Product Gap Analysis

Laurence Stephan

2024-12-06

Contents

1 Introduction

This project aims to identify product market gaps using Amazon ratings data, specifically from the Toys and Games category. Amazon's website currently receives over 200 million unique visitors each month, equating to approximately 2.4 billion visits per year. In 2022, Amazon's total e-commerce sales reached around $142.43 billion, with over 9.7 million sellers operating on the platform . E-commerce, more generally, is estimated to have a market capitalisation exceeding $5 trillion globally, with trillions of products sold annually.

While these markets are enormous, identifying products that consumers theoretically want but don't yet have access to is a challenge that remains largely unknown to the average seller. A significant amount of time and resources is spent on activities such as market testing, proof of concept, and the development of products that ultimately fail to sell. Research suggests that about 90% of startups fail, with many of them in the e-commerce space. Thus, finding ways to anticipate consumer needs and desires to create products with higher success rates—and lower risks for sellers—could serve as an invaluable tool. Such a resource would save time, effort, money, and energy, and help reduce waste in the development process.

Furthermore, offering a tool like this to entrepreneurs, inventors, and small businesses will help reduce ambiguity when making decisions about which areas to focus on. This will enable them to concentrate on building cool products that meet the needs of the public.

To develop this tool, I employed a combination of methods. First, I performed initial pre-processing to reformat the raw JSON data into more comprehensible structures. This allowed me to explore the data to identify which information could be useful for running predictive models and to uncover trends in ratings, both overall and over time.

Then I wrangled the data to manipulate it into a format to be used across a set of machine learning algorithms, to make predictions on which products individual users are most likely to want to purchase based on the their ratings of previous products they've already bought. These ratings can be grouped together alongside other users who have bought a set of the same products and rated them similarly. Once we have our groups, we can recommend products to a user in a group, by looking at the products that have been well rated other users in the group, that the original user hasn't yet purchased. The idea is if a group of people with similar purchase history to one-another have all bought an item and rated it well, but one person in the group hasn't bought that item, they may like to buy that item, and it will probably be well received, in line with how the rest of the group felt about the item.

In order to make these predictions I tested a small set of well known models to make predictions my test data using root mean squared error (RMSE) to understand their accuracy. I then trained and optimised the models, and chose the one with the lowest RMSE.

Once I was able to make predictions for individual users, I extended the analysis to a larger group within the dataset to determine which products were being recommended most frequently. To interpret the results, I selected the top 50 recommendations for each user, ranked each product based on its recommendation position, and divided these rankings by the total number of users analysed. This process provided a weighted score that reflected both how often a product was recommended and its average ranking. The results were then sorted based on the highest average rankings. Additionally, I calculated the frequency with which each product appeared across the entire dataset to assess its saturation in terms of sales. Products with a high recommendation ranking but low previous sales volume were identified as having significant potential for opportunity.

Another issue in society is the use of language to convey ideas, which often fails to communicate a reality beyond the subjective perspective of the observer. Over time, layers of abstract ideas and concepts have developed without rigour or a solid foundation in observation. As a result, traditional labels can become more of a hindrance than a help. One solution to this problem is to observe patterns in human behaviour and use these patterns to create new labels that are more closely related to real-world groupings, rather than relying on arbitrary labels established by historical figures in positions of influence.

To address this challenge and explore further product opportunities, I decided to examine the categories of opportunity using the latent factors uncovered in my model, in order to identify patterns of product groupings. I then reviewed the product names within those groups, and sent them to Llama2 to generate new category names.

These more specific categories are better aligned with the purchasing behaviours of real users, compared to broader labels like "outdoor toys" or "puzzle toys." The products most highly associated with these new categories potentially offer the greatest opportunity, as they are likely to be the most congruent and popular. As a result, new products designed for these categories, similar to their top-performing products, are more likely to sell well.

2 Exploring the data

2.1 Data Overview

	Description	Count
1	Blank titles	2541
2	Blank asin	0
3	Blank reviewerID	0
4	Blank ratings	0
5	NaN ratings	0
6	Na ratings	0

There are 2541 rows with missing titles, which is a small percentage of the data. All the other categories I need appear to be populated correctly, and my ratings data contains no Na or NAN values. To make sure my analysis remains consistent and meaningful, I will remove the rows with blank titles.

```
 "training data"

Rows: 1,696,415
Columns: 10
$ overall        <int> 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 2, 4, 1, 5, 5, 3, 5, 4, 3~
$ reviewTime     <chr> "10 6, 2013", "08 9, 2013", "04 5, 2016", "02 13, 2016"~
$ reviewerID     <chr> "A2LSCFZM2FBZK7", "A3IXP5VS847GE5", "A1274GG1EB2JLJ", "~
$ asin           <chr> "0486427706", "0486427706", "0486427706", "0486427706",~
$ reviewerName   <chr> "Ginger", "Dragonflies &amp;amp; Autumn Leaves", "barbara a~
$ reviewText     <chr> "The stained glass pages are pretty cool. And it is nic~
$ summary        <chr> "Nice book", "Great pictures", "The pictures are great,~
$ unixReviewTime <int> 1381017600, 1376006400, 1459814400, 1455321600, 1449705~
$ title          <chr> "DOVER PUBLICATIONS Stained Glass Color Book Wizards An~
$ price          <chr> "$5.68", "$5.68", "$5.68", "$5.68", "$5.68", "$5.68", "~

 "testing data"

Rows: 188,475
Columns: 10
$ overall        <int> 5, 4, 5, 2, 3, 5, 5, 4, 5, 5, 5, 5, 5, 3, 5, 4, 4, 5, 5~
$ reviewTime     <chr> "05 24, 2015", "01 18, 2015", "10 22, 2014", "10 18, 20~
$ reviewerID     <chr> "ALLSNTNR6N6UL", "AI3JLLMYH16NZ", "A2G7033C3Z61KL", "A3~
$ asin           <chr> "0486427706", "0486448789", "0486448789", "0486448789",~
$ reviewerName   <chr> "Amazon Customer", "Laura Snelson", "Pamela D. Jones", ~
$ reviewText     <chr> "Exactly as described, nice pictures, great transaction~
$ summary        <chr> "nice pictures, great transaction", "It was cute and he~
$ unixReviewTime <int> 1432425600, 1421539200, 1413936000, 1382054400, 1491609~
$ title          <chr> "DOVER PUBLICATIONS Stained Glass Color Book Wizards An~
$ price          <chr> "$5.68", "$3.04", "$3.04", "$3.04", "$17.45", "$17.45",~
```

The training set contains 1,696,411 rows, which make up approximately 90% of the data, while the testing set has 188,479 rows, representing about 10%. Each set contains 10 variables, represented by the columns.

The reviewerID variable represents a unique identifier assigned to each user profile. The asin variable represents a unique identifier assigned to each product. The overall variable represents the rating given by a user to a product on a scale from 1 to 5, where 1 represents the lowest score and 5 represents the highest.

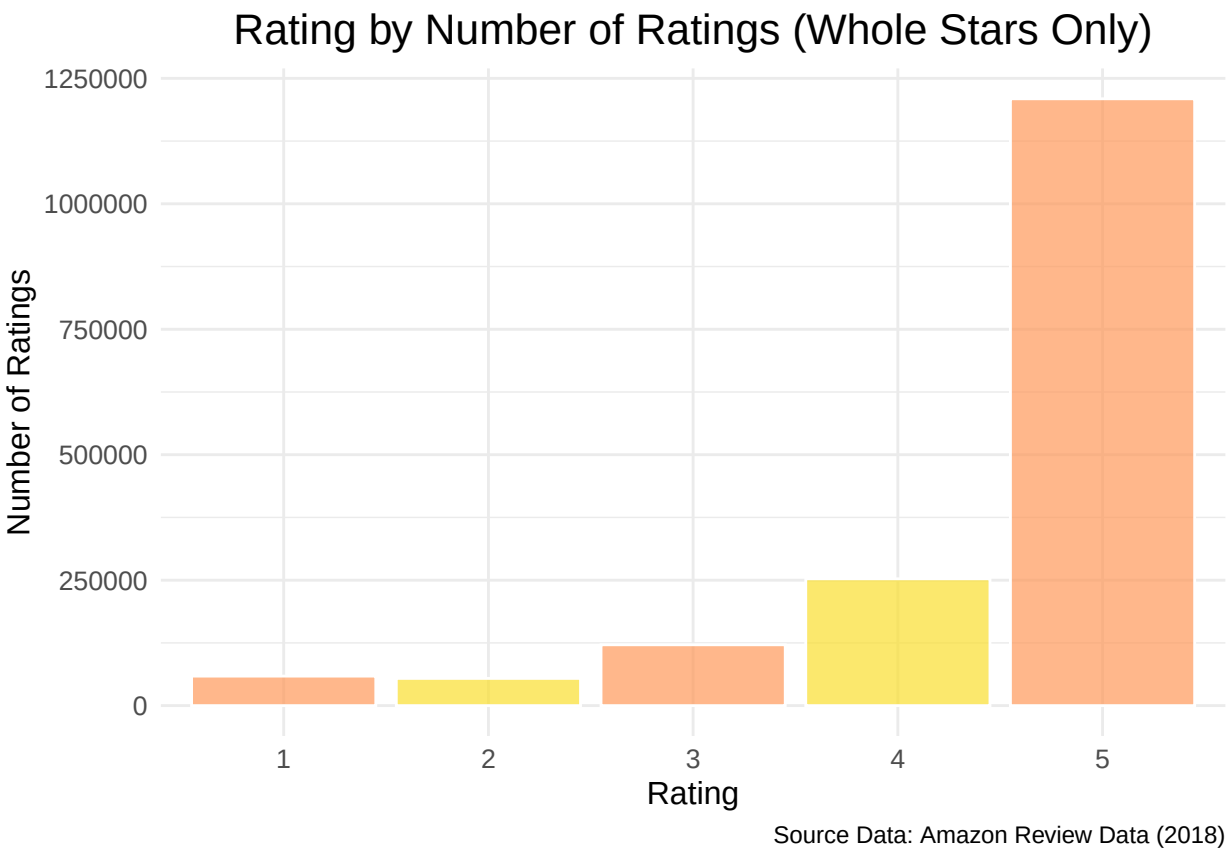
The unixReviewTime and reviewTime variables act as timestamp variables that record the date and time when a product review was submitted. The title variable represents the name of the product. The reviewerName variable contains the first name or a funky nickname used by the reviewer. The price variable shows the cost of the product in dollars. Finally, the reviewText variable contains a long string of words that provide qualitative context to the review rating.

Although the same unique variables may appear across multiple rows, each combination of reviewerID and asin is unique, meaning that each row represents a distinct rating for a specific product by a specific user.

All columns are in format, except for the overall rating, which is in format. For most analyses, the type data is fine. However, when running matrix factorization models, this data will need to be reformatted to work with the models.

2.2 Exploring The Data

An initial investigation of each variable was performed to gain a strong understanding of the data.



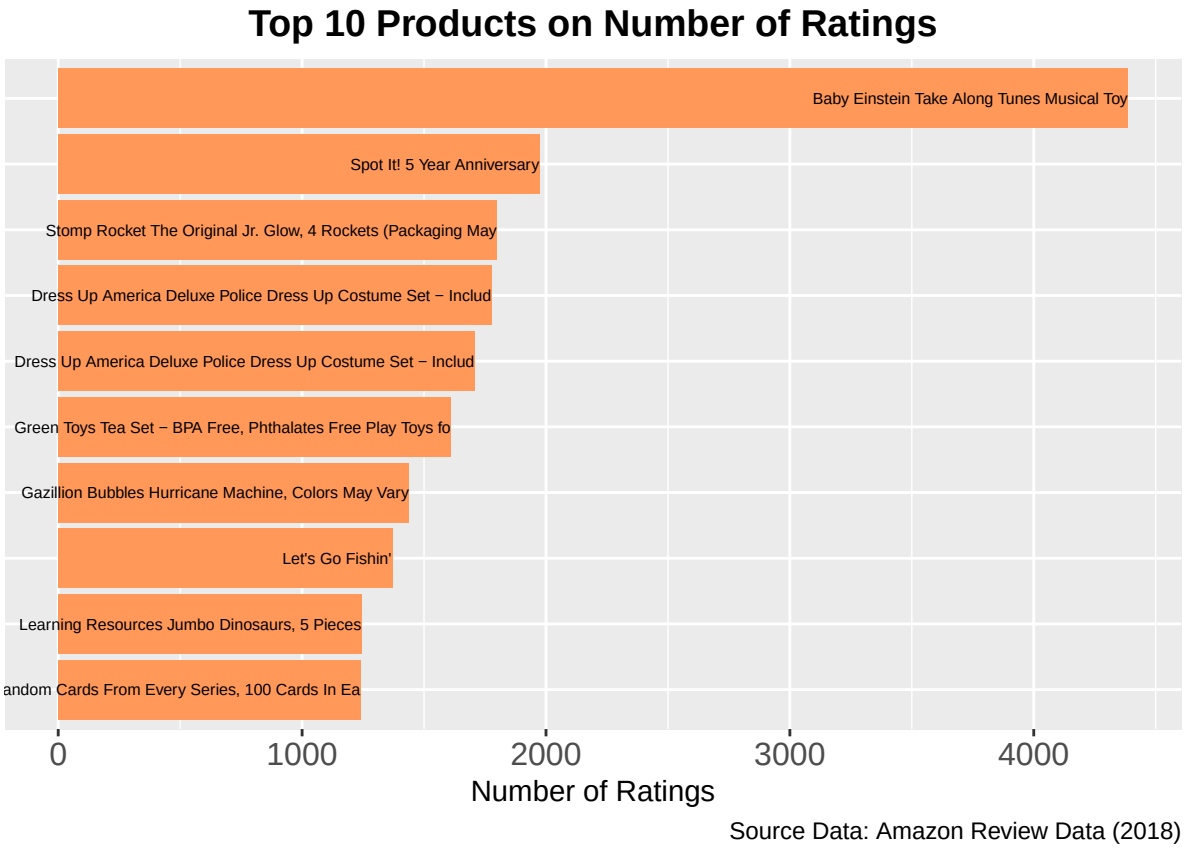
A visual representation of the rating data shows that it only contains full star ratings. It also shows that people tend to leave higher ratings over lower ones, with the mode being 4 stars. Such findings suggest that users are more likely to leave positive reviews over negative ones.

The data was grouped by product title and arranged in descending order based on the number of ratings each product received. This table provides insight into the popularity of products, showing both the volume of ratings and which products are the most widely reviewed.

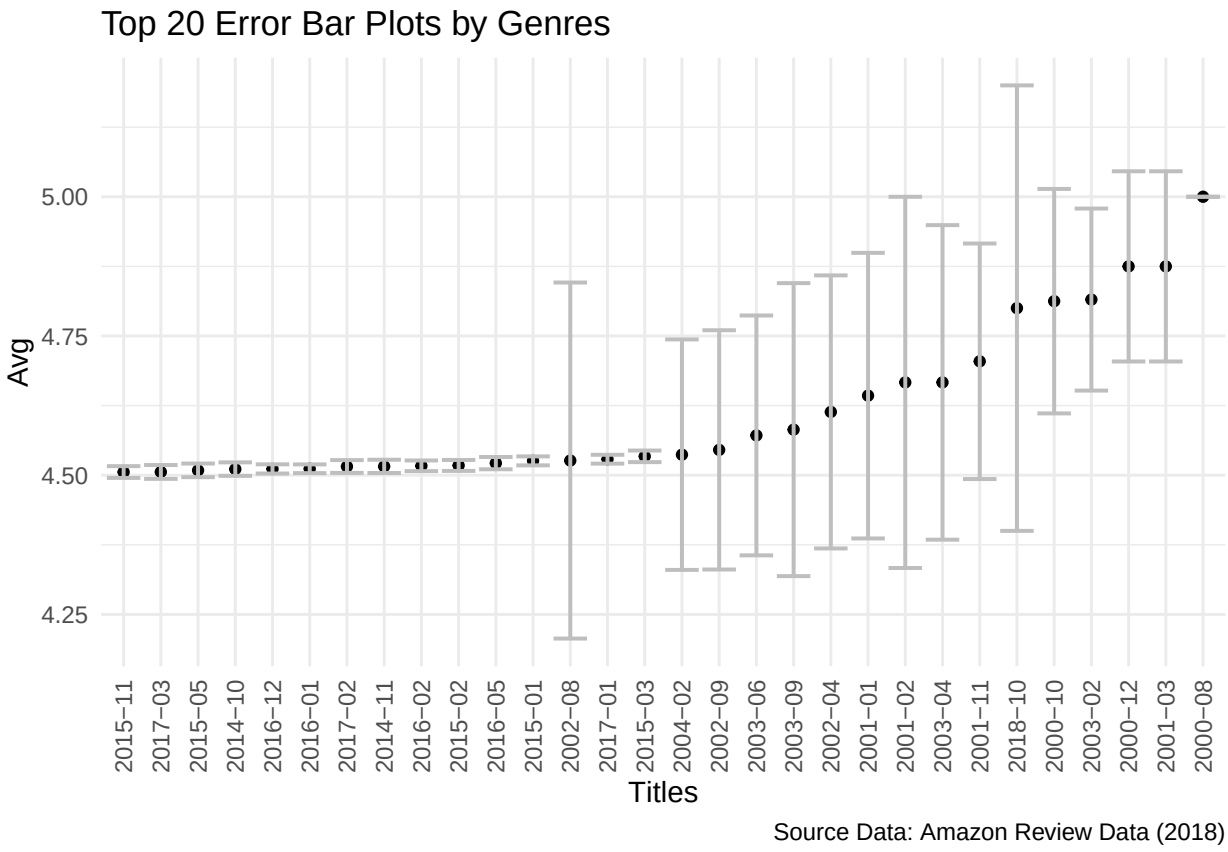
Understanding which products have received the most ratings helps guide decisions for the future. However, it's important to note that the rating count is not adjusted for factors like website traffic or release date. Products that have been available longer naturally have more time to accumulate ratings. Additionally, the release timing of a product can influence its popularity and its potential to gain momentum. For example, products released recently may be trending and have the potential to go viral, while older products may have missed the opportunity to reach a broad audience during their initial release due to fewer users on platforms like Amazon or the internet in general.

Moreover, external factors, such as the increased internet usage and spending during lockdown periods, could have led to products released during that time outselling those released at different times. Essentially, some products may have been launched at the "wrong" time but still have future potential. Therefore, understanding the impact of release timing and website traffic bias is crucial. This highlights the value

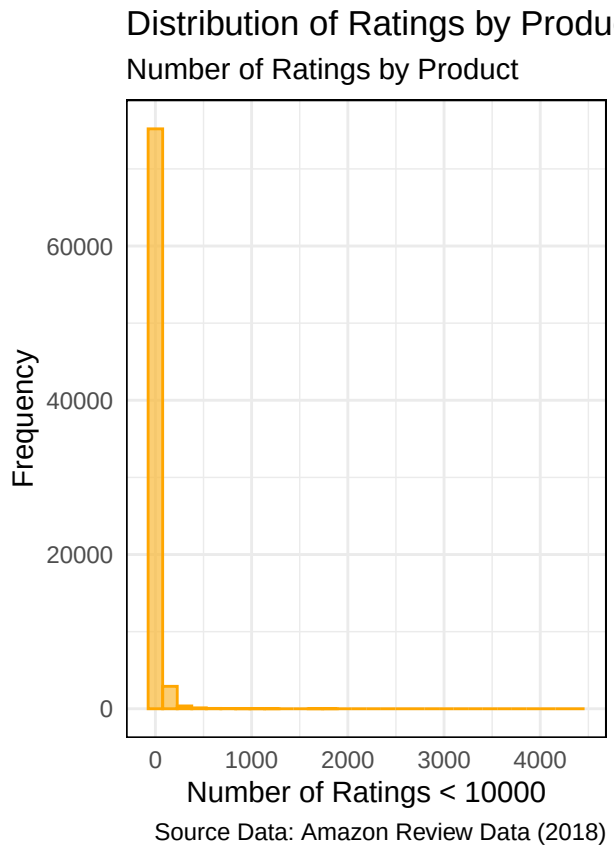
of analyzing data in time-restricted batches and utilizing real-time data to better understand trends and product potential.

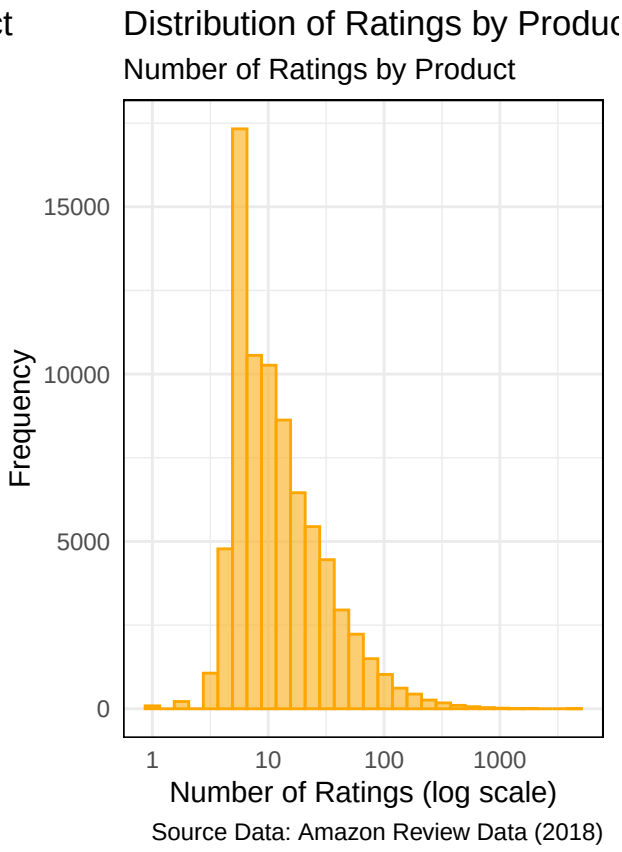


A visual representation of the most sold products really highlights that the 'Baby Einstein Take Along Tunes Musical Toy' is substantially popular than any other product, and in comparison, there is a much smaller gap in purchases for products 2-9.



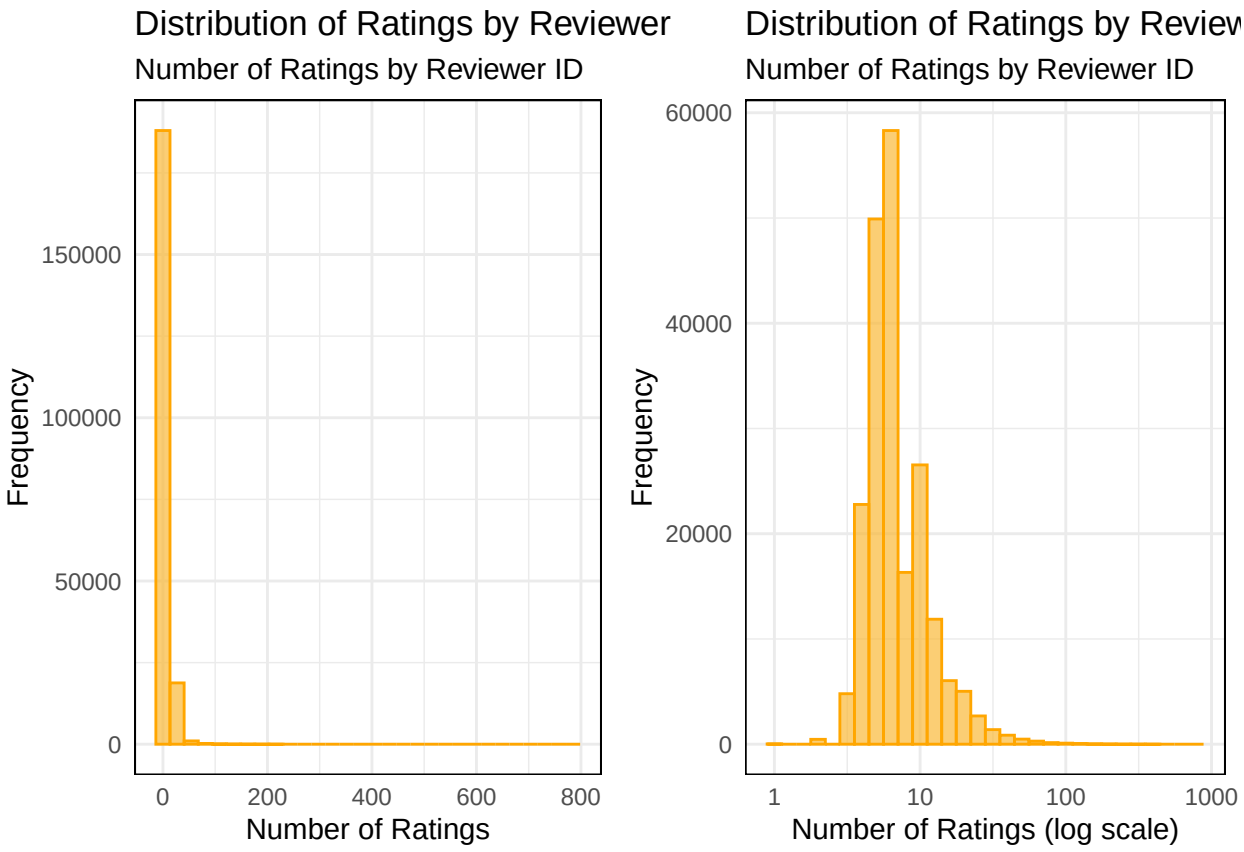
If I group the data by month, I can analyse which months had the highest ratings. The early 2000s seem to have a collection of months with higher average ratings, though with greater variability. In contrast, more recent months show less variability in the ratings.





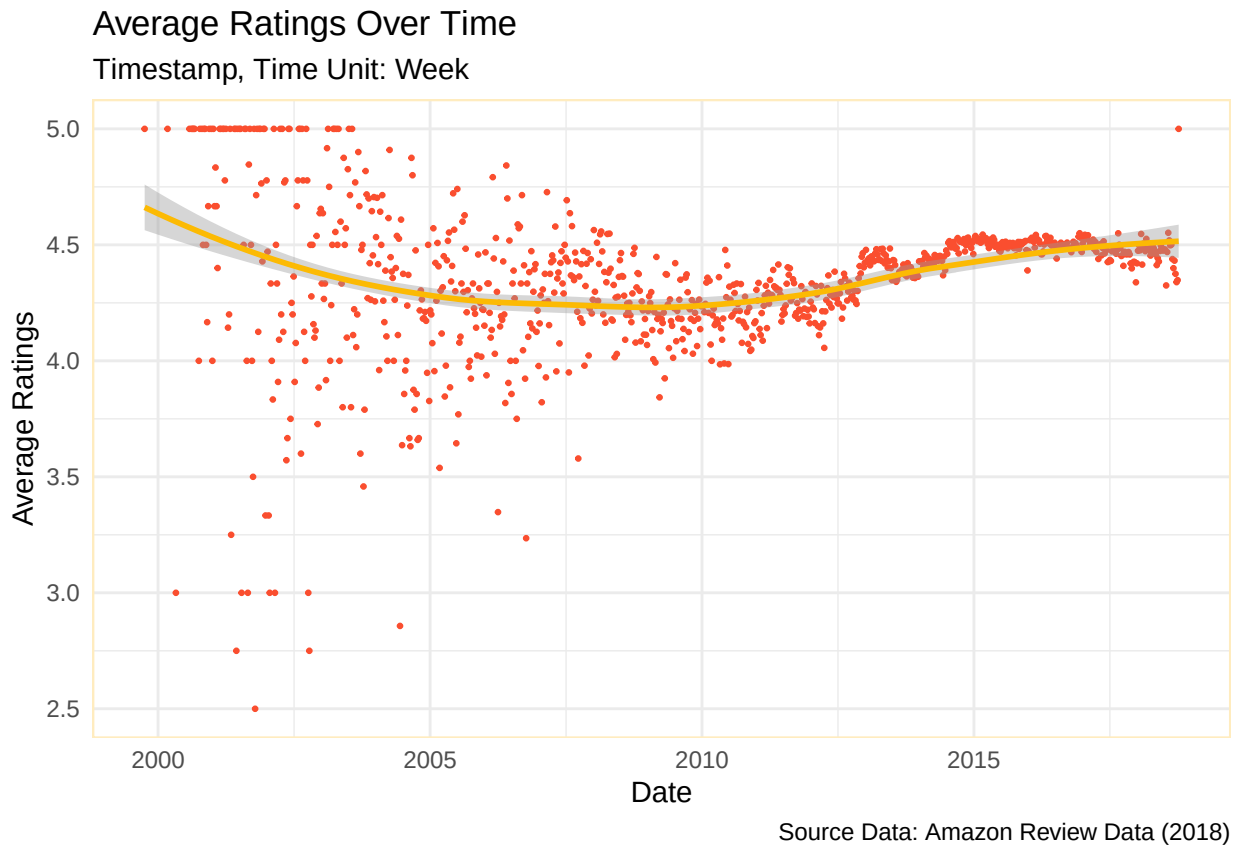
The analysis of ratings by product reveals a highly right-skewed distribution, where most products receive very few ratings, as evidenced by the tall bar near zero, with frequency rapidly declining as the number of ratings increases. This suggests that while the majority of products have low ratings, a small subset of products are rated frequently, dominating the market. In general, certain products disproportionately outsell most others.

A second histogram, plotted on a logarithmic scale, offers a clearer view across orders of magnitude. Although the skewness is less apparent due to the log transformation, it still shows that there are significantly more products with low ratings than those with high ratings. This highlights the rarity of products with many ratings and emphasizes the data sparsity issue inherent in the dataset, making it challenging to predict product preferences accurately, especially in a market with many potentially great but unnoticed products



The analysis of the number of ratings per reviewer reveals a distribution that is heavily skewed to the right, characterised by a rapid decrease in frequency as the number of ratings increases. Most reviewers leave only a few reviews, but there are some superusers who contribute over 700 reviews. This pattern suggests that while the majority of reviewers provide relatively few ratings, a small group contributes a disproportionately high number. This is unsurprising, given the ease of creating new reviewer profiles compared to consistently adding new products to the database.

This observation further highlights the critical importance of addressing scarcity in predictive modelling. It underscores challenges such as accurately predicting reviewer preferences with limited data, or when dealing with reviewers whose interests span a wide spectrum, making it difficult to find reliable nearest neighbours for predictions.



When we look at average ratings over time, subdivided by week, we see unusual variability up until around 2004. There are periods when only 5-star reviews were being left, followed by periods with average ratings as low as 2. However, over time, average reviews have become more homogeneous. The early days of the internet posed many challenges, and the bizarre variability could be attributed to technical issues, bots, or problems with Amazon's data collection process.

In contrast, average ratings now strongly gravitate towards 4.5 stars. This is likely due to optimised algorithms and recommendation systems that provide reviewers with a high level of satisfaction. However, this homogeneity poses a challenge for sellers looking for gaps in the market. The high level of consistency in ratings offers less insight into products that reviewers truly like or dislike. People tend to leave positive reviews (4–5 stars) when there's nothing wrong, but this doesn't necessarily mean they strongly prefer the product over others.

This highlights potential issues with rating scales that lack qualitative context. For example, a 1 might mean "very dissatisfied," a 2 "dissatisfied," a 3 "as expected," a 4 "above expectations," and a 5 "completely exceeded expectations." Therefore, ratings should only be considered one part of a predictive model when analysing product market gaps.

3 Data Preprocessing

I now need to prepare my data to run my models. I will start with some simple Linear Regression models for benchmarking, and then explore a selection of matrix factorization models, which are commonly used in recommendation systems. Each of these models requires different preprocessing techniques to manipulate the data into a format that can be efficiently analysed. For Linear Regression, I'll ensure the data is clean, test for the average, and then add more variables and regularisation. For the matrix factorization models, I'll convert the data into a user-item interaction matrix, and encode the reviewer and product ID's if needed.

3.1 Matrix Transformation

First I analysed the unique reviewer and prodcut ID's to understand the dimensions of my matrix.

```
  n_reviewers n_products
1      208180      78695
```

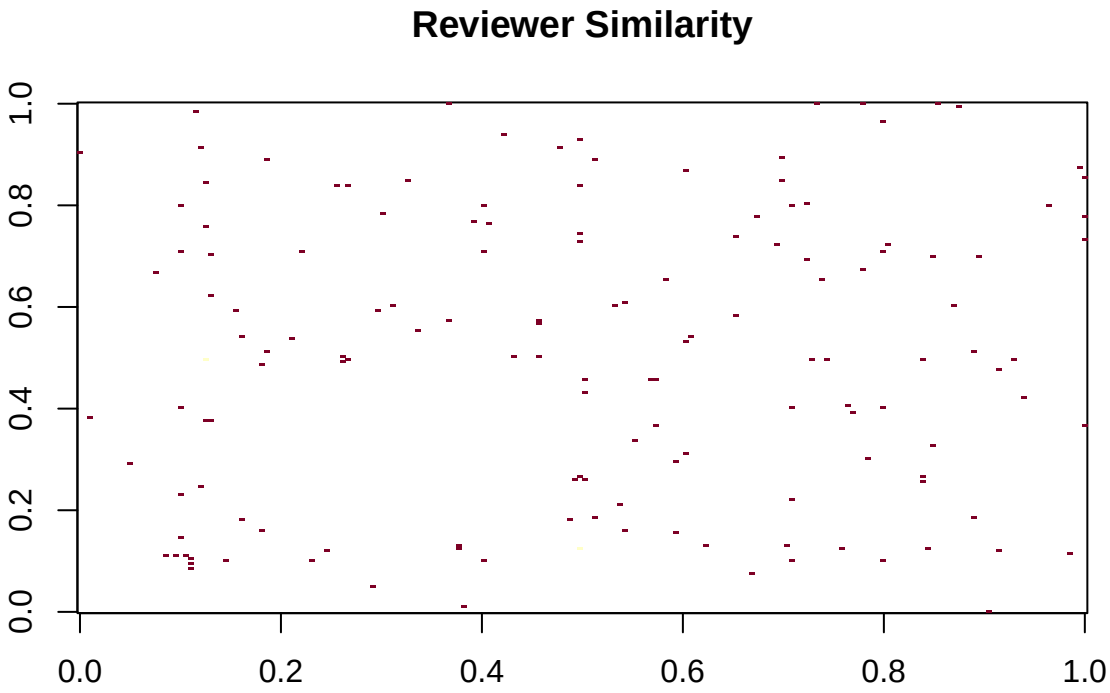
I can see that I have 208180 users, and 78695 items. Notably, as expected the amounts of users is higher higher than the amount of items, as some users have rated many products. However, given the ratio, the matrix will have a large amount of sparsity, which needs to be addressed.

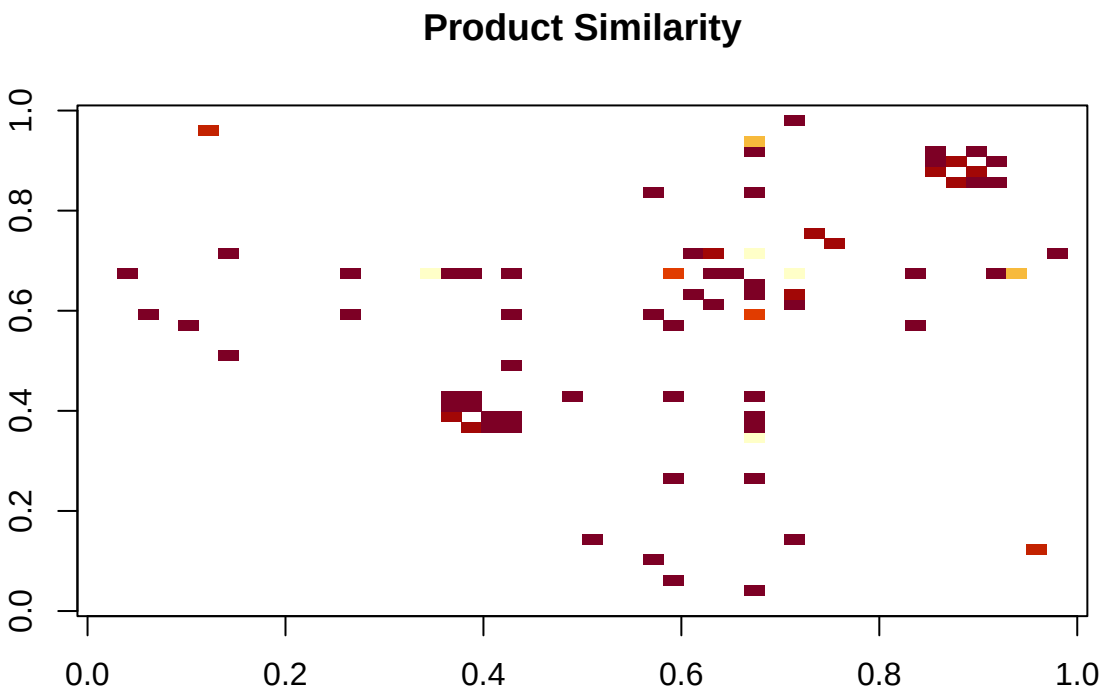
```
10 x 10 sparse Matrix of class "dgCMatrix"

 [1,] . . . . . . . . . .
 [2,] . . . . . . . . . .
 [3,] . . . . . . . . . .
 [4,] . . . . . . . . . .
 [5,] . . . . . . . . . .
 [6,] . . . . . . . . . .
 [7,] . . . . . . . . . .
 [8,] . . . . . . . . . .
 [9,] . . . . . . . . . .
[10,] . . . . . . . . . .

208180 x 78695 rating matrix of class 'realRatingMatrix' with 1590986 ratings.
```

I've used the Matrix.utils package to handle my sparse matrix, which measures the cosine of the angle between two different vectors of an inner product space.


Reviewer Similarity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0



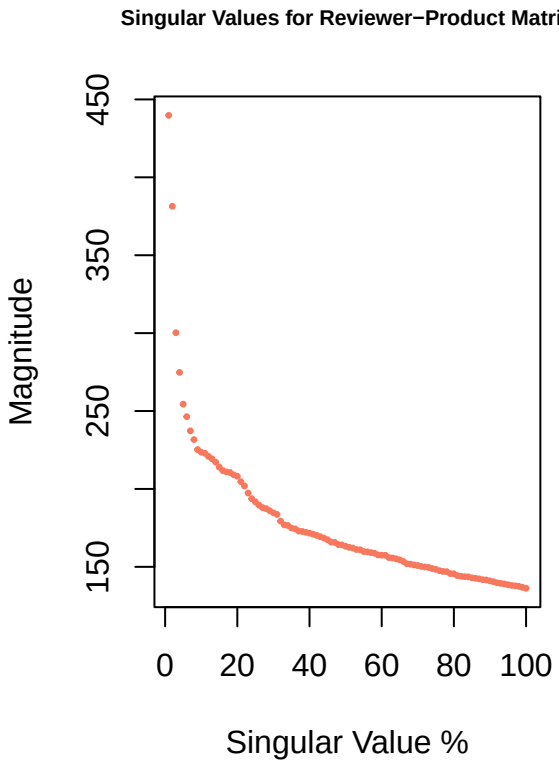

```
"Sparsity of the matrix: 99.99 %"
```

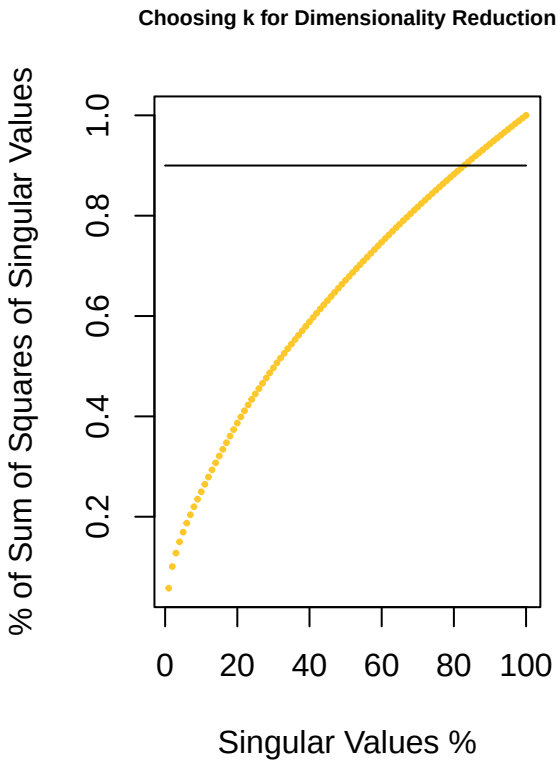
The matrices above were generated using a subset of the first 200 reviewers to reduce computational load. Each row represents a reviewer, and each column represents a product. Consequently, each cell contains a reviewer's rating for a product, with similarity indicated by color. Darker cells denote greater similarity between adjacent reviewers. The matrix is 99.9% sparse, which is typical of large rating-based datasets. This insight offers guidance for model testing, such as considering techniques for handling sparse data, such as matrix factorization methods.

3.2 Dimension Reduction

To address data sparsity, I will reduce the dimensionality using Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). PCA will reduce the data's linear dimensions, while SVD will factorise, rotate, and rescale the data to highlight key patterns.

For efficient local computation, I'll use the IRLBA package, which is optimised for low computational load. IRLBA employs randomised algorithms based on the Lanczos method to quickly and accurately approximate the most significant eigenvalues and vectors of large, sparse matrices.






```
Total Singular Value Sum (all): 3365483

Proportion of Total from First 25: 0.4447748

Proportion of Total from First 50: 0.6711975

Proportion of Total from First 75: 0.8506441

Proportion of Total from First 90: 0.9431124
```

Plotting the cumulative sum of squares for the singular values reveals that the 90% threshold is achieved with just over 80% of the singular values. Interestingly, the first 25% of the singular values only explain 18% of the total variability, with the explanation increasing gradually and uniformly. By the time we reach around 82% of the singular values, the 90% threshold is met. My goal is to identify the smallest number, 'k', of singular values whose squared sum accounts for at least 90% of the total sum of squares. This strategy ensures we capture the vast majority of variability in the data, while maintaining manageable dimensionality for further analysis, avoiding overfitting or excessive complexity in the model. Since 82% of the singular values explain 90% of the variance, we will need to keep the majority of the data, to understand variability. This means we're retaining the core patterns in the data while simplifying the model and removing noise

```
Optimal k Value: 82

Dimensions of U_k: 208180 82

Dimensions of D_k: 82 82

Dimensions of V_k: 82 78695
```

Upon observing that $k = 82$ captures 90% of the variability, I created three matrices: D_k sized 82 x 82, U_k sized 208180 x 82, and V_k sized 82 x 78695. The total number of numeric values required to store these component matrices amounts to $(208180 \times 82) + (82 \times 82) + (82 \times 78695) = 23,530,474$. This reflects a reduction of approximately 99.9% compared to the original 16,382,725,100 entries. Despite the dimensionality reduction, I can still do more work to reduce the memory needed to run my models. To address this, I employ another reduction technique, selecting relevant data using the entire rating matrix.

3.3 Relevant Data

While analysing the data, I noticed a pronounced left skew in both the reviewer rating counts and product ratings, revealing that much of the data offers minimal predictive power. To optimise computational efficiency without compromising the model's accuracy, I introduced a threshold for the minimum number of ratings required for both reviewers and products to be considered in the analysis.

```
Minimum number of products (90th percentile): 13

Minimum number of reviewers (90th percentile): 20

17494 x 18612 rating matrix of class 'realRatingMatrix' with 269116 ratings.
```

By including only products and reviewers within the 90th percentile, any products or reviewers falling within the lowest 10th percentile are excluded from the analysis. Consequently, the dataset comprises products with at least 19 ratings and reviewers who have submitted a minimum of 19 ratings. This results in a matrix featuring 17,620 distinct products and 19,658 distinct reviewers, with a total of 275,522 ratings.

4 Models and Results

The next section will explore several predictive models, including Linear Regression and two Matrix Factorization packages. Each model will be briefly explained, as more detailed descriptions are available elsewhere. The primary focus of this report is to identify gaps in the market. To assess the models' predictive performance, Root Mean Squared Error (RMSE) will be used, with a lower RMSE indicating better accuracy.

The equation to calculate the RMSE can be denoted as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2}$$

4.1 Linear Regression

To start, I will use a set of simple and straightforward linear regression models that incorporate some of the patterns identified during the data exploration process. These models serve as useful benchmarks, providing a basic comparison for more complex approaches. While they offer valuable insights, they typically don't compare directly with matrix factorization techniques, as matrix factorization can capture more complex interactions and latent factors within the data. To establish a baseline, I will begin with a very simple model using the Mean Utility (MU), which represents the average rating. Any model performing worse than this baseline will be considered highly ineffective.

```
RMSE for Mu: 1.000339

Training Time: 0.042 sec

Model Size: 1.4381 MB
```

.....

I employed the average rating, denoted as μ, as a benchmark for making predictions on the test data. In this model, each observation's prediction is simply the global average,

An RMSE of 1 will be used as the baseline for future models. This model serves as a useful benchmark, as any model exhibiting a higher error rate than this baseline will be considered ineffective and will not provide any meaningful improvements over simply predicting the average.

```
RMSE for product Effect: 0.9634633

Training Time: 0.139 sec

Model Size: 7.4429 MB
```

.....

Product Effects: To improve upon my original model by adding product bias to the equation denoted by b_i. This process enabled me to generate predictions with slightly greater precision with an RMSE score of 0.944.

```
RMSE for product + reviewer Effect: 0.9147203

Training Time: 0.182 sec
```

```
Model Size: 23.3278 MB
```

. . . . . .

Product + Reviewer Effects: Next I can apply the same logic for reviewer effect, meaning the model will now take into consideration the difference between ratings from a specific reviewer depicted by b_u, compared to the average of all products. The new model allows me to generate predictions with a considerable improvement in precision achieving an RMSE score of 0.866.

. . . . . .

Regularisation: Next I experimented by adding regularisation to the models to try a smooth out some of the large outlier values, and stop overfitting the data.

Adjusting λ allows us to strike a balance between over and under fitting the training data, while maintaining a simpler model. This balance allows us to maintain accurate pattern capture while stabilising the models behaviour, by reducing sensitivity to noise and sparse data.

This approach assumes that each reviewer's rating deviation from the mean is primarily influenced by the product they are rating. The equations can be shown as:

productID:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} \left(Y_{u,i} - \hat{\mu}\right)$$

reviewerID:

$$\hat{b}_u(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} \left(Y_{u,i} - \hat{\mu} - \hat{b}_i\right)$$

.

In contrast in the second model, I grouped by reviewerId, and then by productId. This approach assumes that each product's rating deviation from the mean is primarily influenced by the reviewer rating it. The equations can be shown as:

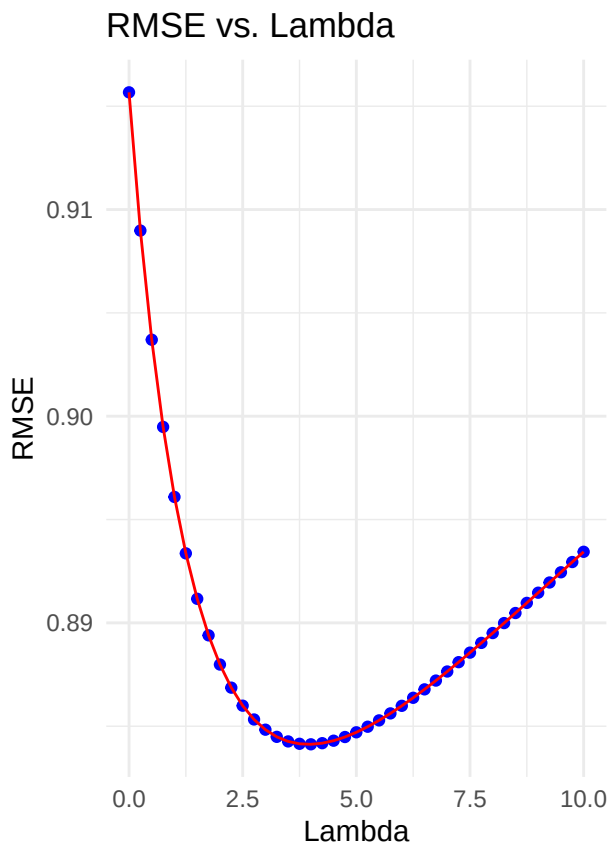
reviewerID:

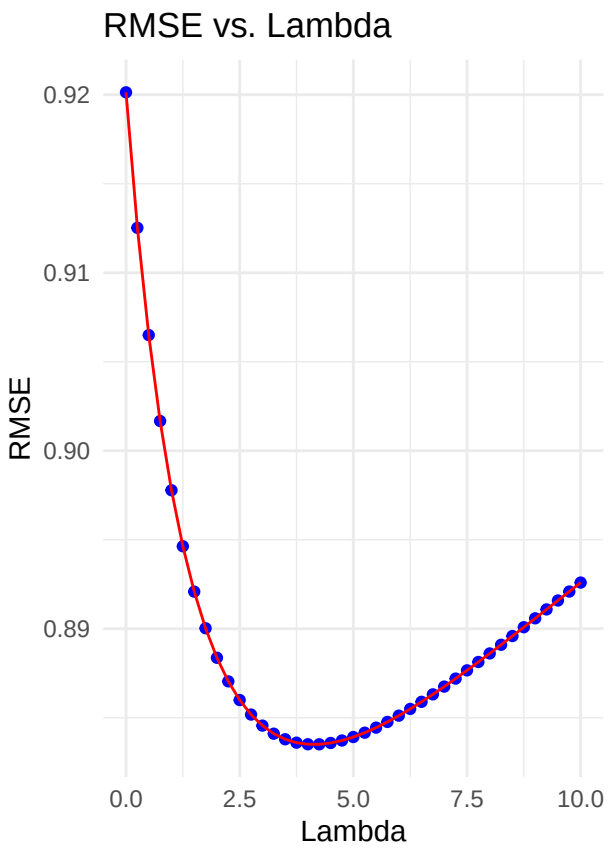
$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} \left(Y_{u,i} - \hat{\mu}\right)$$

productID:

$$\hat{b}_u(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} \left(Y_{u,i} - \hat{\mu} - \hat{b}_i\right)$$

.






```
RMSE for Product ID with Regularisation: 0.8840572

Training Time: 242.43 sec

Model size: 23.3278 MB

RMSE for Reviewer ID with Regularisation:  0.8840033

Training Time: 225.267 sec

Model size: 23.3278 MB
```

I observed that applying regularization to either the Product ID or Reviewer ID variables resulted in an improvement in RMSE, reducing it to 0.88.

This is a summary of a linear regression analysis presented in a table format. The table includes information on different regression methods, their root mean squared error (RMSE), the time taken to perform the analysis, and the size of the data or model used.

The Mu method has the highest RMSE of 1.0003386, but the fastest computation time of 0.042 seconds, and the smallest model size of 1.4381 MB. The Regularised Product + Reviewer Effect and Regularised Reviewer + Product Effect methods have the lowest RMSE values, but also the longest computation times of over 225 seconds and the largest model size of 23.3278 MB. The Product + Reviewer Effects method strikes a balance between performance (RMSE of 0.9147203) and computational requirements (0.182 seconds and 23.3278 MB).

The key reason for this is likely due to the regularisation techniques applied in these models. Regularisation adds a penalty term to the loss function, which encourages the model to learn simpler, more generalisable patterns in the data, rather than fitting too closely to the training data.

Table 1: Results for Linear Regression

Linear Regression			
Method	RMSE	Time (sec)	Size (MB)
Mu	1.0003386	0.042	1.4381
Product Effect	0.9634633	0.139	7.4429
Product + Reviewer Effects	0.9147203	0.182	23.3278
Regularised Product + Reviewer Effect	0.8840572	242.430	23.3278
Regularised Reviewer + Product Effect	0.8840033	225.267	23.3278

4.2 Recommender Engines

Now I will briefly explore the recommenderlab package, which contains a set of algorithms specifically designed for making recommendations using collaborative filtering. The package doesn't accept training and test datasets outlined in this report's confines. Instead, the functions automatically split the data. However, a brief exploration and discussion of some of its features will provide valuable insight into the realm of possibilities.

```
    RMSE
1.266576
```

First, I explored the popular algorithm, which is a non-personalised approach that recommends the most popular items to all users, provided they have not rated them yet. This algorithm serves as a useful benchmark for evaluating the performance of personalised models. It achieved an RMSE score of 1.266576, which is higher than that of the MU algorithm, but it is more relevant for assessing the performance of the other models in this package.

```
    RMSE
0.914213
```

I then examined the user-based collaborative filtering algorithm within the package, which predicts ratings by averaging the ratings of users with similar rating histories to the target user. Due to computational limitations, I was only able to run the model with 10 nearest neighbours and 10% of the data. Despite these constraints, the model still yielded promising results, achieving an RMSE of 0.914213.

```
     RMSE
0.8299064
```

Finally, I investigated the item-based collaborative filtering algorithm, which involves more computationally intensive tuning parameters. These were feasible to use due to the smaller number of items compared to users. This method resulted in the lowest RMSE score of 0.8299064, marking the best performance among the models tested.

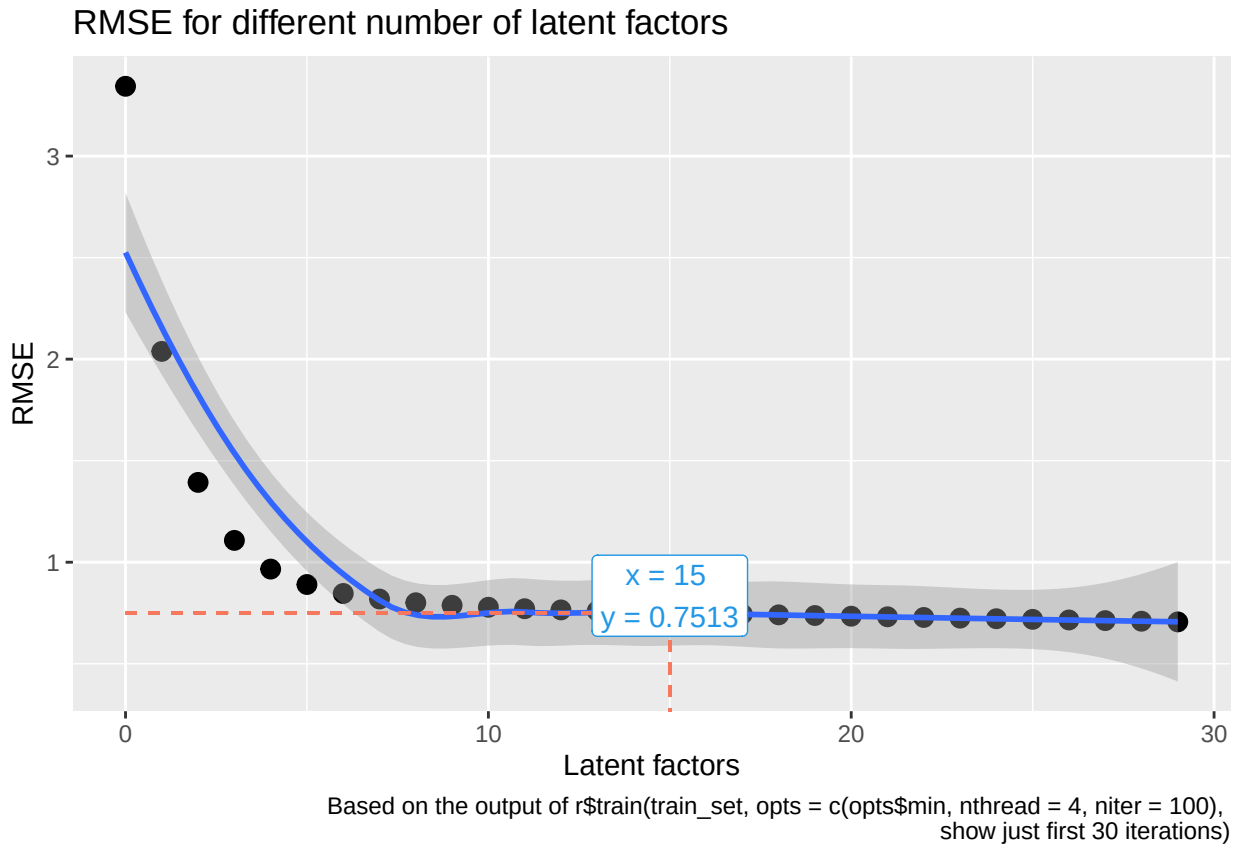
The recommenderlab package offers powerful algorithms for recommendation models that work with sparse matrices. These would be an excellent choice if the primary goal was solely to generate recommendations.

Table 2: Results for Recommenderlab

Recommenderlab	
Method	Size
Popular	1.2665756
UBCF	0.9142130
IBCF	0.8299064

4.3 Matrix Factorisation

In the context of recommender systems, matrix factorization is a commonly employed technique for predicting user-item ratings. The Reco package offers models that effectively capture patterns in the data, enabling strong predictions on previously unseen datasets. Additionally, it provides a comprehensive approach to analyze the inner workings of the model, allowing us to better understand how its parameters function. This insight helps uncover latent factors, which can deepen our understanding of the data and reveal potential market opportunities or gaps.



The figure above illustrates the number of latent factors required to achieve specific RMSE values. As more latent factors are used, the computational cost of the models increases. Notably, there is a sharp decline in RMSE values until they reach approximately 1.2. Beyond this point, the decrease in RMSE slows considerably compared to the additional resources required. For instance, achieving an RMSE of 0.75 requires only 15 latent factors. This suggests an opportunity to optimize the model for greater efficiency, achieving a competitive RMSE with significantly reduced resource usage.

Using matrix factorization, I achieved an RMSE score of 0.9, which underperformed compared to my regularised linear regression models. This result may be attributed to potential overfitting of the data. While fine-tuning the parameters could lead to better performance, for the purposes of this report, the current score is sufficiently lower than that of the MU model. Additionally, matrix factorisation offers the advantage of uncovering latent factors, which can reveal valuable patterns in the data and highlight potential market gaps.

5 Making Predictions

Top Picks for reviewer A3KDZDJMERZ5HW		
No.	Title	Predicted Rating
1	Holiday Sensation Barbie - 1998 - Hallmark Gold Crown Exclus	6.93674
2	Wonderology – Science Kit – Pop Can Robot	6.90220
3	Mickey and Minnie - LOZ Nanoblock Disney Mickey Collection P	6.88936
4	48 Bulk Pack Colorful Building Block Brick Party Favor Suppl	6.53962
5	Badger Basket Three Wheel Doll Jogging Stroller (fits Americ	6.50445
6	Roylco R2442 Color Diffusing Leaves - 6 inches - Pack of 80	6.48936
7	Breaking Games Game of Phones	6.35758
8	Funko POP TV: Last Week Tonight John Oliver Action Figure	6.35515
9	Saitek Mephisto Competition Game Clock Pro	6.25815
10	Funko My Little Pony - Series 3 - Mystery Mini Action Figure	6.24863

The top picks for reviewer A3KDZDJMERZ5HW feature a diverse selection of items. The highest-rated item is the Holiday Sensation Barbie - 1998 - Hallmark Gold Crown Exclusive, with a rating of 6.94, followed closely by the Wonderology – Science Kit – Pop Can Robot at 6.90, and the Mickey and Minnie - LOZ Nanoblock Disney Mickey Collection P at 6.89.

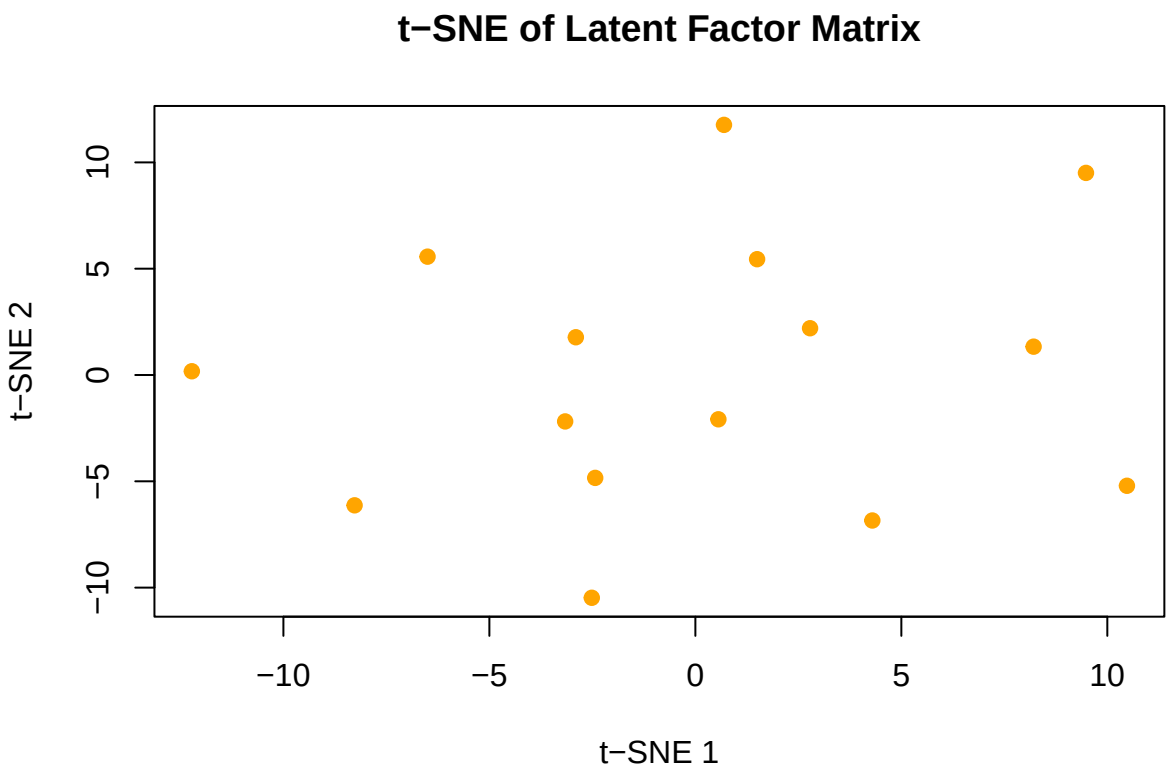
It's important to note that, due to the nature of recommendation engines and how data is scaled, review ratings may not directly correspond to the original 1-5 scale. However, the predictive capabilities of the system remain consistent. These results demonstrate that the recommendation process is functioning correctly and can be used by individuals to gain insights into their own data, specifically regarding the products they are likely to be recommended.

Grouped Recommendations							
Title	Average Position	Percentage Opportunity	Top Position	Lowest Position	Percentage Recommended	Num Recommendations	Occurrences
Schleich Big Spell Set	4.23	94.81	1	50	94.83	19741	3
Uncle Goose Swedish ABC Blocks - Made in USA	6.29	91.29	1	50	91.31	19008	4
Douglas Fox Plumpie	13.35	83.19	1	50	83.22	17325	6
JETTINGBUY 7.2 cm Refill Darts Suction Darts Blue 200 Pcs	16.42	82.71	2	50	82.73	17222	4
2308 - Toothless - 12" DreamWorks Animation OFFICIALLY LICENSED Plush Toy - Happy Feet	15.55	82.10	1	50	82.12	17095	3
Magformers Basic Pentagon Set (12-Pieces) Magnetic Building Blocks, Educational Magnetic ...	18.14	78.87	1	50	78.88	16422	2
Poolmaster 85652 Water Pop Deluxe Lounge (Colors May Vary)	22.97	74.91	1	50	74.92	15596	2
Elf on the Shelf Complete Holiday Gift Bundle: Boy Scout Elf (Brown Eyes), Cuddly Plush Reindeer Elf...	23.76	72.45	2	50	72.47	15086	4
Hi-Float Company Ultra Hi-Float Balloon, 24 oz, Multicolor	24.96	68.91	1	50	68.92	14347	1
USAopoly Yahtzee: Borderlands CL4P-TP Game	25.00	68.10	1	50	68.11	14180	3

To test the system, I made recommendations for individual users and then analysed 10% of the user base to identify potential market gaps. My approach involved reviewing the top 50 products recommended to users and calculating various metrics, including their average ranking positions, highest and lowest positions, the percentage of time they appeared in the top 50, and how often those products had been purchased previously. I then computed an opportunity score by subtracting the number of previous purchases from the number of recommendations for each product, divided by the number of users sampled, and scaled by 100. The rationale behind this is that products with high purchase volumes are likely well-established or in a saturated market, presenting less opportunity. Conversely, products with higher recommendation frequency and fewer past purchases represent the greatest potential. Based on this analysis, I found that Title Schleich Big Spell Set, Uncle Goose Swedish ABC Blocks Douglas and Fox Plumpie products show the most opportunity.

6 Latent Factors

Next, I plan to explore the latent factors generated by the model to uncover underlying patterns in user purchasing behavior. This will help identify potential groups of products that present opportunities, as well as determine which product groups have the highest overall purchase volumes.



The model was constrained to 15 latent factors in order to optimize its predictive accuracy. By visualising these latent factors, we can observe that they are widely dispersed, indicating significant differences between the groups.

The top 50 products for each latent factor are listed in the appendix. These product titles offer valuable insights into consumer purchasing patterns and help identify the themes associated with different groups. To explore these themes further and maintain a systematic approach, I will send each title to Llama3 with the following prompt:

"Here is a collection of products grouped based on their peoples purchases. Please analyse the product names and return with a detailed category name. Keep your answer under 20 words, and don't include any of the product names. Do try and be as specific as possible, as we want category names that really capture the products meaning. Do not write any other comments, we only want the category name."

Category Names From Llama3	
Index	Category Name
1	Outdoor Recreational Equipment and Toys for Children's Learning and Play
2	Toys and Games for Children in Various Categories, Including Vehicles, Puzzles, and Outdoor Play Equipment
3	Toys and Games for Children: Arts and Crafts, Building Sets, Vehicles, and Educational Play Equipment.
4	Toys, Games, and Play Equipment for Children and Adults.
5	Toys, Games, and Leisure Activities for Children and Adults
6	Toys, Games, and Remote-Controlled Devices for Children and Hobbyists
7	Toys and Games for Children Aged 4-12 Years Old with Educational or Creative Focus.
8	Toys and Hobbies for Children's Creative Play and Education.
9	"Children's Party and Playtime Entertainment, Learning, and Leisure Products"
10	Toys and Games for Children aged 3-12, focusing on Creative Play, Learning, and Imagination Development.
11	Toys, Games, and Educational Products for Children and Adults
12	Toys, Games, and Puzzles for Children's Development and Entertainment
13	Children's Toys and Entertainment Products for Recreational and Educational Use
14	Educational Toys and Children's Party Supplies with Recreational Electronics Accessories
15	Toys and games for children's creative development and entertainment

Llama3 has generated 15 category names that capture the essence of the top 50 products within each category. These category titles provide entrepreneurs with a framework for developing products that cater to the needs of items strongly associated with the most highly recommended products in each group.

7 Limitations and Future Research

The ranking system in Amazon's data has two major limitations. The first is that star ratings are limited to whole numbers. Having a scale with half-star ratings or a 1-10 scale would allow for more accurate recommendations, as it provides more nuanced rating criteria. The second limitation is the skew toward more homogeneous reviews over time. A known contributor to this issue is the presence of paid reviews, fake reviews, and bots, which don't represent real customers and, therefore, distort the data. These reviews tend to be overly positive in an effort to boost sales through social proof—products with more positive reviews are perceived as more trustworthy and are more likely to sell.

Another factor contributing to this issue may be Amazon's platform optimisation. Amazon benefits from higher ratings as they lead to more sales, build greater trust in the platform, and create a self-fulfilling cycle of more positive reviews and more purchases. Positive reviews encourage repeat purchases, as prior favourable experiences are strong predictors of future behavior. As a result, Amazon's algorithms likely promote homogeneity in ratings, making ratings themselves less useful for predictions. Instead, previous purchases combined with latent factors may serve as an effective predictive model, eliminating the need for computational resources focused on ratings.

Additional limitations stem from the processing power of my machine, which runs these models locally, as well as the need for up-to-date data. Hosting this application on scalable infrastructure with real-time data would provide valuable insights into product market gaps and enable quick decision-making. This approach could help identify emerging trends in e-commerce and allow for rapid scaling of products in time for peak seasons, such as Christmas.