Guided Project: Creating An Efficient Data Analysis Workflow

Lisa Reiber

26/07/2021

Getting familiar with the data

It's easy to lose context when we're just talking about data analysis in general. The first thing we should do before we do any analysis is to get acquainted with our dataset. There are many, many things to check with a dataset before we dive into any analysis. How much data is there? What kind of data do we actually have on hand? Is there anything "weird" that might interfere with any analyses we might need to do? Is there missing data? Answering these questions now saves we time and effort later.

If we don't check the data beforehand, it's easy to make some false assumptions about the data that can hinder our progress later. Maybe we think that one column looks like a number, but it's actually been read in as a string. Perhaps some things were misspelled. In any case, getting familiar with the data is our first step in the data analysis workflow. Here's a few helpful questions to start out any data analysis. Answer each of these questions for the sales dataset. (instructions from the course are always in italic)

Instructions

1. How big is the dataset?

- Recall we can use the `dim()` function to see how many rows and columns are in a tibble.
- You should take a note of how many columns there are and how many rows there are.
- Write some notes to ourself to keep track of what we see in the dataset.

There are **2000 rows** and **4 columns/variables** in the data

2. What are the column names?

- Recall we can use the 'colnames()' function to return each of the column names of a tibble in a vector.
- What do each of the columns seem to represent?

The names of the columns are: **book, review, state, price**

3. What are the *types* of each of the columns?

- Sometimes you may find that data will look one way, but is actually disguised as another. As mentioned above, a common example of this is numbers that are actually strings in the data!
- You can use the `typeof()` function to learn what the type is for each column.

- This would be a good application of a `for` loop. Since we have the column names from `colnames()`, we can loop through these and check the types of each of the column. Recall that you can access a column in a tibble with double bracket syntax `tibble_data[["column_name"]]` where `tibble_data` is a tibble and `column_name` is a string with the column name. Combining this with `typeof()` will give you back the type of the column itself.

for loop + print Here, the class.output option does not work, because we set the results to 'asis'. In the ouput, the code is therefore within the green box, rather than above it. As a work around, I copied the code and once it is not evaluated (eval = FALSE). When it is evaluated, it will not show the code echo = FALSE).

Helpful here: the results = 'asis' function.

```
for (var in seq_along(reviews_df)) {

  var_type <- typeof(reviews_df[[var]])
  names(var_type) <- names(reviews_df[var])

  var_type %>% glue_data("- `{names(.)}`: {.}") %>% print()
  # var_type %>% cat(labels = paste0("- `", names(.), "`: "),
  #     fill = TRUE,
  #     append = TRUE)
}
```

Variables and their unique types:

- `book`: character
- `review`: character
- `state`: character
- `price`: double

```
reviews_df %>%
  map_chr(typeof) %>%
  as_tibble(rownames = "column") %>%
  flextable() %>%
  theme_alafoli()
```

column	value
book	character
review	character
state	character
price	double

```
# reviews_df %>% map_df(typeof) %>%
#       pivot_longer(cols = everything(),
#                    names_to = "column",
#                    values_to = "type")
```

map

4. What are the unique values present in each of the columns?

- The If we're dealing with numbers, it's good to get a sense of how high and how low the values go. If we're dealing with strings, it's good to see all of the different possible values that are there.
- We want to print the columns and all the unique values per column
- We can use glue, which makes it possible to add the "and" before the last element

```
for (var in seq_along(reviews_df)) {

  unique_values <- reviews_df[[var]] %>%
    unique() %>%
    glue_collapse(sep = ", ", last = ", and ")

  names(unique_values) <- names(reviews_df[var])

  unique_values %>%
    glue_data("- `{names(.)}`: {.}") %>% print()
  # unique_values %>% cat(labels = paste0("- `", names(.), "`: "),
  #     fill = TRUE,
  #     append = TRUE)
}
```

for loop + print **Variables and their unique values:**

- `book`: R Made Easy, R For Dummies, Secrets Of R For Advanced Students, Top 10 Mistakes R Beginners Make, and Fundamentals of R For Beginners
- `review`: Excellent, Fair, Poor, Great, and Good
- `state`: TX, NY, FL, Texas, California, Florida, CA, and New York
- `price`: 19.99, 15.99, 50, 29.99, and 39.99

predefined for loop how to set names to a list in loop

```
list_unique_values <- vector(mode = "list", length = ncol(reviews_df))
names(list_unique_values) <- colnames(reviews_df)

for (var in seq_along(reviews_df)) {
  list_unique_values[[var]] <- reviews_df[[var]] %>% unique() %>% na.omit()
  # don't like this line to get rid of NA values
  # list_unique_values[[var]] <- list_unique_values[[var]][!is.na(list_unique_values[[var]])]
}

list_unique_values %>%
  map(~glue_collapse(.x, sep = ", ", last = ", and ")) %>%
  glue_data("- `{names(.)}`: {.}")
```

- `book`: R Made Easy, R For Dummies, Secrets Of R For Advanced Students, Top 10 Mistakes R Beginners Make, and Fundamentals of R For Beginners
- `review`: Excellent, Fair, Poor, Great, and Good
- `state`: TX, NY, FL, Texas, California, Florida, CA, and New York
- `price`: 19.99, 15.99, 50, 29.99, and 39.99

```
cat_description <- function(myvector, vec_names){
  cat("- `", vec_names, "`: ",
      paste0(unique(myvector), sep = ", "), " \n", sep = "")
}

reviews_df %>% iwalk(cat_description)
```

iwalk

- `book`: R Made Easy, R For Dummies, Secrets Of R For Advanced Students, Top 10 Mistakes R Beginners Make, Fundamentals of R For Beginners,
- `review`: Excellent, Fair, Poor, Great, NA, Good,
- `state`: TX, NY, FL, Texas, California, Florida, CA, New York,
- `price`: 19.99, 15.99, 50, 29.99, 39.99,

map Approach 1 is to print a tibble

```
reviews_df %>%
  map_chr(~glue_collapse(unique(.x) %>% na.omit(), sep = ", ", last = ", and ")) %>%
  as_tibble(rownames = "column") %>%
  flextable() %>%
  theme_alafoli() %>%
  autofit()
```

column	value
book	R Made Easy, R For Dummies, Secrets Of R For Advanced Students, Top 10 Mistakes R Beginners Ma
review	Excellent, Fair, Poor, Great, and Good
state	TX, NY, FL, Texas, California, Florida, CA, and New York
price	19.99, 15.99, 50, 29.99, and 39.99

Approach 2 is to use glue_data to print each row in the fomat we specify

```
reviews_df %>%
  map(~glue_collapse(unique(.x) %>% na.omit(), sep = ", ", last = ", and ")) %>%
  glue_data("- Column `{names(.)}` unique values: {.}.")
```

- Column `book` unique values: R Made Easy, R For Dummies, Secrets Of R For Advanced Students, Top 10 Mistakes R Beginners Make, and Fundamentals of R For Beginners.
- Column `review` unique values: Excellent, Fair, Poor, Great, and Good.
- Column `state` unique values: TX, NY, FL, Texas, California, Florida, CA, and New York.
- Column `price` unique values: 19.99, 15.99, 50, 29.99, and 39.99.

Handling Missing Data

Now that we are more familiar with the data itself, now we can get into the more specific details of data analysis. A large part of a data analyst's job is to take a raw dataset and turn it into a form that we can use

	Number of missing values
book	0
review	206
state	0
price	0

for analysis. Many times, we will not be able to just take a dataset and start analyzing it. It's good practice to examine the data beforehand and make note of any changes we need to make for it. This process has many names, but for future reference, we'll call it data cleaning or data processing.

The first issue we will contend with is the issue of missing data. Missing data is annoying because there's nothing we can really do with it. We can't perform any analysis or calculations with missing data. In R, missing data is typically shown with NA, which stands for "not available". Some other datasets may convey non-existence in a different way, but NA is the most common.

There are two ways that we can deal with missing data: 1) remove any rows or columns that have missing data (typically, rows) or 2) fill in the missing data in an informed, discipline way. This second way is known as imputation, and it's outside of the scope of what we've learned so far. For now, we'll take first approach with this dataset.

Instructions

1. Examine the data and get an understanding of which columns have data missing.

- Since we're going to check through each of the columns, a `for` loop would be useful in this case.

```
n_missings <- vector(mode = "logical", length = ncol(reviews_df))
names(n_missings) <- colnames(reviews_df)

for (var in seq_along(reviews_df)) {
  n_missings[var] <- sum(is.na(reviews_df[[var]]))
}

n_missings %>%
  kableExtra::kbl(col.names = "Number of missing values") %>%
  kableExtra::kable_minimal(full_width = F)
```

for loop + kable

```
reviews_df %>%
  map_chr(~any(is.na(.x))) %>%
  kableExtra::kbl(col.names = "Is Missing?") %>%
  kableExtra::kable_minimal(full_width = F)
```

```
reviews_df %>%
  map_chr(~sum(is.na(.x))) %>%
```

	Is Missing?
book	FALSE
review	TRUE
state	FALSE
price	FALSE

	Number of missing values
book	0
review	206
state	0
price	0

```
  kableExtra::kbl(col.names = "Number of missing values") %>%
  kableExtra::kable_minimal(full_width = F)
```

map+kable

2. Subset Data

Create a new copy of the dataset that removes all of the rows that have missing data.

- Here, you should try to apply the `filter()` and `is.na()` functions to get to this new dataset without missing values.

In the previous section, we found out that there are only missings in the review column. Instead of dropping those rows, we can add a flag that indicated whether there are missings in the row or not. This allows us to use all of the information for questions that do not involve the reviews, and also to compare which books do not have reviews (and why?).

```
reviews_meta <- reviews_df %>%
  rowid_to_column(var = "id") %>%
  mutate(is_complete = rowSums(is.na(.)) == 0)
```

3. Inspect Data

Make a note to ourself on the dimensions of the new dataset. How much data was removed by taking out the rows with missing data?

- There's no explicitly correct answer for this, but do we think that removing that much data will affect the findings of any calculations we end up doing? It's important to keep questions like this in mind as we clean our data.

The new dataset has 206 rows less

```
reviews_meta %>% tabyl(is_complete)
```

is_complete	n	percent
FALSE	206	0.103
TRUE	1794	0.897

- Is there any book with exceptionally high percentage of missings? *–No*

```
reviews_meta %>%
  tabyl(book, is_complete) %>%
  adorn_percentages() %>%
  adorn_rounding(2) %>%
  adorn_title()
```

	is_complete	
book	FALSE	TRUE
Fundamentals of R For Beginners	0.11	0.89
R For Dummies	0.12	0.88
R Made Easy	0.1	0.9
Secrets Of R For Advanced Students	0.11	0.89
Top 10 Mistakes R Beginners Make	0.08	0.92

- Is there any state with exceptionally high percentage of missings? *–No*

```
reviews_meta %>%
  select(-review) %>%
  dplyr::mutate_if(is.numeric, as.character) %>%
  pivot_longer(-c(id, is_complete)) %>%
  tabyl(value, is_complete, name, show_missing_levels = FALSE) %>%
  adorn_percentages() %>%
  adorn_rounding(2) %>%
  adorn_title() %>%
  kbl() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                full_width = F, font_size = 14, position = "left")
```

We can also calculate summary statistics such as the average, minimum and maximum proportion of complete rows per variable value. e.g.: are there differences in the proportion of missings within the categories of the ***books*** *column (aka for different books)?*

In the table below we can see that the missing data is not more prevalent in some books, some states or price categories than in others.

```
reviews_meta %>%
  select(-review, id) %>%
  mutate_if(is.numeric, as.character) %>%
  pivot_longer(-c(id, is_complete), names_to = "column") %>%
  group_by(column, value) %>%
  summarise(prop_complete = round((sum(is_complete) / n()), 3),
            n = n()) %>%
  summarise(avg_prop_complete = round(mean(prop_complete), 3),
            min = min(prop_complete),
            max = max(prop_complete)
            ) %>%
```

	is_complete	
value	FALSE	TRUE
Fundamentals of R For Beginners	0.11	0.89
R For Dummies	0.12	0.88
R Made Easy	0.1	0.9
Secrets Of R For Advanced Students	0.11	0.89
Top 10 Mistakes R Beginners Make	0.08	0.92

	is_complete	
value	FALSE	TRUE
15.99	0.12	0.88
19.99	0.1	0.9
29.99	0.08	0.92
39.99	0.11	0.89
50	0.11	0.89

	is_complete	
value	FALSE	TRUE
CA	0.11	0.89
California	0.1	0.9
FL	0.09	0.91
Florida	0.1	0.9
New York	0.08	0.92
NY	0.1	0.9
Texas	0.12	0.88
TX	0.13	0.87

```
  flextable() %>%
  theme_alafoli() %>%
  autofit()
```

column	avg_prop_complete	min	max
book	0.897	0.880	0.922
price	0.897	0.880	0.922
state	0.897	0.874	0.919

Dealing With Inconsistent Labels

Now that we've removed all of the missing data from the dataset, we have a complete dataset. This is the ideal case that we would like to start any data analysis, so we're working towards a better dataset already.

The next thing that we need to work on is the state column. You may have noticed that the labeling for each state is inconsistent. For example, California is written as both "California" and "CA". Both "California" and "CA" refer to the same place in the United States, so we should try to clean this up. We need to choose one of the ways to refer to the state, and stick to that convention. Making labels/strings more consistent in the data will make things easier to analyze later on, so we'll handle this issue with this screen.

If we're unfamiliar with the shortened postal codes of the states, we can refer to this helpful guide here. It would be helpful to write down the relevant states in our RMarkdown for reference later if we forget, so we don't have to refer back to the site itself.

Instructions

1. What are all the states that are present in the dataset?

- Look back to what we've logged about the state column to see what we have here.

```
reviews_meta %>% tabyl(state) %>%
  adorn_pct_formatting() %>%
  flextable() %>%
  theme_alafoli()
```

state	n	percent
CA	262	13.1%
California	256	12.8%
FL	248	12.4%
Florida	201	10.1%
New York	272	13.6%
NY	259	13.0%
Texas	271	13.6%
TX	231	11.6%

2. With this information in mind, choose a convention to stick to:

- either use the full name like "California" or the postal code like "CA".
- Once we decide on the convention, create a new column using the mutate() function based on the state column that makes the labeling more consistent.

transform state We want it both ways :) Once as a short column where the information is stored in the form of the code of the state `state_code` (can be useful to join more data from other sources). But We also want one column that stored the information in a informative, readable way for tables and figures, hence we also create `state_name`.

```
reviews_state <- reviews_meta %>%
  mutate(state_code = case_when(state == "California" ~ "CA",
                                state == "Florida" ~ "FL",
                                state == "New York" ~ "NY",
                                state == "Texas" ~ "TX",
                                TRUE ~ state),
         state_code = as.factor(state_code),
         state_name = case_when(state == "CA" ~ "California",
                                state == "FL" ~ "Florida",
                                state == "NY" ~ "New York",
                                state == "TX" ~ "Texas",
                                TRUE ~ state),
         state_name = as.factor(state_name)) %>%
  select(id, state_name, state_code)

reviews_state %>%
  tabyl(state_code, state_name) %>%
  flextable() %>%
  theme_alafoli()
```

state_code	California	Florida	New York	Texas
CA	518	0	0	0
FL	0	449	0	0
NY	0	0	531	0
TX	0	0	0	502

Transforming The Review Data

Instructions

The first things we'll handle in the dataset are the reviews themselves. You may have noticed in our data exploration that the reviews take the form of strings, ranging from "Poor" to "Excellent". Our goal is to evaluate the ratings of each of the textbooks, but there's not much we can do with text versions of the review scores. It would be better if we were to convert the reviews into a numerical form.

1. transform review

review_num

- Using the mutate() function create a new column in the dataset called review_num. It should take the original review column and convert it into a numerical form. The column should be coded as following:
  - The case_when() function might be useful here since we know how each of the -iews should be reclassified into numbers.
  - "Poor" should receive a numerical score of 1
  - "Fair" should receive a numerical score of 2
  - "Good" should receive a numerical score of 3
  - "Great" should receive a numerical score of 4
  - "Excellent" should receive a numerical score of 5

is_high_review It would also be helpful to have another column that helps us decide if a score is "high" or not.
- For the sake of this exercise, let's decide that a score of 4 or higher - qualifies as a "high" score.
- Create a new column in the dataset called is_high_review that denotes whether or not the review has a high score or not. In other words, it should be TRUE if review_num is 4 or higher, and FALSE otherwise.

```
reviews_review <- reviews_meta %>%
  mutate(review_num = case_when(review == "Poor" ~ 1,
                                review == "Fair" ~ 2,
                                review == "Good" ~ 3,
                                review == "Great" ~ 4,
                                review == "Excellent" ~ 5,
                                is.na(review) ~ NA_real_,
                                TRUE ~ 99),
         is_high_review = (review_num >= 4),
         ) %>%
  select(id, review, review_num, is_high_review)

reviews_review %>%
  tabyl(review_num, is_high_review) %>%
  flextable() %>%
  add_header_row(
    values = c("", "Review Classification High:"),
    colwidths = c(1, 3)
    ) %>%
  theme_alafoli() %>%
  align(part = "header", i = 1, align = "center")
```

	Review Classification High:		
review_num	FALSE	TRUE	NA_
1	368	0	0
2	369	0	0
3	363	0	0
4	0	349	0
5	0	345	0
	0	0	206

Once we are done cleaning the variables of interest we will assemble them in one set

```
reviews_clean <- reviews_meta %>%
  select(id, is_complete, book, price) %>%
    left_join(reviews_state, by = "id") %>%
    left_join(reviews_review, by = "id")
```

Analyzing The Data

Instructions

It's important to keep the overall goal in mind as we handle all the little details of the cleaning. We are acting as an analyst trying to figure out which books are the most profitable for the company. The initial data wasn't in a form that was ready for analysis, so we needed to do this cleaning to prepare it. A lot of analysts starting their first jobs believe that the analysis part of their job will be the bulk of their work. To the contrary, a lot of our work will focus on data cleaning itself, while by comparison the data analysis might only take a few lines.

With all of our data cleaning, now we're ready to do some analysis of the data. Our main goal is to figure out what book is the most profitable. How will we judge what the "most profitable" book is though? Our dataset represents customer purchases. One way to define "most profitable" might be to just choose the book that's purchased the most. Another way to define it would be to see how much money each book generates overall.

1. define most_profitable

Choose a metric that we will define "most profitable" book.

- Whichever way we choose, we should write down a few notes to ourself to justify our decision and make it clear which method we chose.
- One definition of profitable may use the price column, so we can see how much money a book generated. We may also prefer to count the number of books purchased since it could be interpreted as how popular the book is.

- `profit_rank` We will rank books based on the profit they generated. We measure profit as the total of all book prices.
- `purchased_rank` We will rank books based on purchased. We measure purchased as the number of times the book was sold.
- `review_rank` We will rank books based on reviews. We measure ratings as the average review for one book. We also consider the share of high reviews as potential indicator.
- `overall_rank` from all measures we will build an overall rank. Here, profit is weighted double for two reasons. One, after all a business needs to generate profits and secondly, the variance in the different rankings is highest in the profit ranking which we want to be reflected in the final rank. This means that in the profit ranking, rank one generates more than three times the total profits on rank five while in the review ranking, rank one is not as far away from rank five (3.05 vs 2.83 and 36.6 vs 31.2). I could generate z-scores to make the different tankings comarable but we decided to will keep it simple.

2. Summarize Data

For each book, calculate the chosen metric that we chose to measure "most profitable" book from the data.

```
profitable_book <- reviews_clean %>%
  group_by(book) %>%
  summarise(
    n_purchased = n(),
    avg_price = mean(price, na.rm = TRUE),
```

```
    total_profit = round(sum(price, na.rm = TRUE)),
    avg_review = round(mean(review_num, na.rm = TRUE),2),
    n_high_review = sum(is_high_review, na.rm = TRUE),
    n_complete = sum(is_complete),
    pct_high_review = round(n_high_review / n_complete * 100, 2)
    ) %>%
  mutate(prop_complete = n_complete/n_purchased,
         purchased_rank = rank(-n_purchased),
         profit_rank = rank(-total_profit),
         review_rank = rank(-avg_review)) %>%
  arrange(-total_profit) %>%
  mutate(overall_rank = ((purchased_rank + profit_rank * 2 + review_rank) / 4),
         overall_rank = rank(overall_rank))

profitable_book_by_state <- reviews_clean %>%
  group_by(book, state_name) %>%
  summarise(
    n_purchased = n(),
    avg_price = mean(price, na.rm = TRUE),
    total_profit = round(sum(price, na.rm = TRUE)),
    avg_review = round(mean(review_num, na.rm = TRUE),2),
    n_high_review = sum(is_high_review, na.rm = TRUE),
    pct_high_review = round(n_high_review / n() * 100, 2)
    ) %>%
  mutate(purchased_rank = rank(-n_purchased),
         profit_rank = rank(-total_profit),
         review_rank = rank(-avg_review)) %>%
  arrange(-total_profit) %>%
  mutate(overall_rank = ((purchased_rank + profit_rank * 2 + review_rank) / 4),
         overall_rank = rank(overall_rank))
```

3. Communicate Findings

Investigate the results of our analysis and write out some notes about what books are the most profitable.

Reporting the results

After performing the cleaning and analysis, we might think that that's the end. Throughout the guided project, we've just advised that we write some notes to ourself about exploring the data, cleaning it and analyzing it. Remember that we are performing this analysis for the benefit of the company that hired we, so they'll most likely be expecting some form of report from we. They're not going to do the analysis themselves, so we'll need to be able to explain everything that we did in our report and our findings.

Writing a report or creating some polished product after the analysis is important because we need to be able to communicate our findings to others. This is especially important when the people reading our analysis are not programmers themselves and will not be able to understand the code that we wrote for it. Take some time to organize our notes into a small report. This report doesn't need to be a long essay! It only need to communicate the answer to our company's question: "What's our most profitable book?" There are several sub-questions that might need to be answered along with this question like, "How do we know it's the most profitable?" or "How did we calculate our measure for profitability"? Your report is just trying to demonstrate that we have an answer and that we gave some thought on how to get to this answer.

Aside from our hypothetical situation, writing a good report can help show future employers that we are able to think both programmatically and communicate well. For any future analysis we might do, say in another Guided Project, having a polished final product that can be easily read will always be looked up favorably.

Instructions

1. Structure

A good, common way to structure a report is into three parts: Introduction, Findings, Conclusion

2. Summarize the notes we've made into polished paragraphs for each section. As a guideline, we can try to answer each of the questions below for each section:

- Introduction: What motivated our analysis? What kind of data do we have? What is the main question we're trying to answer?
- Findings: What did we need to do to the data to do our analysis? What things are we calculating to answer our main question?
- Conclusion: What is the answer to our main question? Was there anything that we feel limits our analysis? What should the reader do with our findings?

Report

Introduction

In this analysis we wanted to compare the sales of books to answer the question: "What's our most profitable book?". We defined three ways to define profitable book.

A book if profitable if

1. it generates high profit margins (profit)
2. it gets good customer review ratings (quality)
3. it sells many times (quantity)

Based on those three metrics we ranked the books, whereby the profit dimension is weighted double for two reasons. One, generated profits are the most important metric since a business needs to generate profits and secondly, the variance in the different rankings is highest in the profit metric, meaning that this is the dimension with the highest differences between books.

Findings

As a result of our analysis, we find that the book "Fundamentals of R For Beginners" is the most profitable book in terms of our metrics profit, quality and quantity.

```
profitable_book %>%
  select(book, contains("rank")) %>%
  arrange(overall_rank) %>%
  flextable() %>%
  set_header_labels(
    values = list(book = "Book",
         purchased_rank = "Purchase Volume",
```

```
        profit_rank = "Profit",
        review_rank = "Popularity",
        overall_rank = "Overall")
    ) %>%
  add_header_row(
    values = c("", "Ranking"),
    colwidths = c(1, 4)
    ) %>%
  theme_alafoli() %>%
  autofit() %>%
  align(part = "header", i = 1, align = "center") %>%
  bold(j = "overall_rank", bold = TRUE, part = "all") %>%
  vline(j = "book", part = "body") %>%
  bg(i = 1, bg = "#fceef5")
```

		Ranking		
Book	Purchase Volume	Profit	Popularity	**Overall**
Fundamentals of R For Beginners	1.5	2	2	**1**
Secrets Of R For Advanced Students	3.0	1	4	**2**
Top 10 Mistakes R Beginners Make	5.0	3	1	**3**
R Made Easy	4.0	4	3	**4**
R For Dummies	1.5	5	5	**5**

It generated the second highest profit and customer reviews and was in the top rank with regard to purchasing volume.

```
profitable_book %>%
  arrange(overall_rank) %>%
  select(book,
         n_purchased,
         total_profit,
         pct_high_review,
         avg_price) %>%
  flextable() %>%
  set_header_labels(
    values = list(book = "Book",
         n_purchased = "Purchase Volume",
         total_profit = "Profit",
         pct_high_review = "Share of high reviews",
         avg_price = "Price")
    ) %>%
  theme_alafoli() %>%
  autofit() %>%
  vline(j = "book", part = "body") %>%
  bg(i = 1, bg = "#fceef5")
```

Book	Purchase Volume	Profit	Share of high reviews	Price
Fundamentals of R For Beginners	410	16,396	41.26	39.99
Secrets Of R For Advanced Students	406	20,300	37.50	50.00

Book	Purchase Volume	Profit	Share of high reviews	Price
Top 10 Mistakes R Beginners Make	385	11,546	39.72	29.99
R Made Easy	389	7,776	39.49	19.99
R For Dummies	410	6,556	35.46	15.99

Comparing the overall winner to the most profitable book

The book that stands at the top in terms of our overall ratings is not the book that generated the highest profits. Why are we considering other measures as well? We are considering customer reviews, since we assume that having a book with a high percentage of good consumer reviews and many copies sold is more profitable in the long run, since happy customers are more likely to come back and purchase other books as well.

"Fundamentals of R For Beginners" (the top rated book) generated 16,396$ compared to the book with highest profit "Secrets Of R For Advanced Students" which generated 20,300$. This difference appears because both books sold about the same number of copies, while "Fundamentals of R For Beginners" is in a cheaper price range with 39,99$ compared to 50,00$ for "Secrets Of R For Advanced Students". And while higher priced books generate more profit the more copies are sold, not all books in the store should be expensive. The book store benefits from a diverse set of price ranges, because it needs to compete with the prices of other book stores and also wants to offer books for a broad range of customers with different income levels. This is supported by the fact that the second book with the highest number of purchases "R for Dummies" is in the lowest price range of 15,99$.

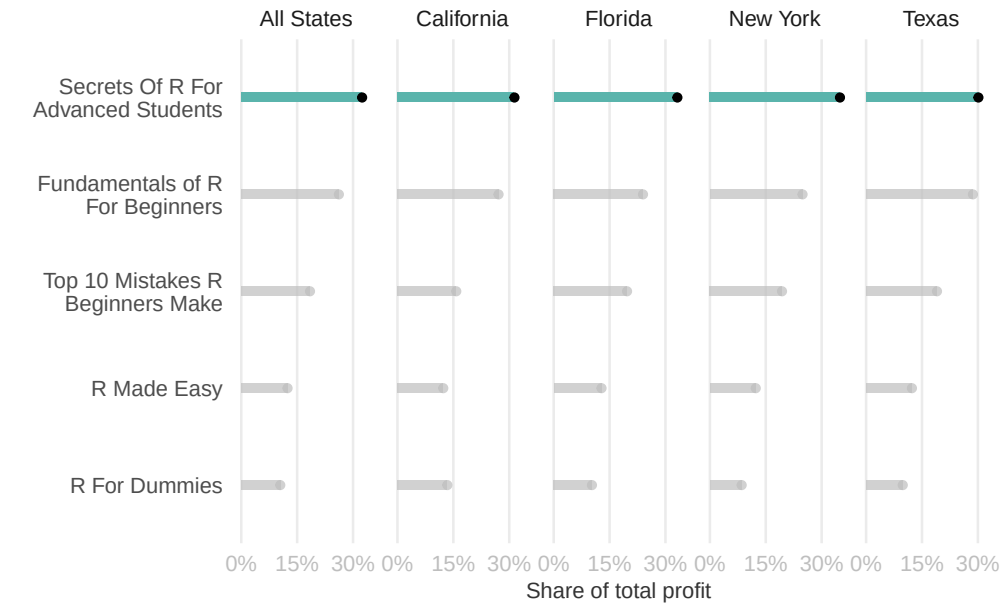
What about local differences? Are there books that are big sellers only in one state?

To go one step further, we analyzed whether the profits generated by the books show large differences between the different states. The following figure helps to see which books contributed most to the overall profits within one state and whether this share differs across states.

We can see that the books seem to sell equally well across all locations and their share of the overall generated profits remains constant across states.

```
profitable_book_by_state %>%
  bind_rows(profitable_book %>%
              mutate(state_name = "All States")
            ) %>%
  group_by(state_name, book) %>%
  summarise(total_profit_book = sum(total_profit),
            total_sales_book = sum(n_purchased)) %>%
  dplyr::add_count(state_name, wt = total_profit_book, name = "total_profit_state") %>%
  mutate(prop = total_profit_book / total_profit_state,
         book = str_wrap(book, 20),
         highlight = state_name == "All States") %>%
  mutate(highest_prop = prop == max(prop)) %>%
  ungroup() %>%
  ggplot(
    aes(x = fct_reorder(book, prop),
        y = prop,
        fill = highest_prop)
    ) +
  geom_col(width = 0.1) +
  geom_point() +
```

```
  gghighlight::gghighlight(
    max(prop),
    max_highlight = 1L,
    use_direct_label = FALSE,
    calculate_per_facet = TRUE) +
  coord_flip() +
  theme_minimal(base_size = 12) +
  #scale_y_continuous(labels = scales::percent) +
  scale_y_continuous(breaks = c(0, 0.15, 0.30),
                     labels = scales::percent) +
  scale_fill_manual(values = c("#5ab4ac", "#5ab4ac20")) +
  facet_wrap(~ state_name, nrow = 1) +
  labs(title = "Books show similar shares of profit across states",
       subtitle = "Composition of profits by book and states, highest share is highlighted",
       x = "",
       y = "Share of total profit") +
  theme(legend.position =  "none",
        #legend.justification = "left",
        plot.title.position = "plot",
        axis.title.x = element_text(colour = "grey20", size = 10),
        axis.text.x = element_text(colour = "grey"),
        panel.grid.minor = element_blank(),
        panel.grid.major.y = element_blank())
```



Conclusion

In summary we found that while "Secrets Of R For Advanced Students" is the book that generates the highest profits overall and in each state, there are some reasons to believe that overall, the book "Fundamentals of R For Beginners" can be seen as the most profitable book because it generated profits but also ranked high in terms of the amount of copies sold and customer satisfaction.

Further Steps

Next the to data that is available right now, it could be interesting to see how the prices and purchase numbers of individual books changed over time. Analysis like these could provide us with insight to the question of what prices the book stores customers can afford and what the relationship between price, popularity and number of copies sold. One could also go about this analyis in a more qualitative way and talk to customers to find out what made them come to this specific store or what the different reasons for buying a book can be. Based on the information from customers, we might find out about new ways in which we can help them to find the books they want or help them to find books for their friends or relatives. Making sure that su