Capturing Heterogeneity in Engagement on a Social Media Platform

Lynd Bacon[1], Peter Lenk[2], Danielle Murray[3],
Jean Durall[3]
[1]Loma Buena Associates, P.O. Box 620960, Woodside, CA 94062
[2]Stephen M. Ross School of Business, University of Michigan, 701 Tappan St., Ann Arbor MI USA
[3]Shutterfly, Inc., 2800 Bridge Parkway, Redwood City, CA 94065 USA

Abstract

Many firms need to understand and manage customer engagement in order to be profitable. This is true in particular for firms that use a "freemium" business model where revenues depend on up-selling users of basic, no-cost digital services. And as with marketing in general, quantifying customer heterogeneity in ways that are relevant to decision-making can provide big returns. But being able to do so in today's digital marketing big data environments presents some particular challenges to providing precise and also timely results. We describe a latent variable model of user engagement on a social media platform for use in a distributed data management environment. Engagement parameter estimation is by way of MCMC simulation procedures. Our approach has roots in psychometrics, but is broadly applicable to any application context in which observable indicators of engagement can be identified for modeling purposes, even in "big data" grid or server cluster environments.

Key Words: user engagement, CRM, Hierarchical Bayes, marketing, big data, social media

1. Introduction

Engagement can be defined in many different ways. It can be thought of as being primarily attitudinal, as it often is in the case of consumers' relationships to brands. Or, it might be defined as primarily behavioral, as it sometimes is in the context of games. For our applications to individuals' use of social media platforms, our preference is to think of engagement as a latent user state or characteristic. We assume that the extent to which a user has it can be indicated by "fallible" (i.e. noisy) categorical variables reflecting the user's behaviors. The observed behavioral variables, or "indicators," as we refer to them here, may be categorical or continuous measures. They are defined based on exploratory research and domain expertise, and they are specific to a particular application. They may be, for example, observed measures summarizing particular activities on a particular platform during a defined period of time. Or they might be linguistic codes representing semantic content inferred from user-generated text expressing opinions, comments, or product evaluations. Our general modeling approach is to use Markov Chain Monte Carlo (MCMC) simulation methods to approximate the joint posterior density of the parameters of interest based on a Hierarchical Bayes specification.

In what follows we describe two simple variations on our latent variable engagement model for daily use as an "engagement thermometer" in a big and distributed data environment. The statistical modeling of large amounts of data is not a new challenge for marketing scientists, of course (e.g. Naik et al., 2008). But with the rapidly growing availability of massive amounts of digital data relevant to marketing, and with the concomitant development of new methods for managing the data, the challenges, as well as the opportunities, for statistical applications are ever greater.

2. Latent Engagement Model

We have previously described (Bacon et al., 2009) the basic engagement model we apply here as our daily engagement thermometer. It is essentially the same as the standard normal ogive Item Response Theory models used in standardized educational testing with binary observed response variables (Bacon et al., 2004). At its simplest, it consists of assuming a single continuous latent variable on which both users and engagement indicators are measured. Given that we have J indicators and N users, we model the probability of indicator j being observed for user i, $P(Y_{ij} = 1)$, as:

$$p(Y_{ij} = 1) = \Phi(z_{ij}),$$

where:

$$z_{ij} = \beta_{1j}\theta_i - \beta^*_{oj}$$

z_{ij} is a continuous latent variable, Φ is the cumulative normal distribution, θ_i is the engagement score of user i, and the β's are parameters for indicator j. β_{1j} indicates how well indicator j discriminates between users whose scores (their θ's) fall above or below it on the latent variable. The larger β_{1j} is, the better indicator j discriminates between users with θ_i's above it and those with θ_i's below it. Appropriately chosen indicators will have nonnegative β_{1j}'s. β_{0j}^* divided by β_{1j} is where indicator j is located on the latent variable, and is in units of θ. The more positive the location of j, the higher is the level of engagement of users for whom j is observed. Figure 1 provides a directed acyclic graph (a DAG) for a response on a single indicator measurement, Y_{ij}. Note that with J indicators and N users, there would be "N" θ_i's, one for each user, and J pairs of β's, one pair for each of the J indicators.

The appropriateness of this model depends on the tenability of a number of assumptions that include unidimensionality of the engagement construct, conditional independence of the indicator responses, and item parameters working the same way for all users. Fortunately, all of these assumptions are testable. In return, this simple specification brings with it some important advantages.

Both the user scores and indicator locations are measured on the same dimension, which has units of θ. Another benefit is that different sets of indicators can be developed so as to measure the same θ_i scores. A third benefit is that if it becomes desirable to take into account new kinds of user data, say from new data sources, or as a result of new features

and capabilities being made available to users, new indicators can be added while still being able to estimate the same latent engagement scales. Finally, the model can be extended easily to multiple dimensions.

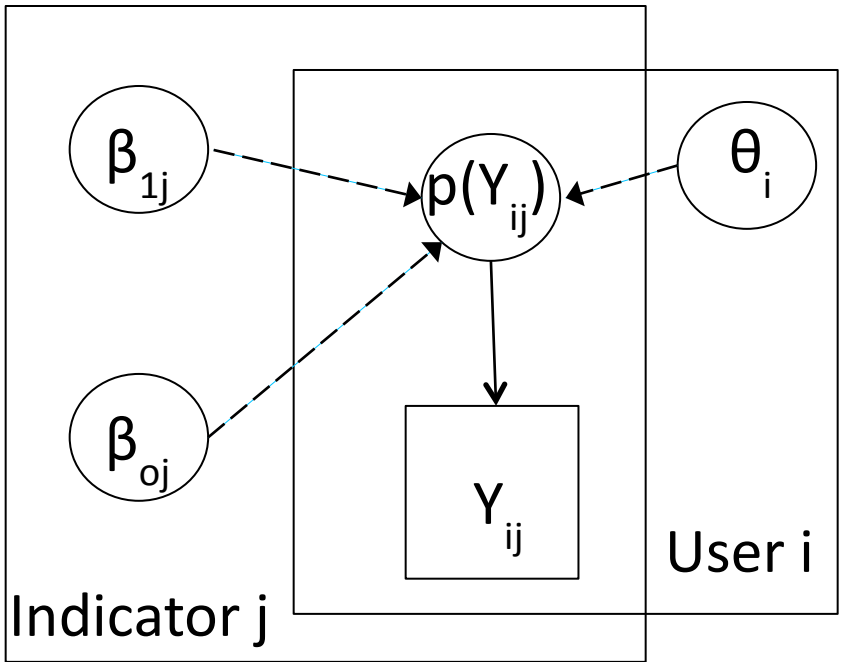

Figure 1. Directed Acyclic Graph for Engagement Indicator Model. This graph is for response on one of "J' binary indicators. By assumption, responses on different indicators are conditionally independent.

Conventional Markov Chain Monte Carlo(MCMC) simulation procedures can be used to estimate the joint posterior density of our model's parameters. MCMC methods are computationally intensive, and so the feasibility of using them for any particular application depends on how much data are involved, and also how quickly results must be obtained. It also depends on how the data are organized and stored. We'll consider the issue of using big and complex data after we consider how our simple model might be estimated using data that are "small enough."

In a "conventional" application context for this sort of model, with say no more than thousands or hundreds of thousands of users, and a perhaps tens or up to a hundred indicators or so, and data that can be quickly read, we could estimate the joint posterior density of our parameters using a Gibbs sampler and data augmentation (Albert & Chib, 1993). Assuming reasonable priors and initial values, our algorithm for estimating the joint posterior density of the parameters might go as follows.

I. For each user i and indicator j, draw z_{ij} given the current values of the other parameters;
II. For each user i, draw θ_i, given the current values of the other parameters;
III. For each indicator j, draw β_{1j} and β_{oj} given the current values of the other parameters;
IV. Go back to Step I until done.

This algorithm is quite simple and easy in principle to implement. Only normal distributions are involved, which can be sampled directly from. All the marginal distributions are simple and straightforward to specify. Note, however, that each of the draws in steps I and II only require that the data or parameters for individual users be available. To sample a particular z_{ij} in step I, for example, all that's needed is user i's Y_{ij} indicator value, the current version of i's θ, θ_i, and the current values of indicator j's β's. So even if the data of users are stored in different locations in a big data environment, say for example on different servers, step I is such that z_{ij} can be estimated local to the server that a user's data is stored on. Sampling in step II can be local in the same manner.

The draws in step III, on the other hand, require having access to all users' most recent θ estimates on every iteration of the algorithm. This step is the main computational bottleneck for big data applications, and it may not even be feasible under certain circumstances. Where the amount of data is quite large but still accessible, and when ample time is available to do estimation, particle filter methods (e.g. Balakrishnan & Madigan, 2006; Ridgeway & Madigan, 2002a,b; Fearnhead, 2005) can speed estimation by reducing the number of times the entirely of the data being modeled needs to be accessed. But for applications like our engagement thermometer the benefit of particle filtering is limited due to there not just being a large amount of data, but also because of how it is stored in a production distributed data management environment, and the uncertain availability of computing resources on servers running multiple, mission critical and time sensitive, processes.

2. Big Data Management and Computing

Big data is often managed in cluster computing environments consisting of a large number of commodity servers. When vast and growing amounts of poorly structured data need to be managed, organizations are turning to alternatives to conventional relational database technologies that are commonly referred to as "noSQL" (not only SQL) data management systems. These systems have been developed for distributed data management in large server clusters. Unlike relational databases, the data they contain is not organized in tables, but is stored as key-value pairs, and they do not enforce a fixed schema or data model. Popular noSQL systems include MongoDB, Cassandra, CouchDB, and Hadoop. Hadoop and Apache project (hadoop.apache.org), is the target platform for our engagement thermometer application. Its main components include the Hadoop distributed file system, and MapReduce (T. White, 2012). Hadoop and other noSQL technologies are widely used to manage petabytes of data at large and some small, firms.

MapReduce is a programming framework popularized by Google that distributes the processing of large, complex data files stored across many servers, the kind of environment that our engagement thermometer application is intended to work in. MapReduce programs are usually written in the computer language Java, but they can also be written in other languages that can use standard system input and output by

employing the utility Hadoop Streaming. MapReduce programs generally include one or more map programs, and at least one reduce program. The Hadoop Apache Project (hadoop.apache.org) and many other resources provide detailed information about the MapReduce framework. But very generally speaking, map programs process data on a local basis, and then their local results are combined, sorted, and processed by one or more reduce programs. When creating MapReduce programs, a general design goal is to push as much computation as possible down on to the "worker" servers that run the mapper processes.

We next describe the implementation of two versions of our daily engagement thermometer model using MapReduce. We make some compromises relative to the algorithm described above in order to make using them feasible in a big and distributed data environment, and exploit a convenient feature of this engagement measurement application that greatly reduces the difficulty of using them.

3. Engagement Thermometer Model in a Hadoop Environment

The challenges of using our engagement model in a big, distributed data environment where computing resources may be unpredictably spread across many important production processes compels making compromises. In what follows we describe two versions of it that approximate the model's full joint posterior density, and we comment on the benefits and challenges of each. For each we assume that the indicators are known a priori. We also assume that the engagement construct is at least approximately unidimensional, that the indicator parameters don't vary nontrivially over subsets of users, and that the observed indicator data are independent conditional on the model's parameters.

Both versions of our model need just trivial parallelization of the required computations. But they involve compromises. The first compromise consists of estimating the indicator parameters, the β's, in a separate process using a sample of all users' indicator data. We refer to this as the indicator "calibration" procedure. The sample used can be randomly drawn with a very simple MapReduce program. Given that the parameters of the posterior marginal β distributions have been estimated, they can then be used to produce daily estimates of users' θ_i's.

This first version of our model produces only point estimates of the θ_i's, given users' indicator data, using point estimates of the indicators' β's obtained from having applied the calibration procedure. Maximum likelihood estimates of the θ_i's are easily obtained by providing β point estimates to mapper programs that calculate the maximum likelihood θ_i's locally on servers where the indicator data reside. The resulting estimates are then returned to reducer programs that sort and select cases based on them for management reporting.

If the number of indicators isn't large, a further simplification is convenient: maximum likelihood θ_i estimates can be computed just once when the indicator β's are estimated, and they can subsequently be passed in key-value pair lookup tables to map programs that match users' indicator patterns to θ_i estimates. Our indicators are binary, and so for 10 indicators, for example, only a maximum of 1,024 key-value pairs would need to be passed to mappers. The actual number might be fewer, given that the model has been developed so as to represent degree or levels of engagement, to begin with. Many

indicator patterns would be unlikely to occur at all. Those that do might vary from day to day, of course, and so it would make sense to just calculate θ_i for every possible data pattern, given the indicators being used, and then to pass the same table to map programs for each reporting period for which θ_i's need to be reported.

One weakness of this first version of our engagement thermometer model is that we end up with no information about variation in users' θ_i's: we are calculating just point estimates. As a result, comparisons between users can't be made on other than a nominal numerical basis. We could approximate an expected value for the variances of users θ_i's based using the results from our sample-based estimation of the indicator parameters. But our second version goes further by using simulation to directly estimate actual θ_i variances.

The application of the second version of our model involves making draws of users' θ_i's from their estimated marginal posterior densities using the indicator parameter estimates of the β's distributions obtained from the calibration procedure, while not updating the β's distribution parameters on each iteration using the current draws of users' θ_i's. That is, in the terms of the steps of our basic algorithm that we outlined above, in step III we draw new values of the β's without conditioning on the current values of other parameters: we consider the means, variances, and covariances of each item's β's to be known, and use them to make Monte Carlo draws of the β's (rather than Markov Chain Monte Carlo draws) from the indicators' bivariate normal distributions, on each iteration.

It turns out that this approach makes more sense than would estimating the full posterior density of all the parameters of our model, even if it was feasible to do the latter. The reason is that our engagement measure needs to be a stable, or fixed, metric if day-to-day comparisons are to be interpretable, e.g. so that θ_i's can be meaningfully compared over days. Otherwise, it wouldn't be possible to distinguish actual changes in users' engagement over time from changes in the measurement scale provided by the engagement model. In what follows we describe applying this second version of our model in a pseudo-distributed Hadoop development environment using MapReduce.

4. Example Application

To demonstrate the use of our second version model, we employ a superset of the social media platform user data we have previously reported results for (Bacon et al., 2009). We loaded this data into a Hadoop development environment implemented on a Linux workstation. We use this pseudo-distributed environment to facilitate code development and testing in advance of deploying it on clusters and in other big data environments. This data consists of 100,000 records that include the nine binary behavioral indicators we defined and modeled previously by estimating the joint posterior density of the parameters using the three step algorithm described above(Ibid.). In brief, these binary indicators were chosen to so as to span a range of engagement intensity. Their exact nature is proprietary to the social media platform firm, and so cannot be further described here.

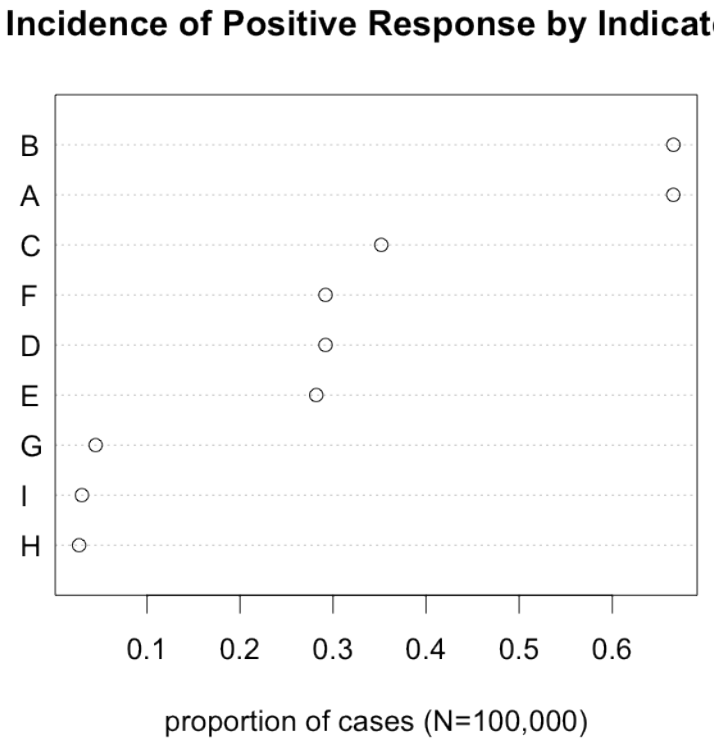

Figure 1. Proportion of users with positive responses to binary indicators, A through I.

The dot charts in Figures 1 and 2 summarize the data. Figure 1 shows the relative incidence of positive responses for each of our nine binary indicators, which we have labeled A through I. In this figure, the indicators are ordered on the ordinate from most frequent at the top to least frequent at the bottom, making it clear that the range of incidences is quite large. Note that all nine indicators had non-zero frequencies of occurrence. Generally speaking, those that correspond to relatively high levels of engagement have the higher frequencies, as observed previously (Ibid.).

Figure 2 summarizes the combinations of positive responses across the nine indicators. The specific patterns are on the ordinate, labeled by the letters of the indicators on which positive responses were observed. The abscissa shows the frequencies of the patterns. 31,712 users had no positive responses, and are not shown in this graph. This particular pattern (no positive indicator responses) was the most frequent in the data.

We implemented our estimation procedures in MapReduce programs by using Hadoop Streaming and code written in the Python language (www.python.org) that employed functions in the Numpy and SciPy packages, scientific computing language extensions for Python(www.numpy.org). To review, our approach consists of first estimating the parameters of the indicator's β distributions in a calibration step, and then applying the results of this step in the estimation of the users' daily θ_i's. The former step should be performed on an as needed basis, e.g. when diagnostics indicate that there may be

changes in the distributions of the indicators' parameters, or if new indicators need to be added. The θ_i estimation step is intended to be done on a daily basis.

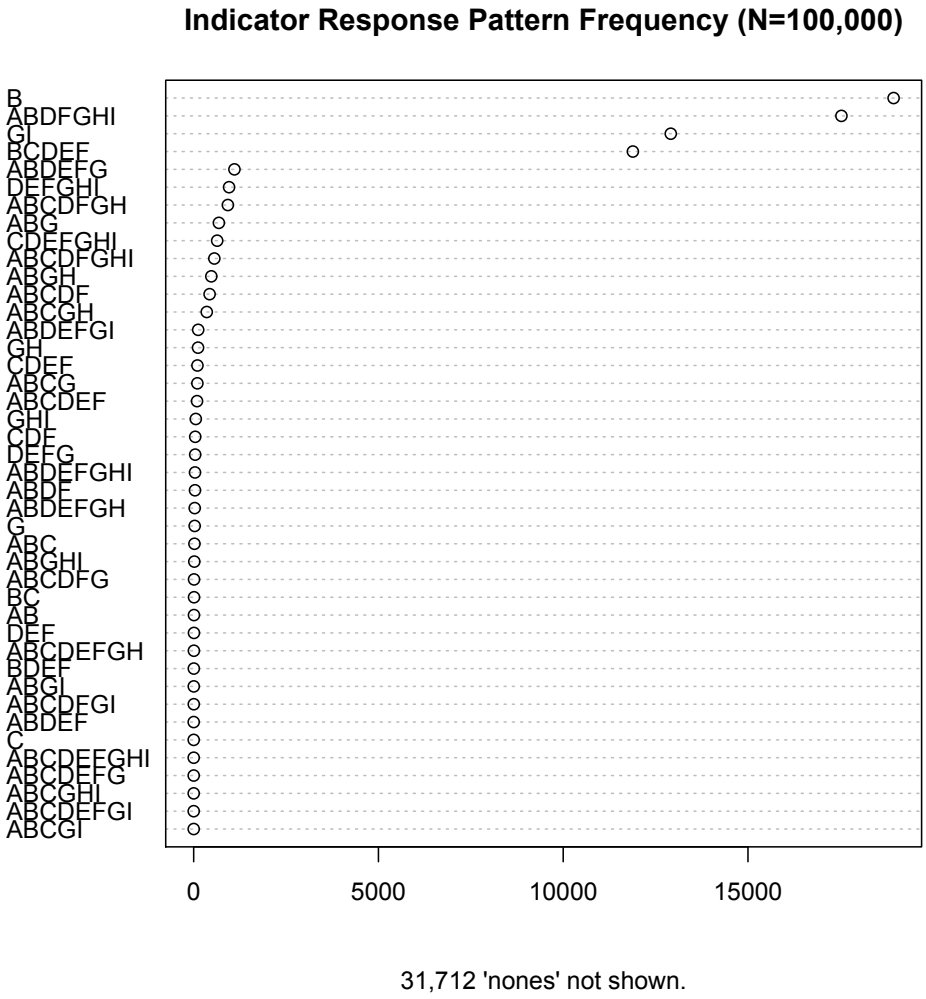

Figure 2. Incidence of specific patterns of positive responses on indicators. Specific patterns of indicators with positive responses are on the ordinate. Frequencies of occurrence are on the abscissa. 31,712 cases without a single positive response are not shown.

To get estimates of the means, variances, and covariances of the indicator β's, we used the data from a random subset of 5,000 users and an initial, indicator "calibration" MapReduce program. We ran our algorithm for 20,000 iterations, dropped the first 10,000 as burn-in, and then retained every fifth sample from the remaining 10,000. Figure 3 summarizes the results for our indicator parameters, where nine indicators are labeled A through I. Each is plotted in terms of the mean of its thinned MCMC chain for β_1 (on the ordinate), and β_o (abscissa), where $\beta_0 = \beta_o^*/\beta_1$. β_o locates an indicator on the engagement scale in units of θ, and relative to the users and to other indicators. Indicators with smaller values of β_0 are those on which positive values are more likely to be

observed. β_1 indicates how well an indicator discriminates between users with lower θ_i's than the indictor's β_0, and those with larger θ_i's. The larger β_1 is, the more sharply the indicator distinguishes between users in terms of their level of engagement.

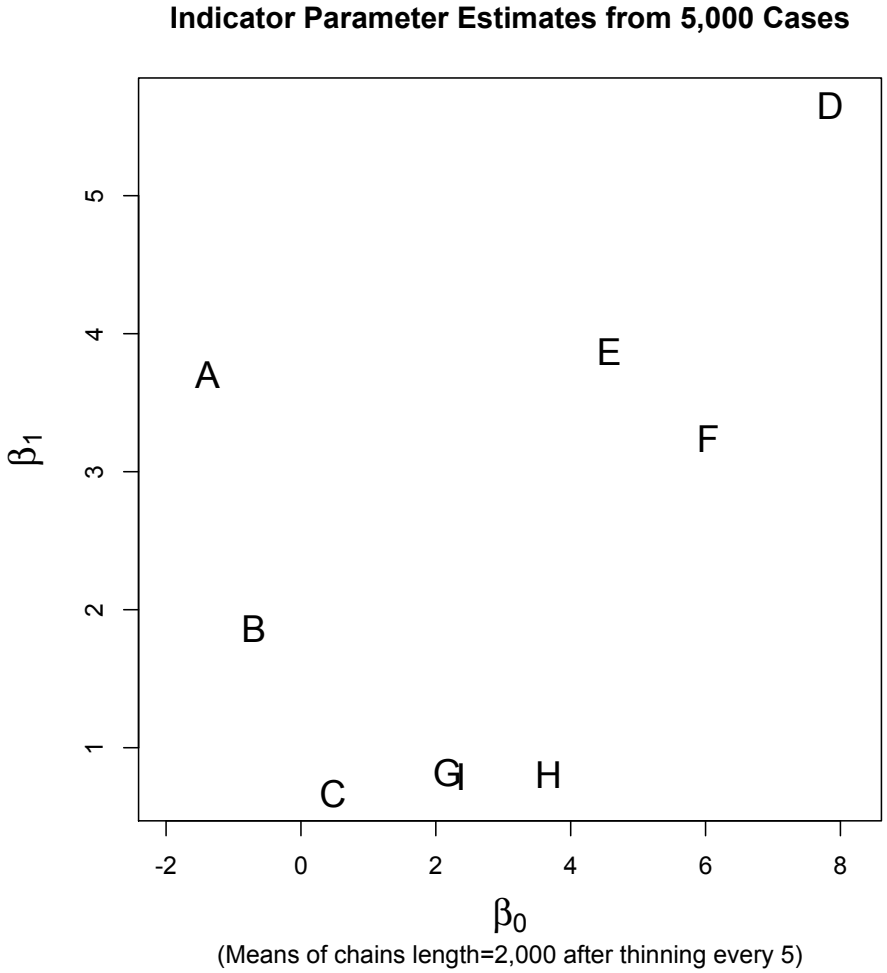

Figure 3. Expected values of indicator β parameters computed based on thinned MCMC chains of length 2,000. β_0 and β_1 are as described in the text. The indicators are labeled by letter, A through I. The abscissa scale is in units of θ.

Daily estimation of user θ_i's employs our estimates of the indicators' β distribution parameters obtained from using our calibration step, and the most current indicator data available for the users. Our algorithm samples from the indicators' β distributions while keeping their means, variances, and covariances fixed, rather than updating them on each iteration. Using our test sample of 100,000 users and a single day's indicator data for them, we ran 20,000 iterations of our algorithm, discarding the first 10,000, and thinning the second 10,000 by keeping every 5th draw to result in chains of length 2,000 for each user. This took about 8 min. on our development workstation.

The histogram in Figure 4 summarizes the distribution of the means of these chains for

our 100,000 users. Clustering of the θ_i's is apparent in this figure, representing high incidence indicator response patterns. A N(0,1) prior was used in estimating them, but it's clear from Figure 4 that the users' engagement levels are not unimodally distributed, the marketing implications of which would be the basis for subsequent exploration.

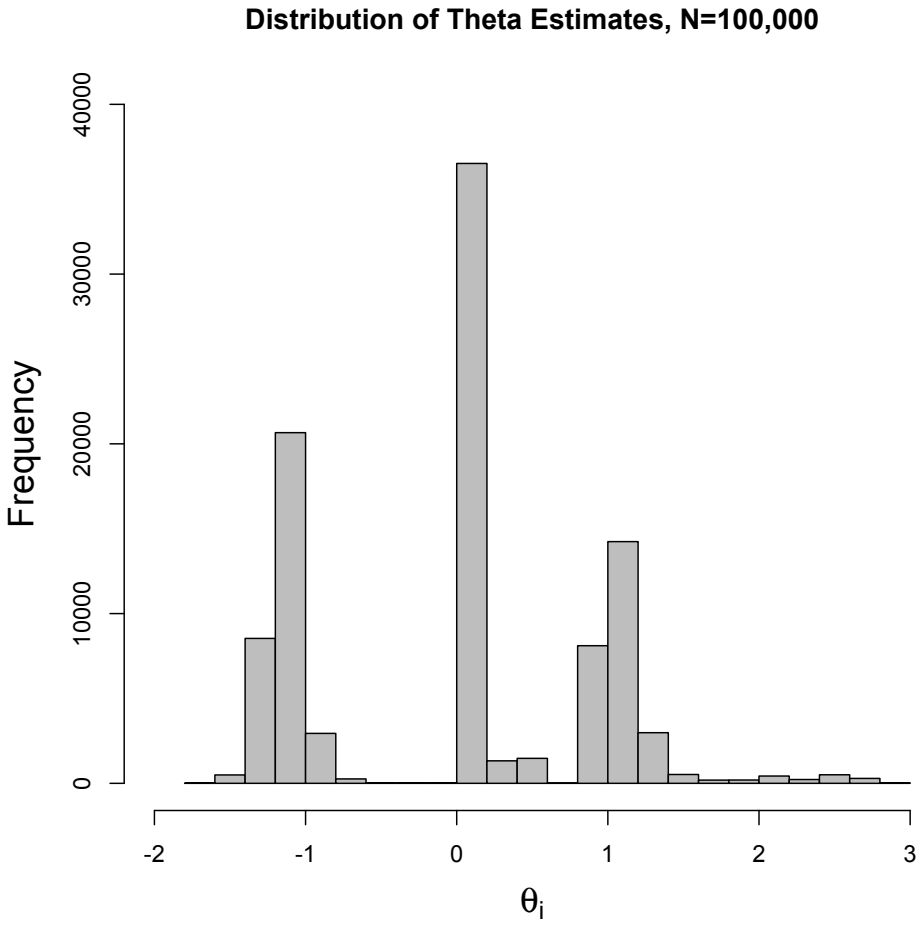

Figure 4. Distribution of estimated mean θ_i's for 100,000 users. Each estimate is based on 2,000 samples of every 5th draw of 10,000 iterations after 10,000 burn-in iterations.

5. Discussion

The two approaches to estimating our engagement model we have described approximate how we would estimate the posterior densities of the model's parameters in a smaller, simpler computing context. They represent some compromises compelled by the intended deployment environment. At the same time they recognize a convenience provided by a goal for this application: producing engagement measures that are on a consistent measurement scale, day to day. The model versions we have described are just two of a number of possible approaches that might be deployed in a big and distributed data management environment. But they illustrate some of the challenges in using even a simple but computationally intensive model in such contexts.

For use in ongoing reporting about user engagement, either model version needs to be complemented with, or supported by, other procedures. Both models require that the parameters of indicator distributions be estimated prior to applying them in estimating the θ_i's. This would be done upon deploying either approach as a separate estimation procedure which we referred to above as the "calibration" procedure. By doing it we are fixing the indicators with respect to the measurement scale on which user engagement is measured. This permits valid day-to-day comparisons of the θ_i engagement scores, a requirement for our model's application.

We discussed selection of indicator variables previously (Bacon et al. 2009). It's important to note here that the general approach we've described is quite flexible in regard to the nature of the indicators and the way they can be selected to emphasize local precision along the dimension of the θ latent variable. The indicators need not be binary; we chose to use only binary indicators for simplicity and for computational ease. The indicators can be ordinal with more than two categories. Once their parameters are estimated they can be selected so that in combination they emphasize reliability at particular locations on latent engagement scale. This is possible because each indicator concentrates the precision with which it discriminates between θ_i's above and below it at its location on the θ scale, β_0, by virtue of the standard normal ogive being the link function governing the predicted probability of a positive response. This indicator discriminability aggregates over indicators, and so a set of indicators can be used that focuses it on particular locations on the underlying scale. So, for example, if the behavior reflected by a specific indicator has particular importance to management decision-making, others can be used in combination with it to enhance the engagement thermometer's ability to distinguish between users with θ_i's above and below it.

One of the benefits of our engagement modeling approach is that new indicators can be developed and applied (and old ones dropped) without necessarily giving up comparability to previously obtained results. This can be accomplished by adding potential new indicators to a version of the calibration procedure while holding fixed the parameters of indicators already in use. Based on the results, new indicators can be selected by considering their β estimates with respect to their location on the engagement variable and their ability to discriminate between users.

In addition to selecting indicators and estimating the parameters of their distributions, other procedures that would be needed to support the production environment deployment of either version of our model include the following. First, we've assumed that our latent engagement variable is unidimensional. This is an assumption that first needs to be tested when the model is initially specified and estimated, and then it should be revisited periodically to determine whether it is still tenable. A convenient way of examining this assumption is to periodically obtain random subsamples of indicator data using a simple MapReduce program and determining whether a sufficient majority, say 2/3's, of their variation is accounted for by a single principal component. The frequency with which this should be done depends on the stability of the measure over time, and can be determined empirically.

Another supporting procedure consists of ensuring that the indicators function in the same way across users over time. In the field of psychometrics, when test items function differently for different subsets of test takers, it's called "differential item functioning,"

and is to be avoided if comparable scores are to be obtained for all individuals being assessed. The question of whether indicators function to measure engagement the same way for all users can be approached in two different ways that are not mutually exclusive. Estimates of indicator β's distributions can be compared across a priori groups to detect differences. A second way to look into differential functioning is models that include a mixing distribution on the indicator parameters. Either of these can be performed using relatively small random samples of the user data.

Yet another support procedure would concern the assumption of conditional independence of indicator responses. The versions of our model assume that responses on our binary indicators are independent conditional upon each model's parameters. This is an assumption whose tenability should be verified before a model is deployed, of course. It should also be revisited periodically as a model is in use as a lack of conditional independence that emerges over time would be indicative of a model misspecification problem. Periodic re-assessment of the conditional independence assumption is yet another supplementary procedure that can be conveniently conducted using random samples of user indicator data.

5. Conclusion

In this paper we have described applying versions of a user engagement model we developed previously (Bacon et al., 2009) in a big and distributed data environment and under a reporting requirement consisting of having to produce daily engagement score estimates for use in monitoring user activity and in support of CRM. Our engagement model is similar to the latent variable models used in standardized, psychometric testing. The two versions discussed here reflect compromises with respect to our previous approach of simultaneously estimating the joint posterior density of all model parameters using MCMC procedures. These compromises make it possible to apply our model in a big, distributed data environment. Using either version requires first identifying engagement indicator variables, and then estimating the distributions of their location and discrimination parameters using random subsamples of user data. For our daily engagement thermometer application, this is a sensible approach given that the engagement measures produced need to be comparable day to day. That is, they need to be on a fixed measurement scale.

References

Albert, J. & Chib, S. (1993) "Bayesian analysis of binary and polychotomous response data." ***Journal of the American Statistical Association***, 88(422), 669-679.

Bacon, L., Lenk, P. & Durall, J. (2004) "Item response theory (IRT) models: Basics, and marketing applications." ***Sawtooth Software Conference Proceedings***, 187-206, Sequim WA: Sawtooth Software, Inc.

Bacon, L., P. Lenk, and D. Murray. 2009. User/Customer Engagement and Sales Conversion in a Social Media Context. In ***JSM Proceedings, Section on Statistics and Marketing***, Alexandria VA: American Statistical Association, pp. 1-8.

Balakrishnan, S. & Madigan, D. (2006) *A One-Pass Sequential Monte Carlo Method for Bayesian Analysis of Massive Datasets.* ***Bayesian Analysis***, 1(2), Pp345-362.

Fearnhead, P. (2005) *Using Random Quasi-Monte-Carlo Within Particle Filters, With Application to Financial Time Series.* ***Journal of Computational and Graphical Statistics***, 14 (4), pp. 751-769.

Naik, P., Wedel, M., Bacon, L., Bodapati, A., Bradlow, E., Kamakura, W., Kruelen, J., Lenk, P., Madigan, D., & Montgomery, A. (2008) Challenges and Opportunities in High-Dimensional Choice Analysis. ***Marketing Letters***, 19, 201-213.

Ridgeway, G. & Madigan D. (2002a) *Bayesian Analysis of Massive Datasets via Particle Filters.* ***Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***, July 23-26, 2002, Edmonton, Alberta, Canada, pp 5-13.

Ridgeway, G. & Madigan D. (2002b) *A Sequential Monte Carlo Method for Bayesian Analysis of Massive Datasets.* ***Journal of Knowledge Discovery and Data Mining***, 7, 301-319.

White, T. ***Hadoop: The Definitive Guide, 3rd Ed.*** Sebastopol, CA: O'Reilly, 2012.