MKTG 646 Final Project

Bayesian Inference
Using a Hierarchical Model, with a Mixture of Normals for the First-Stage Prior

Steven Wong
Department of Civil & Environmental Engineering | Structural Engineering
Stanford University

R codes attached in the end

[Introduction]

This study broadly follows the publication: *State Dependence and Alternative Explanations for Consumer Inertia* by Dubé, Hitsch and Rossi (2010). The goal is to understand how econometric methods can be applied to distinguish between two main sources of consumer inertia: (i) structural state dependence, and (ii) unobserved consumer preference or utility autocorrelation. The idea is to use a mixture of normals to better model persistent heterogeneous consumer preferences, such that source (i) does not appear significant simply by being a proxy for source (ii). Dubé et al. refer to source (ii) as *spurious* state dependence.

As described by Dubé et al., consumer inertia is an observed behavior when a consumer makes consecutive purchases on the same product, despite being faced with known alternatives in the same product category (i.e. not substitutes). However, the two sources imply different price equilibria, and consequently lend themselves to different optimal pricing policies. In particular, *structural* state dependence implies time dependent price equilibria, and thus introduces dynamic pricing incentives.

After accounting for heterogeneous consumer preferences with a mixture of normals, Dubé et al. hypothesized on the potential sources of structural state dependence: loyalty, search and learning. Loyalty is consumers' gravitations towards a particular product brand that does not stem from their innate preferences for the product itself. Search acts as a switching cost: consumers currently purchasing a particular product are faced with additional costs associated with searching for alternatives. Learning also acts as a switching cost: consumers know more about their current choice simply by consuming it, compared to their relative lack of knowledge on the other alternatives. In both of the latter instances, switching cost is a choice barrier placed on the unconsumed alternatives. After several tests, Dubé et al. found loyalty to explain structural state dependence more than the other two.

This study will not explore the aforementioned tests, but will instead focus on the distinction between using a normal versus a mixture of normals 1st-stage prior in a hierarchical model, when accounting for heterogeneous preferences. Aligned with this goal, this study will show an example computation illustrating the distinction, both with and without additionally accounting for state dependence.

[Problem Statement]

In most datasets, there is an information hierarchy, whether explicitly stated as parameters or is latent. The finest level of disaggregation could be at the "individual" level, at which observations are commonly made. Any coarser level of disaggregation would pool or cluster the individual level information into various "sectors", which could be age, gender, geospatial location, time intervals, etc. Oftentimes, the statistical challenge is in making inferences on the individual level, as the number of observations at that level could be sparse enough to render asymptotic methods futile.

The recent explosion of individual level data only aggregate this challenge, since the number of parameters on each individual, k, increases at a much higher rate than the number of observations on each individual, n_i. This is to say: individuals are not performing actions or switching states at a faster rate than before, but we are able to collect more information on each individual and on each change. In fact, given a large state space and a relatively small number of observations, we can very well make no observations on an individual being in an entire subset of the state space. Unbeknown is whether the individual has *low probabilities* of being in those states, or simply *no possibilities*. Overall, the result is a high-dimensional problem on top of a sparse data problem. The two, of course, often goes hand-in-hand.

While coarser level of disaggregation can be informative on a larger sector and can offer a higher level of confidence on point estimations, it looses sight of individual differences. Individual differences are important: product differentiations and market segmentations presuppose heterogeneity. Pooling all the data together, however, assumes homogeneity. Even intentional clustering can precludes true underlying heterogeneity, especially if the clustering is achieved with parameters independent of the dataset at hand. Dubé et al. mention that many literatures segment a subset of parameters to be estimated at the individual level (leaving the rest to be estimated at a higher level of disaggregation), but argue that such cutoff is rather subjective.

To address both the information sparsity and the potential heterogeneity at the individual level, Dubé et al. propose a Bayesian approach that involves choosing a flexible prior over the high dimensional parameter space. In particular, that prior is a mixture of normals. This study illustrates this in further details in the following sections.

[Model Setup]

Let's first define the notations:

$i \in \{1, 2, \ldots, M\}$	=>	the *i*th consumer out of *M* consumers
$t \in \{1, 2, \ldots, n_i\}$	=>	the *t*th observations for the *i*th consumer
$j \in \{1, 2, \ldots, J\}$	=>	the *j*th choice alterative out of *J* choice alternatives
$k \in \{1,2, \ldots, K\}$	=>	the *k*th coefficient out of *K* coefficients
$l \in \{1,2, \ldots, L\}$	=>	the *l*th mixture out of a mixtures of *L* normals

A hierarchical logit model is implemented, with a mixture of L normals as the 1st-stage prior in an attempt to fully capture consumer's heterogeneous preferences. The hierarchical model is especially beneficial when the goal is to assess the joint distribution of a high-dimensional space. The hierarchical model assumes that all parameters are originally independent, but are individually drawn from some prior that is structured by hyperparameters.

We could have chosen probit over logit, since logit can at best be multinomial, while probit could be multivariate, and thus could account for correlation in the error term. Nonetheless, we will follow Dubé et al. and use logit for this study. The model is as follows:

$$y_{ijt} = I[u_{ijt} \geq max(u_{ijt}) \forall j]$$

Where the latent variable – utility – is:

$$u_{ijt} = \alpha_{ij} + P_{jt}\beta_i + \gamma_i I\left[y_{ij(t-1)} = j\right] + \varepsilon_{ijt} \qquad \varepsilon_{ijt} \sim EV\ Type\ I$$

Where terms are:

y_{ijt}	= the *i*th consumer's choice on the *j*th alternative, at time *t*: the consumer only chooses one alternative at a time and only one unit of that alternative. This is slightly simpler than the model presented by Dubé et al., in which only those observations with one *or more* purchases in the product category are included, and in which all the other goods are aggregated as the "outside" goods
α_{ij}	= the intercept term for the *j*th alternative, which is the *i*th consumer's preference on the *j*th alternative, holding price constant, and relative to the *J*th alternative
β_i	= the *i*th consumer's price coefficient
γ_i	= the *i*th consumer's state dependence on the choice made in the last time step. If the choice is the same as that the last time-step, the dependence structure is active. Thus, choice a discrete time Markov chain with a continuous state space. The "present" is defined by {present choice, present utility, present prices} and the one time-step past is defined by {past choice}.

For conciseness and by abusing the notations a bit, we write the entire model in matrix form, for the *t*th observation on the *i*th consumer:

$$u_{it} = \begin{bmatrix} 1 & 0 & 0 & \cdots & P_{i1t} & I[y_{i1(t-1)} = 1] \\ 0 & 1 & 0 & \cdots & P_{i2t} & I[y_{i2(t-1)} = 2] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & P_{i3t} & I[y_{iJ(t-1)} = J] \end{bmatrix} \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \beta_i \\ \gamma_i \end{bmatrix} + \begin{bmatrix} \varepsilon_{i1t} \\ \varepsilon_{i2t} \\ \vdots \\ \vdots \\ \varepsilon_{iJt} \end{bmatrix} = X_{ijt}\beta_i + \varepsilon_{ijt}$$

In this hierarchical model, a mixture of L normals is chosen over a single normal as the 1st-stage prior. The idea behind the change is to allow for flexibility in the distribution from which individual level parameters are drawn. While MCMC allows for subsequent draws to explore the parameter space before eventually converging to an invariant distribution, having a normal prior places a restriction on the degree of exploration, since the relatively thin tails of a normal distribution renders it a strong shrinkage estimator. This is a fine model if the individual level parameters are indeed drawn from a uni-modal distribution.

Given heterogeneity, however, the parameters representing the intercept terms are likely multimodal. If the modals are placed far enough apart and away from the 1st-stage normal prior's mean, the model might never be able to explore this structure and results in a largely uni-modal posterior distribution. This is dangerous, as having a false structure on the high-dimension parameter space may render parameters more or less significant than they might have been otherwise. Bringing the discussion back to consumer inertia: the state dependence coefficient, γ_i, may be more significant than it should have been, if it were to act as a proxy for heterogeneity insufficiently explained by a uni-modal 1st-stage model.

With a mixture of L normals, on the other hand, the high-dimension parameter space has the flexibility to adopt a distribution that is a weighted combination of L normals. The MCMC execution of this model is very similar to what is presented in the lecture notes of MKTG 646, and thus this study will not address it in further details. That said, comparing between a single normal versus a mixture of L normals (Figure1), we see the latter includes a number of additional hyperparamters. One particularly important hyperparameter is L, as L engages in the common tradeoff in statistical modeling between bias and variance, between over- and under-fitting. The example computation in the following section will discuss this topic further. Lastly, we note that we could have also used the full non-parametric Direchlet Process model to let the data itself fully determine the 1st-stage distribution.

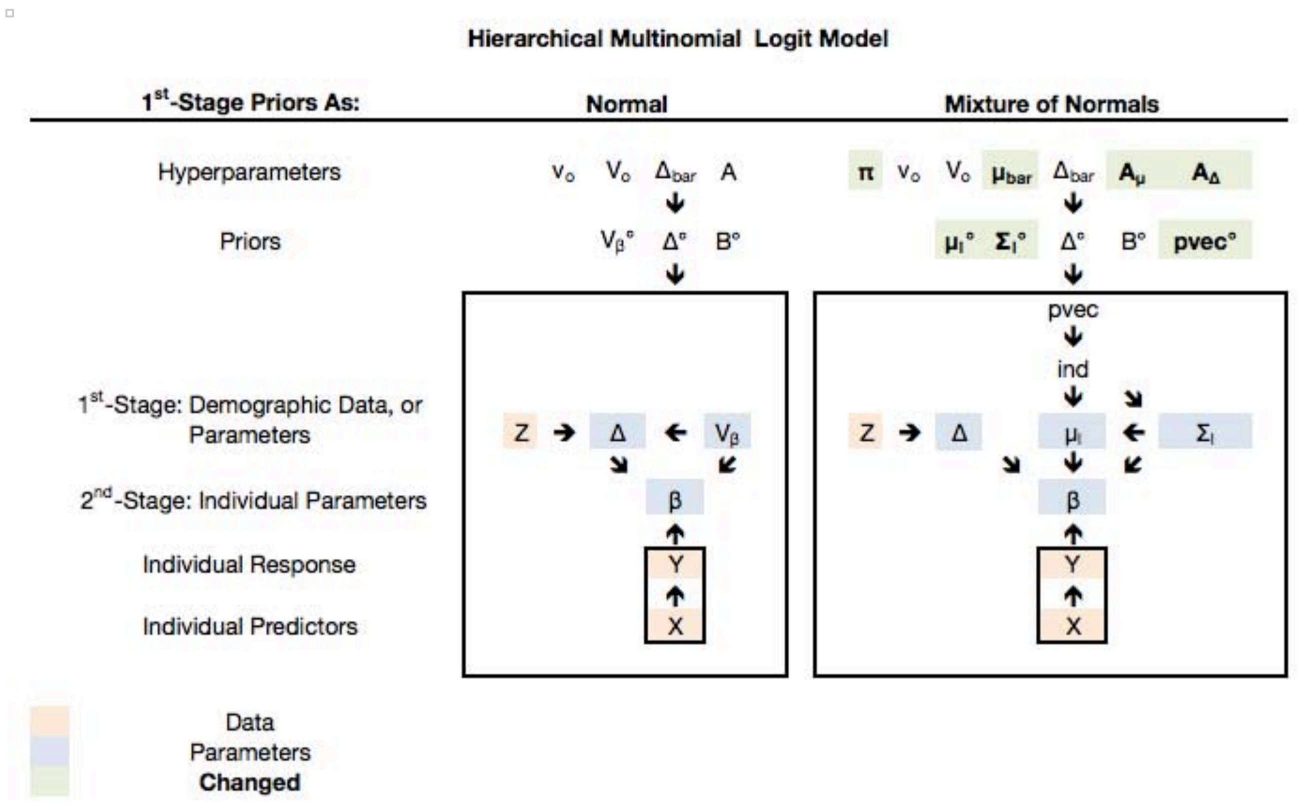

Figure1. Directed graph comparisons between hierarchical multinomial logit models using 1) a normal vs. 2) a mixture of normals as the 1st-stage prior

[Example Study]

As a learning exercise on Dubé et al.'s idea, an analysis is ran on a simulated panel data for which a Bayesian approach to inference may have an advantage. The dataset is on 300 consumers, making choices on 3 alternatives over 10 time periods (Table1). The scenario is meant to represent information sparsity at the individual level.

The parameters used to simulate the panel data aims to bake in heterogeneity by segmenting the consumer set into two main types (Tabe2). Type 2 prefers A to C more than Type 1, who is indifferent between A and C. Both Type 2 and Type 1 dislike B compared to C, but Type 2 dislike B a lot more. The price coefficients for both Type 1 and 2 are set to be negative, implying that A, B and C are all regular goods (i.e. not Giffen or Veblen goods). These coefficients are likely more extreme that what we would normally find in reality for any given product category, but should be prime for illustration purposes.

In simulating individuals' true coefficients, each individual is randomly placed as either Type 1 or Type 2. Then, randomness is added to the coefficients, in such a way that the values in Table2 are the means for normal distributions with standard deviations of 1. Normalized prices are uniformly drawn between (-2, 2) to allow prices to go up and down. Consequently, the utility can be computed, and the purchase is made on the product for which the individual has the maximum utility. To implement state dependence, the current utility on the product that was purchased in the last time step is increased by 1.

Panel Data	**Number**
J (Choice Alternatives: A, B, C)	3
M (Consumers)	300
n_i	10

Table1. Number of simulations

Consumer Type		**Betas**			
Type	Weight	α_A (preference for A, relative to C)	α_B (preference for B, relative to C)	β (Price Coefficient)	γ (State-Dependency)
1	0.5	0	-1	-2	1
2	0.5	3	-8	-6	1
Δ		1	1	1	

Table2. Parameters on the simulated data

The resultant simulated dataset reflects on the aggregate preference on each of the products (Figure 2.). Overall, we see more purchases being made on A than C, and less on B than C. It is notable that each of the products sees some consumers making no purchases on it over the 10 observations. In particular, there are an especially large number of consumers who made no choices on C. As discussed more in the following section, this scenario brings to life our earlier statement on the dichotomy between low probabilities and no possibilities. For instance, just by looking at the data, a naive inference on the consumers who made no choices on C within the 10 observations is that these consumers have infinitely negative preferences towards product C; however, we know through our simulation parameters that this is beyond the realistic range of the coefficients.

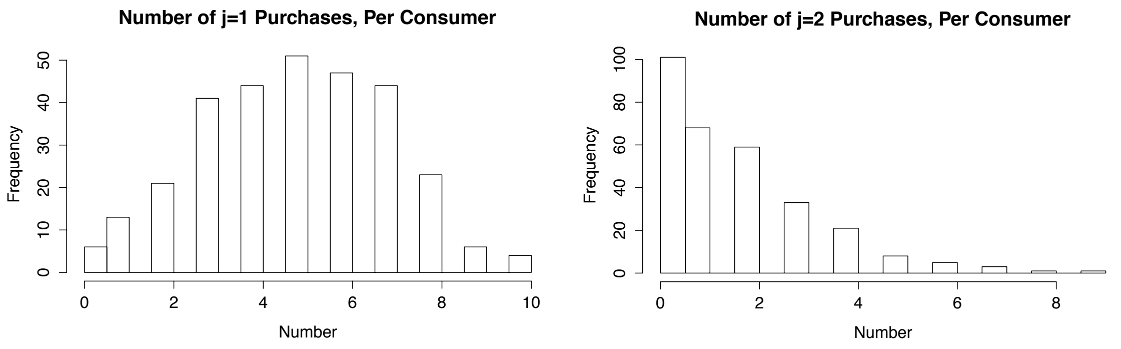

Figure2a. Number of purchases on alternative A (left) and B (right) in the simulated dataset

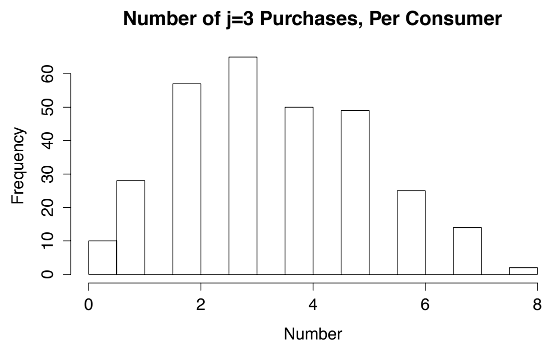

Figure2b. Number of purchases on alternative C in the simulated dataset

[Example Study: Results]

Analysis on the example dataset is performed with four different models (Table4), each with largely the same priors and hyperparameters, besides varying the number of mixtures (Table3). The choice on the four models mimics the choice made by Dubé et al. Each model is performed with 5000 draws, with the last 2500 being used for log-likelihoods computation and parameter estimations. The results herein are performed with the *bayesm* package in R, by Rossi et al (2005); with more time, I would ideally script the code myself! The log-likelihood of the model is computed as follows:

$$loglikelikelihood(Model) \ = \ log(P(\ Data \mid Model)) = log(\prod_i \prod_j \prod_t \frac{exp(X_{ijt}\beta_i)}{\sum_j exp(X_{ijt}\beta_i)})$$

Hyperparameter	Values
π	evenly split
v_o	6
V_o	diag(6)
μ_{bar}	zeros
Δ_{bar}	zeros
A_μ	0.05
A_Δ	0.01

Table3. Model hyperparameters

Model	1st-Stage Priors	γ (State-Dependency)	log-likelihood
1	Normal	No	-794.8
2	Mixture of 5 Normals	No	-716.2
3	Normal	Yes	-803.4
4	Mixture of 5 Normals	Yes	**-660.2**

Table4. Models and their log-likelihoods.

Between the 4 models, the posterior means on each coefficient for the aggregate 300 consumers vary tremendously (Figure3.). In particular, model 4 visually approximates the simulated coefficients the best. In addition, model 1 and 3, remains largely uni-modal, though skewed enough to suggest that uni-modal is not the right distribution. Similar to what Dubé et al. has found for their dataset, the model with a mixture of normals and the state dependency coefficient (model 4) results in the highest log-likelihood. It is notable that MLE (not shown here) could not produce estimates on many consumers' coefficients individually, simply because 10 observations are not sufficient.

This example study has been done with ease, particularly due to our knowledge on the simulation parameters – this certainly would not the case for most real datasets. Drawing from Rossi et al., A common analysis challenge is in choosing appropriate priors, and by extension, appropriate hyperparameters. Setting the priors diffused appears innocuous, since it is making minimal statements on the parameters' scales and locations; however, all priors are informative, as they inevitably contribute to the distributions from which the draws are made. Improper priors can leads to improper posteriors in such a way that preclude the MCMC from reaching an invariant distribution. Priors have particular great influences over the posterior in cases when there is very little data – the priors dominant relative to the likelihood.

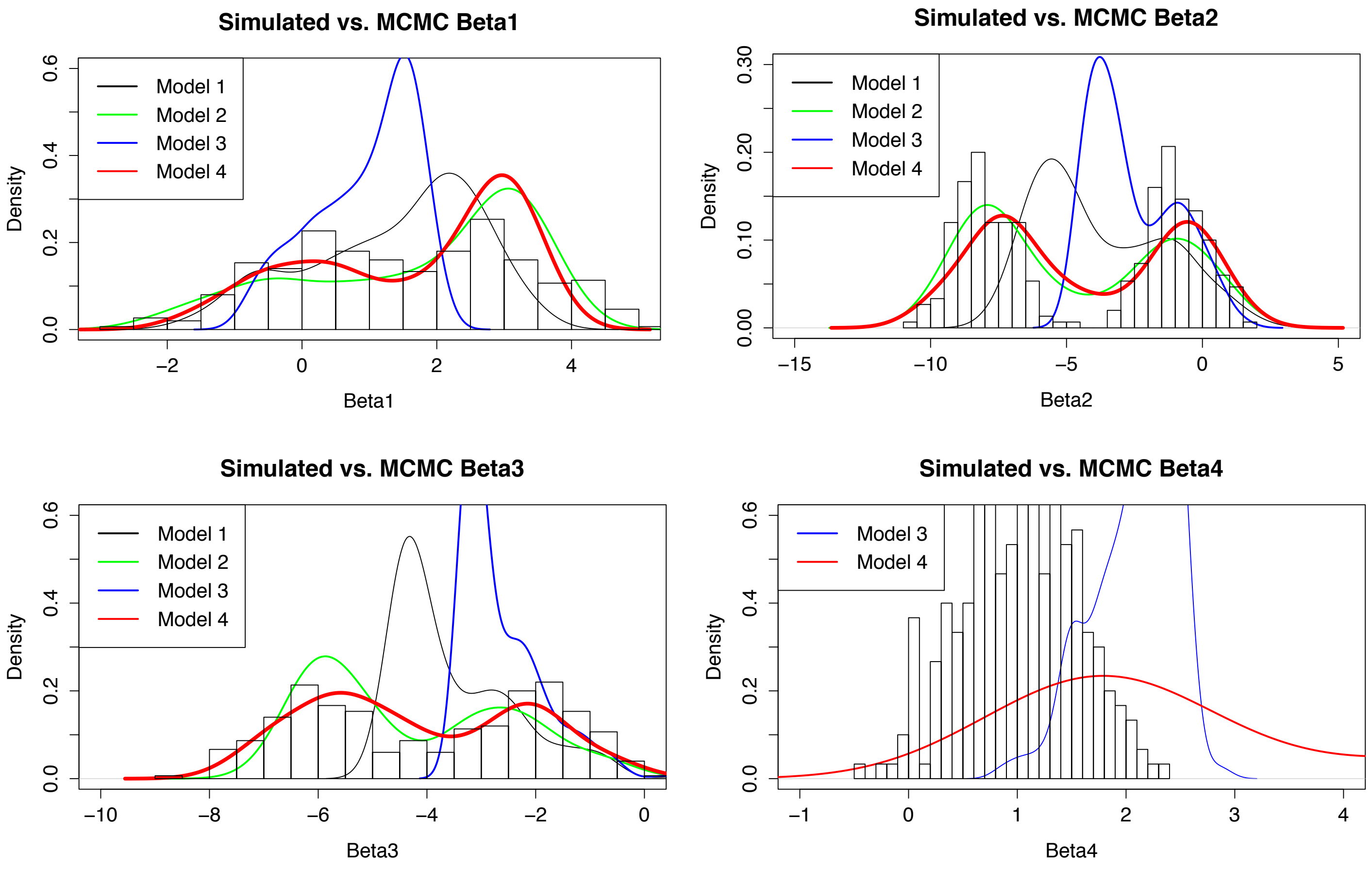

Figure3. Model fits, with simulated data as histograms

From each consumers' standpoint, model 4 also seems to produce less standard error on the estimates for each coefficients, in addition to allowing for more flexible means (Figure4). We expect both results to have contributed to the increase in model 4's log-likelihood. The progression from model $1 \rightarrow 2$ is an improved attribution of heterogeneity, from model $1 \rightarrow 3$ is a decrease in standard error, and from model $1 \rightarrow 4$ is both an improved attribution of heterogeneity and a decrease in standard error. Though the plots show mean ± 1 standard error, we do not expect the individual level parameters to be normally distributed, and Bayesian inference is flexible enough to allow for any distributions.

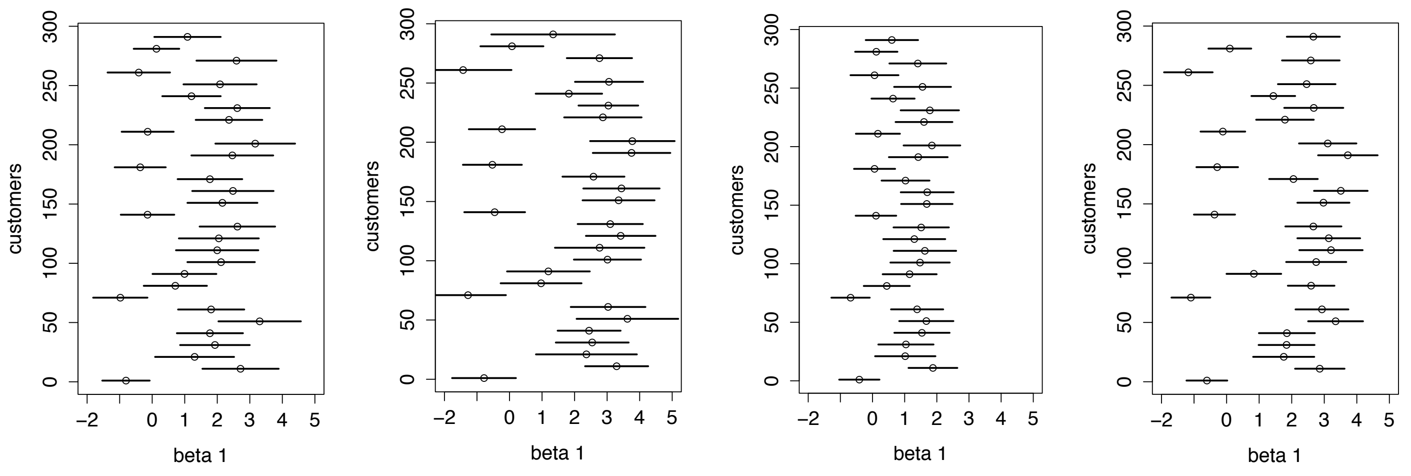

Figure4a. Beta1: intercept term for alterative A, for models 1 to 4 (left to right), showing means ± 1 standard error. Selection shows 30 out of the 300 consumers.

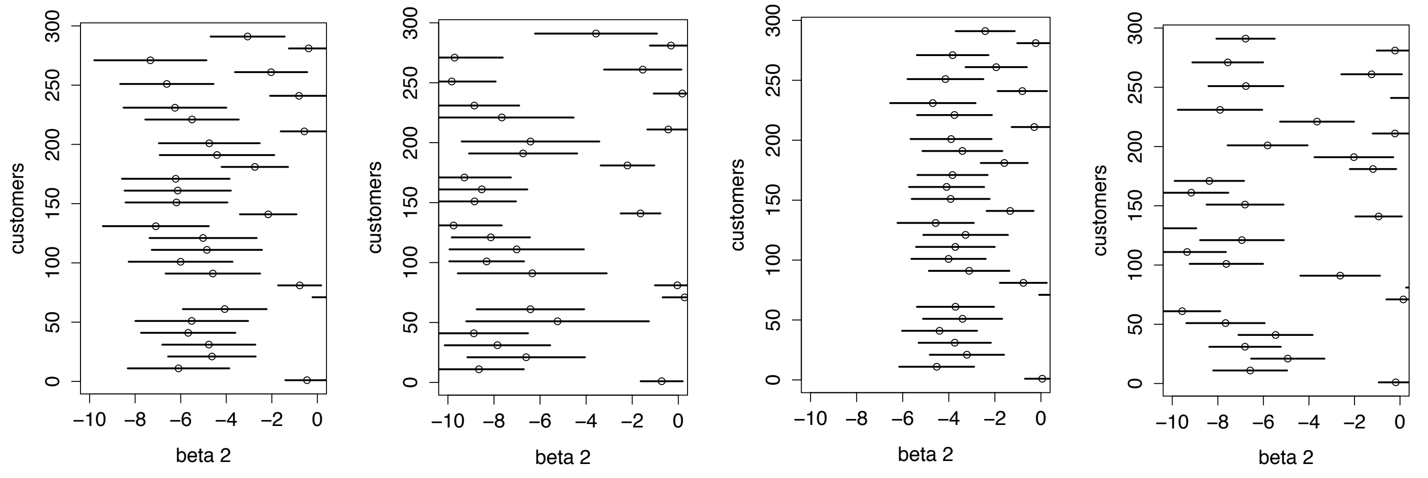

Figure4b. Beta2: intercept term for alterative B, for models 1 to 4 (left to right), showing means ± 1 standard error. Selection shows 30 out of the 300 consumers.

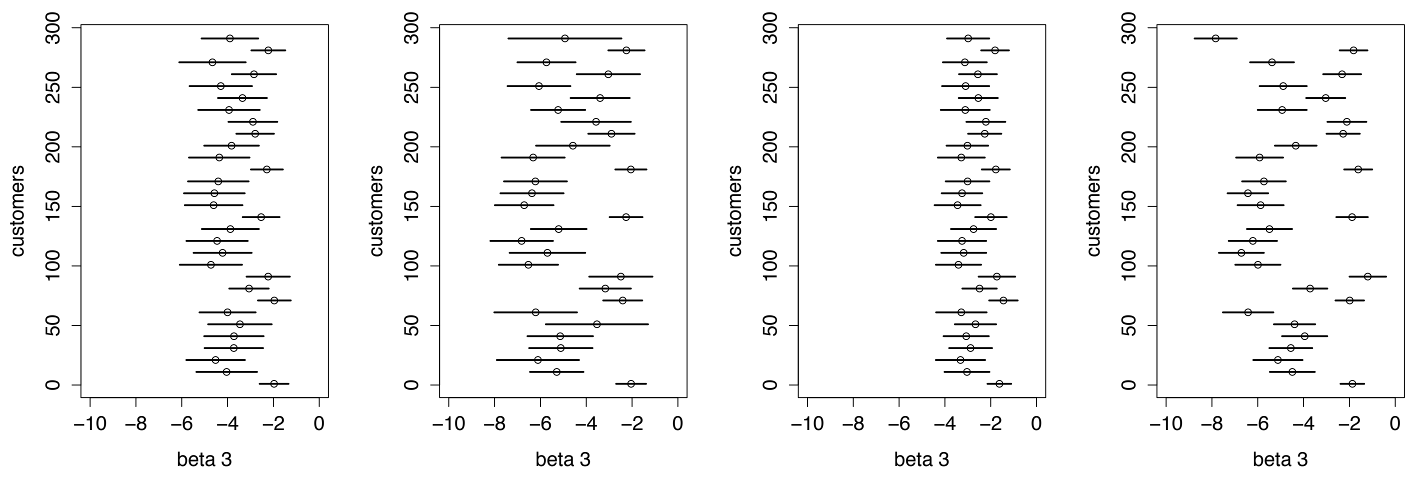

Figure4c. Beta3: price coefficient, for models 1 to 4 (left to right), showing means ± 1 standard error. Selection shows 30 out of the 300 consumers.

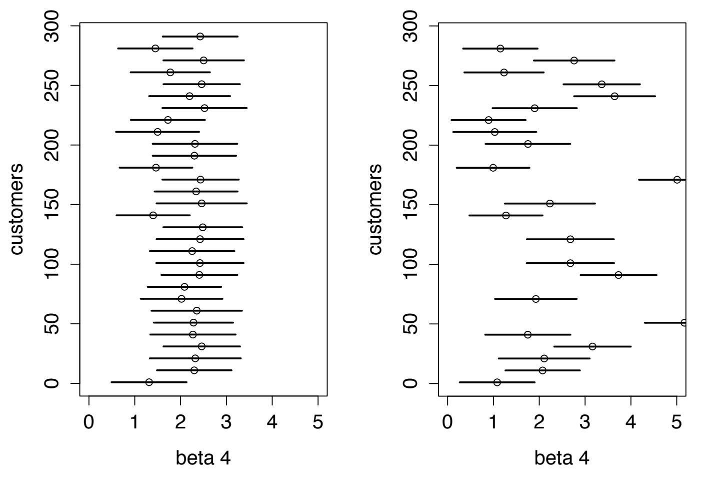

Figure4d. Beta4: state dependence term, for models 3 to 4 (left to right), showing means ± 1 standard error. Selection shows 30 out of the 300 consumers.

With concrete examples, we can bring back the earlier point made on the drawing the line between low probabilities and no possibilities. For instance, by increasing the number of mixtures in our example from L = 5, to L = 20, and to L = 50, we see the posterior distribution on Beta2 drifting off from its expected range (Figure5). Although adding mixtures is essentially computationally free and actually produces higher log-likelihood, we are clearly over-fitting our data with L = 20 and L = 50. The reason behind the drift has to do with the fact that many consumers did not choose product B (whose relative preference is Beta2) over the 10 observations (Figure2). The model with large L can easily over-fits the data, since it is extremely flexible. Having relative preferences drifting off to the very negative values implies that these consumers would never purchase product B, though the implication is likely unsubstantiated with only 10 observations.

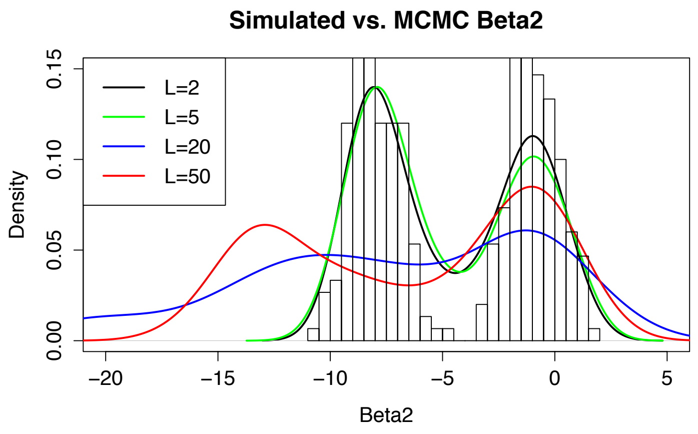

Figure5. Model fits, with increasing number of mixtures

[Conclusion]

This study aimed to discuss some of the advantages of Bayesian inference on datasets that have information sparsity and potential heterogeneity. The particular model used is the hierarchical logit model with a mixture of normals as the 1st-stage prior. The intention of the model is to better account for heterogeneity so that the state dependence coefficient does not stand in as a proxy for unexplained heterogeneity, obscuring the fact if structural state dependence actually exists. Needless-to-day, this study barely scratches the surface of what the model can do, and what Bayesian inference can do in general. The following are some ideas for future explorations:

- incorporate state dependency for more than one past purchases – i.e. widening the present state of the Markov chain
- account for multiple purchases on the same product in a single time step
- account for multiple purchases on different products in the same product category (i.e. what if the consumer purchases both Pepsi and Coke at the same time, but more Pepsi than Coke?)
- incorporate a Hidden Markov model to compute the conditional probability of a consumer making certain choices based on the choice in the previous time step
- separate data into past and present to set up informative priors
- use real data
- remodel with probit instead of logit, and compare accuracies and efficiencies
- remodel with the non-parametric Direchlet Process, to replace the semi-parametric mixture of normals
- script the hierarchical model myself, instead of using the one in the *bayesm* package

[References]

Dubé, J.P., Hitsch, G.J., and Rossi, P.E., "State Dependence and Alternative Explanations for Consumer Inertia." *RAND Journal of Economics*, Vol. 41, No.3, Autumn 2010, pp. 417-445

Rossi, P.E., Allenby, G.M., and McCulloch, R.E., *Bayesian Statistics and Marketing*. New York: John Wiley & Sons, 2005.

Narayanan, Sridhar., *MKTG 646 Lecture Notes*, 2014.

```
################################################################################

# MKTG 646 - Project
# Steven Wong | SUNI: stywong
################################################################################


rm(list=ls())
setwd("~/desktop/Machine_Learning/R")
source("library.r")
source("save_as_eps.r")
setwd("~/desktop/MKTG 646 Project")
options(digits=4)
par(mfrow=c(1,1))
set.seed(646)

################################################################################

# Project: Data Simulation
################################################################################


# [setup] ---------------------------------------------------------------

# consumers
nConsumers = 300
nObvs = 10
ni = rep(nObvs, nConsumers)

# choice alternatives
J = 3

# model coefficients
nCoef = J # no state-dependency
nCoef_state = J+1 # state-dependency

# 1st stage parameters
# identical structure to all of the betas, to let mixture of normals
# dominate the effects in the simulation; (Delta, Z) is random noise
Delta = matrix(rep(1,1*nCoef))
Z = matrix(runif(nConsumers),ncol=1)
Z = t(t(Z)-apply(Z,2,mean))

# number of mixtures
# the number of mixtures can represent the number of types of consumers;
# here we have 2 types of consumers, distributed (50%,50%), and they vary
# in their preferences (to A, B, C) as such, holding price constant:
# - the 1st type is indifferent between A and C
# - the 1st type slightly dislike B, compared to C
# - the 2nd type prefers A to C
# - the 2nd type dislike B strong, compared to C
# - the 2nd type is more sensitive to pricing, than the 1st type
L = 2
mixL = NULL
```

```
mixL[[1]]=list(mu=c(0,-1,-2), rooti=diag(rep(1,3)))
mixL[[2]]=list(mu=c(3,-8,-6), rooti=diag(rep(1,3)))
pvec = c(.5,.5)

#  2nd state parameters
gamma = 1 # 1 time-step state-dependency

# [simulation] ----------------------------------------------------------

# function to simulate data for each consumer
simulateData = function(n, X, beta) {

  # X*beta: without state-dependency
  Xbeta = X %*% beta
  j = nrow(Xbeta)/n
  Xbeta = matrix(Xbeta,byrow=TRUE,ncol=j) # result in matrix form

  # X*beta: with state-dependency = adding utility
  XbetaMaxs = apply(Xbeta,1,max)
  MaxMap = (Xbeta == XbetaMaxs)*matrix(rep(1,n*j),n,j)
  stateDependency = matrix(rep(0,n*j),n,j)
  for (i in 2:n) {
    for (k in 1:J) {
      if ((MaxMap[i-1,k]==1) && (MaxMap[i,k]==1)) {
        stateDependency[i,k] = 1
      }
    }
  }
  Xbeta = Xbeta + gamma*stateDependency

  # Y: each consumer's choice
  probability = exp(Xbeta)
  normalFactor = probability%*%c(rep(1,j))
  probability = probability/as.vector(normalFactor)
  y = vector("double",n)
  for (i in 1:n) {
    yvec = rmultinom(1,1,probability[i,])
    y[i] = c(1:j)%*%yvec
  }

  # state dependency
  X_state = cbind(X,as.vector(t(stateDependency)))

  return(list(y = y, X = X, X_state = X_state, beta = beta, prob =
probability))
}

# simulate data for all consumers
data = NULL
data_state = NULL
for (i in 1:nConsumers) {

  # each consumer's true betas, incorporating error in the draw
  betai = Delta%*%Z[i,] + as.vector(rmixture(1,pvec,mixL)$x)
  betai_state = rbind(betai,rnorm(1,gamma,0.5))
  # generate random +/-2 price unit fluctuations
  Xa = matrix(runif(ni[i]*J, min = -2, max = 2), ncol = J)
  X = createX(J, na = 1, nd = NULL, Xa = Xa, Xd = NULL, base = J)

  # simulate each observation for each consumer
  out = simulateData(ni[i], X, betai)

  data[[i]] = list(y = out$y, X = out$X, beta = betai)
  data_state[[i]] = list(y = out$y, X = out$X_state, beta = betai_state)
}

save(data, Z, data_state, J, nCoef,nCoef_state, nConsumers, nObvs,
file='sim.Rda')
```

```
################################################################################
# MKTG 646 - Project
# Steven Wong | SUNI: stywong
################################################################################


rm(list=ls())
setwd("~/desktop/Machine_Learning/R")
source("library.r")
source("save_as_eps.r")
setwd("~/desktop/MKTG 646 Project")
options(digits=4)
par(mfrow=c(1,1))
set.seed(646)

# load data ---------------------------------------------------
load('sim.Rda')

# Frequentist Approach ----------------------------------------------------

# without state dependence
MLE = list()
for (i in seq(1,nConsumers,10)) {
  X = data[[i]]$X
  y = data[[i]]$y
  MLE[[i]] = optim(c(1,1,1),function(x)
{llmnl(x,y,X)},control=list(fnscale=-1),hessian=T)
}
save(MLE, mu, sd, file='MLE.Rda')

# with state dependence
MLE_S = list()
for (i in seq(1,nConsumers,10)) {
  X = data_state[[i]]$X
  y = data_state[[i]]$y
  MLE_S[[i]] = optim(c(1,1,1,1),function(x)
{llmnl(x,y,X)},control=list(fnscale=-1),hessian=T)
}
save(MLE, mu_s, sd_s, file='MLE_S.Rda')

# Bayesian Approach (Bayesm) ---------------------------------------------------
# there are 4 analyses
# [1] normal 1st-stage prior without state-dependency
# [2] mixture of normals 1st-stage prior without state-dependency
# [3] normal 1st-stage prior with state-dependency
# [4] mixture of normals 1st-stage prior with state-dependency

R = 5000
keep = 1
Mcmc1 = list(R=R,keep=keep)

# hyperparameters
Amu = 1/16
```

```
ncomp = 2

# no state-dependence ---------------------------------------------------
Data1 = list(p=J,lgtdata=data,Z=Z)

# [1]  normal 1st-stage prior
Prior1 = list(ncomp=1, Amu=Amu)
out = rhierMnlRwMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)
save(out, R, keep, file='out1.Rda')

# [2] five mixture of normals 1st-stage prior
Prior1 = list(ncomp=ncomp, Amu=Amu)
out = rhierMnlRwMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)
save(out, R, keep, file='out2.Rda')

# implementing state-dependencne ---------------------------------------------
----
Data1 = list(p=J,lgtdata=data_state)

# [3] normal 1st-stage prior
Prior1 = list(ncomp=1, Amu=Amu)
out = rhierMnlRwMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)
save(out, R, keep, file='out3.Rda')

# [4] five mixture of normals 1st-stage prior
Prior1 = list(ncomp=ncomp, Amu=Amu)
out = rhierMnlRwMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)
save(out, R, keep, file='out4.Rda')
```

```
################################################################################
# MKTG 646 - Project
# Steven Wong | SUNI: stywong
################################################################################

rm(list=ls())
setwd("~/desktop/Machine_Learning/R")
source("library.r")
source("save_as_eps.r")
setwd("~/desktop/MKTG 646 Project")
par(mfrow=c(1,1))
options(digits=4)
set.seed(646)

################################################################################
# Project: Results Plotting
################################################################################

# function to compute the log-likelihood
computellh = function (nConsumers,D,betas) {
  llh = rep(0,nConsumers)
  for(i in 1:nConsumers) {
    llh[i] = llmnl(betas[i,],D[[i]]$y,D[[i]]$X)
  }
  return(sum(llh))
}

################################################################################
# simulated data
################################################################################

load('sim.Rda') # data= list(y, X, beta)
breaks = 20

# betas -----------------------------------------------------
betas_sim = matrix(0, nConsumers, nCoef_state)
colnames(betas_sim) = c("Beta1","Beta2","Beta3","Beta4")
for(i in 1:nConsumers) {
  betas_sim[i,] = data_state[[i]]$beta
}

# # plots
# hist(betas_sim[,1],breaks=breaks,main="Histogram of Simulated
Beta1",xlab="Beta1")
# save_as_eps("betas1_sim.eps",5,8)
# hist(betas_sim[,2],breaks=breaks,main="Histogram of Simulated
Beta2",xlab="Beta2")
# save_as_eps("betas2_sim.eps",5,8)
# hist(betas_sim[,3],breaks=breaks,main="Histogram of Simulated
Beta3",xlab="Beta3")
# save_as_eps("betas3_sim.eps",5,8)
# hist(betas_sim[,4],breaks=breaks,main="Histogram of Simulated
Beta4",xlab="Beta4")
# save_as_eps("betas4_sim.eps",5,8)

# y ------------------------------------------------------
y_sim = matrix(0, nConsumers, nObvs)
y_num = matrix(0, nConsumers, J)
for(i in 1:nConsumers) {
  y_sim[i,] = data_state[[i]]$y
  y_num[i,] = cbind(sum(y_sim[i,]==1),sum(y_sim[i,]==2),sum(y_sim[i,]==3))
}

# # plots
# hist(y_num[,1],breaks=breaks,main="Number of j=1 Purchases, Per
Consumer",xlab="Number")
# save_as_eps("y1_sim.eps",5,8)
# hist(y_num[,2],breaks=breaks,main="Number of j=2 Purchases, Per
Consumer",xlab="Number")
# save_as_eps("y2_sim.eps",5,8)
# hist(y_num[,3],breaks=breaks,main="Number of j=3 Purchases, Per
Consumer",xlab="Number")
# save_as_eps("y3_sim.eps",5,8)

# with state dependency
# computellh(nConsumers,data_state,betas_sim)


################################################################################

# analysis output
################################################################################

# there are 4 analyses
# [1] normal 1st-stage prior without state-dependency
# [2] mixture of normals 1st-stage prior without state-dependency
# [3] normal 1st-stage prior with state-dependency
# [4] mixture of normals 1st-stage prior with state-dependency

burnin = 2500
all_mu_betas = array(0,dim=c(nConsumers,J+1,4))

# [1] ---------------------------------------------------
setwd("~/desktop/MKTG 646 Project/m=1")
load('out1.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
sd_betas = apply(betas,c(1,2),sd)
computellh(nConsumers,data,mu_betas)
all_mu_betas[,1:3,1] = mu_betas

# draws
# plot(seq(1,R,keep),out$betadraw[10,1,],type='l')
```

```
xlim = list(c(-2,5),c(-10,0),c(-10,0),c(0,5))

mu_betas_1plusSd = mu_betas + sd_betas
mu_betas_1minusSd = mu_betas - sd_betas

# every 10 customers
customers = seq(1,300,10)
for (i in 1:3) {
  plot(mu_betas[customers,i], customers,xlim=xlim[[i]],xlab=sprintf("beta
%s",i))
  segments(mu_betas_1minusSd[customers,i],customers,
           mu_betas_1plusSd[customers,i],customers, lwd=2)
  save_as_eps(sprintf("betas_1_%s.eps",i),6,4)
}

# [2] ---------------------------------------------------
# setwd("~/desktop/MKTG 646 Project/m=2")
setwd("~/desktop/MKTG 646 Project/m=5")
# setwd("~/desktop/MKTG 646 Project/m=20&a=0.01")
# setwd("~/desktop/MKTG 646 Project/m=50&a=0.01")
load('out2.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
sd_betas = apply(betas,c(1,2),sd)
computellh(nConsumers,data,mu_betas)
all_mu_betas[,1:3,2] = mu_betas

# draws
# plot(seq(1,R,keep),out$betadraw[10,1,],type='l')

mu_betas_1plusSd = mu_betas + sd_betas
mu_betas_1minusSd = mu_betas - sd_betas

# every 10 customers
customers = seq(1,300,10)
for (i in 1:3) {
  plot(mu_betas[customers,i], customers,xlim=xlim[[i]],xlab=sprintf("beta
%s",i))
  segments(mu_betas_1minusSd[customers,i],customers,
           mu_betas_1plusSd[customers,i],customers, lwd=2)
  save_as_eps(sprintf("betas_2_%s.eps",i),6,4)
}

# do something with GAMMA!!!

# [3] ---------------------------------------------------
setwd("~/desktop/MKTG 646 Project/m=1")
load('out3.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
sd_betas = apply(betas,c(1,2),sd)
computellh(nConsumers,data_state,mu_betas)
all_mu_betas[,,3] = mu_betas
```

```
# draws
# plot(seq(1,R,keep),out$betadraw[25,1,],type='l')

mu_betas_1plusSd = mu_betas + sd_betas
mu_betas_1minusSd = mu_betas - sd_betas

# every 10 customers
customers = seq(1,300,10)
for (i in 1:4) {
  plot(mu_betas[customers,i], customers,xlim=xlim[[i]],xlab=sprintf("beta
%s",i))
  segments(mu_betas_1minusSd[customers,i],customers,
           mu_betas_1plusSd[customers,i],customers, lwd=2)
  save_as_eps(sprintf("betas_3_%s.eps",i),6,4)
}

# [4] ------------------------------------------------
# setwd("~/desktop/MKTG 646 Project/m=2")
setwd("~/desktop/MKTG 646 Project/m=5")
# setwd("~/desktop/MKTG 646 Project/m=20&a=0.01")
# setwd("~/desktop/MKTG 646 Project/m=50&a=0.01")
load('out4.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
computellh(nConsumers,data_state,mu_betas)
all_mu_betas[,,4] = mu_betas

# draws
# plot(seq(1,R,keep),out$betadraw[250,4,],type='l')

mu_betas_1plusSd = mu_betas + sd_betas
mu_betas_1minusSd = mu_betas - sd_betas

# every 10 customers
customers = seq(1,300,10)
for (i in 1:4) {
  plot(mu_betas[customers,i], customers,xlim=xlim[[i]],xlab=sprintf("beta
%s",i))
  segments(mu_betas_1minusSd[customers,i],customers,
           mu_betas_1plusSd[customers,i],customers, lwd=2)
  save_as_eps(sprintf("betas_4_%s.eps",i),6,4)
}

# everything together  --------------------------------------------------------
----
setwd("~/desktop/MKTG 646 Project")
breaks = 20

plot(density(all_mu_betas[,1,1]), lwd=1, col="black",
     main="Simulated vs. MCMC Beta1",xlab="Beta1", xlim=c(-3,5), ylim=c(0,0.6))
lines(density(all_mu_betas[,1,2]), lwd=2, col="green")
lines(density(all_mu_betas[,1,3]), lwd=2, col="blue")
lines(density(all_mu_betas[,1,4]), lwd=4, col="red")
```

```
legend("topleft",c("Model 1","Model 2","Model 3","Model 4"),
       col=c("black","green","blue","red"),lwd=2)
# save_as_eps("betas1_1.eps",5,8)
hist(betas_sim[,1],prob=T,breaks=breaks,add=T)
# save_as_eps("betas1_2.eps",5,8)

plot(density(all_mu_betas[,2,1]), lwd=1, col="black",
     main="Simulated vs. MCMC Beta2",xlab="Beta2", xlim=c(-15,5),
ylim=c(0,0.3))
lines(density(all_mu_betas[,2,2]), lwd=2, col="green")
lines(density(all_mu_betas[,2,3]), lwd=2, col="blue")
lines(density(all_mu_betas[,2,4]), lwd=4, col="red")
legend("topleft",c("Model 1","Model 2","Model 3","Model 4"),
       col=c("black","green","blue","red"),lwd=2)
# save_as_eps("betas2_1.eps",5,8)
hist(betas_sim[,2],prob=T,breaks=breaks,add=T)
# save_as_eps("betas2_2.eps",5,8)

plot(density(all_mu_betas[,3,1]), lwd=1, col="black",
     main="Simulated vs. MCMC Beta3",xlab="Beta3", xlim=c(-10,0),
ylim=c(0,0.6))
lines(density(all_mu_betas[,3,2]), lwd=2, col="green")
lines(density(all_mu_betas[,3,3]), lwd=2, col="blue")
lines(density(all_mu_betas[,3,4]), lwd=4, col="red")
legend("topleft",c("Model 1","Model 2","Model 3","Model 4"),
       col=c("black","green","blue","red"),lwd=2)
# save_as_eps("betas3_1.eps",5,8)
hist(betas_sim[,3],prob=T,breaks=breaks,add=T)
# save_as_eps("betas3_2.eps",5,8)

plot(density(all_mu_betas[,4,3]), lwd=1, col="blue",
     main="Simulated vs. MCMC Beta4",xlab="Beta4", xlim=c(-1,4), ylim=c(0,0.6))
lines(density(all_mu_betas[,4,4]), lwd=2, col="red")
legend("topleft",c("Model 3","Model 4"),
       col=c("blue","red"),lwd=2)
# save_as_eps("betas4_1.eps",5,8)
hist(betas_sim[,4],prob=T,breaks=breaks,add=T)
# save_as_eps("betas4_2.eps",5,8)

# [2] & [4] ----------------------------------------------
all_mu_betas0 = array(0,dim=c(nConsumers,J,4))
all_mu_betas1 = array(0,dim=c(nConsumers,J,4))
all_mu_betas2 = array(0,dim=c(nConsumers,J,4))
all_mu_betas3 = array(0,dim=c(nConsumers,J,4))

setwd("~/desktop/MKTG 646 Project/m=2")
load('out2.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
computellh(nConsumers,data,mu_betas)
all_mu_betas0[,,2] = mu_betas

setwd("~/desktop/MKTG 646 Project/m=5")
load('out2.Rda')
```

```
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
computellh(nConsumers,data,mu_betas)
all_mu_betas1[,,2] = mu_betas

setwd("~/desktop/MKTG 646 Project/m=20&a=0.01")
load('out2.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
computellh(nConsumers,data,mu_betas)
all_mu_betas2[,,2] = mu_betas

setwd("~/desktop/MKTG 646 Project/m=50&a=0.01")
load('out2.Rda')
start = burnin/keep
betas = out$betadraw[,,start:R/keep]
mu_betas = apply(betas,c(1,2),mean)
computellh(nConsumers,data,mu_betas)
all_mu_betas3[,,2] = mu_betas

plot(density(all_mu_betas0[,2,2]), lwd=2, col="black",
     main="Simulated vs. MCMC Beta2",xlab="Beta2", xlim=c(-20,5),
ylim=c(0,0.15))
lines(density(all_mu_betas1[,2,2]), lwd=2, col="green")
lines(density(all_mu_betas2[,2,2]), lwd=2, col="blue")
lines(density(all_mu_betas3[,2,2]), lwd=2, col="red")
legend("topleft",c("L=2","L=5","L=20","L=50"),
       col=c("black","green","blue","red"),lwd=2)
save_as_eps("compare_all_Ls_1.eps",5,8)
hist(betas_sim[,2],prob=T,breaks=breaks,add=T)
save_as_eps("compare_all_Ls_2.eps",5,8)

plot(seq(1,R,keep),out$betadraw[1,1,],type='l')
cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Normal Mixture Distribution",fill=TRUE)
summary(out$nmix)
plot(seq(1,R,keep),out$betadraw[1,1,],type='l')
acf(out$betadraw[1,1,])
```