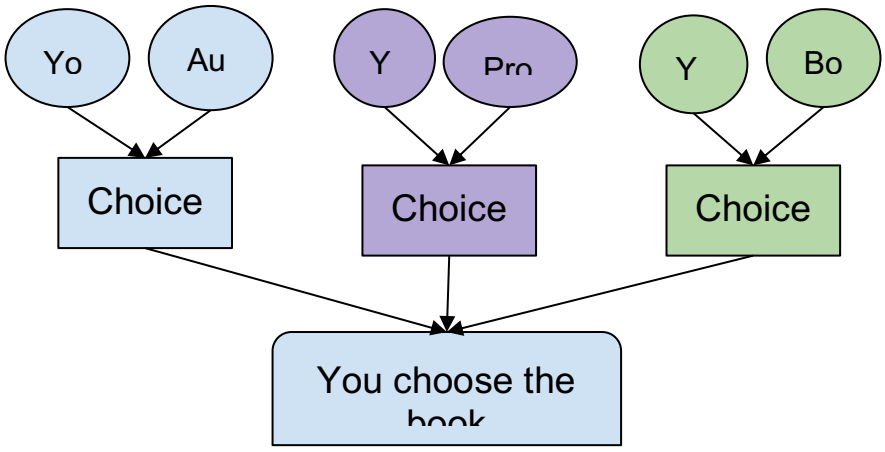
Extra Credit Contest Submission
Valexa Orelien

For this contest, I decided to create a website called The Book Matcher, where people of color can have a diverse set of books selected for them! As a Black woman who loves to read, I've often noticed how rare it is that books featuring characters like me are recommended to me by sites like Goodreads, so I decided to make a book recommender that takes in information about a users race and their favorite genre of book, and then matches them with a list of books that are generated by conditional probabilities that they can either decide to "match with" or discard -- Tinder but for books! For this project, I had to create a lot of my own data in order to have a diverse set of books, and I surveyed people to find data as well.

This program and website works by reading in basic user data from a form: Their race, their favorite genre of book, and how they would like to be matched.*

There are two matching algorithms that I came up with that are both based on prior research, where ultimately I am finding P(user likes books| A = 1, B = 1, and C = 1), where A = author race and user race are the same, B = protagonist race and user race are the same, and C = book genre and users chosen genre are the same, and then returning a ranked list of books from highest probability to lowest.

**Matching Algorithm 1: (bookspython.py)**
The first way I decided to do this was using this decision tree:



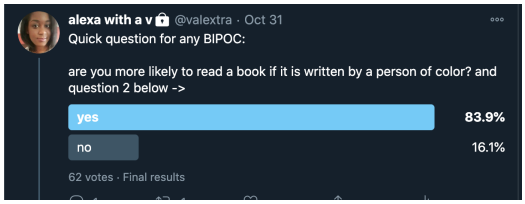

What this means is that I found the probabilities of Choice A = 1, Choice B = 1, and Choice C = 1, and from there inferred you liking the book. The way that each choice is calculated is by knowing that there are two choices: either your details (your race and genre) and the current book details (race and genre) are equal, or they are not. Let's look at choice A. If your race and the author's race are equal, there is a probability that you will like the book given A or not, i.e. this can be represented by a Bernoulli variable. I asked my Twitter followers and they gave me

these p values, which I used as the p values for a Bernoulli to either return 0 or 1. From the polls, I found that:

- P(read a book | same race as author) = .839
- P(read a book | same race as protagonist) = .885

Once I had the values for each choice, I then ranked the probability of you choosing the book as such:

- All 3 choices are 1
- Only 2 choices are equal to 1
- Only 1 choice is equal to 1
- None of the choices equal 1




**Matching Approach 2: (condProbability.py)**

This approach is based on finding P(choose the book | A = 1, B = 1, and C=1) at the same time by using the frequencies of all the possible conditional probabilities and then ranking them. I used a similar approach as we did to find conditional probabilities in pset2. I thought it was important to try this approach because then I could base it on real data where people were required to choose a book only given certain details.

I made a survey (that sadly only got 5 responses, so my data was not as robust as I would have wanted) where I asked people whether or not they would choose a book given the authors race, the protagonist's race, and the book's genre**. From there, I went through each user and each book, and added that data to a table, so each books entry would have this information:

[Author race same, Protag race same, Genre race same, Liked the book]
each entry being either a 1 for "same" or a 0 for different, and then a 1 if they liked the book.

I then iterated through this data table and found the different conditional probabilities observed through frequencies in the tables (i.e. found the probability of P(Choose book | Choice A, B, and C = 1), P(Choose book | only Choice A and C = 1), etc), which I stored in a list.

Then, I got the users information, and went through each book in the overall book dataset to figure out what characteristics were the same or not, and then tagged that book with a probability from the prior information we had acquired in our list. I then return a dictionary of book titles, sorted from highest to lowest probability of liking the book.

**Footnotes:**

*(I am still working on fixing the backend of this, so at the moment it is hardcoded to certain values).

** This actually gave me pretty surprising results, because after parsing the data, I found the highest probability of them liking the book was when only the books genre was the same. However, I asked this on a limited set of books that were kind of obscure, so it is possible that people just said no because there was not enough information.