Consensus Modeling

Michael Betancourt

November 2023

Table of contents

A common problem in many fields is determining an unknown state from multiple, independent assessments. For example we might be interested in determining the status of a patient based on diagnoses of multiple doctors, or the market category of a product based on multiple consumer surveys. The only to combine the individual assessments into a consistent answer, however, is to quantify how accurate the assessments are.

In this chapter we'll review the general structure of consensus models before narrowing our focus to discrete consensus models.

1 General Consensus Models

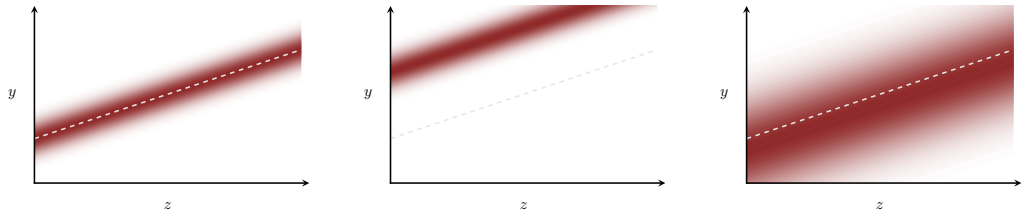
The generative structure of consensus models is relatively straightforward. Instead of trying to invert individual assessments into an consensus estimate of the latent state the generative structure is determined by how the assessments are informed by the unknown latent state. In other words we have to define the *conditional* relationship between the assessments and the latent state.

More formally let's say that we have R reviewers, each of whom provides an assessment $y_r \in Y$ of some unknown state $z \in Z$. The assessments are modeled with a conditional probability distribution

$$\pi(y_r \mid z, \phi_j).$$

I will refer to $\pi(y_r \mid z, \phi_r)$ as the **fidelity model** for each reviewer.

The more strongly y_r is coupled to z the more accurately the rth reviewer's assessments will inform the underlying state (Figure Figure 1). Note that we don't need y_r to concentrate around z to learn about the latent state. If the distribution of y_r is biased away from z in a predictable way then we be able to correct any observed assessments to make accurate inferences of the latent state.



(a) (b) (c)

Figure 1: A fidelity model quantifies the relationship between reviewer assessments y and the true latent state z. (a) In a precise and accurate fidelity model the assessments concentrate around z, making it straightforward to infer the true latent state. (b) Precise but biased fidelity models correspond to assessments that concentrate away from z. If we can infer this bias, however, we can still recover the true latent state. (c) An imprecise fidelity model features little coupling between the assessments and the latent state, making it difficult if not impossible to infer the latent state.

Combining the reviewer fidelity models with a model $\pi(z \mid \psi)$ for the latent state defines a joint observational model over the latent state and the individual assessments,

$$\begin{aligned} \pi(y_1, \ldots, y_R, z \mid \phi_1, \ldots \phi_R, \psi) &= \pi(y_1, \ldots, y_J \mid z, \phi_1, \ldots \phi_R) \, \pi(z \mid \psi) \\ &= \left[\prod_{r=1}^{R} \pi(y_r \mid z, \phi_r) \right] \, \pi(z \mid \psi). \end{aligned}$$

To ease the notation we can also write

$$\mathbf{y} = (y_1, \ldots, y_R)$$
$$\phi = (\phi_1, \ldots, \phi_R)$$

with

$$\pi(\mathbf{y}, z \mid \phi, \psi) = \left[\prod_{r=1}^{R} \pi(y_r \mid z, \phi_r)\right] \pi(z \mid \psi).$$

If the fidelity and latent state model configurations are fixed then a collection of observed assessments,

$$\{\tilde{y}_1, \ldots, \tilde{y}_R\},$$

allows us to infer an unknown latent state,

$$\begin{aligned}\pi(z \mid \tilde{\mathbf{y}}, \phi, \psi) &\propto \pi(\mathbf{y}, z \mid \phi, \psi) \\ &\propto \left[\prod_{r=1}^{R} \pi(\tilde{y}_r \mid z, \phi_r)\right] \pi(z \mid \psi).\end{aligned}$$

Conceptually the reviewer consensus is given by the values of z where the individual likelihood function contributions $\pi(\tilde{y}_r \mid z, \phi_r)$ most strongly overlap (Figure Figure 2).

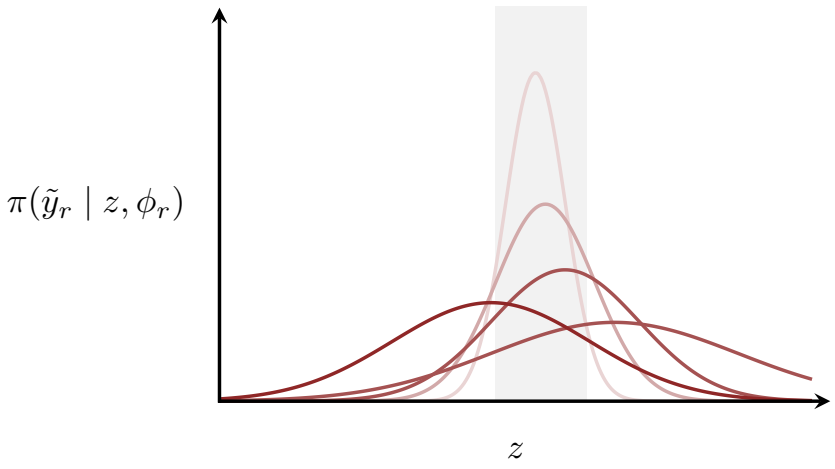


Figure 2: Evaluating the fidelity model for each reviewer on an observed assessment gives a contribution to the likelihood function, $\pi(\tilde{y}_r \mid z, \phi_r)$. The overlap of these contributions defines the consensus of the reviewers.

When the fidelity and latent state model configurations are not known but we have both a collection of observed assessments,

$$\{\tilde{y}_1, \ldots, \tilde{y}_R\},$$

and a known latent state $\tilde{z}$ the consensus model allows us to infer these unknown parameters,

$$\begin{aligned}
\pi(\phi, \psi \mid \tilde{z}, \tilde{\mathbf{y}}) &\propto \pi(\tilde{\mathbf{y}}, \tilde{z}, \phi, \psi) \\
&\propto \pi(\tilde{\mathbf{y}}, \tilde{z} \mid \phi, \psi) \, \pi(\phi, \psi) \\
&\propto \left[\prod_{r=1}^{R} \pi(\tilde{y}_r \mid \tilde{z}, \phi_r) \right] \, \pi(z \mid \psi) \, \pi(\phi, \psi).
\end{aligned}$$

In other words comparing reviewer responses to a known truth allows us to *calibrate* the accuracy of their assessments.

In the worst case the fidelity and latent state model configurations will be unknown and we will have access to only the observed assessments. Here we have to infer the model configurations and the latent state at the same time,

$$\begin{aligned}
\pi(z, \phi, \psi \mid \tilde{\mathbf{y}}) &\propto \pi(\tilde{\mathbf{y}}, z, \phi, \psi) \\
&\propto \pi(\tilde{\mathbf{y}}, z \mid \phi, \psi) \, \pi(\phi, \psi) \\
&\propto \left[\prod_{r=1}^{R} \pi(\tilde{y}_r \mid z, \phi_r) \right] \, \pi(z \mid \psi) \, \pi(\phi, \psi).
\end{aligned}$$

Unfortunately these joint inferences are typically degenerate. For most consensus models the observed assessments can be made manipulated into consistent with *any* latent state by configuring the individual fidelity models appropriately.

Consider, for example, the $Y = Z = \mathbb{R}$ and the fidelity models

$$\pi(y_r | z, \mu, \tau_j) = \text{normal}(y_r | z + \mu, \tau_j).$$

Observed assessments $\tilde{y}_r$ will not directly inform the latent state but rather the additive combination $z + \mu$, and by adjusting μ the consensus can be shifted to any value of z.

In most applications we will not be considering a single assessment but rather a collection of N assessments. Not every reviewer has to participate in every assessment, but if they do then the notation for the joint model simplifies considerably,

$$\pi(\mathbf{y}_1, z_1, \ldots, \mathbf{y}_N, \phi, \psi) = \prod_{n=1}^{N} \prod_{r=1}^{R} \pi(y_{nr} \mid z, \phi_r) \, \pi(z_n \mid \psi) \pi(\phi, \psi).$$

In general these collections will have some combination of assessments with and without accompanying observations of the true latent state. The most complete observations

$$\{\tilde{\mathbf{y}}_n, \tilde{z}_n\}$$

the more precisely we can infer the model configurations ϕ and ψ and the more precisely we can infer unobserved latent states for the incomplete observations.

Without complete observations we will need to find alternative ways to suppress the degeneracies inherit to a particular consensus model. One strategy that is always available is the incorporation of as much domain expertise as possible in the prior model $\pi(\phi, \psi)$. This approach will be limited, however, by the domain expertise available for elicitation.

Another possibility is to integrate the consensus model with another narratively generative model that quantifies observable consequences of the latent state. In other words even if we can't observe the latent state directly any indirect constraints through auxiliary observations can help distinguish the reviewer fidelities from the true latent states.

2 Finite Consensus Models

In general the assessments and latent states can take values in any space. Often, however, the take values in finite spaces where any consensus model takes on a particular form.

First let's assume that the latent state z can take one of only K unordered values with the arbitrary labels $\{1, \ldots, k, \ldots, K\}$. In this case any latent state model reduces to a collection of K probabilities,

$$\pi(z = k, \psi) = \psi_k$$

with

$$0 \leq \psi_k \leq 1 \sum_{k=1}^{K} \psi_k = 1.$$

When the possible values are ordered then an ordinal model is better equipped to accommodate neighboring correlations.

Although mathematically straightforward a discrete latent state introduces a practical issue because we will not be able to use tools like Hamiltonian Monte Carlo to quantify posterior inferences. Fortunately we can work around this by marginalizing the discrete latent state out of the joint observational model,

$$\begin{aligned}
\pi(\mathbf{y} \mid \phi, \psi) &= \sum_{k=1}^{K} \pi(\mathbf{y}, k \mid \phi, \psi) \\
&= \sum_{k=1}^{K} \prod_{r=1}^{R} \pi(y_r \mid k, \phi_r) \, \pi(k \mid \psi) \\
&= \sum_{k=1}^{K} \prod_{r=1}^{R} \pi(y_r \mid k, \phi_r) \, \psi_k \\
&= \sum_{k=1}^{K} \psi_k \prod_{r=1}^{R} \pi(y_r \mid k, \phi_r).
\end{aligned}$$

In words the marginal observational model reduces to a mixture over the possible latent states weighted by the simplex ψ. Instead of trying to find a way to infer the discrete state z we can use tools like Hamiltonian Monte Carlo to efficiently infer the configuration of the continuous simplex ψ, and hence how consistent each latent state configuration is with the observed assessments.

When the assessments can take one of only J unordered values with the arbitrary labels $\{1, \ldots, j, \ldots, J\}$ then the fidelity models similarly reduce to the probabilities

$$\pi(y_r = j \mid k, \phi_r) = \phi_{r,jk}.$$

with

$$0 \leq \phi_{r,jk} \leq 1 \sum_{j=1}^{J} \gamma_{r,jk} = 1.$$

Once we restrict to finite, unordered spaces our models become simplices all the way down.

Mathematically the collection of simplices that defines the fidelity model for each reviewer can be organized into a matrix with one row for each possible assessment outcome and one column for each possible latent state configuration,

$$\Phi_r = \begin{pmatrix} \phi_{r,11} & \cdots & \phi_{r,1k} & \cdots \phi_{r,1K} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{r,j1} & \cdots & \phi_{r,jk} & \cdots \phi_{r,jK} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{r,J1} & \cdots & \phi_{r,Jk} & \cdots \phi_{r,JK} \end{pmatrix}.$$

A matrix of probabilities where the elements across each column sum to one like this is known as a **left stochastic matrix**, or perhaps more appropriately **left probability matrix**.

This finite consensus model manifests a serious degeneracy. Any permutation of the latent states, equivalently any relabeling of the unordered values, can be compensated by permuting the columns of every fidelity matrix without compromising the consistency with a given observation. In other words a likelihood function informed by only observed assessments, and not an observed latent state, will always manifest $K!$ separate modes, one for every one of the possible permutations!

Often reviewers assess the latent state directly, instead of assessing some indirectly consequence of the latent state. In this case we have $Y = Z$ and, in the finite setting, $J = K$ so that every fidelity matrix Φ_j will be square.

If we believe that the reviewers are able to identify the true latent state more often than not then we should have

$$\phi_{r,kk} > \phi_{r,kk'}$$

whenever $k \neq k'$. In other words accurate reviewers are modeled with diagonally dominant fidelity matrix.

Conveniently this constraint eliminates all but one of the peaks in the realized likelihood functions. Unfortunately enforcing this constraint in practice is not straightforward. Enforcing a hard constraint $\phi_{r,kk} > \phi_{r,kk'}$r couples the column simplices together, substantially complicating the viable fidelity matrix configurations. On the other hand we can enforce a soft constraint with an informative prior model over the column simplices, but this may not completely eliminate the undesired modes.

That said neither a hard nor soft constraint is appropriate if the reviewers are not necessarily expected to perform well. When our best inferences are given by doing the exact opposite of what a reviewer suggests then we'll need to consider other modes of the realized likelihood functions.

Finite consensus models with

$$Y = Z = \{1, \ldots, K\}$$

are commonly known as **Dawid-Skene** models after Dawid and Skene (1979), although the methodology was common earlier. See for example Landis and Koch (1977). The problem of estimating the agreement between multiple assessments has an even longer history; see for example Good and Card (1971), Landis and Koch (1975a) and Landis and Koch (1975b).

Regardless of the structure of the latent state space and the reviewer assessment space I prefer to use the more general term "consensus modeling" to emphasize the common construction shared by these models.

3 Demonstration

Let's see how all of these ideas are implemented in practice by drawing inferences from a finite consensus model.

3.1 Set Up

First we set 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))

c_light <- c("#DCBCBC")
c_light_highlight <- c("#C79999")
c_mid <- c("#B97C7C")
c_mid_highlight <- c("#A25050")
c_dark <- c("#8F2727")
```

```
c_dark_highlight <- c("#7C0000")

c_light_teal <- c("#6B8E8E")
c_mid_teal <- c("#487575")
c_dark_teal <- c("#1D4F4F")
```

Then we load `RStan` and my recommended diagnostic suite.

```
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('stan_utility_rstan.R', local=util)
```

3.2 Dirichlet Visualizations

Because we will be working with a finite consensus model we will be employing a lot of simplices, one for the latent state in each assessment and one for the responses of each reviewer given each of the possible latent states. The most convenient prior models over the space of simplices are given by the Dirichlet family of probability density functions.

```
lmult <- function(x, a) {
  if (a == 0) {
    (0)
  } else {
    (a * log(x))
  }
}

dirichlet_pdf <- function(p1, p2, p3, a1, a2, a3) {
  exp(  lmult(p1, a1 - 1) + lmult(p2, a2 - 1) + lmult(p3, a3 - 1)
      + lgamma(a1 + a2 + a3) - lgamma(a1) - lgamma(a2) - lgamma(a3) )
}
```

Conveniently we can completely visualize three-dimensional simplices, and density functions over the possible simplex configurations, in a two-dimensional plots that places the extreme states

$$(1, 0, 0), (0, 1, 0), (0, 0, 1)$$

at the corners of a triangle.

```
disc_colors <- c("#FFFFFF", "#DCBCBC", "#C79999", "#B97C7C",
                 "#A25050", "#8F2727", "#7C0000")
cont_colors <- colormap(colormap=disc_colors, nshades=100)

plot_dirichlet <- function(a1, a2, a3, label_cex=1, main_title=NA) {
  N <- 200
  C <- 1 / sqrt(3)
  xs <- seq(-C - 0.2, C + 0.2, (2 * C + 0.4) / (N - 1))
  ys <- seq(0 - 0.1, 1 + 0.1, 1.2 / (N - 1))
  zs <- matrix(0, nrow=N, ncol=N)

  for (n in 1:N) {
    for (m in 1:N) {
      p3 <- ys[m]
      p2 <- (1 + xs[n] / C - ys[m]) / 2
      p1 <- (1 - xs[n] / C - ys[m]) / 2

      if (p1 >= 0 & p2 >= 0 & p3 >= 0 & p1 + p2 + p3 <= 1) {
        zs[n, m] <- dirichlet_pdf(p1, p2, p3, a1, a2, a3)
      } else {
        zs[n, m] <- NA
      }
    }
  }

  if (is.na(main_title)) {
    par(mar = c(0, 0, 0, 0))
    image(xs, ys, zs, col=rev(cont_colors), axes=FALSE, ann=FALSE)
  } else {
    par(mar = c(0, 0, 1, 0))
    image(xs, ys, zs, col=rev(cont_colors), axes=FALSE, ann=FALSE)
    title(main_title)
  }

  lines( c(-C, 0), c(0, 1), lwd=3)
  lines( c(+C, 0), c(0, 1), lwd=3)
  lines( c(-C, +C), c(0, 0), lwd=3)

  text_delta <- 0.05
  text( 0, 1 + text_delta, "(0, 0, 1)", cex=label_cex)
```

```
  text(-C - text_delta, -text_delta, "(1, 0, 0)", cex=label_cex)
  text(+C + text_delta, -text_delta, "(0, 1, 0)", cex=label_cex)

  tick_delta <- 0.025
  lines( c(0, 0), c(0, tick_delta), lwd=3)
  text(0, 0 - text_delta, "(1/2, 1/2, 0)", cex=label_cex)

  lines( c(+C * 0.5, +C * 0.5 - tick_delta * 0.5 * sqrt(3)),
         c(0.5, 0.5 - tick_delta * 0.5), lwd=3)
  text(C * 0.5 + text_delta * 0.5 * sqrt(3) + 2.5 * text_delta,
       0.5 + text_delta * 0.5, "(0, 1/2, 1/2)", cex=label_cex)

  lines( c(-C * 0.5, -C * 0.5 + tick_delta * 0.5 * sqrt(3)),
         c(0.5, 0.5 - tick_delta * 0.5), lwd=3)
  text(-C * 0.5 - text_delta * 0.5 * sqrt(3) - 2.5 * text_delta,
       0.5 + text_delta * 0.5, "(1/2, 0, 1/2)", cex=label_cex)

  points(0, 1/3, col="white", pch=16, cex=1.5)
  points(0, 1/3, col="black", pch=16, cex=1)
  text(0, 1/3 - 1.5 * text_delta, "(1/3, 1/3, 1/3)", cex=label_cex)
}
```

One convenient way to interpret a Dirichlet probability density function is to factor the parameters $\alpha_1, \ldots, \alpha_K$ into a *template simplex* and a *concentration parameter*,

$$(\alpha_1, \ldots, \alpha_K) = \tau \cdot (\gamma_1, \ldots, \gamma_K),$$

with

$$\sum_{k=1}^{K} \gamma_k = 1.$$

If $\tau > 1$ then the implied probability distribution will concentrate around the simplex configuration $\gamma_1, \ldots, \gamma_K$. The larger τ is the stronger this concentration will be.

On the other hand if $\tau < 1$ then the implied probability distribution will be repelled from the template simplex, concentrating instead on the nearest simplex boundaries.

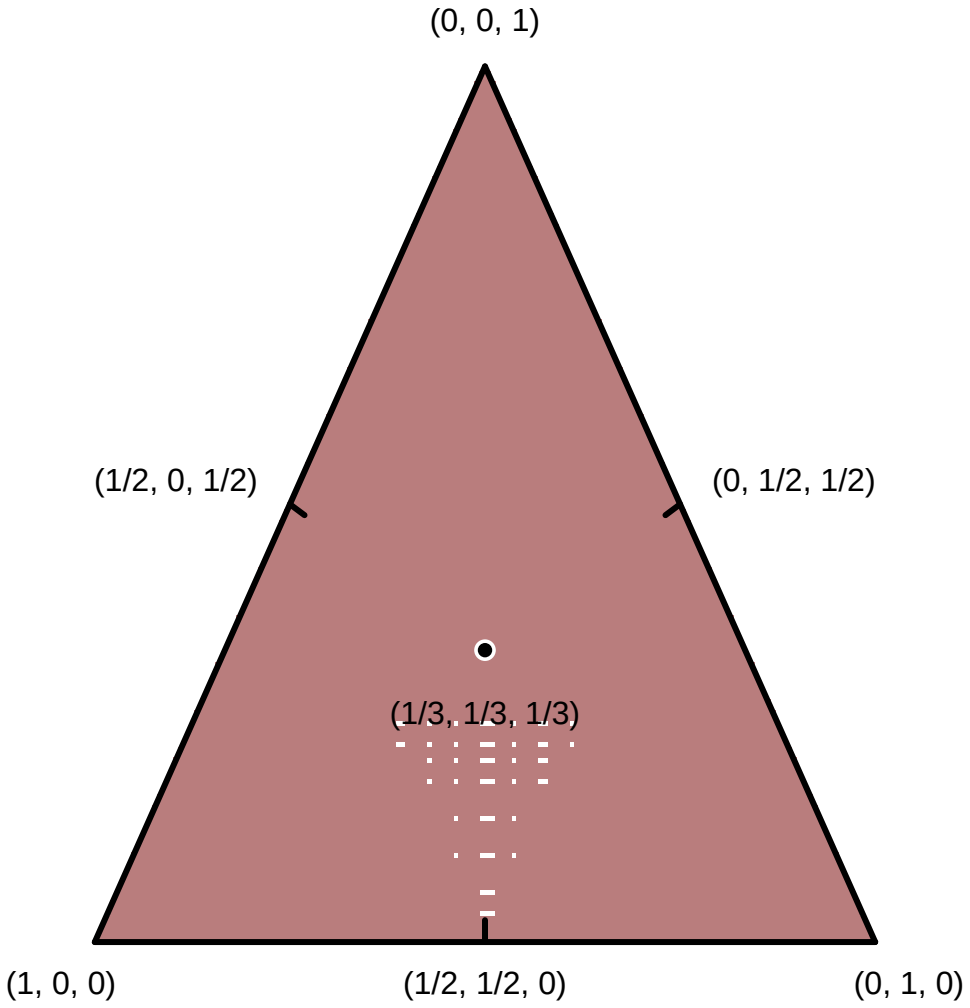
One exception is when any of the α_k are equal to one. In this case the Dirichlet probability density function will not depend on the corresponding simplex component at all.

For example taking

$$\alpha_1 = \ldots = \alpha_k = \ldots = \alpha_K = 1$$

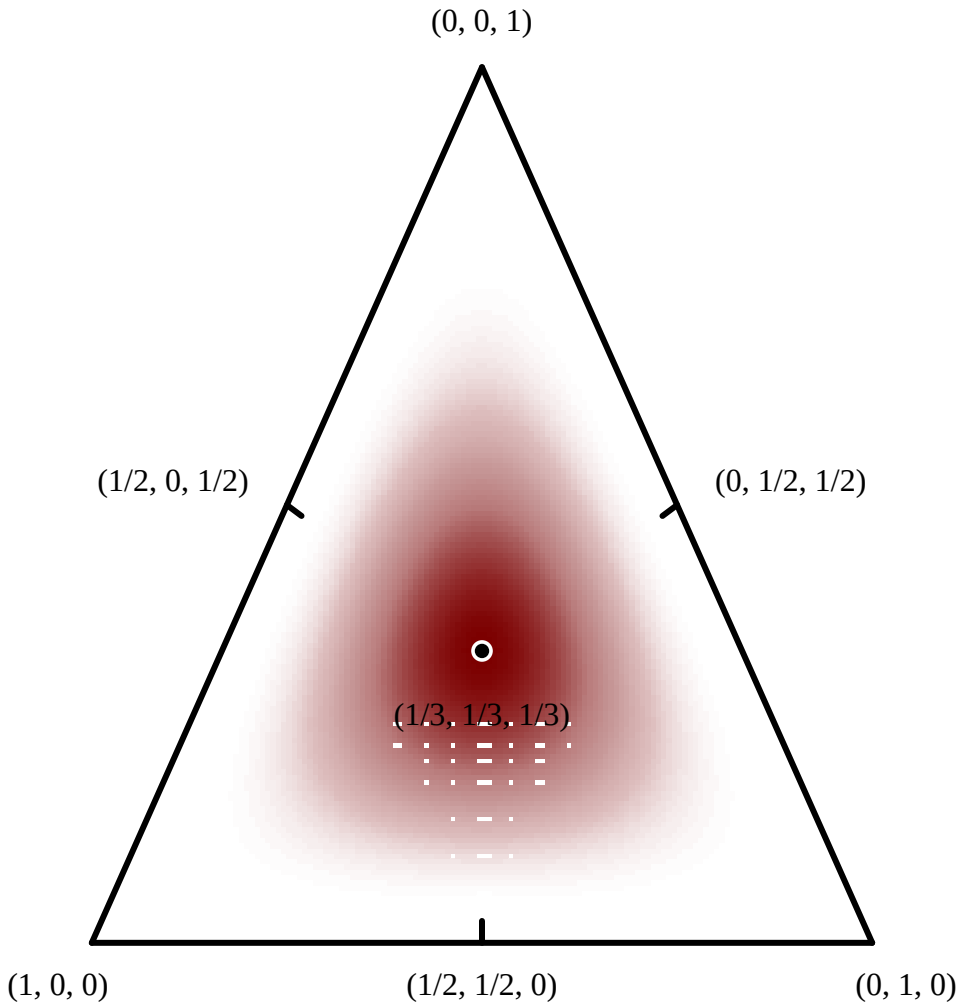
defines a uniform probability density function across the simplex configurations.

```
plot_dirichlet(1, 1, 1)
```



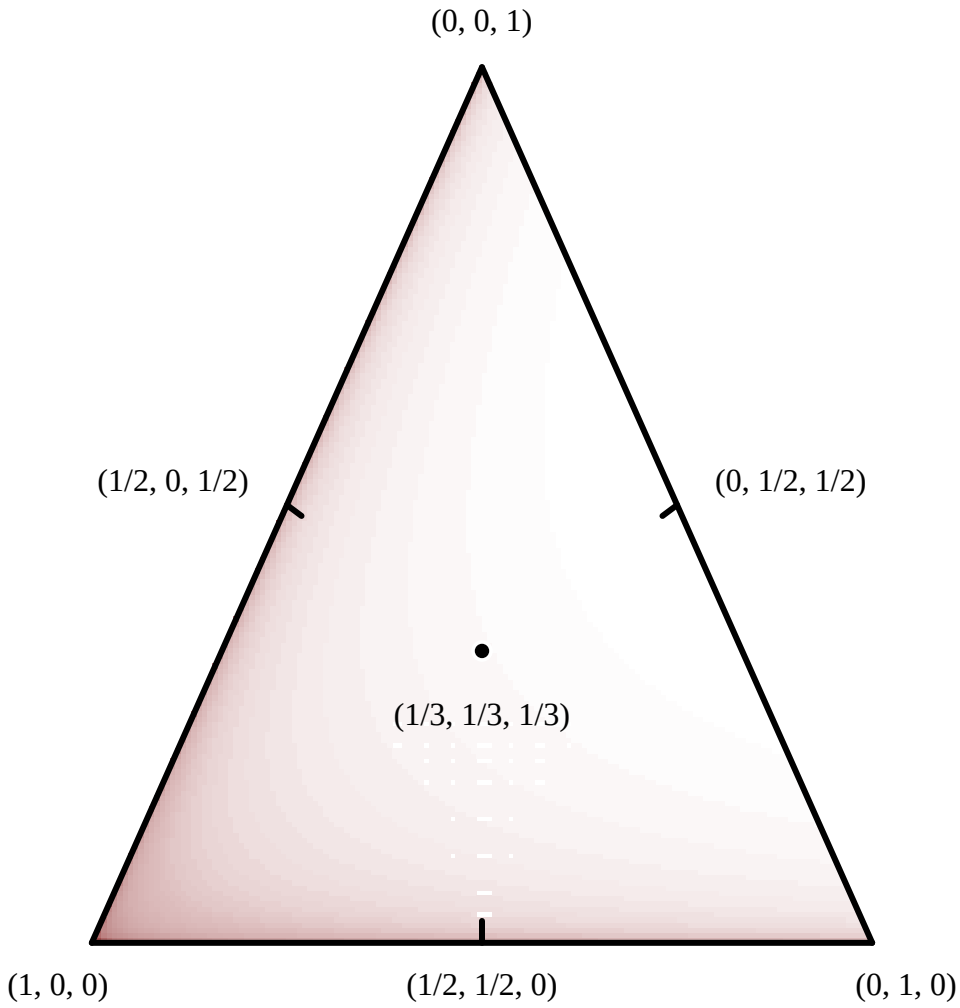
Increasing the parameters in the same way results in a probability density function that concentrates around the equal probability configuration.

```
plot_dirichlet(5, 5, 5)
```



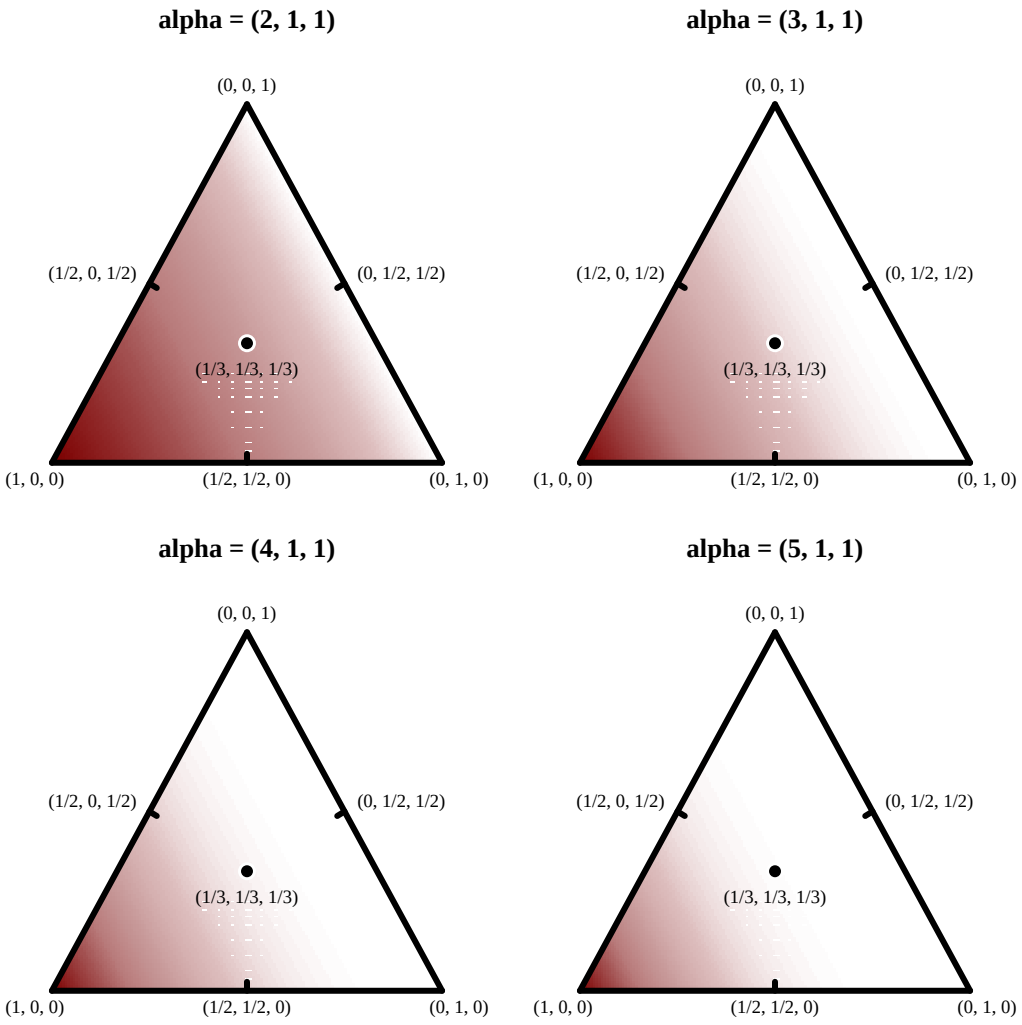
Often domain expertise prefers one simplex component over the others. Taking $\alpha_k = 1$ and $\alpha_{k' \neq k} < 1$ results in a probability density function that concentrates around the configuration where all probability is allocated to kth simplex component. As we move away from this corner the Dirichlet probability density function concentrates on the edges of the simplex.

```
plot_dirichlet(1, 0.8, 0.8)
```



If we instead take $\alpha_k > 1$ and $\alpha_{k' \neq k} = 1$ gives probability density functions that also concentrate around the same corner, but in this case the tails are uniform across the remaining components.

```
par(mfrow=c(2, 2))
plot_dirichlet(2, 1, 1, 0.75, "alpha = (2, 1, 1)")
plot_dirichlet(3, 1, 1, 0.75, "alpha = (3, 1, 1)")
plot_dirichlet(4, 1, 1, 0.75, "alpha = (4, 1, 1)")
plot_dirichlet(5, 1, 1, 0.75, "alpha = (5, 1, 1)")
```



These latter probability density functions are particularly well-suited to isolating a single mode from degenerate inferences common to finite consensus models.

3.3 Simulate Data

Because of the generative structure of the consensus model simulating data is straightforward. Note that these simulations will be somewhat optimistic, assuming that the reviewer fidelities strongly concentrate around the true latent states.

```
simu <- stan(file="stan_programs/simu.stan",
             iter=1, warmup=0, chains=1,
             seed=4838282, algorithm="Fixed_param")
```

```
SAMPLING FOR MODEL 'simu' NOW (CHAIN 1).
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                0.000199 seconds (Sampling)
Chain 1:                0.000199 seconds (Total)
Chain 1:
```

Let's isolate the first 100 assessments as our main data set.

```
data <- list('K' = 3, 'N_reviewers' = 5, 'N_assessments' = 100,
             'y' = extract(simu)$y[1,1:100,])

true_zs <- extract(simu)$z[1,1:100]
```

The last 10 assessments we'll reserve along with the true latent states as a separate calibration data set.

```
calib_data <- list('N_calib_assessments' = 10,
                   'y_calib' = extract(simu)$y[1,101:110,],
                   'z_calib' = extract(simu)$z[1,101:110])
```

3.4 Display Data

Before attempting a fit let's explore our data a bit.

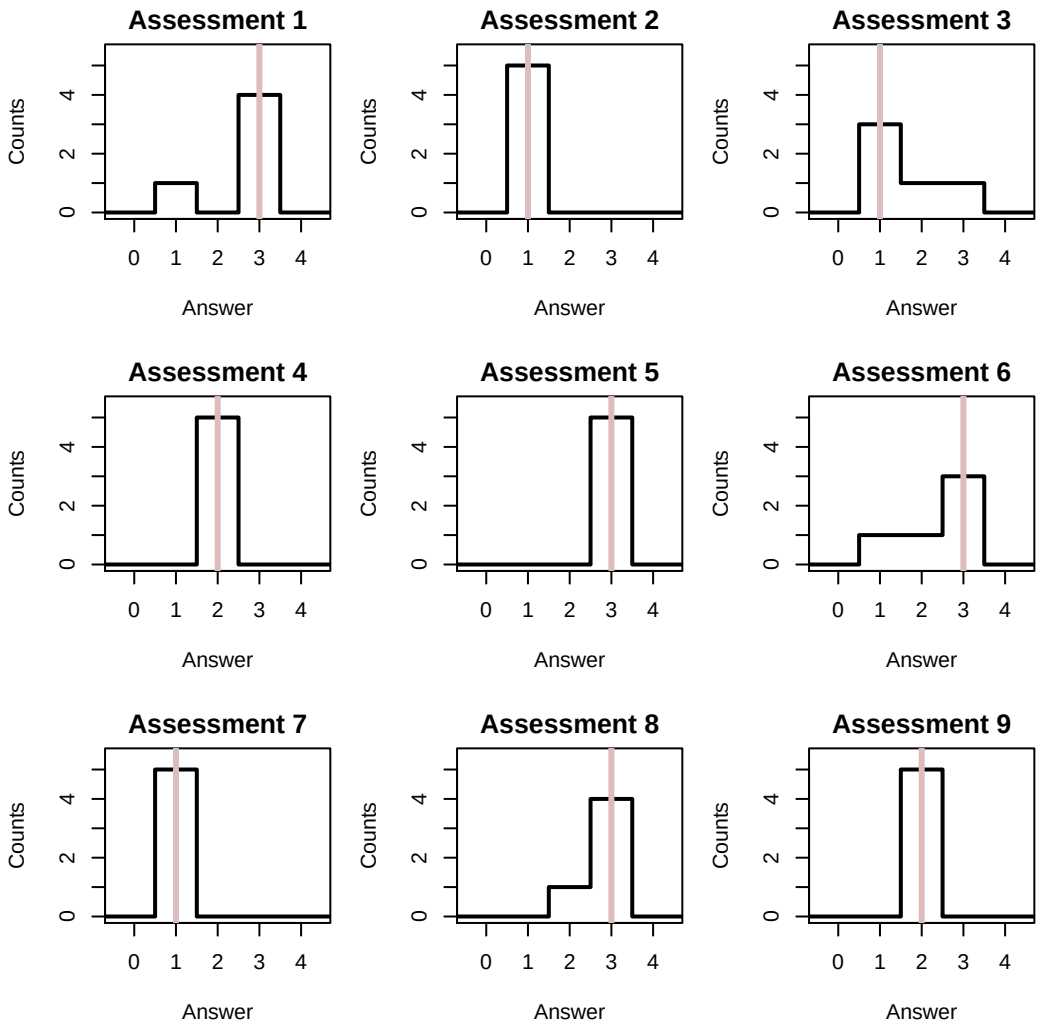
A natural summary for each assessment is a histogram of the reviewer responses.

```
plot_line_hist <- function(s, bin_min, bin_max, delta,
                           xlab, ylim=NA, main="") {
  bins <- seq(bin_min, bin_max, delta)
  B <- length(bins) - 1
  idx <- rep(1:B, each=2)
  x <- sapply(1:length(idx),
              function(b) if(b %% 2 == 1) bins[idx[b]]
                          else            bins[idx[b] + 1])
  x <- c(bin_min - 10, x, bin_max + 10)

  counts <- hist(s, breaks=bins, plot=FALSE)$counts
  y <- counts[idx]
  y <- c(0, y, 0)
```

```
  ymax <- max(y) + 1

  if (any(is.na(ylim))) {
    ylim <- c(0, ymax)
  }

  plot(x, y, type="l", main=main, col="black", lwd=2,
       xlab=xlab, xlim=c(bin_min, bin_max),
       ylab="Counts", ylim=ylim)
}
```

We won't look at every assessment here, but examining the first nine we can see that, while there is often disagreement, the most common reviewer response always seems to align with the true latent state.

```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))

for (n in 1:9) {
  plot_line_hist(data$y[n,], -0.5, data$K + 1.5, 1, "Answer",
                 c(0, 5.5), paste("Assessment", n))
  abline(v=true_zs[n], lwd=3, col=c_light)
}
```



This is consistent with the highly-optimistic accuracies encoded in our simulation.

3.5 Model 1

Let's begin with a lazy elicitation of our domain expertise, defaulting to uniform Dirichlet priors for all of the simplices in our model.

```
fit <- stan(file="stan_programs/fit1.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

The diagnostics are a torrent of split $\hat{R}$ issues for the reviewer fidelities parameters.

```
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.
```

```
samples <- util$extract_expectands(fit)
names <- grep('pred', names(samples), value=TRUE, invert=TRUE)
base_samples <- samples[names]
util$check_all_expectand_diagnostics(base_samples)
```

```
fidelity[1,1,1]:
  Split hat{R} (3.411) exceeds 1.1!

fidelity[2,1,1]:
  Split hat{R} (5.701) exceeds 1.1!

fidelity[3,1,1]:
  Split hat{R} (6.300) exceeds 1.1!

fidelity[4,1,1]:
  Split hat{R} (5.974) exceeds 1.1!

fidelity[5,1,1]:
  Split hat{R} (6.276) exceeds 1.1!

fidelity[2,2,1]:
  Split hat{R} (1.453) exceeds 1.1!

fidelity[4,2,1]:
  Split hat{R} (1.561) exceeds 1.1!

fidelity[5,2,1]:
  Split hat{R} (1.175) exceeds 1.1!

fidelity[1,3,1]:
  Split hat{R} (4.294) exceeds 1.1!

fidelity[2,3,1]:
```

```
  Split hat{R} (6.106) exceeds 1.1!

fidelity[3,3,1]:
  Split hat{R} (8.200) exceeds 1.1!

fidelity[4,3,1]:
  Split hat{R} (6.682) exceeds 1.1!

fidelity[5,3,1]:
  Split hat{R} (5.408) exceeds 1.1!

fidelity[1,1,2]:
  Split hat{R} (5.586) exceeds 1.1!

fidelity[2,1,2]:
  Split hat{R} (6.775) exceeds 1.1!

fidelity[3,1,2]:
  Split hat{R} (6.517) exceeds 1.1!

fidelity[4,1,2]:
  Split hat{R} (4.204) exceeds 1.1!

fidelity[5,1,2]:
  Split hat{R} (5.952) exceeds 1.1!

fidelity[1,2,2]:
  Split hat{R} (6.120) exceeds 1.1!

fidelity[2,2,2]:
  Split hat{R} (5.380) exceeds 1.1!

fidelity[3,2,2]:
  Split hat{R} (7.309) exceeds 1.1!

fidelity[4,2,2]:
  Split hat{R} (3.141) exceeds 1.1!

fidelity[5,2,2]:
  Split hat{R} (5.723) exceeds 1.1!

fidelity[1,3,2]:
  Split hat{R} (6.719) exceeds 1.1!
```

```
fidelity[2,3,2]:
  Split hat{R} (6.158) exceeds 1.1!

fidelity[3,3,2]:
  Split hat{R} (7.135) exceeds 1.1!

fidelity[4,3,2]:
  Split hat{R} (5.094) exceeds 1.1!

fidelity[5,3,2]:
  Split hat{R} (5.489) exceeds 1.1!

fidelity[1,1,3]:
  Split hat{R} (5.066) exceeds 1.1!

fidelity[2,1,3]:
  Split hat{R} (5.372) exceeds 1.1!

fidelity[3,1,3]:
  Split hat{R} (7.116) exceeds 1.1!

fidelity[4,1,3]:
  Split hat{R} (2.741) exceeds 1.1!

fidelity[5,1,3]:
  Split hat{R} (6.576) exceeds 1.1!

fidelity[1,2,3]:
  Split hat{R} (4.109) exceeds 1.1!

fidelity[2,2,3]:
  Split hat{R} (3.631) exceeds 1.1!

fidelity[3,2,3]:
  Split hat{R} (5.696) exceeds 1.1!

fidelity[4,2,3]:
  Split hat{R} (2.195) exceeds 1.1!

fidelity[5,2,3]:
  Split hat{R} (6.132) exceeds 1.1!
```

```
fidelity[1,3,3]:
  Split hat{R} (1.525) exceeds 1.1!

fidelity[3,3,3]:
  Split hat{R} (1.330) exceeds 1.1!

fidelity[4,3,3]:
  Split hat{R} (1.466) exceeds 1.1!


Split Rhat larger than 1.1 suggests that at least one of the Markov
chains has not reached an equilibrium.
```

Because we don't see accompanying empirical effective sample size warnings the most likely culprit is multimodality in the posterior distribution.

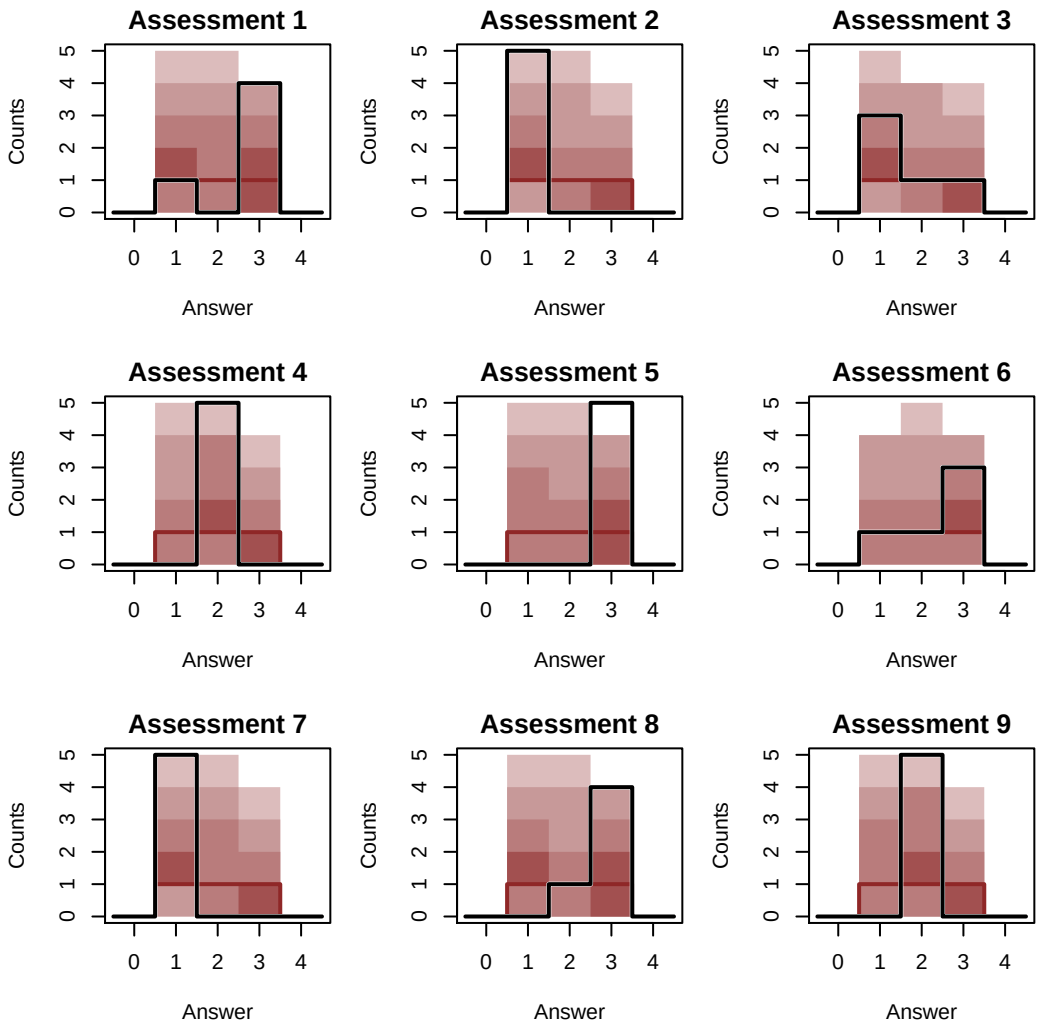
Before exploring for any multimodality, however, let's examine the posterior retrodicive performance. We'll use the histogram of reviewer responses for each assessment as our summary statistics.

```
hist_retro <- function(obs, samples, pred_names,
                       bin_min, bin_max, delta,
                       xlab="", ylim=NA, title="") {
  if (is.na(bin_min)) bin_min <- min(pred)
  if (is.na(bin_max)) bin_max <- max(pred)
  breaks <- seq(bin_min, bin_max, delta)
  B <- length(breaks) - 1

  idx <- rep(1:B, each=2)
  xs <- sapply(1:length(idx),
               function(b) if(b %% 2 == 0) breaks[idx[b] + 1]
                           else            breaks[idx[b]] )

  obs_counts <- hist(obs, breaks=breaks, plot=FALSE)$counts
  pad_obs_counts <- do.call(cbind,
                            lapply(idx, function(n) obs_counts[n]))

  pred <- sapply(pred_names,
                 function(name) c(t(samples[[name]]), recursive=TRUE))
  N <- dim(pred)[1]
  pred_counts <- sapply(1:N,
                        function(n) hist(pred[n,],
                                         breaks=breaks,
```

```
                                        plot=FALSE)$counts)

  probs = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
  cred <- sapply(1:B,
                 function(b) quantile(pred_counts[b,], probs=probs))
  pad_cred <- do.call(cbind, lapply(idx, function(n) cred[1:9, n]))

  if (any(is.na(ylim))) {
    ylim <- c(0, max(c(obs_counts, cred[9,])))
  }

  plot(1, type="n", main=title,
       xlim=c(bin_min, bin_max), xlab=xlab,
       ylim=ylim, ylab="Counts")

  polygon(c(xs, rev(xs)), c(pad_cred[1,], rev(pad_cred[9,])),
          col = c_light, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[2,], rev(pad_cred[8,])),
          col = c_light_highlight, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[3,], rev(pad_cred[7,])),
          col = c_mid, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[4,], rev(pad_cred[6,])),
          col = c_mid_highlight, border = NA)
  lines(xs, pad_cred[5,], col=c_dark, lwd=2)

  lines(xs, pad_obs_counts, col="white", lty=1, lw=2.5)
  lines(xs, pad_obs_counts, col="black", lty=1, lw=2)
}
```

Again we'll consider only the first nine assessments to simplify the presentation.

```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))

for (n in 1:9) {
  pred_names <- grep(paste0('y_pred\\[', n, ','),
                     names(samples), value=TRUE)
  hist_retro(data$y[n,], samples, pred_names, -0.5, data$K + 1.5, 1,
             "Answer", c(0, 5), paste("Assessment", n))
}
```

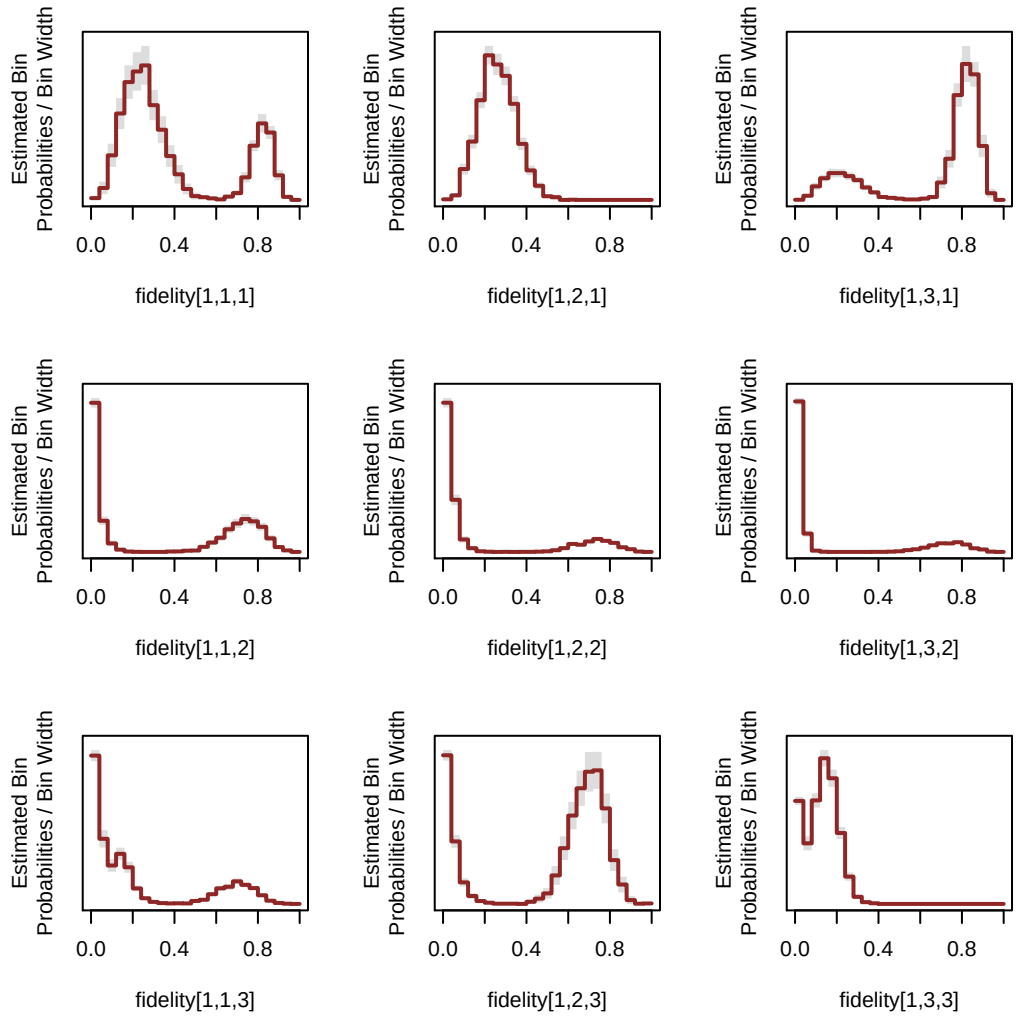


Overall there doesn't seem to be any retrodictive tension, but that is largely due to the lack of precision in the posterior retrodictive distribution.

Let's see if we can find the multimodality hinted at by the diagnostic warnings. Indeed looking at the fidelities of the first reviewer we see clear multimodalities.

```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))

r <- 1
for (k_answer in 1:3) {
  for (k_true in 1:3) {
    name <- paste0('fidelity[', r, ',', k_true, ',', k_answer, ']')
    util$plot_expectand_pushforward(samples[[name]], 25,
```

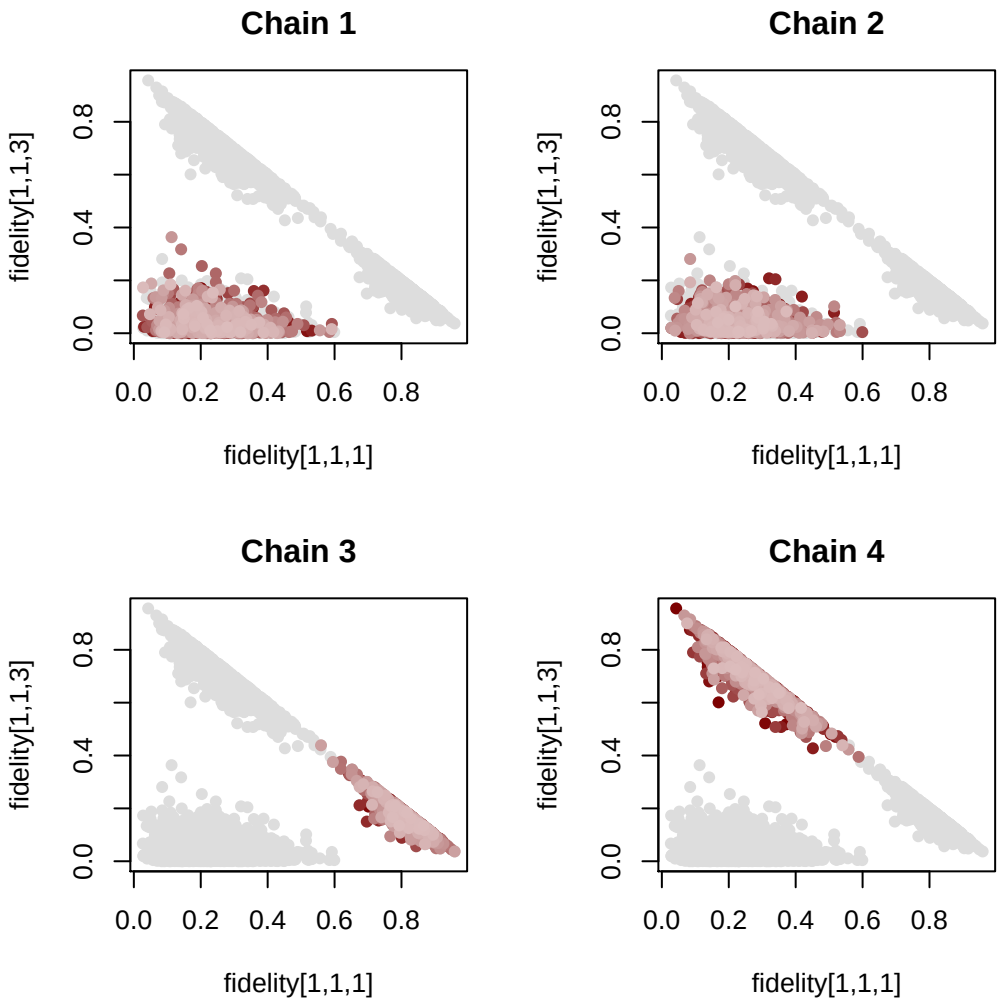
```
                                    name, flim=c(0, 1))
  }
}
```



As always we have to be careful interpreting the number and relative sizes of each of these modes. If there are many modes – here we'd expect $3! = 6$ – then the four Markov chains we ran may not have encountered them all. Moreover if each Markov chain has been captured by a particular mode then we won't have any meaningful information about the relative sizes of those modes.

To better understand the limitations of our initial posterior fit we can visualize the information from each Markov chain separately.

```
r <- 1
k_true <- 1
name1 <- paste0('fidelity[', r, ',', k_true, ',', 1, ']')
name2 <- paste0('fidelity[', r, ',', k_true, ',', 2, ']')
name3 <- paste0('fidelity[', r, ',', k_true, ',', 3, ']')

util$plot_pairs_by_chain(samples[[name1]], name1,
                         samples[[name3]], name3)
```



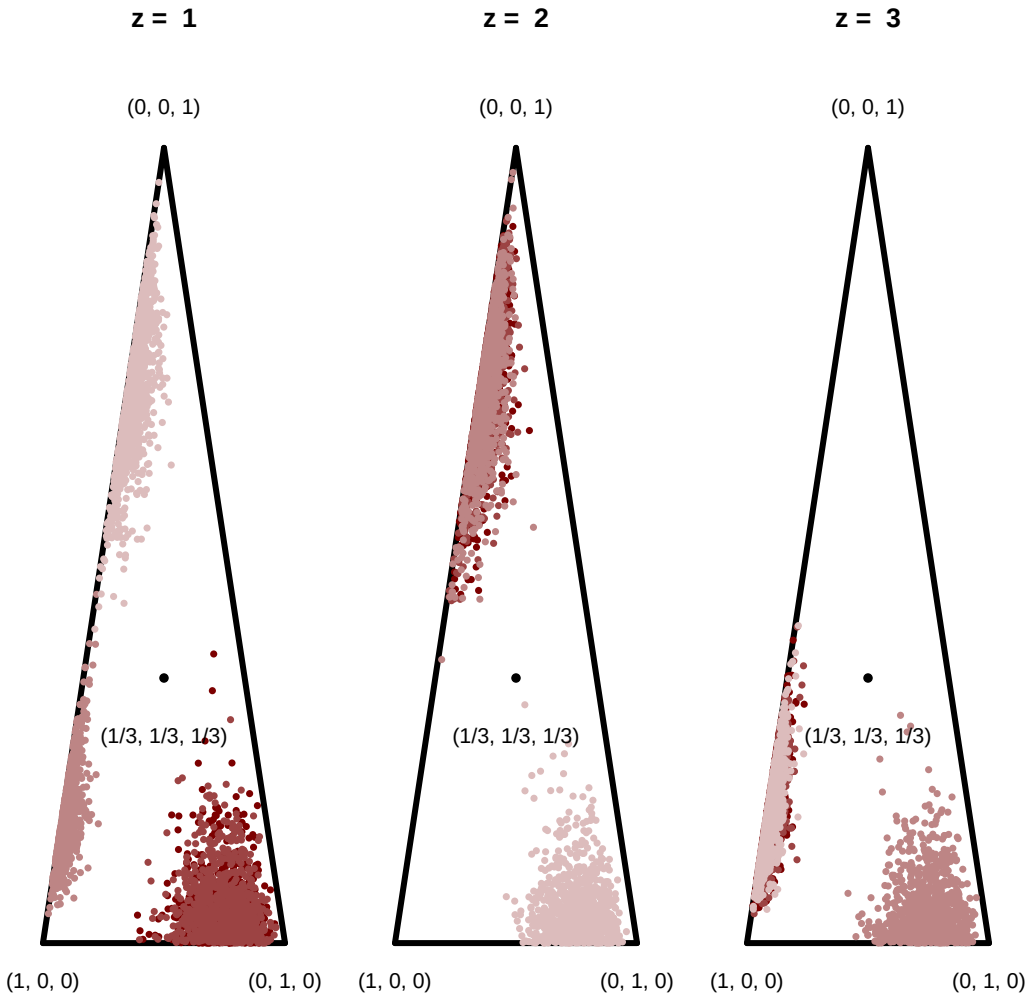
Unsurprisingly each Markov chain was indeed captured into the domain of a single mode.

In this case we can also plot all three simplex components together using the simplex visualization we constructed above.

```
plot_simplex_samples_by_chain <- function(samples, names, title="") {
  vals <- lapply(names, function(name) samples[[name]])
  C <- dim(vals[[1]])[1]
  N <- dim(vals[[1]])[2]

  D <- 1 / sqrt(3)
  xlim <- c(-D - 0.2, +D+ 0.2)
  ylim <- c(-0.1, 1.1)

  par(mar = c(0, 0, 1, 0))
  plot(1, type="n", main=title,
       xlim=xlim, ylim=ylim,
       axes=FALSE)

  lines( c(-D, 0),  c(0, 1), lwd=3)
  lines( c(+D, 0),  c(0, 1), lwd=3)
  lines( c(-D, +D), c(0, 0), lwd=3)

  text_delta <- 0.05
  text( 0, 1 + text_delta, "(0, 0, 1)", cex=1)
  text(-D, -text_delta, "(1, 0, 0)", cex=1)
  text(+D, -text_delta, "(0, 1, 0)", cex=1)

  nom_colors <- c("#DCBCBC", "#C79999", "#B97C7C",
                  "#A25050", "#8F2727", "#7C0000")
  chain_colors <- colormap(colormap=nom_colors, nshades=C)

  for (c in 1:C) {
    points(D * (vals[[2]][c,] - vals[[1]][c,]), vals[[3]][c,],
           pch=16, cex=0.75, col=chain_colors[c])
  }

  points(0, 1/3, col="white", pch=16, cex=1.5)
  points(0, 1/3, col="black", pch=16, cex=1)
  text(0, 1/3 - 1.5 * text_delta, "(1/3, 1/3, 1/3)", cex=1)
}
```

This allows us to clearly see that various modes correspond to permutations of the latent states.

```
r <- 1
par(mfrow=c(1, 3))

for (k_true in 1:3) {
  names <- sapply(1:data$K, function(k) paste0('fidelity[', r, ',',
                                               k_true, ',', k, ']'))
  plot_simplex_samples_by_chain(samples, names,
                                paste0("z =  ", k_true))
}
```



In other words the data are consistent with the first reviewer being accurate, providing the correct answer more often than not, but they are also consistent with the first reviewer being biased, consistently giving the wrong answer given each true latent state. This is exactly the

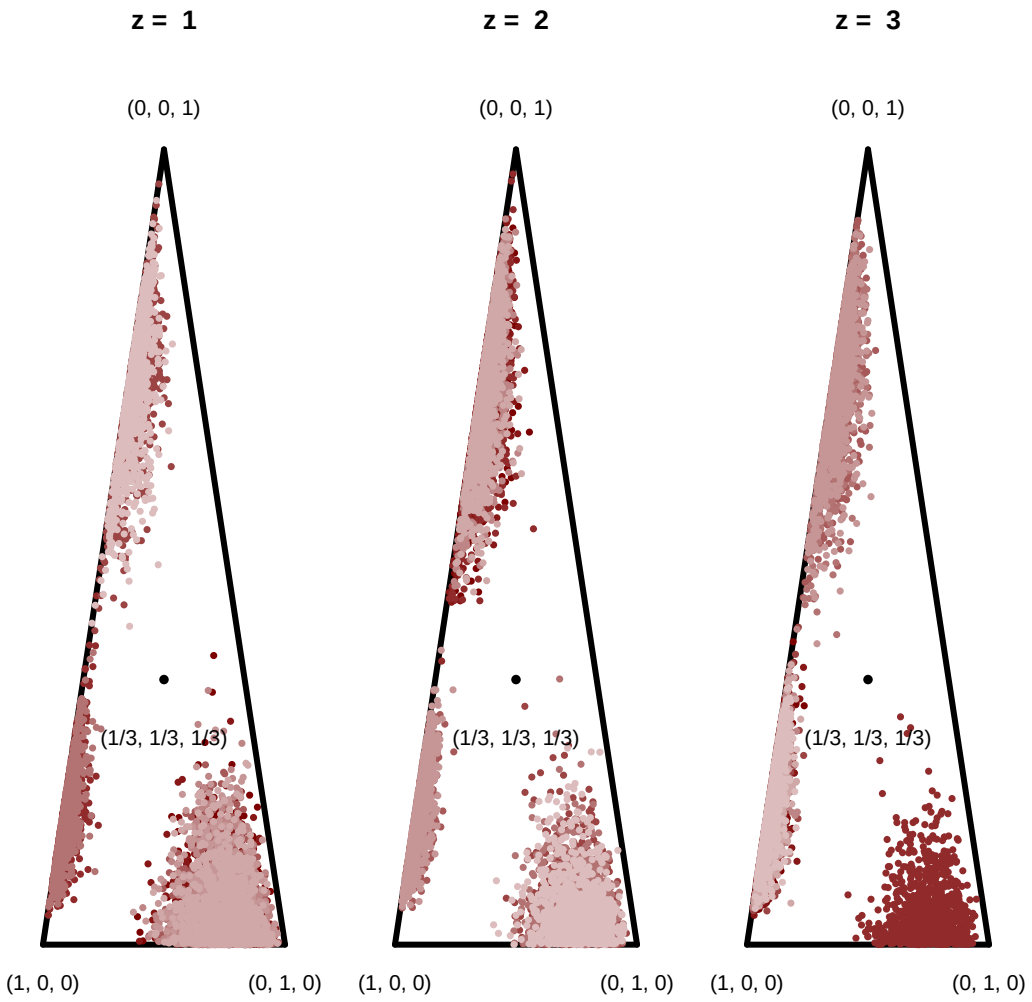
permutation degeneracy that we noted above.

To confirm we can run again but with more Markov chains to see if we can find more of the 6 expected modes.

```
fit <- stan(file="stan_programs/fit1.stan",
            data=data, seed=8438338, chains=10,
            warmup=1000, iter=2024, refresh=0)

samples <- util$extract_expectands(fit)
```

In general it is not trivial to verify how many modes we're seeing in projections of a high-dimensional space, but here three modes manifesting in each simplex is consistent with the expected permutations.

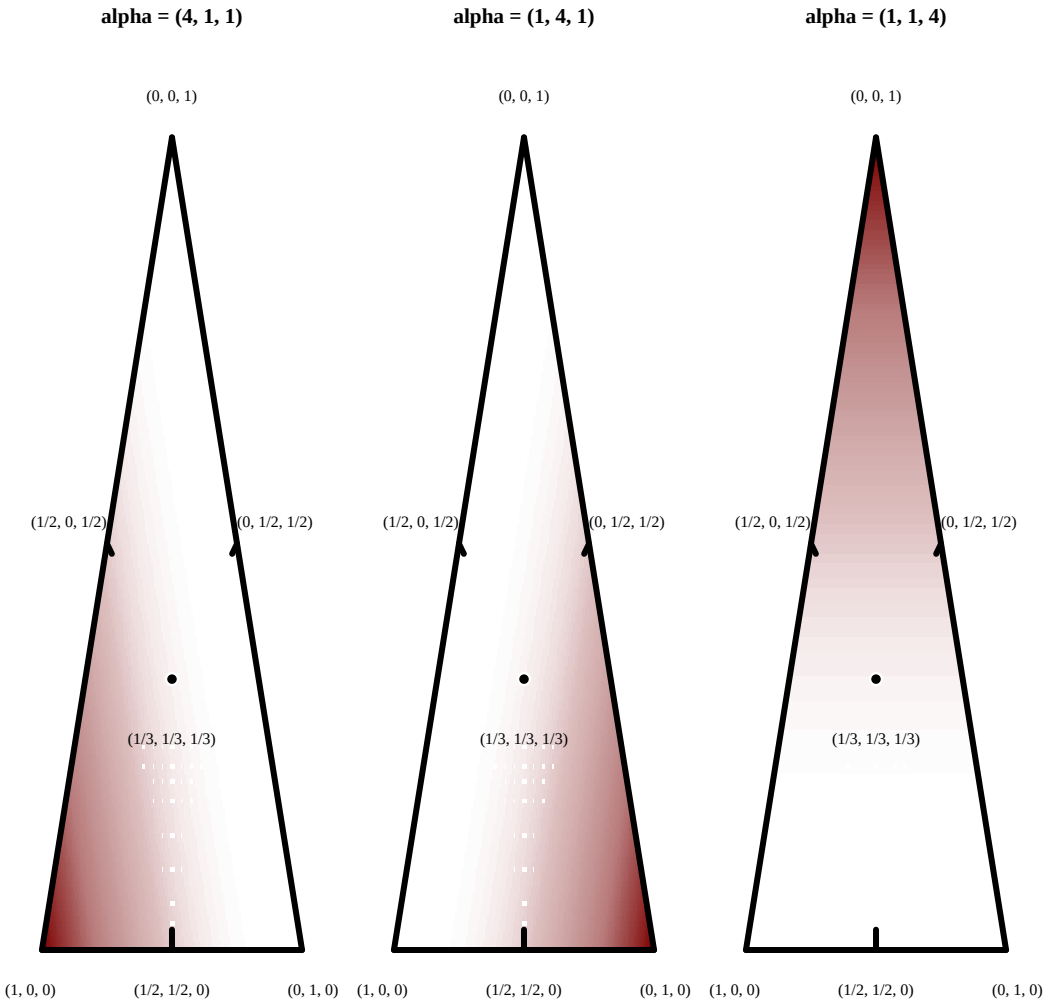
```
r <- 1
par(mfrow=c(1, 3))

for (k_true in 1:3) {
  names <- sapply(1:data$K, function(k) paste0('fidelity[', r, ',',
                                               k_true, ',', k, ']'))
  plot_simplex_samples_by_chain(samples, names,
                                paste0("z =  ", k_true))
}
```



3.6 Model 2

Let's see if we can suppress these degeneracies with a more informative prior model for the fidelities consistent with reasonably accurate reviewers.

```
par(mfrow=c(1, 3))

plot_dirichlet(4, 1, 1, 0.75, "alpha = (4, 1, 1)")
plot_dirichlet(1, 4, 1, 0.75, "alpha = (1, 4, 1)")
plot_dirichlet(1, 1, 4, 0.75, "alpha = (1, 1, 4)")
```



One of the frustrations with multimodal likelihood functions is that a prior model can suppress the modes but not eliminate them entirely. If we initialize our Markov chains too close to the undesired modes then they can be captured no matter how strongly the prior model concentrates away from them.

Here we'll generate custom initializations from the prior model.

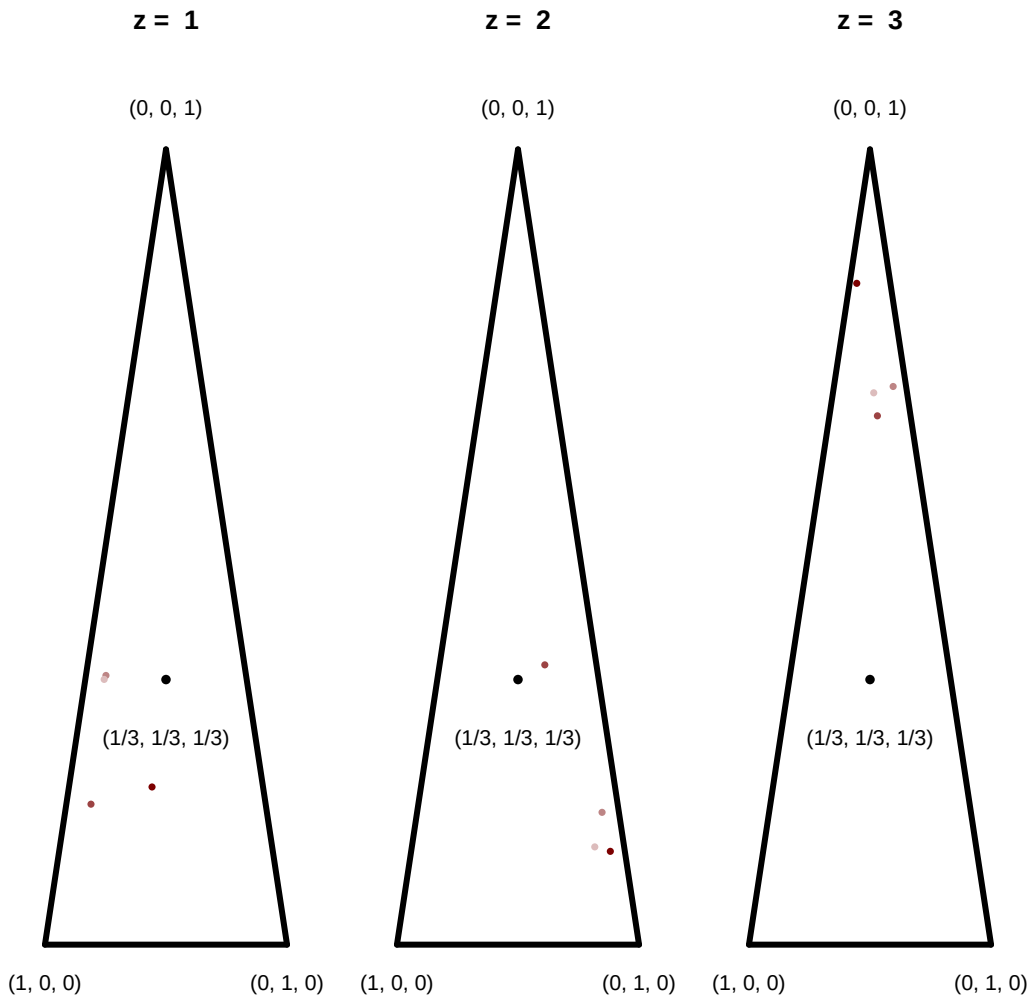
```
dirichlet_rng <- function(a1, a2, a3) {
  gammas <- c(rgamma(1, a1, 1), rgamma(1, a2, 1), rgamma(1, a3, 1))
  gammas / sum(gammas)
}

set.seed(48383499)
```

```
inits <- list()

for (c in 1:4) {
  chain_inits <- list()

  for (n in 1:data$N_assessments) {
    name <- paste0('p[', n, ']')
    chain_inits[[name]] <- dirichlet_rng(1, 1, 1)
  }

  for (r in 1:data$N_reviewers) {
    for (k in 1:data$K) {
      name <- paste0('fidelity[', r, ',', k, ']')
      if (k == 1)
        chain_inits[[name]] <- dirichlet_rng(4, 1, 1)
      else if (k == 2)
        chain_inits[[name]] <- dirichlet_rng(1, 4, 1)
      else if (k == 3)
        chain_inits[[name]] <- dirichlet_rng(1, 1, 4)
    }
  }

  inits[[c]] <- chain_inits
}
```

By construction the initializations for all four Markov chains will be in reasonable neighborhoods.

```
plot_simplex_inits <- function(inits, name, title="") {
  C <- length(inits)

  D <- 1 / sqrt(3)
  xlim <- c(-D - 0.2, +D+ 0.2)
  ylim <- c(-0.1, 1.1)

  par(mar = c(0, 0, 1, 0))
  plot(1, type="n", main=title,
       xlim=xlim, ylim=ylim,
       axes=FALSE)

  lines( c(-D, 0),  c(0, 1), lwd=3)
```

```
  lines( c(+D, 0),  c(0, 1), lwd=3)
  lines( c(-D, +D), c(0, 0), lwd=3)

  text_delta <- 0.05
  text( 0, 1 + text_delta, "(0, 0, 1)", cex=1)
  text(-D, -text_delta, "(1, 0, 0)", cex=1)
  text(+D, -text_delta, "(0, 1, 0)", cex=1)

  nom_colors <- c("#DCBCBC", "#C79999", "#B97C7C",
                  "#A25050", "#8F2727", "#7C0000")
  chain_colors <- colormap(colormap=nom_colors, nshades=C)

  for (c in 1:C) {
    val <- inits[[c]][[name]]
    points(D * (val[2] - val[1]), val[3],
           pch=16, cex=0.75, col=chain_colors[c])
  }

  points(0, 1/3, col="white", pch=16, cex=1.5)
  points(0, 1/3, col="black", pch=16, cex=1)
  text(0, 1/3 - 1.5 * text_delta, "(1/3, 1/3, 1/3)", cex=1)
}

r <- 1
par(mfrow=c(1, 3))

for (k_true in 1:3) {
  name <- paste0('fidelity[', r, ',', k_true, ']')
  plot_simplex_inits(inits, name, paste0("z =  ", k_true))
}
```



Crossing our fingers we try again.

```
# Quantify posterior distribution
fit <- stan(file="stan_programs/fit2.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0,
            init=inits)
```

Sadly, pain. The same pattern of diagnostics suggests that there are still substantial multimodalities in the posterior distribution.

```
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.
```

```
samples <- util$extract_expectands(fit)
names <- grep('pred', names(samples), value=TRUE, invert=TRUE)
base_samples <- samples[names]
util$check_all_expectand_diagnostics(base_samples)
```

```
fidelity[2,2,1]:
  Split hat{R} (1.704) exceeds 1.1!

fidelity[4,2,1]:
  Split hat{R} (1.735) exceeds 1.1!

fidelity[5,2,1]:
  Split hat{R} (1.151) exceeds 1.1!

fidelity[2,3,1]:
  Split hat{R} (1.663) exceeds 1.1!

fidelity[4,3,1]:
  Split hat{R} (1.736) exceeds 1.1!

fidelity[5,3,1]:
  Split hat{R} (1.158) exceeds 1.1!

fidelity[1,2,2]:
  Split hat{R} (4.866) exceeds 1.1!

fidelity[2,2,2]:
  Split hat{R} (5.746) exceeds 1.1!

fidelity[3,2,2]:
  Split hat{R} (5.732) exceeds 1.1!

fidelity[4,2,2]:
  Split hat{R} (3.344) exceeds 1.1!

fidelity[5,2,2]:
  Split hat{R} (5.410) exceeds 1.1!
```

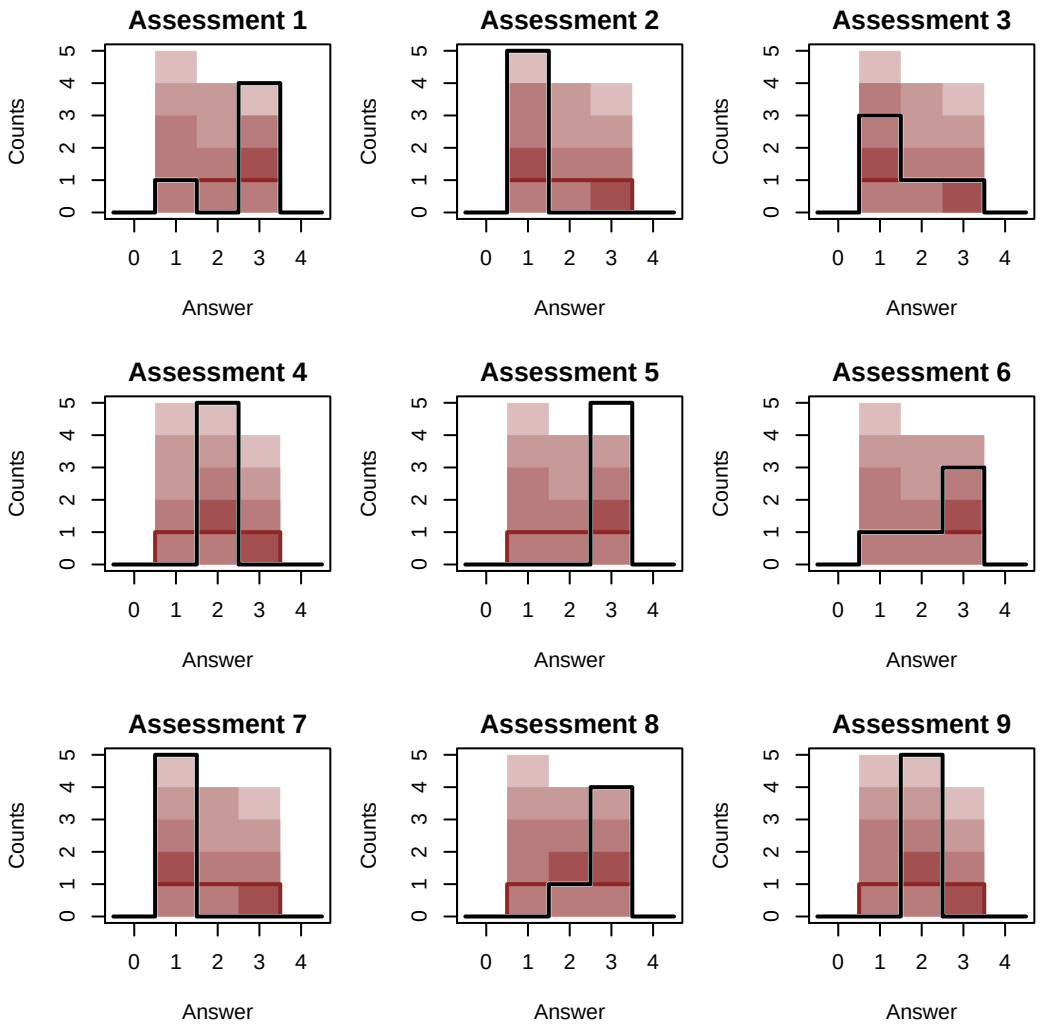
```
fidelity[1,3,2]:
  Split hat{R} (4.777) exceeds 1.1!

fidelity[2,3,2]:
  Split hat{R} (5.117) exceeds 1.1!

fidelity[3,3,2]:
  Split hat{R} (5.572) exceeds 1.1!

fidelity[4,3,2]:
  Split hat{R} (3.139) exceeds 1.1!

fidelity[5,3,2]:
  Split hat{R} (4.727) exceeds 1.1!

fidelity[1,2,3]:
  Split hat{R} (5.078) exceeds 1.1!

fidelity[2,2,3]:
  Split hat{R} (4.536) exceeds 1.1!

fidelity[3,2,3]:
  Split hat{R} (6.078) exceeds 1.1!

fidelity[4,2,3]:
  Split hat{R} (2.503) exceeds 1.1!

fidelity[5,2,3]:
  Split hat{R} (6.343) exceeds 1.1!

fidelity[1,3,3]:
  Split hat{R} (4.294) exceeds 1.1!

fidelity[2,3,3]:
  Split hat{R} (3.791) exceeds 1.1!

fidelity[3,3,3]:
  Split hat{R} (5.585) exceeds 1.1!

fidelity[4,3,3]:
  Split hat{R} (2.162) exceeds 1.1!

fidelity[5,3,3]:
```

```
  Split hat{R} (5.233) exceeds 1.1!
```

```
Split Rhat larger than 1.1 suggests that at least one of the Markov
chains has not reached an equilibrium.
```

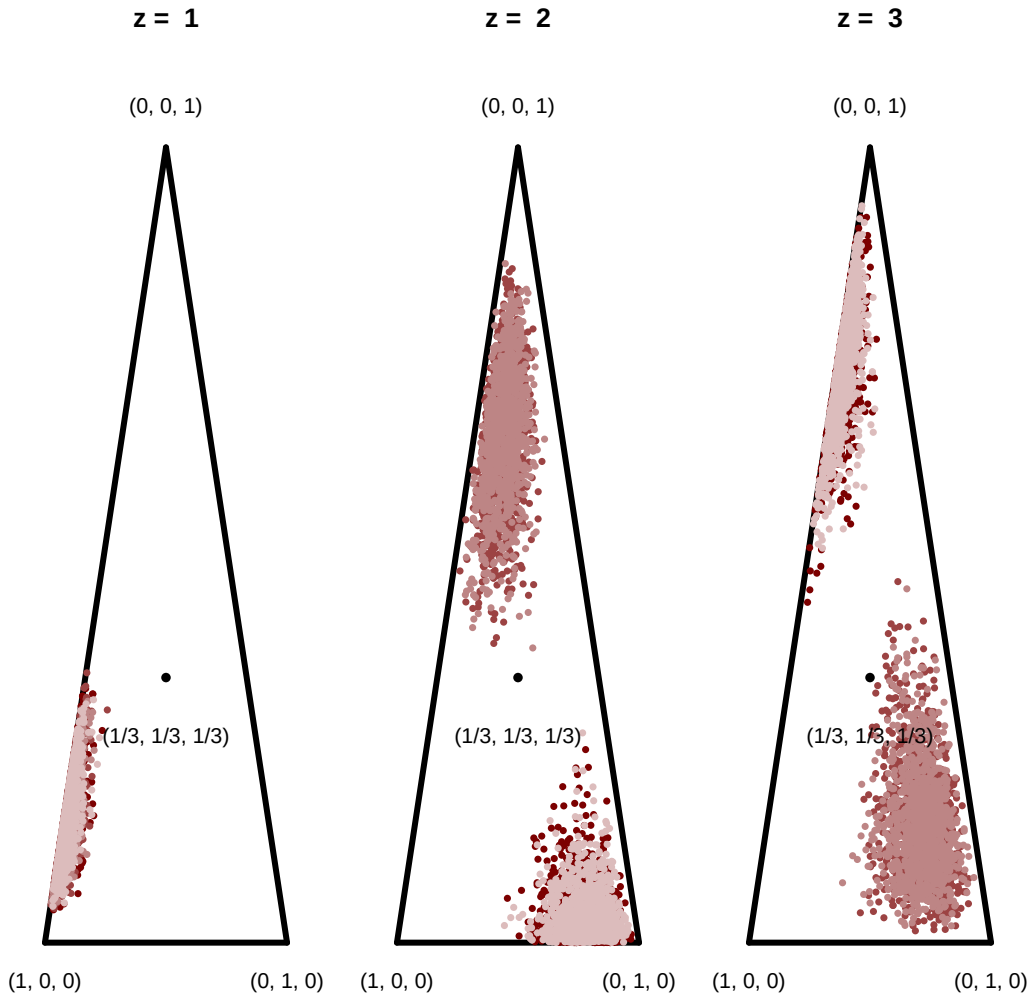
The retrodictive performance hasn't changed at all.

```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))

for (n in 1:9) {
  pred_names <- grep(paste0('y_pred\\[', n, ','),
                     names(samples), value=TRUE)
  hist_retro(data$y[n,], samples, pred_names, -0.5, data$K + 1.5, 1,
             "Answer", c(0, 5), paste("Assessment", n))
}
```



We can clearly see, however, the multimodality.

```
r <- 1
par(mfrow=c(1, 3))

for (k_true in 1:3) {
  names <- sapply(1:data$K, function(k) paste0('fidelity[', r, ',',
                                               k_true, ',', k, ']'))
  plot_simplex_samples_by_chain(samples, names,
                                paste0("z =  ", k_true))
}
```



It doesn't look like our more informative prior model was sufficiently informative to actually suppress the undesired modes. At this point we could consider an even stronger prior model, but in practice we will often be limited by the available domain expertise. Forcing our model to assume highly accurate reviewers just to avoid the degeneracies of the finite consensus model will lead to poor results if the reviewers are not in fact so performant.

3.7 Model 3

Instead of frustrating ourselves with a detailed prior elicitation, let's take advantage of our small calibration data set, fitting the assessments with and without accompanying latent state observations at the same time.

```
data <- c(data, calib_data)
```

```
# Quantify posterior distribution
fit <- stan(file="stan_programs/fit3.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0,
            init=inits)
```

Delightfully the persistent $\hat{R}$ warnings have vanished. This doesn't necessarily mean that we have actually resolved the multimodality, but it is good news.

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

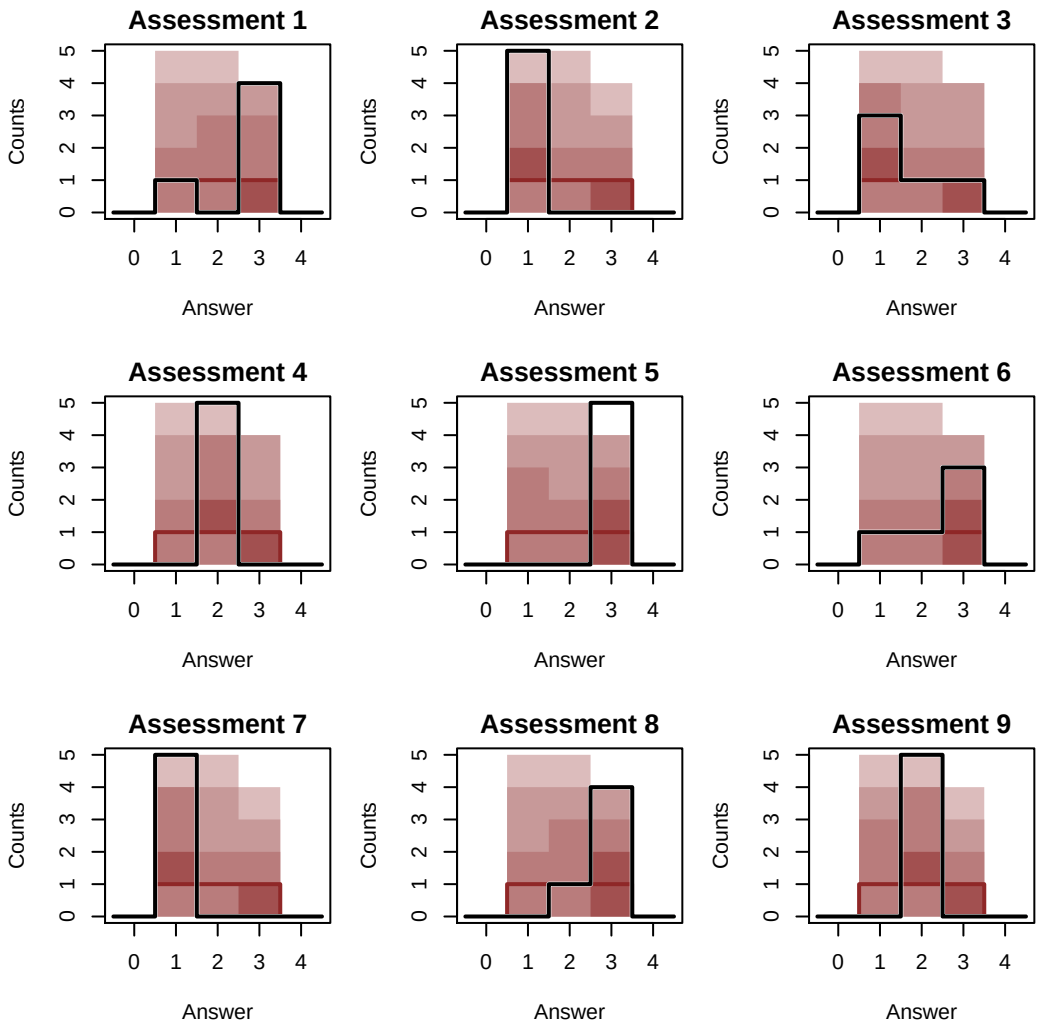
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectands(fit)
names <- grep('pred', names(samples), value=TRUE, invert=TRUE)
base_samples <- samples[names]
util$check_all_expectand_diagnostics(base_samples)
```

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

Before exploring further we need to check that our retrodictive performance has not been compromised.

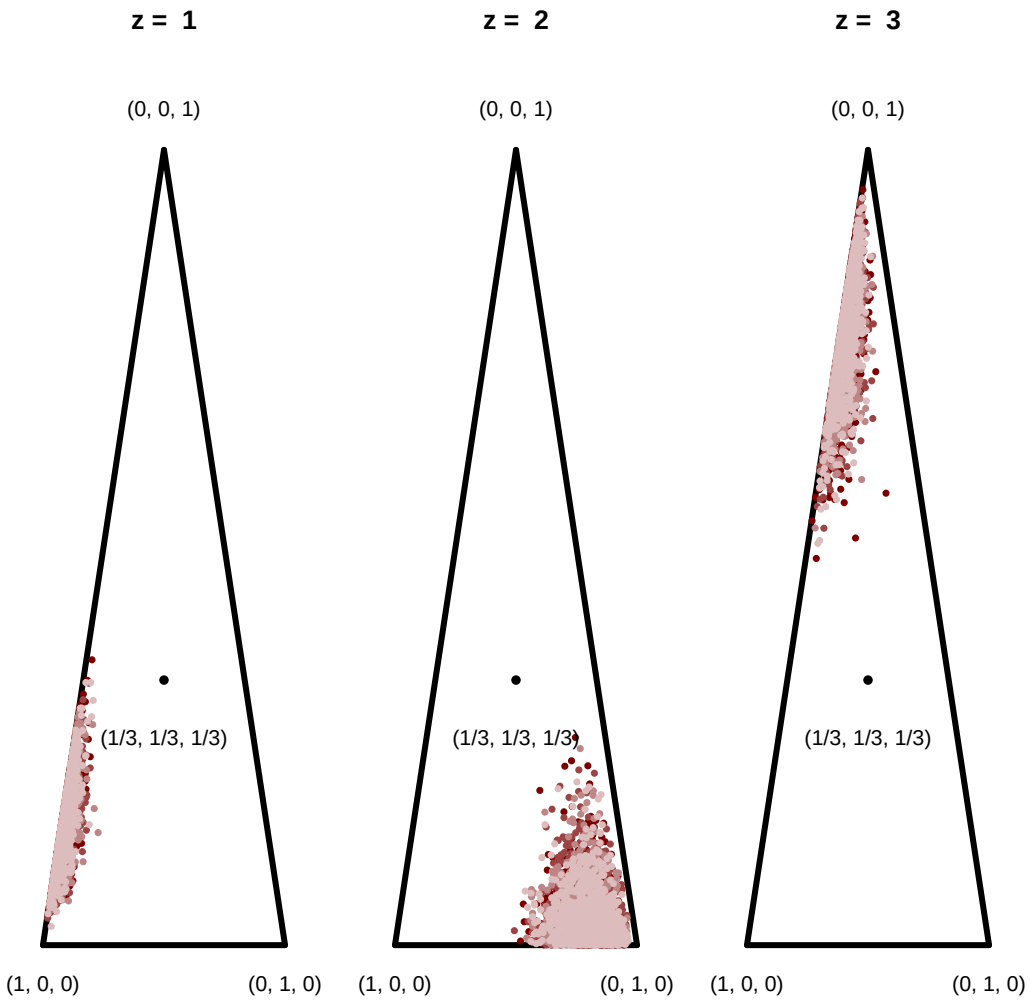
```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))

for (n in 1:9) {
  pred_names <- grep(paste0('y_pred\\[', n, ','),
                     names(samples), value=TRUE)
  hist_retro(data$y[n,], samples, pred_names, -0.5, data$K + 1.5, 1,
             "Answer", c(0, 5), paste("Assessment", n))
}
```



Without any clear sign of model inadequacies we can proceed to investigating our posterior inferences. As we hoped our posterior inferences are finally contained into a single mode within the "accurate" corners of the simplices.

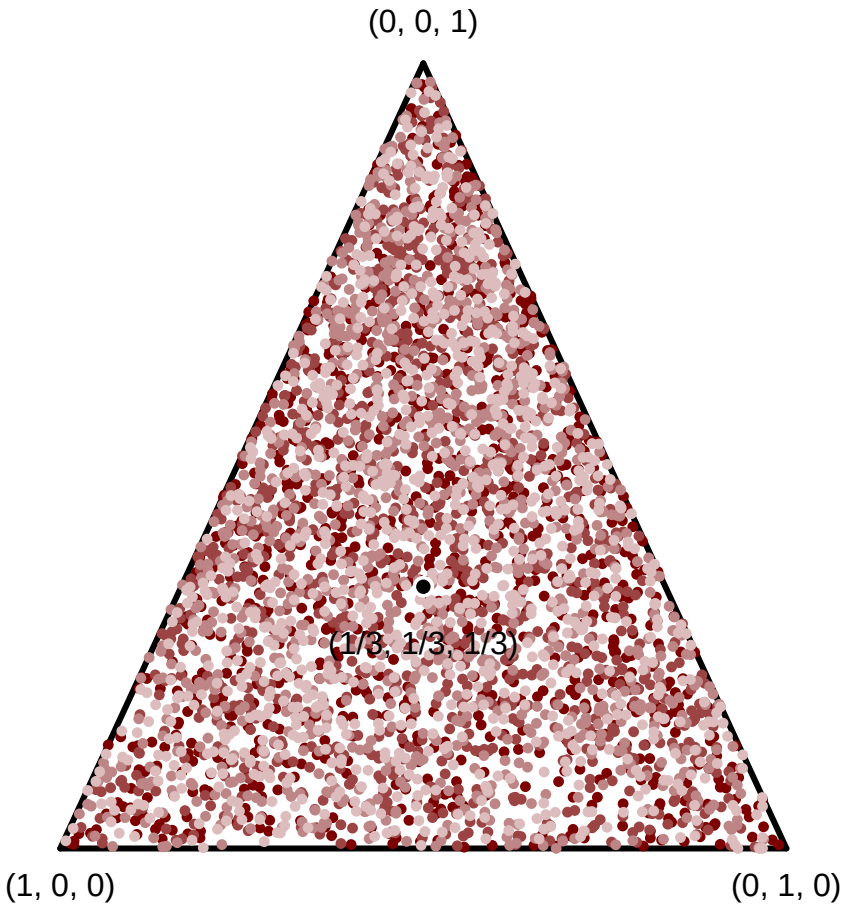
```
r <- 1
par(mfrow=c(1, 3))

for (k_true in 1:3) {
  names <- sapply(1:data$K, function(k)
                  paste0('fidelity[', r, ',', k_true, ',', k, ']'))
  plot_simplex_samples_by_chain(samples, names,
                                paste0("z =  ", k_true))
```

```
}
```



Perhaps surprisingly the posterior inferences for the latent states are not particularly informative. Indeed the Markov chain samples almost look uniformly distributed across the simplices!

```
par(mfrow=c(1, 1))
for (n in c(1)) {
  names <- sapply(1:data$K, function(k) paste0('psi[', n, ',', k, ']'))
  plot_simplex_samples_by_chain(samples, names,
                                paste0("Assessment ", n))
}
```



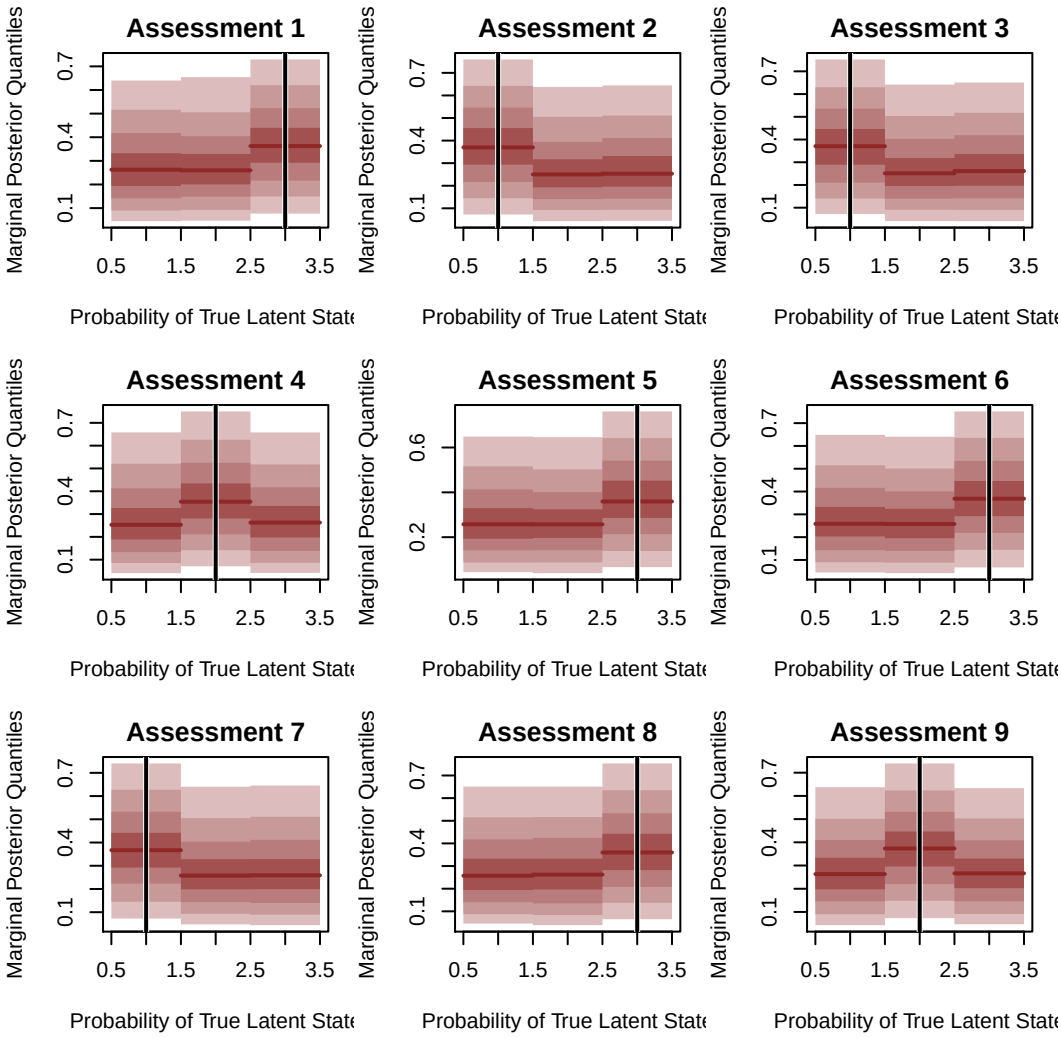
To see if there are any more systematic preference for one state over another we can examine marginal quantiles for each possible value.

```
plot_disc_marginal_quantiles <- function(samples, names,
                                         x_name="", title="") {
  N <- length(names)
  idx <- rep(1:N, each=2)
  xs <- sapply(1:length(idx), function(k) if(k %% 2 == 0) idx[k] + 0.5
                                          else            idx[k] - 0.5)

  probs = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)
  cred <- sapply(1:N, function(n)
                      quantile(c(t(samples[[names[n]]]),
```

```
                                  recursive=TRUE),
                                probs=probs))
  pad_cred <- do.call(cbind, lapply(idx, function(n) cred[1:9,n]))

  ylims <- c(min(cred[1,]), max(cred[9,]))

  plot(1, type="n", main=title,
       xlim=c(0.5, N + 0.5), xlab=x_name,
       ylim=ylims, ylab="Marginal Posterior Quantiles")

  polygon(c(xs, rev(xs)), c(pad_cred[1,], rev(pad_cred[9,])),
          col = c_light, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[2,], rev(pad_cred[8,])),
          col = c_light_highlight, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[3,], rev(pad_cred[7,])),
          col = c_mid, border = NA)
  polygon(c(xs, rev(xs)), c(pad_cred[4,], rev(pad_cred[6,])),
          col = c_mid_highlight, border = NA)
  for (n in 1:N) {
    lines(xs[(2 * n - 1):(2 * n)], pad_cred[5,(2 * n - 1):(2 * n)],
          col=c_dark, lwd=2)
  }
}
```

Using this visualization we can see that the true latent state is slightly preferred to the others.

```
par(mfrow=c(3, 3), mar = c(5, 4, 2, 1))
for (n in 1:9) {
  names <- sapply(1:data$K, function(k) paste0('psi[', n, ',', k, ']'))
  plot_disc_marginal_quantiles(samples, names,
                               x_name="Probability of True Latent State",
                               title=paste("Assessment", n))
  abline(v=true_zs[n], lwd=3, col="white")
  abline(v=true_zs[n], lwd=2, col="black")
}
```



This lack of inferential precision is not uncommon for finite consensus models. Even seemingly small uncertainties in the reviewer fidelities can lead to large uncertainties in the consensus. At the same time informing the reviewer fidelities typically requires a substantial amount of data, especially when precise domain expertise is not available.

Acknowledgements

A very special thanks to everyone supporting me on Patreon:

References

Dawid, A. P., and A. M. Skene. 1979. "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28 (1): 20–28.

Good, I. J., and W. I. Card. 1971. "The Diagnostic Process with Special Reference to Errors." *Methods of Information in Medicine* 10: 176–88.

Landis, J. Richard, and Gary G. Koch. 1975a. "A Review of Statistical Methods in the Analysis of Data Arising from Observer Reliability Studies (Part i)." *Statistica Neerlandica* 29 (3): 101–23.

———. 1975b. "A Review of Statistical Methods in the Analysis of Data Arising from Observer Reliability Studies (Part II)." *Statistica Neerlandica* 29 (3): 151–61.

———. 1977. "The Measurement of Observer Agreement for Categorical Data." *Biometrics* 33 (1): 159–74.

License

The code in this case study is copyrighted by Michael Betancourt and licensed under the new BSD (3-clause) license:

https://opensource.org/licenses/BSD-3-Clause

The text and figures in this case study are copyrighted by Michael Betancourt and licensed under the CC BY-NC 4.0 license:

https://creativecommons.org/licenses/by-nc/4.0/

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
```

```r
sessionInfo()
```

```
R version 4.0.2 (2020-06-22)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Catalina 10.15.7

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/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

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

other attached packages:
[1] colormap_0.1.4       rstan_2.19.3         ggplot2_3.3.1
[4] StanHeaders_2.21.0-3

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.4.6       pillar_1.4.4       compiler_4.0.2     prettyunits_1.1.1
 [5] tools_4.0.2        digest_0.6.25      pkgbuild_1.0.8     jsonlite_1.6.1
 [9] evaluate_0.17      lifecycle_0.2.0    tibble_3.0.1       gtable_0.3.0
[13] pkgconfig_2.0.3    rlang_0.4.6        cli_2.0.2          curl_4.3
[17] parallel_4.0.2     yaml_2.2.1         xfun_0.33          loo_2.2.0
[21] gridExtra_2.3      withr_2.2.0        stringr_1.4.0      dplyr_1.0.0
[25] knitr_1.40         generics_0.0.2     vctrs_0.3.0        stats4_4.0.2
[29] grid_4.0.2         tidyselect_1.1.0   glue_1.4.1         inline_0.3.15
[33] R6_2.4.1           processx_3.4.2     fansi_0.4.1        rmarkdown_2.2
[37] callr_3.4.3        purrr_0.3.4        magrittr_1.5       matrixStats_0.56.0
[41] ps_1.3.3           scales_1.1.1       codetools_0.2-16   ellipsis_0.3.1
[45] htmltools_0.4.0    assertthat_0.2.1   colorspace_1.4-1   V8_3.2.0
[49] stringi_1.4.6      munsell_0.5.0      crayon_1.3.4
```

Stan Program 1 `simu.stan`

```
transformed data {
  int<lower=0> K = 3;
  int<lower=0> N_reviewers = 5;
  int<lower=0> N_assessments = 110;

  vector[K] psi = [ 0.4, 0.3, 0.3 ]';
  vector[K] fidelity[N_reviewers, K];


  for (r in 1:N_reviewers) {
    for (k in 1:K) {
      vector[K] alpha = rep_vector(1, K);

      alpha[k] = 10;
      fidelity[r, k] = dirichlet_rng(alpha);
    }
  }

}

generated quantities {

  int<lower=0, upper=K> z[N_assessments];
  int<lower=0, upper=K> y[N_assessments, N_reviewers];

  for (n in 1:N_assessments) {

    z[n] = categorical_rng(psi);

    for (r in 1:N_reviewers) {

      y[n, r] = categorical_rng(fidelity[r, z[n]]);
    }
  }
}
```

Stan Program 2 `fit1.stan`

```
data {
  int<lower=0> K;
  int<lower=0> N_reviewers;
  int<lower=0> N_assessments;


  int<lower=0, upper=K> y[N_assessments, N_reviewers];
}

parameters {
  // One simplex for the true answers of every assessment

  simplex[K] psi[N_assessments];

  // One simplex per reviewer per possible true answers
  simplex[K] fidelity[N_reviewers, K];
}


model {
  for (n in 1:N_assessments) {
    psi[n] ~ dirichlet(rep_vector(1, K));
  }


  for (r in 1:N_reviewers) {
    for (k in 1:K) {
      vector[K] alpha = rep_vector(1, K);
      fidelity[r, k] ~ dirichlet(alpha);
    }
  }

  for (n in 1:N_assessments) {
    real lds[K];

    // Loop over possible correct answers
    for (k in 1:K) {

      lds[k] = psi[n][k]; // Probability of kth answer being correct
      for (r in 1:N_reviewers) {
        // Probability of response given that the kth answer is correct
        real p_response = fidelity[r, k][y[n, r]];
        lds[k] += log(p_response);
      }

    }
    target += log_sum_exp(lds);
  }
}


generated quantities {
```

Stan Program 3 `fit2.stan`

```
data {
  int<lower=0> K;
  int<lower=0> N_reviewers;
  int<lower=0> N_assessments;

  int<lower=0, upper=K> y[N_assessments, N_reviewers];
}

parameters {
  // One simplex for the true answers of every assessment
  simplex[K] psi[N_assessments];


  // One simplex per reviewer per possible true answers
  simplex[K] fidelity[N_reviewers, K];
}

model {
  for (n in 1:N_assessments) {
    psi[n] ~ dirichlet(rep_vector(1, K));
  }

  for (r in 1:N_reviewers) {
    for (k in 1:K) {
      vector[K] alpha = rep_vector(1, K);

      alpha[k] = 4;
      fidelity[r, k] ~ dirichlet(alpha);
    }
  }

  for (n in 1:N_assessments) {
    real lds[K];

    // Loop over possible correct answers
    for (k in 1:K) {
      lds[k] = psi[n][k]; // Probability of kth answer being correct
      for (r in 1:N_reviewers) {

        // Probability of response given that the kth answer is correct
        real p_response = fidelity[r, k][y[n, r]];
        lds[k] += log(p_response);
      }
    }
    target += log_sum_exp(lds);
  }
}

generated quantities {
  int<lower=0, upper=K> y_pred[N_assessments, N_reviewers];
  for (n in 1:N_assessments) {
```

Stan Program 4 `fit3.stan`