Comparative Sampling Techniques for Estimating Mobile Device Usage

Elias Khan 46497392
Nubbh Kejriwal 87841698
David Yuan 19548411
Luning Yang 97718894

Contributions:

Elias Khan:
Group Leader, selecting the dataset, identifying the variable to estimate, determining the stratifying variable, and defining the ratio and regression variable. Other responsibilities included preparing the project introduction and conducting data analysis of the continuous variable, including vanilla estimates, ratio estimates, regression analysis, and stratified estimates.

Nubbh Kejriwal:
Binary variable data analysis using R. Estimates, standard errors, confidence intervals using simple random sampling (vanilla, ratio, and regression) and stratified sampling (proportional, and optimal allocation)

David Yuan:
Description of dataset, exploratory analysis and figures, data processing written portion, results and outputs table, minor code fixes, project tidying and formatting

Luning Yang:
Results interpretation, discussion and comparison of different methods, conclusion and potential future improvements and research area

1 Introduction

Our project aims to explore how different statistical methods influence the estimation of a population parameter. We will investigate this in the context of two questions: “How much time does the average person spend on their phone?” and “What proportion of the population spends too much time on their phone?” For each question, we will calculate five estimates. The first three estimates will be based on a simple random sample (SRS): a vanilla estimate, a ratio estimate, and a regression estimate. The final two estimates will be derived from stratified samples: one using proportional allocation and the other using optimal allocation. Our goal is to apply statistical theory and confirm that our estimates perform as expected in theory. To achieve this, we will evaluate each estimate against the true population parameter.

2 Description of Dataset

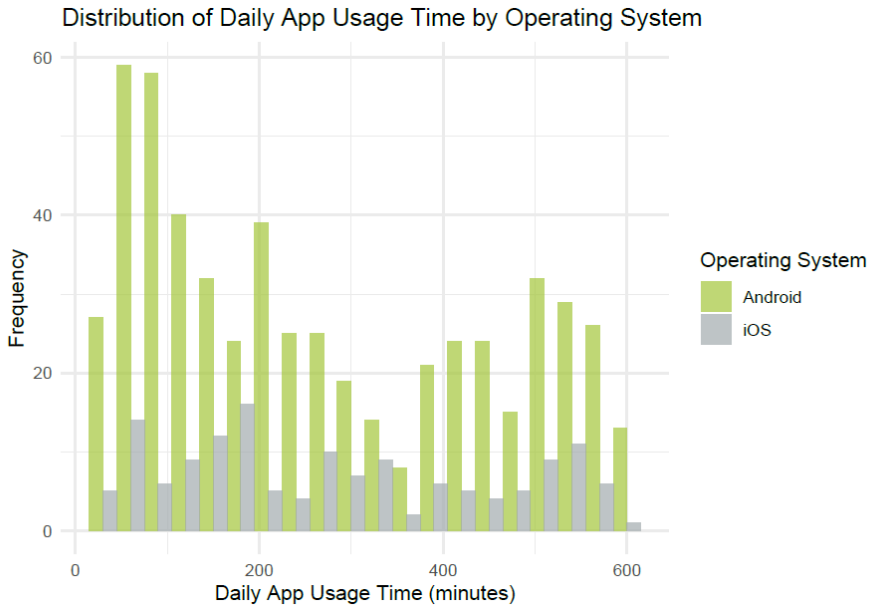


Figure 1, Distribution of Daily App Usage Time by Operating System

We will be using the Mobile Device Usage and User Behavior Dataset created by Vala Khorasani which encompasses 700 unique entries of recent mobile usage data. Key variables include daily app usage time in minutes, data usage, operating system, and demographic details like age and gender. The response variable for this study is the average daily app usage time in minutes, while the auxiliary variable used for regression and ratio estimates later on in the data processing is daily data usage. Stratification will be performed based on the operating system (iOS or Android), as it captures differences in app usage patterns more effectively than other potential stratifiers like gender. For instance, Android users generally show broader usage patterns, but the dataset contains fewer iOS users, making this a suitable stratification variable.

For the purpose of this project, the dataset is treated as the population, allowing us to compare the efficacy of sampling methods by evaluating sample-based estimates against known population

parameters. This treatment ensures a direct assessment of bias, variability, and precision for both Simple Random Sampling (SRS) and Stratified Sampling. The finite population correction (FPC) is incorporated into standard error calculations due to the relatively large sampling fraction (n = 100, N = 700). The binary variable "heavy users" was defined with a threshold of 240 minutes of daily usage, aligning with global average smartphone usage trends (3 hours 15 minutes) and setting a meaningful benchmark for analysis, the justification of the choice will be explored in the Data Processing section. By structuring the dataset in this manner, we ensure that all sampling and estimation techniques are robust and applicable to real-world population studies.

In Figure 1, we see that Android users exhibit broader and generally higher usage times, while iOS users tend to cluster around lower app usage values. This difference emphasizes the value of stratifying by operating system, as it captures this inherent variability. Although Android users dominate the dataset (79.1%), the smaller iOS subgroup's distinct characteristics justify proportional allocation during stratified sampling. The chosen stratification variable effectively balances representation and captures the between-stratum differences necessary to improve the precision of our estimates.

3 Data Processing

We aim to estimate two key parameters from the population of mobile device users: the average daily app usage time (a continuous variable) and the proportion of heavy users (a binary variable). These parameters were chosen for their practical relevance in understanding user behavior and deriving actionable insights. The continuous variable provides a measure of typical user engagement, while the binary variable identifies a specific user segment ("heavy users") for targeted analysis. To achieve this, we employed two fundamental sampling approaches, Simple Random Sampling (SRS) and Stratified Sampling, each applied to both the continuous and binary variables. This dual approach allowed us to compare the precision, reliability, and efficiency of the two sampling strategies across different scenarios.

Simple Random Sampling (SRS) was selected as the baseline method due to its simplicity and unbiased nature. In SRS, every individual in the population has an equal probability of being included in the sample, ensuring a representative subset of the population. For the continuous variable, the mean daily app usage time was estimated using SRS, followed by calculations of its standard error and confidence intervals. Similarly, SRS was employed for the binary variable to estimate the proportion of heavy users. Heavy users were defined as individuals with daily app usage exceeding 240 minutes. This threshold was chosen based on a study by Josh Howarth that reported the average global screen time in 2024 to be 3 hours and 15 minutes; thus, 4 hours (or 240 minutes) was deemed a logical benchmark for heavy usage. The simplicity of SRS makes it ideal for initial estimates, but its precision is often limited in heterogeneous populations, motivating the exploration of more advanced techniques.

Stratified Sampling was employed to enhance the precision of our estimates by reducing variability. This method divides the population into distinct, non-overlapping subgroups (strata) based on a stratification variable that captures population heterogeneity. For this study, the operating system (iOS or Android) was chosen as the stratification variable, as

preliminary analysis indicated that app usage patterns differed significantly between these two user groups. Stratified sampling was applied to both the continuous and binary variables using proportional allocation, ensuring that the sample sizes within each stratum reflected the population structure. Additionally, we explored optimal allocation for the continuous variable, which assigns larger sample sizes to strata with higher internal variability to further minimize variability in the overall estimate.

For the continuous variable, the sample means were computed for each stratum, and a weighted average was used to estimate the population mean. Stratified sampling also allowed for the calculation of stratified standard errors and confidence intervals, offering a more precise measure of reliability compared to SRS. Similarly, for the binary variable, the stratified proportion was calculated as a weighted sum of the proportions in each stratum. The standard error and confidence intervals for the stratified estimate were computed to evaluate the precision of this approach.

Throughout the analysis, the sample size for all methods was set to 100 to ensure uniformity across comparisons. Given the small population size $N = 700$ relative to the sample size $n = 100$ (n/N is relatively large $\approx 14.3\%$), we applied the finite population correction (FPC) to adjust for reduced variability due to sampling without replacement. This will further strengthen the reliability of our estimates and comparison between SRS and stratified sampling across both variables. For all the code used to achieve data processing steps please check the Appendix.

3.1 Simple Random Sampling for Continuous Variable

In this section, our primary goal was to estimate the average daily app usage time for the population of mobile users. This variable represents a continuous measurement. SRS ensures that every individual in the population has an equal probability of being selected, making it an effective tool for generating representative samples, particularly when no prior stratification information is available. For the given dataset, the total population size N is 700, and we selected a sample size n of 100, without replacement. This choice of sample size balances computational feasibility and statistical reliability, ensuring that the sampling fraction $\frac{n}{N} = \frac{100}{700} = 0.143$ is not too small, while also remaining manageable for analysis.

The sample mean $\bar{y}$, an unbiased estimator of the population mean, was computed using the following formula:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

where y_i represents the daily app usage time for the i-th sampled individual. For our SRS sample, the mean daily app usage time was calculated as $\bar{y} = 274.03$ minutes. The variability within the sample was quantified using the sample variance s^2:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} (y_i - \bar{y})^2$$

This yielded a variance of $s^2 = 562.53$. To account for the finite population correction (FPC), which adjusts for the reduced variability when sampling without replacement, the standard error (SE) was computed as:

$$SE = \sqrt{\frac{(1 - n/N)s^2}{n}}$$

Substituting the values, the standard error for the sample mean was determined to be $SE = 15.11$ minutes. Using this standard error, we constructed a 95% confidence interval (CI) for the population mean:

$$CI = \bar{y} \pm z_{\alpha/2} \cdot SE$$

where $z_{\alpha/2} = 1.96$ for a 95% confidence level. The resulting confidence interval was:

$$CI = [244.41, 303.65] \text{ minutes}$$

In addition to the vanilla estimate, ratio and regression estimators were computed. The ratio estimate ($\hat{y}_{ratio} = \bar{y}/\bar{x}_{sample} \cdot \bar{x}_{population} = 278.92$) had $SE = 5.37$ and a narrower $CI = [268.41,289.44]$

The regression estimate ($\hat{y}_{reg} = b_0 + b_1 \cdot \bar{x}_{population} = 278.46$) had $SE = 5.15$ and the tightest $CI = [268.36,288.56]$.

The vanilla estimate provided the closest approximation to the population mean but had the largest confidence interval, suggesting lower precision. Regression and ratio estimates yielded narrower intervals, making them more reliable under the SRS framework. Among these, the regression estimate was the most precise, containing the true population mean within a smaller range.

3.2 Stratified Sampling for Continuous Variable

After SRS we implemented stratified sampling in attempt to enhance the precision of our estimate for average daily app usage time. Stratification divides the population into distinct subgroups (strata) that are internally homogeneous but differ significantly from one another. For this study, the operating system (iOS or Android) was chosen as the stratification variable because of its observed association with app usage patterns. Specifically, Android users had higher average usage times than iOS users in the dataset. Stratified sampling reduces variability by eliminating between-stratum differences, thus improving the reliability of estimates.

The population was divided into two strata: $N_{iOS} = 146$ and $N_{Android} = 554$. Proportional allocation was used to determine the sample sizes for each stratum, calculated as:

$$n_h = n \cdot \frac{N_h}{N}$$

where n_h is the sample size for stratum h, N_h is the population size for stratum h, and N is

the total population size. For $n = 100$, this yielded $n_{iOS} = 21$ and $n_{Android} = 79$. For optimal allocation, larger samples were assigned to strata with greater internal variability ($s_{Android} > s_{iOS}$), resulting in $n_{iOS\ (optimal)} = 16$ and $n_{Android\ (optimal)} = 84$. The stratified mean $\bar{y}_{str}$ was computed as a weighted average of the stratum means:

$$\bar{y}_{str} = \sum_{h=1}^{H} \frac{N_h}{N} \bar{y}_h$$

where $H = 2$ is the number of strata and $\bar{y}_h$ is the sample mean for stratum h. From the sampled data:

$$\bar{y}_{iOS} = 260.34,\ \bar{y}_{Android} = 276.24,\ \bar{y}_{str} = 270.37$$

The standard error (SE) of the stratified mean was calculated as:

$$SE = \sqrt{\sum_{h=1}^{H} \frac{N_h^2}{N^2} \cdot \frac{s_h^2}{n_h} \left(1 - \frac{n_h}{N_h}\right)}$$

where s_h^2 is the sample variance within stratum h. Substituting the values, the standard error was $SE = 16.98$ minutes

For optimal allocation, the sample sizes were adjusted as:

$$n_h = n \cdot \frac{N_h \cdot \sqrt{\hat{p}_h(1 - \hat{p}_h)}}{\sum_{h=1}^{H} N_k \cdot \sqrt{\hat{p}_h(1 - \hat{p}_h)}}$$

Resulting in $n_{iOS\ (optimal)} = 16$, $n_{Android\ (optimal)} = 84$. The optimal allocation estimate was $\bar{y}_{str\ (optimal)} = 265.14$ with $SE = 15.61$ minutes.

Using the standard errors, the 95% confidence interval was:

$$CI\ (proportional\ allocation) = [236.47, 304.27] \text{ minutes}$$
$$CI\ (optimal\ allocation) = [234.53, 295.74] \text{ minutes}$$

While the optimal allocation provided the closest estimate to the true population mean with the narrowest confidence interval, proportional allocation still demonstrated substantial precision gains over SRS. These results highlight stratified sampling's ability to refine estimates by reducing between-stratum variability.

3.3 Simple Random Sampling for Binary Variable

For the binary variable analysis, we estimated the proportion of "heavy users"(p), defined as individuals with daily app usage exceeding 240 minutes. SRS was employed to calculate the sample proportion, its standard error, and confidence interval. From the SRS sample of size $n = 100$, the sample proportion $\hat{p}$ was:

$$\hat{p} = \frac{\text{Number of Heavy Users in Sample}}{n}$$

In our sample, 55 users were classified as heavy users, resulting in $\hat{p} = 0.55$

The standard error (SE) was calculated as:

$$SE = \sqrt{\frac{(1 - n/N)\hat{p}(1 - \hat{p})}{n}}$$

where the FPC adjustment accounts for the large sampling fraction. Substituting the values, the standard error was $SE = 0.0461$

The 95% confidence interval is constructed as:

$$CI = \hat{p} \pm z_{\alpha/2} \cdot SE = [0.459, 0.640]$$

This interval provides a range where the true proportion of heavy users is likely to lie with 95% certainty.

In addition to the vanilla estimate, ratio and regression estimators were computed. The ratio estimate ($\hat{p}_{ratio} = \hat{p}/\bar{x}_{sample} \cdot \bar{x}_{population} = 0.560$) with $SE = 0.0003$ and a very narrow $CI = [0.5592, 0.5605]$

The regression estimate ($\hat{p}_{reg} = b_0 + b_1 \cdot (\bar{x}_{population} - \bar{x}_{sample}) = 0.561$) had $SE = 0.0315$ and $CI = [0.499, 0.623]$.

The vanilla SRS method provided the broadest confidence interval while the ratio and regression estimates offered narrower intervals, though still encompassing the true population proportion. These refinements highlight the importance of leveraging auxiliary data to improve estimation precision.

3.4 Stratified Sampling for Binary Variable

We lastly apply stratified sampling, with the operating system serving as the stratification variable, to see if it refines the estimate of the proportion of heavy users. Using proportional allocation, the sample sizes were:

$$n_h = n \cdot \frac{N_h}{N}$$

$$n_{iOS} = 21, n_{Android} = 79$$

The stratified proportion $\hat{p}_{str}$ was computed as:

$$\hat{p}_{str} = \sum_{h=1}^{H} \frac{N_h}{N} \hat{p}_h = 0.460$$

where $\hat{p}_h$ is the sample proportion of heavy users in stratum h. The standard error was calculated as:

$$SE = \sqrt{\sum_{h=1}^{H} \frac{N_h^2}{N^2} \cdot \frac{\hat{p}_h(1-\hat{p}_h)}{n_h}} = 0.0499$$

The 95% confidence interval was $CI = [0.362, 0.558]$

For optimal allocation, sample sizes were adjusted as:

$$n_h = n \cdot \frac{N_h \cdot \sqrt{\hat{p}_h(1-\hat{p}_h)}}{\sum_{h=1}^{H} N_k \cdot \sqrt{\hat{p}_h(1-\hat{p}_h)}}$$

Resulting in $n_{iOS\ (optimal)} = 16$, $n_{Android\ (optimal)} = 84$. Using the sample sizes, the stratified sample estimate was refined to $\hat{p}_{str\ (optimal)} = 0.520$ with $SE_{str\ (optimal)} = 0.0499$ and $CI = [0.422, 0.618]$

4 Results and Interpretation

Sampling Method	Estimate (Continuous)	Standard Error (Continuous)	Confidence Interval (Continuous)	Estimate (Binary)	Standard Error (Binary)	Confidence Interval (Binary)
True Population	271.13			0.4743		
SRS (Vanilla)	274.03	15.11	[244.41, 303.65]	0.55	0.0461	[0.459, 0.640]
SRS (Ratio Estimate)	278.92	5.37	[268.41, 289.44]	0.56	0.00032	[0.559, 0.560]
SRS (Regression Estimate)	278.46	5.15	[268.36, 288.56]	0.561	0.0315	[0.499, 0.623]
Stratified Sampling (Proportional Allocation)	270.37	16.98	[236.47, 304.27]	0.46	0.0499	[0.362, 0.558]
Stratified Sampling (Optimal Allocation)	265.14	15.61	[234.53, 295.74]	0.52	0.0499	[0.422, 0.618]

Figure 2, Table of Results and Outputs

Continuous Variable

As we can see from the table above, the Vanilla SRS estimate had the largest standard error (15.11)

and the widest confidence interval among the SRS methods, highlighting less precision. Both the Ratio and Regression estimates have narrower confidence intervals, indicating smaller SE and more accuracy.

Stratified Sampling with proportional allocation yielded a mean close to the true value but with a higher standard error. In contrast, optimal allocation reduced the standard error, demonstrating greater efficiency in stratified design.

SRS vs. Stratified Sampling
Comparing these two sampling methods, we can conclude that SRS have a much smaller standard error than that of the Stratified Sampling, though Stratified Sampling with proportional allocation has a closer mean to the true mean, the SRS have a narrower CI for the true mean. Thus, in this scenario, SRS works better.

Binary Variable

The Vanilla SRS proportion estimate has the largest variability. Although the Ratio and Regression estimates produced small SEs and narrower CIs, they slightly overestimated the true proportion. Only the Vanilla SRS's CI includes the true mean.

Stratified Sampling provided different estimates but the same SEs, with Proportional allocation offering a more precise estimate for the true mean.

SRS vs. Stratified Sampling
The SRS estimates for Binary Variable are less precise than the Stratified sampling estimates. Though their standard errors are close to each other, the CIs' of stratified sampling seem better to catch the true mean. Therefore, we would choose stratified sampling for Binary Variable.

5 Discussion

Our Study compared the SRS and Stratified Sampling in estimating continuous and binary variables by using the mobile device usage dataset. Applying statistical coding and analysis, we can see:

Continuous Variable Analysis:

- The Vanilla SRS estimate was unbiased but less efficient than the Ratio and Regression estimates. This inefficiency includes larger confidence intervals and SEs.
- Stratified Sampling showed the advantage of stratification by reducing variability between strata. The choice of the operating system (iOS vs. Android) as a stratifying variable is effective, as it accounted for app usage pattern differences.

Binary Variable Analysis:

- SRS produced a reasonable estimate of the heavy users' proportion but has a higher variability. The Ratio and Regression estimates, although have smaller SEs, were less aligned with the true value, possibly due to over-adjustment.
- Stratified Sampling showed its efficiency by refining these estimates. However, the variability remained noticeable, and proportional allocation was more reliable for its confidence interval.

General Observations:

- **SRS**: Suitable for straightforward analysis but less reliable in heterogeneous populations.
- **Stratified Sampling**: Superior for populations with known subgroups, as it minimizes standard error and provides more stable estimates, particularly with optimal allocation.

Our results affirm the theoretical expectations: stratified sampling generally enhances estimate precision, especially in heterogeneous datasets. Yet, the method's effectiveness hinges on selecting appropriate stratification variables.

6 Conclusions

Our study comprehensively assessed the performance of different sampling techniques (SRS vs. Stratified Sampling) to estimate mobile device usage parameters. The results underscore the importance of method selection in sampling:

- **SRS**: While simple and unbiased, SRS will probably have high variability, limiting its utility for precision-demanding studies.
- **Stratified Sampling**: Consistent improvements in estimate precision, as it reduces overall estimate variability. This method is highly recommended when population subgroups are well-defined and accessible.

Future Improvements:

Based on the strengths and limitations observed in our current analysis, several improvements could be considered for future studies to enhance the accuracy and reliability of our sampling techniques and overall findings:

- Exploring other stratification variables, such as age group, gender, or app usage categories, could further enhance the precision of the estimates. Selecting a variable that captures additional aspects of usage behavior might lead to better representation and reduced variability.
- Increasing sample size or utilizing more sophisticated sampling designs (e.g., cluster sampling) may also enhance estimate reliability.
- Additional simulations could confirm the reliability of our findings under different population structures.

7 References

Howarth, J. (2024a, June 4). *Time spent using smartphones (2024 statistics)*. Exploding Topics. https://explodingtopics.com/blog/smartphone-usage-stats

Khorasani, V. (2024, September 28). *Mobile device usage and User Behavior Dataset*. Kaggle. https://www.kaggle.com/datasets/valakhorasani/mobile-device-usage-and-user-behavior-dataset

8 Appendix

All the coding done for this report are attached

```r
library(sampling)
library(ggplot2)

### Dataset Exploratory (David)

mobile = read.csv(file= "/Users/david/Desktop/STAT 344/user_behavior_dataset.csv")
head(mobile)
```

```
##   User.ID   Device.Model Operating.System App.Usage.Time..min.day.
## 1       1 Google Pixel 5          Android                      393
## 2       2      OnePlus 9          Android                      268
## 3       3   Xiaomi Mi 11          Android                      154
## 4       4 Google Pixel 5          Android                      239
## 5       5      iPhone 12              iOS                      187
## 6       6 Google Pixel 5          Android                       99
##   Screen.On.Time..hours.day. Battery.Drain..mAh.day. Number.of.Apps.Installed
## 1                        6.4                    1872                       67
## 2                        4.7                    1331                       42
## 3                        4.0                     761                       32
## 4                        4.8                    1676                       56
## 5                        4.3                    1367                       58
## 6                        2.0                     940                       35
##   Data.Usage..MB.day. Age Gender User.Behavior.Class
## 1                1122  40   Male                   4
## 2                 944  47 Female                   3
## 3                 322  42   Male                   2
## 4                 871  20   Male                   3
## 5                 988  31 Female                   3
## 6                 564  31   Male                   2
```

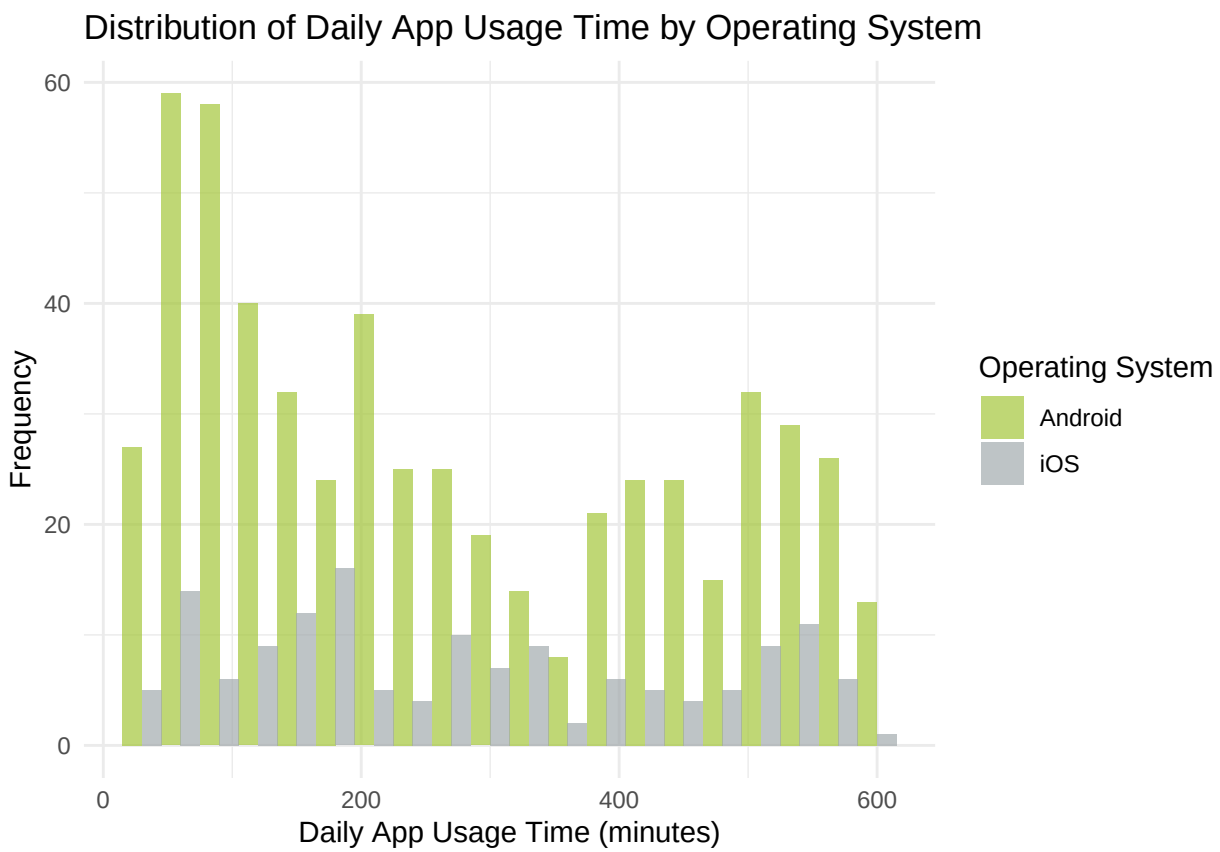
```r
summary_stats <- summary(mobile)
print(summary_stats)
```

```
##     User.ID      Device.Model       Operating.System   App.Usage.Time..min.day.
##  Min.   :  1.0   Length:700         Length:700         Min.   : 30.0
##  1st Qu.:175.8   Class :character   Class :character   1st Qu.:113.2
##  Median :350.5   Mode  :character   Mode  :character   Median :227.5
##  Mean   :350.5                                         Mean   :271.1
##  3rd Qu.:525.2                                         3rd Qu.:434.2
##  Max.   :700.0                                         Max.   :598.0
##  Screen.On.Time..hours.day. Battery.Drain..mAh.day. Number.of.Apps.Installed
##  Min.   : 1.000             Min.   : 302.0          Min.   :10.00
##  1st Qu.: 2.500             1st Qu.: 722.2          1st Qu.:26.00
##  Median : 4.900             Median :1502.5          Median :49.00
##  Mean   : 5.273             Mean   :1525.2          Mean   :50.68
##  3rd Qu.: 7.400             3rd Qu.:2229.5          3rd Qu.:74.00
##  Max.   :12.000             Max.   :2993.0          Max.   :99.00
##  Data.Usage..MB.day.      Age           Gender          User.Behavior.Class
##  Min.   : 102.0      Min.   :18.00   Length:700         Min.   :1.00
##  1st Qu.: 373.0      1st Qu.:28.00   Class :character   1st Qu.:2.00
##  Median : 823.5      Median :38.00   Mode  :character   Median :3.00
##  Mean   : 929.7      Mean   :38.48                      Mean   :2.99
##  3rd Qu.:1341.0      3rd Qu.:49.00                      3rd Qu.:4.00
##  Max.   :2497.0      Max.   :59.00                      Max.   :5.00
```

```r
# Checking missing values
missing_values <- colSums(is.na(mobile))
print(missing_values)
```

```
##                     User.ID                Device.Model
##                           0                           0
##            Operating.System    App.Usage.Time..min.day.
##                           0                           0
## Screen.On.Time..hours.day.     Battery.Drain..mAh.day.
##                           0                           0
##    Number.of.Apps.Installed         Data.Usage..MB.day.
##                           0                           0
##                         Age                      Gender
##                           0                           0
##         User.Behavior.Class
##                           0
```

```r
# Distribution of App Usage Time color-coded by Operating System
ggplot(mobile, aes(x = App.Usage.Time..min.day., fill = Operating.System)) +
  geom_histogram(binwidth = 30, alpha = 0.7, position = "dodge") +
  scale_fill_manual(values = c("#a4c639", "#A2AAAD")) +
  labs( title = "Distribution of Daily App Usage Time by Operating System",
    x = "Daily App Usage Time (minutes)", y = "Frequency",
    fill = "Operating System") +
  theme_minimal()
```




```r
true_mean_app_usage <- mean(mobile$App.Usage.Time..min.day., na.rm = TRUE)
cat("Population Mean (App Usage Time):", true_mean_app_usage, "\n")
```

```
## Population Mean (App Usage Time): 271.1286
```

```
true_proportion_heavy_users <- mean(ifelse(mobile$App.Usage.Time..min.day.
                                           > 240, 1, 0), na.rm = TRUE)
cat("Population Proportion (Heavy Users):", true_proportion_heavy_users, "\n")
```

```
## Population Proportion (Heavy Users): 0.4742857
```

```
#preliminary analysis (Elias)
N=nrow(mobile)
N
```

```
## [1] 700
```

```
y_pop=mobile$App.Usage.Time..min.day.
x_pop=mobile$Data.Usage..MB.day.

y_pop_mean= mean(y_pop)
x_pop_mean= mean(x_pop)

stratafier= mobile$Operating.System

android_data = subset(mobile, Operating.System == "Android")

ios_data = subset(mobile,Operating.System == "iOS")



#### SRS and stratified sampling for continuous variable (Elias)



#part 1: SRS Sample estimation
n <- 100
set.seed(1)
SRS = mobile[sample(nrow(mobile), n, replace = FALSE), ]
head(SRS)
```

```
##     User.ID       Device.Model Operating.System App.Usage.Time..min.day.
## 679     679     Google Pixel 5          Android                       298
## 129     129       Xiaomi Mi 11          Android                       125
## 509     509     Google Pixel 5          Android                       267
## 471     471 Samsung Galaxy S21          Android                       248
## 299     299          iPhone 12              iOS                       264
## 270     270 Samsung Galaxy S21          Android                       381
##     Screen.On.Time..hours.day. Battery.Drain..mAh.day. Number.of.Apps.Installed
## 679                        4.6                    1525                       59
## 129                        2.5                     678                       34
## 509                        5.9                    1740                       45
## 471                        4.6                    1396                       52
## 299                        5.2                    1641                       44
## 270                        6.6                    2160                       69
##     Data.Usage..MB.day. Age Gender User.Behavior.Class
## 679                 814  36 Female                   3
## 129                 465  31   Male                   2
```

```
## 509                 791  22   Male                  3
## 471                 883  40   Male                  3
## 299                 778  44   Male                  3
## 270                1450  20   Male                  4
```

```
y_SRS = SRS$App.Usage.Time..min.day.

#vanilla estimate of y= 274.03, se(vanilla) = 15.11
vanilla = mean(y_SRS)
vanilla
```

```
## [1] 274.03
```

```
sample_residuals = y_SRS-vanilla
sample_variance = (1/(n-1))*sum(sample_residuals^2)
vanilla_se = sqrt((1-n/N)*sample_variance/n)
vanilla_se
```

```
## [1] 15.11244
```

```
#95% confidence interval for vanilla estimate = [244.41,303.65]
critical_value = 1.96
ci_vanilla = c(vanilla-critical_value*vanilla_se,
               vanilla+critical_value*vanilla_se)

#ratio estimate for y = 278.92, se(ratio)= 5.37
x_SRS= SRS$Data.Usage..MB.day.
SRS_x_bar = mean(x_SRS)

ratio_est= (vanilla/SRS_x_bar)*x_pop_mean
ratio_est
```

```
## [1] 278.9239
```

```
ratio_residuals= y_SRS-(vanilla/SRS_x_bar)*x_SRS
ratio_variance = (1/(n-1))*sum(ratio_residuals^2)
ratio_se = sqrt((1-n/N)*ratio_variance/n)
ratio_se
```

```
## [1] 5.365458
```

```
#95% confidence interval for ratio estimate = [268.41,289.44]
ci_ratio = c(ratio_est-critical_value*ratio_se,
             ratio_est+critical_value*ratio_se)
ci_ratio
```

```
## [1] 268.4076 289.4402
```

```
#regression estimate for y = 278.46, se(reg) = 5.15
reg=lm(y_SRS~x_SRS)
b = coef(reg)

reg_est= b[1]+b[2]*x_pop_mean
reg_est
```

```
## (Intercept)
##    278.4595
```

```r
reg_residuals= y_SRS-(b[1]+b[2]*x_SRS)
reg_variance = (1/(n-1))*sum(reg_residuals^2)
reg_se = sqrt((1-n/N)*reg_variance/n)
reg_se
```

```
## [1] 5.154617
```

```r
#95% confidence interval for regression estimate = [268.36,288.56]
ci_reg = c(reg_est-critical_value*reg_se,reg_est+critical_value*reg_se)
ci_reg
```

```
## (Intercept) (Intercept)
##    268.3565    288.5626
```

```r
#Though the vanilla estimate was the closest estimate to the actual value of
#y_bar_pop, it had the largest 95% confidence interval of [244.41,303.65],
#whereas the regression estimate had a confidence interval of [268.36,288.56],
#which is much smaller and still contains the true population value inside of it,
#this lead me to go with the regression estimate as the best estimate for y_bar_pop
#from a simple random sample.




#part 2a: stratified sampling with proportional allocation
N_android = nrow(android_data)
N_android
```

```
## [1] 554
```

```r
N_ios= nrow(ios_data)
N_ios
```

```
## [1] 146
```

```r
#calculate Nh/N
android_pop_prop= N_android/N
ios_pop_prop = N_ios/N

#calculate proportional sample sizes
n_ios_prop= round(ios_pop_prop*n) #21
n_android_prop= round(android_pop_prop*n) #79

#take random samples of each strata from pre-defined sample sizes
set.seed(1)
SRS_ios = ios_data[sample(nrow(ios_data), n_ios_prop, replace = FALSE), ]
set.seed(1)
SRS_android = android_data[sample(nrow(android_data), n_android_prop,
                                  replace = FALSE), ]


#calculate mean and standard error of strata samples
y_bar_ios= mean(SRS_ios$App.Usage.Time..min.day.)
y_bar_android= mean(SRS_android$App.Usage.Time..min.day.)
```

```r
s_ios_squared= var(SRS_ios$App.Usage.Time..min.day.)
s_android_squared= var(SRS_android$App.Usage.Time..min.day.)

se_ios= sqrt((1-n_ios_prop/N_ios)*(s_ios_squared/n_ios_prop))
se_ios
```

```
## [1] 27.624
```

```r
se_android= sqrt((1-n_android_prop/N_android)*(s_android_squared/n_android_prop))
se_android
```

```
## [1] 19.46332
```

```r
#calculate stratified estimate of y_bar = 275.28
y_str = ios_pop_prop*y_bar_ios + android_pop_prop*y_bar_android
y_str
```

```
## [1] 270.3694
```

```r
#calculate standard error of stratified estimate = 16.98
se_y_str = sqrt(((ios_pop_prop^2)*se_ios^2)+((android_pop_prop^2)*se_android^2))




#part 2b: stratified sampling with optimal allocation (assume costs are the same)

# Calculate optimal sample sizes
s_ios = sqrt(s_ios_squared)
s_android = sqrt(s_android_squared)

n_ios_opt = round((N_ios * s_ios) / (N_ios * s_ios + N_android * s_android)
                  * n) # 16
n_ios_opt
```

```
## [1] 16
```

```r
n_android_opt = round((N_android * s_android) / (N_ios * s_ios + N_android
                                                 * s_android) * n) # 84
n_android_opt
```

```
## [1] 84
```

```r
# Take random samples for optimal allocation
set.seed(1)
SRS_ios_opt = ios_data[sample(nrow(ios_data), n_ios_opt, replace = FALSE), ]
set.seed(1)
SRS_android_opt = android_data[sample(nrow(android_data), n_android_opt,
                                      replace = FALSE), ]

# Calculate mean and standard error of strata samples
y_bar_ios_opt = mean(SRS_ios_opt$App.Usage.Time..min.day.)
y_bar_android_opt = mean(SRS_android_opt$App.Usage.Time..min.day.)

s_ios_opt_squared = var(SRS_ios_opt$App.Usage.Time..min.day.)
s_android_opt_squared = var(SRS_android_opt$App.Usage.Time..min.day.)
```

```r
se_ios_opt = sqrt((1 - n_ios_opt / N_ios) * (s_ios_opt_squared / n_ios_opt))
se_ios_opt
```

```
## [1] 24.85296
```

```r
se_android_opt = sqrt((1 - n_android_opt / N_android) * (s_android_opt_squared
                                                         / n_android_opt))
se_android_opt
```

```
## [1] 18.61169
```

```r
# Calculate stratified estimate for optimal allocation = 265.14
y_str_opt = (N_ios / N) * y_bar_ios_opt + (N_android / N) * y_bar_android_opt
y_str_opt
```

```
## [1] 265.138
```

```r
# Calculate se(opt alloc strat est) = 15.61
se_y_str_opt = sqrt(((N_ios / N)^2) * se_ios_opt^2 + ((N_android / N)^2) *
                      se_android_opt^2)
se_y_str_opt
```

```
## [1] 15.6153
```

```r
# 95% CI for stratified estimate using optimal allocation = [234.53,295.744]
ci_str_opt = c(y_str_opt - critical_value * se_y_str_opt, y_str_opt +
                 critical_value * se_y_str_opt)

# It is hard to say which metric is better across the board, the optimal
# allocation estimate is the closest to the actual population mean, but the
# optimal allocation estimate has a smaller confidence interval that does
# contain the mean.


# Print results
list(
  "Vanilla Estimate" = list(mean = vanilla, CI = ci_vanilla),
  "Ratio Estimate" = list(mean = ratio_est, CI = ci_ratio),
  "Regression Estimate" = list(mean = reg_est, CI = ci_reg),
  "Stratified Proportional Allocation" = list(mean = y_str, SE = se_y_str),
  "Stratified Optimal Allocation" = list(mean = y_str_opt, SE = se_y_str_opt,
                                         CI = ci_str_opt)
)
```

```
## $`Vanilla Estimate`
## $`Vanilla Estimate`$mean
## [1] 274.03
##
## $`Vanilla Estimate`$CI
## [1] 244.4096 303.6504
##
##
## $`Ratio Estimate`
## $`Ratio Estimate`$mean
## [1] 278.9239
##
```

```
## $`Ratio Estimate`$CI
## [1] 268.4076 289.4402
##
##
## $`Regression Estimate`
## $`Regression Estimate`$mean
## (Intercept)
##    278.4595
##
## $`Regression Estimate`$CI
## (Intercept) (Intercept)
##    268.3565    288.5626
##
##
## $`Stratified Proportional Allocation`
## $`Stratified Proportional Allocation`$mean
## [1] 270.3694
##
## $`Stratified Proportional Allocation`$SE
## [1] 16.44608
##
##
## $`Stratified Optimal Allocation`
## $`Stratified Optimal Allocation`$mean
## [1] 265.138
##
## $`Stratified Optimal Allocation`$SE
## [1] 15.6153
##
## $`Stratified Optimal Allocation`$CI
## [1] 234.532 295.744
```

```
#### SRS and stratified sampling for Binary Variable (Nubbh)



### Set up
# Dataset
data <- read.csv(file= "/Users/david/Desktop/STAT 344/user_behavior_dataset.csv")
# Define heavy users
data$heavy_user <- ifelse(data$App.Usage.Time..min.day. > 240, 1, 0)
# Total population size
N <- nrow(data)

### Simple Random Sampling (SRS) ###

# Sample size
n_srs <- 100

# Finite Population Correction (FPC)
fpc_srs <- sqrt((N - n_srs) / (N - 1))
```

```r
### SRS (vanilla)
# Perform SRS
set.seed(1)  # For reproducibility
srs_indices <- sample(1:N, n_srs)
srs_sample <- data[srs_indices, ]

# Proportion of heavy users in the sample
p_hat_srs <- mean(srs_sample$heavy_user)

# Standard error (with FPC)
se_srs <- sqrt(p_hat_srs * (1 - p_hat_srs) / n_srs) * fpc_srs

# Confidence Interval
alpha <- 0.05
z <- qnorm(1 - alpha / 2)
ci_lower_srs <- p_hat_srs - z * se_srs
ci_upper_srs <- p_hat_srs + z * se_srs

# Output results
cat("SRS Estimate of Proportion:", p_hat_srs, "\n")
```

```
## SRS Estimate of Proportion: 0.55
```

```r
cat("SRS Standard Error:", se_srs, "\n")
```

```
## SRS Standard Error: 0.0460919
```

```r
cat("SRS 95% Confidence Interval: [", ci_lower_srs, ",", ci_upper_srs, "]\n\n")
```

```
## SRS 95% Confidence Interval: [ 0.4596615 , 0.6403385 ]
```

```r
###SRS (Ratio estimate)
# Auxiliary variable for Ratio and Regression Estimators (Data Usage)
aux_var <- srs_sample$Data.Usage..MB.day.

# Population mean of the auxiliary variable
X_bar <- mean(data$Data.Usage..MB.day.)

# Sample mean of the auxiliary variable
x_bar <- mean(aux_var)

# Ratio Estimator
r <- mean(srs_sample$heavy_user) / x_bar
p_hat_ratio <- r * X_bar

# Standard error for Ratio Estimator
se_ratio <- sqrt((1 - n_srs / N) * sum((srs_sample$heavy_user - r * aux_var)^2)
                 / (n_srs * (x_bar)^2))

# Confidence Interval for Ratio Estimator
ci_lower_ratio <- p_hat_ratio - z * se_ratio
ci_upper_ratio <- p_hat_ratio + z * se_ratio

cat("Ratio Estimate of Proportion:", p_hat_ratio, "\n")
```

```
## Ratio Estimate of Proportion: 0.5598224
```

```r
cat("Ratio Estimator Standard Error:", se_ratio, "\n")
```

```
## Ratio Estimator Standard Error: 0.0003202059
```

```r
cat("Ratio Estimator 95% Confidence Interval: [", ci_lower_ratio, ",",
    ci_upper_ratio, "]\n\n")
```

```
## Ratio Estimator 95% Confidence Interval: [ 0.5591948 , 0.56045 ]
```

```r
###SRS (Regression estimate)
# Regression Estimator
# Fit linear model
reg_model <- lm(heavy_user ~ Data.Usage..MB.day., data = srs_sample)

# Predicted proportion
B <- reg_model$coefficients["Data.Usage..MB.day."]
p_hat_regression <- mean(srs_sample$heavy_user) + B * (X_bar - x_bar)

# Standard error for Regression Estimator
S_square <- sum(reg_model$residuals^2) / (n_srs - 2)
se_regression <- sqrt(S_square * (1 / n_srs + ((X_bar - x_bar)^2) /
                                  sum((aux_var - x_bar)^2)))

# Confidence Interval for Regression Estimator
ci_lower_regression <- p_hat_regression - z * se_regression
ci_upper_regression <- p_hat_regression + z * se_regression

cat("Regression Estimate of Proportion:", p_hat_regression, "\n")
```

```
## Regression Estimate of Proportion: 0.5612395
```

```r
cat("Regression Estimator Standard Error:", se_regression, "\n")
```

```
## Regression Estimator Standard Error: 0.03154232
```

```r
cat("Regression Estimator 95% Confidence Interval: [", ci_lower_regression, ",",
    ci_upper_regression, "]\n\n")
```

```
## Regression Estimator 95% Confidence Interval: [ 0.4994177 , 0.6230613 ]
```

```r
### Edited Stratified Sampling for Binary Variable

# Stratification variable: Operating System
strata_var <- data$Operating.System

# Number of strata
H <- length(unique(strata_var))

# Proportional Allocation
Nh <- table(data$Operating.System) # Population size per stratum
wh <- Nh / N # Stratum weights
n_strat_prop <- round(wh * 100) # Proportional sample sizes

# Ensure total sample size is exactly 100
n_strat_prop[which.max(n_strat_prop)] <- n_strat_prop[which.max(n_strat_prop)]
+ (100 - sum(n_strat_prop))
```

```
## [1] 0
```

```
# Optimal Allocation (assumes known standard deviations per stratum)
ph <- tapply(data$heavy_user, data$Operating.System, mean) # Proportions per stratum
Sh <- sqrt(ph * (1 - ph)) # Standard deviations per stratum
n_strat_opt <- round((Nh * Sh) / sum(Nh * Sh) * 100) # Optimal sample sizes
n_strat_opt
```

```
##
## Android     iOS
##      79      21
```

```
# Adjust to ensure total sample size is exactly 100
n_strat_opt[which.max(n_strat_opt)] <- n_strat_opt[which.max(n_strat_opt)]
+ (100 - sum(n_strat_opt))
```

```
## [1] 0
```

```
# Function to perform stratified sampling and compute estimates
perform_stratified_sampling <- function(data, n_strat, Nh, wh, ph, z) {
  sampled_data <- list()

  # Perform stratified sampling for each stratum
  for (os in names(n_strat)) {
    strat_data <- data[data$Operating.System == os, ]
    sampled_indices <- sample(1:nrow(strat_data), size = n_strat[os],
                              replace = FALSE)
    sampled_data[[os]] <- strat_data[sampled_indices, ]
  }

  # Combine sampled data
  strat_sample <- do.call(rbind, sampled_data)

  # Estimate stratified proportion
  p_hat_strat <- sum(wh * tapply(strat_sample$heavy_user,
                                 strat_sample$Operating.System, mean))

  # Compute standard error
  se_strat <- sqrt(sum((wh^2 * ph * (1 - ph)) / n_strat))

  # Confidence interval
  ci_lower <- p_hat_strat - z * se_strat
  ci_upper <- p_hat_strat + z * se_strat

  list(estimate = p_hat_strat, se = se_strat, ci_lower = ci_lower,
       ci_upper = ci_upper)
}

# Z-score for 95% confidence interval
z <- 1.96 #qnorm(0.975)

# Perform Stratified Sampling: Proportional Allocation
result_prop <- perform_stratified_sampling(data, n_strat_prop, Nh, wh, ph, z)
cat("Estimate:", result_prop$estimate, "\n")
```

```
## Estimate: 0.4599707
```

```r
cat("Standard Error:", result_prop$se, "\n")
```

```
## Standard Error: 0.04989297
```

```r
cat("95% Confidence Interval: [", result_prop$ci_lower, ",",
    result_prop$ci_upper, "]\n\n")
```

```
## 95% Confidence Interval: [ 0.3621805 , 0.557761 ]
```

```r
# Perform Stratified Sampling: Optimal Allocation
result_opt <- perform_stratified_sampling(data, n_strat_opt, Nh, wh, ph, z)
cat("Estimate:", result_opt$estimate, "\n")
```

```
## Estimate: 0.5199931
```

```r
cat("Standard Error:", result_opt$se, "\n")
```

```
## Standard Error: 0.04989297
```

```r
cat("95% Confidence Interval: [", result_opt$ci_lower, ",",
    result_opt$ci_upper, "]\n\n")
```

```
## 95% Confidence Interval: [ 0.4222029 , 0.6177833 ]
```