An Offer you Cannot Refuse? Trends in the Coerciveness of Amazon Book Recommendations

[redacted]

Abstract

Recommender systems can be a helpful tool for recommending content but they can also influence users' preferences. One sociological theory for this influence is that companies are incentivised to influence preferences to make users easier to predict and thus more profitable by making it harder to change preferences. This paper seeks to test that theory empirically. The paper uses *Barrier-to-Exit*, a metric for how difficult it is to change preferences, to analyse a large dataset of Amazon Book Ratings from 1998 to 2018. I focus the analysis on users who have changed preferences according to Barrier-to-Exit. I create a linear mixed-effects model with crossed effects for users and categories to assess the change in Barrier-to-Exit across time. I find a small but highly significant effect, indicating that it has indeed become harder to change preferences for the subset of users - albeit with considerable statistical and methodological caveats. I discuss the strength and limitations as well as the implications of these findings. The paper highlights the challenges of creating context-sensitive *and* generalisable measures for complex socio-technical concepts like "difficulty to change preferences". I conclude with a call for research: to curb the potential threats of preference manipulation, we need more measures that allow us to compare different types of systems.

1 Introduction

What role do recommender systems play in shaping our behaviour? On the one hand, they can seem like a mere convenience: they help us select which music to listen to (Millecamp, Htun, Jin, & Verbert, 2018) or which television show to watch (Bennett & Lanning, 2007). When we provide feedback by liking, rating, buying, or interacting with a product, we hope that our actions help the recommender system "learn" our preferences (Knijnenburg, Reijmer, & Willemsen, 2011).

However, what if the recommender systems do not simply *learn* our preferences but *shape* them? By providing recommendations, the recommender systems can influence the products we engage with, which can shape our preferences (Jiang, Chiappa, Lattimore, György, & Kohli, 2019). This creates a feedback loop that can degenerate into so-called filter bubbles and echo chambers (Jiang et al., 2019).

The drivers of this could be commercial. The companies behind the recommender systems might have incentives for shaping our behaviour to increase profitability. This is what Zuboff terms the *prediction imperative* in her exposition of *Surveillance capitalism* (Zuboff, 2019). The prediction imperative states that to secure revenue streams Big Tech companies must become better at predicting the needs of their users. The first step of this is creating better predictive algorithms i.e. going from simple heuristics to sophisticated machine learning (Raschka, Patterson, & Nolet, 2020). However, as competition increase, the surest way to *predict* behaviour is to *shape* it (Zuboff, 2019). By shaping behaviour, companies increase predictability at the cost of the users' autonomy (Varshney, 2020).

While it may be good business, changing preferences could plausibly count as manipulation. Apart from harming the autonomy of the users (Varshney, 2020), this could have legal implications under the EU AI Act (Franklin, Ashton, Gorman, & Armstrong, 2022; Kop, 2021).

Take the case of Amazon. In 1998, Amazon introduced item-based *collaborative filtering* (Linden, Smith, & York, 2003) - a simple and scalable recommender model that allows them to recommend similar items. Since then their models have evolved to create more personalised features using machine learning on sophisticated features (Smith & Linden, 2017). The effects of this have been more accurate recommendations and - crucially - higher sales (Wells, Danskin, & Ellsworth, 2018).

Following the prediction imperative, the evolution of Amazon recommender systems should have made it more difficult for users to change preferences to make them more predictable and profitable. They might also steer users towards specific categories that are relatively more profitable (Zhu & Liu, 2018).

To make such a claim it is essential to have methods for empirically analysing potential manipulation. Fortunately, (Rakova & Chowdhury, 2019) provide such a measure: *Barrier-to-Exit*. Barrier-to-Exit provides a measure for how much effort a user must exert to show that they have changed their preferences within a given category. It is built on a theoretical foundation of Selbst et al. (2019)'s work on "traps" as well as *systems control theory* as applied to recommender systems (Jiang et al., 2019). The authors posit that recommender systems with a higher Barrier-to-Exit are more coercive.

Methods are ineffective without relevant data to apply them to. This may seem like an insurmountable task: Amazon's recommender system is a complex model that builds on a myriad of advanced features including browsing activity, item features, and buying activity (Smith & Linden, 2017). No one but Amazon has access to this data - and they are unlikely to share it.

Fortunately, we can get around this by relying on proxies. Specifically, we can use publicly available ratings as a proxy for user input. This has the advantage of being accessible through public datasets (Ni, Li, & McAuley, 2019). The disadvantage is that we only have access to a (biased) fraction of the data going into the recommender system.

This paper aims to investigate whether Amazon's recommender system has made it more difficult to change preferences over time. To focus the scope, I will only investigate book recommendations, as these were the initial product of Amazon (Smith & Linden, 2017).

- **RQ1:** Has the Amazon Book Recommender System made it more difficult to change preferences over time?

The motivation for RQ1 follows from our operationalisation of "difficult to change preferences" as having a high Barrier-to-Exit.

First, I will formalise Barrier-to-Exit in the context of Amazon book recommendations. I will discuss the caveats of the technique and how it relates to preference change. Then I will use a large dataset of Amazon book recommendations to calculate the *Barrier-to-Exit* for users who have changed their preferences. I will then analyse the change in *Barrier-to-Exit* over time. Finally, I will discuss the implications of these results.

This paper has two main contributions to the literature: 1) it provides a novel analysis of trends in preference manipulation in a *commercial* rather than *academic*. 2) it assesses the portability of Barrier-to-Exit as a measure for real-world datasets.

1.1 Previous Literature

Several papers have highlighted the need to protect human autonomy and preferences in recommender systems. However, these focus more on the *normative* need to do so, rather than a *descriptive* analysis. With this paper, I aim to fill this gap in the literature.

Most previous empirical analysis of preference manipulation in recommender systems has focused on the MovieLens-dataset (Harper & Konstan, 2016). MovieLens is a movie recommendation platform developed and maintained by the University of Minnesota[1] The advantages of this dataset are that it is a) rigorously documented as it is maintained by an academic group; b) freely available; and c) well-structured making analysis easier. However, because it is a non-commercial project it is not susceptible to surveillance capitalistic imperatives to the same extent as e.g. Amazon (Zuboff, 2019).

(Nguyen, Hui, Harper, Terveen, & Konstan, 2014) analysed whether MovieLens users were exposed to less diverse content over time. While they found a significant (albeit small) decrease, the effect was smaller for users who followed the recommendations than the users who did not. However, as discussed by Rakova and Chowdhury (2019), they focus their analysis on highly-active and highly-nonactive users neglecting the middle. Also, content diversity is an important but incomplete measure of preference manipulation.

Rakova and Chowdhury (2019) also use the MovieLens dataset to define and showcase Barrier-to-Exit. However, their paper is more of a proof-of-concept rather than an analysis. The present paper expands on Rakova and Chowdhury (2019) by investigating a real-world dataset of a commercial context.

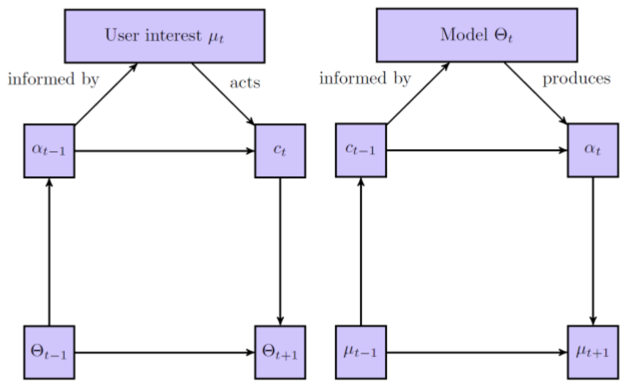
Methods: The core of the paper. What is the data? What are the methods? Why do the methods suit the question? Reading this should be enough to reproduce the results.

2 Methods

2.1 Defining Barrier-to-Exit

On a high level, *Barrier-to-Exit* measures how much effort users must expend to signal that their preferences have changed (Rakova & Chowdhury, 2019). It is defined in terms of how quickly users' *revealed preferences* for a specific category change between *interaction thresholds*. In this section, I will motivate the intuition as well as formalise the concept within the context of Amazon's recommender system.

[1] movielens.org



(a) User Supervisor (b) AI Model Supervisor

Figure 1: A schematic representation of the control flow in Recommender Systems as seen from the user (a) and AI model (b) perspective. Adapted from Rakova and Chowdhury (2019)

.

To understand the role of *Barrier-to-Exit* and how it can be calculated from ratings, let us consider a diagram of the interaction between the user and recommender system (”AI Model”) as seen in figure 1.

Both diagrams (a) and (b) show supervisory loops with the user and the model, respectively, as supervisors. The juxtaposition shows the double-sided interaction as argued in (Jiang et al., 2019). The diagram has multiple elements: μ is user interest, Θ is the Model, α is the shown recommendations, and c is the revealed preferences (i.e the signal the model uses to update recommendations).

While the diagram acts as a conceptual framework for understanding the interaction, we must consider which parts we can measure and which parts we need to model. (Rakova & Chowdhury, 2019) argue that by only analysing how revealed preferences change over time, we can calculate a measure of the effort required to shift preferences; the Barrier-to-Exit.

Note that while the overall feedback loop concerns the whole model, Barrier-to-Exit is defined per category. Categories can be genres, such as ”Thriller” or ”Science Fiction”, or book types such as ”Self-help” or ”Cook Book”. Each book can have several categories.

However, we still need to calculate revealed preferences based on ratings. We calculate revealed preference for a given category using ratings and the assigned categories. Following Rakova and Chowdhury (2019), let c_t^i be a user's revealed preference for category i at time t. We can calculate this using the following formula:

$$c_t^i = \sum_{j=1}^{n} m_{tj}^i \cdot r_{tj} \tag{1}$$

Here, n is the number of books the user has rated within the time frame; r_{tj} is the rating for book j; and m_{tj}^i is the *category-relevance* between the book, j, and the category, i. The cateogory-relevance can be understood as how well a book fits within a given category.

The category-relevance is not an automatically available feature of our data (see 2.2). In contrast, Rakova and Chowdhury (2019) use the MovieLens dataset (CITATION), where category-relevance has been manually annotated for a subset of the data. This makes it possible to use (semi-)supervised learning to annotate the rest of the data (i.e. Kipf & Welling, 2017).

Unfortunately, the Amazon data has no such labels. Instead, I rely on an unsupervised approach using category co-occurrence. Books are given a high category-relevance for a given category if they belong to categories that often co-occur with that category. Imagine a book has the categories ”thriller” and ”horror”. If ”thriller” always occurs together with ”horror”, the book would get a category-relevance to ”thriller” of 1. However, if neither occurs with, say, ”gardening”, the book would get a category-relevance score of 0 for gardening. I normalise the values, so *category-relevance* is between zero and one. To see the definition in code see the `main`-function in appendix C.1.

REMEMBER TO CTRL+F aeznote + XXX + YYY

Now that we have defined preferences, we can define *interaction thresholds* (Rakova & Chowdhury, 2019). Conceptually, interaction thresholds are the users' range of preferences within a given category. If, say, a user only ever rates thrillers 4 stars but rate some cookbooks 1 star and others 5 stars, they would have narrow interaction thresholds for thrillers and broader interaction thresholds for cookbooks.

We define the category-specific upper threshold (X_t^i) and lower threshold (Y_t^i) for a category, i, using a rolling window of size v as follows:

$$X_t^i = mean(c_{t-v}, \dots, c_t) + 2 * std(c_{t-v}, \dots, c_t) \tag{2}$$
$$Y_t^i = mean(c_{t-v}, \dots, c_t) - 2 * std(c_{t-v}, \dots, c_t) \tag{3}$$

Note that time is defined not in discrete steps but in periods. This is because there can be an arbitrary amount of time between two ratings.

The interaction thresholds are then the mean of category interaction thresholds over all available categories. Conceptually, the interaction thresholds model the users expected rating behaviour as seen by the model (because of the link going from c_{t-1} to Ω_t in fig. 1. (Rakova & Chowdhury, 2019) posit that the interaction thresholds for non-coercive systems should adapt to make it easier for users to change preferences.

Finally, we can calculate the barrier to exit for a given category. This is defined as the sum of preferences that fall between ratings *above* (t_X) the thresholds and ratings *below* (t_Y) the thresholds. That is:

$$BtE_{t_y}^i = \sum_{\tau \in (t_x, t_y)} c_\tau^i \quad \text{such that} \quad Y_\tau < c_\tau^i < X_\tau \tag{4}$$

There are some important things to note about the definition of Barrier-to-Exit. First, there can be multiple values of Barrier-to-Exit per user and category. Every time a user has a preference within a category that goes from above the interaction thresholds to below, a Barrier-to-Exit for that time is defined.

Second, Barrier-to-Exit defines users who change preferences. Changing preferences are defined as users going from above the interaction thresholds to below the interaction thresholds.

Third, Barrier-to-Exit cannot be exactly zero. This is because it is only defined when a user has intermediate ratings between the thresholds. If a user has a rating that goes above the interaction thresholds and the next one is below, this would not register in Barrier-to-Exit.

Finally (and crucially), Barrier-to-Exit is only defined for a subset of users. Having a well-defined Barrier-to-Exit requires both a) enough ratings and b) that these ratings change relative to a category. We can thus only draw inferences for this subset of users. I will discuss the implications of this further in the discussion (section 4.2).

In this section, I have provided a mathematical formulation of Barrier-to-Exit along with important caveats. For the code implementation, please refer to appendix C.2.

2.2 Data

For this analysis, I use a dataset of Amazon book reviews (Ni et al., 2019). The raw dataset consists of approximately 51 million ratings by ca. 15 million users in the period 1998 to 2018[2]. All the ratings are on a 1-5 Likert scale.

The dataset was scraped from the Amazon Web Store building on the methodology of CITATION. Unfortunately, since the dataset lacks a datasheet (Gebru et al., 2021), it is difficult to figure out whether it has any issues with coverage or bias. It also makes it harder to replicate the data collection from scratch. Other than that, the dataset is easily accessible and well documented.

One coverage-related aspect we need to be aware of is that we are using *ratings* as a proxy for *interactions*. In the dataset, we do not have access to people who bought a product but did not rate it, nor people who neither bought a product nor rated it. This gives us quite an indirect measure of the actual recommendation process - particularly compared to the MovieLens dataset (Harper & Konstan, 2016; Rakova & Chowdhury, 2019).

While the original dataset is huge, we are only interested in a subset. Specifically, we are interested in users who have changed their preferences. Therefore, we filter to only include users with more than 20 ratings, which follows the conventions in MovieLens (Harper & Konstan, 2016) for which Barrier-to-Exit was originally defined (Rakova & Chowdhury, 2019).

[2]For documentation see: https://nijianmo.github.io/amazon/index.html

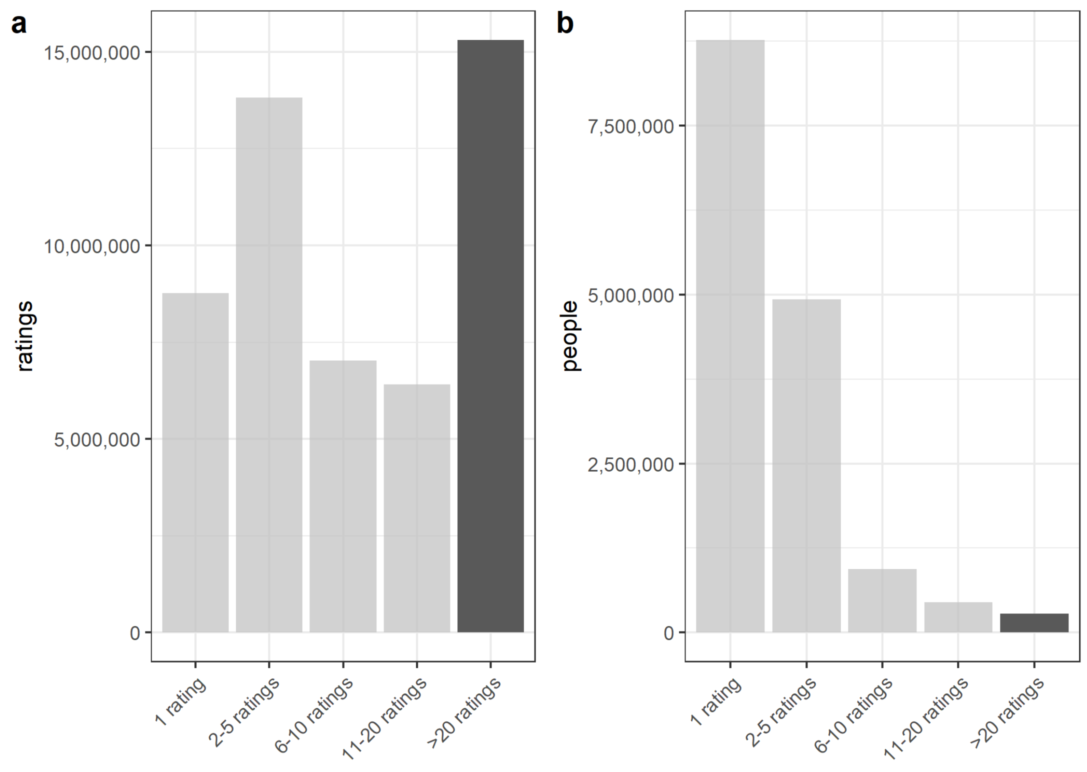


Figure 2: Demographic

Fig. 2 shows the selected subset. It is worth noticing that while our subset retains a substantial fraction of the ratings (30%), we only retain 350,000 users (0.6%). This is typical for user activity, which tends to be fat-tailed (Papakyriakopoulos, Serrano, & Hegelich, 2020). I will discuss the implications for our interpretation in the discussion (4.2).

As we will later see, only a fraction of these has changed preferences according to our definition (see section 2.1). For our final analysis, we have 50,626 users which fit our definition (0.1% of the total).

The rating dataset was merged with a dataset providing categories for each book. The category dataset was from the same source i.e. Ni et al. (2019). To keep the computations simple for calculating category-similarity (see C.1), we only consider categories that have been used on more than 100 books. This approach is valid because the distribution of categories is heavily skewed, meaning that a small number of categories are used on a large number of books. (This is a similar dynamic to user activity; see fig. 2).

2.3 Model

Now that we have operationalised *Barrier-to-Exit* as a measure for recommender system coerciveness in the context of users changing their preferences, let us introduce the statistical model for elucidating the trend.

The first thing to note is that we need a crossed multi-level model (CITATION). Our model should have two levels: user and category. The user level is the most theoretically obvious one. Since each user can have multiple preference changes (with associated *Barrier-to-Exit*), we should control for their individual differences. This is also important as the recommender system will use predictive features that are not accessible in the dataset (Smith & Linden, 2017).

Categories constitute the other level. The role of the category level in our model is to account for item-level features. As explained in the introduction, there are commercial (i.e. companies are following the prediction imperative; (Zuboff, 2019)) and algorithmic reasons (i.e. reducing variability could improve on reward objective (Carroll, Dragan, Russell, & Hadfield-Menell, 2022)) to believe that different categories will have different Barriers-to-Exit. Categories can therefore act as a proxy for these effects.

There are two reasons to include categories as random effects and not fixed effects. The first is the number of categories. There are 300+ categories in our dataset. Modelling these as fixed effects would therefore be infeasible. Secondly, since there we use them as a proxy for item-level variance, it is more convenient to only model the random components (?, ?)

This gives us the following model:

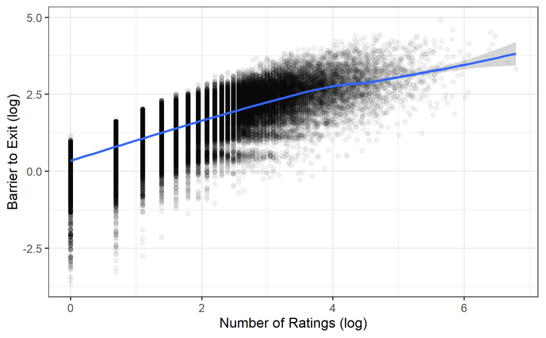
$$y_{ij} = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + e_i + e_j \tag{5}$$

Here, y_{ij} is the log-transformed Barrier-to-Exit for user i and category j; β_0 is the intercept; β_1 is the effect of the number of years since the start of the dataset (1998) measured in quarters; $beta_2$ is the *activity-level*, i.e. the total amount of ratings a user has made in the period a specific *Barrier-to-Exit*-measurement is defined; and e_i and e_j are random intercepts for the user and category, respectively.

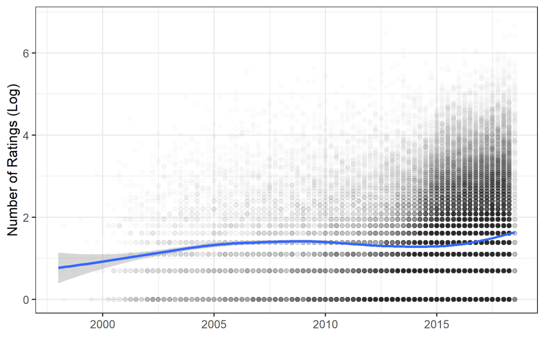
Using a log-transformation of the outcome variable introduces statistical issues (FENG et al., 2014): it can worsen skew and make the results difficult to interpret. Nevertheless, our assumption-check shows that log transforming the outcome and activity creates more homoscedastic and normally distributed residuals (see Appendix A).

Controlling for user activity (β_2) is important. Since *Barrier-to-Exit* is defined using the *sum* of ratings between the interaction thresholds, users with higher activity levels will tend to have higher Barriers-to-Exit. This can be seen in fig. 3a, which shows the relation between number of ratings and barrier-to-exit, which is increasing.

It is also worth noting that activity level is relatively uncorrelated with time (see fig. 3b. This is because activity refers to the activity *within the Barrier-to-Exit* period and not *total activity on Amazon*. The latter has increased substantially as can be seen by the density of the dots in fig. 3b.



(a)



(b)

Figure 3: Plots of the activity level, defined as the number of ratings in the period of Barrier-to-Exit. 3a: The relation between activity-level and Barrier-to-Exit. Notice, the strong linearity. 3b Change in activity-level over time. Note that while there is no strong relation between activity level and time, there has been a substantial increase in the density of points, reflecting Amazon's increased popularity (Wells et al., 2018)
.

To assess validity, I test the assumptions for the model. For the full check see A. There are a few violations worth noting: The residuals and random effects deviated from normality - particularly for the category-level random effects. However, this should have little influence on the estimation of the fixed effects (Schielzeth et al., 2020). Nevertheless, I run an additional analysis with the problematic categories removed to assess the robustness of the findings (see B.2)

2.4 Creating and testing hypotheses

To answer our research question in an inferential framework, we need to transform them into hypotheses with testable implications (Popper, 1970). I propose the following hypothesis:

- **H1**: There has been a significant *increase* in *Barrier-to-Exit* for the Amazon Book Recommender System

In order to test the hypothesis, we assess the significance of the coefficient for time (β_1). We do this with significance tests from the lmerTest-package (Kuznetsova, Brockhoff, & Christensen, 2017), which utilises a t-test with degrees of freedom calculated using the Satterthwaite method (CITATION).

It is worth noting that the method of calculating degrees of freedom in crossed effects models is controversial as might inflate Type I errors (i.e. falsely rejecting the null hypothesis) (Baayen, Davidson, & Bates, 2008). However, when the sample size is sufficiently large, the problem disappears. In our case, the sample size is large, so this is less of an issue.

To assess the model fit we use the marginal and conditional R^2_{LMM} using the `MuMIn`-package (Bartoń, 2022). This uses an improved method to calculate this from (Nakagawa, Johnson, & Schielzeth, 2017).

3 Results

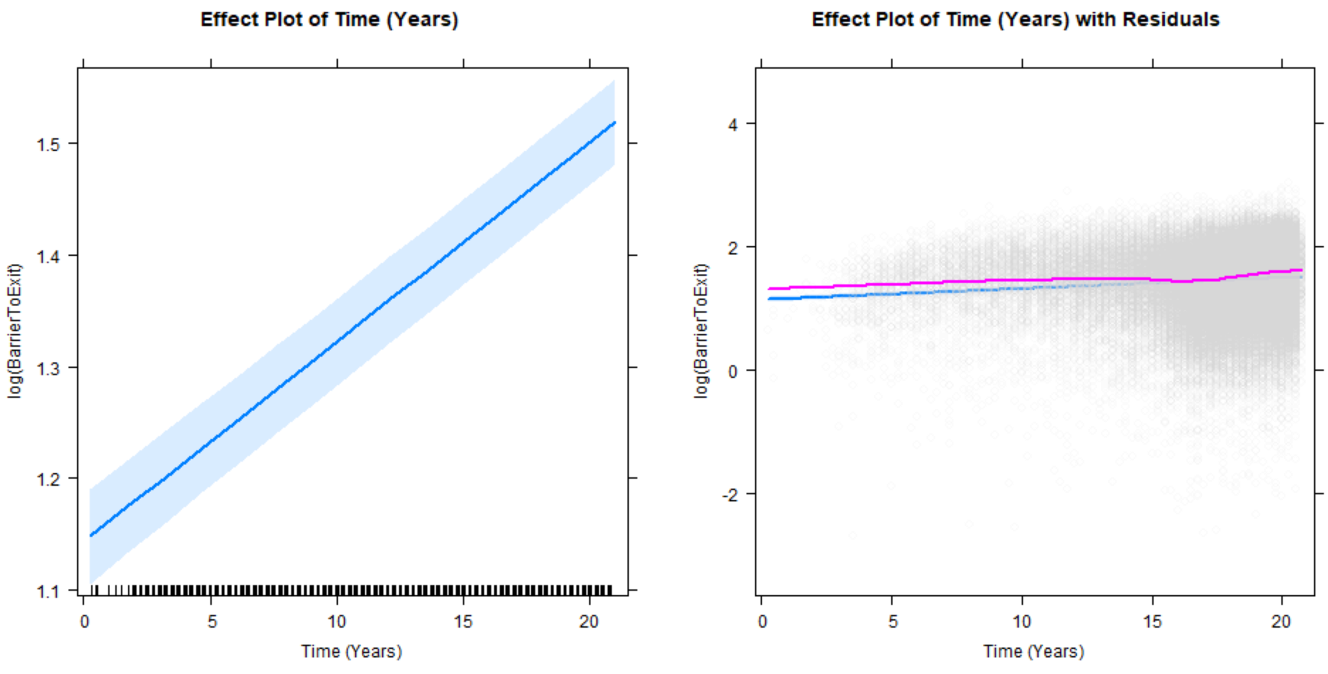
The results from the model can be seen in table 1. The coefficient for time in years is 0.018 (SE=0.001). This is highly significant (T=29.95, $p \ll 0.0001$) The coefficient for ratings is 0.614 (SE=0.001), which is also highly significant (T=450.11, $p \ll 0.0001$).

Table 1: Main Results

	Dependent variable:
	Barrier-to-Exit (log)
Time (years)	0.018***
	(0.001)
Ratings in period (Log)	0.613***
	(0.001)
Intercept	0.300***
	(0.022)
Unique Users (VPC)	51,208 (0.24)
Unique categories (VPC)	330 (0.34)
Observations	84,806
Marginal R^2	0.64
Conditional R^2	0.81
Note:	*p<0.1; **p<0.05; ***p<0.01

The marginal R^2 is 0.64, i.e. 64% of the variance is explained by the fixed effects. The conditional R^2 is 0.81, i.e. 81% of the variance is explained by the random and fixed effects combined. For the random effects, the user-level VPC is 0.24 and The category-level VPC is 0.34. These should be interpreted cautiously as discussed in section 4.2.

A visual representation of these models can be seen in figure 4.



(a) (b)

Figure 4: Effect plots for Year. 4a shows the partial effect plot. 4b shows the same but with residuals added

The partial effects plot in fig 4a shows an increase in Barrier-to-Exit from 1.15 to 1.5. This translates into a growth of approximately 40% over the duration of the study.

4 Discussion

4.1 Key Findings

Recall our research question:

- **RQ1:** Has the Amazon Book Recommender System made it more difficult to change preferences over time?

Our analysis finds a small but highly significant increase in Barrier-to-Exit over time over the period in our dataset. We can therefore reject the null hypothesis, which provides evidence that the Amazon Book Recommender system has indeed made it more difficult to change preferences. However, we must moderate the implications of this finding in light of the limitations of the study. I will discuss these in section (4.2).

While an effect size of 1.8% seems inconsequential, it should be considered in the context of the size of the dataset. Amazon has millions of users, which means that even small effects can have large impacts in aggregate. This is similar to the controversy surrounding the infamous study on emotional contagion by (Kramer, Guillory, & Hancock, 2014).

It is also worth noting that the effect relates to the *logarithm* of Barrier-to-Exit. As can be seen in fig. 4a, the results translate into an approximately 40% increase in Barrier-to-Exit - controlling for the activity and random effects. While extrapolation is a tricky matter (Timmers & Wagenaar, 1977), this does point toward a cautiously worrying trend.

To fully understand the results, we must consider the strengths and limitations of the study.

4.2 Strengths and Limitations

Regarding the robustness of our results, there are both strengths and weaknesses in our analysis. The main strength is the scale of the analysis: by relying on a huge dataset of Amazon book reviews, we were approaching the N=all definition by (Cukier & Mayer-Schoenberger, 2013). This scale allowed us to test for the relatively small effect sizes - even while relying on noise-introducing approximations like defining category-relevance using co-occurrence (see 2.1). The scale also allowed us to test the feasibility of using Barrier-to-Exit in practice, which I will return to later.

However, the design also has some significant limitations particularly with a) the validity of the proxy (i.e. the *construct validity* and b) the potential sample bias.

Let us start with the proxy. The primary issue here is that we have an *indirect* and *incomplete* view into the recommendation process. The ratings only constitute a small part of the interaction with the recommender system. The primary feedback loop is plausibly in the buying and browsing behaviour (Smith & Linden, 2017). This breaks with the original framing from (Rakova & Chowdhury, 2019), which assumes a more direct interaction between ratings and recommender systems. While this assumption holds for MovieLens, it is more problematic for Amazon. The MovieLens recommender system is made around ratings; the "contract" between the user and MovieLens is that the user provides ratings and MovieLens gives personalised recommendations based on those ratings. The ratings on Amazon, however, have a more public purpose: they help other consumers choose products (Leino & Räihä, 2007) In some sense, this makes ratings a strong signal for preferences (?, ?). The main implication of the validity is thus one of coverage: because of the increased "cost" associated with ratings, the average Amazon user in the filtered dataset made 43 ratings while the average MovieLens user has made 740 ratings (Harper & Konstan, 2016)[3]. This is both a cost in terms of time (you have to publicly create a review) and money (most users probably rate products they bought).

This leads us to potential sample bias. Because Barrier-to-Exit requires relatively many reviews to have a well-defined value, our analysis has predominantly very active users. This introduces a bias: we can only draw inferences for this particular subset of users and not the general population of Amazon customers. These active users plausibly represent a substantial fraction of Amazon's revenue. However, though there is plausibly a correlation between the number of ratings a user has made and commercial interest for Amazon, this need not be the case.

Additionally, there are important limitations with the statistical analysis that must be addressed.

First, there some assumptions were violated (see appendix A). Specifically, this has to do with the normality of the residuals and random effects. These violations arise from ill-behaving categories (see 6 in appendix A). Theoretically, this should not affect the fixed effects estimates (Schielzeth et al., 2020). Accordingly, re-fitting the model with the problematic categories removed showed similar results (see appendix B.2. However, it does affect the interpretation of the VPCs of the two categories. Since these are mainly used to control for variability in the levels ((Baayen et al., 2008) and not for testing hypotheses (?, ?), this is a minor problem.

There are two potential statistical causes of these issues. The first is a lack of data from the early years of Amazon. As fig. 4b shows, there are very few observations of Barrier-to-Exit in the early years compared to later. Amazon has grown dramatically in the past 25 years (Wells et al., 2018), which has been fuelled by many more people having access to the internet (?, ?). Statistically, this makes it more difficult to assess the long-term increase as early observations will tend to have high leverage (Fox, 2015).

[3]Bear in mind the long-tailed nature of both distributions.

This leads us to the second issue: problems with transformations. Transforming the data introduces problems for the validity and interpretability of the results (FENG et al., 2014). This makes some researchers argue that it is better to refrain - even when the assumptions are violated (Schielzeth et al., 2020).

In our case, fitting the models without transforming the Barrier-to-Exit made it impossible to fit the models without singular fits (Fox, 2015). We, therefore, had no choice but to account for the non-normality and transforming the variables was an easy and standard choice (see CITATIONS).

However, there are alternatives. For one, we could have utilised *generalised linear mixed-effects models* (glme-models; (Fox, 2003)). Glme-models allow us to specify a link function which makes it possible to account for the non-normality in a more elegant way (Fox, 2015). As Barrier-to-Exit is continuous and right-skewed it might be well-fitted by a gamma-distribution (?, ?). However, glme-models introduce their own suite of issues (Fox, 2015).

The most flexible way would be to model Barrier-to-Exit in a Bayesian framework (?, ?). This would allow us to have more flexibility in the distributions and have measures of uncertainty (?, ?). However, this comes at a noticeable cost in terms of speed - particularly for a large dataset such as ours (Bürkner, 2017).

4.3 Implications and Further Research

Because of the issues discussed above and the small effect size, this paper fails to provide a "smoking gun" for preference manipulation in Amazon's book recommender system. Nevertheless, in attempting to model the evolution of Barrier-to-Exit for Amazon, I have uncovered some important perspectives that can inform further work in investigating preference manipulation.

First, it is essential to have metrics that are defined for the entire population. Metrics can exclude certain people either in their definition or execution. As previously discussed, Barrier-to-Exit is only defined for people who have made several ratings about the same category within the specified time window. This excludes both customers who choose not to (publicly) rate products and customers who only use Amazon sporadically. Previous research suggests that "lurkers" (people who do not actively post) make up a substantial part of the internet population (Nonnecke & Preece, 2000). Investigating preference manipulation for these users is also important - both ethically (Jannach & Adomavicius, 2016) and legally (Franklin et al., 2022). However, with Barrier-to-Exit we lack the data to accomplish this

One way to get broader coverage is to shift the metrics from the *user-level* to the *system-level* - i.e. whether the recommender system manipulates preferences in general. In this paper, we use the aggregate Barrier-to-Exit of many actual users to investigate the trend over time. Focusing on the system allows us to expand our toolbox. This could include using *"sock puppet" auditing* (Sandvig, Hamilton, Karahalios, & Langbort, 2014). Here researchers create bot profiles to interact in controlled ways with the recommender system. This methodology has already been used to investigate whether different recommender systems facilitate radicalization (Ledwich & Zaitsev, 2019). It is important to note that bot-based auditing comes with its own set of practical and ethical limitations that are important to consider (see Sandvig et al., 2014)

Second, there is a dilemma of *portability* (i.e. how well the metric can be used across contexts; see Selbst et al., 2019). On the one hand, socio-technical metrics (like Barrier-to-Exit) need to be tailored to their context. Blindly, "porting" metrics from one domain to other risks making them fit poorly. On the other hand, portability between different systems is necessary for comparison.

Barrier-to-Exit was designed for a content recommendation system based on user ratings. Intuitively, that should make it well-suited for use on Amazon book recommender; The setup of the data is very similar (Harper & Konstan, 2016; Ni et al., 2019). Nevertheless, the difference in context makes it difficult to port Barrier-to-Exit to Amazon: this introduces the issues with sampling bias, which we discussed earlier.

One way to resolve this is to rely on audits made by the companies. After all, Amazon plausibility has a much better estimate of user preferences than we can get from mere ratings. This, however, quickly becomes ethically problematic as it relies on Amazon accurately reporting potential manipulation in their systems. Some researchers believe this type of *self-governance* is positive (Roski, Maier, Vigilante, Kane, & Matheny, 2021), while others are more sceptical (Zuboff, 2019). Either way, mechanisms for verifying these audits are essential for establishing trust and adhering to regulation (Floridi et al., 2022).

Further work should focus on creating measures of preference manipulation in content-based recommender systems. These should focus on having a high-construct validity and a high coverage (i.e. measure "actual" effort of preference change for close to all users).

5 Conclusion

Understanding how recommender systems shape our behaviour is essential to avoid manipulation. In this paper, I investigated the Amazon recommender system concerning whether it has made it harder to change preferences.

By analysing the Barrier-to-Exit (Rakova & Chowdhury, 2019) of more than 50,000 users, I found a small but highly significant increase in Barrier-to-Exit over time, which indicates that it has indeed become harder to change preferences for the analysed users.

However, sampling bias induced by the calculation of Barrier-to-Exit makes it difficult to draw conclusions about the general population of Amazon customers. This highlights the dilemma of portability in measuring socio-technical systems (Selbst et al., 2019): accurately evaluating a concept like "changing preferences" requires adapting to the context of the system, which makes it more difficult to generalise (and compare) to other systems.

Comparing recommender systems is necessary for ensuring that these respect human autonomy (Varshney, 2020) and live up to new regulations such as the EU AI Act (Kop, 2021). Further work, should aim to create auditing procedures and metrics that allow third parties to measure potential preference manipulation in a way that fits within the context of the industry *and* and allows for comparisons between different systems. This will help assess the pressures of *surveillance capitalism* (Zuboff, 2019) on human autonomy.

References

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language*, *59*(4), 390–412. (Publisher: Elsevier)

Bartoń, K. (2022, September). *MuMIn: Multi-Model Inference.* Retrieved 2023-01-08, from https://CRAN.R-project.org/package=MuMIn

Bennett, J., & Lanning, S. (2007). The netflix prize. In *Proceedings of KDD cup and workshop* (Vol. 2007, p. 35). New York.

Bürkner, P.-C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of statistical software*, *80*(1), 1–28.

Carroll, M. D., Dragan, A., Russell, S., & Hadfield-Menell, D. (2022). Estimating and penalizing induced preference shifts in recommender systems. In *International Conference on Machine Learning* (pp. 2686–2708). PMLR.

Cukier, K., & Mayer-Schoenberger, V. (2013). The rise of big data: How it's changing the way we think about the world. *The Best Writing on Mathematics 2014*, 20–32.

FENG, C., WANG, H., LU, N., CHEN, T., HE, H., LU, Y., & TU, X. M. (2014, April). Log-transformation and its implications for data analysis. *Shanghai Archives of Psychiatry*, *26*(2), 105–109. Retrieved 2023-01-10, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/ doi: 10.3969/j.issn.1002-0829.2014.02.009

Floridi, L., Holweg, M., Taddeo, M., Amaya Silva, J., Mökander, J., & Wen, Y. (2022, March). *capAI - A Procedure for Conducting Conformity Assessment of AI Systems in Line with the EU Artificial Intelligence Act* (SSRN Scholarly Paper No. ID 4064091). Rochester, NY: Social Science Research Network. Retrieved 2022-04-01, from https://papers.ssrn.com/abstract=4064091 doi: 10.2139/ssrn.4064091

Fox, J. (2003). Effect Displays in *R* for Generalised Linear Models. *Journal of Statistical Software*, *8*(15). Retrieved 2023-01-10, from http://www.jstatsoft.org/v08/i15/ doi: 10.18637/jss.v008.i15

Fox, J. (2015). *Applied regression analysis and generalized linear models.* Sage Publications.

Franklin, M., Ashton, H., Gorman, R., & Armstrong, S. (2022, May). Missing Mechanisms of Manipulation in the EU AI Act. *The International FLAIRS Conference Proceedings*, *35*. Retrieved 2022-12-31, from https://journals.flvc.org/FLAIRS/article/view/130723 doi: 10.32473/flairs.v35i.130723

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021, December). *Datasheets for Datasets.* arXiv. Retrieved 2022-10-20, from http://arxiv.org/abs/1803.09010 (arXiv:1803.09010 [cs])

Harper, F. M., & Konstan, J. A. (2016, January). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 1–19. Retrieved 2022-12-29, from https://dl.acm.org/doi/10.1145/2827872 doi: 10.1145/2827872

Jannach, D., & Adomavicius, G. (2016, September). Recommendations with a Purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 7–10). Boston Massachusetts USA: ACM. Retrieved 2022-09-01, from https://dl.acm.org/doi/10.1145/2959100.2959186 doi: 10.1145/2959100.2959186

Jiang, R., Chiappa, S., Lattimore, T., György, A., & Kohli, P. (2019, January). Degenerate Feedback Loops in Recommender Systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 383–390). Honolulu HI USA: ACM. Retrieved 2022-09-07, from https://dl.acm.org/doi/10.1145/3306618.3314288 doi: 10.1145/3306618.3314288

Kipf, T. N., & Welling, M. (2017, February). *Semi-Supervised Classification with Graph Convolutional Networks.* arXiv. Retrieved 2022-12-14, from http://arxiv.org/abs/1609.02907 (arXiv:1609.02907 [cs, stat])

Knijnenburg, B. P., Reijmer, N. J., & Willemsen, M. C. (2011, October). Each to his own: how different users call for different interaction methods in recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 141–148). New York, NY, USA: Association for Computing Machinery. Retrieved 2023-01-11, from https://doi.org/10.1145/2043932.2043960 doi: 10.1145/2043932.2043960

Kop, M. (2021, November). EU Artificial Intelligence Act: The European Approach to AI. , 11.

Kramer, A. D., Guillory, J. E., & Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, *111*(24), 8788–8790. (Publisher: National Acad Sciences)

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest package: tests in linear mixed effects models. *Journal of statistical software*, *82*(13). (Publisher: The Foundation for Open Access Statistics)

Ledwich, M., & Zaitsev, A. (2019). Algorithmic extremism: Examining YouTube's rabbit hole of radicalization. *arXiv preprint arXiv:1912.11211*.

Leino, J., & Räihä, K.-J. (2007, October). Case amazon: ratings and reviews as part of recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems* (pp. 137–140). New York, NY, USA: Association for Computing Machinery. Retrieved 2023-01-08, from https://doi.org/10.1145/1297231.1297255 doi: 10.1145/1297231.1297255

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, *7*(1), 76–80. (Publisher: Ieee)

Millecamp, M., Htun, N. N., Jin, Y., & Verbert, K. (2018). Controlling spotify recommendations: effects of personal characteristics on music recommender user interfaces. In *Proceedings of the 26th Conference on user modeling, adaptation and personalization* (pp. 101–109).

Nakagawa, S., Johnson, P. C., & Schielzeth, H. (2017). The coefficient of determination R 2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. *Journal of the Royal Society Interface*, *14*(134), 20170213. (Publisher: The Royal Society)

Nguyen, T. T., Hui, P.-M., Harper, F. M., Terveen, L., & Konstan, J. A. (2014). Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web - WWW '14* (pp. 677–686). Seoul, Korea: ACM Press. Retrieved 2022-12-31, from http://dl.acm.org/citation.cfm?doid=2566486.2568012 doi: 10.1145/2566486.2568012

Ni, J., Li, J., & McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 188–197).

Nonnecke, B., & Preece, J. (2000). Lurker demographics: Counting the silent. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 73–80).

Papakyriakopoulos, O., Serrano, J. C. M., & Hegelich, S. (2020). Political communication on social media: A tale of hyperactive users and bias in recommender systems. *Online Social Networks and Media*, *15*, 100058. (Publisher: Elsevier)

Popper, K. R. (1970). *Normal science and its dangers*. Cambridge University Press Cambridge.

Rakova, B., & Chowdhury, R. (2019, September). *Human self-determination within algorithmic sociotechnical systems*. arXiv. Retrieved 2022-09-07, from http://arxiv.org/abs/1909.06713 (arXiv:1909.06713 [cs])

Raschka, S., Patterson, J., & Nolet, C. (2020, March). Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv:2002.04803 [cs, stat]*. Retrieved 2021-12-27, from http://arxiv.org/abs/2002.04803 (arXiv: 2002.04803)

Roski, J., Maier, E. J., Vigilante, K., Kane, E. A., & Matheny, M. E. (2021). Enhancing trust in AI through industry self-governance. *Journal of the American Medical Informatics Association*, *28*(7), 1582–1590. (Publisher: Oxford University Press)

Sandvig, C., Hamilton, K., Karahalios, K., & Langbort, C. (2014). Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry*, *22*, 4349–4357.

Schielzeth, H., Dingemanse, N. J., Nakagawa, S., Westneat, D. F., Allegue, H., Teplitsky, C., ... Araya-Ajoy, Y. G. (2020, September). Robustness of linear mixed-effects models to violations of distributional assumptions. *Methods in Ecology and Evolution*, *11*(9), 1141–1152. Retrieved 2022-12-27, from https://onlinelibrary.wiley.com/doi/10.1111/2041-210X.13434 doi: 10.1111/2041-210X.13434

Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., & Vertesi, J. (2019, January). Fairness and Abstraction in Sociotechnical Systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 59–68). Atlanta GA USA: ACM. Retrieved 2022-09-07, from https://dl.acm.org/doi/10.1145/3287560.3287598 doi: 10.1145/3287560.3287598

Smith, B., & Linden, G. (2017, May). Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing*, *21*(3), 12–18. Retrieved 2022-12-20, from http://ieeexplore.ieee.org/document/7927889/ doi: 10.1109/MIC.2017.72

Timmers, H., & Wagenaar, W. A. (1977, November). Inverse statistics and misperception of exponential growth. *Perception & Psychophysics*, *21*(6), 558–562. Retrieved 2023-01-10, from https://doi.org/10.3758/BF03198737 doi: 10.3758/BF03198737

Varshney, L. R. (2020, September). *Respect for Human Autonomy in Recommender Systems.* arXiv. Retrieved 2022-12-31, from http://arxiv.org/abs/2009.02603 (arXiv:2009.02603 [cs])

Wells, J. R., Danskin, G., & Ellsworth, G. (2018). Amazon. com, 2018. *Harvard Business School Case Study*(716-402).

Zhu, F., & Liu, Q. (2018). Competing with complementors: An empirical look at Amazon.com. *Strategic Management Journal*, *39*(10), 2618–2642. Retrieved 2023-01-11, from https://onlinelibrary.wiley.com/doi/abs/10.1002/smj.2932 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smj.2932) doi: 10.1002/smj.2932

Zuboff, S. (2019). *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power: Barack Obama's Books of 2019.* Profile Books.

A Validation of Assumptions

There are several assumptions to validate for crossed-linear mixed effects models (Baayen et al., 2008). Specifically, we need to assess:

- **Linearity**: Is the phenomenon the model captures actually linear? This can be tested by looking for trends in the residuals. If no trends are found, linearity holds (?, ?)
- **Homogeneity of Variance**: Is the variance across residuals equal? This can be tested by plotting by residuals against response and looking for cone-like shapes (?, ?).
- **Normality of residuals**: Are the residuals normally distributed? We can assess this by a QQ-plot over the residuals.
- **Normality of random effects**: Per the definition of crossed-effects mixed effects models (Baayen et al., 2008), we expect the residuals to be normal. This can be assessed similarly to the random effects.

Note that testing the assumptions rely on *visual* tests rather than *statistical* tests. This is not for a lack of statistical tests (CITATION GALORE). Rather it is because statistical tests tend to become oversensitive with large datasets (?, ?). Given our dataset has more than 50,000 observations, it is safer to use visual tests.

First let us test for linearity and heteroscedasticity. The residual plot can be seen in fig. 5:

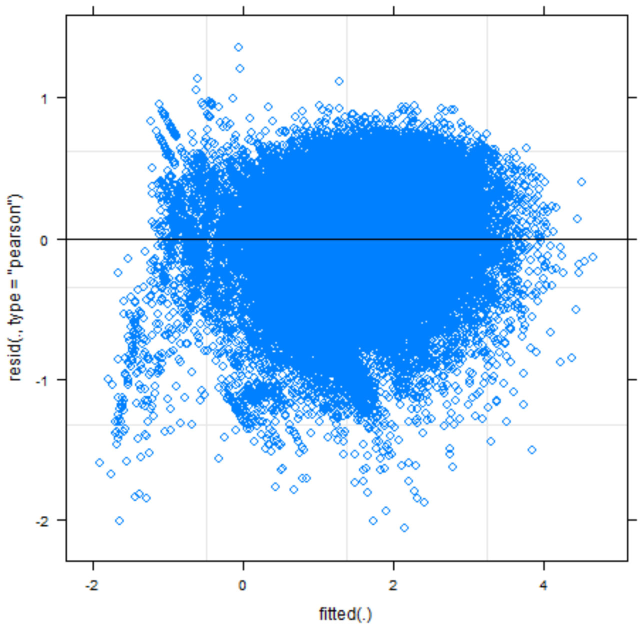


Figure 5: Residuals for main model

At first glance, there seem to be no strong heteroscedasticity or apparent linearity. However, at a closer look, there are some weird outliers in the bottom left corner. Furthermore, there is a weird kind of straight line going from top left corner and down towards the middle. This indicates that there may be something wrong with the model.

Let us now assess the normality of the residuals:

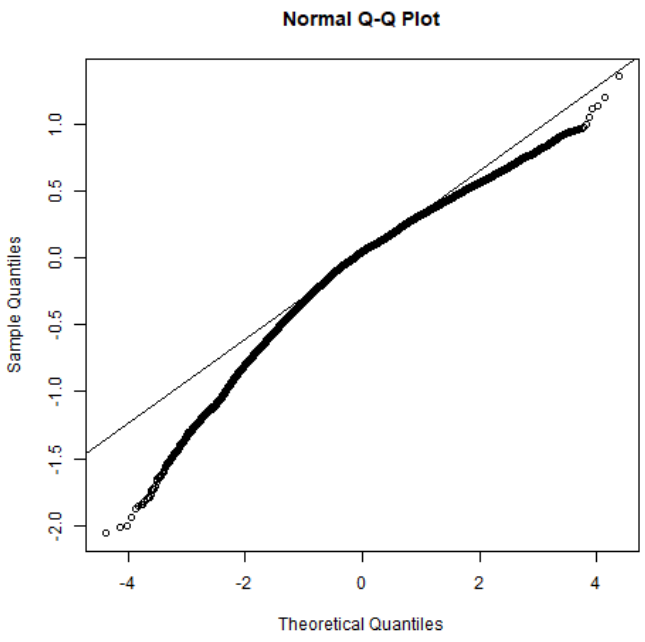


Figure 6: Caption

We notice that the residuals are generally lower than the normality-line. This indicates a left skew in the residuals, i.e. there are smaller outliers. While the fixed effects robust against this type of skew, the the estimates for group-level variances are more effected (Schielzeth et al., 2020). Since these effects are likely to be exasperated by bootstrapping (Schielzeth et al., 2020), this particularly harms our interpretation of **RQ2**.

Finally, let us consider the normality of the random effects. A qq-plot of the random effects for the two levels can be seen below:

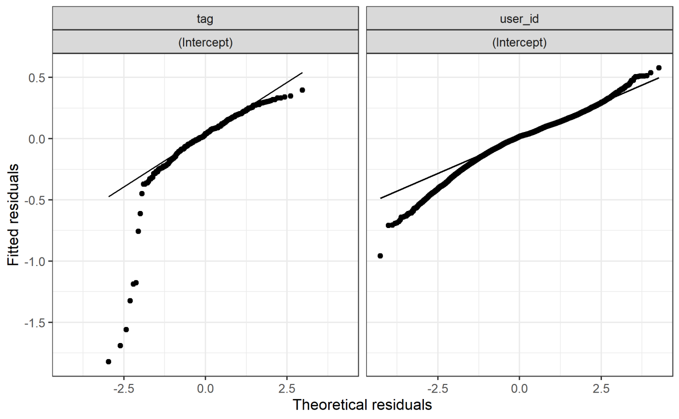


Figure 7: QQ-plot of the random effects per group. "tag" represents categories and "*user_id*" represents users

Here we see a dramatic skew toward the lower level of the of the fitted residuals. This implies that there are weird dynamics in the lower values of Barrier-to-Exit.

The effect of the non-normality is plausibly inflated estimates of category-level variance (Schielzeth et al., 2020). This further invalidates the interpretation of **RQ2**.

While the effects of non-normality on the fixed effects should be minor (Schielzeth et al., 2020), I nevertheless conduct a robustness check. In the robustness check, I remove the problematic categories and refit the main model (eq. 5). The results of this can be seen in Appendix B.2

B Previously Tested Models

B.1 Model without activity-level

The initial model, I had planned to test was a model without activity-level (β_2), but otherwise following eq. 5. That is, all parameters have the exact same definition. For completeness, the old equation can be seen in eq. 6:

$$y_{ij} = \beta_0 + \beta_1 X_{1i} + e_i + e_j \tag{6}$$

However, when I ran the model (also using lmerTest (Kuznetsova et al., 2017)) and plotted the residuals, I got the following:

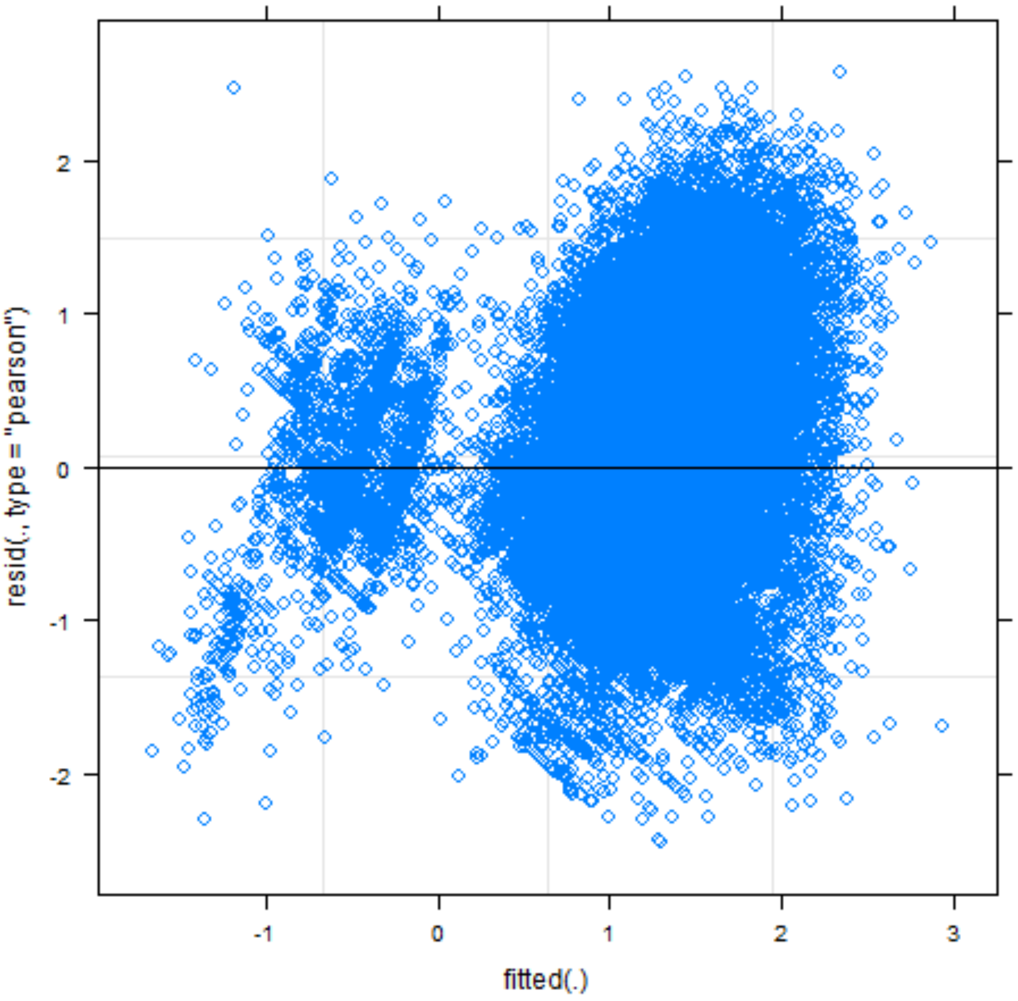


Figure 8: Residuals of initial model. From a visual inspection the residuals are *not* randomly distributed

Just from a brief visual inspection, it is clear to see that the residuals are *not* randomly distributed: There are two distinct "bands" that both seem to trend upward. This breaks the assumptions that the residuals are randomly distributed (?, ?). While many assumption violations are reduced with enough data (Baayen et al., 2008; Schielzeth et al., 2020), non-linearity is not one of them (?, ?).

Fortunately, the non-random residuals were (partially) fixed by introducing activity-level for the reasons described in section 2.3.

B.2 Problematic Categories Removed

Here I fit the main model (eq. 5, with problematic categories removed. I define a problematic category as a category with a fitted random effect of less than -0.5. I obtain this threshold by visually inspecting fig. 7.

The results of this fit can be seen below:

Table 2

	Dependent variable:
	Barrier-to-Exit (Log)
Time (years)	0.018***
	(0.001)
Activity-level (Log)	0.619***
	(0.001)
Intercept	0.321***
	(0.016)
Observations	77,058
Unique users	46668
Unique categories	322
Marginal R^2_{LMM}	0.70
Conditional R^2_{LMM}	0.81
Note:	*p<0.1; **p<0.05; ***p<0.01

The most important finding from this is that there is no difference in our variable of interest (Time in years). It is still 0.018 with a high degree of significance ($p \ll 0.0001$). Also worth noting that the marginal R^2 has increased somewhat ($0.64 \to 0.70$) and the conditional R^2 is roughly the same (0.82 vs 0.81).

C Code

This sections has important code snippets for calculating category-similarity (C.1) as well as implementing Barrier-to-Exit (C.2).

C.1 Tag similarity

```python
"""
Creates a book - category similarity matrix in long format for joining
"""
import json
import numpy as np
import pandas as pd
import re
import logging
from pathlib import Path
from collections import Counter
from numba import jit, njit
from sklearn.feature_extraction.text import CountVectorizer

# import sparse array type from scipy
from scipy.sparse import csr_matrix

logger = logging.basicConfig(
    level=logging.INFO, format="%(asctime)s - %(name)s - %(levelname)s - %(message)s"
)


def read_json(path: Path) -> dict:
    with open(path, "r") as f:
        return json.load(f)
```

```python
def create_df(book_id: str, tags: list) -> pd.DataFrame:
    return pd.DataFrame(columns=tags, index=[book_id]).fillna(1)


def remove_punctuation(text: str) -> str:
    return re.sub(r"[^\w]", "", text)


def tags_to_str(tags: list) -> str:
    return " ".join(remove_punctuation(tag) for category in tags)


@jit
def get_similarity(Xc_dense, tag_array: np.ndarray) -> np.ndarray:
    similarity_scores = Xc_dense[tag_array, :][:, tag_array].mean(axis=1)
    return similarity_scores


@jit
def naive_loop(X: np.ndarray, Xc_dense: np.ndarray, full_df_shape: int) -> np.ndarray:
    final_array = np.zeros(full_df_shape, dtype=np.float32)
    curr_idx = 0
    for row in X:
        similarity_scores = get_similarity(Xc_dense, row)
        final_array[curr_idx : curr_idx + len(similarity_scores)] = similarity_scores
        curr_idx += len(similarity_scores)
    return final_array


def get_popular_tags(books: dict, top: int = 100) -> set:
    tag_count = Counter()
    for _, tags in books.items():
        tag_count.update(tags)
    popular_set = {tag for tag, _ in tag_count.most_common(top + 1)}
    popular_set.remove("Books")
    return popular_set


def filter_books(books: dict, top: int = 10) -> dict:
    popular_set = get_popular_tags(books, top=top)
    return {
        book_id: {tag for tag in tags if tag in popular_set}
        for book_id, tags in books.items()
    }


def get_string_tags(books: dict) -> dict:
    string_tags = {
        book_id: tags_to_str(tags) for book_id, tags in books.items() if tags
    }
    return string_tags


def calc_cooccurance_matrix(X: csr_matrix) -> np.ndarray:
    Xc = X.T * X  # this is co-occurrence matrix in sparse csr format
    # divide each row by its sum
    Xc = Xc.multiply(1 / Xc.sum(axis=1))
    Xc = Xc.astype(np.float32)
    return Xc.toarray()


## The next couple of functions were mainly used for testing and debugging


@jit
def is_sorted(a):
    for i in range(a.size - 1):
        if a[i + 1] < a[i]:
            return False
    return True


@njit
def map_index(idx: np.ndarray) -> np.ndarray:
    new_idx = np.zeros(idx.shape[0], dtype=np.int32)
    i = 0
    for j, num in enumerate(idx[1:]):
        j += 1
        if num != idx[j - 1]:
            i += 1
        new_idx[j] = i
    return new_idx


@njit
def first_difference(a1, a2) -> int:
    max_range = max(a1.shape[0], a2.shape[0])
    for i in range(max_range):
        if a1[i] != a2[i]:
            return i
    return -1


def main():
    # count number of each tag
    logging.info("Reading in books")
    books = read_json(Path("data/book_category.json"))

    logging.info("Filtering books")
    filtered_books = filter_books(books, top=100)

    logging.info("Creating string tags")
    full_string_tags = get_string_tags(filtered_books)

    logging.info("create sparse matrix")
    # create CountVectorizer object
    vectorizer = CountVectorizer(dtype=np.uint16)

    # fit the vectorizer to the tags
    X = vectorizer.fit_transform(full_string_tags.values())
    X[X > 0] = 1

    logging.info("creating idx")
    books = list(full_string_tags.keys())
    num_tags = [len(tag.split()) for tag in full_string_tags.values()]
    book_idx = np.repeat(books, num_tags)
```

```python
    X_dense = X.toarray()
    X_dense = X_dense.astype(np.bool_)
    X_idx, tag_locs = np.where(X_dense)
    assert (
        book_idx.shape[0] == X_idx.shape[0]
    ), "Number of rows in full_df and X_dense do not match"
    logging.info("Calculating co-occurrence matrix")
    Xc_dense = calc_cooccurance_matrix(X)
    logging.info("Calculating similarity scores")
    all_similarity = naive_loop(X_dense, Xc_dense, X_idx.shape[0])

    final_df = pd.DataFrame(
        {"tag_loc": tag_locs, "similarity": all_similarity}, index=book_idx
    )
    # fix types for saving
    final_df["tag_loc"] = final_df["tag_loc"].astype(np.uint16)

    # save to pickle
    final_df.reset_index().to_pickle("data/book_tag_similarity.pkl")


if __name__ == "__main__":
    main()
```

C.2 Barrier-to-Exit

```python
"""
Calculates Barrier to Exit (BTE) for all users with more than n ratings.
"""
import argparse
import pandas as pd
import numpy as np
import logging
import pickle
import random
import multiprocessing as mp
from tqdm import tqdm
from pathlib import Path
from typing import List, Dict, Tuple, Iterable

logging.basicConfig(
    level=logging.DEBUG, format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
)


def filter_dates(df, date):
    YEAR_OFFSET = pd.Timedelta("365 days")
    date_filter = (date - YEAR_OFFSET <= df["date"]) & (df["date"] <= date)
    return df[date_filter]


def get_thresholds(df: pd.DataFrame) -> Tuple[float, float]:
    sum_stats = df.groupby("tag_loc")["score"].agg(["mean", "std"]).dropna()
    upper_thresh = (sum_stats["mean"] + 2 * sum_stats["std"]).mean()
    lower_thresh = (sum_stats["mean"] - 2 * sum_stats["std"]).mean()
    return upper_thresh, lower_thresh


def find_between_tags(df, tag, upper_thresh, lower_thresh):
    tag_filter = df["tag_loc"] == tag
    # filter relevant dates
    tag_df = df[tag_filter]
    above_tag = tag_df["score"] > upper_thresh
    below_tag = tag_df["score"] < lower_thresh
    if not above_tag.any() or not below_tag.any():
        return pd.DataFrame()
    first_above = tag_df[above_tag].iloc[0]
    last_below = tag_df[below_tag].iloc[-1]
    between_crit = (first_above["date"] < tag_df["date"]) & (
        tag_df["date"] < last_below["date"]
    )
    new_df = tag_df[between_crit].groupby(["tag_loc"])["score"].mean().reset_index()
    return new_df


def calculate_thresholds(testdf) -> pd.DataFrame:
    threshdfs = []
    for date in testdf["date"].unique():
        # filter relevant dates
        bte_df = filter_dates(testdf, date)
        # extract thresholds
        upper_thresh, lower_thresh = get_thresholds(bte_df)
        threshdf = pd.DataFrame(
            {"upper": upper_thresh, "lower": lower_thresh}, index=[date]
        )
        threshdfs.append(threshdf)
    return pd.concat(threshdfs)


def faster_calc_thresholds(userdf: pd.DataFrame) -> pd.DataFrame:
    userdf.sort_values("date", inplace=True)
    rollingdf = userdf.groupby("tag_loc").rolling("365d", on="date")
    rollingstd = rollingdf["score"].std().dropna().reset_index()
    rollingmean = rollingdf["score"].mean().reset_index()
    joined = rollingstd.merge(
        rollingmean, on=["tag_loc", "date"], suffixes=("_std", "_mean")
    )
    joined["upper"] = joined["score_mean"] + 2 * joined["score_std"]
    joined["lower"] = joined["score_mean"] - 2 * joined["score_std"]
    return joined.groupby("date")[["upper", "lower"]].mean()


def bte(tag_df: pd.DataFrame) -> List[Dict]:
    """Calculates the Barrier-to-Exit for a given tag with the thresholds given"""
    result = []  # type: List[Dict]
    while True:
        if ~np.any(tag_df["below"]):  # check if there are any "below" values
            break
        if ~np.any(tag_df["above"]):  # check if there are any "above" values
            break
        tag = tag_df.iloc[0]["tag_loc"]
        above_mask = tag_df["above"]  # create a boolean mask for the "above" values
        start_idx = np.argmax(above_mask) + 1  # start one after the first above
        if start_idx == len(tag_df):  # check if we've reached the end of the dataframe
            break
        below_mask = tag_df.iloc[start_idx:][
            "below"
        ]  # create a boolean mask for the "below" values
        if ~np.any(below_mask):
            break
        stop_idx = np.argmax(below_mask) + start_idx

        # skip if stop index is right after start index
        if start_idx == stop_idx:
            tag_df = tag_df.iloc[stop_idx:]
            continue

        between_score = tag_df.iloc[start_idx:stop_idx]["score"].sum()
        n_ratings = tag_df.iloc[start_idx:stop_idx]["n"].sum()
        start_date = tag_df.iloc[start_idx - 1]["date"]
        stop_date = tag_df.iloc[stop_idx]["date"]

        result.append(
            {
                "tag": tag,
                "start_date": start_date,
                "stop_date": stop_date,
                "score": between_score,
                "n": n_ratings,
            }
        )
        tag_df = tag_df.iloc[stop_idx:]

    return result


def flatten_list(l):
    return [item for sublist in l for item in sublist]


def add_thresholds(user_df: pd.DataFrame) -> pd.DataFrame:
    threshdf = faster_calc_thresholds(user_df)
    joined_df = user_df.merge(threshdf, left_on="date", right_index=True).dropna()
    joined_df["above"] = joined_df["score"] > joined_df["upper"]
    joined_df["below"] = joined_df["score"] < joined_df["lower"]
    return joined_df


def bte_calc_fast(user_df: pd.DataFrame):
    user_id = user_df.iloc[0]["user_id"]
    joined_df = add_thresholds(user_df)
    relevant_tags_filter = joined_df["above"] | joined_df["below"]
    relevant_tags = joined_df.loc[relevant_tags_filter, "tag_loc"].unique()
    dfs = [group for tag, group in joined_df.groupby("tag_loc") if tag in relevant_tags]
    results = flatten_list([bte(tag_df) for tag_df in dfs])
    return pd.DataFrame(results).assign(user_id=user_id)


def process_user(filepath: Path) -> pd.DataFrame:
    user_df = pd.read_csv(filepath, index_col=0, parse_dates=["date"])
    return bte_calc_fast(user_df)


def multiprocess_bte(filelist: Iterable[Path], n_jobs=1, n_tqdm=None,) -> pd.DataFrame:
    with mp.Pool(n_jobs) as pool:
        results = list(tqdm(pool.map(process_user, filelist), total=n_tqdm,))
    return pd.concat(results)
```

```python
def write_pickle(obj, path):
    with open(path, "wb") as f:
        pickle.dump(obj, f)


def main(args: argparse.Namespace):

    # read in data
    logging.info("finding files...")
    DATA_DIR = Path(args.data_dir) / "users"
    user_counts = pd.read_csv(DATA_DIR.parent / "user_counts.csv")
    top_users = user_counts[user_counts["count"] > args.num_ratings][
        "user_id"
    ]  # type: pd.Series
    files = [DATA_DIR / f"user_{user_id}_preprocessed.csv" for user_id in top_users]
    logging.info("checking if files exist...")
    for file in files:
        if not file.exists():
            raise FileNotFoundError(f"{file} does not exist!")
    logging.info(f"found {len(files)} files")

    # sample files
    if args.sample:
        logging.info(f"Sampling {args.sample} files...")
        random.seed(42)
        files = random.sample(files, args.sample)

    write_pickle(files, Path(args.output_dir) / "files.pkl")  # save for comparisons

    n_cores = mp.cpu_count() - 1
    n_jobs = args.n_jobs
    logging.info(f"using {n_jobs} cores (out of {n_cores} available)")
    bte_df = multiprocess_bte(filelist=files, n_jobs=n_jobs, n_tqdm=len(files),)
    logging.info("Done calculating BTE!")

    logging.info("Saving BTE...")
    output_path = (
        Path(args.output_dir)
        / f"bte_thresh{args.num_ratings}_sample{args.sample if args.sample else 'full'}.csv"
    )
    bte_df.to_csv(output_path, index=False)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Calculate Barrier-to-Exit for a given dataset",
        formatter_class=argparse.ArgumentDefaultsHelpFormatter,
    )
    parser.add_argument("--data-dir", type=str, help="Path to data", required=True)
    parser.add_argument(
        "--output-dir", type=str, help="Dir to save output", default=".",
    )
    parser.add_argument(
        "--n-jobs", type=int, help="Number of cores to use", default=1,
    )
    parser.add_argument(
        "--sample", type=int, help="Sample size", default=None,
    )
```

```
parser.add_argument(
    "--num-ratings",
    type=int,
    help="Number of ratings to use for threshold",
    default=20,
)
args = parser.parse_args()
main(args)
```