Solution / First Steps in Data Analytics with R

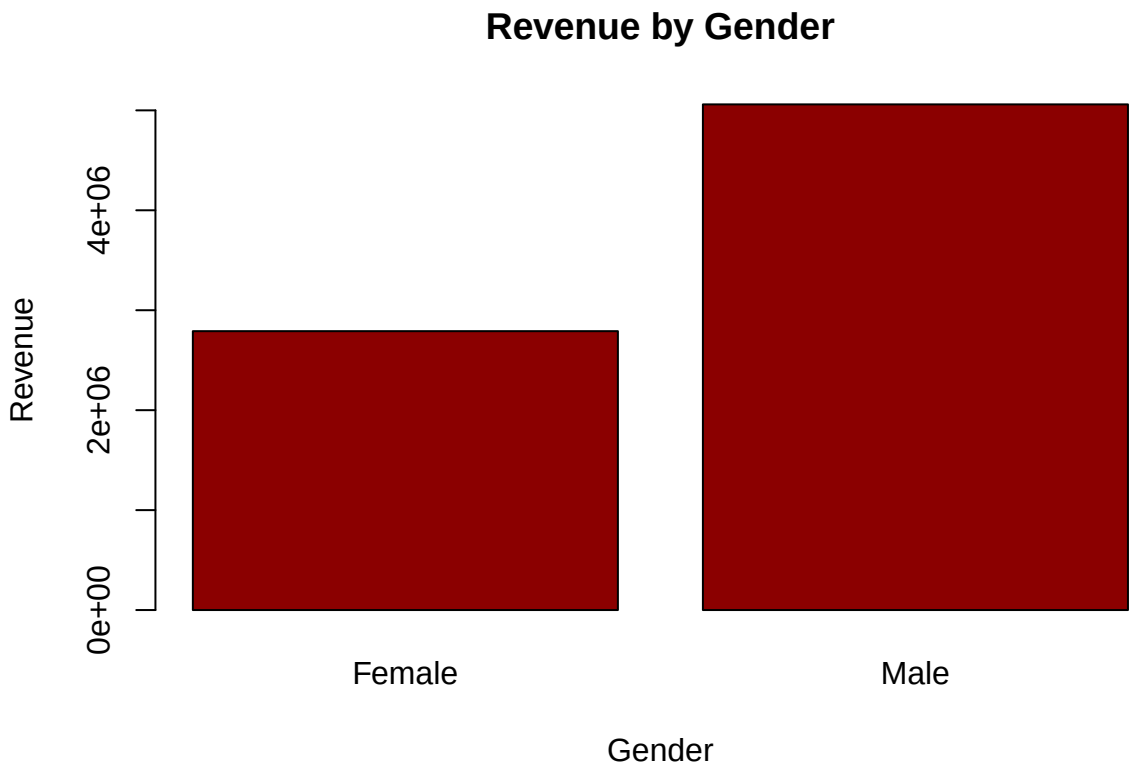
Martin Schedlbauer, PhD

Data Analysis

The customer data set contains 22800 transactions and has the following structure:

```
'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 ...
```

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 barplot indicates that the revenue from "male" customers was almost twice as much as those of "female" customers.

Correlation Analysis

There appears to be a correlation between the number of visits a customer made and the revenue spent on a transaction ($r = 0.74$).

Missing Data Analysis

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. When analyzing the data we could either ignore transactions where the gender is missing, or we could either assume a default gender, assign a neutral gender values, or impute the gender with the most commonly occurring gender. For the number of transactions, we could impute missing values with the mean or using a machine learning algorithm to estimate them from the other variables for the row.

Imputing Missing Values

We will use mode to impute missing gender values and mean to impute missing values for the number of transactions.

```
df$gender[which(is.na(df$gender))] <- mode.gender
df$numtxns[which(is.na(df$numtxns))] <- mean(df$numtxns, na.rm = T)
```

Of course, imputing changes the values for the various statistics, although it might be more meaningful than to simply ignore them. Often the context determines whether ignoring missing values or using imputed values is best.

Using the imputed values, we have a total revenue of all transactions of $ 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".

Training and Validation Samples

Splitting the data into two subsets is common in machine learning. The larger subset is used to train the model, while the smaller subset is held back for validation of the model. For now, we will split the data between training and validation evenly and use the odd numbered cases for training and the even numbered cases for validation, *i.e.*, rows 1, 3, 5, 7, *etc.* are training data while rows 2, 4, 6, *etc.* are validation data.

```
even.rows <- seq(from = 2, to = nrow(df), by = 2)
odd.rows <- seq(from = 1, to = nrow(df), by = 2)

df.train <- df[even.rows,]
df.val <- df[odd.rows,]
```

Assuming that the rows are somewhat random, the mean revenue between the validation and training data sets should be about the same. The actual values are 460.3 for the training sample and 449.6 for the validation sample. The difference is minor and a t-test shows, as expected, that the difference is not statistically significant ($t = 1.888$, $p > 0.05$).

Rather than evenly splitting the data, it is better to take a random sample, particularly when we are not sure that the data is not somehow ordered. The code below takes a random sample of 60% for training, 20% for testing, and another 20% for validation.

```r
# Set the seed so that same sample can be reproduced in the future
set.seed(101)

# Draw 60% of the data from total 'n' rows of the data
sample <- sample.int(n = nrow(df), size = floor(.60*nrow(df)), replace = F)
train <- df[sample, ]

# Take a 50% sample (half) from the remaining 40%
remaining  <- df[-sample, ]
sample <- sample.int(n = nrow(remaining), size = floor(.50*nrow(remaining)), replace = F)
testing <- remaining[sample,]
validation <- remaining[-sample,]
```