Determining Novels for a Book Club

Emily Park, Fang (Emma) Liao, Christen McKenzie, and Nathaniel (Nat) Marcuson

Introduction

In this paper we seek to determine the ideal set of books for a book club, given that each member is allowed to rate the books. To achieve this, we create an integer program (IP) to maximize the 'happiness' pertaining to the book choices. The constraints of this IP include the quantity of books chosen and insurance that every group member gets books that they enjoy. One question that this paper seeks to answer concerns the best way to average the ratings for each book. To answer this question, we compare solutions for taking the median, the mean, and the mean of the squares. Ultimately, the squares solution results in the best book choices.

Furthermore, we seek to optimize other objective functions using integer programming. In one instance, we minimize the price of books chosen for the book club, with an extra constraint that the total ratings must not be too low. In another, we prioritize the happiness of one person while still making sure the rest of the club is relatively happy. This method for determining the best books for a book club is ultimately successful, as each solution produces quality results that capture what the model seeks to optimize.

Background

In discussing the topic for this project, our group was intrigued by process of maximizing happiness, something subjective. We considered maximizing movies for a film festival or song selections for karaoke. However, the interests of our group did not lay in digital media, and Christen suggested a suitable alternative concerning a topic we all appreciate: books. Further discussion on how this could be formulated as a linear program (LP) led to the idea of a book club. People in the same book club will not always enjoy the same books, but an LP can ensure that nobody is terribly unhappy with the book choices.

Our project is similar to the knapsack problem that we did for homework. Instead of maximizing the value of the knapsack, we maximize the goodness or how much all of us would enjoy the books. Each of us chose 20 books, to use for this modified knapsack problem. We then each rated all 80 books, so that each book had a set of four ratings. We use binary variables to represent our books just like how the knapsack problem uses binary variables to represent the items to put in the knapsack. In our project, we maximize the happiness (the book ratings) for 12 separate books, choosing 1 book each month.

Modelling a Nurse Shift Schedule with Multiple Preference Ranks for Shifts and Days-Off [3] is a study orchestrated by Chun-Cheng Lin, Jia-Rong Kang, Wan-Yu Liu, and Der-Jiunn

Deng published in 2014; in this study they also had to deal with this idea of maximizing a subjective value such as preference. In this case, the preference concerned shifts in a hospital. In the study they accomplished this task by setting their objective function to be a maximization of preference for work shifts and days off. The scheduling problem in general, is different than ours since in the nurse scheduling problem they give various weights to different shifts based upon past shift schedules and gives higher preference to nurses who may not have gotten their preferred shift in the last scheduling. It is also different in the sense that the nurses presumably have experience with all sorts of shifts, while members of a book club will not have experience with all of the books that they are rating. For us, although not putting nurses in a schedule at a hospital, we are putting books into a schedule so this is a similar idea, especially since we also aim to maximize preference in a way through our maximization of the ratings.

Throughout our study we are dealing with ratings of books. Commonly, people go to a website like Goodreads and are given what a large swath of the publicly determined ratings. But these Goodreads ratings will not mean as much for a smaller group since the smaller group may have different views than those gathered by Goodreads. Another study solving a similar problem had to do with Recommendation Systems. In these systems they usually take ratings from a large online database and forcast what another user may also be interested in. One such study was done called *Fairness- Aware Group Recommendation with Pareto-Efficiency*[4], and in part of their process they mentioned how there can be approaches to being 'fair' in a model by doing min-max fairness or least misery. This idea of 'Least Misery' is related to our idea of maximizing preference, since it is our belief that by maximizing preferences we will also be accomplishing the task of hopefully approaching a solution that is the least miserable to the group as a whole.

The Model

We have attached the Excel spreadsheet containing our book choices and our corresponding ratings into the appendix section. This spreadsheet was created after the four of us each chose 20 books for a total of 80 books. After having our list of 80 books, we then all rated each of the 80 books on a scale of 1 to 10. In this rating scheme, a 1 would be least desirable and a 10 would be most desireable. Then once we had our data, we could approach our problem of finding a set of books that would lead us all to be the most content. A problem that came up was what a rating of 6 means for one person may have a very different meaning for another group member. To overcome this rating obstacle and the difficulty of answering a subjective question, we tested three different solution methods. The first was through taking the median of scores, the second the mean, and the third through the mean of the squares of each score. As discussed thoroughly in the commentary section, the squares solution resulted in the best set of books. Thus, we will discuss the model of the squares solution in this section.

For our IP problem we have 80 binary variables to represent the 80 books that we have rated. Each binary variable equals one if and only if the book that it represents is to be included in the list of 12 books. Otherwise, the binary variable should equal zero.

In other words, we have:

$$b_n \in \{0, 1\} \quad : \quad n = 1, 2, ...80$$

Objective Function:
We take the mean of the squares of the four ratings attributed to each book and ultimately sum the product of this average rating and the binary variable (b_n for $n = 1, 2, ...80$) that represents each book. The objective function aims to maximize this. In maximizing these average ratings, we are maximizing how likely all of us would enjoy the list of 12 books picked.
We calculated the average ratings by taking $r_{i,n}$ with i corresponding to the rating given by person i, for $i = 1, 2, 3$, or 4, and n corresponding to the n-th book with $n = 1, 2, ...80$. From here we take the sum of the squares of these ratings for each individual book. In other words if we categorize each of these average ratings as a_n with $n = 1, 2, ...80$ corresponding to each of the book indices again then we have

$$a_n = \frac{\sum_{i=1}^{4} (r_{i,n})^2}{4} \quad : \quad n = 1, 2, ...80$$

Once we have calculated this a_n in Matlab we now use them as coefficients for our objective function. We can now write our objective function as follows:

$$Total\ Value = \sum_{n=1}^{80} a_n b_n$$

Next, we model our constraints.

Constraint 1:
In this problem, one of the constraints pertains to the fact that since this is a monthly book club where we only read one book per month, we want the exact number of books overall to equal 12, so the following constraint will accomplish this:

$$\sum_{n=1}^{80} b_n = 12$$

This will add all the binary book variables together and ensure that they equal 12.

Constraint 2 - Constraint 5:
We decided that we want to make sure that everyone gets at least one book that has a rating of ten. So, ideally, everyone gets one of their top choices. We accomplish this by making sure that for every person's set of ratings, $r_{1,n}, r_{2,n}, r_{3,n}, r_{4,n}$, at least one 10 from their set of ratings is chosen. So first we had to find out for each person, which book with index n did they rate a ten.

Constraint 2:
For Christen, the books rated a ten were at $n = 2, 10, 51$, and 76. Therefore, based on this criterion above we decided on, at least one of the books corresponding to b_2, b_{10}, b_{51}, or b_{76} must be chosen. To do this, we craft the following constraint:

$$b_2 + b_{10} + b_{51} + b_{56} \geq 1$$

Constraint 3:
Similarly for Nat, the books rated a ten were at $n = 5, 7, 39, 40, 41, 48, 63, 67$ and 70. Therefore, based on this criterion above we decided on, at least one of the books corresponding to $b_5, b_7, b_{39}, b_{40}, b_{41}, b_{48}, b_{63}, b_{67}$ or b_{70} must be chosen. To do this, we craft the following constraint:

$$b_5 + b_7 + b_{39} + b_{40} + b_{41} + b_{48} + b_{63} + b_{70} \geq 1$$

Constraint 4:
Similarly for Emma, the books rated a ten were at $n = 20, 30, 31, 61$, and 67. Therefore, based on this criterion above we decided on, at least one of the books corresponding to $b_{20}, b_{30}, b_{31}, b_{61}$, or b_{67} must be chosen. To do this, we craft the following constraint:

$$b_{20} + b_{30} + b_{31} + b_{61} + b_{67} \geq 1$$

Constraint 5:
Similarly for Emily, the books rated a ten were at $n = 25$ and 31. Therefore, based on this criterion above we decided on, at least one of the books corresponding to b_{25} or b_{31} must be chosen. To do this, we craft the following constraint:

$$b_{25} + b_{31} \geq 1$$

The preceding constraints force at least one of these b_n values for each person to be 1, and now at least one of these books must be chosen for everyone.
Overall, these ensure that all of us would have a book that we rated 10 in the final list of 12 books.

Constraint 6:
Before this constraint we found in the initial tests that we kept having multiple Harry Potter books show up in the final tally. We collectively agreed that this was not something that we wanted so we created this constraint in order to make sure that we would at most have one Harry Potter book over the course of our twelve month book club. We realized from looking at the Excel file that these were indices 61, 62, 63, 64, 65, 66, 67, and 68. Knowing this, we made the following constraint:

$$b_{61} + b_{62} + b_{63} + b_{64} + b_{65} + b_{66} + b_{67} + b_{68} \leq 1$$

Since these values are less than or equal to one, it now ensures that at most one of these books will show up in our final list.

We used Matlab to generate our objective function and constraints and then used LPSolve to run them. We initially used a Google Sheet to give our ratings for all of the 80 books we

chose and then downloaded it as an Excel file. Then we used Matlab to take in the data with a couple of for-loops. The first for-loop takes in the rating each of us gave for each book, squares them, then takes puts the mean of these ratings for each book in an element in the array aveScore. The index of the array tells us which book it represents. Another for-loop takes in the averaged rating for each book to generate the objective function. Another for-loop corresponding to Constraint 1 makes sure we choose only 12 books.

When we set out to do this model, we went into it wanting to maximize 'happiness' which is a bit of an arbitrary measure. We had to collectively decide what 'happiness' in our scenario would mean. For us, a good measure of 'happiness' would rely largely on our ratings. We created coefficients corresponding to each book by first squaring each rating for an individual book, and then taking the mean of these ratings. We defined 'happiness' as the sum of each book multiplied by its corresponding averages coefficient. This allowed us to give an individual weight to each book based upon the ratings we gave in the Excel file.

We decided that squaring these ratings and then taking the average would be the best method that would accomplish the task of giving a greater weight to higher ratings. This is due to the fact that the square of a large number will be much larger in magnitude to the square of a smaller number. This also helps account for the other problem that we had about the subjectiveness of the middle section (ratings of 4-6) of our rating scale.

Some challenges include sharing the code on Matlab after making edits to it. Other challenges included using appropriate indexing from the Excel file. Since Matlab only gathers the cells that include numerical properties, we ultimately had an 80 by 1 vector for each person with row one corresponding to the rating for book one. But in the Excel file row two contained ratings for book one since the top row of cells which included the heading in Excel went away. For example, when we got results from LPsolve we had to keep in mind that b_{20} would actually be located in row 21 of the Excel file when gathering our results. As a consequence, since we did not have another column in the Excel file that corresponded to each author, we could not parse through and find the J.K. Rowling books via a similar method that we used for Constraint 2-5. Instead for Constraint 6, we had to actually look at the Excel file and when thinking of indexing for which books were J.K. Rowling, we had to keep this aforementioned Excel index difference outlined above in mind so that we referred to the correct index to ensure we only received one book written by J.K.Rowling in our final book list.

In addition, Constraints 2-5 are not always needed in the scenario (for example if the books naturally maximize with everyone getting one of their books of 10), but for later variations (and in general) it is beneficial to have as it acts as another measure of 'fairness' for cases where we are not maximizing the average ratings.

To accomplish the task set forth in Constraint 2-5, we had to write code in Matlab which would keep track of the indices so we could know which book index corresponded to a rating of 10. To do this, we first created a vector in Matlab just containing Christen's ratings called ChristenRatings, a vector containing Nat's ratings called NatRatings, a vector containing Emma's ratings called EmmaRatings, and a vector containing Emily's ratings called EmilyRatings. Then for each of these four vectors, we would create a for-loop where it

cycles through indices 1 to 80 and if for example ChristenRatings(index) is equal to ten, then we saved that index in a new vector that contains all indices corresponding to books where the rating is 10 for that specific person. We called this new vector of just indices for books with 10 ratings tenRatingIndexCM for Christen, tenRatingIndexNM for Nat, tenRatingIndexEL for Emma, and tenRatingIndexEP for Emily. Then with another loop we were able to write out the constraint we needed by looping through these vectors of indices that we found.

Solution

After the Excel file had been completed, we used Matlab to process the data and determine the input file for lpsolve (see appendix for fully commented code). For the purpose of condensing the solution, we will now only look at the solution concerning the mean of the squares. For-loops were used to determine the overall rating for each book by taking the mean of the squares of each score. These average ratings ended up determining the objective function described previously. For-loops were also used to determine the constraints that each person will get one book they rated perfectly in the final solution. A for-loop also ensured that each b_n was binary. However, the JK Rowling constraint was not determined algorithmically; it was manually typed into Matlab. Once all of these constraints were defined in Matlab, they were written to a text file named Project381square_average1.txt. The text file was of the following form:

```
max: 45.750b1 + 73.500b2 + ... + 50.500b79 + 4.500b80;

(Ensures only one books is chosen)
b1+ b2+ ... + b79+ b80=12;

(Ensures Christen gets at least one book rated 10)
b2 +b10 +b51 +b76 >= 1;

(Ensures Nat gets at least one book rated 10)
b5 +b7 +b39 +b40 +b41 +b48 +b63 +b67 +b70 >= 1;

(Ensures Emily gets at least one book rated 10)
b25 +b31 >= 1;

(Ensures Emma gets at least one book rated 10)
b20 +b30 +b31 +b61 +b67 >= 1;

(Ensures only one JK Rowling book is included)
b61 + b62 + b63 + b64 + b65 + b66 + b67 + b68 <= 1;

(Ensures all variables are 1 or 0)
bin b1, b2, b3, ..., b78, b79, b80;
```

Lpsolve took this input and gave the following output in .04 seconds on a Macbook Pro. All

other variables equal zero.

```
Value of objective function: 873.25000000

Actual values of the variables:
b2                              1
b5                              1
b7                              1
b20                             1
b25                             1
b30                             1
b31                             1
b39                             1
b40                             1
b41                             1
b55                             1
b67                             1
```

Thus, when cross referencing these b_n values with the Excel Sheet the books that this solution deems best are *The Princess Bride*, *11/22/63*, *East of Eden*, *1984*, *The New Iberia Blues*, *The Wicked King*, *The Tattooist of Auschwitz*, *We are Displaced*, *The Hate U Give*, *The Stand*, *The Adventures of Tom Sawyer*, and *Harry Potter and the Deathly Hallows*.

Input and output files for the mean and median solutions may be found in the appendix.

Commentary on Solution

In comparison to the above optimal solution, the mean solution resulted in the following book club choices: *The Princess Bride, 11/22/63, East of Eden, 1984, The Wicked King, We Are Displaced, Where the Crawdads Sing, The Tattooist of Auschwitz, The Hate U Give, The Stand, The Adventures of Tom Sawyer,* and *Deathly Hallows*. The median solution we also had considered resulted in the, again, slightly different choices: *The Princess Bride, 11/22/63, 1984, We Are Displaced, Becoming, Where The Crawdads Sing, The Tattooist of Auschwitz, The Hate U Give, The Stand, The Catcher in the Rye, The Adventures of Tom Sawyer,* and *Deathly Hallows*.

Concerning the constraints of the model, it is clear that the 12-book constraint is binding since if the constraint called for any other number of books, the solution would be different. However, the constraint that each person get at least one book they rated 10 varied for each person. There were multiple books for which Nat and Emma rated 10 in all solutions (median, mean, mean of the squares), so their constraints are not binding. There was only one book that Christen rated 10 that appeared in all three solutions, so her constraint is binding. There was only one book that Emily rated perfectly in the median and mean solutions, so her constraint is binding for those solutions. And there were two books she rated 10 in the squares solution, so her constraint was not binding for that solution. Furthermore, the JK Rowling constraint is binding as there is exactly one JK Rowling book in each solution.

After comparing the results for the differing solution methods, little difference is found. Each variation contains the same 9 of 12 possible books: *The Princess Bride*, *11/22/62*, *1984*, *We Are Displaced*, *The Tattooist of Auschwitz*, *The Hate U Give*, *The Stand*, *The Adventures of Tom Sawyer*, and *Harry Potter and the Deathly Hallows*. Furthermore, the median and mean solutions both contain *Where The Crawdads Sing*, and the mean shares *East of Eden* and *The Wicked King* with the squares group. These similarities are expected, as books with generally high ratings will perform well in each category. However, marked differences between the categories do exist, including the frequency of low scores, perfect 10s, and of 8s and 9s.

Because 9 books were exactly the same, it is only logical to compare the 3 differing books, as comparisons between the same books are superfluous. The table on the following page illustrates these differences. Concerning the frequency of low scores, we will qualify a "low score" as being of 5 or less. The median group contained two low scores (3 and 5), while the mean and squaring group both contained no low scores. This disparity in low scores is due to the fact that the median effectively discards the lowest score in a set. For example, the set {9,9,9,9} has the same median as the set {9,9,9,3}. However, the second set has a lower mean and mean of squares than the first set. Thus, books with particularly low ratings will be included less in the mean and squares solutions than the median solution, as they will lower the objective function.

Another marker of the quality of solution is the prevalence of perfect 10s. In the differing books, the median group has one 10, the mean group has two 10s, and the squares group has three 10s. Squaring scores causes the differences between two scores to be more exaggerated as the score increases. For example, the difference between 9^2 and 10^2 is greater than the difference between 7^2 and 8^2, even though the difference in raw score is only 1 in both cases. This causes the squares solution to weight 10s much greater than the mean and median solutions, thus 10s occur more in the squares solution. Furthermore, there are no 10s in the median solution due to a process similar to the one that allows for low scores in the median group. A single 10 will have little effect on the median of book scores, as the 10 will be effectively discarded in calculating the median.

In addition to the low and perfect scores, 'high scores' also demonstrate solution quality. Because perfect 10s are in their own category, a 'high score' will be quantified as rating of 8 or a 9. The median group contained nine 'high scores', the mean group had five, and the squares group had three. The prevalence of 'high scores' in the median group is due to the fact that the middle two scores are used in determining the median. Thus, if the middle two scores for a book are high, the book is more likely to be included in the solution. The squares group contains only three non-10 'high scores' because of the importance placed on 10s previously discussed. This causes the prevalence of 10s to affect the squares solution much more than 8s and 9s, causing less importance to be placed on 'high scores'.

Through purely looking at the prevalence of different scores, it is difficult to objectively determine which solution is best. So, we conducted a side experiment where we took the ratings of the 3 books that were different in each variation and voted on which set was the 'best' or looked like a set of ratings that would most maximize happiness. We voted on the 3 following sets without knowing which variation the 3 sets represented:

Set A (median)	Set B (squares)	Set C (mean)
9,3,9,9	7,10,8,7	7,10,8,7
8,6,9,9	9,6,10,7	6,7,9,10
9,8,5,9	6,7,9,10	8,6,9,9

Each person was asked to place a first place vote on their preferred set, and a second place vote on their next preferred set. Set B received three first place votes and one second place vote. Set A received one first place vote and one second place vote. Set C received two second place votes.

Most of us chose Set B (The Squares Solution) as the set of ratings that looked like all of us would enjoy the most since each book was given a 10 from one of us. Of course, the result of this side experiment is based purely on personal preference and could differ if we let other people vote on what they liked as well. However, it did shed some light on which ratings are more highly valued by our group.

As previously discussed, the square of bigger numbers is a lot larger than the square of smaller numbers and squaring the ratings makes the difference between smaller and bigger numbers even more significant, effectively giving larger numbers even more weight. Apparently, our group places a great importance on the prevalence of perfect 10s over a high frequency of 8s and 9s. Though the squares solution had a 10 for each book, the remaining three ratings were also not low. It makes sense that set B was chosen as each book has at least a 10 from one of us without the remaining scores being too low, with only two scores under 7. Having no low scores was also important to choosing the best solution, as it means that no one will be particularly unhappy with the book choices.

Set A (The Median Solution) received the second highest ratings from us. This may be surprising due to its frequency of low scores and lack of 10s. However, the abundance of 8s and 9s apparently makes up for its other shortcomings. Finally, Set C (The Mean Solution) was blindly rated as the worst. Though it did not contain any low scores, it simply did not have enough 8s, 9s, or more importantly 10s to compete with the other solutions. Furthermore, we all decided on what we individually thought was the best set in another side experiment by looking at the books instead of the ratings. Each person's first choice remained the same, further demonstrating that the squares solution maximized happiness in this particular problem and presented a realistic option for book choices since we all got choices we liked and rated highly.

Variations

Minimizing Price

In our first variation we have chosen to replace the objective function with minimizing the sum of prices of the books. As students, we have a very low budget when it comes to buying books. This variation is useful in helping us determine which books we should buy for our

book club while also making sure we do not spend too much money on books. On the other hand, we also do not want to read just the cheapest books available since these books may not be highly rated. This absence of 'happiness' maximization without another constraint regarding our rating preferences would defeat the purpose of the book club since people would be less likely to want to read the chosen books and be likely to drop out of the group when this is the case. Thus, we also included the constraint that ensures that the sum of the average of squared ratings is greater than or equal to 650, which is also representative of the happiness we could derive from reading the books. This number was selected by looking at past models and the value of the objective function.

The results are a list containing *The Princess Bride*, *Murder on the Orient Express*, *Richard II-Shakespeare*, *1984*, *Alice in Wonderland*, *We are Displaced*, *The Tattooist of Auschwitz*, *The Catcher in the Rye*, *The Count of Monte Cristo*, *The Adventures of Tom Sawyer*, *Northern Lights*, and *Harry Potter and the Sorcerer's Stone*. As compared to the other models, we can definitely see that the list of books chosen in this variation have comparatively lower ratings. This variation only has five books in common with the results of the model that we used previously and the other seven books have relatively lower ratings than the books in the other model. Books such as *Alice in Wonderland*, although very cheap at just $3, has rather low ratings of 6, 4, 4 and 4. This would be a less desirable choice for the groups as a whole.

For the result of the model with objective function that is maximizing the sum of mean of squares, we got the solution previously discussed. Because we maximize the goodness for this model, the original solution has much higher ratings than this solution; however, the sum of the prices is $151.03 which is much more expensive than the sum of prices of books we got from the variation model, $92.66. We can definitely see the variation is modeled to choose cheaper books rather than having higher 'goodness', so this would be a useful variation if the book club just wants some cheap books that may not have the maximum ratings. This variation is especially relevant to people who cannot afford to spend too much money on books.

Maximizing One Member's Perfect Scores

Our second variation gives one person a precedence over the others. To achieve this, we maximize Nat's constraint which had originally ensured that he receives at least one book that he rated a 10, so now it attempts to give him the most 10s possible while still abiding by our other constraints. We chose to maximize Nat's 10s simply because he rated the most books perfectly, so looking at his solution would give the most interesting results. In addition, we made our original objective function into a constraint by setting our total happiness to be greater than or equal to 800. We decided on 800 after looking at what the max value of this constraint since we noticed in our last averaging squares model, the maximum of the objective function was totaled at over 800. This seemed like a good starting point for what we wanted to do with this variation.
Our objective function becomes:

$$Total\ Value = b_5 + b_7 + b_{39} + b_{40} + b_{41} + b_{48} + b_{63} + b_{67} + b_{70}$$

When we first ran this scenario, we had a problem come up since we no longer maximized the 'goodness' ratings. The problem was that it kept choosing most of Nat's books rated 10 but the remainder of the books chosen consisted of ones scored low by all of us, such as *Queen of Air and Darkness* which was rated as a (2, 1, 3, 2) by the four of us.

This did not capture what we truly wanted to model in this scenario, so we had to update our model and add a constraint making sure no books rated 'low' were added. We categorized a book as being rated 'low', if it was rated a 5 or below by any member of the group. We agreed that a book with that rating would be undesirable to the group. In order for this process to work and still pick fairly good books we decided to add this constraint. Full implementation can be seen in the appendix section for code on variations. It boils down to a for-loop deciphering what is and is not a book rated 5 or below. This constraint essentially weeds out 47 books from the possible solution. The specific constraint would be that the sum of all b_n for books with ratings of five or lower be equal to zero.

Since this scenario maximizes one person's preferences over the others, this approach may be applicable to a scenario such as a teacher picking books for a class group reading schedule. Perhaps the teacher wants everyone to have a say in the matter and to let them vote for their picks, but ultimately the teacher wants to have some books (the ones the teacher has rated ten) to be mandatory reading unless they are rated a 5 or below by anyone in the class.

The result of this variation is a book list containing *1984*, *Murder on the Orient Express*, *We Are Displaced*, *My Name is Asher Lev*, *Princess Bride*, *Lord of the Rings*, *Harry Potter and the Deathly Hallows*, *The Stand*, *The Hate U Give*, *Tattooist of Auschwitz*, *East of Eden*, and *11/23/63*. Some of these choices, such as *My Name is Asher Lev*, only had a rating of 7, 6, 6, and 7 from the four of us. Although not a horrible choice that would make us completely unhappy, it does skip some much better choices such as *We are Displaced* by Malala Yousafzai which had a much higher rating (10, 10, 8, 6) overall. It included 7 of the books that Nat rated a 10, only leaving out two choices: one because we have the J.K. Rowling constraint, and the other (*This Book Loves You*) because the rest of us had only rated it a 5. As a whole, the book list does present an outcome of mostly ratings of 8, 9 and 10, but it does present a flaw since the best rated choices are no longer necessarily chosen.

All in all, this variation presents an interesting dilemma that occurs when the 'goodness' function is no longer maximized. Now, the solution requires another restraint in order to capture the behavior we would like it to present. It does achieve a book list that is 'fair' in the sense that everyone gets one book they really like (rated 10) and there are no low rated books despite maximizing the choice of one person over another. We still achieve a quality book list where dropouts in the club are less likely because of these constraints that result in higher rated books.

Conclusion

Finding a book list based on a group's own personal ratings versus the ratings of a site like Goodreads can be a more tailored fit to the preferences of the small group. We conclude that there are a lot of different approaches one can take in hoping to achieve a fair model

that maximizes 'happiness'. We found in our approach that it was achievable to create a book list that was formulated in such a way as to reduce the number of dropouts as the club continued throughout the months. Looking at the bigger picture, this forecasting method could be used to determine the ideal books for much larger book clubs with a variety of different constraints and objective functions. However, this method of determining the best books is not perfect. One potential flaw could occur if someone had only rated one book 10 and everyone else hated it and had rated it lowly. This would cause that book to be included in the solution, even though it has a poor rating. This was not a problem for us because everyone rated at least book perfectly, and most of those books had overall high ratings. Another potential problem could simply occur with innacurate ratings. Ratings are mostly done without ever having read the book, so a great book could end up being rated poorly due to it being wrongly judged by its cover.

Ultimately, we found taking the mean of the squares to result in the best solution by maximizing the happiness of our group. From this, we conclude that the difference between high scores is much greater than the difference between low scores. Thus, in rating systems done by non-professionals a 10 is much better than a 9, while the scores 2 and 3 are very similar. However, it must be noted that none of the optimal solutions previously focused on resulted in a bad choice of books. For instance, originally the solution which maximized Nat's ratings of 10 was not the ideal solution for everyone before some updates. But once the constraint that no scores be lower than 6 was included, nobody was unhappy with the book selection. Thus, the framework of our model allows for quality solutions to be obtained no matter the specific goals of the problem.

Appendix

The squares code (in Matlab)

```
%% Average the square of each score
clear all; close all; clc;

filename = 'myExample2.xlsx';

% initializes what columns of rankings corresponds to who from the excel file.
ChristenRatings = xlsread(filename,'C:C');
NatRatings = xlsread(filename,'D:D');
EmilyRatings = xlsread(filename,'E:E');
EmmaRatings = xlsread(filename,'F:F');
ChristensPicks = 1:1:20;
NatsPicks = 21:1:40;
EmmasPicks = 41:1:60;
EmilysPicks= 61:1:80;

% Averages the squared ratings for each individual book and puts
%them in a vector named aveScore,
```

```
% with the first index corresponding to the first book all the way up to
% 80.
ave2Score = zeros(1,80);
for index = 1:80
    ave2Score(index) =  ((ChristenRatings(index)^2 + NatRatings(index)^2 +
                            EmilyRatings(index)^2 + EmmaRatings(index)^2)/ 4);
end

% Creates Objective Function which will work to maximize our average ratings
% we calculated
fid = fopen('Project381square_average1.txt', 'w');
firstAve = ave2Score(1);
fprintf(fid, 'max: %.3fb1', firstAve);
for index = 2:80
    roundedAve = ave2Score(index);
    % We consider 'goodness' this rounded average of ratings
    goodness = roundedAve;
    fprintf(fid, [' + %.3fb%d'], goodness,  index);
end
fprintf(fid, ';\n');

%Writes constraint that we only have exactly 12 books
for g = 1:80

    fprintf(fid, '+ b%d', g);
end
fprintf(fid, '=12;\n');

% Everyone gets one of their books rated 10.
% This makes Christen get a rated 10 book and writes these constraints
% to a text file.
tenRatingIndexCM = zeros(1,1);
for r= 1:80
    if ChristenRatings(r) == 10
       tenRatingIndexCM = [tenRatingIndexCM r];
    end
end

tenRatingIndexCM = tenRatingIndexCM(2:end);
lengthCM = length(tenRatingIndexCM);
for i = 1:lengthCM - 1
     currentIndex = tenRatingIndexCM(i);
     fprintf(fid, 'b%d +', currentIndex);
end
```

```
lastTen = (tenRatingIndexCM(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

% Everyone gets one of their books rated 10.
% This makes Nat get a rated 10 book and writes these constraints
% to a text file.
tenRatingIndexNM = zeros(1,1);
for r= 1:80
    if NatRatings(r) == 10
       tenRatingIndexNM = [tenRatingIndexNM r];
    end
end

tenRatingIndexNM = tenRatingIndexNM(2:end);
lengthNM = length(tenRatingIndexNM);
for i = 1:lengthNM - 1
     currentIndex = tenRatingIndexNM(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexNM(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

% Everyone gets one of their books rated 10.
% This makes Emily get a rated 10 book and writes these constraints
% to a text file.
tenRatingIndexEP = zeros(1,1);
for r= 1:80
    if EmilyRatings(r) == 10
       tenRatingIndexEP = [tenRatingIndexEP r];
    end
end

tenRatingIndexEP = tenRatingIndexEP(2:end);
lengthEP = length(tenRatingIndexEP);
for i = 1:lengthEP - 1
     currentIndex = tenRatingIndexEP(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEP(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

% Everyone gets one of their books rated 10.
% This makes Emma get a rated 10 book and writes these constraints
% to a text file.
```

```
tenRatingIndexEL = zeros(1,1);
for r= 1:80
    if EmmaRatings(r) == 10
       tenRatingIndexEL = [tenRatingIndexEL r];
    end
end

tenRatingIndexEL = tenRatingIndexEL(2:end);
lengthEL = length(tenRatingIndexEL);
for i = 1:lengthEL - 1
     currentIndex = tenRatingIndexEL(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEL(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Makes sure only 1 of each author is chosen
%for J.K.Rowling
%We know that 61-68 we only want 1.
fprintf(fid, 'b61 + b62 + b63 + b64 + b65 + b66 + b67 +
                                                b68 <= 1;\n');

%writes binary variables
fprintf(fid, 'bin ');
for h = 1:79
   fprintf(fid, ['b' '%d, '], h);
end
fprintf(fid, ['b' '80;']);

fclose(fid);
```

The following code is used to determine the objective function in the median solution. All else is the same as the squares code

```
%Takes the median of the ratings for each book and puts them in a vector
%'medScore'
medScore = zeros(1,80);
for index = 1:80
    scoresForBook = [ChristenRatings(index) NatRatings(index)
                                    EmilyRatings(index) EmmaRatings(index)];
    medScore(index) = median(scoresForBook);
end

%Creates Objective Function
```

```
fid = fopen('Project381median.txt', 'w');
firstAve = medScore(1);
fprintf(fid, 'max: %.3fb1', firstAve);
for index2 = 2:80
    roundedAve = medScore(index2);
    goodness = roundedAve;
    fprintf(fid, [' + %.3fb%d'], goodness,  index2);
end
fprintf(fid, ';\n');
```

The following code is used to determine the objective function in the mean solution. All else is the same as the squares code

```
% Averages rating scores and puts them in a vector named aveScore,
% with the first index corresponding to the first book all the way up to
% 80.
aveScore = zeros(1,80);
for index = 1:80
    aveScore(index) = ((ChristenRatings(index) + NatRatings(index)
                        + EmilyRatings(index)+EmmaRatings(index))/ 4);
end

% Creates Objective Function which will work to maximize our average ratings
% we calculated
fid = fopen('Project381Average.txt', 'w');
firstAve = aveScore(1);
fprintf(fid, 'max: %.3fb1', firstAve);
for index2 = 2:80
    roundedAve = aveScore(index2);
    % We consider 'goodness' this rounded average of ratings
    goodness = roundedAve;
    fprintf(fid, [' + %.3fb%d'], goodness,  index2);
end
fprintf(fid, ';\n');
```

The code for maximizing Nat's 10s

```
%% Average the square of each score and Maximize one persons 10s
clear all; close all; clc;


%imports an excel file myExample2.xlsx
filename = 'myExample2.xlsx';
```

```
%Creates vector of Christen Rating's from Excel
ChristenRatings = xlsread(filename,'C:C');
%Creates vector of Nat's Ratings from Excel
NatRatings = xlsread(filename,'D:D');
%Creates vector of Emily's Ratings from Excel
EmilyRatings = xlsread(filename,'E:E');
%Creates vector of Emma's Ratings from Excel
EmmaRatings = xlsread(filename,'F:F');



fid = fopen('Project381VariationMaxTensFinal.txt', 'w');

% This finds all the books that Nat rated a 10 and saves it to a vector
% called tenRatingIndexNM
tenRatingIndexNM = zeros(1,1);
for r= 1:80
    if NatRatings(r) == 10
       tenRatingIndexNM = [tenRatingIndexNM r];
    end
end

% This sets up an objective function maximizing the number of books Nat rated a ten
fprintf(fid, 'max:');
tenRatingIndexNM = tenRatingIndexNM(2:end);
lengthNM = length(tenRatingIndexNM);
for i = 1:lengthNM - 1
     currentIndex = tenRatingIndexNM(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexNM(end));
fprintf(fid, 'b%d;\n', lastTen);


% For each book, this sums the squares of each rating the four of
% us gave to that book,
% and then takes the average and saves it to a vector ave2Score.
ave2Score = zeros(1,80);
for index = 1:80
    ave2Score(index) =  ((ChristenRatings(index)^2 + NatRatings(index)^2 +
                            EmilyRatings(index)^2 + EmmaRatings(index)^2)/ 4);
end

% This uses that previously mentioned ave2Score vector of averages and writes out a
```

```
% constraint set to make sure that the sum of the products of each book
% multiplied by their corresponding weighted average. This was formerly the
% objective function that maximized 'goodness', but in this scenario is
% given the value of 800 at the minimum to ensure better choices.
firstAve = ave2Score(1);
fprintf(fid, ' %.3fb1', firstAve);
for index = 2:80
    roundedAve = ave2Score(index);
    goodness = roundedAve;
    fprintf(fid, [' + %.3fb%d'], goodness,  index);
end
fprintf(fid, '>= 800;\n');

%Writes Constraint for exactly 12 books
for g = 1:80

    fprintf(fid, '+ b%d', g);
end
fprintf(fid, '=12;\n');


% Everyone gets one of their books rated 10.
% This makes Christen get a rated 10 book and writes these constraints
% to a text file.
tenRatingIndexCM = zeros(1,1);
for r= 1:80
    if ChristenRatings(r) == 10
       tenRatingIndexCM = [tenRatingIndexCM r];
    end
end

tenRatingIndexCM = tenRatingIndexCM(2:end);
lengthCM = length(tenRatingIndexCM);
for i = 1:lengthCM - 1
     currentIndex = tenRatingIndexCM(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexCM(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);


% Everyone gets one of their books rated 10.
% This makes Emily get a rated 10 book and writes these constraints
% to a text file.
```

```
tenRatingIndexEP = zeros(1,1);
for r= 1:80
    if EmilyRatings(r) == 10
       tenRatingIndexEP = [tenRatingIndexEP r];
    end
end

tenRatingIndexEP = tenRatingIndexEP(2:end);
lengthEP = length(tenRatingIndexEP);
for i = 1:lengthEP - 1
     currentIndex = tenRatingIndexEP(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEP(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);



% Everyone gets one of their books rated 10.
% This makes Emma get a rated 10 book and writes these constraints
% to a text file.
tenRatingIndexEL = zeros(1,1);
for r= 1:80
    if EmmaRatings(r) == 10
       tenRatingIndexEL = [tenRatingIndexEL r];
    end
end

tenRatingIndexEL = tenRatingIndexEL(2:end);
lengthEL = length(tenRatingIndexEL);
for i = 1:lengthEL - 1
     currentIndex = tenRatingIndexEL(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEL(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Makes sure only 1 of each author is chosen
%for J.K.Rowling
%We know that 61-68 we only want 1.
fprintf(fid, 'b61 + b62 + b63 + b64 + b65 + b66 + b67 + b68 <= 1;\n');

%Creates Vector of Low Indices, i.e, a vector of indices corresponding to
%books that any of us rated 'low' (5 or below). This ensures that no really
```

```
%low book is chosen and is needed now in this variation since we no longer
% maximizing the 'goodness' function.
newIndex = 1;
lowindex = zeros(1,1);
for index = 1:80
    if(EmmaRatings(index) < 6 | EmilyRatings(index) < 6 | NatRatings(index) < 6 |
                                            ChristenRatings(index) < 6)
      lowindex(newIndex) = index;
      newIndex = newIndex + 1;
    end
end
lowIndexLength = length(lowindex);

%Writes a constraint making it so none of these low-index books show up in
%our list.
for j = 1: lowIndexLength
    currentIndex = lowindex(j);
    fprintf(fid, '+ b%d', currentIndex);
end
fprintf(fid, '= 0;\n', currentIndex);

%Writes binary variables constraint
fprintf(fid, 'bin ');
for h = 1:79
   fprintf(fid, ['b' '%d, '], h);
end
fprintf(fid, ['b' '80;']);

fclose(fid);
```

The code for minimizing price

```
%% Minimizing the price and Constraint on 'goodness'>=650
clear all; close all; clc;

filename = 'myExample2.xlsx';

% initializes what columns of rankings corresponds to who from the excel file.
ChristenRatings = xlsread(filename,'C:C');
NatRatings = xlsread(filename,'D:D');
EmilyRatings = xlsread(filename,'E:E');
EmmaRatings = xlsread(filename,'F:F');
Prices = xlsread(filename,'G:G');
ChristensPicks = 1:1:20;
```

```
NatsPicks = 21:1:40;
EmmasPicks = 41:1:60;
EmilysPicks= 61:1:80;

ave2Score = zeros(1,80);
for index = 1:80
    ave2Score(index) =  ((ChristenRatings(index)^2 + NatRatings(index)^2 +
    EmilyRatings(index)^2 + EmmaRatings(index)^2)/ 4);
end

% Writes the objective function that the sum of the prices of the
% books chosen be minimized
fid = fopen('minimizePrice.txt', 'w');
firstPrice = Prices(1);
fprintf(fid, 'min: %.3fb1', firstPrice);
for index = 2:80
    roundedPrice = Prices(index);
    price = roundedPrice;
    fprintf(fid, [' + %.3fb%d'], price,  index);
end
fprintf(fid, ';\n');

% Writes the constraint that we have exactly 12 books
for g = 1:80
    fprintf(fid, '+ b%d', g);
end
fprintf(fid, '=12;\n');

% Iterates through Christen's ratings and keeps track of
% which books she rated 10
tenRatingIndexCM = zeros(1,1);
for r= 1:80
    if ChristenRatings(r) == 10
       tenRatingIndexCM = [tenRatingIndexCM r];
    end
end
tenRatingIndexCM = tenRatingIndexCM(2:end);
lengthCM = length(tenRatingIndexCM);

% Writes the constraint that the solution must contain at least
% one of the books Christen rated 10
for i = 1:lengthCM - 1
     currentIndex = tenRatingIndexCM(i);
     fprintf(fid, 'b%d +', currentIndex);
```

```
end
lastTen = (tenRatingIndexCM(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Ensures the same thing for Nat
tenRatingIndexNM = zeros(1,1);
for r= 1:80
    if NatRatings(r) == 10
       tenRatingIndexNM = [tenRatingIndexNM r];
    end
end
tenRatingIndexNM = tenRatingIndexNM(2:end);
lengthNM = length(tenRatingIndexNM);
for i = 1:lengthNM - 1
     currentIndex = tenRatingIndexNM(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexNM(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Ensures the same thing but for Emily
tenRatingIndexEP = zeros(1,1);
for r= 1:80
    if EmilyRatings(r) == 10
       tenRatingIndexEP = [tenRatingIndexEP r];
    end
end
tenRatingIndexEP = tenRatingIndexEP(2:end);
lengthEP = length(tenRatingIndexEP);
for i = 1:lengthEP - 1
     currentIndex = tenRatingIndexEP(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEP(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Ensures the same thing but for Emma
tenRatingIndexEL = zeros(1,1);
for r= 1:80
    if EmmaRatings(r) == 10
       tenRatingIndexEL = [tenRatingIndexEL r];
    end
end
tenRatingIndexEL = tenRatingIndexEL(2:end);
```

```
lengthEL = length(tenRatingIndexEL);
for i = 1:lengthEL - 1
     currentIndex = tenRatingIndexEL(i);
     fprintf(fid, 'b%d +', currentIndex);
end
lastTen = (tenRatingIndexEL(end));
fprintf(fid, 'b%d >= 1;\n', lastTen);

%Makes sure only 1 of each author is chosen
%for J.K.Rowling
%We know that 61-68 we only want 1.
%So, we say that
fprintf(fid, 'b61 + b62 + b63 + b64 + b65 + b66 + b67 + b68 <= 1;\n');

firstAve = ave2Score(1);
fprintf(fid, '%.3fb1', firstAve);
for index = 2:80
    roundedAve = ave2Score(index);
    goodness = roundedAve;
    fprintf(fid, [' + %.3fb%d'], goodness,  index);
end
fprintf(fid, '>=650;\n');

%writes binary variables
fprintf(fid, 'bin ');
for h = 1:79
   fprintf(fid, ['b' '%d, '], h);
end
fprintf(fid, ['b' '80;']);

fclose(fid);
```

Input File for the Max 10s

```
max:b5 +b7 +b39 +b40 +b41 +b48 +b63 +b67 +b70;

(Sets 'goodness' averages multiplied by binary books greater than or equal to 800)
45.750b1 + 73.500b2 + 42.500b3 + ... + 44.500b76 + 33.750b77 + 9.750b78 + 50.500b79 + 4.

(Makes sure only 12 books are chosen)
+ b1+ b2+ b3+ ... + b79+ b80=12;
(Constraint that Christen gets at least 1 book she rated 10)
b2 +b10 +b51 +b76 >= 1;
(Constraint that Emily gets at least 1 book she rated 10)
```

```
b25 +b31 >= 1;
(Constraint that Emma gets at least 1 book she rated 10)
b20 +b30 +b31 +b61 +b67 >= 1;
(ensures that only one JK Rowling book is chosen)
b61 + b62 + b63 + b64 + b65 + b66 + b67 + b68 <= 1;
(ensures that no low rated book chosen)
+ b6+ b8+ b10+ b12+ b13+ ...+ b73+ b74+ b75+ b76+ b77+ b78+ b80= 0;
(Variables are only 1 or 0)
bin b1, b2, b3, ..., b78, b79, b80;
```

Input File for the Minimizing Price

```
min: 12.520b1 + 7.990b2 + 11.990b3 + 7.190b4 + 15.710b5 + 8.950b6 + 12.230b7 + 5.990b8 +

(used to make sure only 12 books are chosen)
+ b1+ b2+ b3+ b4+ b5+ b6+ b7+ b8+ b9+ b10+ b11+ b12+ b13+ b14+ b15+ b16+ b17+ b18+ b19+

(used to ensure Christen gets at least one book rated 10)
b2 +b10 +b51 +b76 >= 1;
(used to ensure Nat gets at least one book rated 10)
b5 +b7 +b39 +b40 +b41 +b48 +b63 +b67 +b70 >= 1;
(used to ensure Emily gets at least one book rated 10)
b25 +b31 >= 1;
(used to ensure Emma gets at least one book rated 10)
b20 +b30 +b31 +b61 +b67 >= 1;
(used to ensure only one JK Rowling book is chosen)
b61 + b62 + b63 + b64 + b65 + b66 + b67 + b68 <= 1;
(Ensures that the overall 'goodness' value is greater than 650)
45.750b1 + 73.500b2 + 42.500b3 +... + 53.750b71 + 10.750b72 + 28.750b73 + 22.500b74 + 19
(makes all variables binary, 0 or 1)
bin b1, b2, b3,...,b78, b79, b80;
```

Output File for Max 10s

```
Value of objective function: 7.00000000

Actual values of the variables:
b5                            1
b7                            1
b39                           1
b40                           1
b41                           1
b67                           1
b70                           1
```

```
b2                              1
b3                              1
b4                              1
b20                             1
b31                             1
(All other variables 0)

real    0m0.056s
user    0m0.003s
sys 0m0.010s
```

Output File for Minimizing Price

```
Value of objective function: 92.66000000

Actual values of the variables:
b2                              1
b4                              1
b8                              1
b18                             1
b20                             1
b31                             1
b39                             1
b52                             1
b54                             1
b55                             1
b57                             1
b61                             1

real    0m0.028s
user    0m0.004s
sys 0m0.006s
```

The excel file containing our scores and the prices [5] (it is on the following pages)

b41	The Stand - Stephen King	8	10	8	9	11.55
b42	The Sun Also Rises - Ernest Hemingway	7	9	8	7	14.4
b43	Justin Beiber: Just Getting Started - Justin Beiber	1	1	3	1	12.09
b44	Brave New World - Aldous Huxley	8	7	6	7	12.23
b45	Let Freedom Ring - Sean Hannity	1	1	6	1	7.33
b46	The Circle - Dave Eggers	7	5	9	8	9.57
b47	The Sellout - Paul Beatty	4	4	3	3	11.23
b48	This Book Loves You - Pewdiepie	5	10	5	5	12.47
b49	Slaughterhouse-Five - Kurt Vonnegut	8	8	7	7	13.39
b50	Fifty Shades of Grey - E.L. James	1	1	6	1	9.79
b51	Frankenstein - Mary Shelley	10	2	4	5	7.25
b52	The Catcher in the Rye - J.D. Salinger	9	8	5	9	7.19
b53	Trump: The Art of the Deal - Donald Trump	1	3	6	1	13.41
b54	The Count of Monte Cristo - Alexander Dumas	9	5	7	6	7.95
b55	The Adventures of Tom Sawyer - Mark Twain	9	7	8	9	8.73
b56	Warlight - Michael Ondaatje	6	4	8	7	18.32
b57	Northern Lights - Phillip Pullman	4	2	9	1	4.92
b58	Cherry - Nico Walker	5	4	5	6	18.32
b59	An Inconvienent Book - Glenn Beck	1	3	5	3	8.24
b60	Miles To Go - Miley Cyrus	5	3	4	1	13.86
b61	Harry Potter and the Sorcerer's Stone - J.K. Rowling	7	7	8	10	7.2
b62	Harry Potter and the Chamber of Secrets - J.K. Rowling	8	6	8	8	9.47
b63	Harry Potter and the Prisoner of Azkaban - J.K. Rowling	7	10	7	7	7.39
b64	Harry Potter and the Goblet of Fire - J.K. Rowling	7	8	8	7	10.29
b65	Harry Potter and the Order of the Pheonix - J.K. Rowling	7	5	9	6	6.87
b66	Harry Potter and the Half-Blood Prince - J.K. Rowling	9	9	8	8	10.39
b67	Harry Potter and the Deathly Hallows - J.K. Rowling	7	10	7	10	13.49
b68	Fantastic Beasts and Where To Find Them - J.K. Rowling	6	3	5	9	11.77
b69	The Lightning Thief - Rick Riordan	7	4	5	7	22.5
b70	The Lord of the Rings - J. R. R. Tolkien	6	10	6	8	15.99
b71	The Martian- Andy Weir	7	6	7	9	8.99
b72	Twilight - Stephanie Meyer	5	1	4	1	10.19
b73	Confessions of a Shopaholic - Sophie Kinsella	7	1	4	7	12.19
b74	Lord of the Flies - William Golding	2	6	5	5	10.87
b75	Crazy Rich Asian - Kevin Kwan	3	4	6	4	12
b76	The Girl on the Train- Paula Hawkins	10	2	5	7	10.5
b77	The World Is Flat - Thomas Friedman	7	6	7	1	12.05
b78	The Hive Queen - Tui T. Sutherland	1	3	5	2	13.59
b79	Every Breath - Nicholas Sparks	6	6	7	9	20.5
b80	Queen of Air and Darkness - Cassandra Clare	2	1	3	2	15.23

	Book	Christen's ratings	Nat's ratings	Emily's ratings	Emma's ratings	Price
b1	Death on the Nile-Agatha Christie	7	6	7	7	12.52
b2	The Princess Bride -William Goldman	10	8	7	9	7.99
b3	My Name is Asher Lev-by Chaim Potok	7	6	6	7	11.99
b4	Murder on the Orient Express-Agatha Christie	6	7	7	6	7.19
b5	11/22/63 - Stephen King	8	10	8	8	15.71
b6	Flowers for Algernon - Keyes	6	5	4	2	8.95
b7	East of Eden-Steinbeck	7	10	8	7	12.23
b8	Richard II-Shakespeare	7	2	4	2	5.99
b9	A Gentleman in Moscow-Amor Towles	9	6	7	7	18.9
b10	Rules of Civility - Amor Towles	10	5	7	6	12.8
b11	The Razor's Edge - W.Sommerset Maugham	9	7	7	6	13.19
b12	Franny and Zooey- J.D Salinger	6	7	5	5	7.2
b13	Casino Royale -Ian Flemming	6	5	8	5	11.99
b14	Gone Girl -Gillian Flynn	6	3	5	6	10.89
b15	In Pieces (The birography of the actress Sally Field)	7	5	4	4	19.72
b16	Alexander Hamilton -(by Ron Chernow)	7	8	6	4	18
b17	Frarenheit 451-Ray Bradbury	7	8	3	7	8.99
b18	Alice in Wonderland-Lewis Carroll	6	4	4	4	3
b19	One Hundred Years of Solitude - Marquez	6	3	5	4	11.39
b20	1984-George Orwell	8	9	7	10	7.49
b21	Crucible - James Rollins	7	4	8	7	18.76
b22	Liar Liar - James Patterson	5	7	7	6	16.99
b23	Turning Point - Danielle Steel	5	4	8	7	19.96
b24	An Anonymous Girl - Greer Hendricks	8	4	9	6	16.14
b25	The New Iberia Blues - James Lee Burke	9	6	10	7	23.52
b26	The Truths We Hold: An American Journeys - Kamala Harris	2	6	3	2	20.56
b27	King of Scars - Leigh Bardugo	7	4	9	8	21.63
b28	Two Can Keep a Secret - Karen M. McManus	6	3	8	6	12.12
b29	96 Words for Love - Rachel Roy	5	2	7	7	12.32
b30	The Wicked King - Holly Black	6	7	9	10	15.99
b31	We are Displaced - Malala Yousafzai	8	6	10	10	14.82
b32	The Gilded Wolves - Roshani Chokshi	8	7	8	8	12.91
b33	Love and Ruin - Paula McLain	7	8	7	6	11.59
b34	Becoming - Michelle Obama	9	3	9	9	21.21
b35	The Life-Changing Magic of Tidying Up - Marie Kondo	9	8	7	2	8.79
b36	Where the Crawdas Sing - Delia Owens	8	6	9	9	25.2
b37	Educated - Tara Westover	7	7	9	8	18.08
b38	You: A Novel - Caroline Kepnes	8	6	8	8	11.96
b39	The Tattooist of Auschwitz - Heather Morris	8	10	9	9	10.19
b40	The Hate U Give - Angie Thomas	7	10	9	9	12.42

Bibliography

1. "George Dantzig." Wikipedia. December 09, 2018. Accessed February 06, 2019. https://en.wikipedia.org/wiki/George_Dantzig.

2. Corné Van Dooren. "A Review of the Use of Linear Programming to Optimize Diets, Nutritiously, Economically and Environmentally." June 21, 2018. Accessed February 5, 2019. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6021504/.

3. Chun-Cheng Lin, Jia-Rong Kang, Wan-Yu Liu, and Der-Jiunn Deng. "Modelling a Nurse Shift Schedule with Multiple Preference Ranks for Shifts

and Days-Off." Mathematical Problems in Engineering 2014 (March 2014). https://www.researchgate.net/publication/274920678_Modelling_a_Nurse_Shift_Schedule_with_Multiple_Preference_Ranks_for_Shifts_and_Days-Off.

4. Lin Xiao, Zhang Min, Zhang Yongfeng, Ma Shaoping, Liu Yiqun, and Gu Zhaoquan. "Figure 2f From: Irimia R, Gottschling M (2016) Taxonomic Revision of Rochefortia Sw. (Ehretiaceae, Boraginales). Biodiversity Data Journal 4: E7720. Fairness-Aware Group Recommendation with Pareto-Efficiency, August 27, 2017. https://cseweb.ucsd.edu/classes/fa17/cse291-b/reading/p107-xiao.pdf.

5. Amazon. https://amazon.com.