Towards Mitigating Bias in Online Reviews: An Application to Amazon.com

Charlie Walker
Stanford University

Scott Buttinger
Stanford University

Sam Wharton
Stanford University

1 Introduction

Many of today's e-commerce platforms include a user generated review and rating system that is designed to give customers the best possible information about the quality of available products or services. These platforms leverage positive endorsements to promote items that may be of interest to a consumer, while negatively reviewed items are pushed to the bottom of a user's search results. The accumulation of positive reviews carries enormous economic value for merchants within the platform, yet simultaneously introduces vulnerability to bias. It is thus in the best interest of consumers and merchants to minimize bias in the way that reviews are created, aggregated, and presented. This project investigates a Bayesian method for mitigating bias in an attempt to more accurately represent true product quality on the popular e-commerce marketplace, Amazon.com.

2 Previous Work

2.1 Social Influence: A Randomized Experiment

L. Muchnik et al. investigate the influence of popular opinion on decision making. To do so, the research team analyzed a large-scale social news aggregation website in which users contribute news articles and discuss them. Users of the site discuss posted articles by writing comments that can be upvoted or downvoted by other users of the site. Each comment keeps a running counter equal to the number of upvotes minus the number of downvotes.

Over the course of the study, the researchers observed significant positive herding effects in many categories of news, as they found that users were more likely to upvote a comment that had been up-treated upon creation than one that was untreated. Similarly, the study observed statistically significant opinion change (the likelihood that a user upvotes a comment when they most frequently downvote comments, or vice-versa), and an increase in turnout (the likelihood that a user rates a given comment rather than just reading it).

2.2 Optimal Aggregation of Consumer Ratings: An Application to Yelp.com

Dai et al. develop a model to construct optimal ratings for restaurants in the Yelp network. Dai et al. develop a structural framework that allows reviewers to vary in accuracy (some reviewers are more erratic than others), stringency (some reviewers leave systematically lower ratings), and social concerns (some reviewers may conform to prior reviews). They also note that, whether it be due to shifting preferences or shifting quality, the "true quality" of a product vis-à-vis consumer tastes may evolve over time, implying that current product quality is better reflected in recent reviews.

Using their model, Dai et al. estimate key parameters to evaluate optimal average ratings across restaurants, and compare their results to the arithmetic mean by Yelp. Stringency and accuracy are identified by evaluating how far different reviewers are (in expectation) from long-run averages of the products they review. Changes in product quality are modelled as a martingale process. The authors find that most of the optimal-vs-simple-average difference is driven by evolution of restaurant quality.

2.3 Star Quality: Aggregating Reviews to Rank Products and Merchants

In "Star Quality: Aggregating Reviews to Rank Products and Merchants", M. McGlohon et. al. cite a Wall Street Journal article that notes the "average rating for top review sites is a huge 4.3 out of 5

stars," a feature replicated in our data (see Figure 2).

The authors use nine separate techniques to produce a more accurate aggregate score with data from Google Product Search. These include median reviews, lower bound of a normal and binomial confidence interval, percentile of order statistic across websites, and filtering for anonymous or spam detection. One issue they find is that they could not easily test the accuracy of their algorithms because a product's true quality is not known. To work around this, they develop a method of making a test set by taking pairs of reviews from the same author and see if their methods can choose which product received a higher rating. Using this, they find that none of their algorithms were any more powerful than simply taking the average aggregate rating; the average aggregate ranking is 70% accurate across all 3 datasets, with the lower bound of the normal confidence interval replicating those results, and all other methods slightly worse.

2.4 Commentary

These three papers jointly motivate our problem, and emphasize the importance of mitigating bias (as well as the consensus that online bias should be mitigated). Given our data and evaluation of the different methods that have been used in the literature, we chose to explore a methodology proposed by Ge et al. in their paper *A Bayesian Model for Calibrating Review Scores*. Their model, and our extensions, are discussed in Section 4.

3 Data

3.1 Overview

We use an Amazon product dataset compiled by Julian McAuley at UCSD.[1] The data was collected by crawling the Amazon website and contains product metadata and review information for more than 9 million reviews in 22 product categories, with dates spanning May 1996 to July 2014. For each review we have kept the following information:

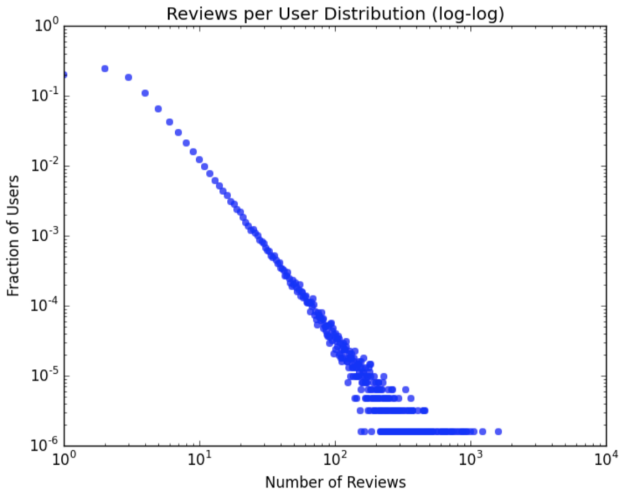
- Reviewer ID
- Reviewed product's ASIN (Amazon Standard Identification Number)
- Number of "helpful" votes
- Product category
- Overall (i.e. average) product rating
- Review date and time

McAuley's dataset is restricted to those users and products with at least 5 reviews. We further restricted our data to ensure that we had sufficient richness per user and reviewer to fit our model while being cognizant of computational restraints. Our final dataset contained approximately 5000 reviews from 300 users and 300 products. Each user and product had a minimum of 10 and 5 reviews, respectively. We also generated a "helpfulness" scale equal to the percent of helpful votes on a rating, translated upwards by one (the reasoning for becomes clear in Section 4). Data was split into 80% train and 20% test.

3.2 Network Characteristics

Viewing our dataset as a bipartite graph, we plot the number of reviews per user in the full dataset (i.e. prior to our constraints and transformations).

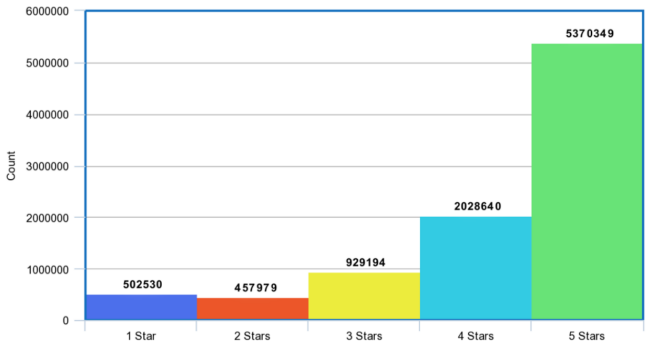
Figure 1: Distribution of Reviews Per User



In Figure 2 we create a histogram of star rating counts from the full Amazon dataset. This histogram replicates a well-documented J-shaped distribution of ratings in e-commerce ratings. Although not objectively problematic, there are certainly reasons why marketplaces would prefer a less skewed distribution of ratings.

[1] http://jmcauley.ucsd.edu/data/amazon/

Figure 2: Rating Histogram



4 Model

4.1 Model Baseline: Platt-Burges

To formalize our problem, we first introduce some notation: index products with p and reviewers with u ("user"). r_{pu} is the score reviewer u gave to paper p, g_p is the unobserved objective quality ("goodness") of product p, and b_u is the unobserved bias of reviewer u. We assume reviewer bias is constant for a given reviewer u. Further, let H_u indicate the average helpfulness of review r_{hu}, as rated by their peers on Amazon (see Section 3). Platt and Burges developed a model for the NIPS conference that can be written as:

$$r_{pu} = g_p + b_u + \epsilon_{pu} \tag{1}$$
$$\epsilon_{pu} \sim \mathcal{N}(0, \sigma) \tag{2}$$

Platt and Burges solve the above system by minimizing the regularized least squares problem:

$$L = \frac{1}{2}\Sigma_p \Sigma_{u \in R_p}(r_{pu} - b_u - g_p)^2 + \frac{1}{2}\lambda \Sigma_u b_u^2 \tag{3}$$

where λ is the regularization parameter, and R_u is the set of reviews for product p. Ge et al. propose a Bayesian reformulation of the Platt-Burges model (which we use as our baseline) which allows for the development of more sophisticated models.

4.2 Bayesian Platt-Burges

The Ge et al. reformulation of the Platt-Burges baseline begins by placing the following priors on the g_p and b_u:

$$g_p \sim \mathcal{N}(3, 1) \tag{4}$$
$$b_u \sim \mathcal{N}(0, c) \tag{5}$$

where the hyperparameters corresponding approximately to the distribution of ratings in the Amazon database (the true distribution is $r_{pu} \sim \mathcal{N}(4, 1)$, but we intentionally adjust our hyperparameters downwards to yield a posterior centred closer to the middle of the 1-5 scale.) The prior on b_u has a regularization effect similar to the effect of λ in the Platt-Burges model. The joint density of review scores r_{pu} is:

$$r_{pu} \sim \mathcal{N}(g_p + b_u, 1/A_{pu}) \tag{6}$$
$$A_{pu} = \begin{cases} 1 & u \text{ reviewed } p \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where A_{pu} can be interpreted as the pu^{th} entry of the adjacency matrix of our data. This initial reformulation is exactly equal to the Platt-Burges model, and it is used as our baseline.

4.3 Model 1

Our first generalization of Platt-Burges comes from reformulating out density of review scores as:

$$r_{pu} \sim \mathcal{N}(g_p + b_u, 1/(dA_{pu})) \tag{8}$$
$$d \sim \mathcal{N}(a_d, b_d) \tag{9}$$

This model is immediately more general than the baseline above because of the variance parameter d, representing the precision of the Gaussian.

4.4 Model 2

Our second model expands on Model 1, and incorporates helpfulness ratings: in our dataset, each review is accompanied by a helpfulness rating left by other users of the site; users have the opportunity to rate a review as being "helpful" or "unhelpful." We thus have a value between 0 and 1 indicating the helpfulness of a score. Rescaling helpfulness to a value bounded between 1 and 2, Model 2 is defined by:

$$r_{pu} \sim \mathcal{N}(g_p + b_u, 1/(dH_{pu})) \tag{10}$$

Since $H_{pu} > 1 \ \forall \ p, u$, and is increasing in the helpfulness rating on a given review, a higher helpfulness rating decreases the variance on that particular review.

In general, the three models above can be expressed in terms similar to the Platt-Burges specification:

$$r_{pu} = g_p + b_u + \epsilon_{pu} \tag{11}$$
$$\epsilon_{pu} \sim \mathcal{N}(0, \tau_{pu}) \tag{12}$$

The variance scaling parameters in Models 1 and 2 above have the effect of reducing the size of ϵ_{pu}.

5 Inference

With the specifications above, the crux of our problem is estimating the latent variables in our model; specifically, g_p, b_u, c, d. Although an EM algorithm could be used for problems similar to ours, we chose to perform inference using Markov chain Monte Carlo (MCMC). Specifically, we used STAN, which provides an excellent engine for Bayesian inference. STAN is based on the No-U-Turn Sampler (NUTS), which provides several advantages over Hamiltonian Monte Carlo (HMC): HMC is highly sensitive to user-specific parameters, and NUTS is generally as efficient as a well tuned HMC method, while maintaining its advantages over other MCMC methods.

Compiling and fitting the STAN models detailed in the Appendix estimates the posterior distribution of our latent variables:

$$P(g, b|R_{train}) = \frac{P(R_{train}|g, b)P(g)P(b)}{\int P(g)P(b)P(R_{train}|g, b)dgdb} \tag{13}$$

where the priors on g and b are given by equations (4) and (5).

6 Model Evaluation

Evaluating the strength of our model is generally challenging since, by the definition of our problem, we do not observe the ground truth values of g_p and b_u. The key metric by which we evaluate and compare our models is RMSE, calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{|R_{test}|} \sum_{r_{ij} \in R_{test}} (r_{pu} - \overline{g_p} - \overline{b_u})^2} \tag{14}$$

where R_{test} is the hold-out set of test reviews. $\overline{g_p}$ and $\overline{b_u}$ are the mean of the draws from (14). That is,

$$\overline{g_p} = \frac{1}{S} \sum_s g_i^{(s)}$$
$$\overline{b_u} = \frac{1}{S} \sum_s b_i^{(s)}$$

This metric is tractable because we restricted our dataset to reviewers and products who have multiple reviews (see Section 3). We thus have an average estimated "goodness" for every product in R_{test} and average estimated bias for every user. If the assumptions of our model are correct, the value $b_u + g_p$ should be a close estimate for the product-reviewer pairing b_{up} in our test set.

7 Results

Figures 5-10 in the Appendix show estimated posterior distributions for reviewer bias and product quality for a subset of reviewers and products in our data. The different models generated different distributions for individual products and reviewers, and performed differently on a range of metrics.

7.1 Prediction Accuracy

Table 1 shows the RMSE for our three models: the baseline Platt-Burges, Model 1, and Model 2.

Table 1: RMSE

	Baseline	**Model 1**	**Model 2**
RMSE	0.9322	0.8353	0.8403

Our extensions to the Platt-Burges model were both able to outperform the baseline model. However, the simpler model 1 performed better in terms of predictive accuracy than model 2. This is surprising given that the second model explicitly incorporated reviewer helpfulness as a variance scale, and we would expect the model with greater flexibility to outperform a simpler one. This may be a sign that helpfulness votes are not a reliable metric by which to evaluate the quality of a particular review (we discuss other potential variance scales in a later section).

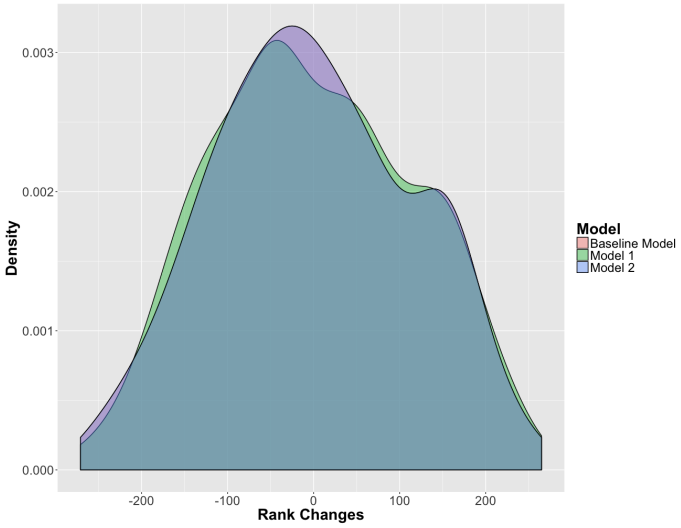
Evaluating whether an RMSE of 0.84 reflects a good fit requires some subjectivity. Although the number

is low in absolute terms, we showed in Figure 2 how the majority of product reviews are skewed towards the 4-5 range; a prediction error of 0.8 could thus be considered dramatic in light of the structure of our initial data.

7.2 Rank Changes

In Figure 3 we show how our model impacted the ranking of products in our subsample. This was motivated by the idea that raw scores are not in themselves as important as relative scores. We see that the density of rank changes is very similar for all three models, with most of the mass centred around 0. We were surprised to see that the tails of the density were quite thick, indicating a fair degree of movement.

Figure 3: Change in Product Rank



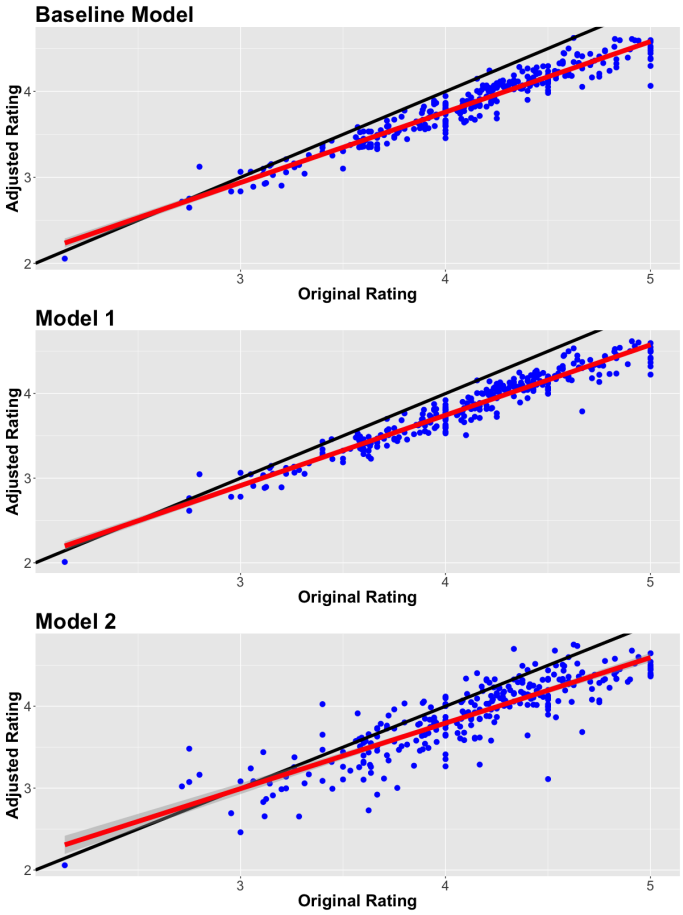
7.3 Ratings Changes

Figure 4 plots the adjusted rating against the original rating. All of our models have a line of best fit with slope < 1, indicating that most reviews are adjusted downwards. To some degree this is expected, as our prior (Equation 4) assumed a mean rating below that of our data.

8 Conclusion

The purpose of this paper was to demonstrate a Bayesian reformulation of Platt and Burges' NIPS reviewer calibration technique, and apply extensions to the model to a novel dataset. Our results

Figure 4: Calibrated Scores for Different Models



demonstrate that there are groups of reviewers that systematically rate products higher or lower than their peers, and that this latent bias can be detected with a straightforward Bayesian model.

Moving forward, there are a number of directions in which this research can be taken. There are some obvious improvements upon our model: we could, for example, develop more sophisticated variance scales that take into account factors such as product expertise (leveraging the very rich product category information in our dataset).

Extending our work, behavioural research could examine whether calibrated reviews are more useful to users of e-commerce platforms. Further, if platforms such as Amazon were to use models detecting reviewer bias, they could implement behavioural nudges for their users to encourage more objective ratings; alternately, they could weight reviews with high bias differently in their aggregation methods than those with less bias.

9 Works Cited

Dai, W., G. Jin, J. Lee, & M. Luca. 2014. Optimal Aggregation of Consumer Ratings: an Application to Yelp.com. *NBER Working Paper.*

Ge, H., Welling, M., Ghahramani, Z. 2013. A Bayesian Model for Calibrating Review Scores. http://mlg.eng.cam.ac.uk/hong/nipsrevcal.pdf.

McGlohon, M., Glance, N., Reiter, Z. 2010. Star Quality: Aggregating Reviews to Rank Products and Merchants, in: Proceedings of Fourth International Conference on Weblogs and Social Media (ICWSM), pp. 114-121.

Muchnik, L. Aral, S., Taylor, S. 2013. Social Influence Bias: A Randomized Experiement. *Science* 341.

10 Appendix

10.1 Posterior Distribution of Paper Goodness and Review Bias (sample)

10.1.1 Baseline Model

Figure 5: Reviewer Bias

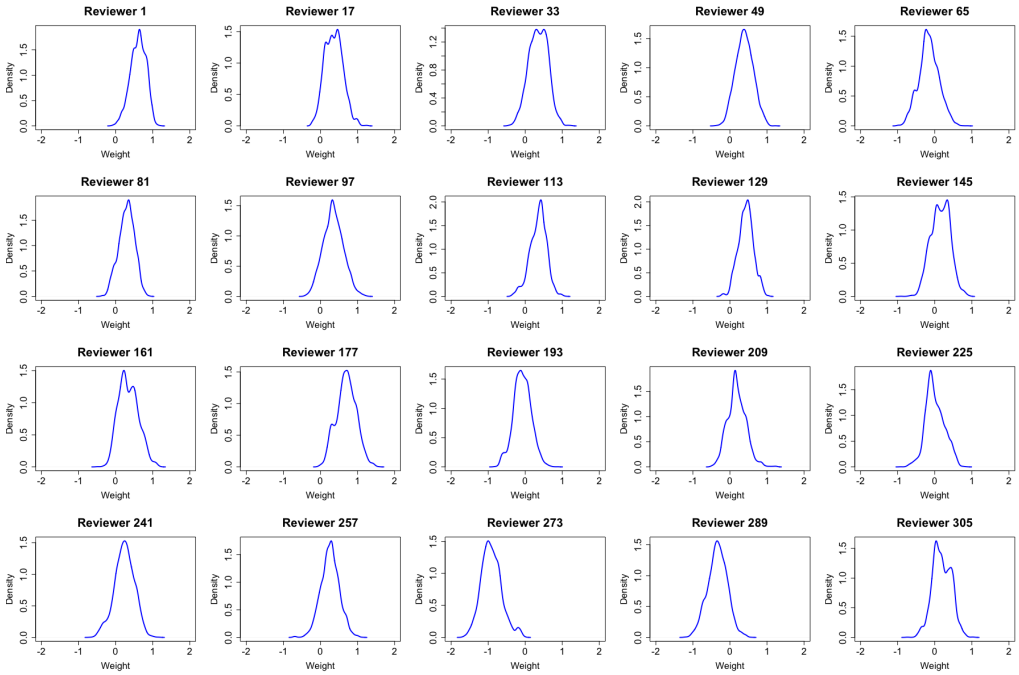
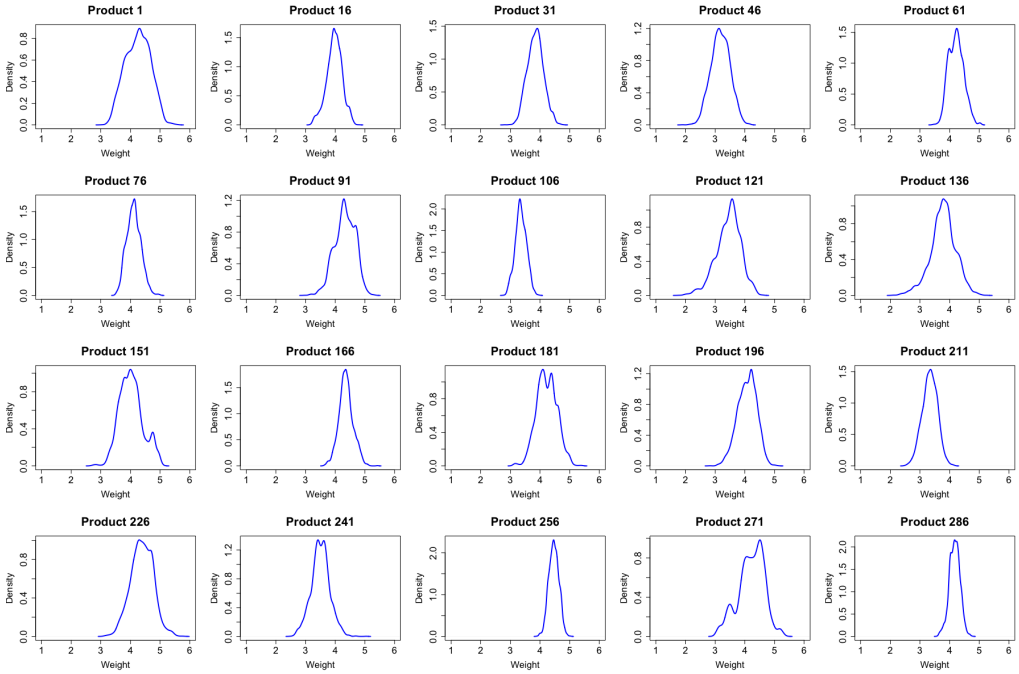


Figure 6: True Quality



10.1.2 Model 1

Figure 7: Reviewer Bias

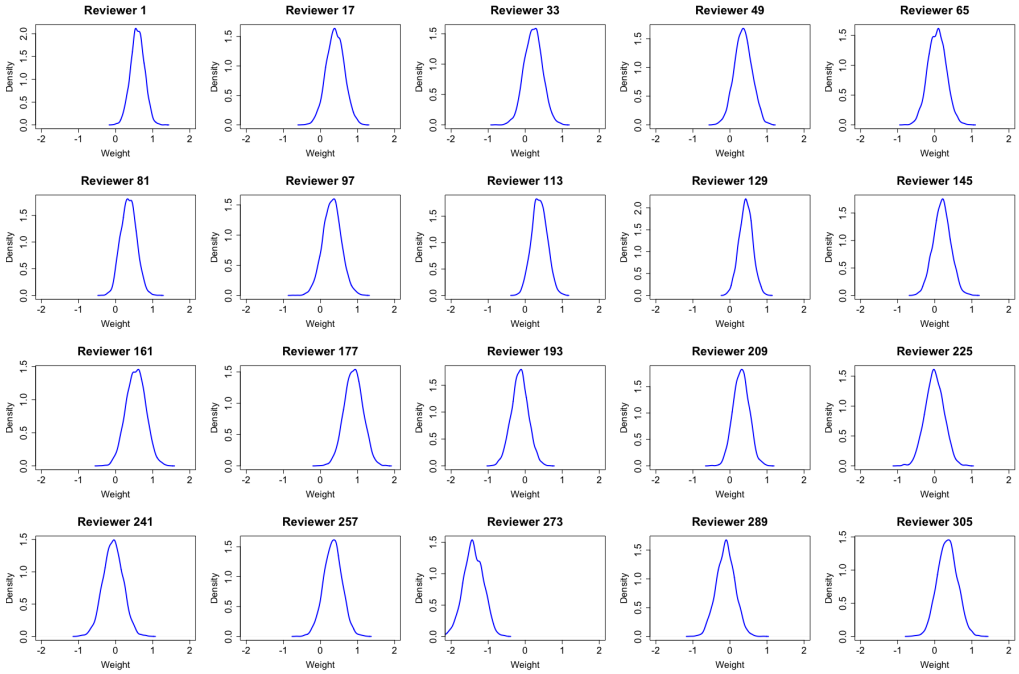
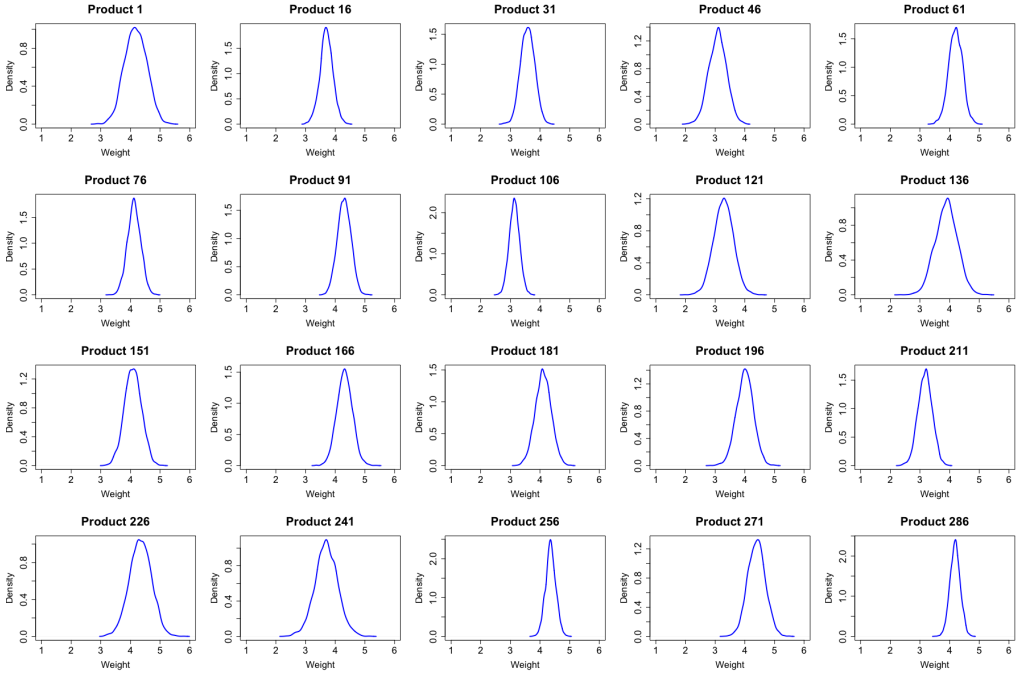


Figure 8: True Quality



10.1.3 Model 2

Figure 9: Reviewer Bias

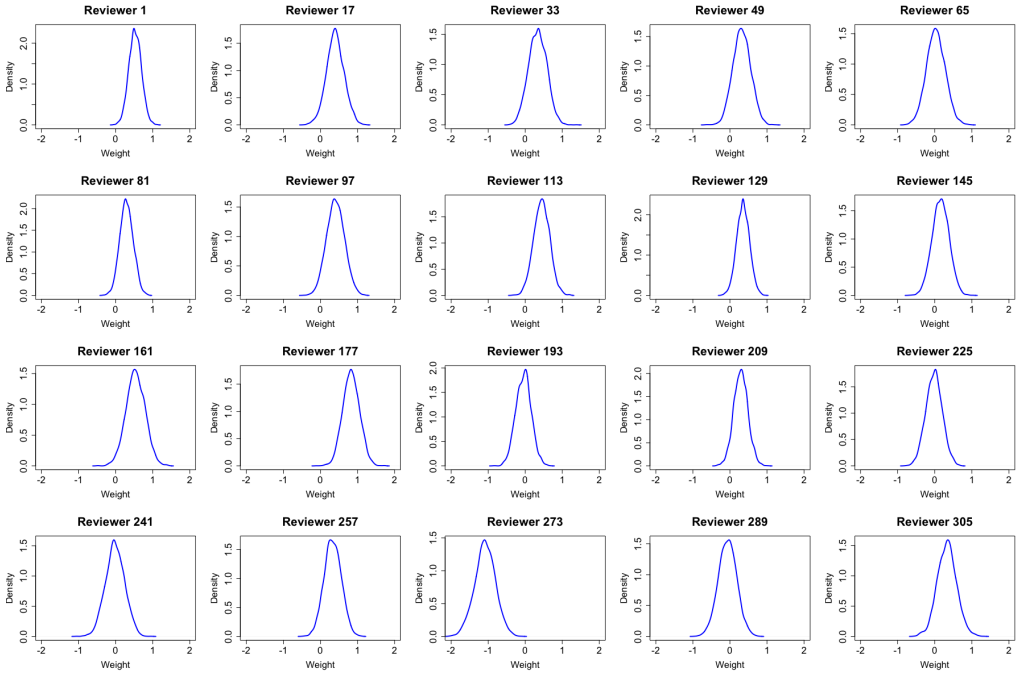
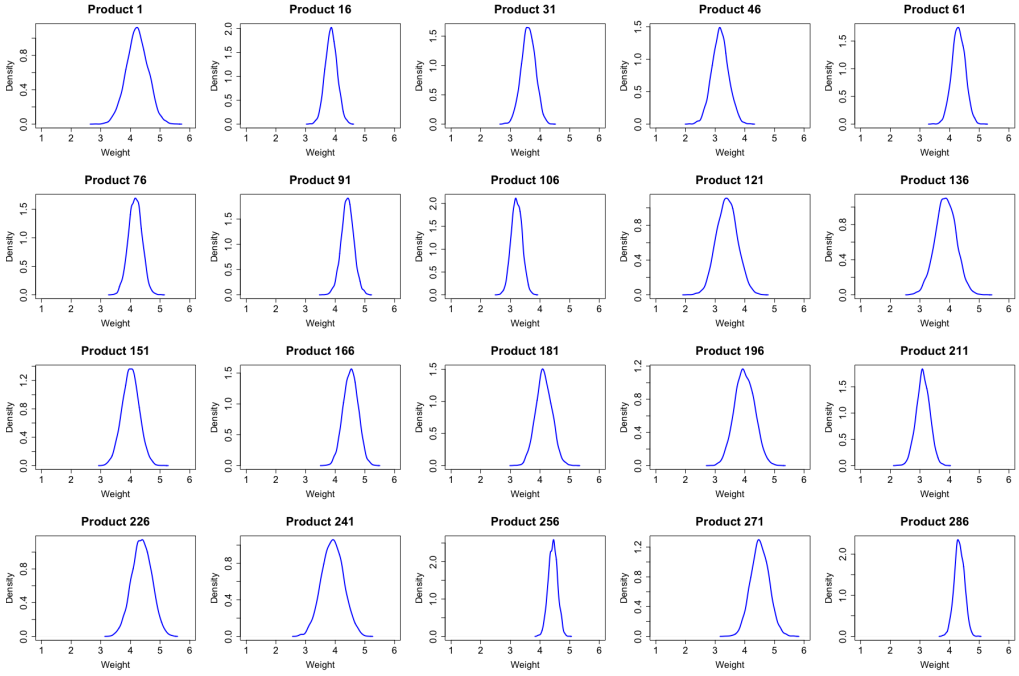


Figure 10: True Quality



10.2 STAN Code

10.2.1 Baseline

```
data {
    int<lower=1> I; // number of products
    int<lower=1> N; // number of reviews
    int<lower=1> J; // number of reviewers
    vector<lower=0> [N] scores; // review scores vector
    vector<lower=0> [N] helpfulness; // review helpfulness
    int reviewerID[N]; // reviewer ID for each review
    int prodID[N]; // product ID for each review
}
parameters {
    vector [I] truescore;
    vector [J] reviewbias;
    real<lower=0> c;
    real<lower=0> d;
}
model {
    c ~ gamma(1,1);

    truescore ~ normal(3,1); // TODO: change hyperparameters for data
    reviewbias ~ normal(0, 1/c);
    for (i in 1:N) {
        scores[i] ~ normal(truescore[prodID[i]]
                           + reviewbias[reviewerID[i]], 1);
    }
}
```

10.2.2 Model 1

```
data {
    int<lower=1> I; // number of products
    int<lower=1> N; // number of reviews
    int<lower=1> J; // number of reviewers
    vector<lower=0> [N] scores; // review scores vector
    vector<lower=0> [N] helpfulness; // review helpfulness
    int reviewerID[N]; // reviewer ID for each review
    int prodID[N]; // product ID for each review
}
parameters {
    vector [I] truescore;
    vector [J] reviewbias;
    real<lower=0> c;
    real<lower=0> d;
}
model {
    c ~ gamma(1,1);
    d ~ gamma(1,1);

    truescore ~ normal(3,1); // TODO: change hyperparameters for data
    reviewbias ~ normal(0, 1/c);
    for (i in 1:N) {
        scores[i] ~ normal(truescore[prodID[i]]
                           + reviewbias[reviewerID[i]], 1/d);
```

```
    }
}
```

10.2.3 Model 2

```
data {
    int<lower=1> I; // number of products
    int<lower=1> N; // number of reviews
    int<lower=1> J; // number of reviewers
    vector<lower=0> [N] scores; // review scores vector
    vector<lower=0> [N] helpfulness; // review helpfulness
    int reviewerID[N]; // reviewer ID for each review
    int prodID[N]; // product ID for each review
}
parameters {
    vector [I] truescore;
    vector [J] reviewbias;
    real<lower=0> c;
    real<lower=0> d;
}
model {
    c ~ gamma(1,1);
    d ~ gamma(1,1);

    truescore ~ normal(3,1); // TODO: change hyperparameters for data
    reviewbias ~ normal(0, 1/c);
    for (i in 1:N) {
        scores[i] ~ normal(truescore[prodID[i]]
                           + reviewbias[reviewerID[i]], 1/((helpfulness[i]+1)*d));
    }
}
```