Bae's Theorem

Michael Betancourt

December 2024

Table of contents

In 2006 Netflix announced the infamous Netflix Prize. The competition challenged anyone to use a data set of customer movie reviews to inform predictions for a second, held-out set of customer movie reviews. Superficially the Netflix challenge was a mixed success, although from a broader perspective it demonstrated the perils of poorly-chosen metrics for predictive performance and the subtleties of data privacy. Wikipedia summarizes the history well.

Beyond the Netflix Prize itself the corresponding data set is a nice example of some of the problems that can arise in a wide range of practical applications. For instance not only are customer preferences limited to five star ratings but also the interpretation of those ratings

are ambiguous and typically not consistent across customers. Some customers are generous with their five star ratings while some are meager with not only their five star ratings but also their four star and sometimes even three star ratings.

In addition to the idiosyncratic rating scales any analysis of this data also has to contend with idiosyncratic customer preferences. Because not every customer will agree on the quality of a given movie we have to decide whether we want to try to learn an aggregate preference across the entire population or the individual customer preferences.

In this chapter I develop a Bayesian analysis of a subset of the Netflix training data set, not in an attempt to win the Netflix Prize decades too late but rather to demonstrate some strategies for approaching these analysis challenges.

Importantly this analysis will not be the first time that Netflix has been associated with Bayesian inference. In 2016 Amy Hogan (@alittlestats) presented her influential Bae's Theorem,

$$p(\text{chill} \mid \text{Netflix}) = \frac{p(\text{Netflix} \mid \text{chill})\, p(\text{chill})}{p(\text{Netflix})}.$$

1 Setup

As always we begin by setting up our local `R` environment.

```
par(family="serif", las=1, bty="l",
    cex.axis=1, cex.lab=1, cex.main=1,
    xaxs="i", yaxs="i", mar = c(5, 5, 3, 5))
```

```
library(rstan)
rstan_options(auto_write = TRUE)            # Cache compiled Stan programs
options(mc.cores = parallel::detectCores()) # Parallelize chains
parallel:::setDefaultClusterOptions(setup_strategy = "sequential")
```

```
util <- new.env()
source('mcmc_analysis_tools_rstan.R', local=util)
source('mcmc_visualization_tools.R', local=util)
```

2 Data Exploration

The full Netflix Prize training data set consisted of 100,480,507 customer-movie pairs, each accompanied by an ordinal rating between one and five "stars", with one being the worst

rating and five being the best. The observed ratings spanned 480,189 anonymized customers and 17,770 movies.

To allow for a more manageable demonstration I reduced the full data set by considering only the first 1000 movies and then randomly subsampling 100 customers and 200 movies with probabilities proportional to the total number of ratings. This left 2,415 total ratings.

Finally to facilitate model implementations, and add another layer of anonymizing obfuscating, I relabeled the selected customers and movies with contiguous indices.

```
data <- read_rdump('data/ratings.data.R')

cat(sprintf("%s Customers", data$N_customers))
```

```
100 Customers
```

```
cat(sprintf("%s Movies", data$N_movies))
```

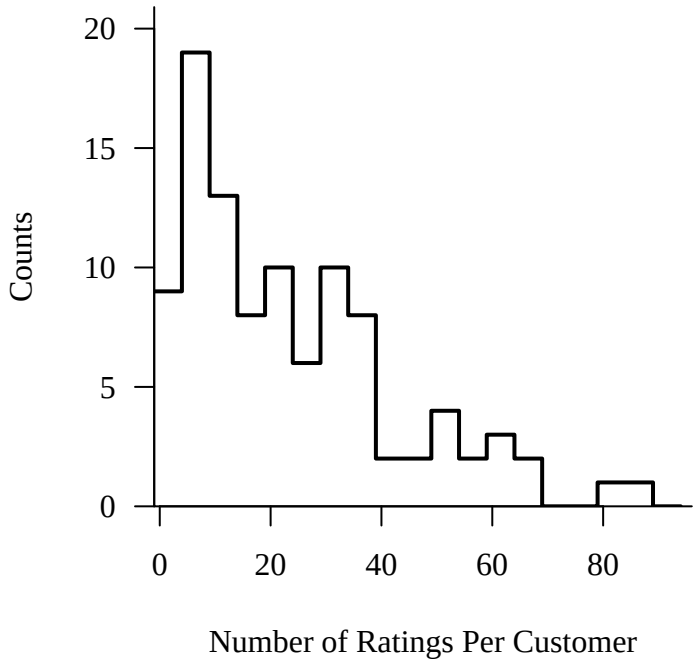
```
200 Movies
```

```
cat(sprintf("%s Total Ratings", data$N_ratings))
```

```
2415 Total Ratings
```

Despite the data subsampling favoring customers with more ratings, most of the selected customers rated only a few movies. Overall the training data set is relatively sparse, with only a few customers contributing most of the ratings.

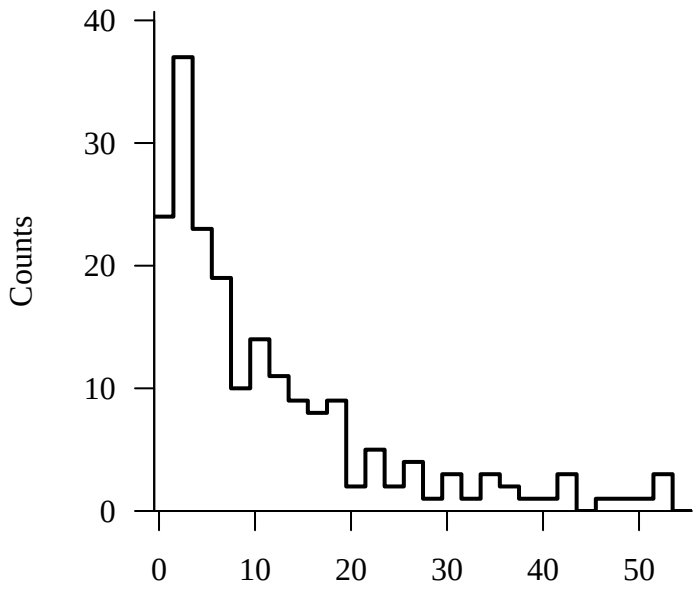
```
par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(table(data$customer_idxs),
                    -0.5, 95.5, 5,
                    xlab="Number of Ratings Per Customer")
```



Similarly most of the selected movies have only a few ratings.

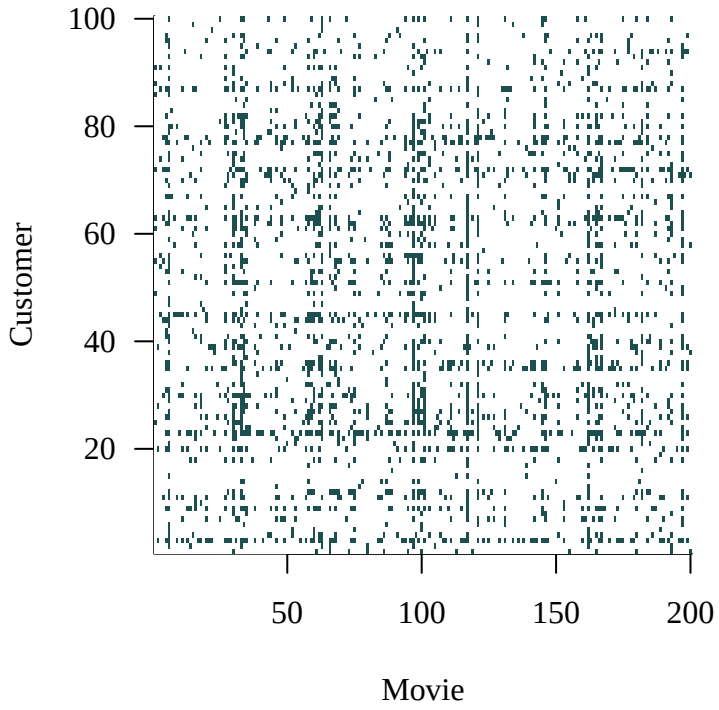
```
par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(table(data$movie_idxs),
                    -0.5, 55.5, 2,
                    xlab="Number of Ratings Per Movie")
```

```
Warning in check_bin_containment(bin_min, bin_max, values): 2 values (1.0%)
fell above the binning.
```



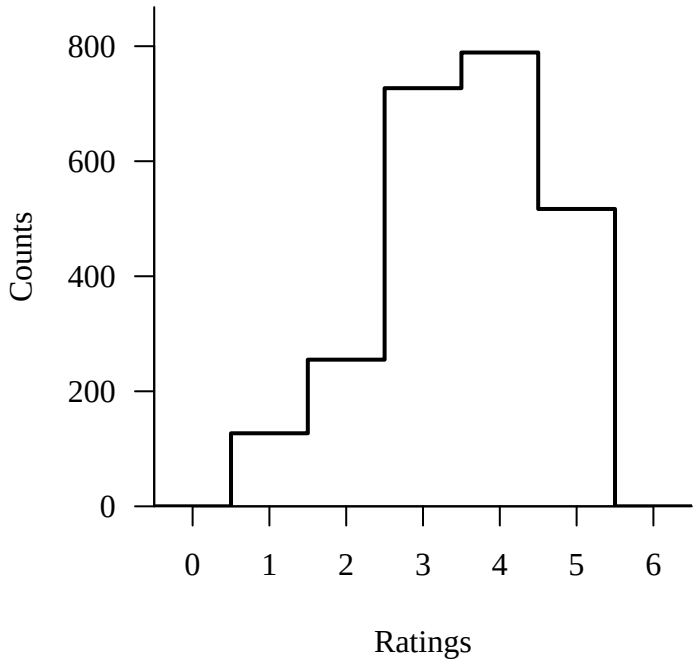
This sparsity is even more evident if we visualize the customer-movie pairings.

```
xs <- seq(1, data$N_movies, 1)
ys <- seq(1, data$N_customers, 1)
zs <- matrix(0, nrow=data$N_movies, ncol=data$N_customers)

for (n in 1:data$N_ratings) {
  zs[data$movie_idxs[n], data$customer_idxs[n]] <- 1
}

par(mfrow=c(1, 1), mar = c(5, 5, 1, 1))

image(xs, ys, zs, col=c("white", util$c_dark_teal),
      xlab="Movie", ylab="Customer")
```



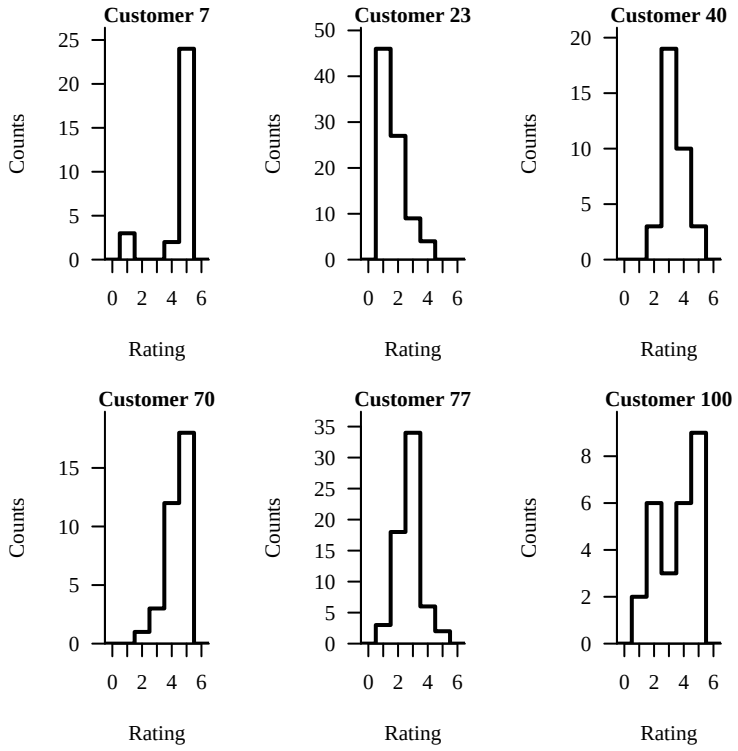
The observed ratings are slightly biased towards large values, with far more four star ratings than two star ratings. From the data alone, however, we cannot determine whether or not this is because most movies that had been rated were relatively good or because most customers were just generous with their ratings.

```
par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(data$ratings,
                    -0.5, 6.5, 1, xlab="Ratings")
```



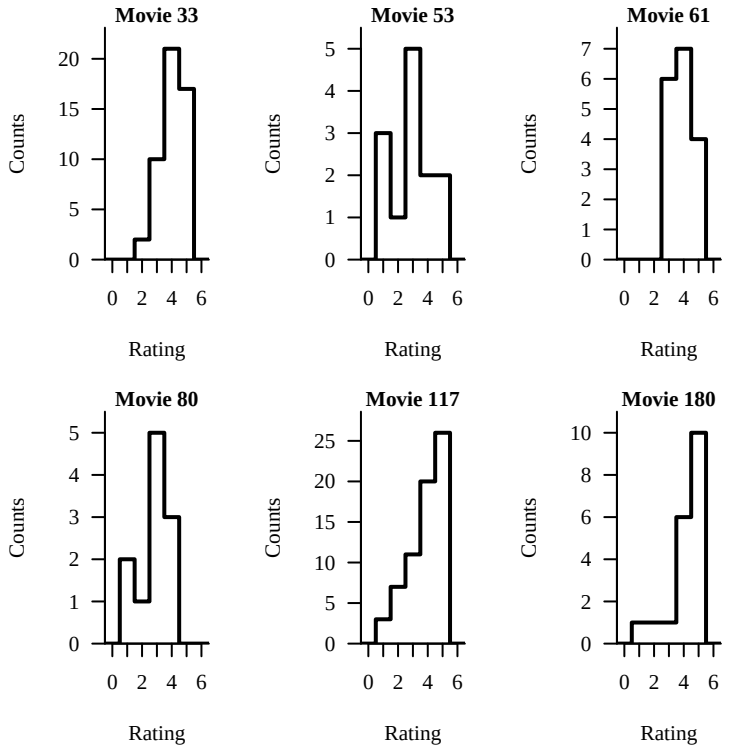
That said there is substantial heterogeneity in the rating behavior across customers. Customer 70, for example, gave many high ratings while Customer 23 gave many low ratings.

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (c in c(7, 23, 40, 70, 77, 100)) {
  util$plot_line_hist(data$ratings[data$customer_idxs == c],
                      -0.5, 6.5, 1,
                      xlab="Rating", main=paste('Customer', c))
}
```



Similarly we see strong variation in the observed ratings across movies. Movies 117 and 180, for example, are particularly well-reviewed.

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (m in c(33, 53, 61, 80, 117, 180)) {
  util$plot_line_hist(data$ratings[data$movie_idxs == m],
                      -0.5, 6.5, 1,
                      xlab="Rating", main=paste('Movie', m))
}
```



When critiquing any model of these ratings we want to be able to interrogate this variation in rating behavior across customers and movies. Even with the subsampled data, however, visualizing a histogram of ratings for each customer and movie would be far too ungainly. A more scalable, albeit less informative, approach is to compute a scalar summary of the ratings within each group, and then construct a histogram of those **stratified** summaries.

The only problem here is identifying useful summary statistics. Because ordinal spaces are not equipped with a distinguished metric empirical moments are ill-defined; in particular given any order-preserving map

$$f : \{1, 2, 3, 4, 5\} \to \mathbb{R}$$

we can construct a *distinct* empirical mean,

$$\hat{mu}_f = \frac{1}{N} \sum_{n=1}^{N} f(r_n),$$

and corresponding higher-order empirical moments.

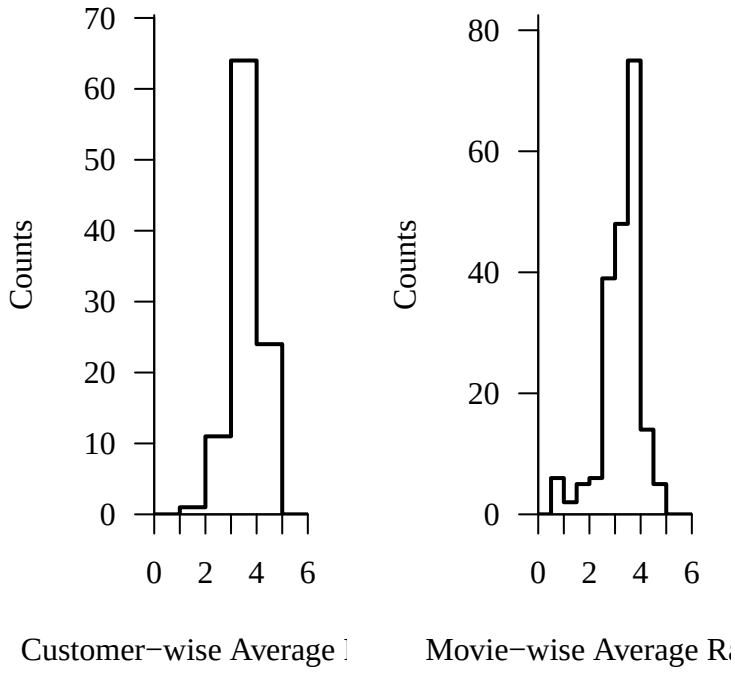
That said, most of these empirical moments share similar qualitative behaviors. If a stratified histogram is peaked towards central values, for example, then most empirical means will end up somewhere near that peak. Even if empirical moments capture different behaviors, however, they can still be useful for comparing observed and posterior predictive behaviors.

For this case study I'll assume a metric with equal unit distances between neighboring ordinal elements, with the resulting empirical mean

$$\hat{mu}_f = \frac{1}{N} \sum_{n=1}^{N} r_n.$$

We can then stratify this mean by customers and movies and histogram the resulting ensemble of summaries.

```
par(mfrow=c(1, 2), mar=c(5, 5, 2, 1))

mean_rating_customer <-
  sapply(1:data$N_customers,
         function(c) mean(data$ratings[data$customer_idxs == c]))
util$plot_line_hist(mean_rating_customer,
                    0, 6, 1,
                    xlab="Customer-wise Average Ratings")


mean_rating_movie <-
  sapply(1:data$N_movies,
         function(m) mean(data$ratings[data$movie_idxs == m]))
util$plot_line_hist(mean_rating_movie,
                    0, 6, 0.5,
                    xlab="Movie-wise Average Ratings")
```



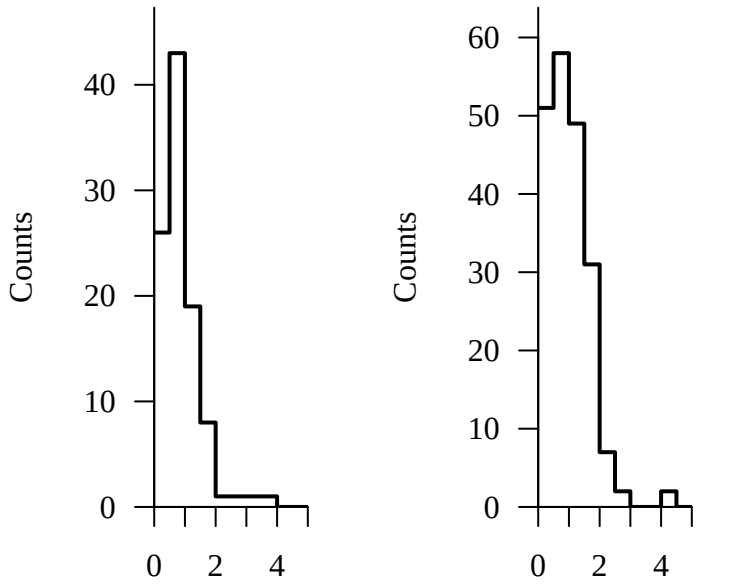
Of course an empirical mean captures only some of the rating behavior within each strata. We can capture more with a summary that is sensitive to the dispersion of ratings in each strata, such as the empirical variance or empirical entropy. Note one nice advantage of the empirical entropy relative to the empirical variance is that it does not require a choice of metric over the ordinal values.

Here let's go with the empirical variance based on the same assumptions as our empirical mean, or rather a modified empirical variance that defaults to zero when a strata consists of only one value.

```
safe_var <- function(vals) {
  if (length(vals) == 1)
    (0)
  else
    (var(vals))
}
```

```
par(mfrow=c(1, 2), mar=c(5, 5, 2, 1))

var_rating_customer <-
  sapply(1:data$N_customers,
         function(c) safe_var(data$ratings[data$customer_idxs == c]))
util$plot_line_hist(var_rating_customer,
                    0, 5, 0.5,
```

```
                    xlab="Customer-wise Rating Variances")

var_rating_movie <-
  sapply(1:data$N_movies,
         function(m) safe_var(data$ratings[data$movie_idxs == m]))
util$plot_line_hist(var_rating_movie,
                    0, 5, 0.5,
                    xlab="Movie-wise Rating Variances")
```



Customer−wise Rating Va Movie−wise Rating Vari

The main limitation with these stratified summary statistics is that they are sensitive to only the marginal variation across movies and customers. In other words they are sensitive to heterogeneity across movies or customers but not heterogeneity across movies and customers *at the same time*. Unfortunately because each customer-movie pair has at most one rating, and most have no ratings, we can't just stratify summary statistics by both customer and movie.

One potential compromise is to construct empirical covariances for each pair of customers or movies. For example given our assumed metric the empirical covariance between two movies m_1 and m_2 is defined by

$$\hat{\rho}_{m_1 m_2} = \hat{\rho}_{m_2 m_1} = \frac{1}{N_C - 1} \sum_{c=1}^{N_C} (r_{cm_1} - \hat{\mu}_{m_1}) \cdot (r_{cm_2} - \hat{\mu}_{m_2})$$

where N_C is the number of customers, r_{cm} is the rating given to movie m by customer c, and

$$\hat{\mu}_m = \frac{1}{N_C} \sum_{c=1}^{N_C} r_{cm}.$$

The immediate issue is that, because not every customer rates every movie, many of the r_{cm} in these sums will be undefined. All we can do here is limit the sums to the customers who have rated both movies.

More formally let c_m denote the set of customers who have rated movie m and $N_{C,m}$ denote the number of elements in that set. Similarly let $\mathsf{c}_{m_1 m_2}$ denote the set of customers who have rated *both* movies m_1 and m_2 with $N_{C,m_1 m_2}$ the number of elements in that set. Then we can define

$$\hat{\rho}_{m_1 m_2} = \hat{\rho}_{m_2 m_1} = \frac{1}{N_{C,m_1 m_2} - 1} \sum_{c \in \mathsf{c}_{m_1 m_2}} (r_{cm_1} - \hat{\mu}_{m_1}) \cdot (r_{cm_2} - \hat{\mu}_{m_2})$$

with

$$\hat{\mu}_m = \frac{1}{N_{C,m}} \sum_{c \in \mathsf{c}_m} r_{cm}$$

the movie-wise empirical means that we've already constructed.

All of this said given the relative sparsity of the observed ratings the set $\mathsf{c}_{m_1 m_2}$ will be empty for most pairs of movies. Even fewer pairs of movies will have the $N_{C,m_1 m_2} > 1$ needed for $\hat{\rho}_{m_1 m_2}$ to be well-defined, let alone $N_{C,m_1 m_2}$ large enough for $\hat{\rho}_{m_1 m_2}$ to provide an informative summary.

We can avoid any ill-defined or poorly informative empirical covariances by including only those movie pairs with $N_{C,m_1 m_2}$ sufficiently large enough in the final histogram. This also has the added benefit of reducing the total number of summaries that we have to bin into the final histogram visualization. Here I will require $N_{C,m_1 m_2} > 7$.

Now that we've carefully laid out the math the implementation is relatively straightforward. First we loop over the observed ratings twice, incrementing the partial sums for each pair of movies when appropriate. This gives us

$$\hat{\Sigma}_{m_1 m_2} = \sum_{c \in \mathsf{c}_{m_1 m_2}} (r_{cm_1} - \hat{\mu}_{m_1}) \cdot (r_{cm_2} - \hat{\mu}_{m_2})$$

and

$$N_{C,m_1 m_2} = \sum_{c \in \mathsf{c}_{m_1 m_2}} 1.$$

```
covar_rating_movie <- matrix(0,
                             nrow=data$N_movies,
                             ncol=data$N_movies)
movie_pair_counts <- matrix(0,
                            nrow=data$N_movies,
                            ncol=data$N_movies)

for (n1 in 1:data$N_ratings) {
  for (n2 in 1:data$N_ratings) {
    if (data$customer_idxs[n1] == data$customer_idxs[n2]) {
      m1 <- data$movie_idxs[n1]
      m2 <- data$movie_idxs[n2]
      y <- (data$ratings[n1] - mean_rating_movie[m1]) *
           (data$ratings[n2] - mean_rating_movie[m2])
      covar_rating_movie[m1, m2] <- covar_rating_movie[m1, m2] + y
      covar_rating_movie[m2, m1] <- covar_rating_movie[m2, m1] + y
      movie_pair_counts[m1, m2] <- movie_pair_counts[m1, m2] + 1
      movie_pair_counts[m2, m1] <- movie_pair_counts[m2, m1] + 1
    }
  }
}
```

Next we compute

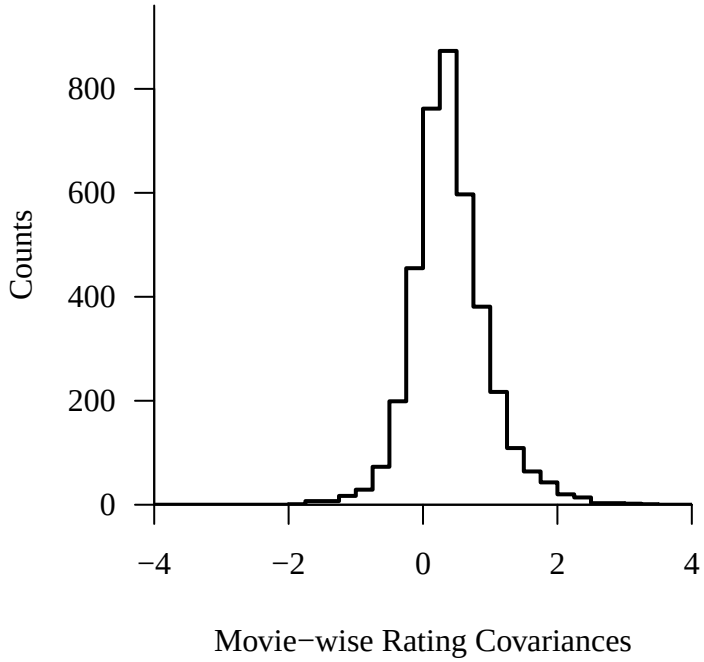
$$\hat{\rho}_{m_1 m_2} = \frac{\hat{\Sigma}_{m_1 m_2}}{N_{C,m_1 m_2} - 1}$$

for each pair of movies where $N_{C,m_1 m_2}$ is larger than 7.

```
m_pairs <- list()
covar_rating_movie_filt <- c()

for (m1 in 2:data$N_movies) {
  for (m2 in 1:(m1 - 1)) {
    if (movie_pair_counts[m1, m2] > 7) {
      m_pairs[[length(m_pairs) + 1]] <- c(m1, m2)
      covar_rating_movie_filt <- c(covar_rating_movie_filt,
                                   covar_rating_movie[m1, m2] /
                                     (movie_pair_counts[m1, m2] - 1))
    }
  }
}
```

Finally we bin these values into a histogram that visualizes the range of these partial empirical covariance behaviors.

```
par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(covar_rating_movie_filt,
                    -4, 4, 0.25,
                    xlab="Movie-wise Rating Covariances")
```



In order to construct posterior retrodictive checks later on we will need to select the posterior predictive values for these same selected movie pairs. We might as well construct the appropriate variable names now and have them ready.

```
covar_rating_movie_filt_names <-
  sapply(m_pairs,
         function(p) paste0('covar_rating_movie_pred[',
                            p[1], ',', p[2], ']'))
```

All of this said I think that there is still a lot of opportunity for better summary statistics in applications like these, such as summaries that are more compatible with the structure of an ordinal space and don't require the assumption of an arbitrary metric and summaries that better capture couplings between different strata.

3 Homogeneous Customer Model

Now that we've familiarized ourselves with the data we can make our first attempt at modeling the data generating process that, well, generated it. Our models will be built on a foundation of ordinal pairwise comparison modeling techniques.

Given a latent logistic probability density function we will use cut points to derive baseline ordinal probabilities for each star rating. This baseline will not be tied to any particular movie but rather a hypothetical default movie implied by the configuration of an induced Dirichlet prior model.

Next we will assume that movies systematically shift these baseline probabilities to lower or higher ratings depending on their quality. This is implemented with affinity parameters for each movie that shift the latent logistic probability density function, and hence the derived ordinal probabilities, based on customer preference. Initially we will assume that the rating behavior is homogeneous across customers so that we need only a single set of cut points to model all of the data.

Lastly assuming that our domain expertise about consumer preferences is exchangeable we will model the movie affinity parameters hierarchically. To avoid degeneracy in the customer-movie comparisons we will need to anchor the population location of this hierarchical model to zero, the same anchor location used in the induced Dirichlet prior model. Given the sparsity of observed ratings we'll implement this hierarchical model with a monolithic non-centered parameterization.

```
fit <- stan(file="stan_programs/model1.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Despite the initial model complexity there are no diagnostic issues indicating suspect computation.

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('gamma_ncp',
                                         'tau_gamma',
```

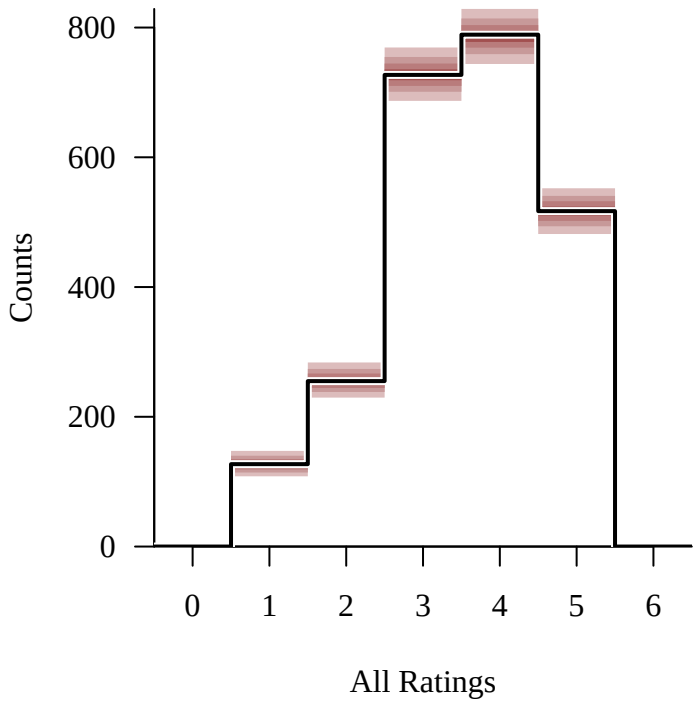
```
                                              'cut_points'),
                                            check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

Consequently we're ready to investigate this model's retrodictive performance.

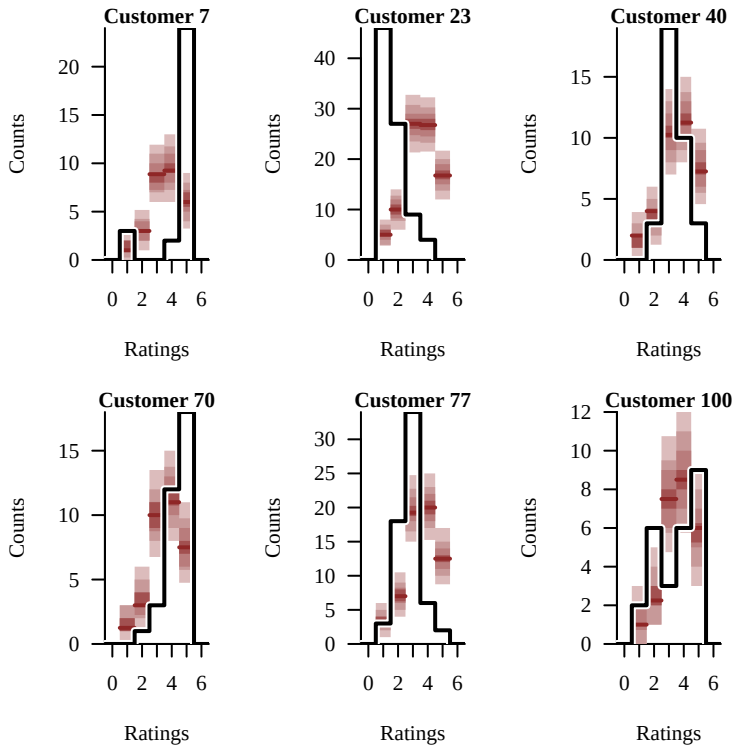
The model appears to be flexible enough to capture the behavior of the aggregate ratings.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples1, 'rating_pred', -0.5, 6.5, 1,
                         baseline_values=data$ratings,
                         xlab="All Ratings")
```



On the other hand the retrodictive performance is much worse if we look at individual customer behaviors. In particular there is much more heterogeneity in the observed data than what the model can reproduce, which isn't surprising given that we explicitly assumed homogeneous customer behavior.

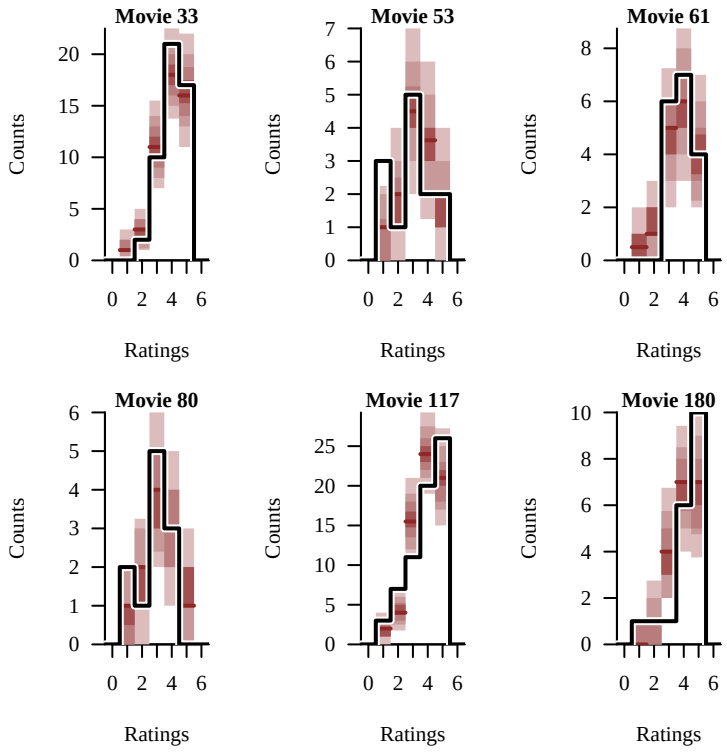
```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (c in c(7, 23, 40, 70, 77, 100)) {
  names <- sapply(which(data$customer_idxs == c),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples1, names)

  customer_ratings <- data$ratings[data$customer_idxs == c]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=customer_ratings,
                           xlab="Ratings",
                           main=paste('Customer', c))
}
```



On the other hand the model doesn't seem to have a problem capturing the heterogeneity in the observed ratings stratified across movies, at least for this quick spot check.

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (m in c(33, 53, 61, 80, 117, 180)) {
```

```
  names <- sapply(which(data$movie_idxs == m),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples1, names)

  movie_ratings <- data$ratings[data$movie_idxs == m]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=movie_ratings,
                           xlab="Ratings",
                           main=paste('Movie', m))
}
```



To better investigate the variation in retrodictive performance, however, we need to look beyond just a few customers and movies. We can examine the behavior of all customers and movies at the same time with histograms of the stratified summary statistics we discussed above.

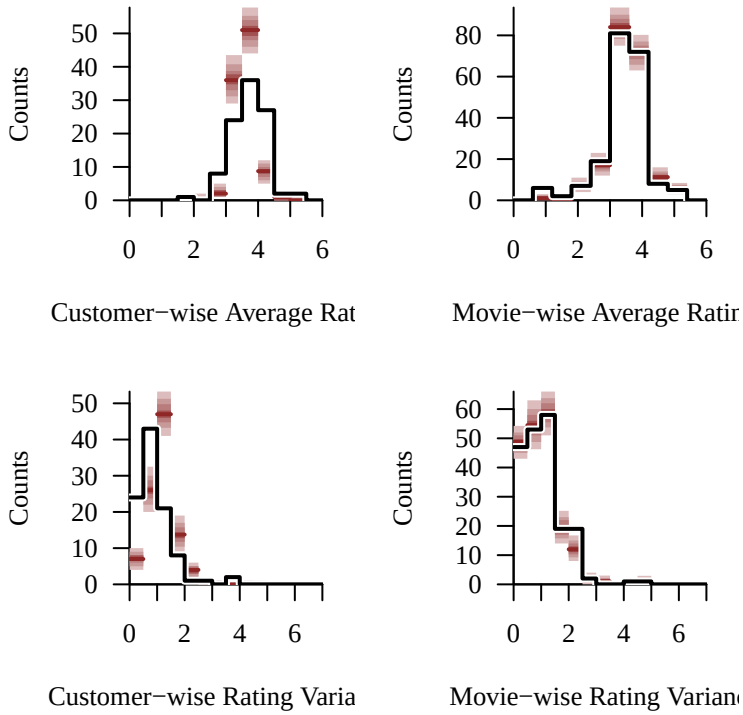
Here we see that the posterior predictive behavior of the customer-wise empirical means are more narrowly distributed than what we see in the observed data. At the same time the posterior predictive customer-wise empirical variances concentrate at larger values than the observed customer-wise empirical variances.

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples1, 'mean_rating_customer_pred',
                         0, 6, 0.5,
                         baseline_values=mean_rating_customer,
                         xlab="Customer-wise Average Ratings")

util$plot_hist_quantiles(samples1, 'mean_rating_movie_pred',
                         0, 6, 0.6,
                         baseline_values=mean_rating_movie,
                         xlab="Movie-wise Average Ratings")

util$plot_hist_quantiles(samples1, 'var_rating_customer_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_customer,
                         xlab="Customer-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 239 predictive values (0.1%) fell above the binning.
```

```
util$plot_hist_quantiles(samples1, 'var_rating_movie_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_movie,
                         xlab="Movie-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 1287 predictive values (0.2%) fell above the binning.
```

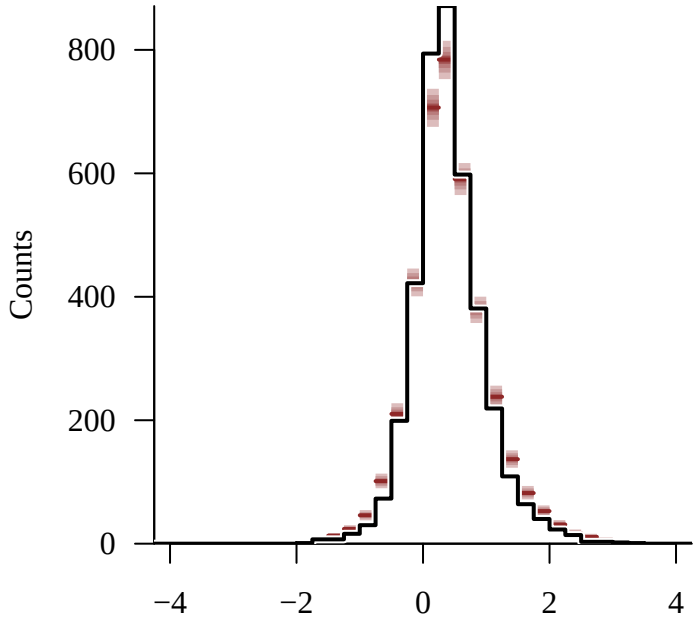


Finally the collection of selected movie empirical covariances appears to be more heavy-tailed in the observed data relative to the posterior predictions.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

filtered_samples <-
  util$filter_expectands(samples1,
                         covar_rating_movie_filt_names)

util$plot_hist_quantiles(filtered_samples, 'covar_rating_movie_pred',
                         -4.25, 4.25, 0.25,
                         baseline_values=covar_rating_movie_filt,
                         xlab="Filtered Movie-wise Rating Covariances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 273 predictive values (0.0%) fell below the binning.
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2329 predictive values (0.0%) fell above the binning.
```



4 Independent, Heterogeneous Customer Model

All of our retrodictive checks tell a consistent story – customers do not rate movies the same way as each other. Fortunately it's straightforward to model each customer's idiosyncratic behavior by allowing them with their own set of cut points, and hence baseline rating probabilities.

```
fit <- stan(file="stan_programs/model2.stan",
            data=data, seed=8438330,
            warmup=1000, iter=2024, refresh=0)
```

Frustratingly, the computation suffers from a few stray divergences.

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

```
  Chain 4: 2 of 1024 transitions (0.2%) diverged.

  Divergent Hamiltonian transitions result from unstable numerical
trajectories.  These instabilities are often due to degenerate target
geometry, especially "pinches".  If there are only a small number of
```

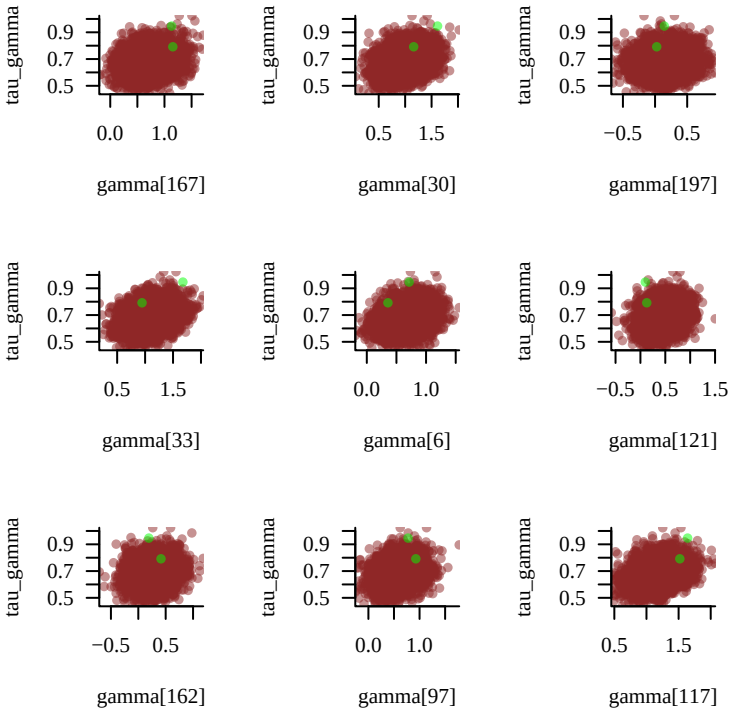
```
divergences then running with adept_delta larger than 0.801 may reduce
the instabilities at the cost of more expensive Hamiltonian
transitions.
```

```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('gamma_ncp',
                                         'tau_gamma',
                                         'cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

One possibility is that the replicated cut points are messing with the hierarchical geometry of the movie affinities. That said the movie affinities most susceptible to degenerate behavior in a non-centered parameterization are those with the most ratings, and those do not exhibit any obvious geometric pathologies.

```
idxs <- as.numeric(names(tail(sort(table(data$movie_idxs)), 9)))
names <- sapply(idxs, function(m) paste0('gamma[', m, ']'))
util$plot_div_pairs(names, 'tau_gamma', samples2, diagnostics)
```



At this point we could spend some time investigating for any degenerate behavior between the cut points that could be causing problems, or even between some cut points and some movie affinities. Given the small number of divergences, however, let's just run again with a less aggressive step size adaptation and cope with the increased computational cost. We can always come back to this investigation later if this ends up being our final model.

```
fit <- stan(file="stan_programs/model2.stan",
            data=data, seed=8438330,
            warmup=1000, iter=2024, refresh=0,
            control=list('adapt_delta'=0.9))
```

Fortunately that seems to have done the trick and now our diagnostics are squeaky clean.

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

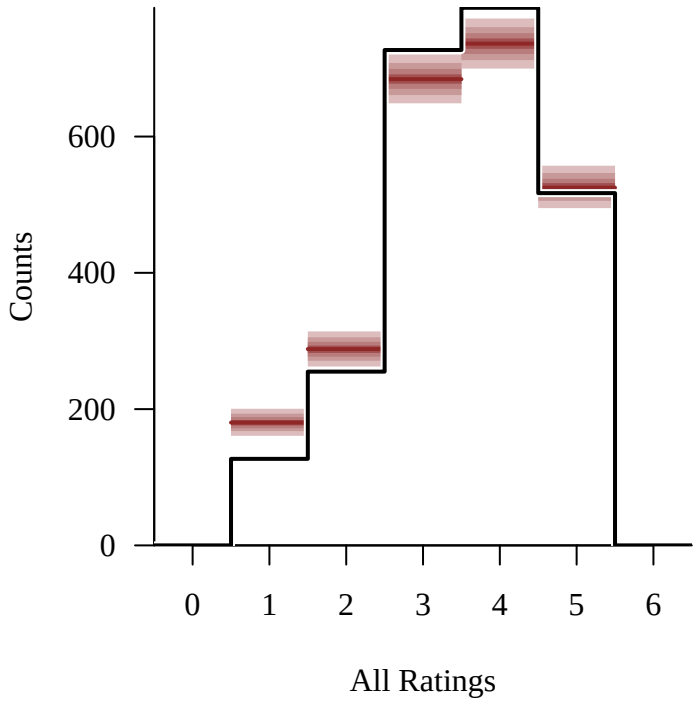
```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('gamma_ncp',
                                         'tau_gamma',
                                         'cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

Has our retrodictive performance improved?

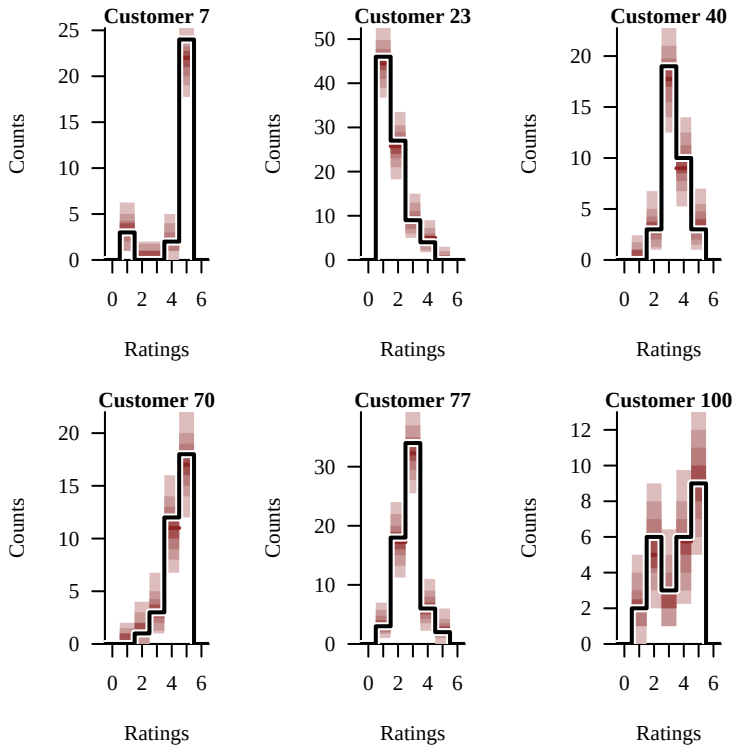
Interestingly the behavior of the aggregate ratings isn't quite as consistent between the observed data and posterior predictions as it was before. That said the increased retrodictive tension isn't necessarily large enough to be a concern yet.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples2, 'rating_pred', -0.5, 6.5, 1,
                         baseline_values=data$ratings,
                         xlab="All Ratings")
```



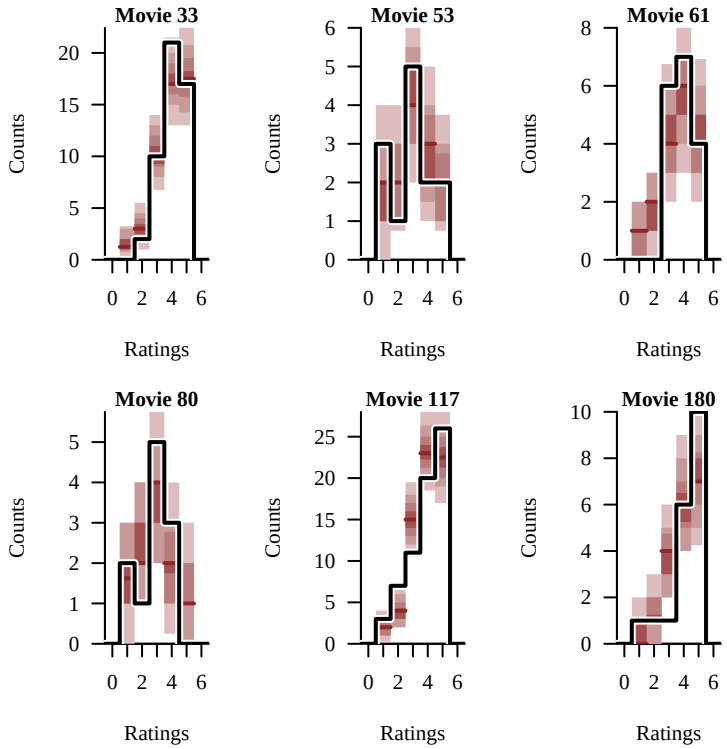
More importantly the retrodictive performance for the customers that we've spot checked is much better.

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (c in c(7, 23, 40, 70, 77, 100)) {
  names <- sapply(which(data$customer_idxs == c),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples2, names)

  customer_ratings <- data$ratings[data$customer_idxs == c]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=customer_ratings,
                           xlab="Ratings",
                           main=paste('Customer', c))
}
```



This improvement in the customer-wise behaviors hasn't come at any cost to the movie-wise retrodictive performance, at least for this limited spot check.

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (m in c(33, 53, 61, 80, 117, 180)) {
  names <- sapply(which(data$movie_idxs == m),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples2, names)

  movie_ratings <- data$ratings[data$movie_idxs == m]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=movie_ratings,
                           xlab="Ratings",
                           main=paste('Movie', m))
}
```



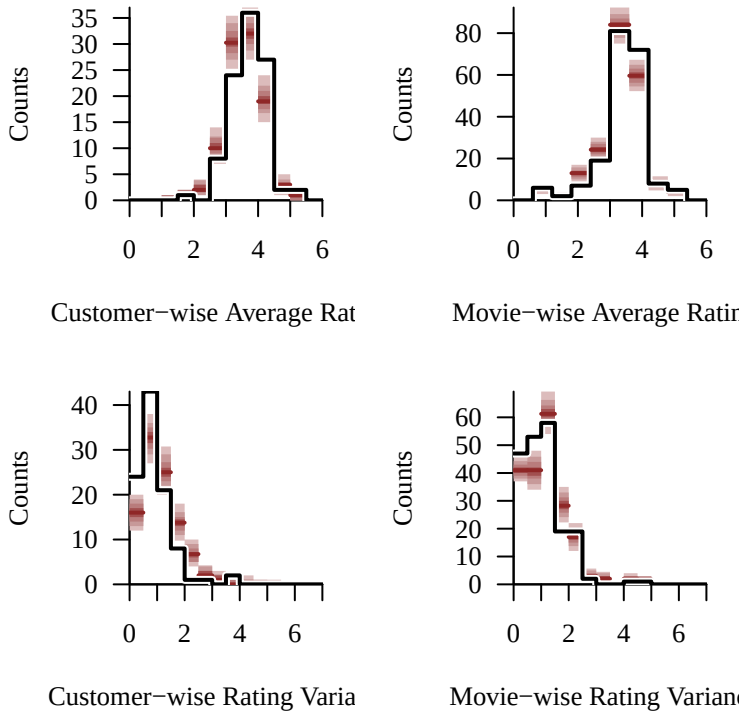
Do the histograms of stratified summary statistics tell a similar story? The retrodictive tension in the customer-wise empirical means and empirical variances has decreased, although some remains in the empirical variances.

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples2, 'mean_rating_customer_pred',
                         0, 6, 0.5,
                         baseline_values=mean_rating_customer,
                         xlab="Customer-wise Average Ratings")

util$plot_hist_quantiles(samples2, 'mean_rating_movie_pred',
                         0, 6, 0.6,
                         baseline_values=mean_rating_movie,
                         xlab="Movie-wise Average Ratings")

util$plot_hist_quantiles(samples2, 'var_rating_customer_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_customer,
                         xlab="Customer-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 400 predictive values (0.1%) fell above the binning.
```

```
util$plot_hist_quantiles(samples2, 'var_rating_movie_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_movie,
                         xlab="Movie-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2589 predictive values (0.3%) fell above the binning.
```

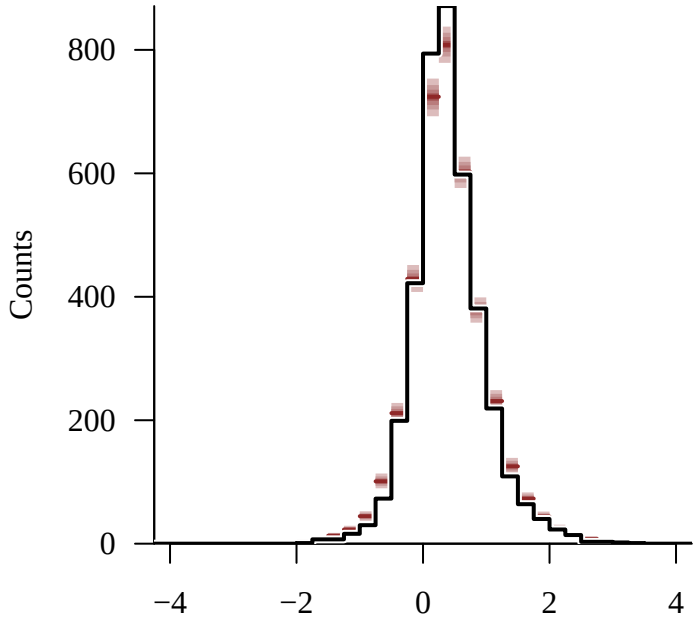


Moreover the retrodictive tension in the collection of selected movie empirical covariances remains.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

filtered_samples <-
  util$filter_expectands(samples2,
                         covar_rating_movie_filt_names)

util$plot_hist_quantiles(filtered_samples, 'covar_rating_movie_pred',
                         -4.25, 4.25, 0.25,
                         baseline_values=covar_rating_movie_filt,
                         xlab="Filtered Movie-wise Rating Covariances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 370 predictive values (0.0%) fell below the binning.
```

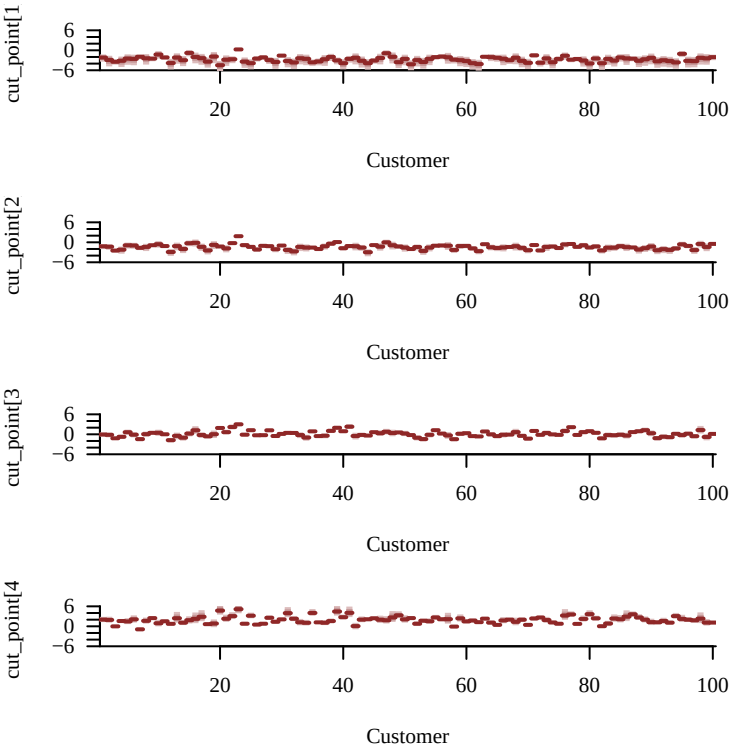
```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 248 predictive values (0.0%) fell above the binning.
```



Filtered Movie−wise Rating Covariances

If we were content with these mild retrodictive tensions then we would move on to exploring the posterior inferences themselves. For example we could visualize the marginal posterior inferences for the interior cut points of each customer.

```
par(mfrow=c(4, 1), mar=c(5, 5, 1, 1))

for (k in 1:4) {
  names <- sapply(1:data$N_customers,
                  function(r) paste0('cut_points[', r, ',', k, ']'))
  util$plot_disc_pushforward_quantiles(samples2, names,
                                       xlab="Customer",
                                       display_ylim=c(-6, 6),
                                       ylab=paste0('cut_point[', k, ']'))
}
```



The interior cut points are a bit more interpretable when visualized together, especially when comparing different customers. For example to compare Customer 23 and Customer 70 we might overlay the marginal posterior visualizations of the four interior cut points of each customer in adjacent plots.

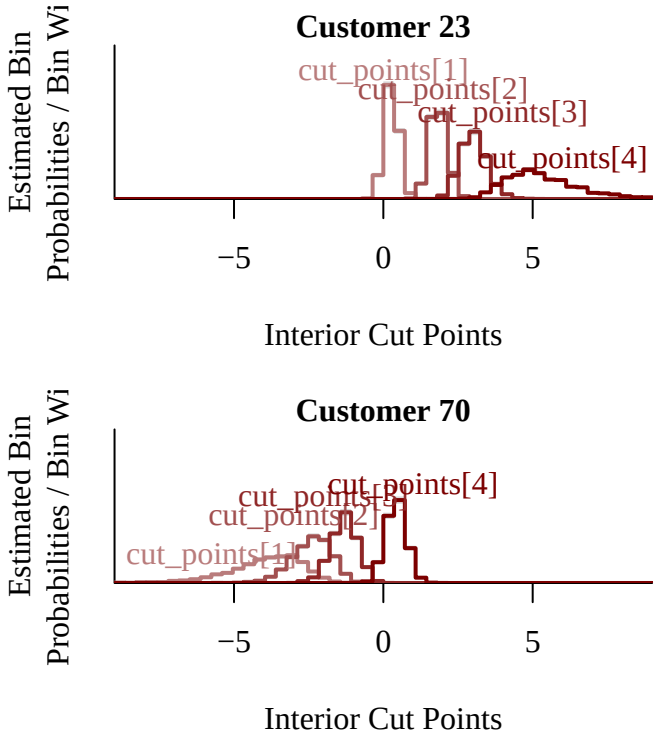
```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

cols <- c(util$c_mid, util$c_mid_highlight,
          util$c_dark, util$c_dark_highlight)

c <- 23

k <- 1
name <-paste0('cut_points[', c, ',', k, ']')
util$plot_expectand_pushforward(samples2[[name]],
                                50, flim=c(-9, 9), ylim=c(0, 2),
                                col=cols[k],
                                display_name='Interior Cut Points',
                                main=paste('Customer', c))
```

```
for (k in 2:4) {
  name <-paste0('cut_points[', c, ',', k, ']')
  util$plot_expectand_pushforward(samples2[[name]],
                                  50, flim=c(-9, 9),
                                  col=cols[k], border="#BBBBBB88",
                                  add=TRUE)
}

text(0, 1.65, "cut_points[1]", col=util$c_mid)
text(2, 1.4, "cut_points[2]", col=util$c_mid_highlight)
text(4, 1.1, "cut_points[3]", col=util$c_dark)
text(6, 0.5, "cut_points[4]", col=util$c_dark_highlight)

c <- 70

k <- 1
name <-paste0('cut_points[', c, ',', k, ']')
util$plot_expectand_pushforward(samples2[[name]],
                                50, flim=c(-9, 9), ylim=c(0, 2),
                                col=cols[k],
                                display_name='Interior Cut Points',
                                main=paste('Customer', c))
```

```
Warning in util$plot_expectand_pushforward(samples2[[name]], 50, flim = c(-9, :
8 values (0.2%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples2[[name]], 50, flim = c(-9, :
0 values (0.0%) fell above the histogram binning.
```

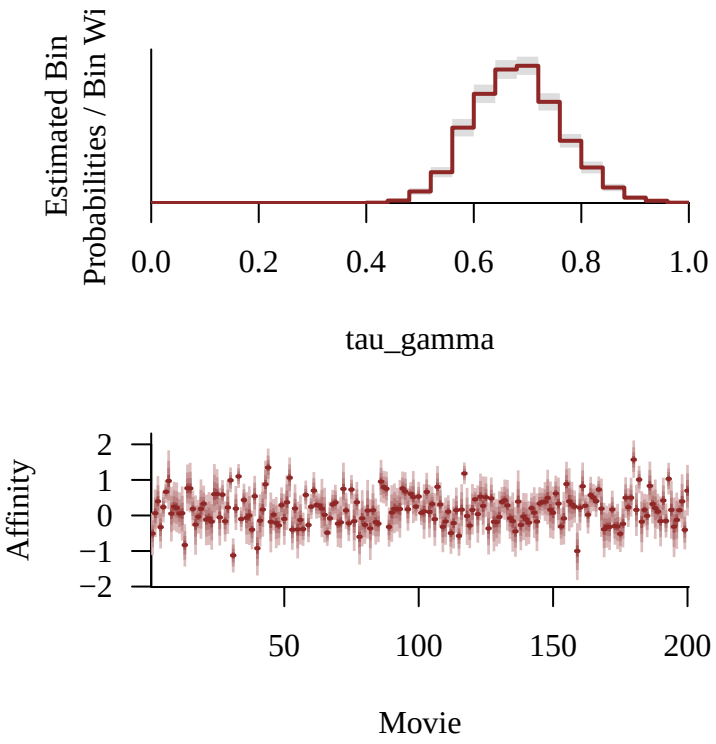
```
for (k in 2:4) {
  name <-paste0('cut_points[', c, ',', k, ']')
  util$plot_expectand_pushforward(samples2[[name]],
                                  50, flim=c(-9, 9),
                                  col=cols[k], border="#BBBBBB88",
                                  add=TRUE)
}

text(-5.75, 0.35, "cut_points[1]", col=util$c_mid)
text(-3, 0.85, "cut_points[2]", col=util$c_mid_highlight)
text(-2, 1.1, "cut_points[3]", col=util$c_dark)
text(1.0, 1.25, "cut_points[4]", col=util$c_dark_highlight)
```



This allow us to see that Customer 23 is pretty stingy; a movie affinity needs to be pretty large in order for the probability of a large rating to become non-negligible. On the other hand Customer 70 is much more generous; they are likely to give even a mediocre movie a high rating.

At the same time we can investigate the inferred affinities for each movie.

```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples2[['tau_gamma']],
                                25, flim=c(0, 1),
                                display_name='tau_gamma')

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma[', m, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     xlab="Movie",
                                     ylab="Affinity")
```



While there is a lot of uncertainty, not surprising given the limited data, we can definitely see some trends. For example the concentration of the `tau_gamma` marginal posterior distribution away from zero indicates substantial variation in the movie affinities. Moreover despite the uncertainties we can clearly differentiate the best movies from the worst movies. We'll do even more with these movie affinity inferences in the next section.

5 Hierarchical Customer Model

At this point we need to address a stark asymmetry in the last model. Although we're accounting for heterogeneity in both customer and movie behaviors, we're modeling only the latter hierarchically. We don't have any domain expertise that obstructs the exchangeability of the customers so why don't we model the individual interior cut points hierarchically as well? All we need is an appropriate multivariate population model.

Conveniently the induced Dirichlet prior naturally composes with the hyper Dirichlet population model that I discussed in my die fairness case study. This makes for a natural interior cut point population model that pools each customer's baseline ratings towards a common behavior.

Note that this is not the most general hierarchical model that we might consider. This model assumes that the heterogeneity in the interior cut points is independent of the heterogeneity

in the movie affinities; more generally those heterogeneities could be coupled together. That said I think that it is pretty reasonable to assume that how each customer translates their preferences into a movie rating is independent of those particular preferences.

```
fit <- stan(file="stan_programs/model3.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

The hierarchical model over the interior cut points has already proved useful – the mild computational issues that we had considered in the last model fit have vanished.

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

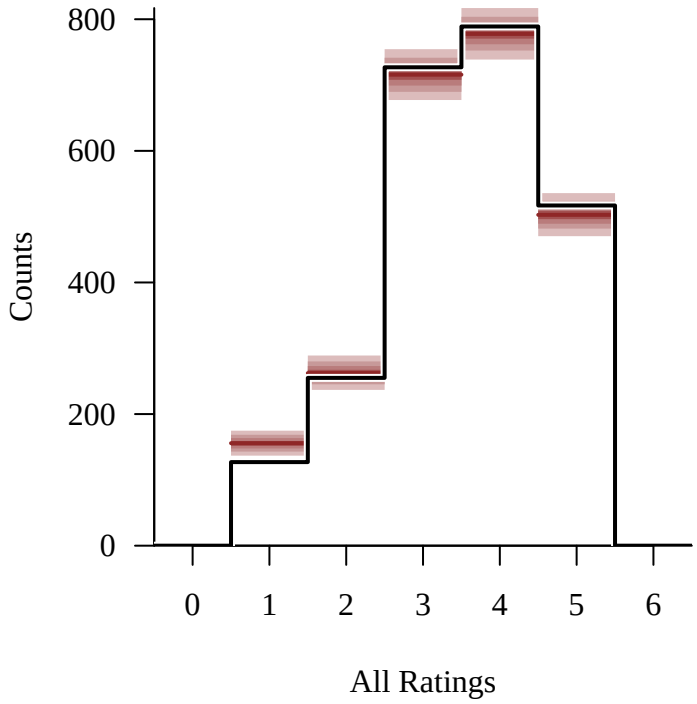
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples3 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples3,
                                       c('gamma_ncp',
                                         'tau_gamma',
                                         'cut_points',
                                         'mu_q', 'tau_q'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

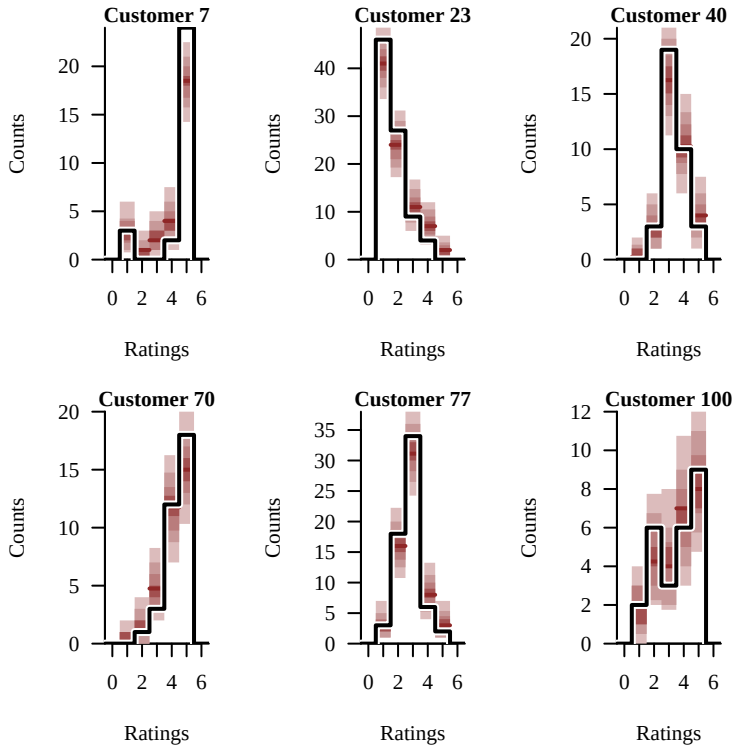
```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

Overall the retrodictive performance is a little bit better than in the previous model. In particular the agreements in the aggregate ratings histogram and stratified empirical covariances histogram have improved slightly. On the other hand the retrodictive tension in the stratified empirical variances stubbornly persists. We shouldn't expect too much performance, however, given that the hierarchical structure doesn't add any flexibility to the customer behaviors beyond what we had already included.

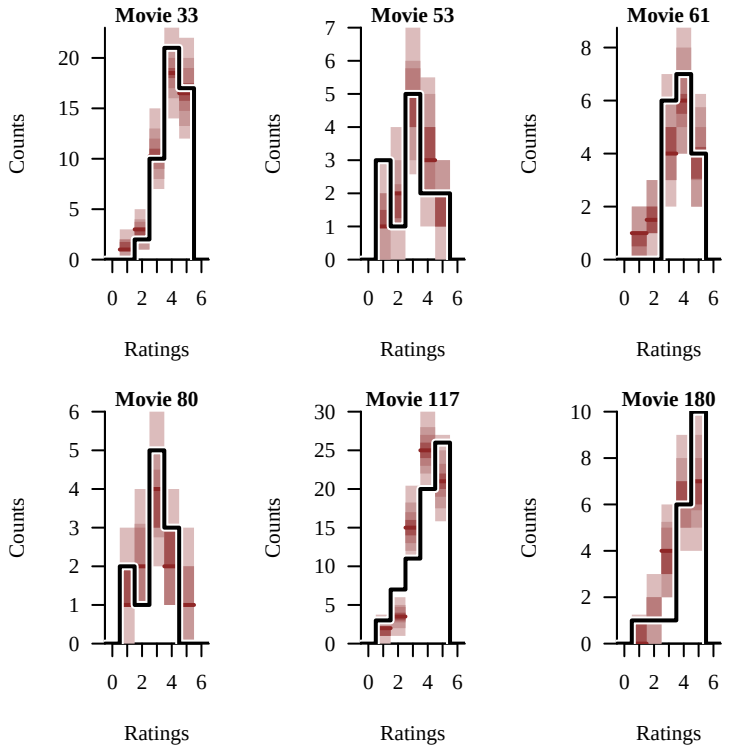
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples3, 'rating_pred', -0.5, 6.5, 1,
                         baseline_values=data$ratings,
                         xlab="All Ratings")
```




```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (c in c(7, 23, 40, 70, 77, 100)) {
  names <- sapply(which(data$customer_idxs == c),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples3, names)

  customer_ratings <- data$ratings[data$customer_idxs == c]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=customer_ratings,
                           xlab="Ratings",
                           main=paste('Customer', c))
}
```




```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (m in c(33, 53, 61, 80, 117, 180)) {
  names <- sapply(which(data$movie_idxs == m),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples3, names)

  movie_ratings <- data$ratings[data$movie_idxs == m]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=movie_ratings,
                           xlab="Ratings",
                           main=paste('Movie', m))
}
```

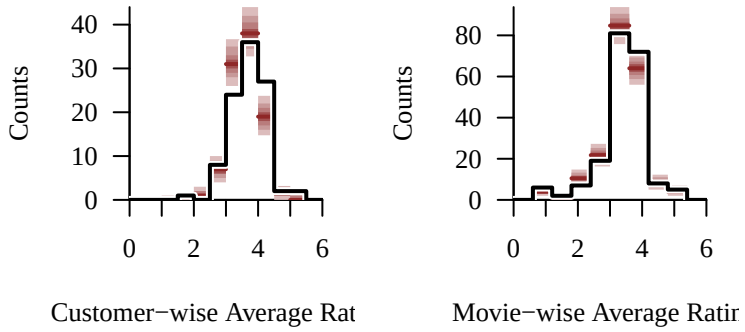


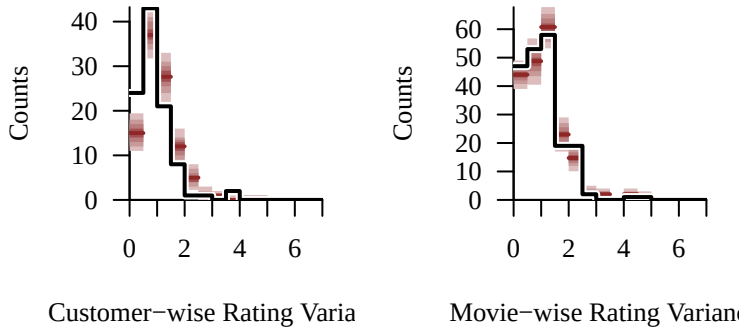

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples3, 'mean_rating_customer_pred',
                         0, 6, 0.5,
                         baseline_values=mean_rating_customer,
                         xlab="Customer-wise Average Ratings")

util$plot_hist_quantiles(samples3, 'mean_rating_movie_pred',
                         0, 6, 0.6,
                         baseline_values=mean_rating_movie,
                         xlab="Movie-wise Average Ratings")

util$plot_hist_quantiles(samples3, 'var_rating_customer_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_customer,
                         xlab="Customer-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 227 predictive values (0.1%) fell above the binning.
```

```
util$plot_hist_quantiles(samples3, 'var_rating_movie_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_movie,
                         xlab="Movie-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2264 predictive values (0.3%) fell above the binning.
```

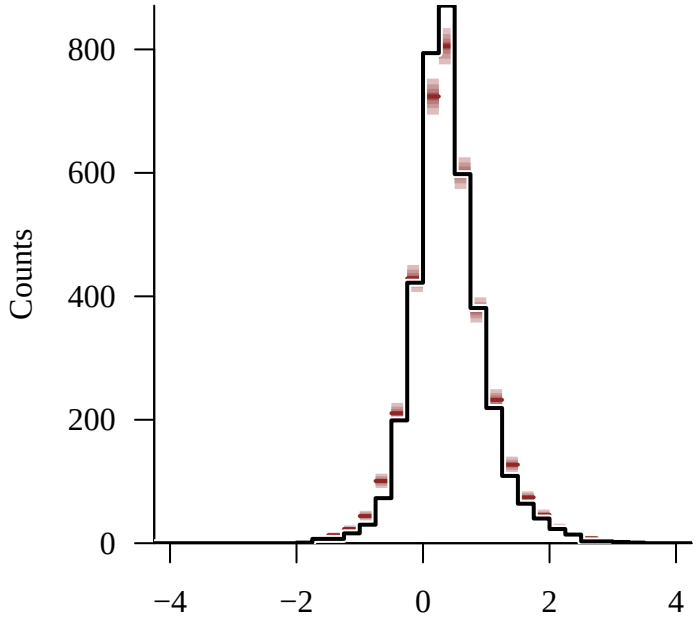





```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

filtered_samples <-
  util$filter_expectands(samples3,
                         covar_rating_movie_filt_names)

util$plot_hist_quantiles(filtered_samples, 'covar_rating_movie_pred',
                         -4.25, 4.25, 0.25,
                         baseline_values=covar_rating_movie_filt,
                         xlab="Filtered Movie-wise Rating Covariances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 355 predictive values (0.0%) fell below the binning.
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 432 predictive values (0.0%) fell above the binning.
```

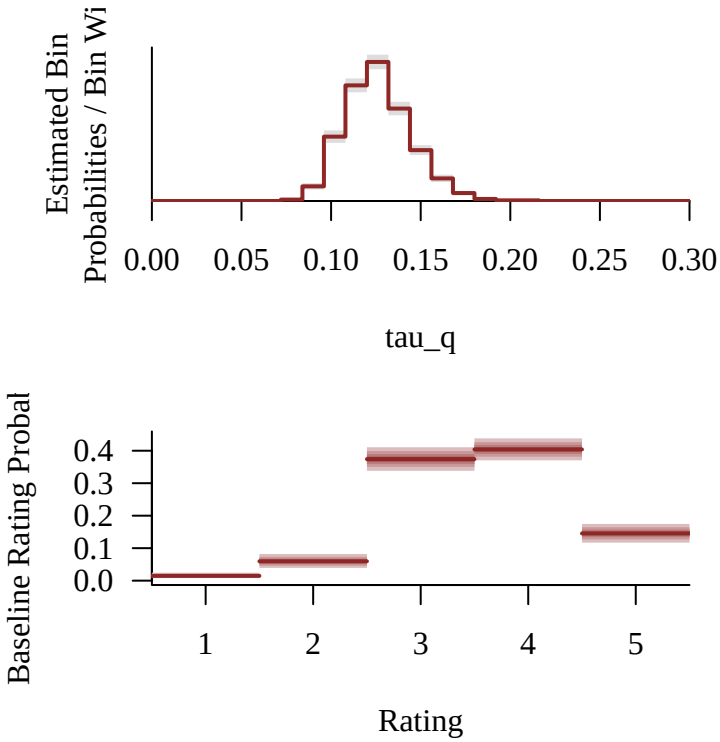


Filtered Movie−wise Rating Covariances

Assuming that we are satisfied with this retrodictive performance there many ways that we can explore and utilize the associated inferences.

For example with the new hierarchical structure we can investigate what we learned about not only each individual customer but also the population of customers. The marginal posterior distribution for `tau_q` suggests a small but non-negligible heterogeneity in the interior cut points. Moreover the inferences for the baseline probabilities `mu_q` suggest that customers are relatively optimistic, with four star ratings being more probable than tree star ratings for a neutral movie with zero affinity.

```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples3[['tau_q']],
                                25, flim=c(0, 0.3),
                                display_name='tau_q')

names <- sapply(1:5, function(k) paste0('mu_q[', k, ']'))
util$plot_disc_pushforward_quantiles(samples3, names,
                                     xlab="Rating",
                                     ylab="Baseline Rating Probability")
```



The regularizing influence of the hierarchical model is strongest for those customers with the fewest ratings. For example Customer 42 has only three observed ratings, and the hierarchical inferences for their interior cut points shift and narrow pretty substantially relative to the inferences from the previous model.

```
c <- 42
cat(sprintf("Customer %s: %s ratings",
            c, table(data$customer_idxs)[c]))
```

```
Customer 42: 3 ratings
```

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

lab2_xs <- c(1.5, 2, -4, -3)
lab2_ys <- c(0.15, 0.25, 0.35, 0.35)

lab3_xs <- c(-6, -5, 3, 3.5)
lab3_ys <- c(0.25, 0.4, 0.5, 0.6)

for (k in 1:4) {
  name <- paste0('cut_points[', c, ',', k, ']')
```

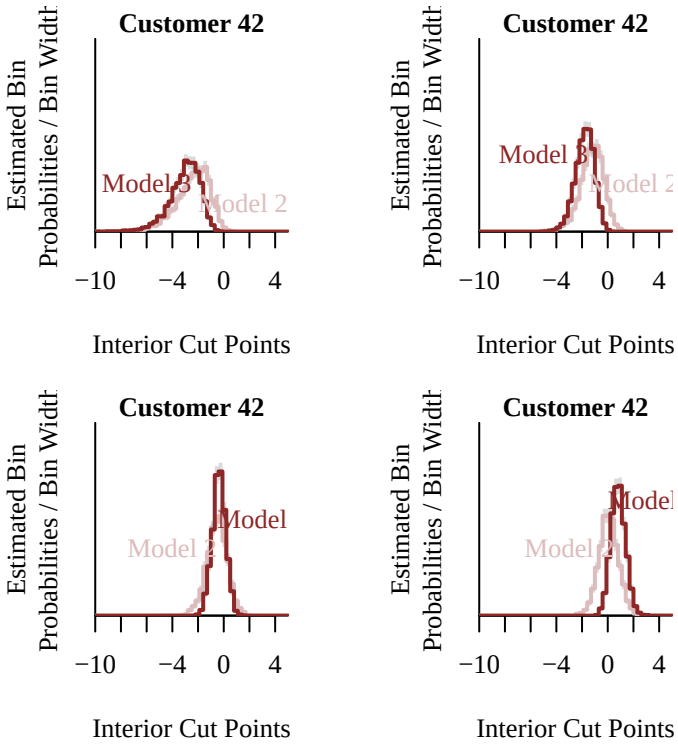
```
  util$plot_expectand_pushforward(samples2[[name]],
                                  50, flim=c(-10, 5), ylim=c(0, 1.0),
                                  col=util$c_light,
                                  display_name='Interior Cut Points',
                                  main=paste('Customer', c))

  name <- paste0('cut_points[', c, ',', k, ']')
  util$plot_expectand_pushforward(samples3[[name]],
                                  50, flim=c(-10, 5),
                                  col=util$c_dark, border="#BBBBBB88",
                                  add=TRUE)

  text(lab2_xs[k], lab2_ys[k], "Model 2", col=util$c_light)
  text(lab3_xs[k], lab3_ys[k], "Model 3", col=util$c_dark)
}
```

```
Warning in util$plot_expectand_pushforward(samples2[[name]], 50, flim = c(-10,
: 1 value (0.0%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples2[[name]], 50, flim = c(-10,
: 0 value (0.0%) fell above the histogram binning.
```

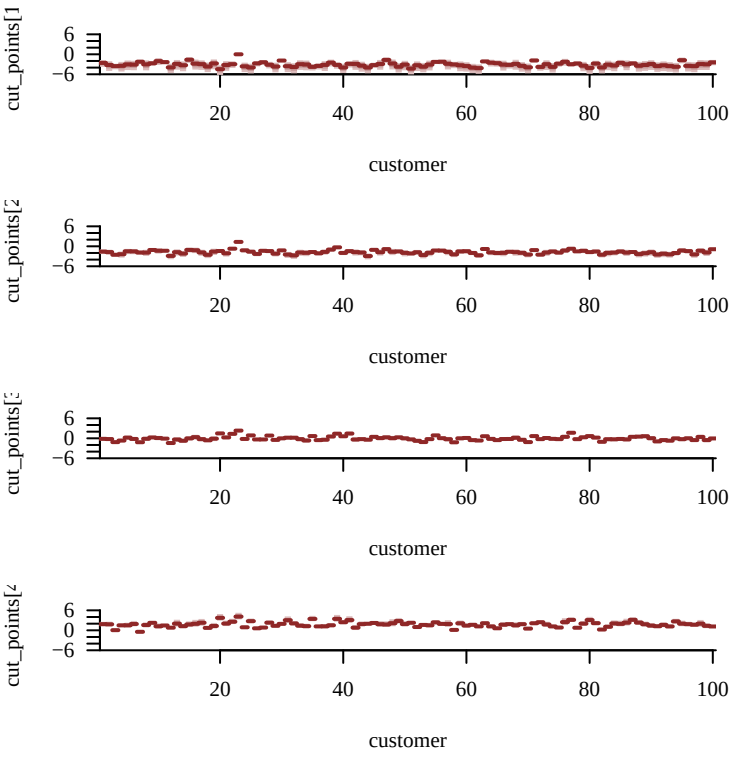
```
Warning in util$plot_expectand_pushforward(samples3[[name]], 50, flim = c(-10,
: 1 value (0.0%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples3[[name]], 50, flim = c(-10,
: 0 value (0.0%) fell above the histogram binning.
```



We can also see the impact of the hierarchical model if we examine the interior cut point inferences for the two models across all customers. The more uncertain customer inferences in the previous model, especially for the first and last cut points, are pulled towards the other customer inferences.

```
par(mfrow=c(4, 1), mar=c(5, 5, 1, 1))

for (k in 1:4) {
  names <- sapply(1:data$N_customers,
                  function(c) paste0('cut_points[', c, ',', k, ']'))

  yname <- paste0('cut_points[', k, ']')
  util$plot_disc_pushforward_quantiles(samples3, names,
                                       xlab="customer",
                                       display_ylim=c(-6, 6),
                                       ylab=yname)
}
```



The hierarchical influence, however, doesn't change the qualitative details. For example Customer 23 is still stingy with their ratings while Customer 70 is still generous.

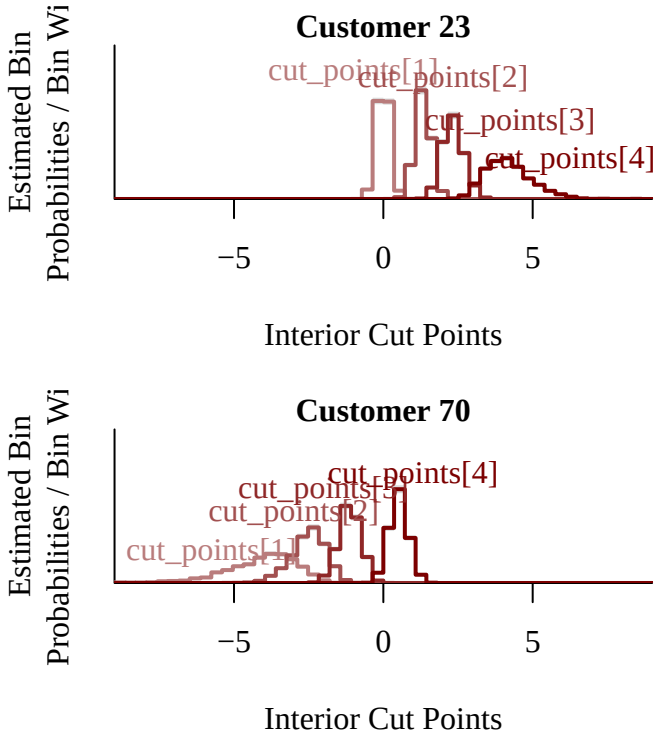
```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

cols <- c(util$c_mid, util$c_mid_highlight,
          util$c_dark, util$c_dark_highlight)

c <- 23

k <- 1
name <-paste0('cut_points[', c, ',', k, ']')
util$plot_expectand_pushforward(samples3[[name]],
                                50, flim=c(-9, 9), ylim=c(0, 2),
                                col=cols[k],
                                display_name='Interior Cut Points',
                                main=paste('Customer', c))


for (k in 2:4) {
  name <-paste0('cut_points[', c, ',', k, ']')
```

```
  util$plot_expectand_pushforward(samples3[[name]],
                                  50, flim=c(-9, 9),
                                  col=cols[k], border="#BBBBBB88",
                                  add=TRUE)
}

text(-1, 1.65, "cut_points[1]", col=util$c_mid)
text(2, 1.55, "cut_points[2]", col=util$c_mid_highlight)
text(4.25, 1, "cut_points[3]", col=util$c_dark)
text(6.25, 0.5, "cut_points[4]", col=util$c_dark_highlight)

c <- 70

k <- 1
name <-paste0('cut_points[', c, ',', k, ']')
util$plot_expectand_pushforward(samples3[[name]],
                                50, flim=c(-9, 9), ylim=c(0, 2),
                                col=cols[k],
                                display_name='Interior Cut Points',
                                main=paste('Customer', c))
```

```
Warning in util$plot_expectand_pushforward(samples3[[name]], 50, flim = c(-9, :
11 values (0.3%) fell below the histogram binning.
```

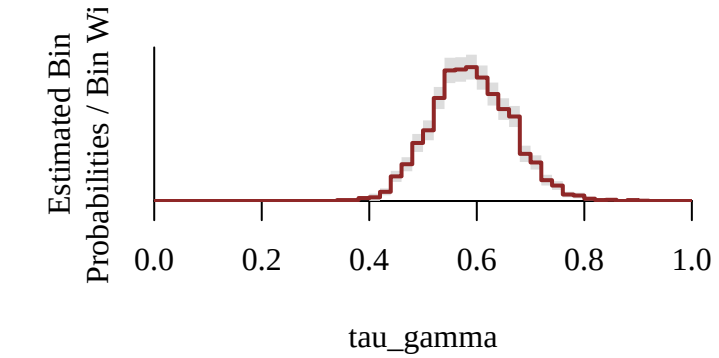
```
Warning in util$plot_expectand_pushforward(samples3[[name]], 50, flim = c(-9, :
0 values (0.0%) fell above the histogram binning.
```

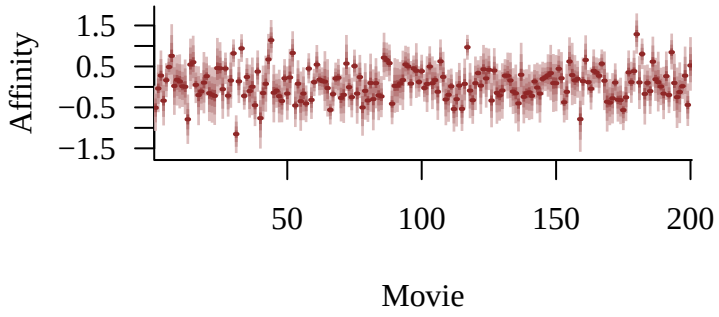
```
for (k in 2:4) {
  name <-paste0('cut_points[', c, ',', k, ']')
  util$plot_expectand_pushforward(samples3[[name]],
                                  50, flim=c(-9, 9),
                                  col=cols[k], border="#BBBBBB88",
                                  add=TRUE)
}

text(-5.75, 0.4, "cut_points[1]", col=util$c_mid)
text(-3, 0.9, "cut_points[2]", col=util$c_mid_highlight)
text(-2, 1.2, "cut_points[3]", col=util$c_dark)
text(1.0, 1.4, "cut_points[4]", col=util$c_dark_highlight)
```



Of course we can still investigate the behavior of individual movies and the hierarchical population of movies. The hierarchical regularization of the interior cut points allows the observed ratings to slightly better inform the movie affinities.

```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples3[['tau_gamma']],
                                50, flim=c(0, 1),
                                display_name='tau_gamma')

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma[', m, ']'))
util$plot_disc_pushforward_quantiles(samples3, names,
                                     xlab="Movie",
                                     ylab="Affinity")
```





Now let's go one step further than we did in the previous section and use these movie inferences to rank the movies by their expected affinities. This is just one heuristic for ranking items based on their inferred qualities, but one that has the advantage of being relatively fast to compute.

```
expected_affinity <- function(m) {
  util$ensemble_mcmc_est(samples3[[paste0('gamma[', m, ']')]])[1]
}

expected_affinities <- sapply(1:data$N_movies,
                              function(m) expected_affinity(m))

post_mean_ordering <- sort(expected_affinities, index.return=TRUE)$ix
```

We can then use this ranking to select the five worst movies for this particular set of customers.

```
print(data.frame("Rank"=200:196,
                 "Movie"=head(post_mean_ordering, 5)),
      row.names=FALSE)
```

```
 Rank Movie
  200    31
  199   159
  198    13
  197    40
  196   175
```

To be a bit less pessimistic we could also consider the five best movies for this particular set of customers.

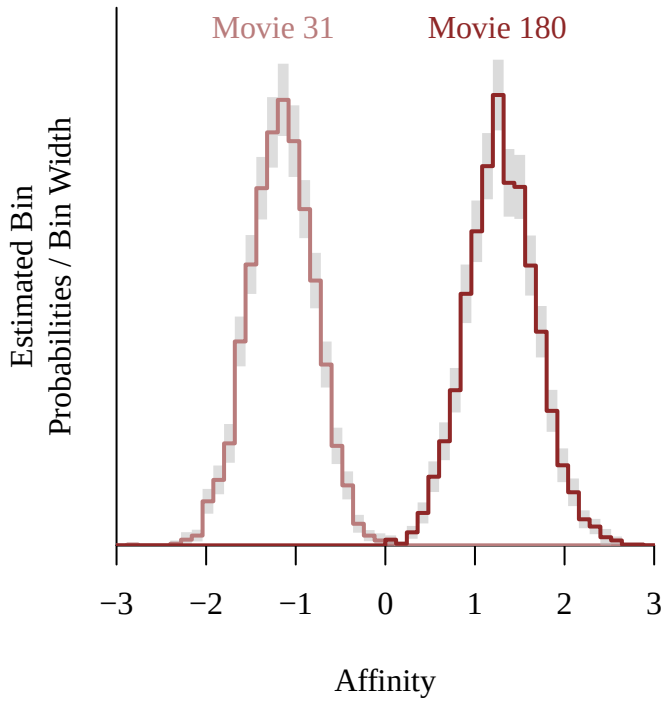
```
print(data.frame("Rank"=5:1,
                 "Movie"=tail(post_mean_ordering, 5)),
      row.names=FALSE)
```

```
 Rank Movie
    5   193
    4    33
    3   117
    2    44
    1   180
```

We also have a variety of ways to make inferential comparisons between two movies at a time. For example we could just overlay the marginal posterior distributions for each movie affinity.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

m1 <- head(post_mean_ordering, 1)
name <-paste0('gamma[', m1, ']')
util$plot_expectand_pushforward(samples3[[name]],
                                50, flim=c(-3, 3),
                                ylim=c(0, 1.35),
                                col=util$c_mid,
                                display_name='Affinity')
text(-1.25, 1.3, paste('Movie', m1), col=util$c_mid)

m2 <- tail(post_mean_ordering, 1)
name <-paste0('gamma[', m2, ']')
util$plot_expectand_pushforward(samples3[[name]],
                                50, flim=c(-3, 3),
                                col=util$c_dark,
```

```
                                    border="#BBBBBB88",
                                    add=TRUE)
text(1.25, 1.3, paste('Movie', m2), col=util$c_dark)
```



Even better we can directly compute the probability that one movie affinity is larger than the other. Here there is little ambiguity whether or not the top ranked movie is actually better than the worst ranked movie.

```
var_repl <- list('g1' = paste0('gamma[', m1,']'),
                 'g2' = paste0('gamma[', m2,']'))

p_est <-
  util$implicit_subset_prob(samples3,
                            function(g1, g2) g1 < g2,
                            var_repl)

format_string <- paste0("Posterior probability that movie %i affinity ",
                        "> movie %i affinity = %.3f +/- %.3f.")
cat(sprintf(format_string, m1, m2, p_est[1], 2 * p_est[2]))
```

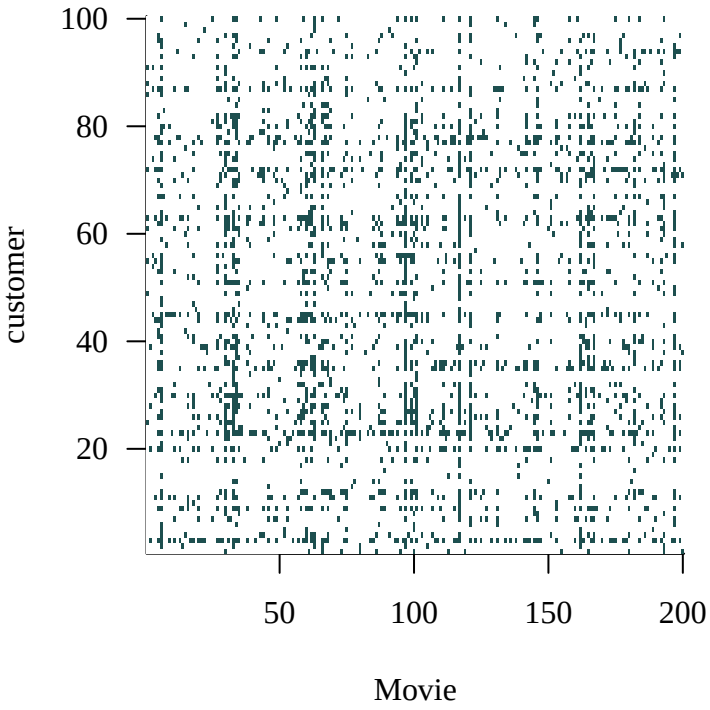
```
Posterior probability that movie 31 affinity > movie 180 affinity = 1.000 +/- 0.000.
```

6 Hierarchical Customer Model With Heterogeneous Affinities

There are two main limitations with the last model. Firstly there is the mild retrodictive tension in a few of the summary statistics. Secondly it makes the strong assumption that once the different interpretations of ratings are taken into account all customers have the same opinion about each movie. This for example prevents us from making personalized recommendations to each customer.

Incorporating individual tastes into the model would inform much more nuanced inferences and predictions. It could even resolve some of the retrodictive tension. The immediate challenge with trying to infer personal movie preferences, however, is that customers rate only a very small proportion of the available movies.

```
xs <- seq(1, data$N_movies, 1)
ys <- seq(1, data$N_customers, 1)
zs <- matrix(0, nrow=data$N_movies, ncol=data$N_customers)

for (n in 1:data$N_ratings) {
  zs[data$movie_idxs[n], data$customer_idxs[n]] <- 1
}

par(mfrow=c(1, 1), mar = c(5, 5, 1, 1))

image(xs, ys, zs, col=c("white", util$c_dark_teal),
      xlab="Movie", ylab="customer")
```



In the machine learning literature the problem of filling in unobserved comparisons, like customer-movie ratings in this application, is often known as "matrix completion" in analogy to filling in the missing cells in this pairwise comparison matrix.

Because most of the ratings are unobserved the only way to inform individual customer preferences for each movie is to pool correlations in each movie affinities across customers. For example we might model each customers' movie affinities as common baseline affinities plus individual deviations,

$$\gamma_c = \gamma_0 + \delta_c.$$

We can then pool these individual deviations together with a multivariate normal population model,

$$p(\delta_c) = \text{multi-normal}(\mathbf{0}, \Sigma)$$

with

$$\Sigma = \tau_\gamma^T \cdot \Phi_\gamma \cdot \tau_\gamma.$$

Whatever we learn about the population baseline γ_0, the population scales τ_γ, and the population correlations Φ_γ fill in whatever elements of any particular customer affinity vector γ_c that might be unobserved.

Because our domain expertise about the movies is still exchangeable we can model the baseline movie affinities hierarchically as well,

$$p(\gamma_{0,m}) = \text{normal}(0, \tau_{\gamma_0}).$$

We're now modeling the interior cut points, the baseline movie affinities, *and* the individual customer affinities hierarchically all at the same time. Pretty neat.

Given the sparsity of observed ratings we'll implement the normal and multivariate normal hierarchical models with non-centered parameterizations.

```
fit <- stan(file="stan_programs/model4.stan",
            data=data, seed=8438338, init=0,
            warmup=1000, iter=2024, refresh=0)
```

People complain about frustration with diagnostic warnings, but how bad can they be if we don't see serious warnings for this complex model with 40,706 degrees of freedom? Turns out Hamiltonian Monte Carlo is pretty good!

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

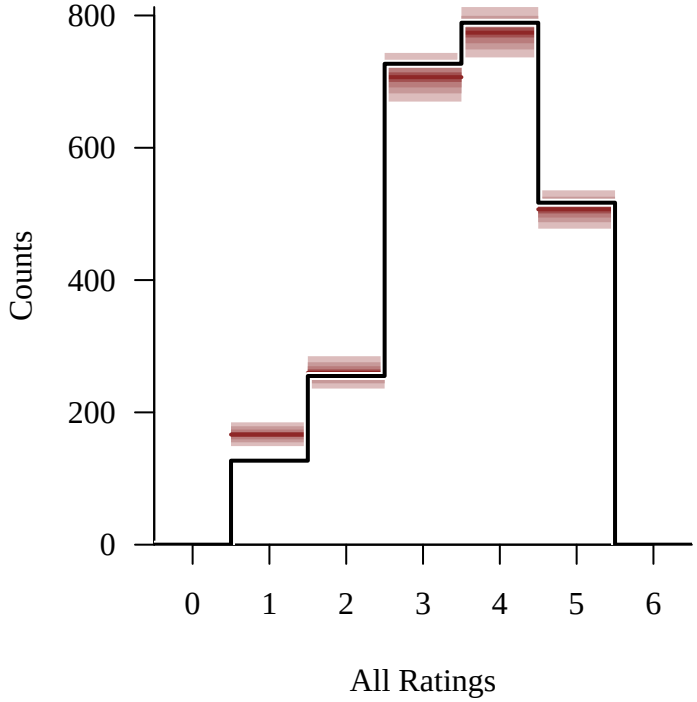
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples4 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples4,
                                       c('gamma0_ncp',
                                         'tau_gamma0',
                                         'delta_gamma_ncp',
                                         'tau_delta_gamma',
                                         'L_delta_gamma',
                                         'cut_points',
                                         'mu_q', 'tau_q'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples,
                                     exclude_zvar=TRUE)
```

```
tau_delta_gamma[97]:
  Chain 1: hat{ESS} (39.565) is smaller than desired (100).


Small empirical effective sample sizes result in imprecise Markov chain
Monte Carlo estimators.
```

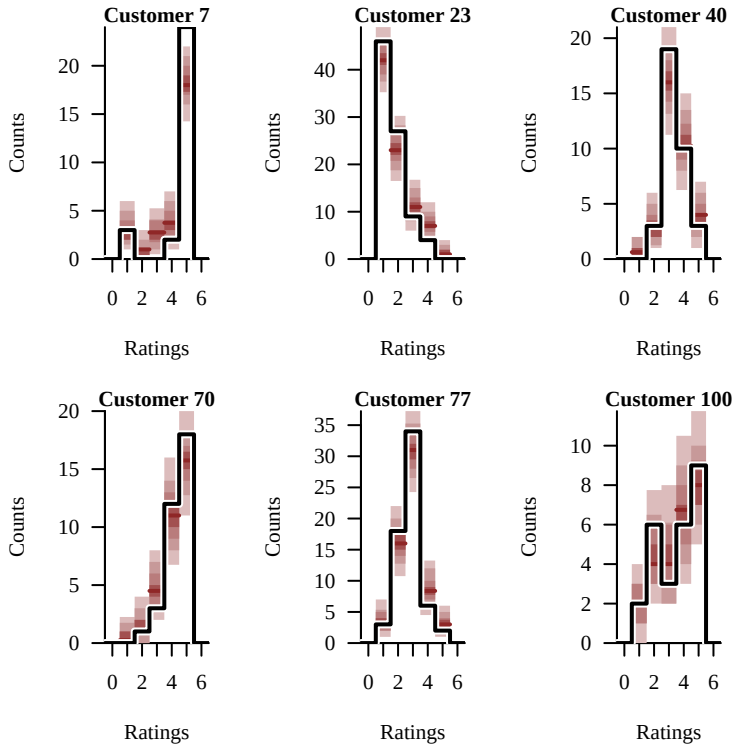
Unfortunately all of this added sophisticated doesn't actually seem to improve the retrodictive performance much. Even the retrodictive tension in the empirical covariances, which should be particularly sensitive to added flexibility in the new model, is similar to what we saw with the previous model. One possibility is that the multivariate normal population model is isn't sufficiently heavy-tailed to accommodate the more extreme tastes of the customers.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples4, 'rating_pred', -0.5, 6.5, 1,
                         baseline_values=data$ratings,
                         xlab="All Ratings")
```

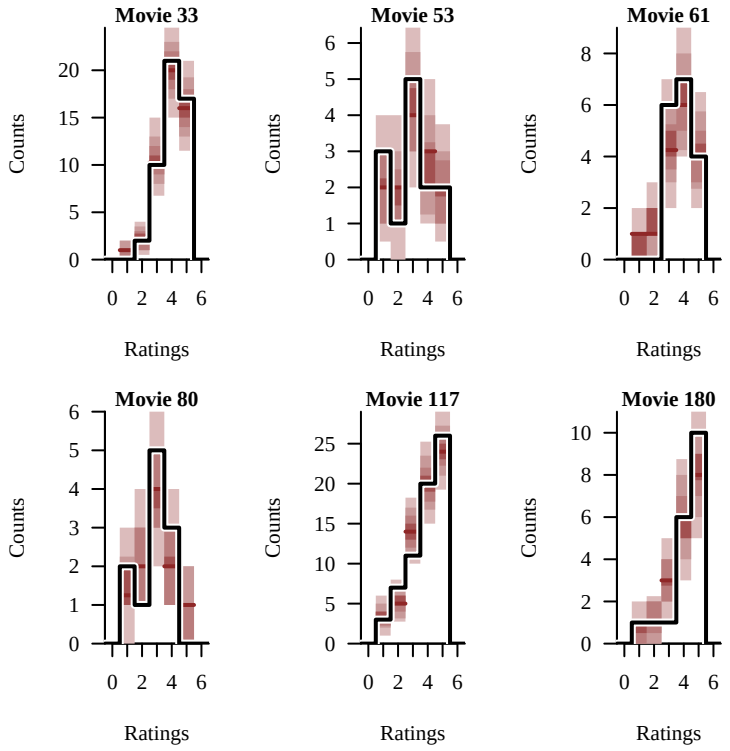



```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (c in c(7, 23, 40, 70, 77, 100)) {
  names <- sapply(which(data$customer_idxs == c),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples4, names)

  customer_ratings <- data$ratings[data$customer_idxs == c]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
```

```
                            baseline_values=customer_ratings,
                            xlab="Ratings",
                            main=paste('Customer', c))
}
```




```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (m in c(33, 53, 61, 80, 117, 180)) {
  names <- sapply(which(data$movie_idxs == m),
                  function(n) paste0('rating_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples4, names)

  movie_ratings <- data$ratings[data$movie_idxs == m]
  util$plot_hist_quantiles(filtered_samples, 'rating_pred',
                           -0.5, 6.5, 1,
                           baseline_values=movie_ratings,
                           xlab="Ratings",
                           main=paste('Movie', m))
}
```

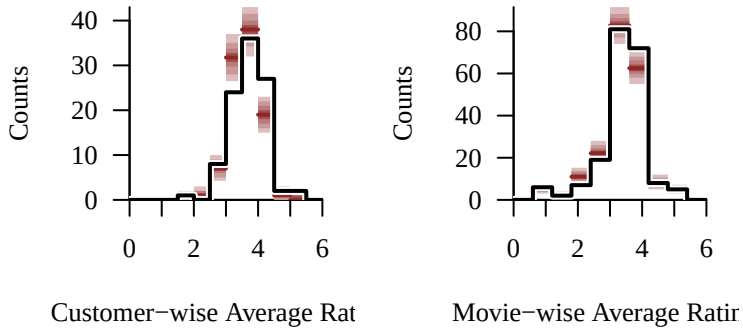


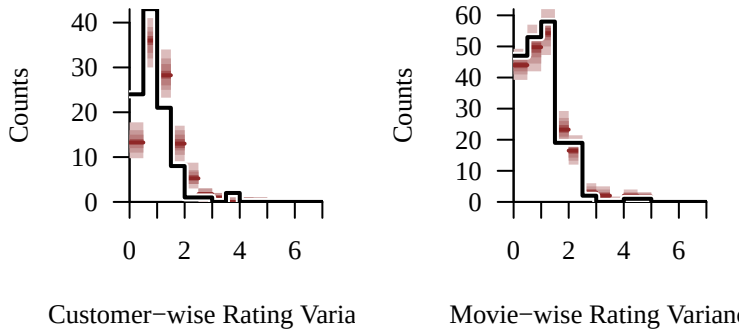

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_hist_quantiles(samples4, 'mean_rating_customer_pred',
                         0, 6, 0.5,
                         baseline_values=mean_rating_customer,
                         xlab="Customer-wise Average Ratings")

util$plot_hist_quantiles(samples4, 'mean_rating_movie_pred',
                         0, 6, 0.6,
                         baseline_values=mean_rating_movie,
                         xlab="Movie-wise Average Ratings")

util$plot_hist_quantiles(samples4, 'var_rating_customer_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_customer,
                         xlab="Customer-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 307 predictive values (0.1%) fell above the binning.
```

```
util$plot_hist_quantiles(samples4, 'var_rating_movie_pred',
                         0, 7, 0.5,
                         baseline_values=var_rating_movie,
                         xlab="Movie-wise Rating Variances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2321 predictive values (0.3%) fell above the binning.
```

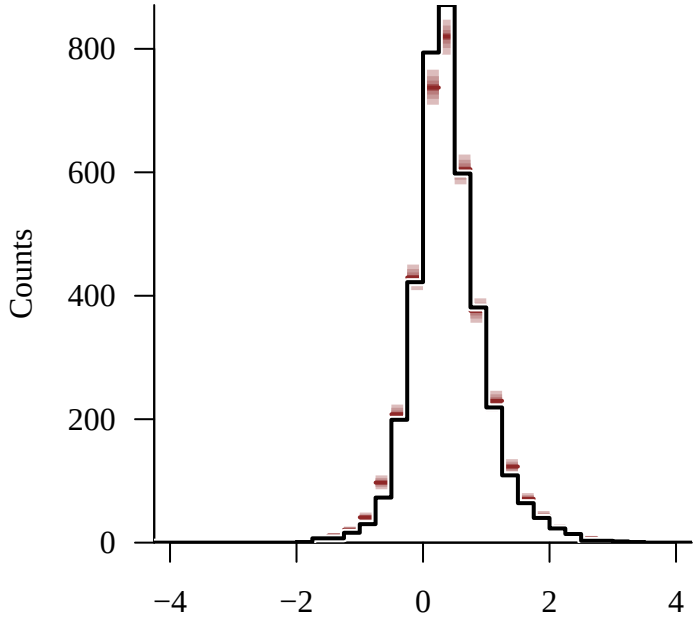





```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

filtered_samples <-
  util$filter_expectands(samples4,
                         covar_rating_movie_filt_names)

util$plot_hist_quantiles(filtered_samples, 'covar_rating_movie_pred',
                         -4.25, 4.25, 0.25,
                         baseline_values=covar_rating_movie_filt,
                         xlab="Filtered Movie-wise Rating Covariances")
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 113 predictive values (0.0%) fell below the binning.
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 95 predictive values (0.0%) fell above the binning.
```



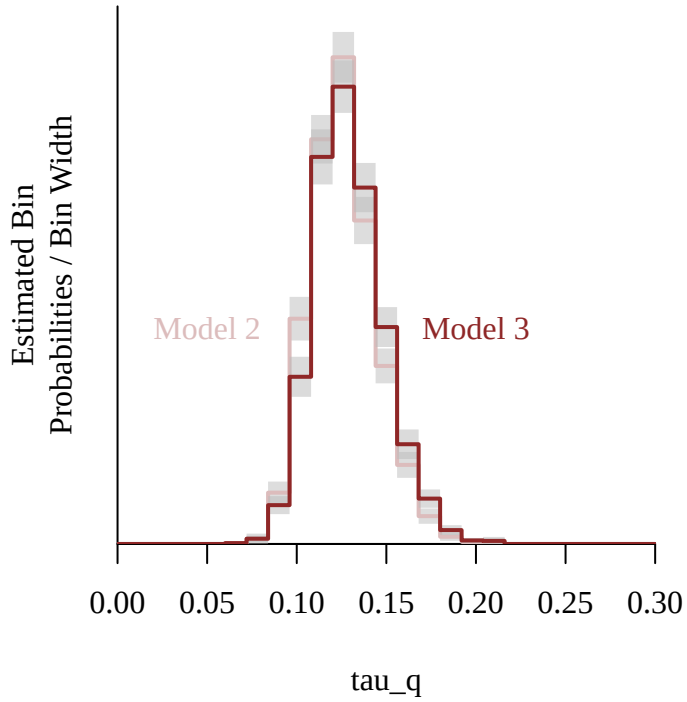
Filtered Movie−wise Rating Covariances

The lack of any substantial improvements in the retrodictive performance suggests that we might not have included enough data to really resolve individual customer preferences quite yet. We can directly quantify how well we can resolve individual customer preferences by examining our posterior inferences.

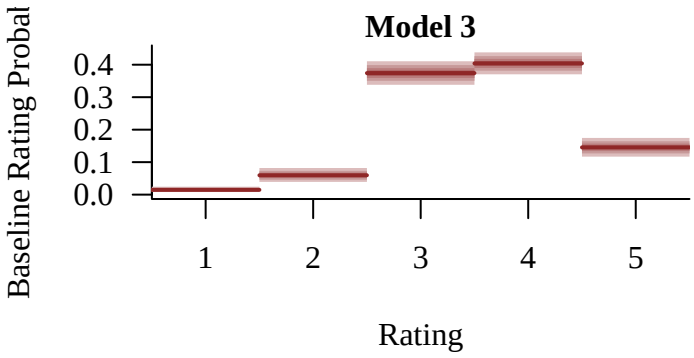
Posterior inferences for the interior cut point population behaviors are mostly consistent with the previous model, although the baseline rating probabilities do slightly shift down to be more centered around 3. This suggest that the previous model may have been contorting itself a bit to account for the variation in customer tastes.

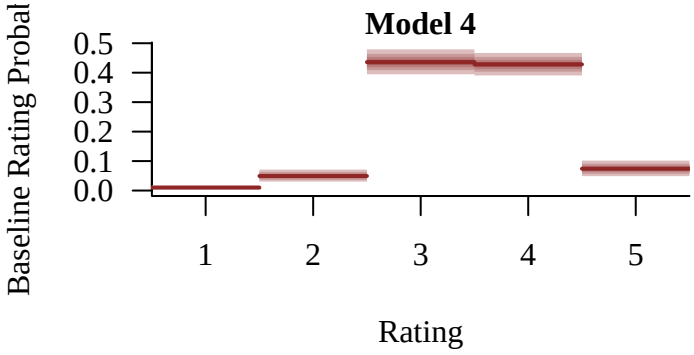
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples3[['tau_q']],
                                25, flim=c(0, 0.3),
                                col=util$c_light,
                                display_name='tau_q')
text(0.05, 10, "Model 2", col=util$c_light)

util$plot_expectand_pushforward(samples4[['tau_q']],
                                25, flim=c(0, 0.3),
```

```
                                    col=util$c_dark,
                                    border="#BBBBBB88",
                                    add=TRUE)
text(0.2, 10, "Model 3", col=util$c_dark)
```




```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:5, function(k) paste0('mu_q[', k, ']'))
util$plot_disc_pushforward_quantiles(samples3, names,
                                     xlab="Rating",
                                     ylab="Baseline Rating Probability",
                                     main="Model 3")

names <- sapply(1:5, function(k) paste0('mu_q[', k, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
                                     xlab="Rating",
                                     ylab="Baseline Rating Probability",
                                     main="Model 4")
```

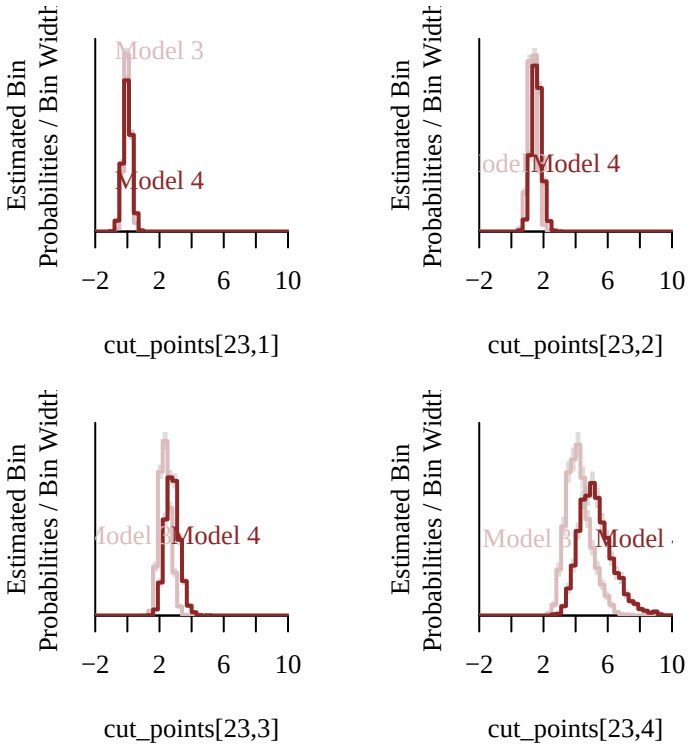




Similarly some of the individual customer interior cut points change slightly. For example the cut points for Customer 23 shift a bit towards larger values.

```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

c <- 23

lab3_xs <- c(2, -0.5, 0, 1)
lab3_ys <- c(1.75, 0.5, 0.5, 0.25)

lab4_xs <- c(2, 4, 5.5, 8)
lab4_ys <- c(0.5, 0.5, 0.5, 0.25)

for (k in 1:4) {
  name <-paste0('cut_points[', c, ',', k, ']')
  util$plot_expectand_pushforward(samples3[[name]],
                                  40, flim=c(-2, 10),
                                  col=util$c_light,
                                  display_name=name)
  util$plot_expectand_pushforward(samples4[[name]],
                                  40, flim=c(-2, 10),
                                  col=util$c_dark,
```

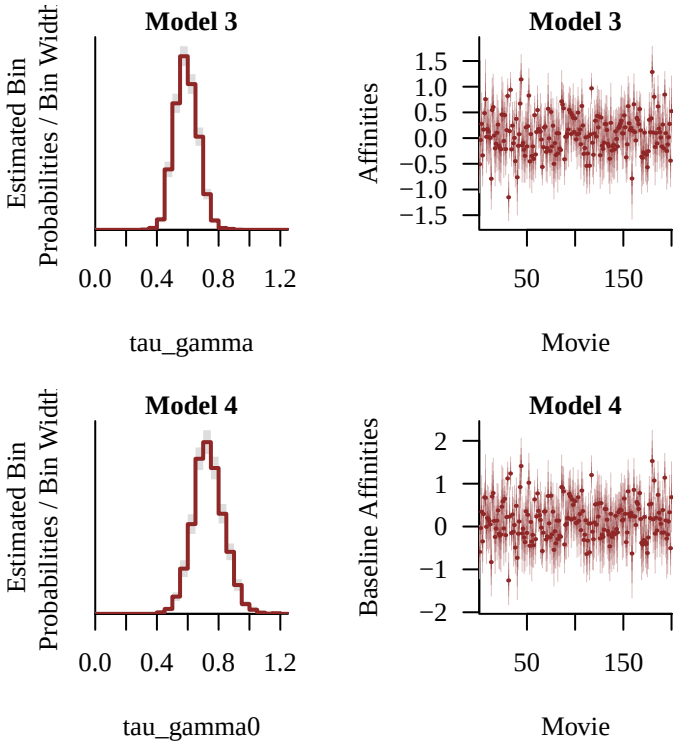
```
                                    border="#BBBBBB88",
                                    add=TRUE)

  text(lab3_xs[k], lab3_ys[k], "Model 3", col=util$c_light)
  text(lab4_xs[k], lab4_ys[k], "Model 4", col=util$c_dark)
}
```



The baseline movie affinities emulate the universal movie preferences in the previous model and indeed inferences for them are similar, if slightly more heterogeneous.

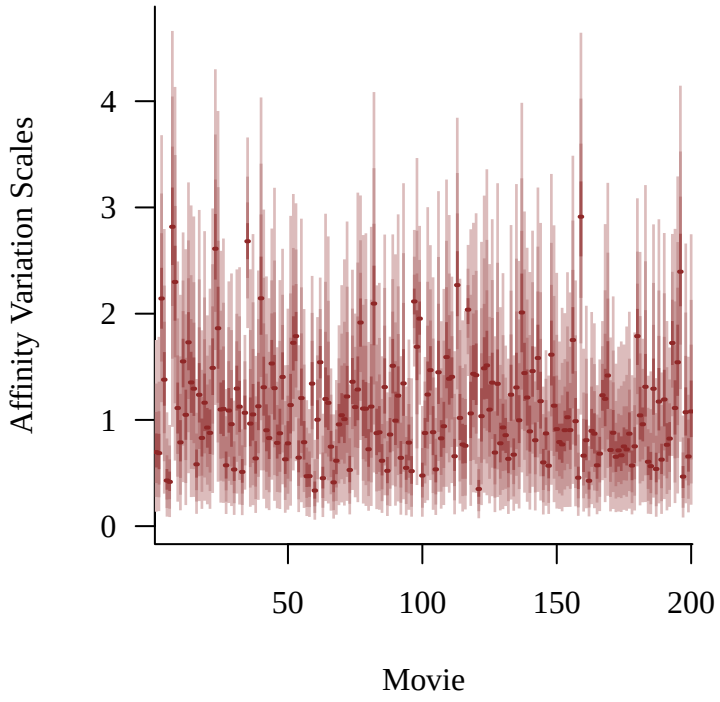
```
par(mfrow=c(2, 2), mar=c(5, 5, 1, 1))

util$plot_expectand_pushforward(samples3[['tau_gamma']],
                                25, flim=c(0, 1.25),
                                display_name="tau_gamma",
                                main="Model 3")

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma[', m, ']'))
util$plot_disc_pushforward_quantiles(samples3, names,
                                     xlab="Movie",
```

```
                                        ylab="Affinities",
                                        main="Model 3")


util$plot_expectand_pushforward(samples4[['tau_gamma0']],
                                25, flim=c(0, 1.25),
                                display_name="tau_gamma0",
                                main="Model 4")

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma0[', m, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
                                        xlab="Movie",
                                        ylab="Baseline Affinities",
                                        main="Model 4")
```



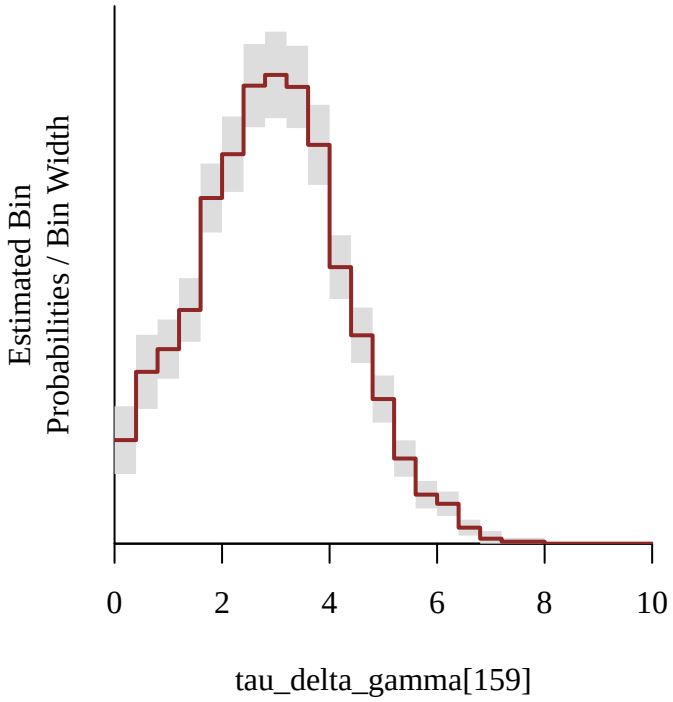
Now, however, we can investigate the preferences idiosyncratic to each customer. For example the individual movie affinity scales quantify how much the customers disagree about a particular movie.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$N_movies,
                function(m) paste0('tau_delta_gamma[', m, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
                                     xlab="Movie",
                                     ylab="Affinity Variation Scales")
```



Overall there is a lot of uncertainty, with the inferences for most of the movie affinity scales concentrating around zero. That said a few stand out. For example the posterior inferences for $\tau_{\gamma,159}$ are starting to pull away from zero, suggesting that customers tend to disagree about the quality of this movie more than usual.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

m <- 159
name <- paste0('tau_delta_gamma[', m, ']')
util$plot_expectand_pushforward(samples4[[name]],
                                25, flim=c(0, 10),
                                display_name=name)
```



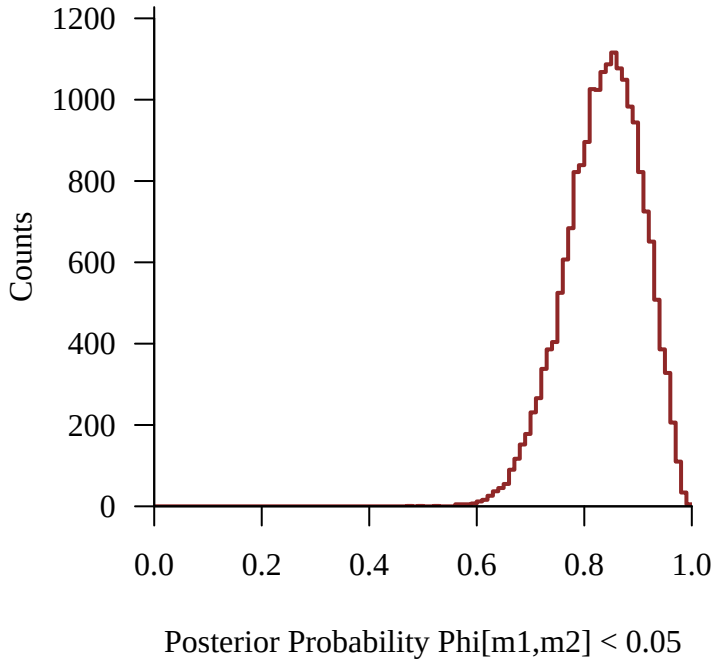
The inferred correlations in the multivariate normal population model allow us to inform predictions for how a customer would react to unrated movies given the movies they have rated. While most of the correlations are consistent with zero, here approximated by how much of their marginal posterior probability concentrates on values below 0.05, a few are consistent with larger values.

Note that I've had to break out some custom, heavily-optimized code here to calculate the 19,900 posterior probabilities reasonably efficiently.

```
apply_pushforward_expectation <- function(expectand_vals_list,
                                          pushforward_expectand,
                                          input_names) {
  arg_name <- formalArgs(pushforward_expectand)

  C <- dim(expectand_vals_list[[1]])[1]
  S <- dim(expectand_vals_list[[1]])[2]

  expectand_vals_env <- as.environment(expectand_vals_list)
  access_val <- function(name) {
    expectand_vals_env[[name]][c, s]
  }

  I <- length(input_names)
```

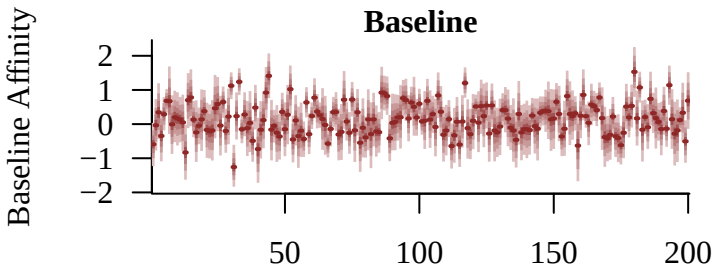
```
  psh_fwd_exp_vals <- as.list(rep(NA, I))
  pushforward_vals <- matrix(NA, nrow=C, ncol=S)

  for (i in 1:I) {
    for (c in 1:C) {
      for (s in 1:S) {
        pushforward_vals[c, s] <-
          as.numeric(do.call(pushforward_expectand,
                             setNames(list(access_val(input_names[[i]])),
                                      arg_name)))
      }
    }
    psh_fwd_exp_vals[[i]] <- util$ensemble_mcmc_est(pushforward_vals)[1]
  }
  return(as.numeric(psh_fwd_exp_vals))
}
```

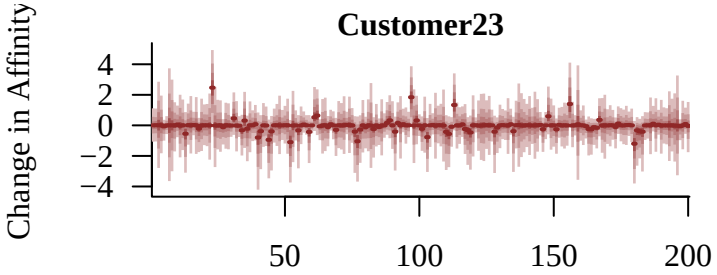
```
M <- data$N_movies * (data$N_movies - 1) / 2
input_names <- as.list(rep(NA, M))

idx <- 1
for (m1 in 2:data$N_movies) {
  for (m2 in 1:(m1 - 1)) {
    input_names[[idx]] <- paste0('Phi[', m1, ',', m2, ']')
    idx <- idx + 1
  }
}

corr_probs <- apply_pushforward_expectation(samples4,
                                            function(phi) phi < 0.05,
                                            input_names)

par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(corr_probs, 0, 1, 0.01, col=util$c_dark,
                    xlab="Posterior Probability Phi[m1,m2] < 0.05")
```



That said it's more practical to investigate the consequence of these correlations. In particular we can look at the movie affinities for each customer by adding together the common baselines with their individual preferences. Here we'll consider Customer 23.

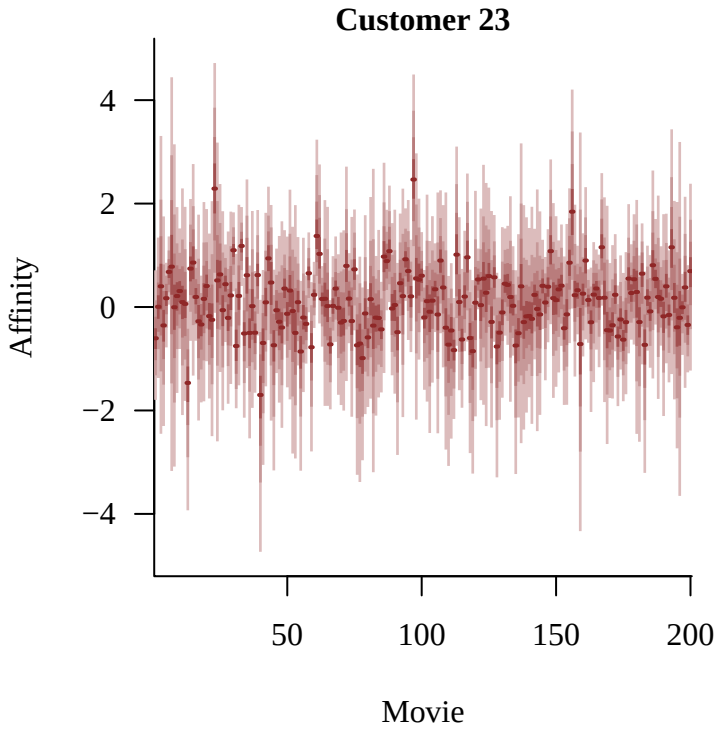
```
par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

c <- 23

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma0[', m, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
                                     xlab="Movie",
                                     ylab="Baseline Affinity",
                                     main="Baseline")

names <- sapply(1:data$N_movies,
                function(m) paste0('delta_gamma[', c, ',', m, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
                                     xlab="Movie",
                                     ylab="Change in Affinity",
                                     main=paste0('Customer', c))
```






```
expectands <- sapply(1:data$N_movies,
                     function(m)
                       local({ idx = m; function(x1, x2)
                                          x1[idx] + x2[idx] }) )
names(expectands) <- sapply(1:data$N_movies,
                            function(m)
                            paste0('gamma[', c, ',', m, ']'))

var_repl <- list('x1'=array(sapply(1:data$N_movies,
                                   function(m)
                                   paste0('gamma0[', m, ']'))),
                 'x2'=array(sapply(1:data$N_movies,
                                   function(m)
                                   paste0('delta_gamma[', c,
                                          ',', m, ']'))))

affinity_samples <-
  util$eval_expectand_pushforwards(samples4,
                                   expectands,
                                   var_repl)
```

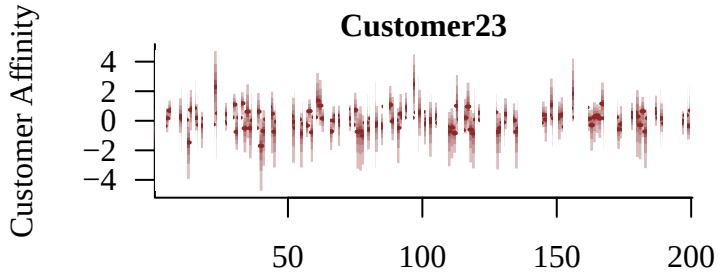
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$N_movies,
                function(m) paste0('gamma[', c, ',', m, ']'))

util$plot_disc_pushforward_quantiles(affinity_samples, names,
                                     xlab="Movie",
                                     ylab="Affinity",
                                     main=paste('Customer', c))
```

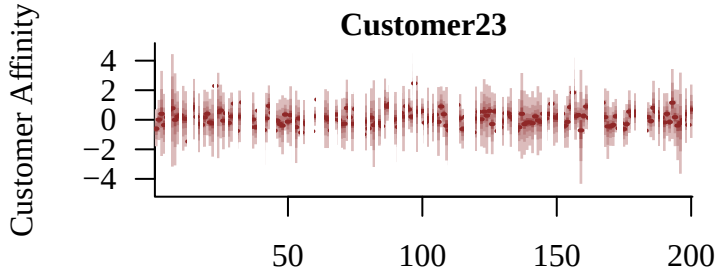


Even better we can separately visualize the movie affinities that are directly informed by observed ratings and those informed by only the multivariate normal hierarchical model. Note that the affinities for the movies that Customer 23 did not rate are not only much more uncertain but also much more uniform.

```
rated_movie_idxs <- data$movie_idxs[data$customer_idxs == c]
unrated_movie_idxs <- setdiff(1:data$N_movies, rated_movie_idxs)

par(mfrow=c(2, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$N_movies,
```

```
                function(m) paste0('gamma[', c, ',', m, ']'))
util$plot_disc_pushforward_quantiles(affinity_samples, names,
                                     xlab="Rated Movie",
                                     ylab="Customer Affinity",
                                     main=paste0('Customer', c))
for (m in unrated_movie_idxs) {
  polygon(c(m - 0.5, m + 0.5, m + 0.5, m- 0.5),
          c(-4.75, -4.75, 4.75, 4.75), col="white", border=NA)
}

util$plot_disc_pushforward_quantiles(affinity_samples, names,
                                     xlab="Unrated Movie",
                                     ylab="Customer Affinity",
                                     main=paste0('Customer', c))
for (m in rated_movie_idxs) {
  polygon(c(m - 0.5, m + 0.5, m + 0.5, m- 0.5),
          c(-4.75, -4.75, 4.75, 4.75), col="white", border=NA)
}
```

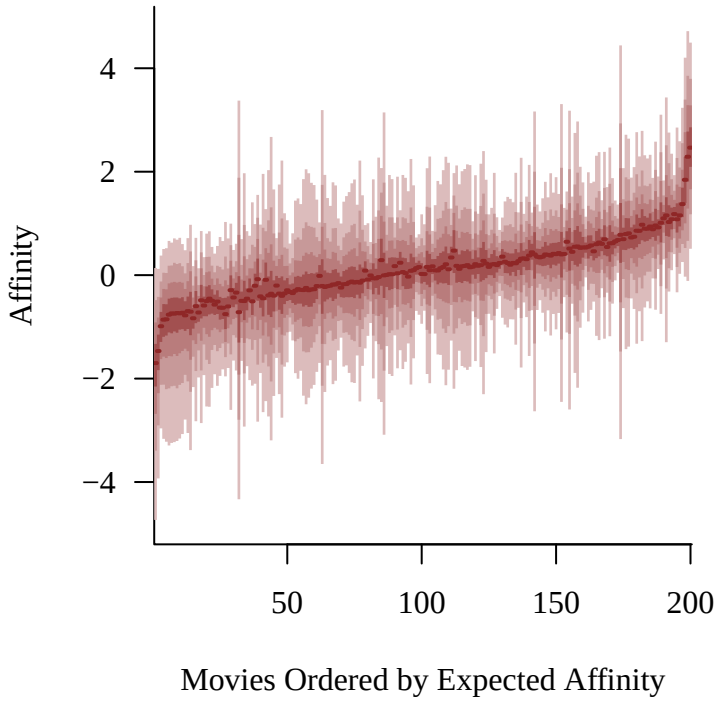




The immediate benefit of modeling individual preferences is that we can now make movie recommendations bespoke to Customer 23.

```
expected_affinity <- function(m) {
  name <- paste0('gamma[', c, ',', m, ']')
  util$ensemble_mcmc_est(affinity_samples[[name]])[1]
}

expected_affinities <- sapply(1:data$N_movies,
                              function(m) expected_affinity(m))

post_mean_ordering <- sort(expected_affinities, index.return=TRUE)$ix
```

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(post_mean_ordering,
                function(m) paste0('gamma[', c, ',', m, ']'))

xname <- "Movies Ordered by Expected Affinity"
util$plot_disc_pushforward_quantiles(affinity_samples, names,
                                     xlab=xname,
                                     ylab="Affinity")
```



From this we can infer what movies we think Customer 23 will like the least.

```
print(data.frame("Rank"=200:196,
                 "Movie"=head(post_mean_ordering, 5)),
      row.names=FALSE)
```

```
 Rank Movie
  200    40
  199    13
  198    78
  197    55
  196   119
```

As well as what movies we think they will like the most.

```
print(data.frame("Rank"=5:1,
                 "Movie"=tail(post_mean_ordering, 5)),
      row.names=FALSE)
```

```
 Rank Movie
    5   167
    4    61
    3   156
    2    23
    1    97
```

Of course there's not much utility in recommending a customer a movie that they've already seen. A much more useful recommendation is for movies that they haven't yet seen but might enjoy.

Here let's assume that a movie has been unrated by a customer only if the customer has not yet seen it. Consequently the recommendation task comes down to inferring the unrated movies with the highest affinities for Customer 23.

```
expected_affinities <- sapply(unrated_movie_idxs,
                              function(m) expected_affinity(m))

post_mean_ordering <- sort(expected_affinities, index.return=TRUE)$ix
```

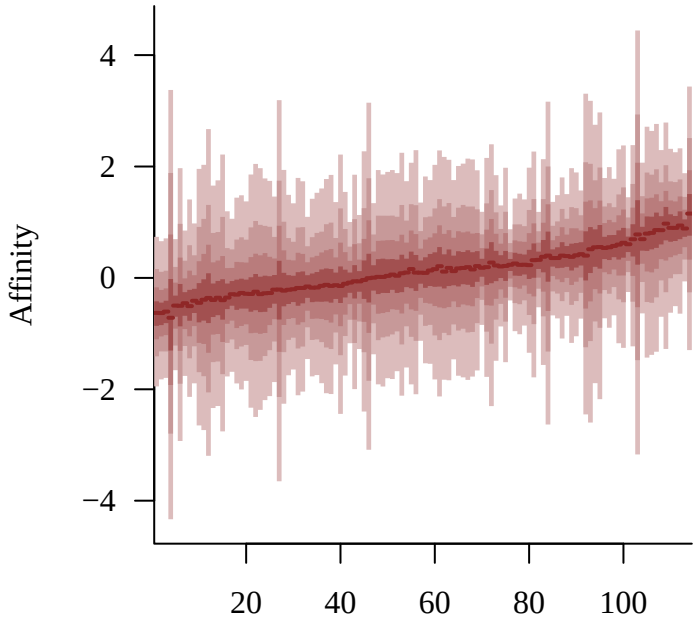
We can finally present a list of the top new movies to recommend to Customer 23.

```
print(data.frame("Rank"=10:1,
                 "Movie"=tail(unrated_movie_idxs[post_mean_ordering], 10)),
      row.names=FALSE)
```

```
 Rank Movie
   10    72
    9   186
    8    15
    7   155
    6    86
    5   161
    4   107
    3    43
    2    87
    1   193
```

All of this said we should have only mild confidence in these recommendations given the large uncertainties.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(unrated_movie_idxs[post_mean_ordering],
                function(m) paste0('gamma[', c, ',', m, ']'))

xname <- "Unrated Movies Ordered by Expected Affinity"
util$plot_disc_pushforward_quantiles(affinity_samples, names,
                                     xlab=xname,
                                     ylab="Affinity")
```



Unrated Movies Ordered by Expected Affinity

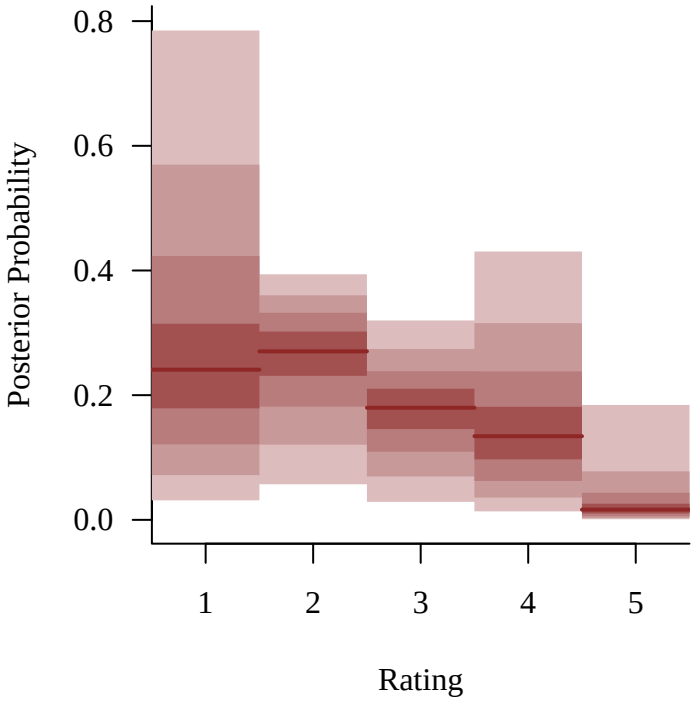
One subtlety with recommendations is that in most applications we cannot evaluate their performance directly. For example absent any additional interrogation of Customer 23 the only indication of how much they agree with one our recommendations is how well they rate the movie in the future.

Fortunately we can use our inferences to predict not only movie recommendations but also how we think Customer 23 would rate them. This would allow us to for example compare predicted rankings to actual rankings.

```
movie_idx <- tail(unrated_movie_idxs[post_mean_ordering], 1)

logistic <- function(x) {
  if (x > 0) {
    1 / (1 + exp(-x))
  } else {
    e <- exp(x)
    e / (1 + e)
  }
}

expectands <- list(function(c, gamma) 1 - logistic(gamma - c[1]),
                   function(c, gamma) logistic(gamma - c[1]) -
                                      logistic(gamma - c[2]),
```

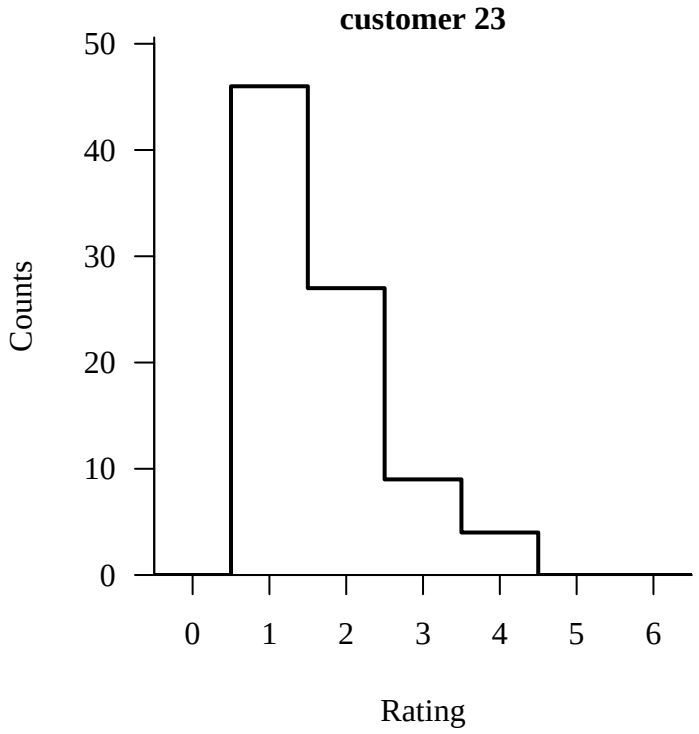
```
                    function(c, gamma) logistic(gamma - c[2]) -
                                       logistic(gamma - c[3]),
                    function(c, gamma) logistic(gamma - c[3]) -
                                       logistic(gamma - c[4]),
                    function(c, gamma) logistic(gamma - c[4]))
names(expectands) <- c('p[1]', 'p[2]', 'p[3]', 'p[4]', 'p[5]')

var_repl <- list('c'=array(sapply(1:4,
                           function(k)
                           paste0('cut_points[', c, ',', k, ']'))),
                 'gamma'=paste0('gamma[', c, ',', movie_idx, ']'))

for (k in 1:4) {
  name <- paste0('cut_points[', c, ',', k, ']')
  affinity_samples[[name]] <- samples4[[name]]
}

prob_samples <-util$eval_expectand_pushforwards(affinity_samples,
                                                expectands,
                                                var_repl)
```

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_disc_pushforward_quantiles(prob_samples, names(expectands),
                                     xlab="Rating",
                                     ylab="Posterior Probability")
```



Interestingly we don't actually predict a particularly high rating for our top recommendation. In hindsight, however, this shouldn't be unexpected given how austere Customer 23 is with their stars.

```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

util$plot_line_hist(data$ratings[data$customer_idxs == c],
                    -0.5, 6.5, 1,
                    xlab="Rating", main=paste('customer', c))
```



Another benefit of this hierarchical approach is that we are not limited to making inferences and predictions for existing customers. In particular we can also make inferences and predictions for new customers by sampling new interior cut points and movie affinities from the respective hierarchical population models. With the dearth of observed ratings these predictions won't be all that precise, but at the same time that uncertainty prevents us from making overly confident claims.

7 Computational Considerations

I want to emphasize that this case study is first and foremost a demonstrative analysis. In particular I reduced the data not for any statistical reason by rather to ensure that the models would not take too long to run. Ultimately the final model took about three hours to run on my laptop which wasn't too onerous, especially given the total number of parameters.

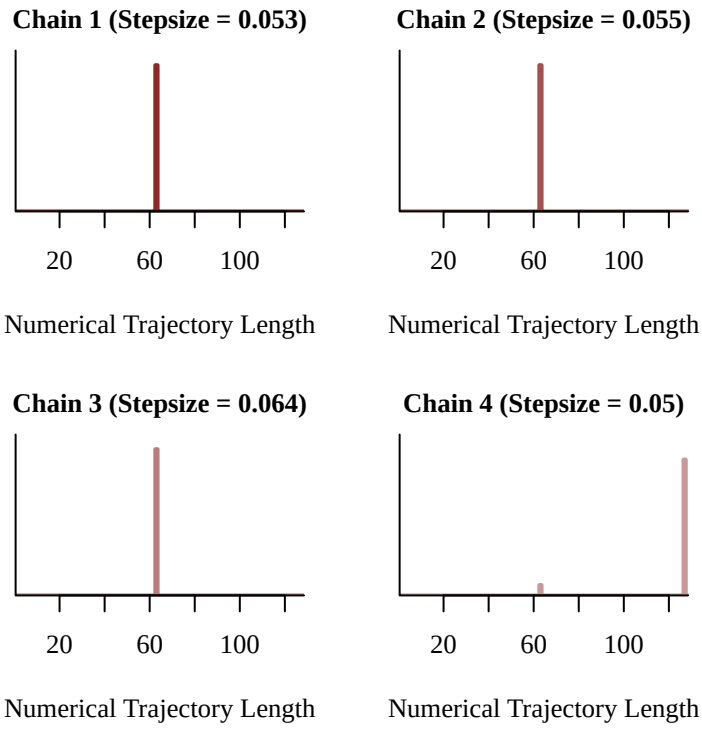
That said I do think it is useful to at least consider what the different priorities might be for a more realistic analysis where a specific inferential goal would be driving the amount of data to include and different computational resources might be available. What *would* it take to speed up the fit of the final model or scale it up to a larger data set?

Recall that the overall cost of running Hamiltonian Monte Carlo can roughly be decomposed into the number of iterations, the number of model evaluations per iteration, and the cost

of each model evaluation. The number of model evaluations per iteration is driven by the posterior geometry and how hard the Hamiltonian Monte Carlo algorithm has to work to explore it. For a fixed data set the two main ways that we can reduce computation is to improve the posterior geometry or speed up the model evaluations.

When working with hierarchical models we need to be considerate of the potentially problematic geometries to which they are prone. In this case study we seemed to do okay with a monolithic non-centered parameterization for the normal and multivariate normal hierarchies, but we could possibly improve the geometry by non-centering the parameters corresponding to more prolific movies and customers. Before considering that, however, we can estimate the potential for improvement by examining the length of the numerical Hamiltonian trajectories, and hence how many model evaluations were needed per iteration, in our last fit.

```
util$plot_num_leapfrogs_by_chain(diagnostics)
```



Despite the complexity of the final model the numerical Hamiltonian trajectories weren't all that long. Even in an ideal case we can't do do much better than ten or so leapfrog steps per trajectory; consequently the maximum possible speed up that we could get from improving the geometry here would be less than an order of magnitude! That's not trivial but it suggests that the computational cost is not being dominated by the number of model evaluations but rather the cost of each model evaluation itself.

So how can we speed up the model evaluations? One immediate strategy is parallelization, especially if we're working with computers blessed with lots and lots of threads. For example we can, at least in theory, parallelize the many matrix-vector products that are required in the `transformed parameters` block. That said achieving these potential speed ups in practice is always frustrated by the subtle input/output costs of passing all of the needed information to each thread and back in each model evaluation.

Either approach to speeding up the fitting of the final model will be challenging, requiring careful investigations and implementations and offering no guarantee of success. Even worse neither of these strategies will really be able to compete with the quadratic cost of evaluating the model, both in terms of $N_{\text{customer}} \cdot N_{\text{movie}}$ and N_{movie}^2, if we attempt to add more customers and/or movies. For example scaling up from 200 movies to 2000 movies, still only a fraction of the total data set and an even more negligible part of the full data a company like Netflix would have available, would require a 100 fold increase in the cost of evaluating the model. Even if the posterior geometry doesn't get any worst that pushes three hours to over a full day of computation.

In practice we can fight quadratic scaling only so far. Ultimately the problem is that the final model always has to compare every movie to every other movie. Consequently the most effective scaling strategy is often to limit the number of movies to which each movie is compared. More formally we need to introduce an appropriate sparsity structure on the movies so that most of the N_{movie}^2 covariances are zero.

Many methods attempt to learn a sparsity structure consistent with the observed data, dynamically turning off covariances that end up too small. This, however, is an outrageously difficult learning problem and approximate results tend to be fragile without unreasonable amounts of data. We can usually do much better by taking advantage of our domain expertise to motivate appropriate sparsity structures directly.

For example we could first group movies into genres before modeling common baseline affinities, correlated deviations across genres, and perhaps even correlated deviations for each movie within each genre. This effectively introduces a block-diagonal structure to the full covariance matrix which scales much more efficiently without sacrificing *all* of the correlations that can help inform predictions for unrated movies.

Given the sparsity of the observed ratings we can learn only so much. Consequently we might as well build a more restricted model of meaningful behaviors that we have a hope of resolving than attempting to learn details about which we just don't have enough information.

8 Conclusion

In this case study I developed a relatively sophisticated analysis of consumer feedback that accounts for not only how each customer interprets the possible ordinal ratings in different ways but also the variation in their preferences. To learn anything about these behaviors in

spite of the sparsity of the data we had to leverage our domain expertise and some formidable modeling techniques.

If anything I hope that this case study demonstrates how powerful hierarchical modeling techniques can be when used carefully. In particular to achieve our final inferences we used our domain expertise to sketch out the data generating process first, and only then considered opportunities for heterogeneous behaviors.

By starting with the broad features of the data generating process we established an explicit context that made is easier to identify not only what behaviors were heterogeneous but also what heterogeneous behaviors might be coupled together. Moreover the structure of those behaviors motivated appropriate population models. In this way we were able to develop an elaborate model with multiple, multivariate hierarchies without becoming to overwhelmed in the process.

Hierarchical modeling is so much more than one-dimensional normal population models!

Acknowledgements

I thank jd for helpful comments.

A very special thanks to everyone supporting me on Patreon: Adam Fleischhacker, Adriano Yoshino, Alejandro Navarro-Martínez, Alessandro Varacca, Alex D, Alexander Noll, Andrea Serafino, Andrew Mascioli, Andrew Rouillard, Andrew Vigotsky, Ara Winter, Austin Rochford, Avraham Adler, Ben Matthews, Ben Swallow, Benoit Essiambre, Bertrand Wilden, boot, Bradley Kolb, Brendan Galdo, Bryan Chang, Brynjolfur Gauti Jónsson, Cameron Smith, Canaan Breiss, Cat Shark, CG, Charles Naylor, Chase Dwelle, Chris Jones, Christopher Mehrvarzi, Colin Carroll, Colin McAuliffe, Damien Mannion, dan mackinlay, Dan W Joyce, Dan Waxman, Dan Weitzenfeld, Daniel Edward Marthaler, Daniel Saunders, Darshan Pandit, Darthmaluus , David Galley, David Wurtz, Doug Rivers, Dr. Jobo, Dr. Omri Har Shemesh, Dylan Maher, Ed Cashin, Edgar Merkle, Eli Witus, Eric LaMotte, Ero Carrera, Eugene O'Friel, Felipe González, Fergus Chadwick, Finn Lindgren, Francesco Corona, Geoff Rollins, Guilherme Marthe, Håkan Johansson, Hamed Bastan-Hagh, haubur, Hector Munoz, Henri Wallen, hs, Hugo Botha, Ian, Ian Costley, idontgetoutmuch, Ignacio Vera, Ilaria Prosdocimi, Isaac Vock, Isidor Belic, jacob pine, Jair Andrade, James C, James Hodgson, James Wade, Janek Berger, Jarrett Byrnes, Jason Martin, Jason Pekos, Jason Wong, jd, Jeff Burnett, Jeff Dotson, Jeff Helzner, Jeffrey Erlich, Jerry Lin , Jessica Graves, Joe Sloan, John Flournoy, Jonathan H. Morgan, Jonathon Vallejo, Joran Jongerling, Josh Knecht, June, Justin Bois, Kádár András, Karim Naguib, Karim Osman, Kristian Gårdhus Wichmann, Lars Barquist, lizzie , LOU ODETTE, Luís F, Marcel Lüthi, Marek Kwiatkowski, Mariana Carmona, Mark Donoghoe, Markus P., Márton Vaitkus, Matthew, Matthew Kay, Matthew Mulvahill, Matthieu LEROY, Mattia Arsendi, Matěj Kolouch Grabovský, Maurits van der Meer, Max, Michael Colaresi,

Michael DeWitt, Michael Dillon, Michael Lerner, Mick Cooney, Mike Lawrence, MisterMentat , N Sanders, N.S. , Name, Nathaniel Burbank, Nic Fishman, Nicholas Clark, Nicholas Cowie, Nick S, Ole Rogeberg, Oliver Crook, Olivier Ma, Patrick Kelley, Patrick Boehnke, Pau Pereira Batlle, Pete St. Marie, Peter Johnson, Pieter van den Berg , ptr, quasar, Ramiro Barrantes Reynolds, Raúl Peralta Lozada, Ravin Kumar, Rémi , Rex Ha, Riccardo Fusaroli, Richard Nerland, Robert Frost, Robert Goldman, Robert kohn, Robin Taylor, Ryan Gan, Ryan Grossman, Ryan Kelly, S Hong, Sean Wilson, Sergiy Protsiv, Seth Axen, shira, Simon Duane, Simon Lilburn, Simone Sebben, sssz, Stefan Lorenz, Stephen Lienhard, Steve Harris, Stew Watts, Stone Chen, Susan Holmes, Svilup, Tao Ye, Tate Tunstall, Tatsuo Okubo, Teresa Ortiz, Theodore Dasher, Thomas Siegert, Thomas Vladeck, Tobychev , Tony Wuersch, Tyler Burch, Virginia Fisher, Vladimir Markov, Wil Yegelwel, Will Farr, Will Lowe, Will Wen, woejozney, yolhaj , yureq , Zach A, Zad Rafi, and Zhengchen Cai.

License

A repository containing all of the files used to generate this chapter is available on GitHub.

Original Computing Environment

```
writeLines(readLines(file.path(Sys.getenv("HOME"), ".R/Makevars")))
```

```
CC=clang

CXXFLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-macr
CXX=clang++ -arch x86_64 -ftemplate-depth-256

CXX14FLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-ma
CXX14=clang++ -arch x86_64 -ftemplate-depth-256
```

```
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.4.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] colormap_0.1.4     rstan_2.32.6       StanHeaders_2.32.7

loaded via a namespace (and not attached):
 [1] gtable_0.3.4       jsonlite_1.8.8     compiler_4.3.2     Rcpp_1.0.11
 [5] parallel_4.3.2     gridExtra_2.3      scales_1.3.0       yaml_2.3.8
 [9] fastmap_1.1.1      ggplot2_3.4.4      R6_2.5.1           curl_5.2.0
[13] knitr_1.45         tibble_3.2.1       munsell_0.5.0      pillar_1.9.0
[17] rlang_1.1.2        utf8_1.2.4         V8_4.4.1           inline_0.3.19
[21] xfun_0.41          RcppParallel_5.1.7 cli_3.6.2          magrittr_2.0.3
[25] digest_0.6.33      grid_4.3.2         lifecycle_1.0.4    vctrs_0.6.5
[29] evaluate_0.23      glue_1.6.2         QuickJSR_1.0.8     codetools_0.2-19
[33] stats4_4.3.2       pkgbuild_1.4.3     fansi_1.0.6        colorspace_2.1-0
[37] rmarkdown_2.25     matrixStats_1.2.0  tools_4.3.2        loo_2.6.0
[41] pkgconfig_2.0.3    htmltools_0.5.7
```

Stan
Program 1 `model1.stan`

```
functions {
  // Log probability density function over cut point
  // induced by a Dirichlet probability density function
  // over baseline probabilities and latent logistic
  // density function.
  real induced_dirichlet_lpdf(vector c, vector alpha) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);

    // Log Jacobian correction
    real logJ = 0;
    for (k in 1:(K - 1)) {
      if (c[k] >= 0)
        logJ += -c[k] - 2 * log(1 + exp(-c[k]));
      else
        logJ += +c[k] - 2 * log(1 + exp(+c[k]));
    }

    return dirichlet_lpdf(p | alpha) + logJ;
  }
}

data {
  int<lower=1> N_ratings;
  array[N_ratings] int<lower=1, upper=5> ratings;

  int<lower=1> N_customers;
  array[N_ratings] int<lower=1, upper=N_customers> customer_idxs;

  int<lower=1> N_movies;
  array[N_ratings] int<lower=1, upper=N_movies> movie_idxs;
}

parameters {
  vector[N_movies] gamma_ncp; // Non-centered movie affinities
  real<lower=0> tau_gamma;    // Movie affinity population scale

  ordered[4] cut_points;      // Customer rating cut points
}

transformed parameters {
  // Centered movie affinities
  vector[N_movies] gamma = tau_gamma * gamma_ncp;
}

model {
  // Prior model
  gamma_ncp ~ normal(0, 1);
  tau_gamma ~ normal(0, 5 / 2.57);
```

Stan
Program 2 `model2.stan`

```
functions {
  // Log probability density function over cut point
  // induced by a Dirichlet probability density function
  // over baseline probabilities and latent logistic
  // density function.
  real induced_dirichlet_lpdf(vector c, vector alpha) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);

    // Log Jacobian correction
    real logJ = 0;
    for (k in 1:(K - 1)) {
      if (c[k] >= 0)
        logJ += -c[k] - 2 * log(1 + exp(-c[k]));
      else
        logJ += +c[k] - 2 * log(1 + exp(+c[k]));
    }

    return dirichlet_lpdf(p | alpha) + logJ;
  }
}

data {
  int<lower=1> N_ratings;
  array[N_ratings] int<lower=1, upper=5> ratings;

  int<lower=1> N_customers;
  array[N_ratings] int<lower=1, upper=N_customers> customer_idxs;

  int<lower=1> N_movies;
  array[N_ratings] int<lower=1, upper=N_movies> movie_idxs;
}

parameters {
  vector[N_movies] gamma_ncp; // Non-centered movie qualities
  real<lower=0> tau_gamma;    // Movie quality population scale

  array[N_customers] ordered[4] cut_points; // Customer rating cut points
}

transformed parameters {
  vector[N_movies] gamma = tau_gamma * gamma_ncp;
}

model {
  // Prior model
  gamma_ncp ~ normal(0, 1);
  tau_gamma ~ normal(0, 5 / 2.57);
```

Stan
Program 3 `model3.stan`

```
functions {
  // Log probability density function over cut point
  // induced by a Dirichlet probability density function
  // over baseline probabilities and latent logistic
  // density function.
  real induced_dirichlet_lpdf(vector c, vector alpha) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);

    // Log Jacobian correction
    real logJ = 0;
    for (k in 1:(K - 1)) {
      if (c[k] >= 0)
        logJ += -c[k] - 2 * log(1 + exp(-c[k]));
      else
        logJ += +c[k] - 2 * log(1 + exp(+c[k]));
    }

    return dirichlet_lpdf(p | alpha) + logJ;
  }
}

data {
  int<lower=1> N_ratings;
  array[N_ratings] int<lower=1, upper=5> ratings;

  int<lower=1> N_customers;
  array[N_ratings] int<lower=1, upper=N_customers> customer_idxs;

  int<lower=1> N_movies;
  array[N_ratings] int<lower=1, upper=N_movies> movie_idxs;
}

parameters {
  vector[N_movies] gamma_ncp; // Non-centered movie affinities
  real<lower=0> tau_gamma;    // Movie affinity population scale

  array[N_customers] ordered[4] cut_points; // Customer rating cut points

  simplex[5] mu_q;     // Rating simplex population location
  real<lower=0> tau_q; // Rating simplex population scale
}
```



```
transformed parameters {
  // Centered movie affinities
  vector[N_movies] gamma = tau_gamma * gamma_ncp;
}

model {
```

Stan
Program 4 `model4.stan`

```
functions {
  // Log probability density function over cut point
  // induced by a Dirichlet probability density function
  // over baseline probabilities and latent logistic
  // density function.
  real induced_dirichlet_lpdf(vector c, vector alpha) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);

    // Log Jacobian correction
    real logJ = 0;
    for (k in 1:(K - 1)) {
      if (c[k] >= 0)
        logJ += -c[k] - 2 * log(1 + exp(-c[k]));
      else
        logJ += +c[k] - 2 * log(1 + exp(+c[k]));
    }

    return dirichlet_lpdf(p | alpha) + logJ;
  }
}
data {
  int<lower=1> N_ratings;
  array[N_ratings] int<lower=1, upper=5> ratings;

  int<lower=1> N_customers;
  array[N_ratings] int<lower=1, upper=N_customers> customer_idxs;

  int<lower=1> N_movies;
  array[N_ratings] int<lower=1, upper=N_movies> movie_idxs;
}

parameters {
  // Baseline movie affinities population model
  vector[N_movies] gamma0_ncp;
  real<lower=0> tau_gamma0;

  // Individual customer affinity population model
  array[N_customers] vector[N_movies] delta_gamma_ncp;
  vector<lower=0>[N_movies] tau_delta_gamma;
  cholesky_factor_corr[N_movies] L_delta_gamma;

  array[N_customers] ordered[4] cut_points; // Customer rating cut points

  simplex[5] mu_q;     // Rating simplex population location
  real<lower=0> tau_q; // Rating simplex population scale
}

transformed parameters {
```