Assignment1 Memo

Dr. Niladri Chakraborty

2024-02-23

Q1. (i) Draw a random sample of real numbers of size 1000, with replacement, from the interval (1, 10) (i.e., excluding the endpoints). The population size (the set from where you draw the sample) should be 449. Hence, choose the range accordingly. Calculate the sample mean. You do not have to print all the random samples. Store the sample mean in a variable, say, x1. Then again draw another set of 1000 random samples from the same set, with replacement, calculate the sample mean, and store the sample mean in a different variable, say, x2.

Repeat the process of drawing random samples for 5 times. So, you should have 5 sample mean values stored in variables x1, x2, x3, x4, x5. [2×5 = 10]

(ii) This process of resampling with replacement is the fundamental idea of what we call Bootstrapping. Now calculate the grand mean and the standard deviation of the sample means.

Ans.

```
# Step 1: Define the population
set.seed(42)  # For reproducibility
population <- seq(1.01, 9.99, length.out = 449)  # Adjusted to ensure the
population size is 449 and within (1, 10)

# Step 2: Draw samples and calculate means
sample_means <- numeric(5)  # Initialize a vector to store sample means

for (i in 1:5) {
  samples <- sample(population, size = 1000, replace = TRUE)  # Draw 1000
random samples
  sample_means[i] <- mean(samples)  # Calculate and store the sample mean
}

# Naming the variables for clarity
x1 <- sample_means[1]
x2 <- sample_means[2]
x3 <- sample_means[3]
x4 <- sample_means[4]
x5 <- sample_means[5]

# Step 3: Calculate the grand mean and standard deviation of the sample means
```

```r
grand_mean <- mean(sample_means)
std_deviation <- sd(sample_means)

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

```
## [1] "Grand Mean: 5.5078855625"
```

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

```
## [1] "Standard Deviation of Sample Means: 0.0726241623774469"
```

Q2. (i) Draw a random sample of integers of size 10 000, with replacement, from the interval [1, 10] (i.e., including the endpoints). You do not have to print all the random samples. Calculate the sample mean. Store the sample mean in a variable, say, y1. Then again draw another set of 10 000 random samples from the same set, with replacement, calculate the sample mean, and store the sample mean value in a different variable, say, y2.

Repeat the process of drawing random samples for 5 times. So, you should have 5 sample mean values stored in variables y1, y2, y3, y4, y5. [2×5 = 10]

(ii) Now calculate the grand mean and the standard deviation of the sample means. Is the standard deviation smaller than that in Q1? Explain why.

Ans.

Draw random samples and calculate sample means

```r
set.seed(42)  # Ensure reproducibility

# Initialize a vector to store the sample means
sample_means_y <- numeric(5)

for (i in 1:5) {
  samples <- sample(1:10, size = 10000, replace = TRUE)  # Draw 10,000 random samples
  sample_means_y[i] <- mean(samples)  # Calculate and store the sample mean
}

# Store the sample means in separate variables for clarity
y1 <- sample_means_y[1]
y2 <- sample_means_y[2]
y3 <- sample_means_y[3]
y4 <- sample_means_y[4]
y5 <- sample_means_y[5]
```

Calculate the grand mean and standard deviation

```r
# Calculate the grand mean of the sample means
grand_mean_y <- mean(sample_means_y)
```

```r
# Calculate the standard deviation of the sample means
std_deviation_y <- sd(sample_means_y)

# Print the results
print(paste("Grand Mean:", grand_mean_y))
```

```
## [1] "Grand Mean: 5.49376"
```

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

```
## [1] "Standard Deviation of Sample Means: 0.025632654954179"
```

Explanation:

The larger sample size (10,000) in Q2 compared to Q1 (1,000) generally leads to sample means that are closer to the population mean, thereby reducing the standard deviation of the sample means

Q3. (i) Create three vectors X, Y, Z with all negative integers (any value you like) and each vector has 3 elements. [6]

(ii) Combine the three vectors to become a 3×3 matrix A where each column represents a vector. [3]

(iii) Then obtain the transpose of the matrix A, denoted by A', and multiply it with the original matrix A to obtain the product A'A. Calculate the trace and the determinant of the product A'A. Is it possible to have a negative trace? Explain your answer. [6]

Ans.

Step 1: Creating vectors

```r
X <- c(-1, -2, -3)
Y <- c(-4, -5, -6)
Z <- c(-7, -8, -9)
```

Step 2: Combining Vectors into a Matrix

```r
A <- cbind(X, Y, Z)  # Combine vectors into a matrix
```

Step 3: Operations on Matrix A

```r
A_transpose <- t(A)  # Transpose of A
product <- A_transpose %*% A  # Matrix multiplication
```

Step 4: Calculate the Trace and the Determinant of the Product $A'A$

```r
trace_A_prime_A <- sum(diag(product))  # Trace of A'A
determinant_A_prime_A <- det(product)  # Determinant of A'A
```

The trace of a matrix is the sum of its diagonal elements. Since $A'A$ results in a matrix with positive diagonal elements (because squaring any real number, whether positive or

negative, results in a non-negative number, and the diagonal elements of $A'A$ are sums of squares of elements of A, the trace of $A'A$ cannot be negative.

Q4. Create the following matrix with suitable R functions. Manual entries will not be accepted.

[,1] [,2] [,3]

[1,] 1 5 2017

[2,] 3 4 2017

[3,] 5 3 2017

[4,] 7 2 2017

[5,] 9 1 2017

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
```