# Dataquest Guided Project: Creating An Efficient Data Analysis Workflow

Cindy Zhang

8/2/2020

## Contents



## Introduction

This is my solution to Dataquest's COVID-19 Guided Project from Course 3 (Course Flow, Iteration and Functions in R), which evaluates which books from a publishing company are the most profitable.

More details, such as descriptions for variables, can be found in the "ReadMe" file of this project's repository in GitHub.

## Findings

### Step 1: Getting Familiar with The Data

I loaded the data as a data frame and examined its dimensions, column titles, and column types:

```
booksales_df <- data.frame(read.csv("book_reviews.csv"), stringsAsFactors = FALSE)
dim(booksales_df)
```

```
## [1] 2000    4
```

```
colnames(booksales_df)
```

```
## [1] "book"   "review" "state"  "price"
```

```
for (i in colnames(booksales_df)) {
  print(class(booksales_df[[i]]))
}
```

```
## [1] "factor"
## [1] "factor"
## [1] "factor"
## [1] "numeric"
```

There are 2000 rows and 4 columns in the dataframe. Each of the columns in `booksales_df` seems to represent one entry per unique customer that purchased books from the company. When I ran the for loop above to display each column's type, I found that each column is stored as a factor. This is obviously not ideal for working with this dataset, as columns such as `user_submitted_review` should be a character type and there are not levels to the data in these columns as a factor type would suggest. I used the `purrr` package to convert the factor type columns to characters below:

```
booksales_df %>%
  map_if(is.factor, as.character) %>%
  as_tibble -> booksales_df
```

To understand how high and low the values for each column go, I ran a for loop that prints unique values for each column below:

```
for (i in colnames(booksales_df))
  print(unique(booksales_df[[i]]))
```

```
## [1] "R Made Easy"                          "R For Dummies"
## [3] "Secrets Of R For Advanced Students" "Top 10 Mistakes R Beginners Make"
## [5] "Fundamentals of R For Beginners"
## [1] "Excellent" "Fair"      "Poor"      "Great"     NA          "Good"
## [1] "TX"         "NY"         "FL"         "Texas"      "California"
## [6] "Florida"    "CA"         "New York"
## [1] 20 16 50 30 40
```

## Step 2: Handling Missing Data

I ran a for loop to display which columns had NA values and how many:

```
for (i in colnames(booksales_df))
  print(sum(is.na(booksales_df[[i]])))
```

```
## [1] 0
## [1] 206
## [1] 0
## [1] 0
```

The results show that the `review` column contains 206 NA values. To remove the NA values from the dataset, I created a new dataframe that filters out the NA values from the `review` column:

```
booksales_df_filter <- booksales_df %>%
  filter(!(is.na(review)))
```

Filtering out NA values took out 206 observations from the dataset, which is about 10.3 percent of the original dataset removed. This may have an impact on calculations performed later in this project.

## Step 3: Dealing With Inconsistent Labels

Back in Step 1 when I used a for loop to examine unique values for each column, I discovered that the `state` column had inconsistent labeling (e.g., "CA" and "California"). To standardize the state labeling for each row as abbreviations, I created a new column, `state_abb`, that converted each entry under the original `state` into an abbreviation using an ifelse statement. I then removed the original `state` column from the `booksales_df_filter` dataframe.

```
booksales_df_filter <- booksales_df_filter %>%
  mutate(state_abb = ifelse(state=="California"|state=="CA", "CA", ifelse(state=="Florida"|state=="FL",
booksales_df_filter$state=NULL
```

## Step 4: Transforming the Review Data

To evaluate the reviews under the `review` column, it is helpful to convert them into numeric scores. I used the `case_when()` function to convert the text reviews to a 1 to 5 rating system.

```
booksales_df_filter <- booksales_df_filter %>%
  mutate(review_num = case_when(
    review == "Poor" ~ 1,
    review == "Fair" ~ 2,
    review == "Good" ~ 3,
    review == "Great" ~ 4,
    review == "Excellent" ~ 5
  ))
```

I added another column that displays whether the review is a high score or not- a "high" score is a 4 or higher.

```
booksales_df_filter <- booksales_df_filter %>%
  mutate(is_high_review = case_when(
    review_num >= 4 ~ "TRUE",
    review_num < 4 ~ "FALSE"
  ))
```

## Step 5: Analyzing the Data

After cleaning the data in the previous steps, I analyzed the data to determine which books are the most profitable. I chose to measure profitability by total revenue generated by each book (normally profitability is measured by subtracting costs from revenue, but this dataset does not contain this information).

```r
booksales_df_filter_sum <- booksales_df_filter %>%
  group_by(book) %>%
  summarize(revenue = sum(price)) %>%
  arrange(desc(revenue))
booksales_df_filter_sum
```

```
## # A tibble: 5 x 2
##   book                                revenue
##   <chr>                                 <dbl>
## 1 Secrets Of R For Advanced Students  18000
## 2 Fundamentals of R For Beginners     14636.
## 3 Top 10 Mistakes R Beginners Make    10646.
## 4 R Made Easy                          7036.
## 5 R For Dummies                        5772.
```

## Conclusion

Based on the definition of profitability established back in Step 5, Secrets Of R For Advanced Students is the most profitable textbook. As mentioned in Step 5, the more accurate definition of profitability is the difference between total revenue and total costs, but the dataset does not contain this information. Additional information that could improve this analysis would be dates each customer purchased a textbook; knowing when each purchase would made would help us understand profit trends over time. It would also make the review data more useful because you could examine if sales of poorly reviewed textbooks dropped off over time.