Great reads and ratings with Goodreads

Analyzing Goodreads for what affects a book's reach (# of ratings) and average rating.

For: CS145 (Data Systems and Data Management)'s final open-ended project

Why: To connect my interests in computation with literary metadata analysis

By: Shana Hadi, @shanaeh

Setting up BigQuery and Dependencies

Run the cell below (shift + enter) to authenticate your project.

Note that you need to fill in the `project_id` variable with the Google Cloud project id you are using for this course. You can see your project ID by going to https://console.cloud.google.com/cloud-resource-manager

```
# Run this cell to authenticate yourself to BigQuery.
from google.colab import auth
auth.authenticate_user()
project_id = "cs145-project3-books"
```

```
#All our fun python libraries
import io, os, copy, math, random, csv
import numpy as np
import scipy.stats as stats
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
```

Project Overview: Analyzing Books with High Reach (# of Ratings)

Goodreads, a site for cataloguing, reading, and recommending books, crowdsources ratings, tags, and reviews from users. Authors can engage with and respond to readers, while readers can organize and reflect on their books.

I have noticed that the star ratings tend to skew towards the upper end, suggesting that readers are more likely to rate and review books that they have already read and view favorably. Rather than ask what correlates with a visibly skewed higher average rating, I am interested in what correlates with a work's higher number of ratings, which I will use as a proxy for a higher "reach" of readers.

Analysis of Dataset: Top 10K Most Rated Goodreads Books

I will use this dataset, which has the top 10,000 most rated books and almost 600 million (596,873,216) total ratings, https://www.kaggle.com/zygmunt/goodbooks-10k. This is valid up to late 2017 (since then, the rankings have changed, and new books have entered the site). I chose this set since it consolidates different versions of books into one "work" and its ratings; it also distinguishes among 1-5 star ratings. I also separately created 1 table of authors, with their name, average_ratings across works, and ratings_count.

In sum, this dataset has 6 tables, totalling 255 MB.

1. top10k_all_books (23 columns: includes 6 ids with isbn and isbn13 for validating data across multiple sets, versions_count, authors, original_publication_year, original_title, title, language_code, average_rating, ratings_count, work_ratings_count, work_text_reviews_count, ratings_1star, ratings_2star, ratings_3star, ratings_4star, and ratings_5star, and 2 image_urls for cross-checking)
2. to_read (2 columns: user_id, work_id)
3. book_tags (3 columns: goodreads_book_id, tag_id, count)
4. string_tags (2 columns: tag_id, tag_name)
5. goodreads_book_authors (5 columns: name, average_rating, author_id, text_reviews_count, ratings_count)
6. user_work_rating (3 columns: user_id, work_id, rating) Primarily used to self-validate the data, especially for the book tags that are haphazardly organized with multiple ids for the same tag

---

The primary table is top10k_all_books, which has the books and their multiple ids (the keys), along with attributes such as their various ratings and versions. For counting # of ratings, I use work_ratings_count since this aggregates counts across all versions of a book (e.g. The Hunger Games, which has been published many times).

From top10k_all_books, I can join the work_id with the to_reads table to count the number of users who shelved/tagged the work as "to read."

The tables book_tags and string_tags can be joined with tag_id, and I can use these tables to join with the primary table to relate tags to books (which tags have the most books, and then aggregate the # of ratings). These tags can also be used to approximate fuzzy genre categories (books will have multiple tags).

I can use user_work_rating to validate the user_ids in table 2, and for counting the number of ratings overall to make sure it matches based on author name (not ideal, but it was all that was available in the set)

Basic Ratings Stats

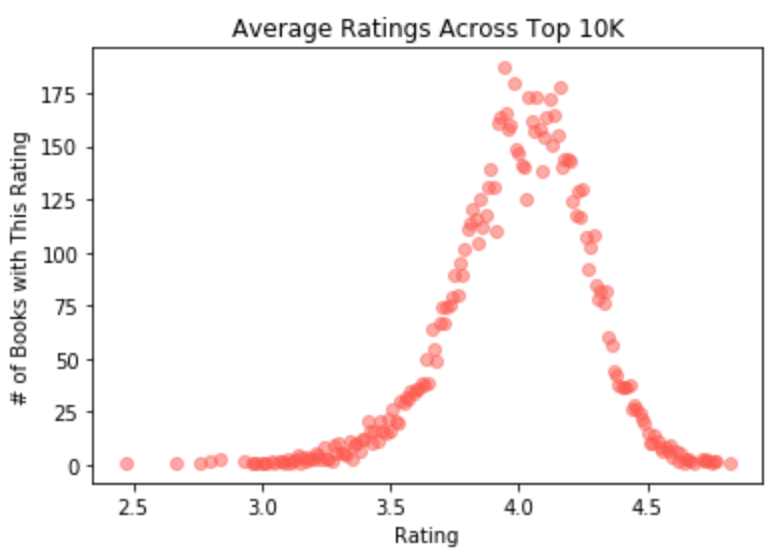
To establish a grounding in the data, I visualized the average ratings, # of ratings per star type, and the top 10 most rated books.

Since my project centers on the question of which books maximize a high # of ratings, I thought the former would show the relative skew of ratings (sites like Rotten Tomatoes are more likely to have a normal distribution centered aorund 3), while latter would give insight on what types of books have a high reach on Goodreads (which may not be necessarily applicable to the larger reading world).

Average Ratings (Clustered on Mean 4.03)

```
%%bigquery --project $project_id avg_ratings
SELECT COUNT(book_id) AS count, average_rating
FROM `goodreads-top10k.all_books.metadata`
GROUP BY average_rating
```

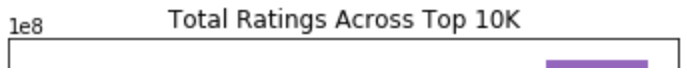
```
%matplotlib inline

data = avg_ratings
ax = plt.gca() #gca means get current axes
ax.scatter(data["average_rating"], data["count"], color="#fc5a50", alpha=0.5)

plt.title("Average Ratings Across Top 10K")
plt.xlabel("Rating")
plt.ylabel("# of Books with This Rating")
plt.show()
```



# of Ratings Per Star 1-5

```
%%bigquery --project $project_id ratings

SELECT SUM(ratings_1star) AS one, SUM(ratings_2star) AS two, SUM(ratings_3star) AS three, SUM(ratings
FROM `goodreads-top10k.all_books.metadata`
```

```
%matplotlib inline

plt.title("Total Ratings Across Top 10K")
plt.bar("One Star", ratings["one"])
plt.bar("Two Stars", ratings["two"])
plt.bar("Three Stars", ratings["three"])
plt.bar("Four Stars", ratings["four"])
plt.bar("Five Stars", ratings["five"])
plt.xlabel("Rating")
plt.ylabel("# of Ratings, in 100k increments")
plt.show()
```



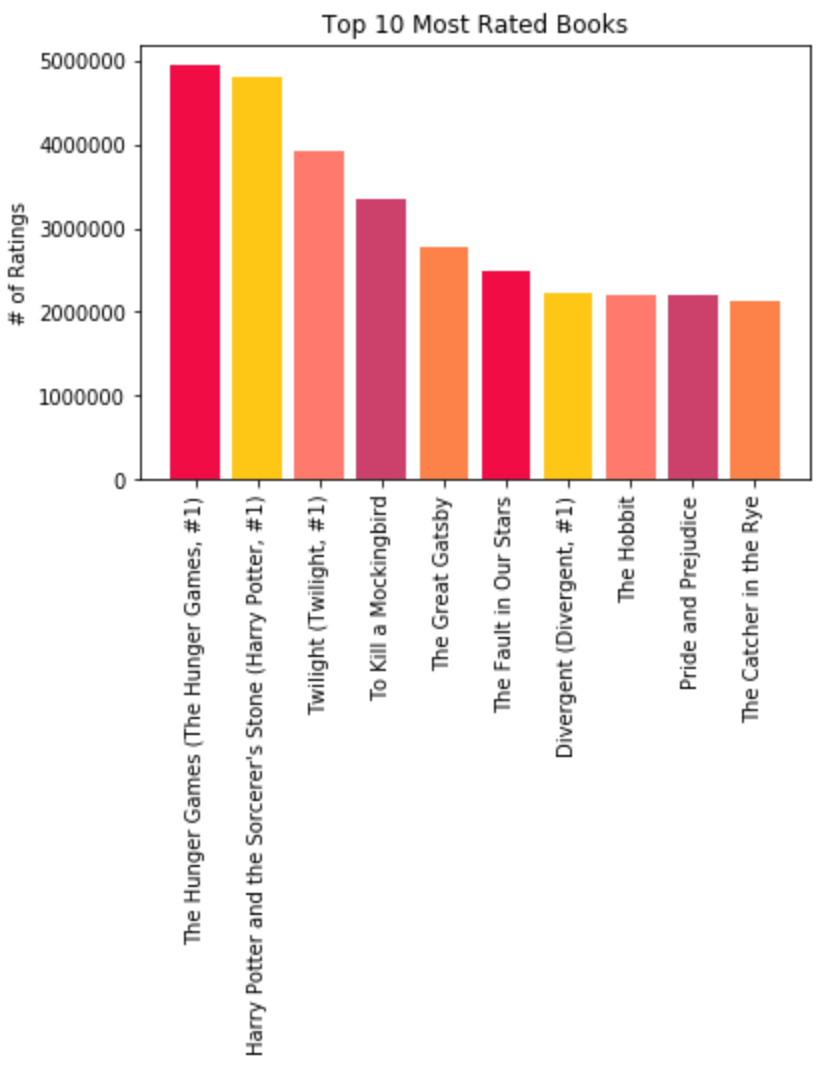
Top 10 Most Rated Books

```
%%bigquery --project $project_id top10
SELECT title, work_ratings_count
FROM `goodreads-top10k.all_books.metadata`
ORDER BY work_ratings_count DESC
LIMIT 10
```

```
%%bigquery --project $project_id
SELECT title, average_rating, work_ratings_count
FROM `goodreads-top10k.all_books.metadata`
ORDER BY work_ratings_count DESC
LIMIT 10
```

	title	average_rating	work_ratings_count
0	The Hunger Games (The Hunger Games, #1)	4.34	4942365
1	Harry Potter and the Sorcerer's Stone (Harry P...	4.44	4800065
2	Twilight (Twilight, #1)	3.57	3916824
3	To Kill a Mockingbird	4.25	3340896
4	The Great Gatsby	3.89	2773745
5	The Fault in Our Stars	4.26	2478609
6	Divergent (Divergent, #1)	4.24	2216814
7	The Hobbit	4.25	2196809
8	Pride and Prejudice	4.24	2191465
9	The Catcher in the Rye	3.79	2120637

```
%matplotlib inline

plt.title("Top 10 Most Rated Books")
plt.bar(top10["title"], top10["work_ratings_count"], color=["#f10c45", "#fec615", "#ff796c", "#cb416b
plt.xticks(top10["title"],  rotation='90')
plt.ylabel("# of Ratings")
plt.show()
```



Data Exploration: # of Ratings and More!

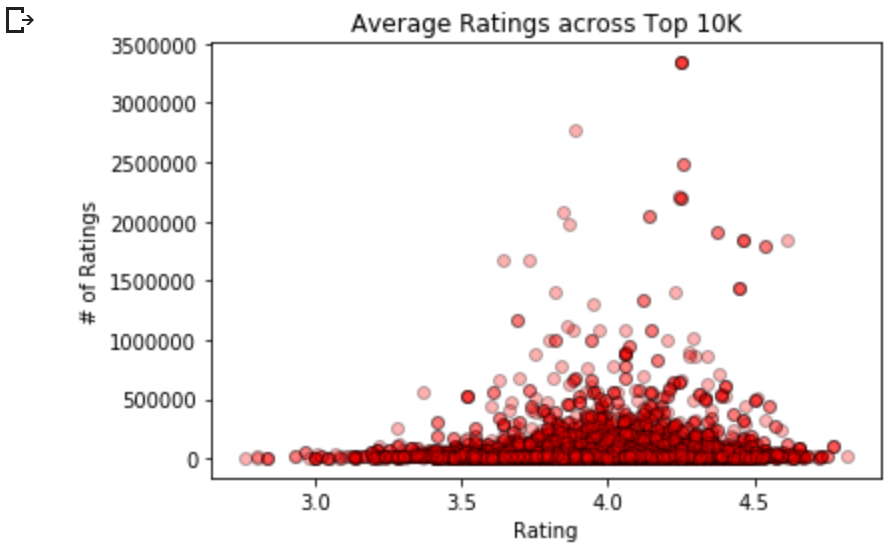
I separated this section into 3 "Stages" of inquiry; the latter sections have multipart visualizations.

Stage 1: Quickly Quantifiable Queries (QQQ)

Feature 1: Average Ratings

```
%%bigquery --project $project_id avg_ratings

SELECT average_rating, work_ratings_count
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books`
ORDER BY average_rating DESC
```

```
%matplotlib inline

data = avg_ratings.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["average_rating"], data["work_ratings_count"], color = "red", alpha = 0.3, edgecolors
plt.title("Average Ratings across Top 10K")
plt.xlabel("Rating")
plt.ylabel("# of Ratings")
plt.show()
```



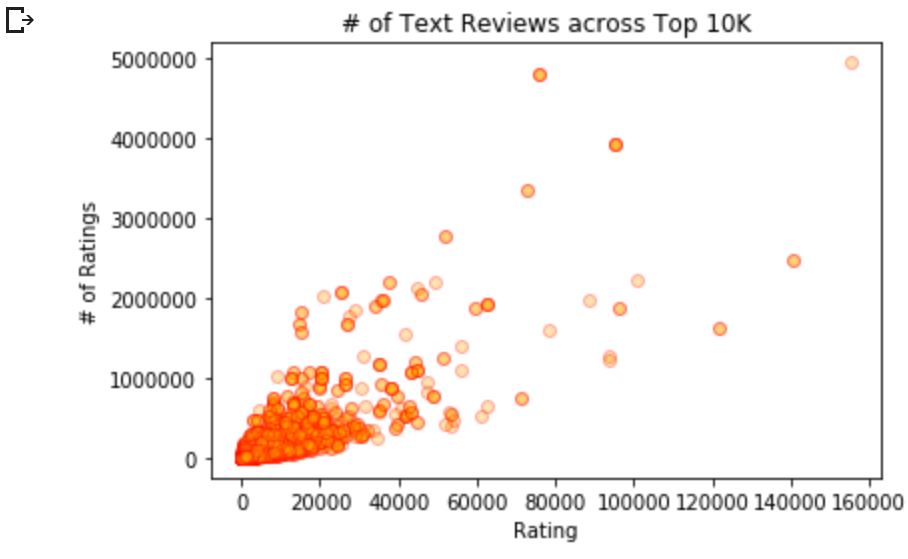
Goodreads average ratings are clustered around 4.03, but there is a possible correlation in that the works with higher reach tend to peak with scores of 4.0 and higher (between [4.0,4.5] seems best, particularly around 4.2). Here, # of ratings can get into the 2 million range, with a few higher peaks.

From the grounding charts in the previous section, we can see that for the top 10 books, 7 of them have an average_rating near 4.2, which supports this analysis.

Feature 2: # of Text Reviews

```
%%bigquery --project $project_id text_reviews_count

SELECT work_text_reviews_count, work_ratings_count
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books`
ORDER BY average_rating DESC
```

```
%matplotlib inline

data = text_reviews_count.sample(15000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["work_text_reviews_count"], data["work_ratings_count"], color = "orange", alpha = 0.3
plt.title("# of Text Reviews across Top 10K")
plt.xlabel("Rating")
plt.ylabel("# of Ratings")
plt.show()
```



As expected, there is a visibly significant correlation (can perceive an upwards trend). As more readers rate the book, they are also more likely

Stage 2: Marvelously Multiple-Col Mixed Queries (MMMQ)

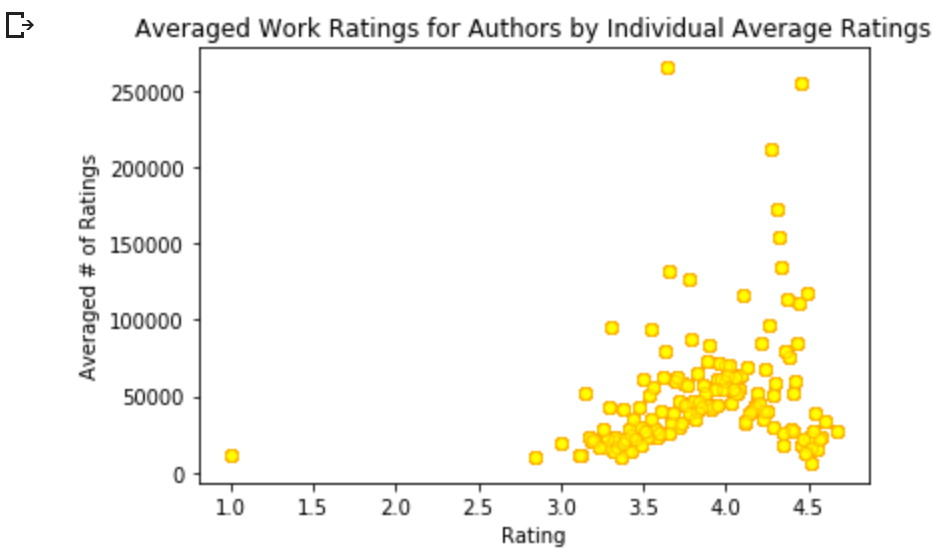
Feature 3: Authors and their Prolific Ways (4 Parts)

The name of authors can act as a "brand-name" for a book's success -- more well-known, or highly acclaimed, authors will have higher ratings, but to what degree, especially when compared with the rest of the author population?

3a: Averaged Work Ratings for Authors by Individual Average Ratings

```
%%bigquery --project $project_id authors_avg_rating

SELECT authors.average_rating, AVG(books.work_ratings_count) as work_ratings_count
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_metadata.goodreads_book_authors` authors
WHERE books.authors = authors.name
GROUP BY authors.average_rating
ORDER BY authors.average_rating DESC
```

```
%matplotlib inline

data = authors_avg_rating.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["average_rating"], data["work_ratings_count"], color = "yellow", alpha = 0.3, edgeco
plt.title("Averaged Work Ratings for Authors by Individual Average Ratings")
plt.xlabel("Rating")
plt.ylabel("Averaged # of Ratings")
plt.show()
```



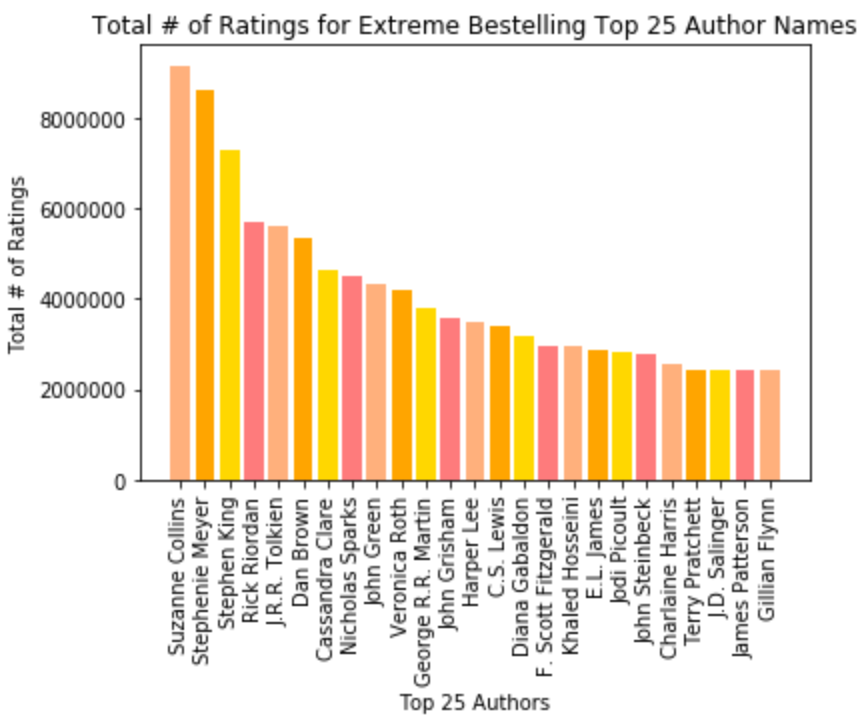
Here, we are measuring the average_rating of authors (across all of their works) and seeing how this affects the averaged # of ratings per book. Rather than selecting for the sum of ratings for all their works (which will be more biased towards prolific authors, without accounting for the success of an individual book), I hypothesized the finding the average would make it easier to predict the relative success of their works individually.

These scores seem to cluster around 4.03, which is the mean of Goodreads ratings; however, the peaks with a higher averaged # of ratings are around 4.3, suggesting that authors with high average_ratings (which means that they write works with high average_ratings) will also have works that have a higher reach.

3b: Authors by Name, Total Work Ratings (High # of Ratings, Many Books)

```
%%bigquery --project $project_id top_authors_name

SELECT authors.name, SUM(books.work_ratings_count) as sum_ratings_count
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_metadata.goodreads_book_authors` authors
WHERE books.authors = authors.name
GROUP BY authors.name
ORDER BY SUM(books.work_ratings_count) DESC
LIMIT 25
```

```
%matplotlib inline

plt.title("Total # of Ratings for Extreme Bestelling Top 25 Author Names")
plt.bar(top_authors_name["name"], top_authors_name["sum_ratings_count"], color=['#ffb07c', 'orange',
plt.xlabel("Top 25 Authors")
plt.xticks(top_authors_name["name"],  rotation='vertical')
plt.ylabel("Total # of Ratings")
plt.show()
```



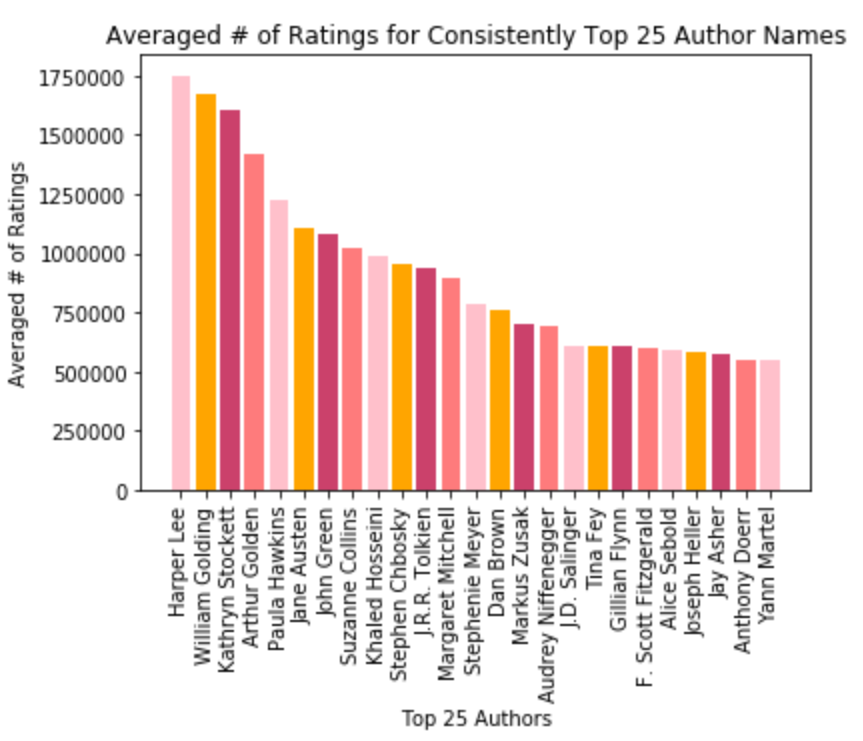
We are selecting for 25 authors with the highest total # of ratings (which means extreme bestsellers, or a high number of bestsellers). In this list, Suzanne Collins and Stephenie Meyer are the top 2, which fits in line with the Top 10 most rated books. J.K. Rowling may be excluded as her series may have multiple versions (US and UK) that were not consolidated within this dataset, especially as her bestsellers (e.g. Harry Potter #1) came out before Goodreads first launched. However, as J.R.R. Tolkien also makes this list, then this might not be as significant a hypothesis.

Nonetheless, these names correlate with books with high reach -- however, since they do not make up a majority of the data set (they are the exception, not the rule), forming an ML model based on these names alone would not be fully applicable.

3c: Authors by Name, Averaged Work Ratings (Very High # of Ratings, Fewer Books)

```
%%bigquery --project $project_id top_authors_name2

SELECT authors.name, AVG(books.work_ratings_count) as avg_ratings_count
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_metadata.goodreads_book_authors` authors
WHERE books.authors = authors.name
GROUP BY authors.name
ORDER BY AVG(books.work_ratings_count) DESC
LIMIT 25
```

```
%matplotlib inline

plt.title("Averaged # of Ratings for Consistently Top 25 Author Names")
plt.bar(top_authors_name2["name"], top_authors_name2["avg_ratings_count"], color=['pink', 'orange', '
plt.xlabel("Top 25 Authors")
plt.xticks(top_authors_name2["name"],  rotation='vertical')
plt.ylabel("Averaged # of Ratings")
plt.show()
```



Instead of looking for raw output, here we are looking for highly rated fewer books. Harper Lee, who wrote only one book, the canonized and highly acclaimed To Kill A Mockingbird (Go Set a Watchman did not make this dataset since it was only published recently), reasonably takes first place. There is some overlap with 3c, especially for books that were INCREDIBLY bestselling (such as Collins' The Hunger Games or Meyers' Twilight), but it is also more inclusive of less prolific but canonized authors.

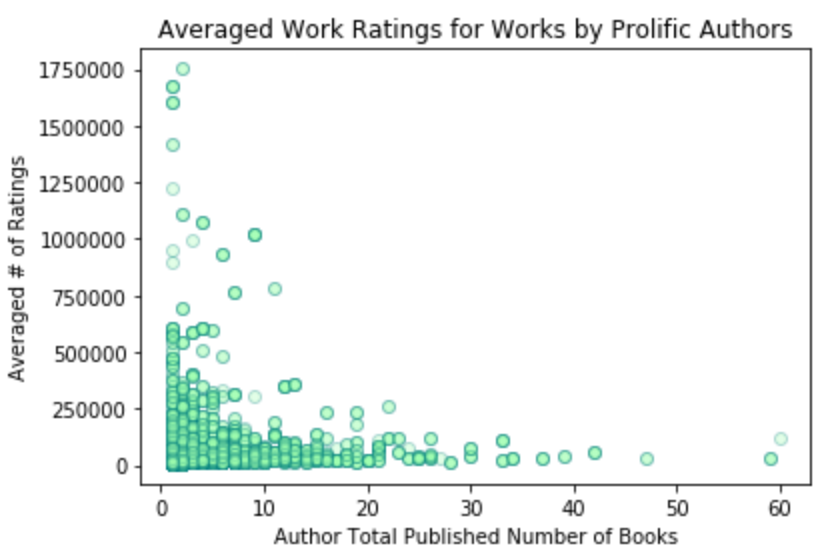
For example, Jane Austen, with six official novels and one other proto-novel, finds a lovely spot on this list, while other established literary authors like William Golding and J.D. Salinger are here. This could suggest that with our goal of a "high reach" of books, this could be linked to

what readers have exposure to in high school curriculums; they then rate these on Goodreads, even though their original publication years predate the 2000s.

3d: Averaged Work Ratings for Works by Prolific Authors

```
%%bigquery --project $project_id authors_num_books

SELECT authors.name, COUNT(work_id) as count_books, AVG(books.work_ratings_count) as avg_ratings_cour
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_metadata.goodreads_book_authors` authors
WHERE books.authors = authors.name
GROUP BY authors.name
ORDER BY COUNT(work_id) DESC
```

```
%matplotlib inline

data = authors_num_books.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["count_books"], data["avg_ratings_count"], color = "#9ffeb0", alpha = 0.3, edgecolor

plt.title("Averaged Work Ratings for Works by Prolific Authors")
plt.xlabel("Author Total Published Number of Books")
plt.ylabel("Averaged # of Ratings")
plt.show()
```



Averaged ratings for a work don't seem to get a boost even if they were written by prolific authors who have written many other works. This was contrary to my expectations, as I thought works by bestselling authors would be far higher than the others. And then with this graph, I later recalled that very few authors are bestsellers, even if they are prolific.

Nevertheless, since there are several points near the 1 million range for the range [0,10] above the rest, it's possible that for some authors, their prolific publishing (e.g. Stephen King, Agatha Christie) means that their works overall will get a high number of ratings. However, they are the exception and not the norm. (Debunked in model 1: Originally, I thought selecting for name, as in part 3b and 3c, would be more useful for ML, but most authors are not prolific enough to successfully publish multiple times).

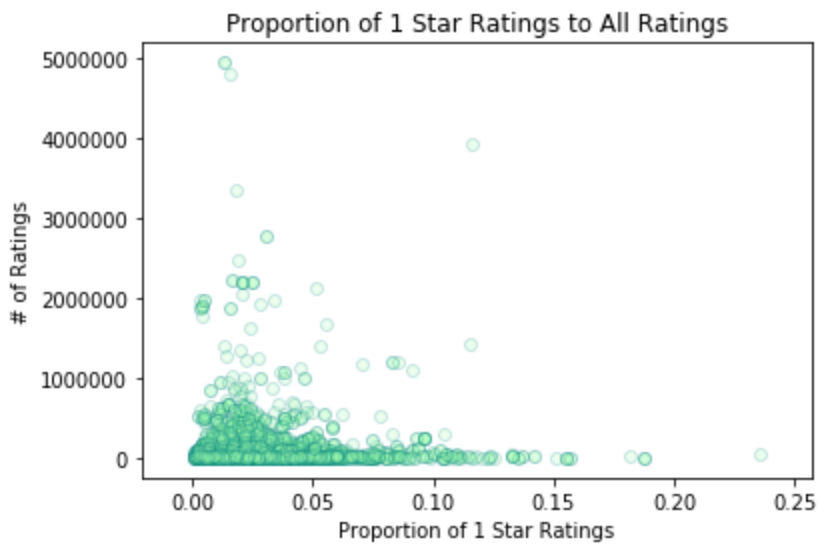
Feature 4: The Walk of Stars (5 Parts)

Rather than use the raw # of X star ratings (where X = 1,2,3,4,5), which will undoubtedly correlate with an overall high # of star ratings, I chose to use the proportions -- e.g. what fraction of the ratings are each score for the books, and does it correlate with a high # of star ratings?

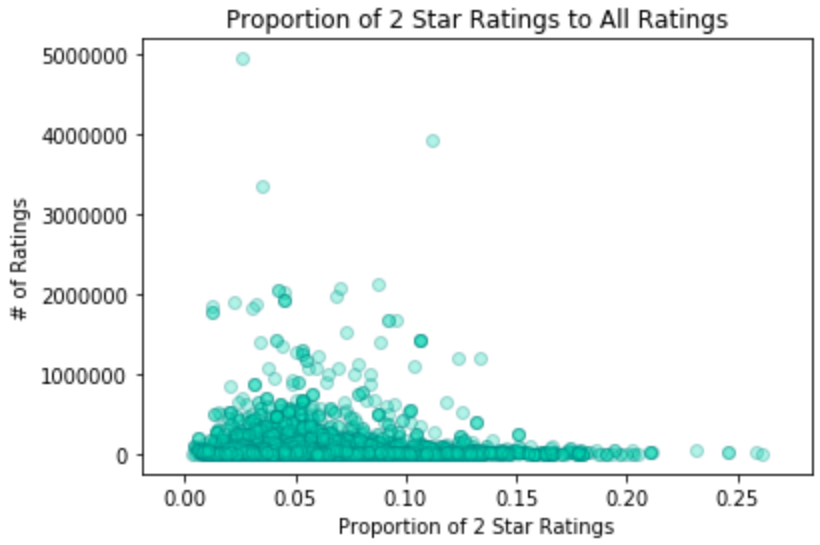
I chose to first separate them into 5 graphs for legibility, and they are combined into 1 graph at the end.

```
%%bigquery --project $project_id frac_stars

SELECT work_ratings_count, ratings_1star/work_ratings_count as frac_1star,
ratings_2star/work_ratings_count as frac_2star,
ratings_3star/work_ratings_count as frac_3star,
ratings_4star/work_ratings_count as frac_4star,
ratings_5star/work_ratings_count as frac_5star
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books`
ORDER BY average_rating DESC
```

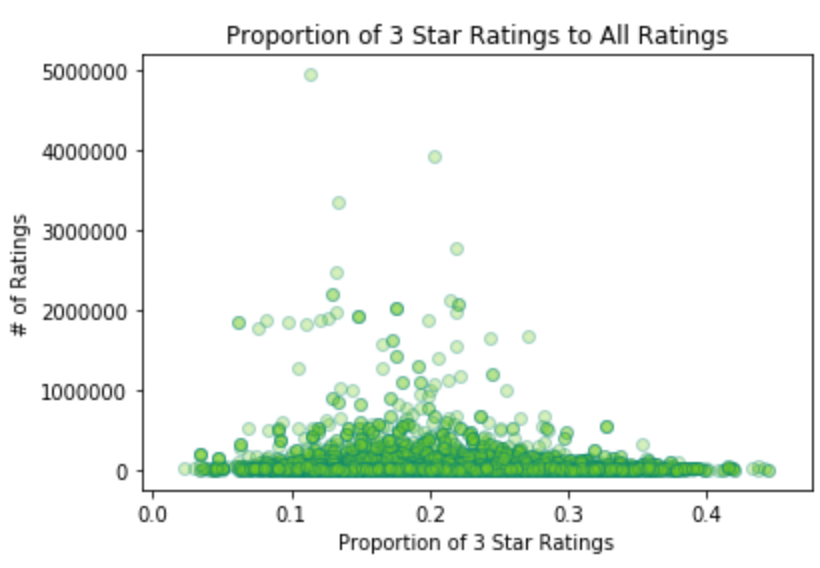
```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_1star"], data["work_ratings_count"], color = "#9ffeb0", alpha = 0.2, edgecolor

plt.title("Proportion of 1 Star Ratings to All Ratings")
plt.xlabel("Proportion of 1 Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```



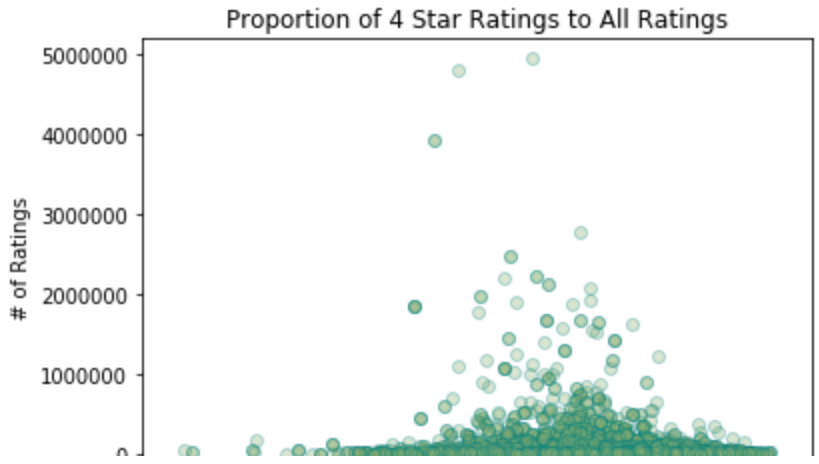

```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_2star"], data["work_ratings_count"], color = "#04d8b2", alpha = 0.3, edgecolor

plt.title("Proportion of 2 Star Ratings to All Ratings")
plt.xlabel("Proportion of 2 Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```



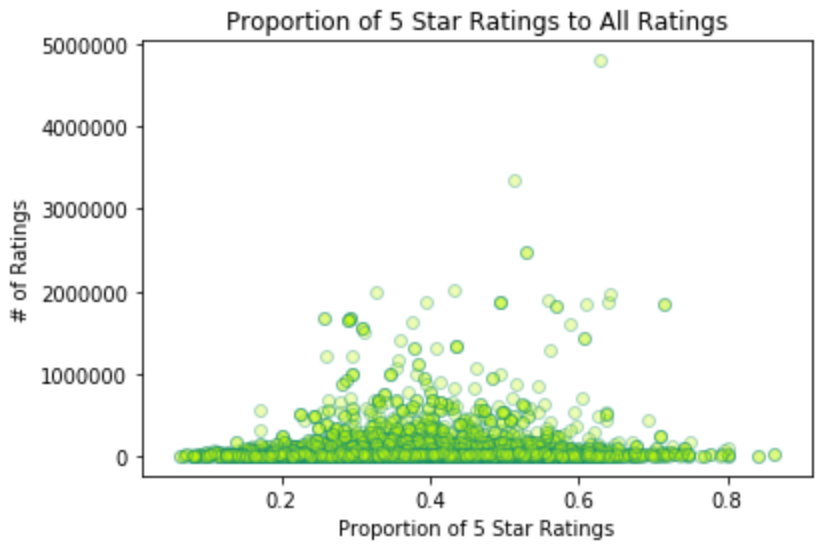

```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_3star"], data["work_ratings_count"], color = "#76cd26", alpha = 0.3, edgecolor

plt.title("Proportion of 3 Star Ratings to All Ratings")
plt.xlabel("Proportion of 3 Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```



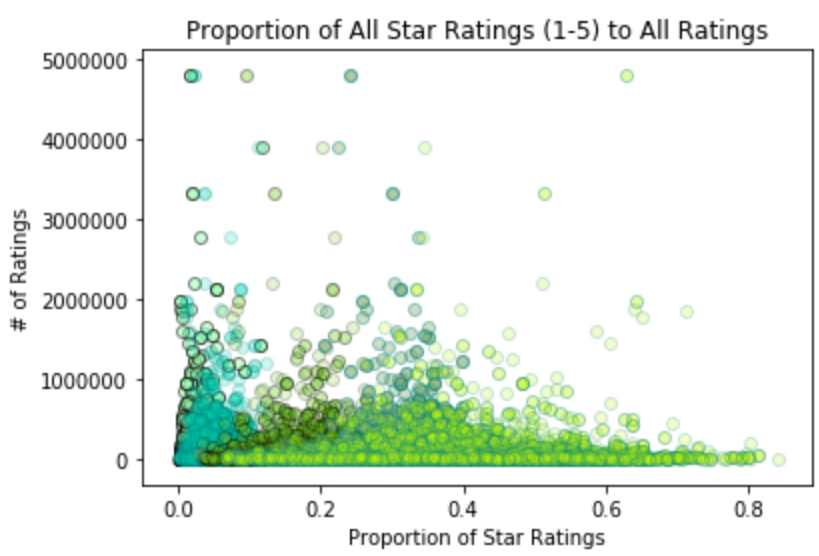

```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_4star"], data["work_ratings_count"], color = "#87ae73", alpha = 0.3, edgecolor

plt.title("Proportion of 4 Star Ratings to All Ratings")
plt.xlabel("Proportion of 4 Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```




```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_5star"], data["work_ratings_count"], color = "#c1f80a", alpha = 0.3, edgecolor

plt.title("Proportion of 5 Star Ratings to All Ratings")
plt.xlabel("Proportion of 5 Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```




```
%matplotlib inline

data = frac_stars.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["frac_1star"], data["work_ratings_count"], color = "#9ffeb0", alpha = 0.5, edgecolor
ax.scatter(data["frac_2star"], data["work_ratings_count"], color = "#04d8b2", alpha = 0.2, edgecolor
ax.scatter(data["frac_3star"], data["work_ratings_count"], color = "#76cd26", alpha = 0.2, edgecolor
ax.scatter(data["frac_4star"], data["work_ratings_count"], color = "#87ae73", alpha = 0.4, edgecolor
ax.scatter(data["frac_5star"], data["work_ratings_count"], color = "#c1f80a", alpha = 0.2, edgecolor

plt.title("Proportion of All Star Ratings (1-5) to All Ratings")
plt.xlabel("Proportion of Star Ratings")
plt.ylabel("# of Ratings")
plt.show()
```



While Goodreads skews towards higher ratings, I wondered if there were a certain star proportion that would mroe strongly correlate with a higher number of overall ratings.

4-star ratings seem more distinctive in that the spread is mostly contained within [0.2, 0.5] and have a higher bump, as compared to 3-star ratings withing [0.1, 0.5] with a lower bump, or 5-star ratings which are more widely spread out across [0.1, 0.8]. And as expected, 1-star ratings and 2-star ratings are quite low, and limited to [0, 0.2].

Perhaps this is not so distinctive, when accounting for how the average_rating for Goodreads is near 4.03, and in the top 10, 7 have average_ratings near 4.24.

Stage 3: Complex Columnally Coordinated Queries (CCCQ)

Feature 5: To Read the To-Reads and Super-User Frenzy (3 Parts)

5a) Users and their To-Reads

On Goodreads and more generally, users will tend to only rate a book after they have finished reading it. Users can save books that interest them as "to-reads," which means they anticipate eventually reading them, but have not yet read (and rated) them. Will books that are on many to-reads lists have a higher rating count, when accounting for popularity and the "reach" of this book? All books are still from the top10k.

```
%%bigquery --project $project_id to_read
SELECT books.work_ratings_count AS work_ratings_count, COUNT(user_id) as num_users
FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.to_read` to_read
WHERE books.id = to_read.work_id
GROUP BY to_read.work_id, books.work_ratings_count
ORDER BY num_users
```

```
%matplotlib inline

data = to_read.sample(10000, replace=True)
ax = plt.gca() #gca means get current axes
ax.scatter(data["num_users"], data["work_ratings_count"], color = "#0485d1", alpha = 0.3, edgecolor =

plt.title("Reach of Books Marked To-Read")
plt.xlabel("# of Users Marked Book To-Read")
plt.ylabel("# of Ratings")
plt.show()
```

There is perceivable curve upwards, suggesting that as more users mark a book to-read, the higher # of ratings the book will have. This makes sense as users are more likely to anticipate reading books that they have heard of and are widely exposed to, which in turn would be books with a high number of ratings -- a self-perpetuating cycle of popularity.

5b) "Super-to-readers" and their Listings

While the above query shows a distinctive trend, what if we subset the users to only the top 500 most active to-readers (0.01 of the 50,000), and award them the title of "Super-to-readers, deciders of taste"?

```
%%bigquery --project $project_id super_to_read

SELECT books.work_ratings_count AS work_ratings_count,
COUNT(to_read.user_id) as num_users

FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.to_read` to_read,
(SELECT DISTINCT user_id, COUNT(work_id) as num_to_reads
FROM `cs145-project3-books.goodreads_top10k.to_read` to_read
GROUP BY user_id
ORDER BY num_to_reads DESC
LIMIT 500) AS top_500_users

WHERE books.id = to_read.work_id
AND to_read.user_id = top_500_users.user_id

GROUP BY to_read.work_id, books.work_ratings_count
ORDER BY num_users
```

```
%matplotlib inline

data = super_to_read.sample(1000, replace=True)
ax = plt.gca() #gca means get current axes
```

```
 5 ax.scatter(data[ 'num_users' ], data[ 'work_ratings_count' ], color = '#065211', alpha = 0.3, edgecolor =
 6
 7 plt.title("Reach of Books Marked To-Read")
 8 plt.xlabel("# of Top 500 Users Marked Book To-Read")
 9 plt.ylabel("# of Ratings")
10 plt.show()
```

Perhaps there is a more upwards trend than 5a, but this is not as strong as my initial prediction, which was that super-to-readers (who mark more books "to-read") would tend to choose already popular books (with high # of ratings). I thought the ratings would be nearer to the millions, but many are still clustered below 1 million. 5a would be stronger for the ML portion, especially since it includes the general population of readers.

Feature 6: Use the Tags, BigQuery

I decided to filter for particular tags in the dataset to approximate genre categories for works. While books such as Twilight will be tagged in multiple categories, such as "fantasy," "vampire," "romance," and "young-adult," I hoped that the overall aggregate per genre could be distinguished.

For example, works on Goodreads are more likely to be classified as fiction (suggesting that there is more fiction on Goodreads with high # of ratings), even though more nonfiction is published and sold per year than fiction (https://www.forbes.com/sites/adamrowe1/2018/08/30/traditional-publishers-are-selling-way-more-non-fiction-than-fiction/#22062a256d0e).

Caution: do note that unfortunately, the tags are stored haphazardly in this dataset. For example, the same tag "to-read" has 7642 different "count" values, suggesting that there are different tag-ids for the same same. I have chosen to sum count values for the same tag name, e.g. "to-read" has a count of 140,718,761 times used, which seems reasonable.

Note that tags are defined by users, and there are approximately 50,000 unique user_ids on the dataset, and books can have multiple tags.

Interlude: Top 15 Most-Used Tags

For reference, before I start filtering out tags.

```
1 %%bigquery --project $project_id
2 SELECT DISTINCT tag_name, SUM(tags.count) as sum_count
3 FROM
4 `cs145-project3-books.goodreads_top10k.book_tags` tags,
5 `cs145-project3-books.goodreads_top10k.string_tags` tag_names
6 WHERE tags.tag_id = tag_names.tag_id
7 GROUP BY tag_name
8 ORDER BY sum_count DESC
9 LIMIT 15
```

	tag_name	sum_count
0	to-read	140718761
1	currently-reading	7507958
2	favorites	4503173

Interlude: Counts for the Top 15 Genre Tags + 2 Category Tags

4	fantasy	3548157

I combed through 70+ tags by hand, and excluded non-genre tags, such as "library," "kindle," and "owned books," to have an estimation of 15 genres as well as +2 for "fiction" and "nonfiction" to approximate the averaged work counts for the top most-tagged genres. For tags like "science-fiction" and "sci-fi," or "young-adult" and "ya," I opted to use the full name.

```
%%bigquery --project $project_id

SELECT DISTINCT tag_name, SUM(tags.count) as sum_count
FROM
`cs145-project3-books.goodreads_top10k.book_tags` tags,
`cs145-project3-books.goodreads_top10k.string_tags` tag_names
WHERE tags.tag_id = tag_names.tag_id
AND tag_name IN ("fiction", "fantasy", "young-adult", "adventure", "romance",
"mystery", "non-fiction", "historical-fiction", "science-fiction", "paranormal-romance",
"horror", "urban-fantasy", "thriller", "humor", "chick-lit", "memoir", "children")
GROUP BY tag_name
ORDER BY sum_count DESC
```

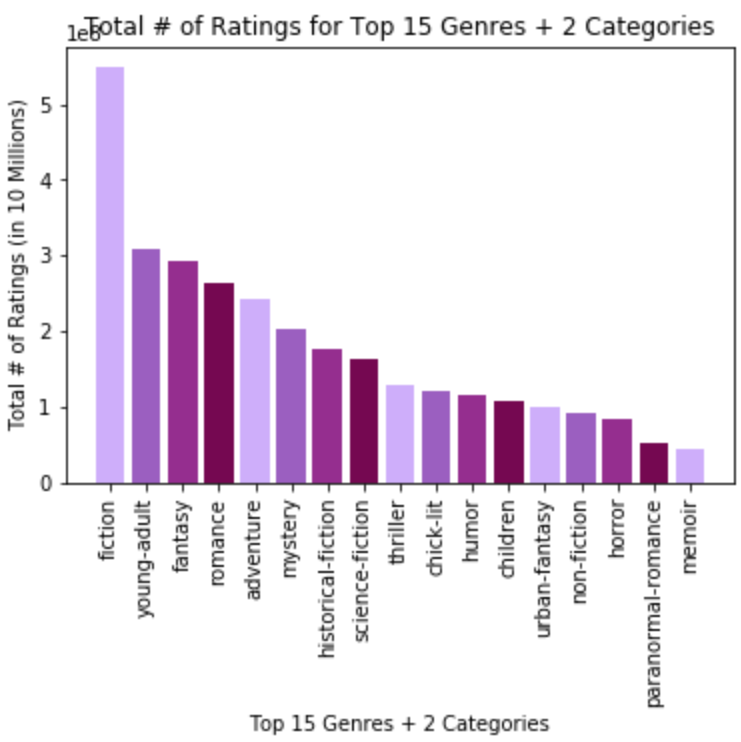
	tag_name	sum_count
0	fiction	3688819
1	fantasy	3548157
2	young-adult	1848306
3	romance	1231926
4	mystery	872282
5	non-fiction	857901
6	historical-fiction	815421
7	science-fiction	703866
8	horror	415467
9	urban-fantasy	374689
10	thriller	309276
11	adventure	295933
12	humor	262340
13	chick-lit	251072
14	paranormal-romance	221939
15	children	218410
16	memoir	183477

Sum # of Ratings for 15 Genres + 2 Categories

```
%%bigquery --project $project_id genre_tags_sum
SELECT tag_name, SUM(books.work_ratings_count) as sum_ratings_count
FROM
`cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.book_tags` tags,
`cs145-project3-books.goodreads_top10k.string_tags` tag_names

WHERE books.best_book_id = tags.goodreads_book_id
AND tags.tag_id = tag_names.tag_id
AND tag_name IN ("fiction", "fantasy", "young-adult", "adventure", "romance",
"mystery", "non-fiction", "historical-fiction", "science-fiction", "paranormal-romance",
"horror", "urban-fantasy", "thriller", "humor", "chick-lit", "memoir", "children")

GROUP BY tag_name
ORDER BY sum_ratings_count DESC
```

```
%matplotlib inline

plt.title("Total # of Ratings for Top 15 Genres + 2 Categories")
plt.bar(genre_tags_sum["tag_name"], genre_tags_sum["sum_ratings_count"], color=["#ceaefa", "#9b5fc0",
plt.xlabel("Top 15 Genres + 2 Categories")
plt.xticks(genre_tags_sum["tag_name"],  rotation='vertical')
```

```
7 plt.ylabel( 'Total # of Ratings (in 10 Millions)' )
8 plt.show()
```



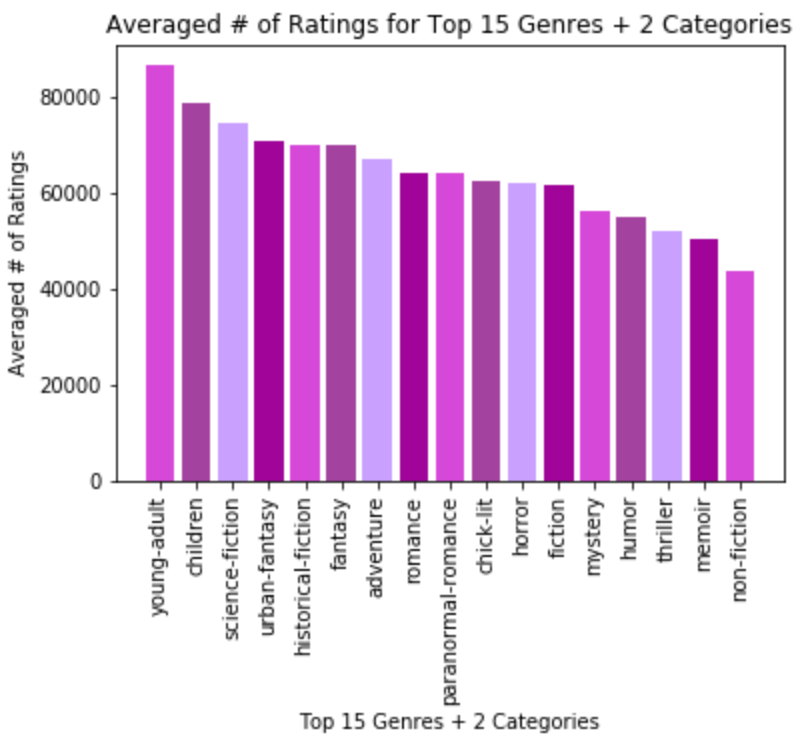
I sought to see which genres have the highest user tagging, and how they correlate with a total # of ratings for works with this tag. As expected, fiction has the highest total # of ratings (since it is a general category), followed by young-adult, fantasy, and romance. This is evidenced from the top 3 works (which are tagged young-adult and romance, even Harry Potter, and two are tagged fantasy), suggesting that these categories tend to skew highly.

Averaged # of Ratings for 15 Genres + 2 Categories

Unlike the total sum per genre, I averaged the votes to see how it will affect the average book and reducing the effect of extreme bestsellers (such as Harry Potter) on the tagging.

```
 1 %%bigquery --project $project_id genre_tags
 2 SELECT tag_name, AVG(books.work_ratings_count) as avg_ratings_count
 3 FROM
 4 `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
 5 `cs145-project3-books.goodreads_top10k.book_tags` tags,
 6 `cs145-project3-books.goodreads_top10k.string_tags` tag_names
 7
 8 WHERE books.best_book_id = tags.goodreads_book_id
 9 AND tags.tag_id = tag_names.tag_id
10 AND tag_name IN ("fiction", "fantasy", "young-adult", "adventure", "romance",
11 "mystery", "non-fiction", "historical-fiction", "science-fiction", "paranormal-romance",
12 "horror", "urban-fantasy", "thriller", "humor", "chick-lit", "memoir", "children")
13
14 GROUP BY tag_name
15 ORDER BY avg_ratings_count DESC
```

```
1 %matplotlib inline
2
3 plt.title("Averaged # of Ratings for Top 15 Genres + 2 Categories")
4 plt.bar(genre_tags["tag_name"], genre_tags["avg_ratings_count"], color=["#d648d7", "#a442a0", "#caa01
5 plt.xlabel("Top 15 Genres + 2 Categories")
6 plt.xticks(genre_tags["tag_name"],  rotation='vertical')
7 plt.ylabel("Averaged # of Ratings")
8 plt.show()
```



When averaging across books (thus fiction and non-fiction fall lower on the graph), young-adult, children, science-fiction, urban-fantasy, and historical-fiction are the top 5, suggesting that these tagged categories are more likely to correlate with books with higher ratings. Using the average reduces the effect of extreme bestsellers, and more accurately reflects the whole dataset.

Data Prediction: Estimate # of Ratings

I am using logistic regression to estimate the # of ratings, based on the factors listed per model.

Train on the set of books up to end of 2007, starting with The Epic of Gilgamesh first published in 1750 BCE (6097 entries).

Test on a 4-year period of books published from the start of 2008 up to end of 2011 (1844 entries).

Predict a 5-year period of books originally published from the start of 2012 to 2017 (2038 entries).

Exclude 21 books with null years (mostly series collections and movie companions).

```
# Initialize BiqQuery client
from google.cloud import bigquery
client = bigquery.Client(project=project_id)
```

```
# Run this cell to create a dataset to store your model, or create in the UI

model_dataset_name = 'bqml_project3_goodreads'

dataset = bigquery.Dataset(client.dataset(model_dataset_name))
dataset.location = 'US'
client.create_dataset(dataset)
```

Model 1: Author Name, # of Text Reviews, # of Users Marked Book To-Read

I am using these factors as they seem to correlate best with the explored data above. Each factor comes from one stage of my analysis, in hopes of increasing the power of this model.

Author name strongly ties with high reviews if they are already established (and it helps if they are highly prolific). Similarly, the # of text reviews is strongly linked with a higher # of ratings (see above). Lastly, users are more likely to mark a book to-read if they are exposed to it, which necessitates the book to be more popular, which means that it is more likely to have a higher reach (# of ratings).

Training

```
%%bigquery --project $project_id

CREATE OR REPLACE MODEL `cs145-project3-books.bqml_project3_goodreads.model1`
OPTIONS(model_type='linear_reg', input_label_cols=["work_ratings_count"]) AS

SELECT work_ratings_count, books.authors as author_name, books.work_text_reviews_count as text_reviev

FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.to_read` to_read

WHERE books.id = to_read.work_id
AND original_publication_year <= 2007
AND original_publication_year IS NOT NULL
GROUP BY to_read.work_id, author_name, text_review_count, work_ratings_count
```

```
%%bigquery --project $project_id

# Run cell to view training stats

SELECT
  *
FROM
  ML.TRAINING_INFO(MODEL `bqml_project3_goodreads.model1`)
```

	training_run	iteration	loss	eval_loss	learning_rate	duration_ms
0	0	7	3.837100e+09	7.942167e+09	1.6	3284
1	0	6	3.959594e+09	8.289439e+09	1.6	3812
2	0	5	4.186348e+09	8.704168e+09	0.8	3336

Testing

```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.EVALUATE(MODEL `bqml_project3_goodreads.model1`,
  (
SELECT work_ratings_count, books.authors as author_name, books.work_text_reviews_count as text_review

FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.to_read` to_read

WHERE books.id = to_read.work_id
AND original_publication_year <= 2011
AND original_publication_year >= 2008
AND original_publication_year IS NOT NULL
GROUP BY to_read.work_id, author_name, text_review_count, work_ratings_count
))
```

	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
0	53697.164817	2.022389e+10	2.727226	33144.40896	0.416601	0.48797

Predicting

```
%%bigquery --project $project_id

SELECT
  *
FROM
  ML.PREDICT(MODEL `bqml_project3_goodreads.model1`,
  (
SELECT work_ratings_count, books.authors as author_name, books.work_text_reviews_count as text_review

FROM `cs145-project3-books.goodreads_top10k.top10k_all_books` books,
`cs145-project3-books.goodreads_top10k.to_read` to_read

WHERE books.id = to_read.work_id
AND original_publication_year >= 2012
AND original_publication_year IS NOT NULL
GROUP BY to_read.work_id, author_name, text_review_count, work_ratings_count
LIMIT 20
))
```

	predicted_work_ratings_count	work_ratings_count	author_name	text_review_count	num_to_readers
0	48229.678605	50489	Harlan Coben	4793	84
1	30041.975996	33348	Madeleine Roux	4166	98
2	-8330.400430	24367	A. Meredith Walters	1179	41
3	-9454.473240	18185	Janet Evanovich, Lee Goldberg	1730	12

In sum: some potential flaws are that Goodreads launched in the past two decades; the prediction harder for more recent or contemporary works.

In my working ML Model, I used the author name, # of text reviews, and # of ratings. In an earlier prototype, I chose to use book tags instead of author name, but I suspect that the downside of the second model is the aforementioned poor organizing of the tag_ids (which are not unique), which may have affected its training.

In any case, the model is accurate for a few books, such as Harlan Coben and Madeleine Roux, with a difference of only 3000. I suspect that the #-based categories have a greater effect than the author names, since works that come out after 2013, which the model was not trained on, have more skewed predictions.

This suggests that at least these two factors are good predictors for future book reach predictions.

12	6461.345462	30842	Mariana Zapata	3311	21

Conclusion: Goodreads has Reads with High Reach

From my analysis, my work suggests that Goodreads as a whole has a wide variety of reads with a high # of ratings. In particular, text reviews, proportion of 4 stars to other star ratings, author names, established book tags, and # of users that mark the work to-reads correlate most strongly with a work that has a high # of ratings.

Some limitations of my work, as evidenced from Model 2, is the messy data for the book tags, which could have potentially skewed the data. Though the graphs seem reasonable for the top tags, even after filtering out extraneous tags, the poor mapping of tag id to names reduces my interval of confidence. I wonder if established genre categories would have been more useful, had this set included them.

Nonetheless, I personally thought that using the proportion of 4-star ratings to the rest of the dataset was particularly interesting, as it suggested that even the distribution of ratings for the books (despite the skew of average_ratings and the ratings in general on Goodreads) could have a factor in which books have a perceived high "reach." Likewise, even the quicker queries had a significant effect, such as # of text reviews; author names often do act as "brand-names" in deciding the reach of their other books.

In future steps, it would be interesting to bring in other datasets, such as the NYT bestsellers, to see how this model fares with critically acclaimed works that may not have a wide reach of readers (e.g. they are not bestsellers like Stephen King's works). And what about works that escape the confines of the recently launched Goodreads, such as classics like The Odyssey? Or, what about a deeper venture into the most popular genres, e.g. Young Adult, which will have a larger sample?

There are many new pages here to uncover.
