Practice / First Steps in Data Analytics

Nithya Sarabudla

01-21-2024

Question 1

```
# Load the data set from the link above into a dataframe
customer_data <- read.csv("/Users/nithyasarabudla/Downloads/customertxndata.csv")
# Inspect the data set using the str() function
str(customer_data)
```

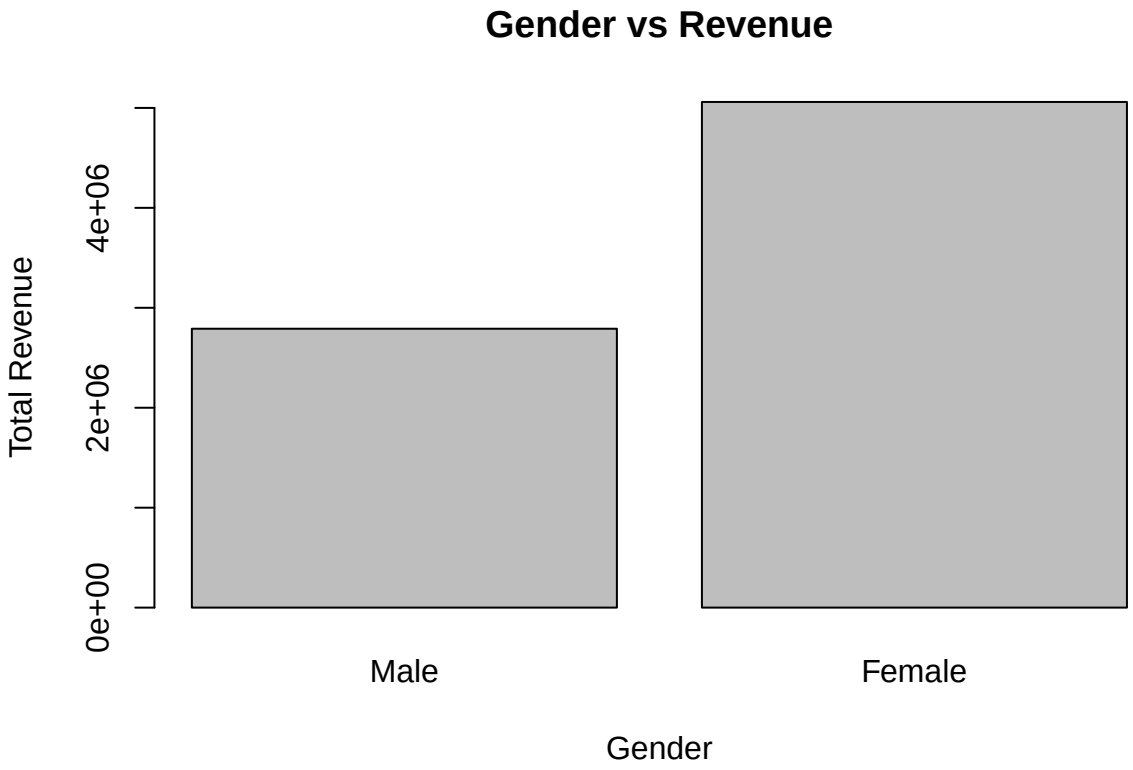
```
## 'data.frame':    22800 obs. of  5 variables:
##  $ numvisits: int  7 20 22 24 1 13 23 14 11 24 ...
##  $ numtxns  : int  0 1 1 2 0 1 2 1 1 2 ...
##  $ OS       : chr  "Android" "iOS" "iOS" "iOS" ...
##  $ gender   : chr  "Male" NA "Female" "Female" ...
##  $ rev      : num  0 577 850 1050 0 ...
```

Question 3

Data Analysis

The total revenue of all transactions is $ 10,372,524, while the median transaction amount is $ 344.65 with a standard deviation of 425.99. The average number of visits was 12. The majority of customers were "Male".

Question 4



The above barplot suggests that the revenue generated by male customers was nearly double that of female customers.

Question 5

```
# Pearson Moment of Correlation between number of visits and revenue
correlation <- cor(customer_data$numvisits, customer_data$rev, method = 'pearson')
correlation
```

```
## [1] 0.7388448
```

Pearson moment of correlation determines how closely two variables are related. Correlation between variables means if one variable's value changes, the other tends to change in the same way. -1 indicates the perfect negative correlation, +1 indicates the positive linear correlation and 0 means no correlation. 0 to 0.30 is negligible correlation, 0.30 to 0.50 is moderate correlation and 0.50 to 1 is highly correlation.

Here the Pearson correlation coefficient between the number of visits and revenue is 0.72. This indicates a positive correlation, suggesting that as the number of visits increases, the revenue tends to increase. The strength of the correlation is high.

Question 6

```
# Identify columns with missing data
missing_columns <- colnames(customer_data)[colSums(is.na(customer_data)) > 0]
# Comment on missing data in markdown
cat("The following columns have missing data:", paste(missing_columns, collapse = ", "), ".\n")
```

```
## The following columns have missing data: numtxns, gender .
```

```
cat("To recognize missing data, I used the `is.na()` function to check for NA values in each column.\n")
```

```
## To recognize missing data, I used the ‘is.na()‘ function to check for NA values in each column.
```

```
# Count the number of missing values for each variable
missing_gender <- length(which(is.na(customer_data$gender)))
missing_txns <- length(which(is.na(customer_data$numtxns)))

# Display the number of missing values for each variable
cat("Number of missing values for 'gender':", missing_gender, "\n")
```

```
## Number of missing values for ’gender’: 5400
```

```
cat("Number of missing values for 'numtxns':", missing_txns, "\n")
```

```
## Number of missing values for ’numtxns’: 1800
```

The data set has a few columns that have missing data, specifically, there are 5400 values for “gender” and 1800 values for “numtxns” missing from the data set.

For ‘gender’, as it is a categorical variable, a common strategy is to impute missing values with the mode (most frequently occurring gender).For ‘numtxns’, representing the number of transactions, imputation could be done using the mean or median value of the non-missing transactions.

Question 7

```
# Impute missing values for 'numtxns' with the mean (rounded to the nearest whole number)
mean_numtxns <- round(mean(customer_data$numtxns, na.rm = TRUE))
customer_data$numtxns[is.na(customer_data$numtxns)] <- mean_numtxns

# Impute missing values for 'gender' with the mode
mode_gender <- names(sort(table(customer_data$gender), decreasing = TRUE))[1]
customer_data$gender[is.na(customer_data$gender)] <- mode_gender
```

Data Analysis

The total revenue of all transactions is $ 10,372,524, while the median transaction amount is $ 344.65 with a standard deviation of 425.99. The average number of visits was 12. The majority of customers were “Male”.

The total transaction amount (revenue), mean number of visits, median revenue, and standard deviation of revenue and the most common gender remained consistent. There is no change in the values.

Data Splitting for Model Training and Validation

Question 8

```
training_data <- customer_data[seq(1, nrow(customer_data), by = 2), ]
validation_data <- customer_data[seq(2, nrow(customer_data), by = 2), ]
```

This section of code splits the original dataset into two equally sized datasets for training and validation purposes. It assigns every odd-numbered case (rows 1, 3, 5, 7, etc.) to the training dataset and every even-numbered case (rows 2, 4, 6, etc.) to the validation dataset.

Question 9

```
## Calculate Mean Revenue for Training and Validation Data
mean_revenue_training <- mean(training_data$rev)
mean_revenue_validation <- mean(validation_data$rev)
```

After splitting the dataset into training and validation sets, the mean revenue for the training dataset is calculated as 449.6, while the mean revenue for the validation dataset is 460.3. This difference in mean revenue between the training and validation datasets suggests that there may be some variability in revenue distribution between the two sets.

Question 10

```
## Data Splitting for Model Training, Testing, and Validation

# Set the seed for reproducibility
set.seed(77654)

# Define the proportions for training, testing, and validation
train_prop <- 0.6
test_prop <- 0.2
validation_prop <- 0.2

# Calculate the sample sizes based on proportions
train_size <- floor(train_prop * nrow(customer_data))
test_size <- floor(test_prop * nrow(customer_data))
validation_size <- nrow(customer_data) - train_size - test_size

# Use sample function to split the data
train_indices <- sample(1:nrow(customer_data), size = train_size, replace = FALSE)
test_indices <- sample(setdiff(1:nrow(customer_data), train_indices), size = test_size, replace = FALSE)
validation_indices <- setdiff(1:nrow(customer_data), c(train_indices, test_indices))

# Create training, testing, and validation datasets
training_data <- customer_data[train_indices, ]
testing_data <- customer_data[test_indices, ]
validation_data <- customer_data[validation_indices, ]
```

```r
# Display the dimensions of the resulting datasets
cat("Dimensions of Training Data:", dim(training_data), "\n")
```

```
## Dimensions of Training Data: 13680 5
```

```r
cat("Dimensions of Testing Data:", dim(testing_data), "\n")
```

```
## Dimensions of Testing Data: 4560 5
```

```r
cat("Dimensions of Validation Data:", dim(validation_data), "\n")
```

```
## Dimensions of Validation Data: 4560 5
```