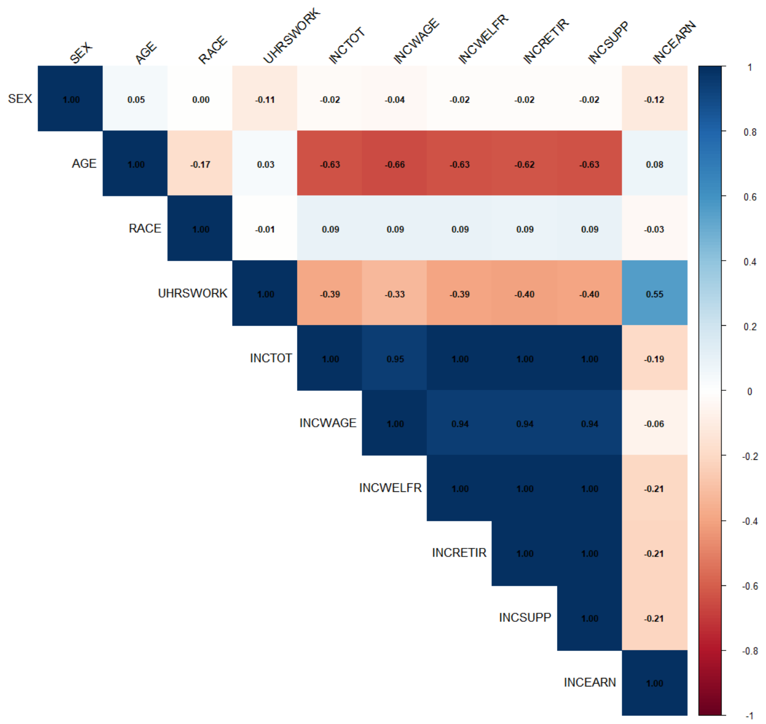


Figure 3: Correlation between consecutive variables in 2022

2 Main Analysis

2.1 Sampling Methodology

We will mainly use 2 sampling methods in this project: ***SRS*** & ***Stratified Sampling***. And we will use **proportional allocation** to determine the sample size of each stratum in **Stratified Sampling**.

- For the **1st** target parameter: *mean income*,

  we will use **both vanilla & ratio estimation** under **SRS**; and **only vanilla estimation** under **Stratified Sampling**. (in **ratio estimation**, we will use ***UHRSWORK*** as the auxiliary variable.)

- For the **2nd** target parameter: *proportion of people whose income is above a certain threshold*,

  we will **drop the ratio estimation** under **SRS**, with other methods **remaining the same**.

And we will also conduct multiple simulation samplings to compare the performance of these methods.

The reason we choose these 2 sampling methods are:

- both *SRS* & *Stratified Sampling* are **simple & easy to implement**, therefore **cost-effective** in real-world applications
- *Stratified Sampling* ensures **representation of all important subpopulations** in the final sample & **reduces sampling error** by creating homogeneous groups

2.1.1 Choosing Stratification Variables

Based on our dataset, we selected **4 key demographic variables** for stratification:

1. **Geographic Location (STATEICP)**
2. **Gender (SEX)**
   - Binary classification (Male/Female)
3. **Age Groups (AGE_GROUP)**
   - Custom categorization:
     - 0-18: Youth/Dependents
     - 19-30: Early Career
     - 31-50: Mid Career
     - 51-70: Late Career
     - 71+: Retirement Age
4. **Race (RACE)**
   - Nine categories:
     - White
     - Black/African American
     - American Indian or Alaska Native
     - Chinese
     - Japanese
     - Other Asian or Pacific Islander
     - Other race
     - Two major races
     - Three or more major races

These stratification variables were chosen because they:

- Represent key demographic factors affecting income distribution
- Have clear, well-defined categories
- Provide sufficient sample sizes within each stratum

- Allow for meaningful further socioeconomic analysis (e.g. income inequality, poverty rate, etc.)

2.1.2 Determining Sample Size

To determine an appropriate sample size for our study, we considered several factors:

1. **Precision Requirements**:
   - We only aim for a **m.o.e. of less than ±1%** for our proportion estimates
   - (the value of income is consecutive, hence we cannot give a safe guess of the variance of income)
   - Desired confidence level: 95%
2. **Resource Constraints**:
   - Computing limitations for personal computer
   - Cost of sampling a single person in real world can be high
3. **Population Characteristics**:
   - High variability in income data & data across years

   - Large population size (N > 2,500,000 for all 3 years)

Based on these considerations, we calculated the required sample size using:

$$n = \frac{z_{0.975}^2 \cdot \hat{p}(1-\hat{p})}{\delta^2}$$

where:

- $\hat{p}$ is our guessed ratio of the true proportion of people whose income is above the given threshold, and we believe this ratio is **less than 0.35**.

- δ is the maximum margin of error we can accept, which we set to **0.01**.
- $z_{0.975}$ is the z-score corresponding to the 95% confidence level, which is **1.96**.

Finally, we chose a sample size of $\mathbf{n = 8740}$ for all our sampling schemes later used, which:

- ***Provides sufficient precision for our estimates***
- ***Minimizes potential real-world sampling costs***
- Allows for efficient & swift computation in our simulation studies
- Large enough to support stratified sampling with multiple strata

2.1.3 Formulas

For ***SRS-Vanilla estimation*** on each year's data, we use the following formula to calculate the estimate & its SE:

$$\hat{\bar{y}}_{vanilla} = \frac{1}{n}\sum_{i\in S} y_i$$

$$SE(\hat{\bar{y}}_{vanilla}) = \sqrt{(1-\frac{n}{N})\frac{s^2}{n}}, \text{ where } s^2 = \frac{1}{n-1}\sum_{i\in S}(y_i - \hat{\bar{y}}_{vanilla})^2$$

For ***SRS-Ratio estimation*** on each year's data, we use the following formula to calculate the estimate & its SE:

$$\hat{\bar{y}}_{ratio} = \hat{R}\bar{x}_P, \text{ where } \hat{R} = \frac{\sum_{i\in S} y_i}{\sum_{i\in S} x_i}$$

$$SE(\hat{\bar{y}}_{ratio}) = \sqrt{(1-\frac{n}{N})\frac{s_e^2}{n}}, \text{ where } s_e^2 = \frac{1}{n-1}\sum_{i\in S}(y_i - \hat{R}x_i)^2$$

where:

- y_i is the income for person i
- x_i is the usual number of hours worked per week for person i
- $\bar{x}_P$ is the known population mean of the usual number of hours worked per week
- n is the sample size
- N is the population size

For ***Stratified Sampling- Vanilla estimation*** on each year's data, we use the following formula to calculate the estimate & its SE:

$$\bar{y}_{str} = \hat{\bar{y}}_p = \sum_{h=1}^{H} \frac{n_h}{n} \cdot \bar{y}_{S_h}$$

$$SE(\bar{y}_{str}) = \sqrt{\sum_{h=1}^{H}\left(\frac{n_h}{n}\right)^2 \cdot (1-\frac{n_h}{n}) \cdot \frac{s_{s_h}^2}{n_h}}, \text{ where } s_{s_h}^2 = \frac{1}{n_h - 1}\sum_{i\in S_h}(y_i - \bar{y}_{S_h})^2$$

where:

- n_h is the sample size of stratum h
- n is the total sample size
- $\bar{y}_{S_h}$ is the sample mean of stratum h

And by the simulation of conducting all sampling schemes on 3 years' data for **500** times, we have the following average estimates & CIs for mean estimate & proportion estimate (***Figure 4***)[2]:

2.2 Results & Conclusion

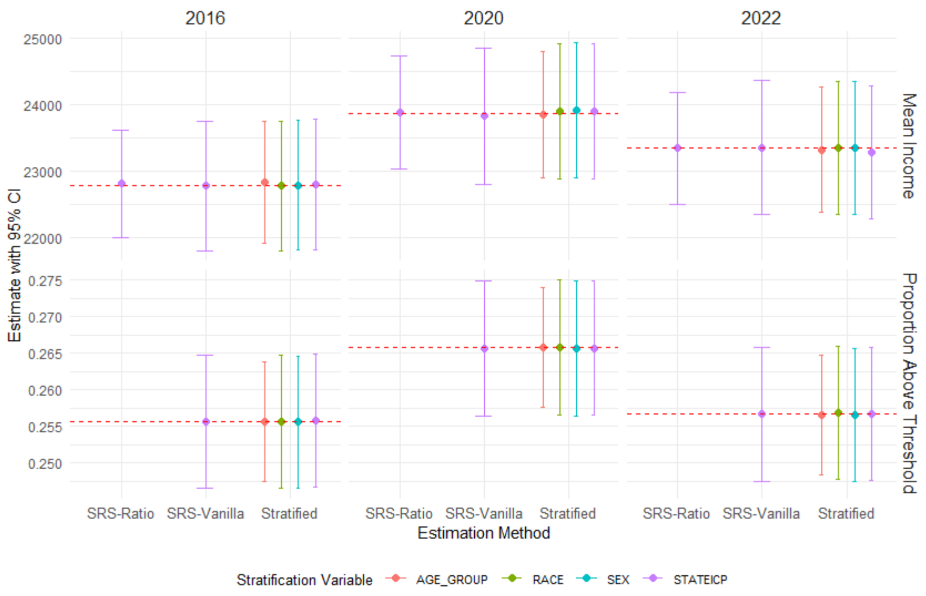


Figure 4: Avg. Estimates and CIs Using Various Sampling and Estimation Methods Across 2016, 2020, 2022

[2]Red dashed lines in ***Figure 4*** indicate true population values

Looking at the above figure, we can easily see that:

- Mean income showed an *increasing trend* **from 2016** (~22,800) **to 2020** (~24,000), followed by a *slight decline* from **2020** (~24,000) to **2022** (~23,500)
- Proportion of people whose income is above \$30000 also showed an *increasing trend* **from 2016** (~0.25) **to 2020** (~0.28), followed by a *slight decline* from **2020** (~0.28) to **2022** (~0.27)

This suggests U.S. had witnessed a **better growth in people's well-being in terms of income** during *Trump's presidency (2016-2020)* but then a **slight drop** during *Biden's early term (2020-2022).*

Which then leads to the ***conclusion*** that:

1. There is a significant difference in citizen economic outcomes between these 2 administrative periods:
   1. $\Delta_{2016-2020}(\text{mean income}) \approx \$1200 > 0$
   2. $\Delta_{2020-2022}(\text{mean income}) \approx -\$500 < 0$
   3. $\Delta_{2016-2020}(\text{proportion of people whose income is above \$30000}) \approx 3\% > 0$
   4. $\Delta_{2020-2022}(\text{proportion of people whose income is above \$30000}) \approx -1\% < 0$

2. *Trump* had done a **better job** than *Biden* in boosting the income levels of the **general population** during his presidency

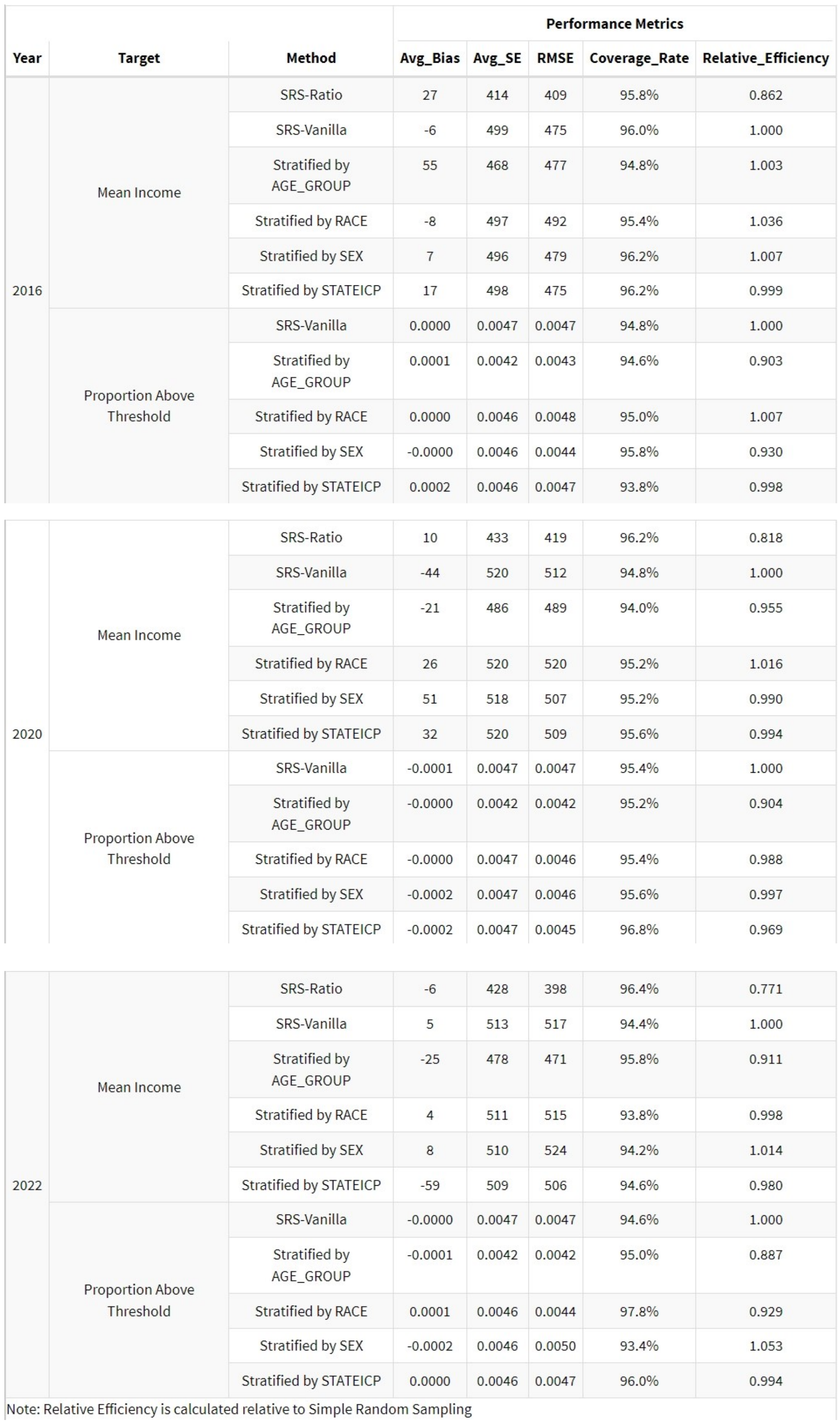
2.3 Comparison of Sampling Methods

Beyond our *conclusion*, we want to take a further look at the ***performance metrics of different estimation methods*** throughout the whole simulation process in all 3 years.

These metrics include:

1. **Average Bias**
   - Avg. difference between the estimate and the true population parameter
2. **Average SE**
   - Avg. standard errors of the estimates
3. **RMSE**
   - Rooted Mean Squared Error of estimate compared to the true population parameter
4. **Coverage Rate**
   - Proportion of CIs that contain the true population parameter
5. ***Relative Efficiency***
   - Ratio of the variance of the RMSE to the RMSE of the SRS estimator

And the results are shown in ***Figure 5***:



			Performance Metrics				
Year	Target	Method	Avg_Bias	Avg_SE	RMSE	Coverage_Rate	Relative_Efficiency
2016	Mean Income	SRS-Ratio	27	414	409	95.8%	0.862
		SRS-Vanilla	-6	499	475	96.0%	1.000
		Stratified by AGE_GROUP	55	468	477	94.8%	1.003
		Stratified by RACE	-8	497	492	95.4%	1.036
		Stratified by SEX	7	496	479	96.2%	1.007
		Stratified by STATEICP	17	498	475	96.2%	0.999
	Proportion Above Threshold	SRS-Vanilla	0.0000	0.0047	0.0047	94.8%	1.000
		Stratified by AGE_GROUP	0.0001	0.0042	0.0043	94.6%	0.903
		Stratified by RACE	0.0000	0.0046	0.0048	95.0%	1.007
		Stratified by SEX	-0.0000	0.0046	0.0044	95.8%	0.930
		Stratified by STATEICP	0.0002	0.0046	0.0047	93.8%	0.998
2020	Mean Income	SRS-Ratio	10	433	419	96.2%	0.818
		SRS-Vanilla	-44	520	512	94.8%	1.000
		Stratified by AGE_GROUP	-21	486	489	94.0%	0.955
		Stratified by RACE	26	520	520	95.2%	1.016
		Stratified by SEX	51	518	507	95.2%	0.990
		Stratified by STATEICP	32	520	509	95.6%	0.994
	Proportion Above Threshold	SRS-Vanilla	-0.0001	0.0047	0.0047	95.4%	1.000
		Stratified by AGE_GROUP	-0.0000	0.0042	0.0042	95.2%	0.904
		Stratified by RACE	-0.0000	0.0047	0.0046	95.4%	0.988
		Stratified by SEX	-0.0002	0.0047	0.0046	95.6%	0.997
		Stratified by STATEICP	-0.0002	0.0047	0.0045	96.8%	0.969
2022	Mean Income	SRS-Ratio	-6	428	398	96.4%	0.771
		SRS-Vanilla	5	513	517	94.4%	1.000
		Stratified by AGE_GROUP	-25	478	471	95.8%	0.911
		Stratified by RACE	4	511	515	93.8%	0.998
		Stratified by SEX	8	510	524	94.2%	1.014
		Stratified by STATEICP	-59	509	506	94.6%	0.980
	Proportion Above Threshold	SRS-Vanilla	-0.0000	0.0047	0.0047	94.6%	1.000
		Stratified by AGE_GROUP	-0.0001	0.0042	0.0042	95.0%	0.887
		Stratified by RACE	0.0001	0.0046	0.0044	97.8%	0.929
		Stratified by SEX	-0.0002	0.0046	0.0050	93.4%	1.053
		Stratified by STATEICP	0.0000	0.0046	0.0047	96.0%	0.994

Note: Relative Efficiency is calculated relative to Simple Random Sampling

Figure 5: Sampling Methods Performance Comparison

From ***Figure 4 & 5*** combined, we can make the following comparisons:

- All 3 estimation methods (SRS-Ratio, SRS-Vanilla, and Stratified) **successfully captured the true population parameters** (as shown by the red dashed lines falling within the confidence intervals)
  - All methods produced **consistent** estimates with overlapping confidence intervals (coverage rate ≈ 95%)
  - All methods yield **unbiased** estimates as they cluster around the true population values
- ***SRS*** generally performs better than ***Stratified Sampling*** in terms of all metrics, which is expected as stratified sampling *requires more information about the population* & is *more complex to implement*.
- ***SRS-Ratio*** outperforms ***SRS-Vanilla*** in terms of RMSE **in all 3 years**, which indicates that the ***ratio estimation*** is more precise than ***vanilla estimation*** when the ***auxiliary variable*** is **strongly correlated** with the **income**.
- In ***Stratified Sampling***, different stratification variables (AGE_GROUP, RACE, SEX, STATEICP) produced very similar results, and the **strata variable that has the highest relative efficiency differs** across these 3 years. Specifically:
  - For **2016**, taking ***RACE*** as strata variable can yield a **higher relative efficiency** than *all other methods used* in both mean(≈ 1.036) & proportion(≈ 1.007) estimation
  - For **2020**, taking ***RACE*** as strata variable can only yield a **higher relative efficiency** than all other methods in both mean(≈ 1.016) estimation, while in proportion estimation, taking ***SEX*** as strata variable can yield the highest efficiency than other strata variables used, but still lower than ***SRS***(≈ 0.997)
  - For **2022**, taking ***SEX*** as strata variable can yield a **higher relative efficiency** than *all other methods used* in both mean(≈ 1.014) & proportion(≈ 1.053) estimation

3 Discussion

3.1 Limitations & Considerations

Limitations for our project mainly comes from the following aspects:

- **Sampling methods:**
  - ***SRS***
    - Higher chance of missing rare but important instances (considering weights)
    - Can be logistically challenging to implement in real world in large geographical areas
    - Can be inefficient for highly variated data
    - Can be more expensive and time-consuming when population is widely dispersed
  - ***Stratified Sampling***
    - Requires prior knowledge of key population strata characteristics & sizes
    - More complex & costly to implement than SRS
    - *With proportional allocation specifically*:
      - May **not provide enough samples from smaller strata** for meaningful analysis
      - Not optimal when **costs of sampling vary** between strata
      - **Less efficient when variability of interest is similar** across strata
- **Data:**
  - **Data Timeliness**
    - Our newest data during the Biden administration is only available until 2022, which may not be ***sufficiently long*** to capture the long-term trends in income distribution.
    - Though it is already suffice to make the above conclusions.
  - **Population Data is actually a large sample**
    - (See ***Section 3.2***)
- **Lack of real-world factors:**
  - There're many other factors in real world that may affect people's income & well-being that were *not included* in our dataset & analysis, such as:
    - **Economic lag effects from policy implementation**
    - **Impact of COVID-19 pandemic on 2020-2022 data**
    - **External global economic factors**

    * **Structural changes in the economy**

3.2 Data Justification

Another thing to note is that our *population* data from IPUMS USA already represents ***a sample of*** the total US population, we consider using it as our sampling population methodologically sound for several reasons:

- IPUMS USA employs scientifically rigorous sampling methods that **ensure representativeness of the US population**
- The large sample size and sophisticated sampling design of IPUMS data provides **sufficient statistical power**
- The data undergoes thorough **quality control** & **harmonization** processes
- The sampling weights (*though not used in this project*) provided by IPUMS allow for appropriate **population-level inference**
- This approach is **consistent with common practice** in social science research using complex survey data

3.3 Generalizability

As for the **generalizability** of our conclusions, we have the following considerations:

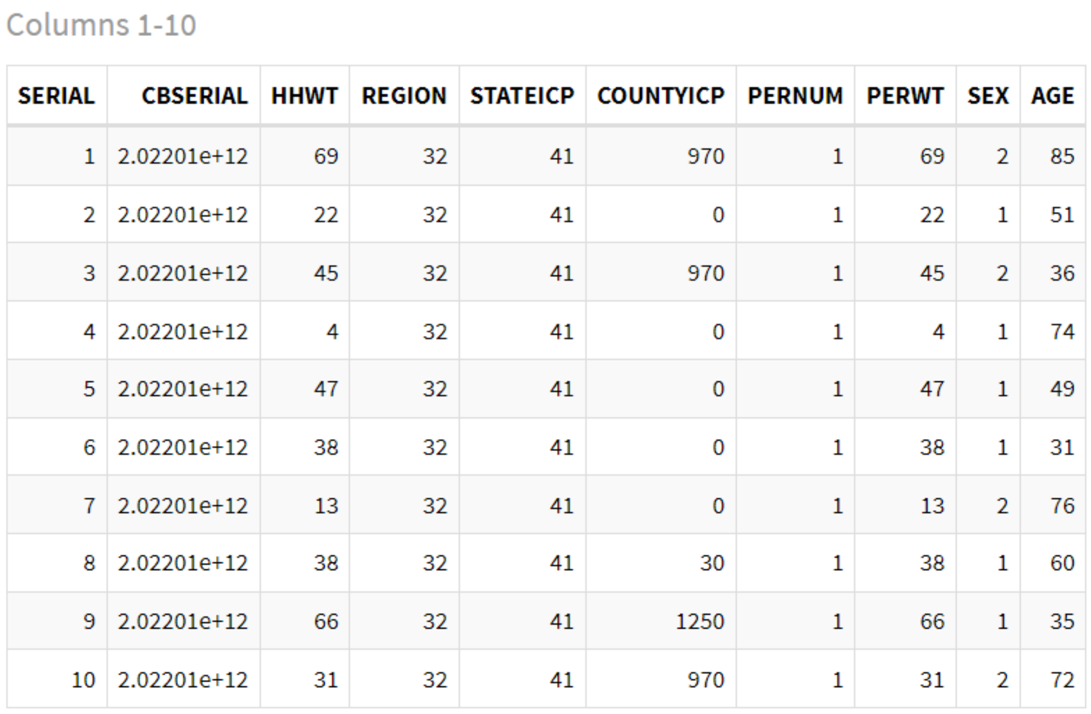
- First, while our conclusions can be reasonably generalized to the larger US citizen population due to IPUMS USA's rigorous sampling methodology and our comprehensive stratification approach, caution should be exercised when extending these findings to **non-citizen populations** or **those operating in informal economies**.

- Second, generalizing these conclusions to other presidential terms **requires careful consideration**, as our analysis period (2016-2022) encompassed unique circumstances, particularly the **COVID-19 pandemic's unprecedented economic impact**. Additionally, different presidential terms may face **distinct economic challenges**, **policy environments**, and **global conditions** that could significantly influence income patterns, making **direct comparisons potentially problematic** without accounting for these contextual differences.

4 Appendix

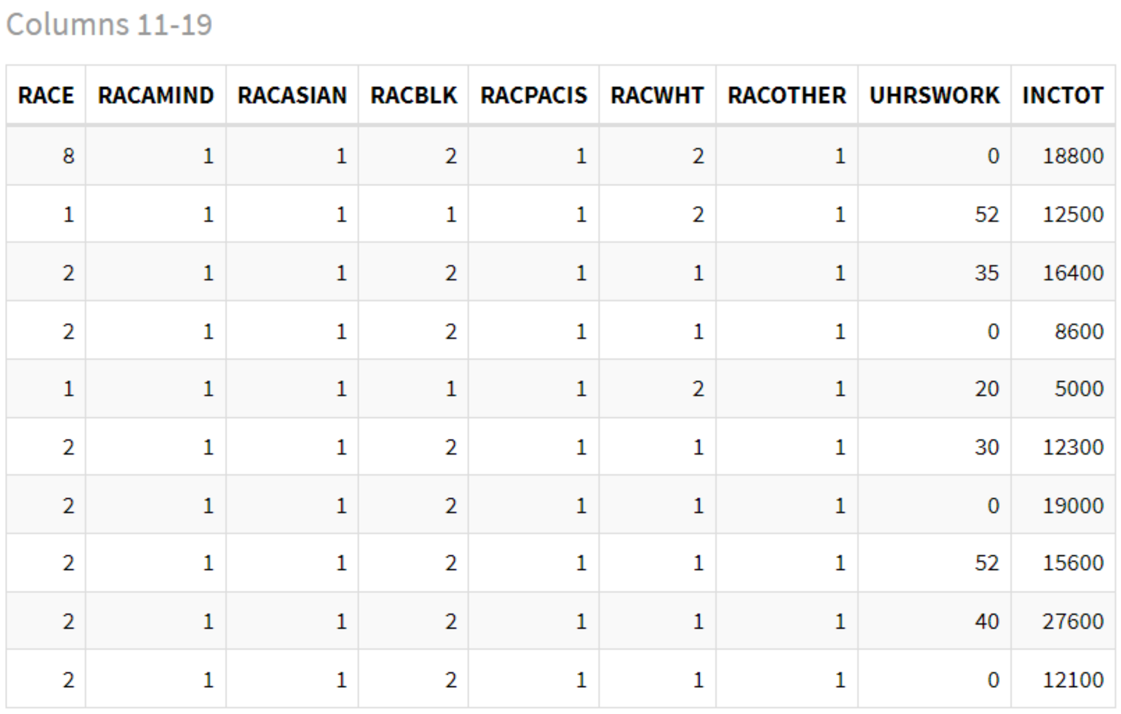
4.1 Dataset Overview

Here's a glimpse of the first 10 rows from our 2022 dataset:



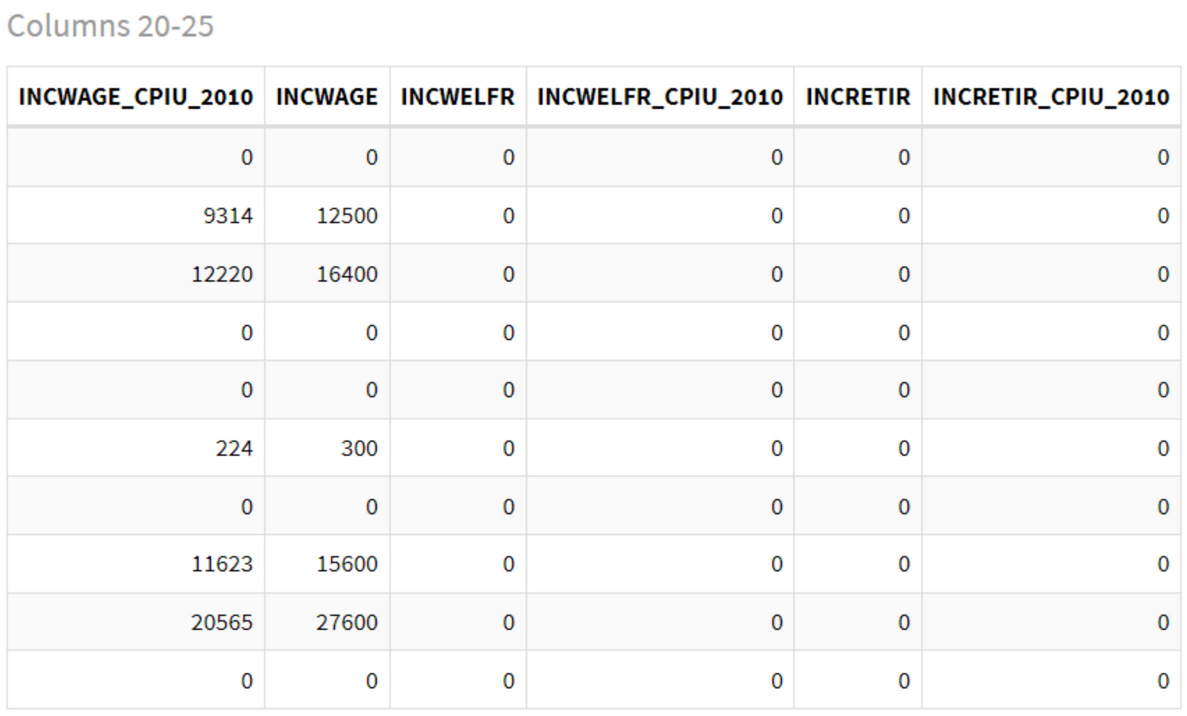
Columns 1-10

SERIAL	CBSERIAL	HHWT	REGION	STATEICP	COUNTYICP	PERNUM	PERWT	SEX	AGE
1	2.02201e+12	69	32	41	970	1	69	2	85
2	2.02201e+12	22	32	41	0	1	22	1	51
3	2.02201e+12	45	32	41	970	1	45	2	36
4	2.02201e+12	4	32	41	0	1	4	1	74
5	2.02201e+12	47	32	41	0	1	47	1	49
6	2.02201e+12	38	32	41	0	1	38	1	31
7	2.02201e+12	13	32	41	0	1	13	2	76
8	2.02201e+12	38	32	41	30	1	38	1	60
9	2.02201e+12	66	32	41	1250	1	66	1	35
10	2.02201e+12	31	32	41	970	1	31	2	72

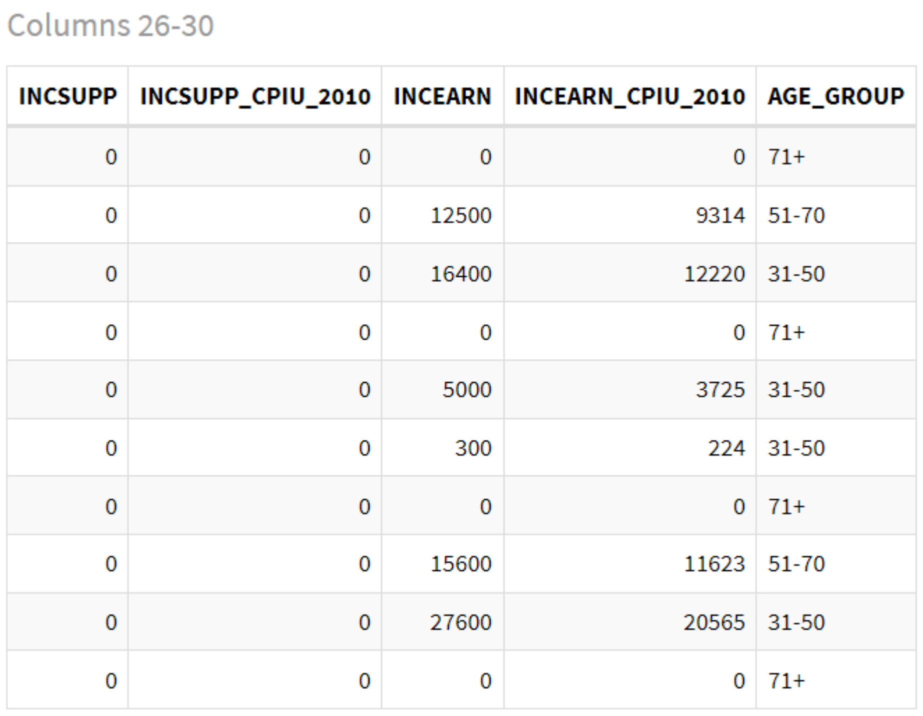


Columns 11-19

RACE	RACAMIND	RACASIAN	RACBLK	RACPACIS	RACWHT	RACOTHER	UHRSWORK	INCTOT
8	1	1	2	1	2	1	0	18800
1	1	1	1	1	2	1	52	12500
2	1	1	2	1	1	1	35	16400
2	1	1	2	1	1	1	0	8600
1	1	1	1	1	2	1	20	5000
2	1	1	2	1	1	1	30	12300
2	1	1	2	1	1	1	0	19000
2	1	1	2	1	1	1	52	15600
2	1	1	2	1	1	1	40	27600
2	1	1	2	1	1	1	0	12100


Columns 20-25

INCWAGE_CPIU_2010	INCWAGE	INCWELFR	INCWELFR_CPIU_2010	INCRETIR	INCRETIR_CPIU_2010
0	0	0	0	0	0
9314	12500	0	0	0	0
12220	16400	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
224	300	0	0	0	0
0	0	0	0	0	0
11623	15600	0	0	0	0
20565	27600	0	0	0	0
0	0	0	0	0	0


Columns 26-30

INCSUPP	INCSUPP_CPIU_2010	INCEARN	INCEARN_CPIU_2010	AGE_GROUP
0	0	0	0	71+
0	0	12500	9314	51-70
0	0	16400	12220	31-50
0	0	0	0	71+
0	0	5000	3725	31-50
0	0	300	224	31-50
0	0	0	0	71+
0	0	15600	11623	51-70
0	0	27600	20565	31-50
0	0	0	0	71+

4.2 Code

4.2.1 Main Script (main.R)

Main analysis script containing data processing, simulation setup and result visualization.

```
# nolint: object_name_linter
#installation of necessary packages
source("utils.R")
library(tidyverse)
library(ggplot2)
library(ipumsr)



################################################################################
#------------------------------Research Description-------------------------------#
################################################################################



# We have 6 kinds of income types, and we should choose the one(or some) that can
# most reflect the well-being of the average citizen in the US.
# The 6 income types are:
#   1. INCTOT: total personal income
#         INCTOT reports each respondent's total pre-tax personal income or losses from all sources for
#   2. INCWAGE: wage and salary income
#         INCWAGE reports each respondent's total pre-tax wage and salary income, that is, money receiv
#   3. INCWELFR: welfare(public assistance) income
#         INCWELFR reports how much pre-tax income (if any) the respondent received during the previous
#   4. INCRETIR: retirement income
#         INCRETIR reports how much pre-tax retirement, survivor, and disability pension income, other
#   5. INCSUPP: supplementary security income
#         INCSUPP reports how much pre-tax income (if any) the respondent received from Supplemental Se
#   6. INCEARN: total personal earned income
#         INCEARN reports income earned from wages or a person's own business or farm for the previous



################################################################################
#------------------Data Import & Data Processing - 1st Stage-----------------------#
################################################################################

#----------DO NOT try to import the .csv file directly. It will not work.----------#
#----------You need to import the data using the ipumsr package----------#
#----------Also, DO NOT change the file name of <*.dat>----------#

ddi <- read_ipums_ddi("./data/usa_222016_01.xml")
rawdf_222016 <- read_ipums_micro(ddi)

#split data into 3 years: 2016, 2020, 2022

rawdf_2016 <- rawdf_222016[rawdf_222016$YEAR == 2016,]
rawdf_222016 <- rawdf_222016[rawdf_222016$YEAR != 2016,]
rawdf_2020 <- rawdf_222016[rawdf_222016$YEAR == 2020,]
rawdf_222016 <- rawdf_222016[rawdf_222016$YEAR != 2020,]
rawdf_2022 <- rawdf_222016[rawdf_222016$YEAR == 2022,]
rawdf_222016 <- rawdf_222016[rawdf_222016$YEAR != 2022,]
rm(rawdf_222016)
```

```r
#check if there's any NA in each df
colSums(is.na(rawdf_2016))
colSums(is.na(rawdf_2020))
colSums(is.na(rawdf_2022))
#No NA in any df!!!


#Select only relevant columns that are relevant to our study:
#i.e. keep what we care about
select_relevant_columns <- function(df) {
  df %>%
    select(-c(YEAR, SAMPLE, CLUSTER, STRATA, GQ, WKSWORK1, RACED))
}

df_2016 <- select_relevant_columns(rawdf_2016)
df_2020 <- select_relevant_columns(rawdf_2020)
df_2022 <- select_relevant_columns(rawdf_2022)


##################################################################################
#---------------Find plausible auxiliary variable for Ratio Estimation-------------#
##################################################################################


explore_correlations <- function(df, title = "Finding Auxiliary Variable", save_plot = FALSE) {
  library(corrplot)

  # Create results directory if it doesn't exist
  if (save_plot && !dir.exists("./results")) {
    dir.create("./results")
  }

  # Get numeric columns
  numeric_cols <- sapply(df, is.numeric)
  numeric_data <- df[, numeric_cols]

  # Calculate correlation matrix
  cor_matrix <- cor(numeric_data, use = "complete.obs")

  # Create correlation plot
  if (save_plot) {
    # Save to file
    png(paste0("./results/", title, "_correlation_plot.png"),
        width = 1200, height = 1200, res = 100)
    corrplot(cor_matrix,
             method = "color",
             type = "upper",
             tl.col = "black",
             tl.srt = 45,
             addCoef.col = "black",
             number.cex = 0.7,
             title = paste("Correlation Plot -", title))
    dev.off()
  } else {
    # Display in R interface
    corrplot(cor_matrix,
             method = "color",
             type = "upper",
             tl.col = "black",
             tl.srt = 45,
             addCoef.col = "black",
             number.cex = 0.7,
             title = paste("Correlation Plot -", title))
  }

  # Return correlation matrix
  return(cor_matrix)
}

# To see whether there is a single variable that is highly correlated with a person's earned income
explore_correlations(df_2016[,-c(1,2,3,4,5,6,7,8,12:17,20,23,25,27,29)], "2016_Data", save_plot = TRUE)
explore_correlations(df_2020[,-c(1,2,3,4,5,6,7,8,12:17,20,23,25,27,29)], "2020_Data", save_plot = TRUE)
explore_correlations(df_2022[,-c(1,2,3,4,5,6,7,8,12:17,20,23,25,27,29)], "2022_Data", save_plot = TRUE)
# THen can find that "UHRSWORK" is highly correlated with "INCEARN" in all 3 years
# So we can use "UHRSWORK" as the auxiliary variable for Ratio Estimation



######################################################################################
#---------------------------------Estimation Planning--------------------------------#
######################################################################################

# Next, what we're going to do is:
# 1. Find the threshold for INCEARN, say its value is income.thres, a pre-set constant
#    Determine a uniform total sample size in consideration of real-problem cost constraints & other fa
# 2. We have 2 variables of interst:
#       1st: average personal earned income in the US, that is, average INCEARN (colname: INCEARN_CPIU_
#       2nd: the proportion of people whose INCEARN is above income.thres
#
#    Then we want to sample a certain proportion of the population to estimate these 2 variables
#    Find the true value of these 2 variables using our full dataset for results evaluation
# 3. Apply these estimation methods on 2 different target variables under 11 estimation methods:
#       Target Variable 1: mean INCEARN
#       Methods used:
#         Simple Random Sampling:
#            vanilla estimation
#            ratio estimation
#            (take UHRSWORK as auxiliary variable)
#         Stratified Sampling with proportional allocation:
#           use 4 strata differention variables separately:
#             - STATE
#               (colname: STATEICP)
#             - SEX
#               (colname: SEX)
#             - AGE GROUP
#               (colname: AGE)
#                (need to add a new column to the df to categorize age into groups)
#                (Age groups: 0-18, 19-30, 31-50, 51-70, 71+)
#              - RACE
#                (colname:RACE)
#                (integer 1 to 9 each representing a different race:
#                 1: White,
#                 2: Black/African American,
#                 3: American Indian or Alaska Native,
#                 4: Chinese,
#                 5: Japanese,
#                 6: Other Asian or Pacific Islander,
#                 7: Other race, nec,
#                 8: Two major races,
#                 9: Three or more major races
#                 )
#        Target Variable 2: proportion of people whose INCEARN is above income.thres
#        Methods used:
#          Simple Random Sampling:
#             vanilla estimation
#          Stratified Sampling with proportional allocation:
#           use 4 strata differention variables separately:
#             - STATE
#               (colname: STATEICP)
#             - SEX
#               (colname: SEX)
#             - AGE GROUP
#               (colname: AGE)
#               (need to add a new column to the df to categorize age into groups)
#               (Age groups: 0-18, 19-30, 31-50, 51-70, 71+)
#             - RACE
#               (colname:RACE)
# 4. Each time step (3) is taken, there should be 11 different estimates and their corresponding SEs an
#    and they should be stored for further comparison
# 4. Simulate step 2 for <> times
# 5. Compare the average value of the following results for each of the 11 estimation scheme:
#       estimate
#       SE
#       CI
#    with the corresponding true population parameter
# 6. Plot the results in a vertical-layout 6 row, 1 col grid
#    with each graph in a row representing a different estimation method



####################################################################################
#-----------------------------Data Processing - 2nd Stage--------------------------#
####################################################################################

# Add age groups to the datasets
add_age_groups <- function(df) {
  df %>%
    mutate(AGE_GROUP = case_when(
      AGE <= 18 ~ "0-18",
      AGE <= 30 ~ "19-30",
      AGE <= 50 ~ "31-50",
      AGE <= 70 ~ "51-70",
      TRUE ~ "71+"
    ))
}

df_2016 <- add_age_groups(df_2016)
df_2020 <- add_age_groups(df_2020)
df_2022 <- add_age_groups(df_2022)


####################################################################################
####################################################################################
#----------------------------------Main Analysis-----------------------------------#
####################################################################################
####################################################################################

# All the functions necessary for the main analysis are defined in utils.R
# Referring to utils.R for more details

####################################################################################
#------------------------------Define Global Params--------------------------------#
####################################################################################
# 1. Set random seed for reproducibility
set.seed(2024)

# 2. Pre-determined parameters
income.thres <- 30000  # threshold for INCEARN
sample_size <- 8740   # total sample size for each simulation

# 3. Set stratification variables and which years' data to analyse
strata_vars <- c("STATEICP", "SEX", "AGE_GROUP", "RACE")
years <- c("2016", "2020", "2022")

# 4. number of simulations
n_sims <- 500


####################################################################################
#-----------------------------Sampling Simulation----------------------------------#
####################################################################################


# Initialize results storage
big_results <- list()

# Run simulations for each year
for(year in years) {
  df <- get(paste0("df_", year))
  big_results[[year]] <- compare_sampling_methods(
    data = df,
    tarv_1 = "INCEARN_CPIU_2010",
```

```r
    tarv_2 = "INCEARN_CPIU_2010",
    thres = income.thres,
    n = sample_size,
    strata_vars = c("STATEICP", "SEX", "AGE_GROUP", "RACE"),
    n_simulations = n_sims
  )
}


#####################################################################################
#-----------------------------Result Visualization----------------------------------#
#####################################################################################


# Visualize estimation results across years
yearly_plots <- plot_yearly_performance(big_results)

#---------------------------------------------------------------------------#
#----------------use this to export overall result plot------------------#
#---------------------------------------------------------------------------#
yearly_plots


# Save plots (unnecessary)
ggsave("mean_estimation_yearly_comparison.png",
       yearly_plots$mean_plots,
       width = 15,
       height = 20)
ggsave("proportion_estimation_yearly_comparison.png",
       yearly_plots$proportion_plots,
       width = 15,
       height = 20)


# Display results as tables (unnecessary)
yearly_comparisons <- compare_yearly_performance(big_results)

# Save mean estimation results as HTML (unnecessary)
mean_table <- knitr::kable(yearly_comparisons$mean_estimates,
                          format = "html",
                          caption = "Mean Estimation Performance Across Years")
writeLines(mean_table, "./results/mean_estimation_table.html")

# Save proportion estimation results as HTML (unnecessary)
prop_table <- knitr::kable(yearly_comparisons$proportion_estimates,
                          format = "html",
                          caption = "Proportion Estimation Performance Across Years")
writeLines(prop_table, "./results/proportion_estimation_table.html")

# Save results as CSV (unnecessary)
write.csv(yearly_comparisons$mean_estimates,
          "./results/mean_estimation_results.csv",
          row.names = FALSE)
```

```r
write.csv(yearly_comparisons$proportion_estimates,
          "./results/proportion_estimation_results.csv",
          row.names = FALSE)


###################################################################################
#----------------------------Performance Comparison--------------------------------#
###################################################################################

# Here, we will look at the boarder picture of the performance metrics of
# different estimation methods throughout the whole simulation process in all 3 years
# Including:
#   1. Average Bias
#      (Avg. difference between the estimate and the true population parameter)
#   2. Average SE
#      (Avg. standard errors of the estimates)
#   2. RMSE
#      (Rooted Mean Squared Error of estimate compared to the true population parameter)
#   3. Coverage Rate
#      (Proportion of CIs that contain the true population parameter)
#   5. **Relative Efficiency**
#      (Ratio of the variance of the RMSE to the RMSE of the SRS estimator)

perf_table = create_performance_table(big_results)
perf_table

# And we can see that... (see main report for further explanation)
```

4.2.2 Utility Functions (utils.R)

Helper functions for sampling, estimation and visualization used in the main analysis.

```r
library(dplyr)
library(stats)
library(ggplot2)
library(haven)
library(knitr)
library(kableExtra)


###################################################################################
#-------------------------------File Description-----------------------------------#
###################################################################################

# This is a file containing necessary code snippets & sampling functions & simulation functions for the
# The functions are designed to be used in the main.R file under the same directory
# The file is structured as follows:
# 1. Snippet functions
# 2. Sampling functions
# 3. Simulation functions
# 4. Visualization functions
# 5. Other sampling functions
```

```r
#####################################################################################
#-------------------------------File Description-----------------------------------#
#####################################################################################

# This is a file containing necessary code snippets & sampling functions & simulation functions for the
# The functions are designed to be used in the main.R file under the same directory
# The file is structured as follows:
# 1. Snippet functions
# 2. Sampling functions
# 3. Simulation functions
# 4. Visualization functions
# 5. Other sampling functions

#####################################################################################
#--------------------------------------Basic---------------------------------------#
#####################################################################################


############################################################################ Function 1.1
# Calculate proportion of a variable above a threshold
calculate_proportion <- function(x, threshold) {
  mean(x > threshold)
}

############################################################################ Function 1.2
# Calculate variance of a binary variable
calculate_proportion_var <- function(x, threshold) {
  p <- mean(x > threshold)
  var <- p * (1 - p)
  return(var)
}

#####################################################################################
#-------------------------------------Sampling-------------------------------------#
#####################################################################################


############################################################################ Function 2.1
# Simple Random Sampling -- estimating mean -- vanilla estimatior
srs_sampling_est_mean_vanilla <- function(data, tarv, n) {
  # Input validation
  if (!tarv %in% names(data)) {
    stop("Target variable not found in dataset")
  }

  N <- nrow(data)
  if (n >= N) {
    stop("Sample size must be less than population size")
  }

  # Take simple random sample
  sample.ids <- sample(N, size = n, replace = FALSE)
  sample_data <- data[sample.ids, ]
  # Calculate estimate
  est <- mean(sample_data[[tarv]])

  # Calculate standard error
  var.smp <- var(sample_data[[tarv]])
  se <- sqrt((1 - n/N) * var.smp/n)

  # Create confidence interval
  CI <- c(est - 1.96 * se,
          est + 1.96 * se)

  return(list(
    estimate = est,
    se = se,
    ci = CI
  ))
}


############################################################################# Function 2.2
# Simple Random Sampling -- estimating proportion -- vanilla estimator
srs_sampling_est_prop_vanilla <- function(data, tarv, thres, n) {
  # Input validation
  if (!tarv %in% names(data)) {
    stop("Target variable not found in dataset")
  }

  N <- nrow(data)
  if (n >= N) {
    stop("Sample size must be less than population size")
  }

  # Take simple random sample
  sample.ids <- sample(N, size = n, replace = FALSE)
  sample_data <- data[sample.ids, ]

  # Calculate proportion estimate
  est <- calculate_proportion(sample_data[[tarv]], thres)

  # Calculate standard error
  var.smp <- est * (1 - est)  # variance for binary variable
  se <- sqrt((1 - n/N) * var.smp/n)

  # Create confidence interval
  CI <- c(est - 1.96 * se,
          est + 1.96 * se)

  return(list(
    estimate = est,
    se = se,
    ci = CI
  ))
}
```

```r
############################################################################## Function 2.3
# Simple Random Sampling -- estimating mean -- ratio estimator
srs_sampling_est_mean_ratio <- function(data, tarv_1, aux_var, n) {
  # Input validation
  if (!all(c(tarv_1, aux_var) %in% names(data))) {
    stop("Target or auxiliary variables not found in dataset")
  }

  N <- nrow(data)
  if (n >= N) {
    stop("Sample size must be less than population size")
  }

  # Take simple random sample
  sample.ids <- sample(N, size = n, replace = FALSE)
  sample_data <- data[sample.ids, ]

  # Calculate ratio estimate
  r_hat <- sum(sample_data[[tarv_1]]) / sum(sample_data[[aux_var]])
  X_bar <- mean(data[[aux_var]])  # Population mean of auxiliary variable
  tarv_1.est <- r_hat * X_bar

  # Calculate variance of ratio estimate
  y <- sample_data[[tarv_1]]
  x <- sample_data[[aux_var]]
  d <- y - r_hat * x
  var_d <- var(d)

  # Standard error
  tarv_1.se <- sqrt((1 - n/N) * var_d/n)

  # Create confidence interval
  tarv_1.CI <- c(tarv_1.est - 1.96 * tarv_1.se,
                 tarv_1.est + 1.96 * tarv_1.se)

  return(list(
    estimate = tarv_1.est,
    se = tarv_1.se,
    ci = tarv_1.CI,
    ratio = r_hat
  ))
}


############################################################################## Function 2.4
# Stratified Sampling with Proportional Allocation -- estimating mean -- vanilla estimator
str_prop_sampling_est_mean_vanilla <- function(data, tarv, n, strata_var) {
  # Input validation
  if (!all(c(tarv, strata_var) %in% names(data))) {
    stop("Required variables not found in dataset")
  }
  # Calculate stratum sizes
  N <- nrow(data)
  N_hs <- table(data[[strata_var]])
  strata_names <- names(N_hs)

  # Calculate proportional allocation
  n_hs <- round((N_hs/N) * n)

  # Initialize results storage
  tarv.bar.hs <- numeric(length(strata_names))
  tarv.var.smp.hs <- numeric(length(strata_names))

  # Sample from each stratum
  STR.sample <- data.frame()
  for (i in seq_along(strata_names)) {
    stratum_data <- if(strata_var == "AGE_GROUP") {
      data[data[[strata_var]] == strata_names[i], ]
    } else {
      data[data[[strata_var]] == as.integer(strata_names[i]), ]
    }

    # Sample from strarum i
    if (nrow(stratum_data) > 0) {  # make sure this stratum have data
      sample_indices <- sample.int(nrow(stratum_data), size = n_hs[i], replace = FALSE)
      stratum_sample <- stratum_data[sample_indices, ]
      STR.sample <- rbind(STR.sample, stratum_sample)
    }
  }

  # Calculate estimate for each stratum
  tarv.bar.hs <- tapply(STR.sample[[tarv]], STR.sample[[strata_var]], mean)

  # Calculate sample variance for each stratum
  tarv.var.smp.hs <- tapply(STR.sample[[tarv]], STR.sample[[strata_var]], var)

  # Calculate overall estimate
  tarv.str.est <- sum((N_hs/N) * tarv.bar.hs)

  # Calculate standard error
  tarv.se <- sqrt(sum((N_hs/N)^2 * (1 - n_hs/N_hs) * tarv.var.smp.hs/n_hs))

  # Create confidence interval
  tarv.CI <- c(tarv.str.est - 1.96 * tarv.se,
               tarv.str.est + 1.96 * tarv.se)

  return(list(
    estimate = tarv.str.est,
    se = tarv.se,
    ci = tarv.CI,
    strata_estimates = tarv.bar.hs
  ))
}
```

```r
############################################################################ Function 2.5
# Stratified Sampling with Proportional Allocation -- estimating proportion -- vanilla estimator
str_prop_sampling_est_prop_vanilla <- function(data, tarv, thres, n, strata_var) {
  # Input validation
  if (!all(c(tarv, strata_var) %in% names(data))) {
    stop("Required variables not found in dataset")
  }

  # Calculate stratum sizes
  N <- nrow(data)
  N_hs <- table(data[[strata_var]])
  strata_names <- names(N_hs)

  # Calculate proportional allocation
  n_hs <- round((N_hs/N) * n)

  # Initialize results storage
  tarv.bar.hs <- numeric(length(strata_names))
  tarv.var.smp.hs <- numeric(length(strata_names))

  # Sample from each stratum
  STR.sample <- data.frame()
  for (i in seq_along(strata_names)) {
    stratum_data <- if(strata_var == "AGE_GROUP") {
      data[data[[strata_var]] == strata_names[i], ]
    } else {
      data[data[[strata_var]] == as.integer(strata_names[i]), ]
    }

    # sample from strarum i
    if (nrow(stratum_data) > 0) {  # make sure this stratum have data
      sample_indices <- sample.int(nrow(stratum_data), size = n_hs[i], replace = FALSE)
      stratum_sample <- stratum_data[sample_indices, ]
      STR.sample <- rbind(STR.sample, stratum_sample)
    }
  }


  # Calculate proportion estimate for each stratum
  tarv.bar.hs <- tapply(STR.sample[[tarv]], STR.sample[[strata_var]],
                        calculate_proportion, threshold = thres)

  # Calculate sample variance for each stratum
  tarv.var.smp.hs <- tapply(STR.sample[[tarv]], STR.sample[[strata_var]],
                            calculate_proportion_var, threshold = thres)

  # Calculate overall estimate
  tarv.str.est <- sum((N_hs/N) * tarv.bar.hs)

  # Calculate standard error
  tarv.se <- sqrt(sum((N_hs/N)^2 * (1 - n_hs/N_hs) * tarv.var.smp.hs/n_hs))

  # Create confidence interval
  tarv.CI <- c(tarv.str.est - 1.96 * tarv.se,
               tarv.str.est + 1.96 * tarv.se)

  return(list(
    estimate = tarv.str.est,
    se = tarv.se,
    ci = tarv.CI,
    strata_estimates = tarv.bar.hs
  ))
}


##################################################################################
#------------------------------------Simulation------------------------------------#
##################################################################################


########################################################################### Function 3.1
# Simulating different sampling & estimation methods
# + evaluating performance across different sampling methods & strata variables used
compare_sampling_methods <- function(data, tarv_1, tarv_2, thres, n, strata_vars, n_simulations = 1000)

  if (!require("progress")) {
    install.packages("progress")
    library(progress)
  }

  library(haven)

  # haven_labelled --> double
  data <- data %>%
    mutate(across(where(is.labelled), ~as.numeric(zap_labels(.))))

  # calculate true mean and proportion
  true_mean <- mean(data[[tarv_1]])
  true_prop <- mean(data[[tarv_2]] > thres)

  # prepare results storage
  results <- list()
  for(strata_var in strata_vars) {
    results[[strata_var]] <- list(
      # Target variable 1: mean income
      mean_estimates = list(
        srs_vanilla = data.frame(
          estimate = numeric(n_simulations),
          se = numeric(n_simulations),
          ci_lower = numeric(n_simulations),
          ci_upper = numeric(n_simulations)
        ),
        srs_ratio = data.frame(
          estimate = numeric(n_simulations),
          se = numeric(n_simulations),
          ci_lower = numeric(n_simulations),
          ci_upper = numeric(n_simulations)
        ),
        str_prop = data.frame(
          estimate = numeric(n_simulations),
          se = numeric(n_simulations),
          ci_lower = numeric(n_simulations),
          ci_upper = numeric(n_simulations)
        )
      ),

      # Target variable 2: proportion above threshold
      prop_estimates = list(
        srs_vanilla = data.frame(
          estimate = numeric(n_simulations),
          se = numeric(n_simulations),
          ci_lower = numeric(n_simulations),
          ci_upper = numeric(n_simulations)
        ),
        str_prop = data.frame(
          estimate = numeric(n_simulations),
          se = numeric(n_simulations),
          ci_lower = numeric(n_simulations),
          ci_upper = numeric(n_simulations)
        )
      )
    )
  }

  # setup progress bar
  pb <- progress_bar$new(
    format = paste0("Year ", year, " - Simulation progress [:bar] :percent eta: :eta"),
    total = n_simulations,
    clear = FALSE,
    width = 80
  )

  # run simulations
  for(i in 1:n_simulations) {
    # SRS only sample once in each iteration
    srs_vanilla_mean <- srs_sampling_est_mean_vanilla(data, tarv_1, n)
    srs_ratio_mean <- srs_sampling_est_mean_ratio(data, tarv_1, "UHRSWORK", n)
    srs_vanilla_prop <- srs_sampling_est_prop_vanilla(data, tarv_2, thres, n)


    # Stratified sampling
    for(strata_var in strata_vars) {
      # estimation
      str_prop_mean <- str_prop_sampling_est_mean_vanilla(data, tarv_1, n, strata_var)
      str_prop_prop <- str_prop_sampling_est_prop_vanilla(data, tarv_2, thres, n, strata_var)

      # store results for mean estimates
      results[[strata_var]]$mean_estimates$srs_vanilla[i,] <- c(
        srs_vanilla_mean$estimate,
        srs_vanilla_mean$se,
        srs_vanilla_mean$ci
      )

      results[[strata_var]]$mean_estimates$srs_ratio[i,] <- c(
        srs_ratio_mean$estimate,
        srs_ratio_mean$se,
        srs_ratio_mean$ci
      )

      results[[strata_var]]$mean_estimates$str_prop[i,] <- c(
        str_prop_mean$estimate,
        str_prop_mean$se,
        str_prop_mean$ci
      )


      # store results for proportion estimates
      results[[strata_var]]$prop_estimates$srs_vanilla[i,] <- c(
        srs_vanilla_prop$estimate,
        srs_vanilla_prop$se,
        srs_vanilla_prop$ci
      )

      results[[strata_var]]$prop_estimates$str_prop[i,] <- c(
        str_prop_prop$estimate,
        str_prop_prop$se,
        str_prop_prop$ci
      )
    }


    # update progress bar
    pb$tick()
  }

  # performance for each strata variable
  performance <- list()
  for(strata_var in strata_vars) {
    performance[[strata_var]] <- list(
      mean_estimates = list(),
      prop_estimates = list()
    )


    # performance metric for mean estimates
    for(method in c("srs_vanilla", "srs_ratio", "str_prop")) {
      performance[[strata_var]]$mean_estimates[[method]] <- list(
        estimates = results[[strata_var]]$mean_estimates[[method]]$estimate,
        ci_lower = results[[strata_var]]$mean_estimates[[method]]$ci_lower,
        ci_upper = results[[strata_var]]$mean_estimates[[method]]$ci_upper,
        bias = mean(results[[strata_var]]$mean_estimates[[method]]$estimate - true_mean),
        rmse = sqrt(mean((results[[strata_var]]$mean_estimates[[method]]$estimate - true_mean)^2)),
```

```r
        coverage = mean(results[[strata_var]]$mean_estimates[[method]]$ci_lower <= true_mean &
                       results[[strata_var]]$mean_estimates[[method]]$ci_upper >= true_mean),
        avg_ci_width = mean(results[[strata_var]]$mean_estimates[[method]]$ci_upper -
                         results[[strata_var]]$mean_estimates[[method]]$ci_lower)
      )
    }


    # performance metric for proportion estimates
    for(method in c("srs_vanilla", "str_prop")) {
      performance[[strata_var]]$prop_estimates[[method]] <- list(
        estimates = results[[strata_var]]$prop_estimates[[method]]$estimate,
        ci_lower = results[[strata_var]]$prop_estimates[[method]]$ci_lower,
        ci_upper = results[[strata_var]]$prop_estimates[[method]]$ci_upper,
        bias = mean(results[[strata_var]]$prop_estimates[[method]]$estimate - true_prop),
        rmse = sqrt(mean((results[[strata_var]]$prop_estimates[[method]]$estimate - true_prop)^2)),
        coverage = mean(results[[strata_var]]$prop_estimates[[method]]$ci_lower <= true_prop &
                       results[[strata_var]]$prop_estimates[[method]]$ci_upper >= true_prop),
        avg_ci_width = mean(results[[strata_var]]$prop_estimates[[method]]$ci_upper -
                         results[[strata_var]]$prop_estimates[[method]]$ci_lower)
      )
    }
  }


  # yield results
  return(list(
    true_values = list(
      mean = true_mean,
      proportion = true_prop
    ),
    simulation_results = results,
    performance = performance
  ))
}


############################################################################### Function 3.2
compare_strata_performance <- function(performance_results) {
  # initialize data frames
  mean_comparisons <- data.frame()
  prop_comparisons <- data.frame()

  # iterate each strata variable
  for(strata_var in names(performance_results)) {
    # comparison for mean estimates
    mean_perf <- data.frame(
      strata_var = strata_var,
      method = c("SRS-Vanilla", "SRS-Ratio", "Stratified"),
      estimate = c(
        mean(performance_results[[strata_var]]$mean_estimates$srs_vanilla$estimates),
        mean(performance_results[[strata_var]]$mean_estimates$srs_ratio$estimates),
        mean(performance_results[[strata_var]]$mean_estimates$str_prop$estimates)
      ),
      ci_lower = c(
        mean(performance_results[[strata_var]]$mean_estimates$srs_vanilla$ci_lower),
        mean(performance_results[[strata_var]]$mean_estimates$srs_ratio$ci_lower),
        mean(performance_results[[strata_var]]$mean_estimates$str_prop$ci_lower)
      ),
      ci_upper = c(
        mean(performance_results[[strata_var]]$mean_estimates$srs_vanilla$ci_upper),
        mean(performance_results[[strata_var]]$mean_estimates$srs_ratio$ci_upper),
        mean(performance_results[[strata_var]]$mean_estimates$str_prop$ci_upper)
      ),
      bias = c(
        performance_results[[strata_var]]$mean_estimates$srs_vanilla$bias,
        performance_results[[strata_var]]$mean_estimates$srs_ratio$bias,
        performance_results[[strata_var]]$mean_estimates$str_prop$bias
      ),
      rmse = c(
        performance_results[[strata_var]]$mean_estimates$srs_vanilla$rmse,
        performance_results[[strata_var]]$mean_estimates$srs_ratio$rmse,
        performance_results[[strata_var]]$mean_estimates$str_prop$rmse
      ),
      coverage = c(
        performance_results[[strata_var]]$mean_estimates$srs_vanilla$coverage,
        performance_results[[strata_var]]$mean_estimates$srs_ratio$coverage,
        performance_results[[strata_var]]$mean_estimates$str_prop$coverage
      ),
      ci_width = c(
        performance_results[[strata_var]]$mean_estimates$srs_vanilla$avg_ci_width,
        performance_results[[strata_var]]$mean_estimates$srs_ratio$avg_ci_width,
        performance_results[[strata_var]]$mean_estimates$str_prop$avg_ci_width
      )
    )
    mean_comparisons <- rbind(mean_comparisons, mean_perf)

    # comparison for proportion estimates
    prop_perf <- data.frame(
      strata_var = strata_var,
      method = c("SRS-Vanilla", "Stratified"),
      estimate = c(
        mean(performance_results[[strata_var]]$prop_estimates$srs_vanilla$estimates),
        mean(performance_results[[strata_var]]$prop_estimates$str_prop$estimates)
      ),
      ci_lower = c(
        mean(performance_results[[strata_var]]$prop_estimates$srs_vanilla$ci_lower),
        mean(performance_results[[strata_var]]$prop_estimates$str_prop$ci_lower)
      ),
      ci_upper = c(
        mean(performance_results[[strata_var]]$prop_estimates$srs_vanilla$ci_upper),
        mean(performance_results[[strata_var]]$prop_estimates$str_prop$ci_upper)
      ),
      bias = c(
        performance_results[[strata_var]]$prop_estimates$srs_vanilla$bias,
        performance_results[[strata_var]]$prop_estimates$str_prop$bias
      ),
      rmse = c(
        performance_results[[strata_var]]$prop_estimates$srs_vanilla$rmse,
        performance_results[[strata_var]]$prop_estimates$str_prop$rmse
      ),
      coverage = c(
        performance_results[[strata_var]]$prop_estimates$srs_vanilla$coverage,
        performance_results[[strata_var]]$prop_estimates$str_prop$coverage
      ),
      ci_width = c(
        performance_results[[strata_var]]$prop_estimates$srs_vanilla$avg_ci_width,
        performance_results[[strata_var]]$prop_estimates$str_prop$avg_ci_width
      )
    )
    prop_comparisons <- rbind(prop_comparisons, prop_perf)
  }

  return(list(
    mean_estimates = mean_comparisons,
    proportion_estimates = prop_comparisons
  ))
}


############################################################################# Function 3.3
# integrate simulation results across multiple years
compare_yearly_performance <- function(yearly_results) {
  # initialize data frames
  all_years_mean <- data.frame()
  all_years_prop <- data.frame()

  # iterate each year
  for(year in names(yearly_results)) {
    # get performance comparisons for each year
    year_comparisons <- compare_strata_performance(yearly_results[[year]]$performance)

    # for SRS-Vanilla, SRS-Ratio: only keep result for the 1st strata variable
    year_comparisons$mean_estimates <- year_comparisons$mean_estimates %>%
      filter((method != "SRS-Vanilla" & method != "SRS-Ratio") | strata_var == "STATEICP")

    year_comparisons$proportion_estimates <- year_comparisons$proportion_estimates %>%
      filter((method != "SRS-Vanilla" & method != "SRS-Ratio") | strata_var == "STATEICP")

    # add year column
    year_comparisons$mean_estimates$year <- year
    year_comparisons$proportion_estimates$year <- year

    # integrate results
    all_years_mean <- rbind(all_years_mean, year_comparisons$mean_estimates)
    all_years_prop <- rbind(all_years_prop, year_comparisons$proportion_estimates)
  }

  return(list(
```

```r
    mean_estimates = all_years_mean,
    proportion_estimates = all_years_prop
  ))
}




####################################################################################
#-----------------------------------Visualization----------------------------------#
####################################################################################


############################################################################# Function 4.1
# Visualize performance across multiple years
plot_yearly_performance <- function(yearly_results) {
  library(ggplot2)
  library(tidyr)
  library(dplyr)
  library(gridExtra)

  # get comparisons across years
  comparisons <- compare_yearly_performance(yearly_results)
  print(names(comparisons$mean_estimates))
  # transform into graph data
  mean_plot_data <- comparisons$mean_estimates %>%
    select(year, strata_var, method, estimate, ci_lower, ci_upper) %>%
    mutate(
      year = factor(year),
      estimate_type = "Mean Income"
    )

  prop_plot_data <- comparisons$proportion_estimates %>%
    select(year, strata_var, method, estimate, ci_lower, ci_upper) %>%
    mutate(
      year = factor(year),
      estimate_type = "Proportion Above Threshold"
    )

  # integrate plot data
  plot_data <- rbind(mean_plot_data, prop_plot_data)

  # get true value for each estimate
  true_values <- sapply(names(yearly_results), function(year) {
    c(yearly_results[[year]]$true_values$mean,
      yearly_results[[year]]$true_values$proportion)
  })

  true_value_df <- data.frame(
    year = rep(names(yearly_results), each = 2),
    estimate_type = rep(c("Mean Income", "Proportion Above Threshold"), length(yearly_results)),
    true_value = c(true_values)
  )
  # main grapg
  ggplot(plot_data, aes(x = method, y = estimate, color = strata_var)) +
    # add CI
    geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper),
                  width = 0.2,
                  position = position_dodge(width = 0.8)) +
    # add estimate
    geom_point(position = position_dodge(width = 0.8), size = 2) +
    # add inference for true value
    geom_hline(data = true_value_df,
               aes(yintercept = true_value),
               linetype = "dashed",
               color = "red") +

    # splpit large plot into small ones
    facet_grid(estimate_type ~ year, scales = "free_y") +
    # set theme
    theme_minimal() +
    theme(
      axis.text.x = element_text(size = 10),
      legend.position = "bottom",
      legend.title = element_text(size = 10),
      legend.text = element_text(size = 8),
      strip.text = element_text(size = 12)
    ) +

    # set labels
    labs(
      x = "Estimation Method",
      y = "Estimate with 95% CI",
      color = "Stratification Variable",
      title = "Comparison of Different Estimation Methods Across 2016, 2020, 2022",
      subtitle = "Red dashed lines indicate true population values"
    )
}


############################################################################ Function 4.2
# Create performance metrics summary table using kable package
create_performance_table <- function(yearly_results) {
  library(knitr)
  library(kableExtra)
  library(dplyr)

  # Initialize empty data frame for storing results
  performance_summary <- data.frame()

  # Process each year's results
  for(year in names(yearly_results)) {
    # Get performance metrics for current year
    perf <- yearly_results[[year]]$performance
    true_values <- yearly_results[[year]]$true_values
    # Process each stratification variable
    for(strata_var in names(perf)) {
      # Mean estimation methods
      for(method in c("srs_vanilla", "srs_ratio", "str_prop")) {
        mean_metrics <- perf[[strata_var]]$mean_estimates[[method]]

        # Create row for mean estimation
        mean_row <- data.frame(
          Year = year,
          Target = "Mean Income",
          Method = case_when(
            method == "srs_vanilla" ~ "SRS-Vanilla",
            method == "srs_ratio" ~ "SRS-Ratio",
            method == "str_prop" ~ paste("Stratified by", strata_var)
          ),
          Avg_Bias = mean_metrics$bias,
          Avg_SE = mean_metrics$avg_ci_width/(2*1.96),
          RMSE = mean_metrics$rmse,
          Coverage_Rate = mean_metrics$coverage,
          Relative_Efficiency = mean_metrics$rmse /
            perf[[strata_var]]$mean_estimates$srs_vanilla$rmse
        )
        # Only add if this combination doesn't exist yet
        if(!any(duplicated(rbind(performance_summary, mean_row)))) {
          performance_summary <- rbind(performance_summary, mean_row)
        }
      }

      # Proportion estimation methods
      for(method in c("srs_vanilla", "str_prop")) {
        prop_metrics <- perf[[strata_var]]$prop_estimates[[method]]

        # Create row for proportion estimation
        prop_row <- data.frame(
          Year = year,
          Target = "Proportion Above Threshold",
          Method = case_when(
            method == "srs_vanilla" ~ "SRS-Vanilla",
            method == "str_prop" ~ paste("Stratified by", strata_var)
          ),
          Avg_Bias = prop_metrics$bias,
          Avg_SE = prop_metrics$avg_ci_width/(2*1.96),
          RMSE = prop_metrics$rmse,
          Coverage_Rate = prop_metrics$coverage,
          Relative_Efficiency = prop_metrics$rmse /
            perf[[strata_var]]$prop_estimates$srs_vanilla$rmse
        )
        # Only add if this combination doesn't exist yet
        if(!any(duplicated(rbind(performance_summary, prop_row)))) {
          performance_summary <- rbind(performance_summary, prop_row)
        }
      }
    }
  }

  # Format the data
  performance_summary <- performance_summary %>%
    # Group by different targets
    group_by(Target) %>%
    mutate(
      # Process Mean Income and Proportion separately
      across(c(Avg_Bias, Avg_SE, RMSE),
            ~case_when(
              Target == "Mean Income" ~ sprintf("%.0f", round(., 0)),  # Show integers only
              Target == "Proportion Above Threshold" ~ sprintf("%.4f", round(., 4))  # Show 4 decimal p
            )),
      # Format Relative Efficiency to 3 decimal places
      Relative_Efficiency = sprintf("%.3f", round(Relative_Efficiency, 3)),
      # Format Coverage Rate as percentage
      Coverage_Rate = sprintf("%.1f%%", Coverage_Rate * 100)
    ) %>%
    ungroup() %>%
    # Sort the data
    arrange(Year, Target, Method, .by_group = TRUE)

  # Create kable table with hierarchical structure
  kable(performance_summary,
        format = "html",
        align = rep('c', ncol(performance_summary)),  # Center align all columns
        caption = "<b>Sampling Methods Performance Comparison</b>") %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "bordered"),
                  full_width = FALSE,
                  position = "left",
                  font_size = 12) %>%
    add_header_above(c(" " = 3,
                      "Performance Metrics" = 5)) %>%
    collapse_rows(columns = 1:2,  # Merge duplicate values in Year and Target columns
                 valign = "middle") %>%  # Vertical center alignment
    row_spec(0, bold = TRUE) %>%  # Bold header
    footnote(general = "Note: Relative Efficiency is calculated relative to Simple Random Sampling",
            general_title = "",
            footnote_as_chunk = TRUE)
}


####################################################################################
#--------------------------------Other Sampling-----------------------------------#
####################################################################################


############################################################################ Function 5.1
# Stratified Sampling with Equal Allocation
str_equal_sampling <- function(data, tarv_1, tarv_2, thres, n, strata_var) {
  # Input validation
  if (!all(c(tarv_1, tarv_2, strata_var) %in% names(data))) {
    stop("Required variables not found in dataset")
```

```
}

# Calculate stratum sizes
N <- nrow(data)
N_hs <- table(data[[strata_var]])
strata_names <- names(N_hs)

# divide n equally among strata
H <- length(strata_names)  # number of strata
n_hs <- rep(floor(n/H), H)  # equal allocation
# Add remaining samples to maintain total n
remainder <- n - sum(n_hs)
if (remainder > 0) {
  n_hs[1:remainder] <- n_hs[1:remainder] + 1
}

# Initialize results storage
tarv_1.bar.hs <- numeric(length(strata_names))
tarv_2.bar.hs <- numeric(length(strata_names))
tarv_1.var.smp.hs <- numeric(length(strata_names))
tarv_2.var.smp.hs <- numeric(length(strata_names))

# Sample from each stratum
STR.sample <- data.frame()

for (i in seq_along(strata_names)) {
  row.ids <- which(data[[strata_var]] == strata_names[i])
  sample.ids <- sample(row.ids, size = n_hs[i], replace = FALSE)
  STR.sample <- rbind(STR.sample, data[sample.ids,])
}

# Calculate estimate for each stratum
tarv_1.bar.hs <- tapply(STR.sample[[tarv_1]], STR.sample[[strata_var]], mean)
tarv_2.bar.hs <- tapply(STR.sample[[tarv_2]], STR.sample[[strata_var]], calculate_proportion, thresho

# Calculate sample variance for each stratum
tarv_1.var.smp.hs <- tapply(STR.sample[[tarv_1]], STR.sample[[strata_var]], var)
tarv_2.var.smp.hs <- tapply(STR.sample[[tarv_2]], STR.sample[[strata_var]], calculate_proportion_var,

# Calculate overall estimates
tarv_1.str.est <- sum((N_hs/N) * tarv_1.bar.hs)
tarv_2.str.est <- sum((N_hs/N) * tarv_2.bar.hs)

# Calculate standard errors
tarv_1.se <- sqrt(sum((N_hs/N)^2 * (1 - n_hs/N_hs) * tarv_1.var.smp.hs/n_hs))
tarv_2.se <- sqrt(sum((N_hs/N)^2 * (1 - n_hs/N_hs) * tarv_2.var.smp.hs/n_hs))

# Create confidence intervals
tarv_1.CI <- c(tarv_1.str.est - 1.96 * tarv_1.se,
             tarv_1.str.est + 1.96 * tarv_1.se)
tarv_2.CI <- c(tarv_2.str.est - 1.96 * tarv_2.se,
             tarv_2.str.est + 1.96 * tarv_2.se)
```

```r
  return(list(
    tarv_1 = list(
      estimate = tarv_1.str.est,
      se = tarv_1.se,
      ci = tarv_1.CI,
      strata_estimates = tarv_1.bar.hs
    ),
    tarv_2 = list(
      estimate = tarv_2.str.est,
      se = tarv_2.se,
      ci = tarv_2.CI,
      strata_estimates = tarv_2.bar.hs
    )
  ))
}


############################################################################ Function 5.2
# Stratified Sampling with Optimal Allocation
str_opt_sampling <- function(data, tarv_1, tarv_2, thres, n, strata_var,
                            strata_s2_guess = NULL, strata_cost = NULL) {
  # Input validation
  if (!all(c(tarv_1, tarv_2, strata_var) %in% names(data))) {
    stop("Required variables not found in dataset")
  }

  # Calculate stratum sizes and standard deviations
  N <- nrow(data)
  strata_names <- unique(data[[strata_var]])
  N_h <- tapply(data[[tarv_1]], data[[strata_var]], length)



  # Use provided variance guesses or calculate from data
  if (is.null(strata_s2_guess)) {
    S_h <- tapply(data[[tarv_1]], data[[strata_var]], sd)
  } else {
    S_h <- sqrt(strata_s2_guess)
  }

  # Default cost is 1 for all strata if not provided
  if (is.null(strata_cost)) {
    strata_cost <- rep(1, length(N_h))
  }



  # Calculate optimal allocation considering N_h, S_h, and cost
  opt_allocation_factor <- N_h * S_h / sqrt(strata_cost)
  n_hs <- round(n * opt_allocation_factor / sum(opt_allocation_factor))

  # Initialize results storage
  tarv_1.bar.hs <- numeric(length(strata_names))
```

```r
tarv_2.bar.hs <- numeric(length(strata_names))
tarv_1.var.smp.hs <- numeric(length(strata_names))
tarv_2.var.smp.hs <- numeric(length(strata_names))

# Sample from each stratum
STR.sample <- data.frame()

for (i in seq_along(strata_names)) {
  row.ids <- data[data[[strata_var]] == strata_names[i], ]
  sample.ids <- sample(nrow(row.ids), size = n_hs[i], replace = FALSE)
  stratum_sample <- row.ids[sample.ids, ]
  STR.sample <- rbind(STR.sample, stratum_sample)
}


# Calculate estimate for each stratum
tarv_1.bar.hs <- tapply(STR.sample[[tarv_1]], STR.sample[[strata_var]], mean)
tarv_2.bar.hs <- tapply(STR.sample[[tarv_2]], STR.sample[[strata_var]], calculate_proportion, thresho

# Calculate sample variance for each stratum
tarv_1.var.smp.hs <- tapply(STR.sample[[tarv_1]], STR.sample[[strata_var]], var)
tarv_2.var.smp.hs <- tapply(STR.sample[[tarv_2]], STR.sample[[strata_var]], calculate_proportion_var,

# Calculate overall estimates
tarv_1.est <- sum((N_h/N) * tarv_1.bar.hs)
tarv_2.est <- sum((N_h/N) * tarv_2.bar.hs)

# Calculate standard errors
tarv_1.se <- sqrt(sum((N_h/N)^2 * (1 - n_hs/N_h) * tarv_1.var.smp.hs/n_hs))
tarv_2.se <- sqrt(sum((N_h/N)^2 * (1 - n_hs/N_h) * tarv_2.var.smp.hs/n_hs))

# Create confidence intervals
tarv_1.CI <- c(tarv_1.est - 1.96 * tarv_1.se,
             tarv_1.est + 1.96 * tarv_1.se)
tarv_2.CI <- c(tarv_2.est - 1.96 * tarv_2.se,
             tarv_2.est + 1.96 * tarv_2.se)

return(list(
  tarv_1 = list(
    estimate = tarv_1.est,
    se = tarv_1.se,
    ci = tarv_1.CI,
    strata_estimates = tarv_1.bar.hs,
    strata_ns = n_hs
  ),
  tarv_2 = list(
    estimate = tarv_2.est,
    se = tarv_2.se,
    ci = tarv_2.CI,
    strata_estimates = tarv_2.bar.hs,
    strata_ns = n_hs
  )
))
}
```