2448030\_MDS272\_L4.R

Kevin

2024-10-25

#install.packages("readxl")  
#Q1  
#1  
# Loading necessary libraries  
library(readxl)  
  
# Loading the dataset  
data = read\_excel("C:\\Users\\kmgs4\\Downloads\\LAB\_4\_Data.xlsx")  
sales\_data = data$Sales  
  
# Removing any NA values from the data  
sales\_data = na.omit(sales\_data)  
  
# Setting seed for reproducibility  
set.seed(123)  
  
# Random sampling with replacement (SRSWR)  
sample\_with\_replacement = sample(sales\_data, size = 20, replace = TRUE)  
  
# Random sampling without replacement (SRSWOR)  
sample\_without\_replacement = sample(sales\_data, size = 20, replace = FALSE)  
  
# Calculating the sample means  
mean\_with\_replacement = mean(sample\_with\_replacement)  
mean\_without\_replacement = mean(sample\_without\_replacement)  
  
#2  
# Calculating population mean  
population\_mean = mean(sales\_data)  
  
# Checking the difference between sample means and population mean  
cat("Population Mean:", population\_mean, "\n")

## Population Mean: 1869.036

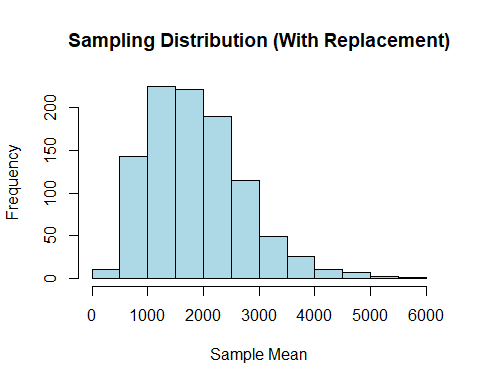
cat("Mean with Replacement:", mean\_with\_replacement, "\n")

## Mean with Replacement: 1135.14

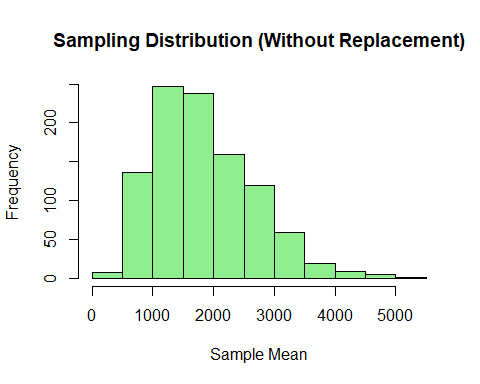
cat("Mean without Replacement:", mean\_without\_replacement, "\n")

## Mean without Replacement: 3878.107

#3  
# Generating sampling distributions by taking multiple samples  
n\_samples = 1000  
sample\_means\_with\_replacement = replicate(n\_samples, mean(sample(sales\_data, size = 20, replace = TRUE)))  
sample\_means\_without\_replacement = replicate(n\_samples, mean(sample(sales\_data, size = 20, replace = FALSE)))  
  
# Plotting histograms  
hist(sample\_means\_with\_replacement, main = "Sampling Distribution (With Replacement)", xlab = "Sample Mean", col = "lightblue")



hist(sample\_means\_without\_replacement, main = "Sampling Distribution (Without Replacement)", xlab = "Sample Mean", col = "lightgreen")



#4  
# Calculating standard errors  
se\_with\_replacement = sd(sample\_means\_with\_replacement)  
se\_without\_replacement = sd(sample\_means\_without\_replacement)  
  
cat("Standard Error with Replacement:", se\_with\_replacement, "\n")

## Standard Error with Replacement: 860.0216

cat("Standard Error without Replacement:", se\_without\_replacement, "\n")

## Standard Error without Replacement: 824.2397

# Verifying that standard error in SRSWOR is smaller than in SRSWR  
cat("Is Standard Error without Replacement smaller?", se\_without\_replacement < se\_with\_replacement, "\n")

## Is Standard Error without Replacement smaller? TRUE

#Q2  
#a)  
# Simulating the data  
binomial\_data = rbinom(50, size = 10, prob = 0.5) # Binomial distribution  
poisson\_data = rpois(50, lambda = 3) # Poisson distribution  
normal\_data = rnorm(50, mean = 10, sd = 2) # Normal distribution  
  
# Displaying the first few values of each  
head(binomial\_data)

## [1] 5 5 6 6 4 2

head(poisson\_data)

## [1] 6 2 2 7 2 3

head(normal\_data)

## [1] 8.257747 9.369174 10.206386 7.810378 13.180413 10.965939

#b) Performing the same tasks for Binomial, Normal, and Poisson  
  
# Binomial Sampling  
set.seed(123)  
sample\_with\_replacement\_binomial = sample(binomial\_data, size = 20, replace = TRUE)  
sample\_without\_replacement\_binomial = sample(binomial\_data, size = 20, replace = FALSE)  
  
mean\_with\_replacement\_binomial = mean(sample\_with\_replacement\_binomial)  
mean\_without\_replacement\_binomial = mean(sample\_without\_replacement\_binomial)  
  
population\_mean\_binomial = mean(binomial\_data)  
cat("Binomial Population Mean:", population\_mean\_binomial, "\n")

## Binomial Population Mean: 5.08

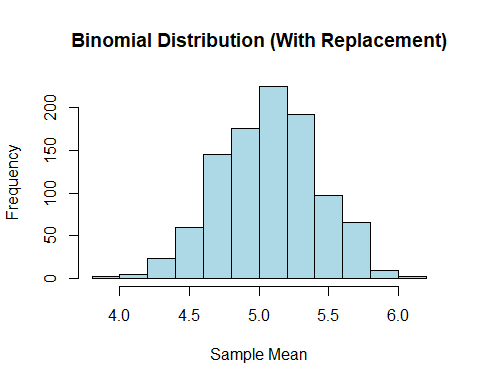
cat("Binomial Mean with Replacement:", mean\_with\_replacement\_binomial, "\n")

## Binomial Mean with Replacement: 5.8

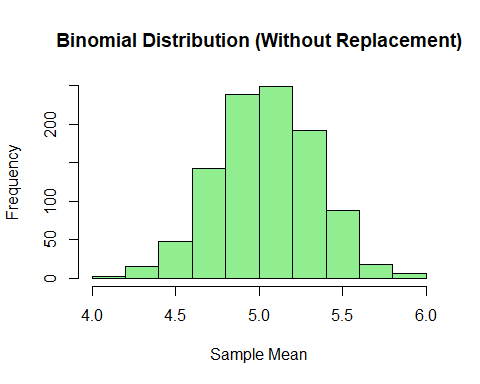
cat("Binomial Mean without Replacement:", mean\_without\_replacement\_binomial, "\n")

## Binomial Mean without Replacement: 5.15

sample\_means\_binomial\_with\_replacement = replicate(n\_samples, mean(sample(binomial\_data, size = 20, replace = TRUE)))  
sample\_means\_binomial\_without\_replacement = replicate(n\_samples, mean(sample(binomial\_data, size = 20, replace = FALSE)))  
  
hist(sample\_means\_binomial\_with\_replacement, main = "Binomial Distribution (With Replacement)", xlab = "Sample Mean", col = "lightblue")



hist(sample\_means\_binomial\_without\_replacement, main = "Binomial Distribution (Without Replacement)", xlab = "Sample Mean", col = "lightgreen")



# Normal Sampling  
set.seed(123)  
sample\_with\_replacement\_Normal = sample(normal\_data, size = 20, replace = TRUE)  
sample\_without\_replacement\_Normal = sample(normal\_data, size = 20, replace = FALSE)  
  
mean\_with\_replacement\_Normal = mean(sample\_with\_replacement\_Normal)  
mean\_without\_replacement\_Normal = mean(sample\_without\_replacement\_Normal)  
  
population\_mean\_Normal = mean(normal\_data)  
cat("Normal Population Mean:", population\_mean\_Normal, "\n")

## Normal Population Mean: 9.898481

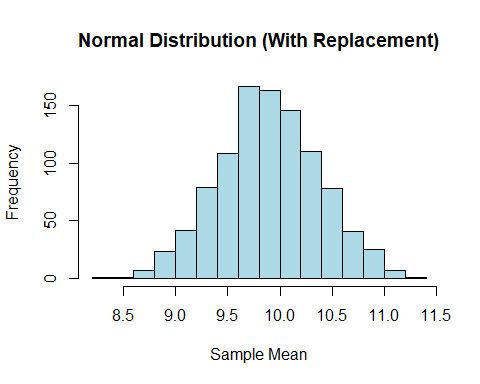
cat("Normal Mean with Replacement:", mean\_with\_replacement\_Normal, "\n")

## Normal Mean with Replacement: 10.28725

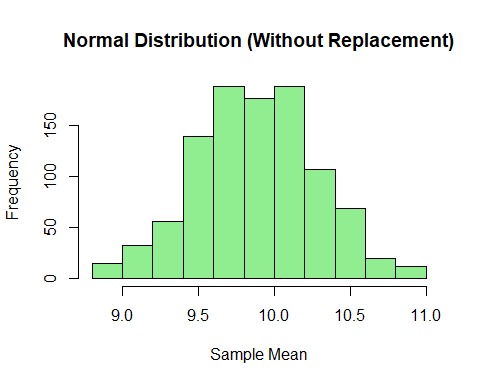
cat("Normal Mean without Replacement:", mean\_without\_replacement\_Normal, "\n")

## Normal Mean without Replacement: 9.749657

sample\_means\_Normal\_with\_replacement = replicate(n\_samples, mean(sample(normal\_data, size = 20, replace = TRUE)))  
sample\_means\_Normal\_without\_replacement = replicate(n\_samples, mean(sample(normal\_data, size = 20, replace = FALSE)))  
  
hist(sample\_means\_Normal\_with\_replacement, main = "Normal Distribution (With Replacement)", xlab = "Sample Mean", col = "lightblue")



hist(sample\_means\_Normal\_without\_replacement, main = "Normal Distribution (Without Replacement)", xlab = "Sample Mean", col = "lightgreen")



# Poisson Sampling  
set.seed(123)  
sample\_with\_replacement\_poisson = sample(poisson\_data, size = 20, replace = TRUE)  
sample\_without\_replacement\_poisson = sample(poisson\_data, size = 20, replace = FALSE)  
  
mean\_with\_replacement\_poisson = mean(sample\_with\_replacement\_poisson)  
mean\_without\_replacement\_poisson = mean(sample\_without\_replacement\_poisson)  
  
population\_mean\_poisson = mean(poisson\_data)  
cat("Poisson Population Mean:", population\_mean\_poisson, "\n")

## Poisson Population Mean: 3.12

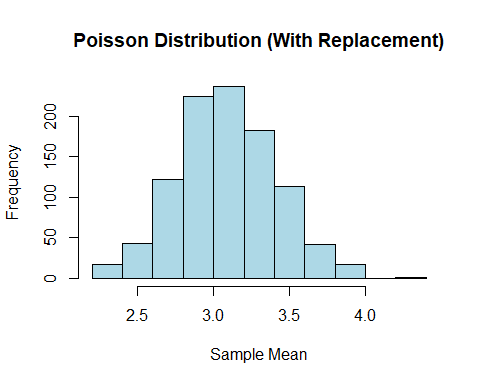
cat("Poisson Mean with Replacement:", mean\_with\_replacement\_poisson, "\n")

## Poisson Mean with Replacement: 3.2

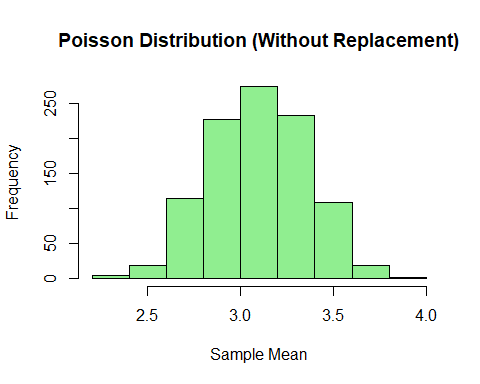
cat("Poisson Mean without Replacement:", mean\_without\_replacement\_poisson, "\n")

## Poisson Mean without Replacement: 3.1

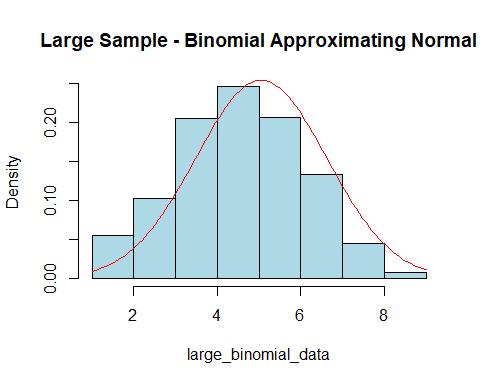
sample\_means\_poisson\_with\_replacement = replicate(n\_samples, mean(sample(poisson\_data, size = 20, replace = TRUE)))  
sample\_means\_poisson\_without\_replacement = replicate(n\_samples, mean(sample(poisson\_data, size = 20, replace = FALSE)))  
  
hist(sample\_means\_poisson\_with\_replacement, main = "Poisson Distribution (With Replacement)", xlab = "Sample Mean", col = "lightblue")



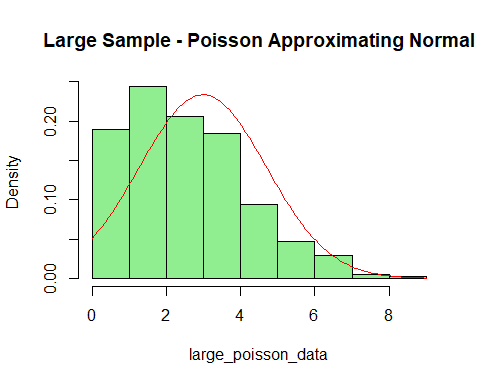
hist(sample\_means\_poisson\_without\_replacement, main = "Poisson Distribution (Without Replacement)", xlab = "Sample Mean", col = "lightgreen")



#c)  
# Generating larger samples for Binomial and Poisson to verify Normal approximation  
large\_binomial\_data = rbinom(1000, size = 10, prob = 0.5)  
large\_poisson\_data = rpois(1000, lambda = 3)  
  
hist(large\_binomial\_data, prob = TRUE, main = "Large Sample - Binomial Approximating Normal", col = "lightblue")  
curve(dnorm(x, mean = mean(large\_binomial\_data), sd = sd(large\_binomial\_data)), col = "red", add = TRUE)



hist(large\_poisson\_data, prob = TRUE, main = "Large Sample - Poisson Approximating Normal", col = "lightgreen")  
curve(dnorm(x, mean = mean(large\_poisson\_data), sd = sd(large\_poisson\_data)), col = "red", add = TRUE)



#Q3  
# Exploring new functions in R  
  
#1. Using `any()` to check if any elements are TRUE  
vec = c(FALSE, FALSE, TRUE, FALSE)  
result = any(vec)  
print(result)

## [1] TRUE

#2. Using `all()` to check if all elements are TRUE  
vec = c(TRUE, TRUE, FALSE)  
result = all(vec)  
print(result)

## [1] FALSE

#3. Using `which()` to find positions of elements greater than a threshold  
vec = c(10, 20, 30, 40, 50)  
nos = which(vec > 25)  
print(nos)

## [1] 3 4 5