Assignment 1 Memo

Dr. Niladri Chakraborty

2025-03-04

Q1.

Ans.

```r
# Define population
population_size = 881
population = seq(1.1, 9.9, length.out = population_size)

# Number of samples per iteration
sample_size = 100

# Initialize variables to store sample means
x1 = mean(sample(population, sample_size, replace = TRUE))
x2 = mean(sample(population, sample_size, replace = TRUE))
x3 = mean(sample(population, sample_size, replace = TRUE))
x4 = mean(sample(population, sample_size, replace = TRUE))
x5 = mean(sample(population, sample_size, replace = TRUE))

# Store sample means in a vector
sample_means = c(x1, x2, x3, x4, x5)

# Compute grand mean and standard deviation of sample means
grand_mean = mean(sample_means)
sd_sample_means = sd(sample_means)

# Print the grand mean and standard deviation
print(paste("Grand Mean:", grand_mean))
```

```
## [1] "Grand Mean: 5.39612"
```

```r
print(paste("Standard Deviation of Sample Means:", sd_sample_means))
```

```
## [1] "Standard Deviation of Sample Means: 0.238231498756986"
```

Q2.

```r
# Define population for integer sampling
integer_population = 1:10

# Number of samples per iteration
sample_size = 100

# Initialize variables to store sample means
```

```r
y1 = mean(sample(integer_population, sample_size, replace = TRUE))
y2 = mean(sample(integer_population, sample_size, replace = TRUE))
y3 = mean(sample(integer_population, sample_size, replace = TRUE))
y4 = mean(sample(integer_population, sample_size, replace = TRUE))
y5 = mean(sample(integer_population, sample_size, replace = TRUE))

# Store sample means in a vector
integer_sample_means = c(y1, y2, y3, y4, y5)

# Compute grand mean and standard deviation of sample means
integer_grand_mean = mean(integer_sample_means)
integer_sd_sample_means = sd(integer_sample_means)

# Print the grand mean and standard deviation
print(paste("Grand Mean:", grand_mean))

## [1] "Grand Mean: 5.39612"

print(paste("Standard Deviation of Sample Means:", integer_sd_sample_means))

## [1] "Standard Deviation of Sample Means: 0.378774339151955"

# Compare standard deviations
print("\nComparison with Q1:")

## [1] "\nComparison with Q1:"

if (integer_sd_sample_means < sd_sample_means) {
  print("The standard deviation of the sample means from the integer
population is smaller than that in Q1.")
} else {
  print("The standard deviation of the sample means from the integer
population is larger than or equal to that in Q1.")
}

## [1] "The standard deviation of the sample means from the integer
population is larger than or equal to that in Q1."

# Explanation
# cat() function works similarly as print() function but it is more robust.

cat("In Q1, the population consists of real numbers spread between 1.1 and
9.9, leading to potentially higher variance.\n")

## In Q1, the population consists of real numbers spread between 1.1 and 9.9,
leading to potentially higher variance.

cat("In this case, the population consists of only 10 discrete integers,
which may lead to a smaller standard deviation of sample means due to reduced
variability.\n")
```

```
## In this case, the population consists of only 10 discrete integers, which
may lead to a smaller standard deviation of sample means due to reduced
variability.
```

Q3.

```
# Set seed for reproducibility
set.seed(123)

# Define population for integer sampling
integer_population = 1:10

# Number of samples per iteration
sample_size = 100

# Initialize variables to store sample means
y1 = mean(sample(integer_population, sample_size, replace = TRUE))
y2 = mean(sample(integer_population, sample_size, replace = TRUE))
y3 = mean(sample(integer_population, sample_size, replace = TRUE))
y4 = mean(sample(integer_population, sample_size, replace = TRUE))
y5 = mean(sample(integer_population, sample_size, replace = TRUE))

# Store sample means in a vector
integer_sample_means = c(y1, y2, y3, y4, y5)

# Compute grand mean and standard deviation of sample means
integer_grand_mean = mean(integer_sample_means)
integer_sd_sample_means = sd(integer_sample_means)

# Print results
cat("Sample Means (Integer Population):", integer_sample_means, "\n")

## Sample Means (Integer Population): 6.03 6.52 5.25 5.65 5.52

cat("Grand Mean (Integer Population):", integer_grand_mean, "\n")

## Grand Mean (Integer Population): 5.794

cat("Standard Deviation of Sample Means (Integer Population):",
integer_sd_sample_means, "\n")

## Standard Deviation of Sample Means (Integer Population): 0.4935889

# Compare standard deviations
cat("\nComparison with Q1:\n")

##
## Comparison with Q1:
```

```
if (integer_sd_sample_means < sd_sample_means) {
  cat("The standard deviation of the sample means from the integer population
is smaller than that in Q1.\n")
} else {
  cat("The standard deviation of the sample means from the integer population
is larger than or equal to that in Q1.\n")
}

## The standard deviation of the sample means from the integer population is
larger than or equal to that in Q1.

# Explanation
cat("\nExplanation:\n")

##
## Explanation:

cat("The standard deviation of sample means is influenced by the spread of
the underlying population.\n")

## The standard deviation of sample means is influenced by the spread of the
underlying population.

cat("In Q1, the population consists of real numbers spread between 1.1 and
9.9, leading to potentially higher variance.\n")

## In Q1, the population consists of real numbers spread between 1.1 and 9.9,
leading to potentially higher variance.

cat("In this case, the population consists of only 10 discrete integers,
which may lead to a smaller standard deviation of sample means due to reduced
variability.\n")

## In this case, the population consists of only 10 discrete integers, which
may lead to a smaller standard deviation of sample means due to reduced
variability.

# Define three vectors X, Y, Z with negative integers
X = c(-2, -5, -7)
Y = c(-3, -6, -9)
Z = c(-1, -4, -8)

# Combine into a 3x3 matrix A
A = cbind(X, Y, Z)

# Compute the transpose of A
A_transpose = t(A)

# Compute the product A'A
A_product = A_transpose %*% A
```

```r
# Compute trace and determinant of A'A
trace_A_product = sum(diag(A_product))
det_A_product = det(A_product)

# Print results
cat("\nMatrix A:\n")

##
## Matrix A:

print(A)

##       X  Y  Z
## [1,] -2 -3 -1
## [2,] -5 -6 -4
## [3,] -7 -9 -8

cat("\nTranspose of A (A'):\n")

##
## Transpose of A (A'):

print(A_transpose)

##   [,1] [,2] [,3]
## X   -2   -5   -7
## Y   -3   -6   -9
## Z   -1   -4   -8

cat("\nProduct A'A:\n")

##
## Product A'A:

print(A_product)

##    X   Y  Z
## X 78  99 78
## Y 99 126 99
## Z 78  99 81

cat("\nTrace of A'A:", trace_A_product, "\n")

##
## Trace of A'A: 285

cat("Determinant of A'A:", det_A_product, "\n")

## Determinant of A'A: 81

# Explanation
cat("\nExplanation:\n")
```

```
##
## Explanation:
```

```
cat("The trace of a matrix is the sum of its diagonal elements. Since A'A is a product of a matrix and its transpose, it is always a positive semi-definite matrix.\n")
```

```
## The trace of a matrix is the sum of its diagonal elements. Since A'A is a
product of a matrix and its transpose, it is always a positive semi-definite
matrix.
```

```
cat("This means that its diagonal elements (which contribute to the trace) are always non-negative. Therefore, the trace cannot be negative.\n")
```

```
## This means that its diagonal elements (which contribute to the trace) are
always non-negative. Therefore, the trace cannot be negative.
```

Q4.

Ans.

```
# Generating the sequences
first_column = seq(1, 9, by = 2)
second_column = seq(5, 1, by = -1)
third_column = rep(2017, 5)  # Repeat 2017 five times

# Combining the columns to form the matrix
matrix = cbind(first_column, second_column, third_column)

# Printing the matrix
print(matrix)
```

```
##      first_column second_column third_column
## [1,]            1             5         2017
## [2,]            3             4         2017
## [3,]            5             3         2017
## [4,]            7             2         2017
## [5,]            9             1         2017
```