IDS 702 HW 1

Your Name Here

Instructions: Use this template to complete your assignment. When you click "Render," you should get a PDF document that contains both your answers and code. You must show your work/justify your answers to receive credit. Submit your rendered PDF file on Gradescope. Remember to render frequently, as this will help you to catch errors in your code before the last minute.

Add your name in the Author section in the header

Exercise 1

For this question, you should type out the formulas that you use to do the calculations (click Insert > Equation > Inline math). You can read more about technical writing in Quarto here. You can use the provided code chunks to use R as a calculator.

```
# Declare and store the given values in variables in case you intend to use
# R as a calculator, Actual variable names do not matter
p_a = 0.2 #P(A)
p_b = 0.4 #P(B)
p_anb = 0.08 #P(AnB)
```

a.
Ques: $P(A|B)$

Ans:
$P(A|B) = \frac{P(A \cap B)}{P(B)}$

$P(A|B) = \frac{0.08}{0.4}$

```
#declare new variable to store the value to use in future if required
p_ab = p_anb/p_b #P(A|B)

#print the value for the user
sprintf("Value of P(A|B) is: %.3f",p_ab)
```

```
[1] "Value of P(A|B) is: 0.200"
```

b.
Ques: $P(A \cup B)$

Ans:
$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$P(A \cup B) = 0.2 + 0.4 - 0.08$

```
#declare new variable to store the value to use in future if required
p_aub =  p_a + p_b - p_anb #P(AuB)

#print the value for the user
sprintf("Value of P(AuB) is: %.3f",p_aub)
```

```
[1] "Value of P(AuB) is: 0.520"
```

c.
Ques:Are A and B mutually exclusive?

Ans:A and B are **NOT** mutually exclusive since they have an intersection probability of > 0 which implies that the two events can occur at the same time.

d.
Ques: Are A and B independent?

Ans:The events A and B are **independent** since the probability of both of the them occurring together is the same as the product of their individual probabilities i.e. $P(A \cap B) = P(A) * P(B)$

Exercise 2

a.
Ques:Which probability distribution best fits this situation?

Ans: The probability distribution function that best fits this situation is the **Binomial Distribution**, for the following reasons:

- The outcomes are binary i.e. Fraud or no-Fraud
- The events can be assumed to be independent of each other, i.e. one customers outcome does not impact other customers
- probability of success is consistent across trials

b.
Ques:How many customers should they expect to experience fraud?

Ans:
The expected value of a Binomial Distribution is given by:
$E(X) = n.p$

where n is the number of trials and p is the probability of success

```
prob_fraud = 0.2 #probability of experiencing fraud ~p
num_cust = 200 #number of customers i.e. the number of trials ~n

#Calculate the value
expect_fraud = prob_fraud * num_cust

#Print for user
sprintf("The bank can expect %d new customers to experience fraud",expect_fraud)
```

```
[1] "The bank can expect 40 new customers to experience fraud"
```

c.
Ques:What is the probability that 50 customers will experience fraud?

Ans:
To calculate this we can use the PMF of the binomial distribution:
$P(S_n = s) = \frac{n!}{(n-s)!s!} p^s q^{n-s}$

where q = 1-p and s is the number of observations of success that we want to count the probability for

```
desired_obs = 50 # the number of observations for which we want to calculate the probability

#you can calculate manually using the formula provided above, or directly use the R function
prob_50 = dbinom(desired_obs, num_cust, prob_fraud)

#Print for user
sprintf("The probability of %d new customers experiencing fraud is %.3f",desired_obs, prob_50
```

```
[1] "The probability of 50 new customers experiencing fraud is 0.015"
```

d.
Ques: What is the probability that no more than 50 customers will experience fraud?

Ans:
This can be done using the Cumulative Distribution Function of the binomial distribution.
$P(X \leq s) = \sum_{k=0}^{s} P(S_n = k)$

i.e. we should sum up the individual PMFs of all the values up to 50(inclusive) which will give us the required value.

```
#you can calculate manually using the formula provided above, or directly use the R function
prob_nomore_50 = pbinom(desired_obs, num_cust, prob_fraud)

#Print for user
sprintf("The probability that no more than %d new customers will experience fraud is %.3f",de
```

```
[1] "The probability that no more than 50 new customers will experience fraud is 0.966"
```

Exercise 3

You are required to show the code you use to complete each part of this exercise. You must also write your narrative answers below the code.

```
population <- read.csv("https://raw.githubusercontent.com/anlane611/datasets/main/population
```

a.

```
mean(population$Y)
```

```
[1] 19.97579
```

b.

```
# Load the dplyr package
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
# Calculate the mean of Y and count the number of observations for each level of X1
result <- population %>%
  group_by(X1) %>%
  summarize(
    mean_Y = mean(Y, na.rm = TRUE),  # Mean of Y
    count = n()                      # Number of observations
  )

# View the result
print(result)
```

```
# A tibble: 5 x 3
     X1 mean_Y count
  <int>  <dbl> <int>
1     1   20.0 19000
2     2   20.0 20100
3     3   20.0 20200
4     4   20.0 22900
5     5   20.0 17800
```

c.

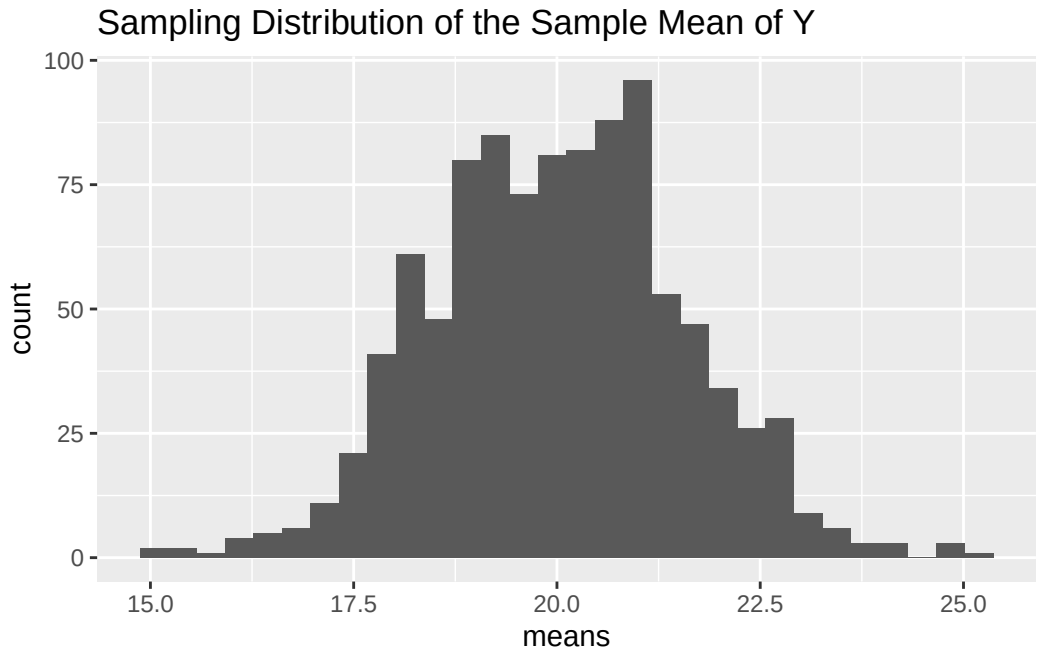
```r
# Calculate the mean of Y and count the number of observations for each level of X1
result_2 <- population %>%
  group_by(X2) %>%
  summarize(
    mean_Y = mean(Y, na.rm = TRUE),  # Mean of Y
    count = n()                      # Number of observations
  )

# View the result
print(result_2)
```

```
# A tibble: 2 x 3
     X2 mean_Y count
  <int>  <dbl> <int>
1     0   33.7   807
2     1   19.9 99193
```

d.

```
library(ggplot2)

SRSmeans <- data.frame(means=NA) #create an empty dataframe to store the sample means
for(i in 1:1000){
  set.seed(i) #ensure that we have a different sample each time
  SRS.sample <- population |> slice(sample(1:n(), size=10))
  SRSmeans[i,1] <- SRS.sample |> summarise(mean_Y = mean(Y))
}

ggplot(SRSmeans, aes(x=means)) +
  geom_histogram()+
  labs(title="Sampling Distribution of the Sample Mean of Y")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Calculate the mean of the sample means
mean_of_sample_means <- mean(SRSmeans$means)

# Calculate the population mean of Y
population_mean <- mean(population$Y)
```

```
mean_of_sample_means
```

```
[1] 20.01105
```

```
population_mean
```

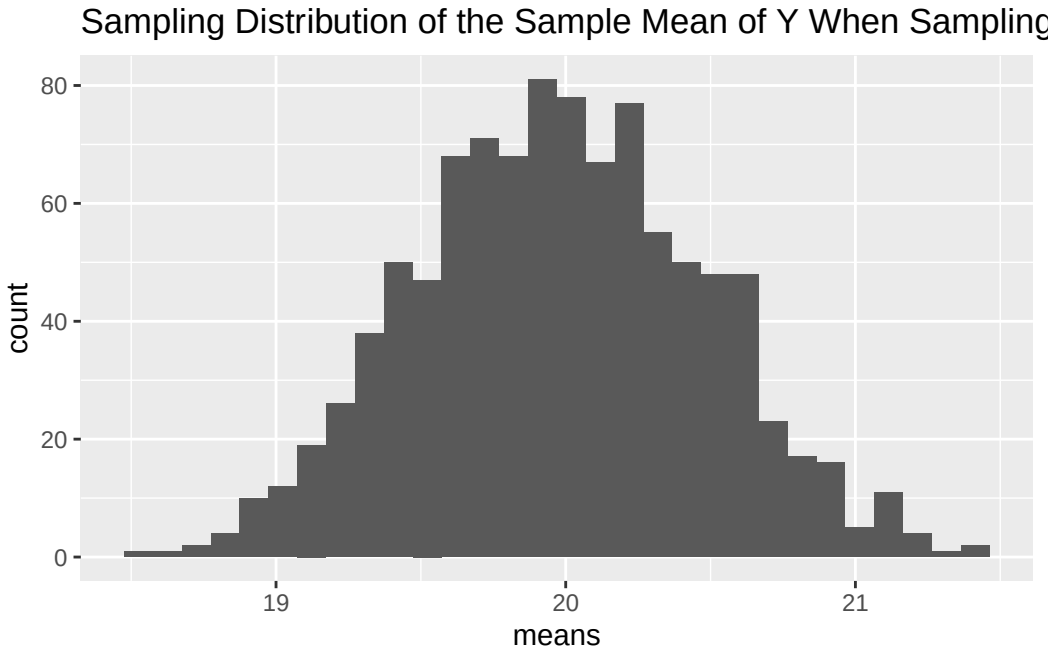
```
[1] 19.97579
```

The histogram shows an approximately normal distribution. Its mean is 20.01, and it is close to the population mean.

e.

```
SRSmeans_100 <- data.frame(means=NA) #create an empty dataframe to store the sample means
for(i in 1:1000){
  set.seed(i) #ensure that we have a different sample each time
  SRS.sample <- population |> slice(sample(1:n(), size=100))
  SRSmeans_100[i,1] <- SRS.sample |> summarise(mean_Y = mean(Y))
}
ggplot(SRSmeans_100, aes(x=means)) +
  geom_histogram()+
  labs(title="Sampling Distribution of the Sample Mean of Y When Sampling 100")
```

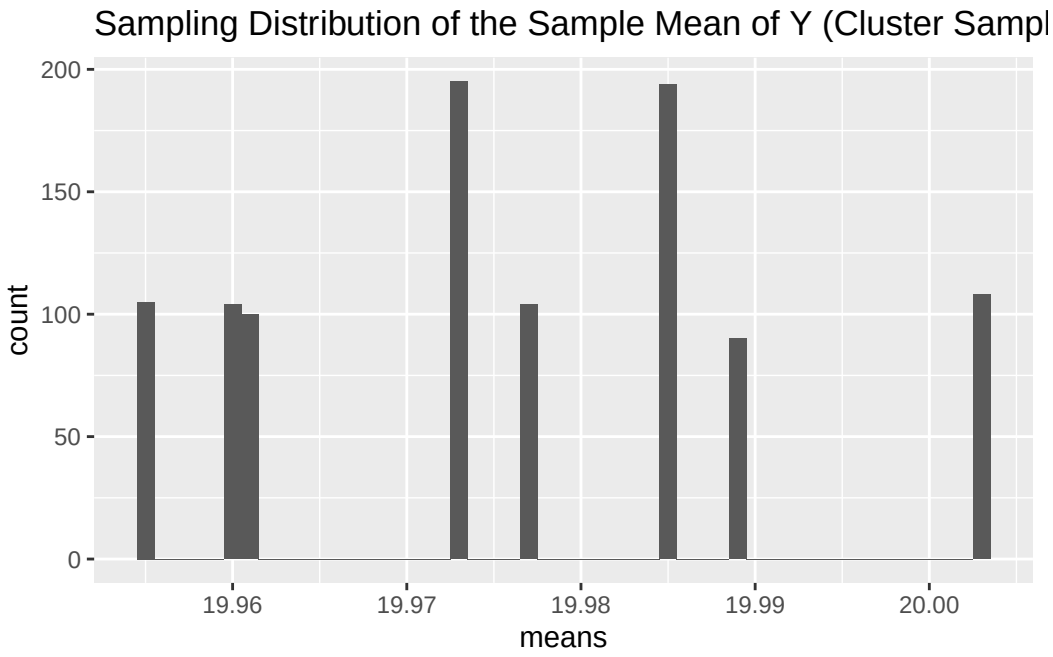
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This histogram and the histogram you generated in part d are similar, but this one is slightly smoother and more mass near the mean (i.e., lower SE).

f.

```r
ClusterMeans <- data.frame(means=NA)

for(i in 1:1000) {
  set.seed(i)

  selected_clusters <- sample(unique(population$X1), size = 2)

  Cluster.sample <- population |> filter(X1 %in% selected_clusters)

  # Calculate the mean of Y in the cluster sample and store it
  ClusterMeans[i, 1] <- Cluster.sample |> summarise(mean_Y = mean(Y))
}

ggplot(ClusterMeans, aes(x=means)) +
  geom_histogram(binwidth = 0.001) +
  labs(title="Sampling Distribution of the Sample Mean of Y (Cluster Sampling)")
```




```
# Calculate the mean of the sample means
mean_of_cluster_means <- mean(ClusterMeans$means)

# Print the result
mean_of_cluster_means
```

```
[1] 19.97604
```

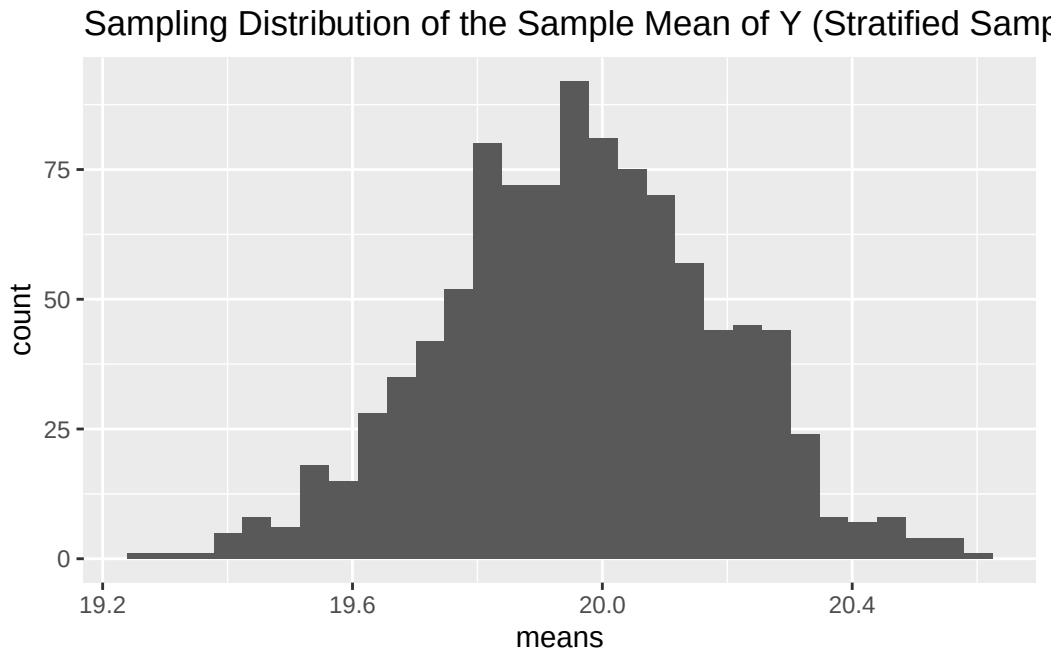
The mean is close to population mean, but the distribution is spread out and not close to normal.

g.

```
StratifiedMeans <- data.frame(means=NA)

for(i in 1:1000) {
  set.seed(i)

  stratified_sample <- population |>
    group_by(X1) |>
    slice(sample(1:n(), size=100, replace=TRUE)) |>
    ungroup()
```

```
  # Calculate the mean of Y in the stratified sample and store it
  StratifiedMeans[i, 1] <- stratified_sample |> summarise(mean_Y = mean(Y))
}

ggplot(StratifiedMeans, aes(x=means)) +
  geom_histogram() +
  labs(title="Sampling Distribution of the Sample Mean of Y (Stratified Sampling with X1)")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Calculate the mean of the sample means
mean_of_stratified_means <- mean(StratifiedMeans$means)

# Print the result
mean_of_stratified_means
```

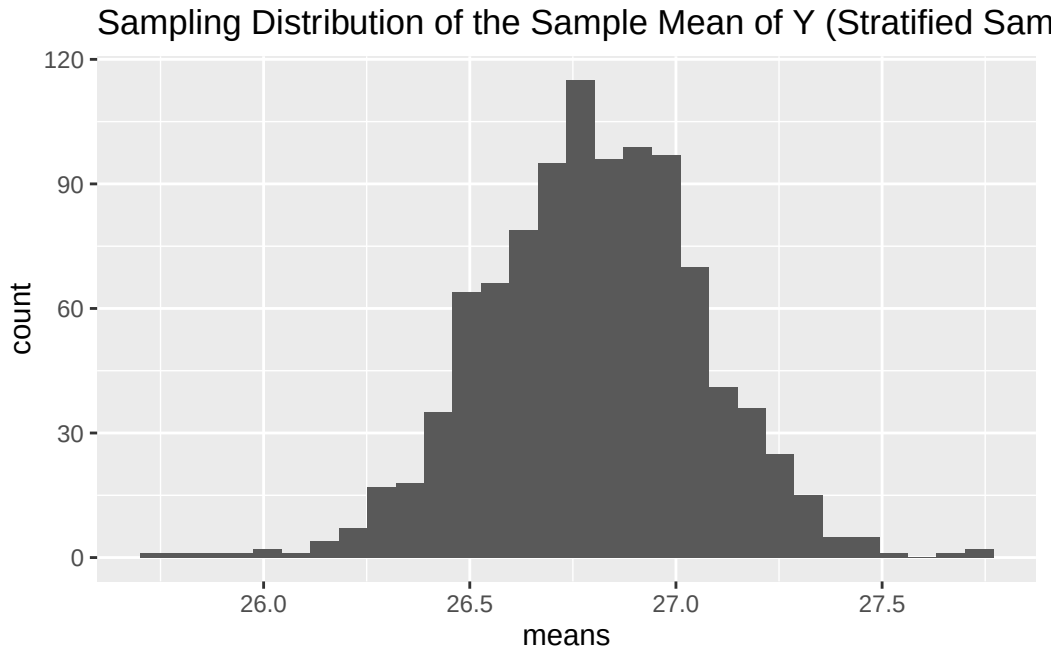
```
[1] 19.9634
```

The mean of the sampling distribution is 19.96. The sampling distribution is approximately normal.

h.

```r
StratifiedMeans_X2 <- data.frame(means=NA)

for(i in 1:1000) {
  set.seed(i)

  stratified_sample <- population |>
    group_by(X2) |>
    slice(sample(1:n(), size=100, replace=TRUE)) |>
    ungroup()

  # Calculate the mean of Y in the stratified sample and store it
  StratifiedMeans_X2[i, 1] <- stratified_sample |> summarise(mean_Y = mean(Y))
}

ggplot(StratifiedMeans_X2, aes(x=means)) +
  geom_histogram() +
  labs(title="Sampling Distribution of the Sample Mean of Y (Stratified Sampling with X2)")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
# Calculate the mean of the sample means
mean_of_stratified_means_X2 <- mean(StratifiedMeans_X2$means)

# Print the result
mean_of_stratified_means_X2
```

```
[1] 26.8008
```

The mean of the sampling distribution is 26.8. The sampling distribution is approximately normal.

i.

In both cases, the sampling distribution is approximately normal. The difference lies in the mean of the sampling distribution. X_1 is balanced, so the resulting stratified sample is representative of the population. Therefore, the sampling distribution of the mean is centered at the population mean. The stratified sample based on X_2, on the other hand, oversamples the tail of the population distribution, so the resulting sampling distribution is not centered at the population mean.

Bonus (optional)

Taking random samples from the clusters should result in a sampling distribution that is closer to normal than in part f because the possible values of the mean are less discrete (the more clusters there are to be sampled, the closer to normal it will be).