Ordinal Modeling

Michael Betancourt

February 2025

Table of contents

Many of the spaces we encounter in applications are ordered, such as the integers and real lines. Productive probabilistic models on these spaces need to be compatible with the ordering structures, although the nature of that compatibility is not always obvious. The influence of an ordering on probabilistic models can be even more subtle in more generic spaces.

In this chapter we'll consider techniques for modeling ordinal data that take values in finite spaces that are ordered. The discussion will emphasize potential inferential degeneracies as well as effective moderation strategies.

1 Ordinal Data

Ordinal spaces are formally defined as a finite set,

$$y_1, \ldots, y_k, \ldots, y_K,$$

whose elements are strictly ordered,

$$y_1 < \ldots < y_k < \ldots < y_K.$$

While the integer indices conveniently communicate the ordering of the ordinal values they do *not* imply that the elements are themselves integers. Unlike the integers ordinal spaces

are not generally equipped with metric or algebraic structures. Although an ordering defines neighboring elements we have no way to quantify how far apart two neighbors are, let alone how uniform those distances might be across the space.

Instead an ordering defines only a qualitative notion of "locality", with an element being more local to its immediate neighbors than the neighbors of its neighbors and so on. This in turn implies a vague notion of "continuity", with behaviors associated with more local elements tending to be more similar than those associated with less local elements.

That said these qualitative similarities are often much weaker than what we would expect on more rigid spaces such as the integers. Because neighboring elements can be arbitrarily far apart the coupling between neighboring behaviors can be arbitrarily weak. Moreover the coupling will not, in general, be uniform across different neighbors.

A common application of ordinal spaces is modeling data that are collected by soliciting from individuals judgements or preferences that are limited to discrete but ordered values. For example the possible responses in a survey might be

$$\text{Disagree}, \text{Undecided}, \text{Agree}$$

or

$$\text{Strongly Disagree}, \text{Disagree}, \text{Neutral}, \text{Agree}, \text{Strongly Agree}.$$

These discrete values can require less precise elicitation than a more informative continuous response, potentially resulting in more reliable responses.

In these applications the ordinal values are often referred to as **Likert scales**, pronounced "Lick-urt", after the social psychologist Rensis Likert. To remain as general as possible, and avoid the heritage of twentieth century racial psychology to which Likert scales were first applied, I will maintain the "ordinal" terminology in this chapter.

2 Modeling Ordinal Data

The immediate challenge in building probabilistic models on ordinal spaces is respecting the ordering without relying on additional structure. In particular models that rely on additional structure, such as an assumed metric, tend to be too rigid to adequately model ordinal data. For example models defined over integer intervals, such as binomial models, are not generally useful. Nor are models given by truncating probabilistic models defined over the integers, such as truncated Poisson models.

If we ignore ordering altogether then an ordinal space becomes a simple finite space. In this case every probability distribution can be specified with atomic probability allocations,

$$p_k = \pi(\{x_k\}),$$

that define a $(K-1)$-simplex configuration,

$$0 \leq p_1, \ldots, p_k, \ldots, p_K \leq 1$$
$$\sum_{k=1}^{K} p_k = 1.$$

Specifying a probability distribution over a finite space with atomic probabilities is sometimes referred to as a **categorical model** with the individual elements referred to as **categories**.

We can also define every probability distribution over an ordinal space with atomic probability allocations. The ordering, however, allows us to consider notions of continuity across these allocations. In particular a more continuous ordinal model would couple neighboring probability allocations together, with p_k tending to be more similar to p_{k-1} and p_{k+1} than it is p_{k-2} and p_{k+2} (Figure 1).

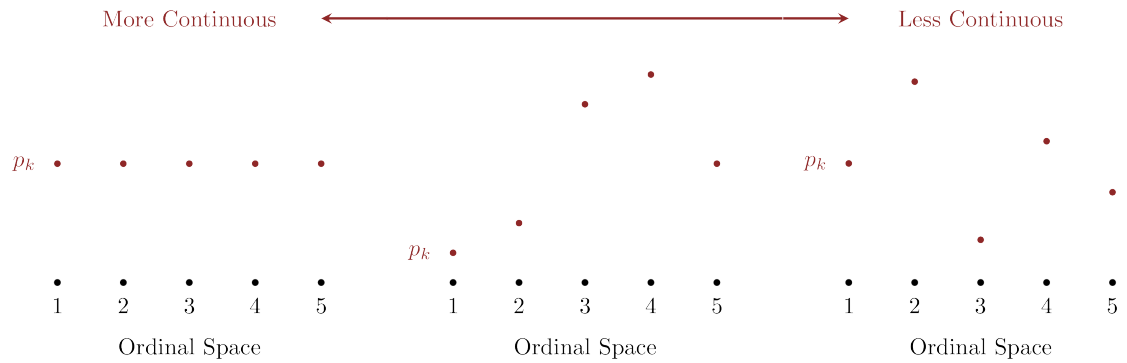


Figure 1: In a categorical model any ordering of the elements is arbitrary and we consequently cannot define even a qualitative notion of continuous probability allocations. Ordinal spaces, however, are equipped with a distinguished ordering that defines neighboring elements and a vague notion of continuity where neighboring probabilities are more similar than non-neighboring probabilities.

Because notions of continuity on ordinal spaces are only qualitative, however, it can be difficult to elicit appropriate domain expertise in a given application. When we are not able to elicit precise constraints then there's no real loss from ignoring the ordering and using a categorical model.

Moreover in circumstances where we do have quantified domain expertise it can often be implemented with a simple categorical model complemented by an informative prior model. In particular if our domain expertise manifests as a preference for probabilities that vary around a baseline simplex configuration ρ then we could use a Dirichlet prior model with the configuration (Figure 2)

$$\alpha_k = \rho_k / \tau + 1.$$

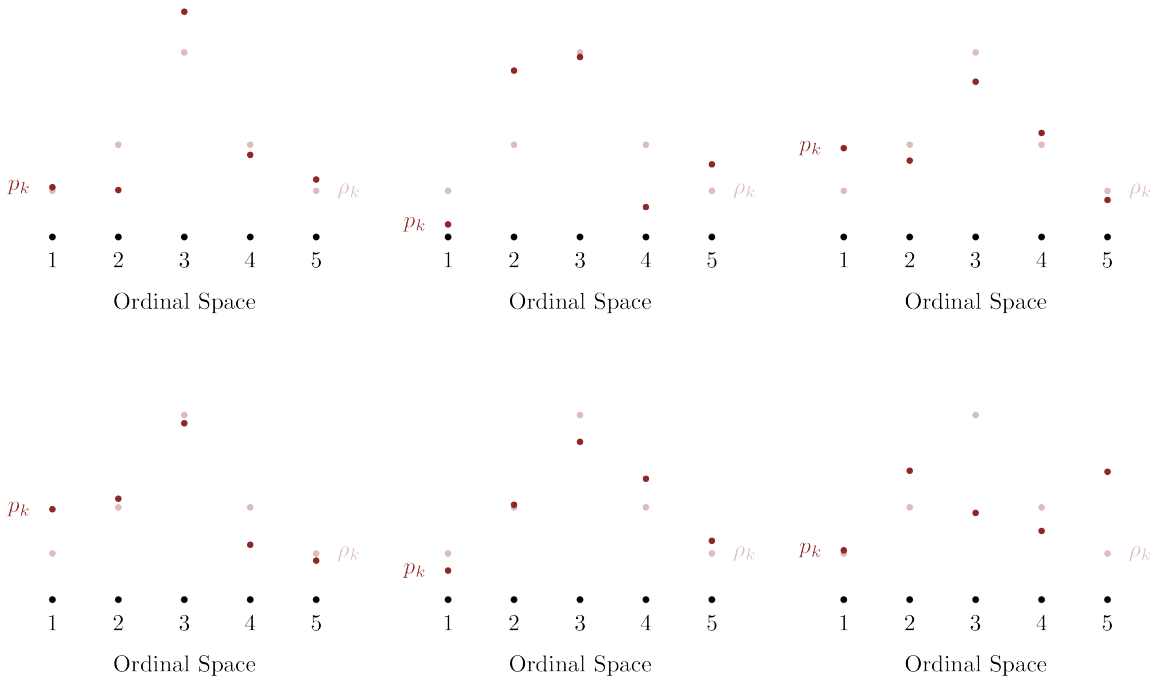


Figure 2: A Dirichlet prior model with the configuration parameters $\alpha_k = \rho_k/\tau + 1$ concentrates around the baseline simplex configuration $(\rho_1, \ldots, \rho_K)$, here shown in light red. Samples from this prior model, here shown in dark red, cluster around this baseline and preserve its basic shape.

Even vague notions of continuity can become a much more relevant consideration when we are modeling *heterogeneity* in ordinal probabilities. Specifically we often want whatever shape that happens to manifest in the ordinal probabilities of some *baseline* circumstance to be somewhat preserved in the other circumstances. In other words the problem of interest is often not modeling independent ordinal probabilities but rather modeling *systematic perturbations* of the baseline ordinal probabilities (Figure 3).

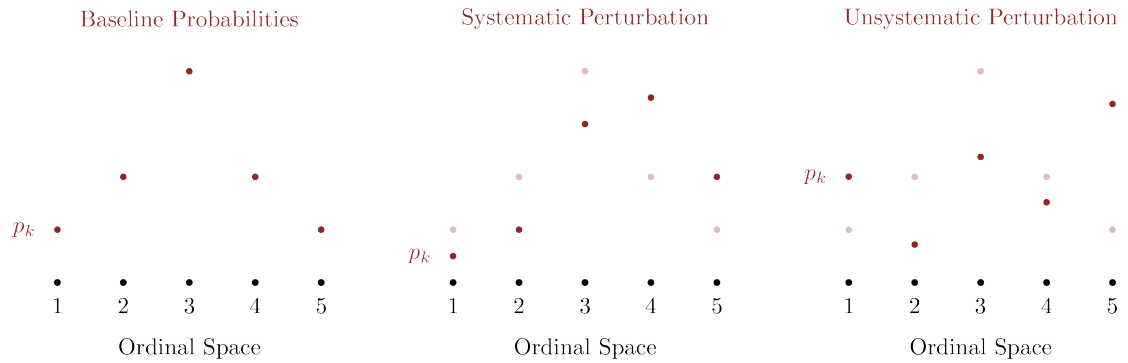


Figure 3: Heterogeneous ordinal probabilities can always be modeled independently of each other. In many applications, however, our domain expertise is consistent with more systematic variations around some baseline behavior. This raises the question of how we can build ordinal models that isolate the relevant variations.

3 Coupling Ordinal Probabilities

One of the most effective ways to limit the behavior of ordinal probabilities is not to model them directly but rather to *derive* them from a latent probability space that is already equipped with a more manageable form of continuity.

3.1 Discretizing A Continuous Probability Space

Continuity of behaviors is much easier to quantify on continuous spaces. In particular let's consider a real line $X = \mathbb{R}$ equipped with a probability distribution π specified by the probability density function $p : X \to \mathbb{R}^+$.

Any partition of X is defined by a sequence of disjoint but connected intervals,

$$\begin{aligned} I_1 &= (-\infty, c_1] \\ &= ... \\ I_k &= (c_{k-1}, c_k] \\ &= ... \\ I_K &= (c_{K-1}, +\infty). \end{aligned}$$

Together these intervals define a finite space

$$(I_1, ..., I_k, ..., I_K).$$

Moreover because all points in one interval are strictly less than or larger than all of the points in any other interval,

$$x < x'$$

for all $x \in I_k$ and $x' \in I_{k' > k}$, these elements are naturally ordered,

$$I_k < I_{k' > k}.$$

Consequently the set of intervals is an ordinal space.

The boundaries between these intervals

$$c_0 = -\infty < c_1 < ... < c_k < ... < c_{K-1} < c_K = +\infty$$

are known as **cut points** as they, well, cut the real line into pieces. I will refer to the $K - 1$ points

$$-\infty < c_1 < ... < c_k < ... < c_{K-1} < +\infty$$

as **interior cut points**.

Now that we can map a real line to a K element ordinal space we can push the latent probability distribution forward to ordinal probabilities. In particular the probability allocated to each ordinal element is the probability allocated to the defining interval (Figure 4),

$$p_k = \pi(\, (c_{k-1}, c_k] \,) = \int_{c_{k-1}}^{c_k} \mathrm{d}x' \, p(x').$$

Given a cumulative distribution function

$$\Pi(x) = \int_{-\infty}^{x} \mathrm{d}x' \, p(x')$$

we can equivalently calculate the derived ordinal probabilities as (Figure 5)

$$p_k = \Pi(c_k) - \Pi(c_{k-1}).$$

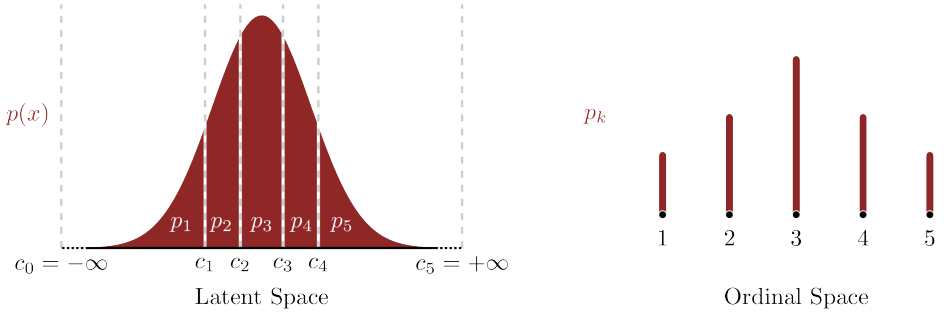


Figure 4: Cut points partition a latent real line into ordered intervals which defines an ordinal space. They also partition a latent probability density function into interval probabilities which defines a probability distribution over that ordinal space.

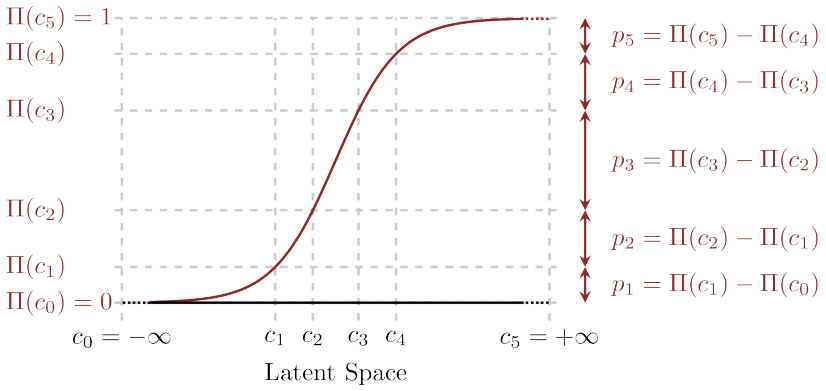


Figure 5: We can quickly compute interval probabilities, and hence induced ordinal probabilities, by subtracting cumulative distribution functions.

If the cumulative distribution function is available in closed form then this form allows us to quickly compute the ordinal probabilities with just a few subtractions.

For any probability distribution on a real line we have

$$\begin{aligned}
\Pi(-x) &= \int_{-\infty}^{-x} \mathrm{d}x' \, p(x') \\
&= -\int_{+\infty}^{x} \mathrm{d}x'' \, p(-x'') \\
&= \int_{x}^{\infty} \mathrm{d}x'' \, p(-x'') \\
&= 1 - \int_{-\infty}^{x} \mathrm{d}x'' \, p(-x'').
\end{aligned}$$

If the latent probability density function is even, $p(-x) = p(x)$, then

$$\begin{aligned}
\Pi(-x) &= 1 - \int_{-\infty}^{x} \mathrm{d}x'' \, p(-x'') \\
&= 1 - \int_{-\infty}^{x} \mathrm{d}x'' \, p(x'') \\
&= 1 - \Pi(x).
\end{aligned}$$

Consequently in this case we can also write the ordinal probabilities as (Figure 6)

$$\begin{aligned}
p_k &= \Pi(c_k) - \Pi(c_{k-1}) \\
&= (\Pi(c_k) - 1) - (\Pi(c_{k-1}) - 1) \\
&= -(1 - \Pi(c_k)) + (1 - \Pi(c_{k-1})) \\
&= -\Pi(-c_k) + \Pi(-c_{k-1}) \\
&= \Pi(-c_{k-1}) - \Pi(-c_k).
\end{aligned}$$

This form is more common in some references.

3.2 The Duality Between Cut Points and Ordinal Probabilities

If the distances between the interior cut points are similar then the resulting ordinal probabilities will manifest the basic shape of the latent probability density function. In particular the more rigid the latent probability density function is the more similar probabilities allocated to neighboring intervals will tend to be relative to the allocations to non-neighboring intervals.

That said if the configuration of the interior cut points, and the lengths of the intervals between them, is arbitrary then the resulting ordinal probabilities will also be arbitrary *regardless of the shape of the latent probability density function.* More formally for *any* fixed latent probability

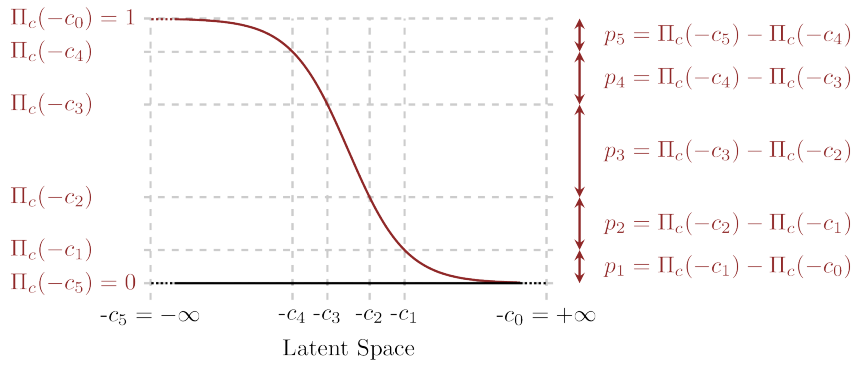


Figure 6: Interval probabilities can also be written as differences of complementary cumulative distribution functions $\Pi_c(x) = 1 - \Pi(x)$ provided we also negate the cut points.

density function we have a bijection between the interior cut points and the first $K-1$ ordinal probabilities, with the last ordinal probability then given by the simplex constraint,

$$p_K = 1 - \sum_{k'=1}^{K-1} p_{k'}.$$

We've already constructed the map from interior cut points to the first $K-1$ ordinal probabilities,

$$p_k = \begin{cases} \Pi(c_k), & k = 1 \\ \Pi(c_k) - \Pi(c_{k-1}), & 1 \leq k \leq K-1 \end{cases}.$$

The inverse mapping is straightforward once we recognize that if we sum over the first $k' < K$ ordinal probabilities then most of the cumulative distribution functions cancel,

$$\sum_{k'=1}^{k} p_{k'} = \sum_{k'=1}^{k} \Pi(c_{k'}) - \Pi(c_{k'-1}) = \Pi(c_k).$$

Consequently the kth interior cut point can be derived from the first k ordinal probabilities,

$$c_k = \Pi^{-1}\left(\sum_{k'=1}^{k} p_{k'} \right).$$

In fact this is really nothing more than the **stick-breaking** map that is often used to unconstrained simplices in probabilistic programming tools like Stan. Configuring the first interior cut point

$$c_1 \in (-\infty, +\infty)$$

is equivalent to configuring the first break of the stick,

$$p_1 \in [0, 1].$$

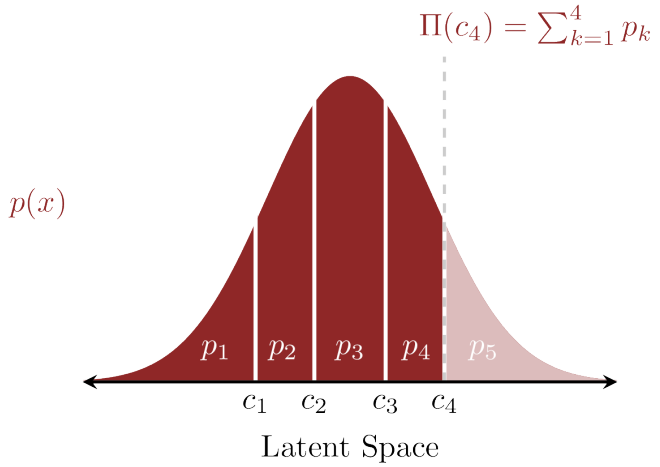


Figure 7: The cumulative distribution function evaluated at the kth cut point c_k is always the sum of the first k ordinal probabilities. Consequently we can compute any cut point by summing the appropriate ordinal probabilities and then inverting the cumulative distribution function.

Similarly configuring the second interior cut point

$$c_2 \in (c_1, +\infty)$$

is equivalent to the breaking the remaining stick,

$$p_2 \in [p_1, 1],$$

and so on,

$$c_k \in (c_{k-1}, +\infty) \iff p_k \in [p_{k-1}, 1].$$

Intuitively the interior cut points can *always* reconfigure themselves to compensate for any change in the latent probability density function and recover any configuration of the ordinal probabilities (Figure 8). Mechanically all we have to do is swap out the cumulative distribution function and its inverse. Consequently the ordinal probabilities will not, in general, inherit the shape of the latent probability density function.

One immediate consequence of this duality is that so long as the interior cut points are unconstrained we are free to choose a latent probability density function that simplifies the implementation. For example we can limit consideration to latent probability density functions with both analytic cumulative distribution functions and inverse cumulative distribution functions. Similarly we might make our choice based on computational cost considerations.

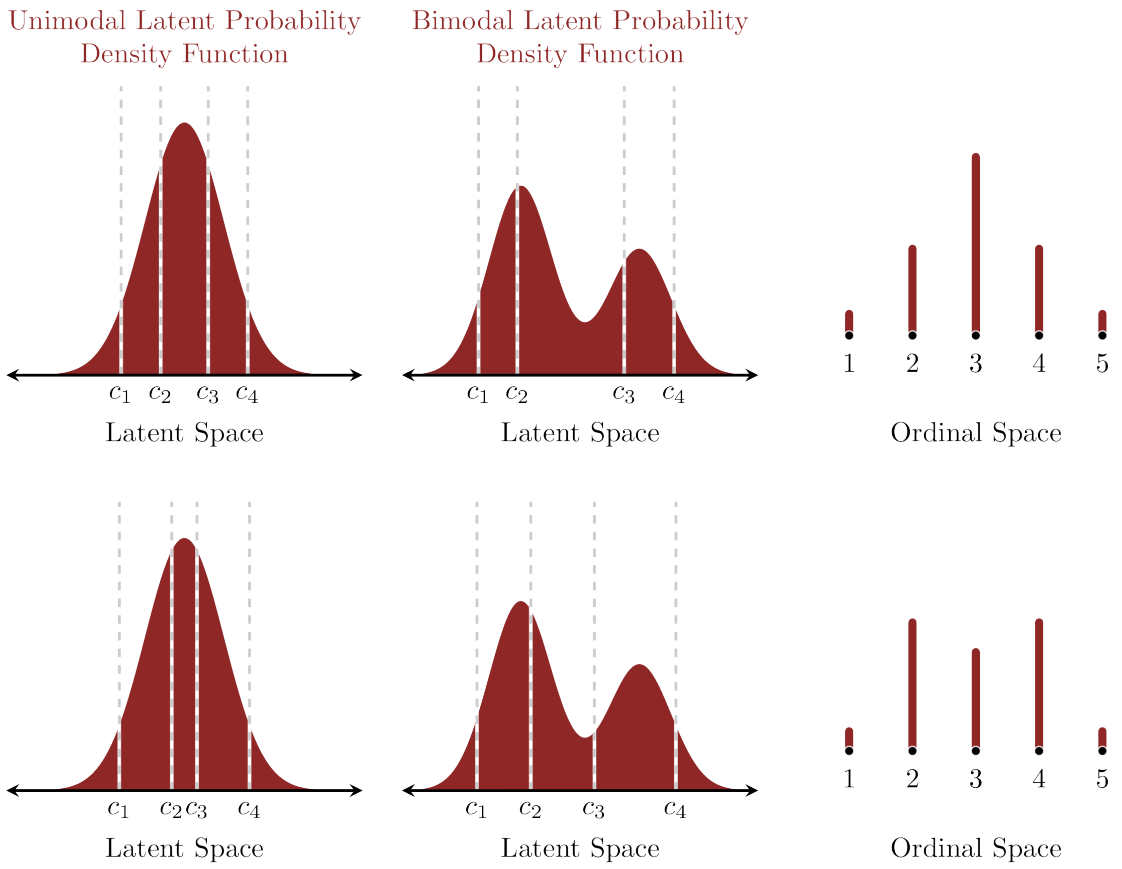


Figure 8: So long as the interior cut points are unconstrained the cut point construction can recover any ordinal probability configuration regardless of the shape of the latent probability density function. For example both a unimodal and bimodal latent probability density function can recover both unimodal and bimodal ordinal probabilities.

If we take a logistic probability density function

$$p(x; \mu, \sigma) = \frac{1}{\sigma} \frac{\exp\left(-\frac{x-\mu}{\sigma}\right)}{\left(1 + \exp\left(-\frac{x-\mu}{\sigma}\right)\right)^2}$$

then the cumulative distribution function is given by the analytic logistic function,

$$\Pi(x; \mu, \sigma) = \frac{1}{1 + \exp\left(-\frac{x-\mu}{\sigma}\right)},$$

and its inverse is given by the analytic logit function,

$$\Pi^{-1}(p; \mu, \sigma) = \mu + \sigma \log \frac{p}{1-p}.$$

These analytic forms make this model particularly convenient to implement in practice.

The terminology for these models isn't always consistent. Usually, however, they are named after the latent cumulative distribution function or its inverse. For example the above model is referred to as both the **ordered logit** model and the **ordered logistic** model.

Similarly an ordinal model based on a normal probability density function is commonly referred to as an **ordered probit** model, with "probit" a common name for the inverse normal cumulative distribution function. The probit function is available in many statistical programming environments but it is not quite as convenient as the logit function.

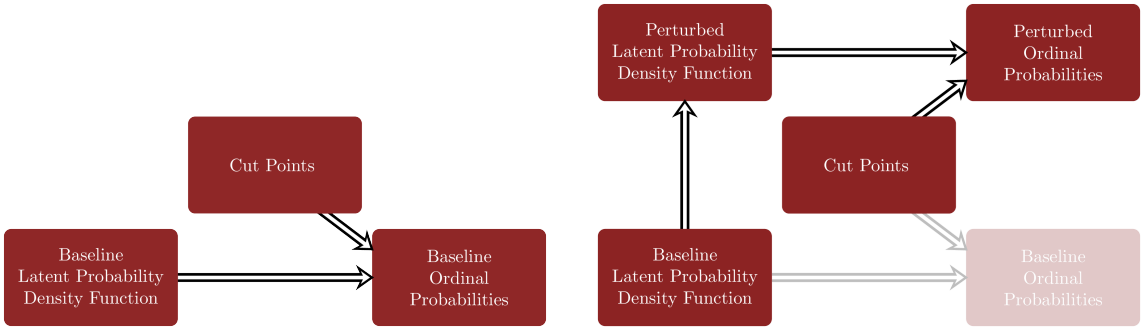
3.3 Perturbing Ordinal Probabilities

If the cut point construction is mathematically equivalent to directly modeling the ordinal probabilities then what, if any, benefit does this approach offer? The advantage appears only once we consider heterogeneous ordinal probabilities.

Given a fixed latent probability density function any constraint on baseline ordinal probabilities informs the interior cut points (Figure 9a). If we then fix the interior cut points but perturb this baseline probability distribution then the ordinal probabilities will vary in a systematic way (Figure 9b).

In order to increase p_k we have to increase latent probability density function $p(x)$ across the interval $(c_{k-1}, c_k]$ (Figure 10). The less flexible $p(x)$ is the more this will require also increasing $p(x)$ across the neighboring intervals, and hence increasing the neighboring ordinal probabilities p_{k-1} and p_{k+1}. Equivalently if we engineer $p(x)$ to decrease p_k then we will likely also decrease the neighboring probabilities p_{k-1} and p_{k+1}.

For example we might perturb the latent probability distribution by translating the latent real line to the right, shifting the latent probability density function towards smaller values. If the interior cut points are fixed then this pushes the baseline ordinal probabilities towards smaller ordinal values while preserving the baseline shape. Likewise translating X to the left shifts the



(a) (b)

Figure 9: The main utility of the cut point construction is to model systematic variations in ordinal probabilities. (a) Given a fixed baseline latent probability density function the cut points will inform the baseline ordinal probabilities. Consequently any inferences for the baseline ordinal probabilities will inform consistent cut point configurations. (b) Once the cut points have been informed we can perturb the latent probability density function around the baseline configuration to perturb the ordinal probabilities in a similar way.

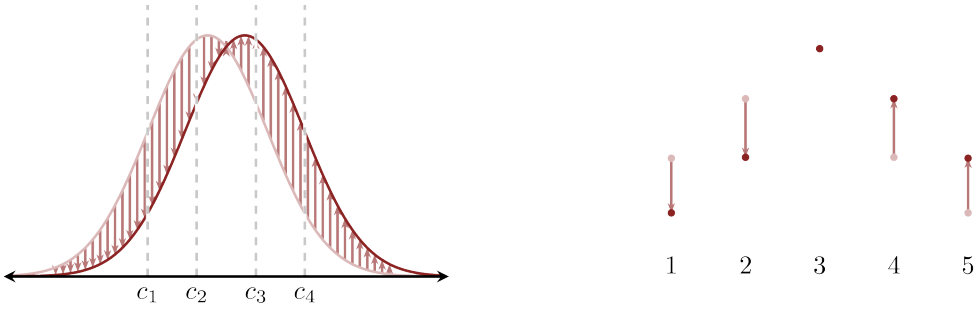


Figure 10: Perturbing the shape of the latent probability density function perturbs the ordinal probabilities in a similar way. If the latent probability density function mostly decreases or increases between two cut points then the corresponding ordinal probability will decrease or increases as well. The more rigid the latent probability density function is the more strongly neighboring ordinal probabilities will be coupled to each other.

latent probability density function towards larger values and pushes the ordinal probabilities to larger ordinal values (Figure 11). This can be useful when we want to model how varying conditions might consistently increase or decrease the sentiment of people answering a survey question.

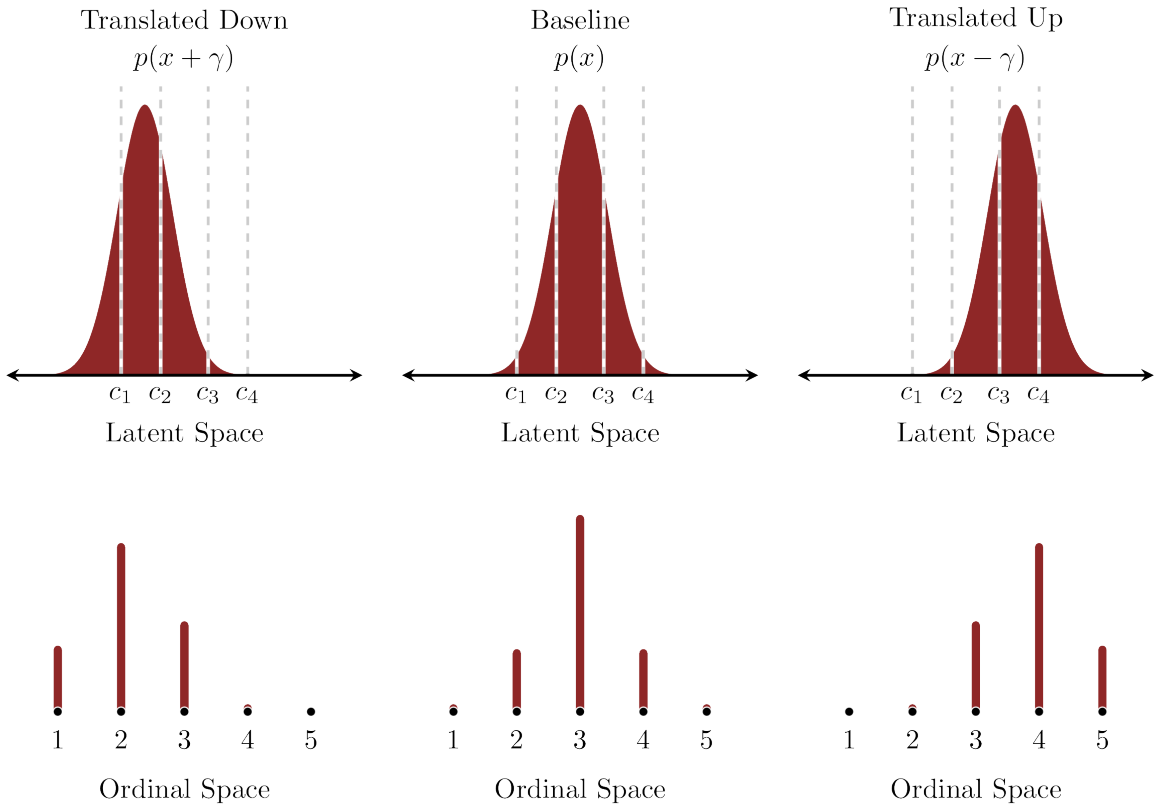


Figure 11: When the cut points are fixed translating the latent probability density function from its baseline configuration shifts the ordinal probabilities in the same direction.

Similarly we might dilate the latent real line, causing the latent probability density function and the resulting ordinal probabilities to narrow, or contract the latent real line to widen the latent probability density function and the resulting ordinal probabilities (Figure 12). This behavior can be useful when varying conditions might increase or decrease the consistency of the sentiment of survey responders.

3.4 Modeling Heterogeneity with Derived Cut Points

The general strategy for modeling heterogeneous ordinal probabilities is to

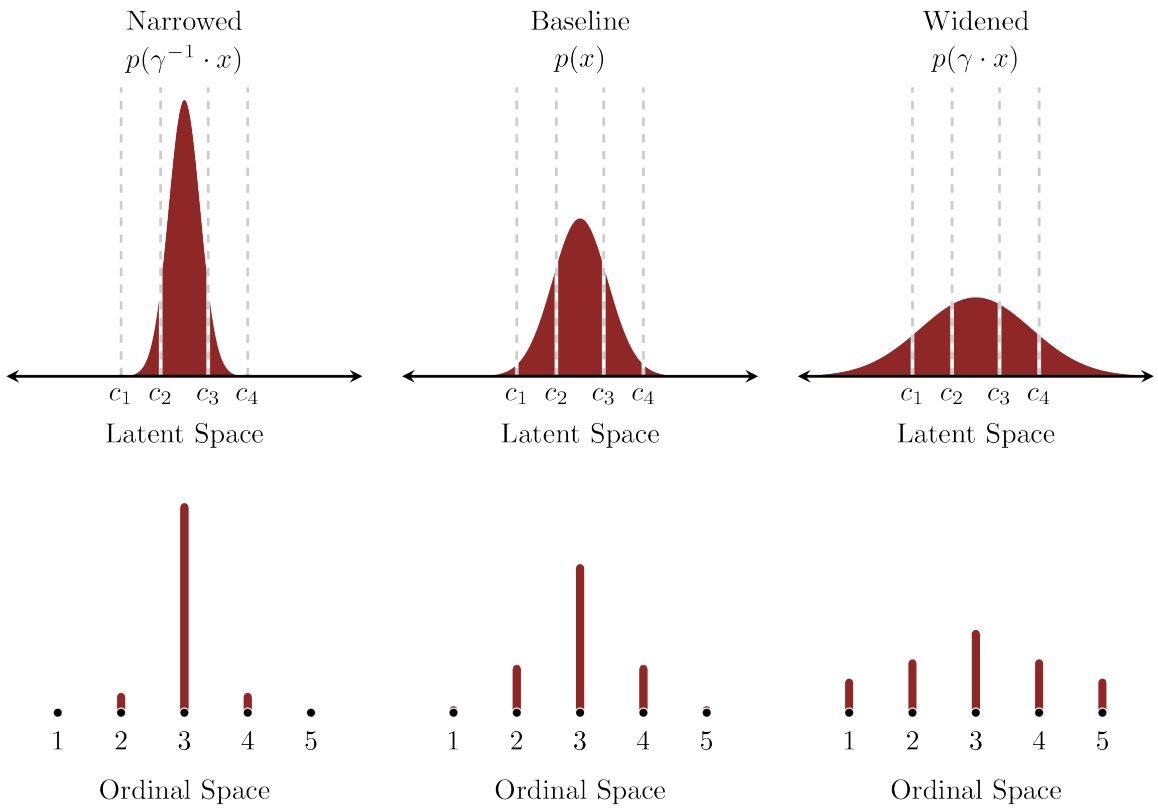


Figure 12: When the cut points are fixed dilating or contracting the latent probability density function from its baseline configuration narrows or widens the ordinal probabilities, respectively.

- fix a baseline latent probability density function,
- inform interior cut points from the ordinal probabilities,
- fix the interior cut points,
- vary the ordinal probabilities by varying the configuration of the latent probability density function.

The most direct way to implement this in practice is to model the interior cut points and the configuration of the latent probability density function, deriving the baseline and varying ordinal probabilities as needed.

That said we can use the duality between interior cut points and ordinal probabilities to implement this approach another way. Instead of modeling the interior cut points directly and deriving the baseline ordinal probabilities,

$$p_k = \Pi(c_k) - \Pi(c_{k-1}),$$

we can equivalently model the baseline ordinal probabilities and then derive the interior cut points,

$$c_k = \Pi^{-1}\left(\sum_{k'=1}^{k} p_{k'}\right).$$

Given these derived interior cut points we can then derive varying ordinal probabilities from varying latent probability density function configurations as before.

Mathematically these two approaches are equivalent. In practice, however, the interior cut points and baseline ordinal probabilities could exhibit different inferential degeneracies that could be more or less costly depending on the particular observed data and prior model.

Note that in most probabilistic programming tools simplex variables are handled by first mapping to an unconstrained space with something like the stick breaking map. Moreover as we saw in Section 3.2 the mapping from interior cut points to ordinal probabilities is very similar to stick breaking. Consequently any differences in the implementation of these models with either interior cut points or baseline ordinal probabilities should be small in practice, although there can always be exceptions.

3.5 Interpretations of the Cut Point Construction

In general the cut point construction does not need to model any particular data generating process. Rather it can be a heuristic technique to ensure reasonable variations in the ordinal probabilities. One nice advantage of this perspective is that, because the latent probability distribution isn't modeling any particular phenomenon, we can choose the possible latent probability density function configurations based on implementation convenience or computational performance.

That said we can use the cut point construction to model an explicit data generating process when appropriate. For example when ordinal data arises from the censoring of some unobserved, real-valued behavior then we can interpret the latent probability distribution as a model of this behavior with interior cut points modeling how that behavior is censored in observations.

This narratively generative perspective can be especially useful when trying to combine inconsistent data into a single analysis. Consider, for instance, two surveys with the possible responses

$$\text{Disagree}, \text{Neutral}, \text{Agree}$$

and

$$\text{Disagree}, \text{Agree}$$

respectively. Even if the survey responders interpret "Disagree" and "Agree" consistently between the two surveys it is not obvious to what responses in the second survey the "Neutral" responses in the first survey would correspond.

If we believe that the survey questions are eliciting some unobserved, real-valued behavior and we are comfortable modeling that behavior with a latent probability density function then we can model both surveys with two sets of cut points to accommodate the inconsistent survey designs. In this case the set for the first survey would contain two interior cut points while the set for the second survey would contain a single interior cut point.

4 Degeneracies of the Cut Point Construction

As with most useful modeling techniques, the cut point construction is vulnerable to some particular inferential degeneracies that can frustrate an analysis if we are not prepared to manage them.

4.1 Latent Non-identifiability

As we saw in Section 3.1 the cut point construction defines a map from $K - 1$ interior cut points into K ordinal probabilities given a fixed latent probability density function. This map is also invertible, allowing us to take K ordinal probabilities into $K - 1$ interior cut points, once again given a fixed latent probability density function. In other words a latent probability density function and interior cut points completely specify ordinal probabilities while a latent probability density function and ordinal probabilities completely specify interior cut points.

The cut point construction is actually a bit more general than this. Any configuration of the interior cut points and ordinal probabilities specify consistent latent probability density functions, although the specification is not unique in this case. Constraining any *two* of the three objects in the cut point construction always informs the third.

If we specify only *one* of these objects, however, then there will be infinitely many compatible configurations for the other two. Without care this redundancy can lead to non-identified ordinal models which in turn result in frustrating inferential degeneracies.

To demonstrate what can go wrong let's say that we do not completely fix the latent probability density function but instead limit it to a family of translated probability density functions,

$$p_\delta(x) = p_0(x + \delta)$$

for some initial probability density function p_0. This translation family then defines the cumulative distribution functions

$$\begin{aligned}
\Pi_\delta(x) &= \int_{-\infty}^{x} \mathrm{d}x' \, p_\delta(x') \\
&= \int_{-\infty}^{x} \mathrm{d}x' \, p_0(x' + \delta) \\
&= \int_{-\infty}^{x+\delta} \mathrm{d}z' \, p_0(z') \\
&= \Pi_0(x + \delta)
\end{aligned}$$

and hence the ordinal probabilities

$$\begin{aligned}
p_k &= \Pi_\delta(c_k) - \Pi_\delta(c_{k-1}) \\
&= \Pi_0(c_k + \delta) - \Pi_0(c_{k-1} + \delta).
\end{aligned}$$

Notice, however, that we get the exact same ordinal probabilities if we set the translation to zero but shift all of the interior cut points by the same amount,

$$\begin{aligned}
p_k &= \Pi_{\delta=0}(c'_k) - \Pi_{\delta=0}(c'_{k-1}) \\
&= \Pi_{\delta=0}(c_k + \delta) - \Pi_{\delta=0}(c_{k-1} + \delta) \\
&= \Pi_0(c_k + \delta) - \Pi_0(c_{k-1} + \delta).
\end{aligned}$$

In other words for each δ we have two equivalent ways of reproducing the same ordinal probabilities.

Similarly translating the latent probability density function and interior cut points by any amount but in *opposite* directions always returns the same initial ordinal probabilities (Figure 13),

$$\begin{aligned}
p_k &= \Pi_\delta(c'_k) - \Pi_\delta(c'_{k-1}) \\
&= \Pi_\delta(c_k - \delta) - \Pi_\delta(c_{k-1} - \delta) \\
&= \Pi_0(c_k - \delta + \delta) - \Pi_0(c_{k-1} - \delta + \delta) \\
&= \Pi_0(c_k) - \Pi_0(c_{k-1}).
\end{aligned}$$

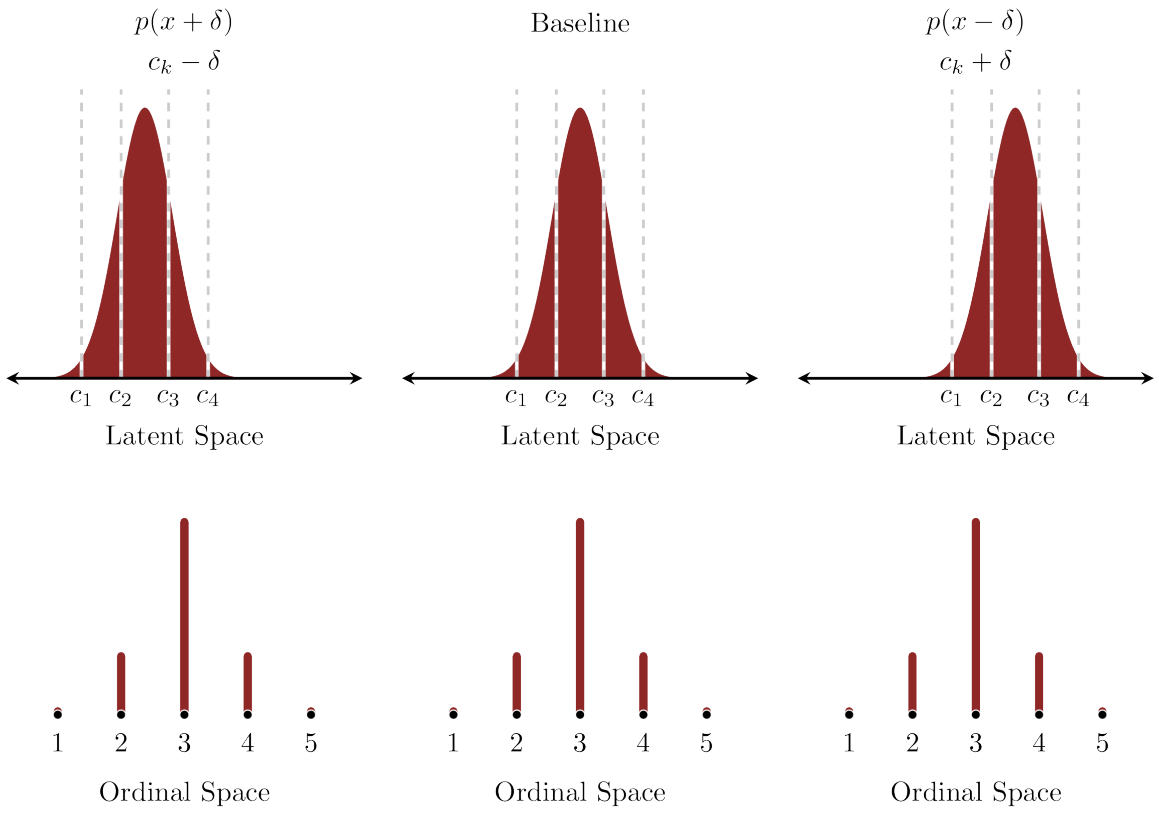


Figure 13: Any change to the latent probability density function can be compensated by a complementary change to the interior cut points so that the ordinal probabilities remain the same. Consequently inferences for the ordinal probabilities cannot inform the behavior of the latent probability density function and interior cut points at the same time.

Consequently any constraint on the ordinal probabilities alone, such as from observed data or domain expertise encoded into a prior model, cannot disentangle the behavior of the latent probability density function and the interior cut points *at the same time*.

This degeneracy between the behavior of the latent probability density function and the interior cut points extends far beyond just translations, although the math becomes a bit more involved.

Consider a family of latent probability density functions, each element of which is generated by mapping the latent real line into itself with an order-preserving function. We can then write each latent probability density function in the family as

$$p_f(x) = p_0\left(f^{-1}(x)\right) \left| \frac{\mathrm{d}f}{\mathrm{d}x}\left(f^{-1}(x)\right) \right|^{-1}$$

for some initial probability density function p_0 and any bijective, monotonically increasing function

$$f : \mathbb{R} \to \mathbb{R}.$$

The cumulative distribution function for any member of this family is then given by

$$\begin{aligned}
\Pi_f(x) &= \int_{-\infty}^{x} \mathrm{d}x' \, p_f(x') \\
&= \int_{-\infty}^{x} \mathrm{d}x' \, p_0\left(f^{-1}(x')\right) \left| \frac{\mathrm{d}f}{\mathrm{d}x}\left(f^{-1}(x')\right) \right|^{-1} \\
&= \int_{-\infty}^{f^{-1}(x)} \mathrm{d}x'' \, p_0\left(x''\right) \\
&= \Pi_0\left(f^{-1}(x)\right).
\end{aligned}$$

Given a function f the ordinal probabilities become

$$\begin{aligned}
p_k &= \Pi_f(c_k) - \Pi_f(c_{k-1}) \\
&= \Pi_0\left(f^{-1}(c_k)\right) - \Pi_0\left(f^{-1}(c_{k-1})\right).
\end{aligned}$$

At the same time if we use the initial probability density function p_0 but transform the interior cuts points using the inverse of f,

$$c'_k = f^{-1}(c_k),$$

then the resulting ordinal probabilities are exactly the same as above,

$$\begin{aligned}
p_k &= \Pi_0\left(c'_k\right) - \Pi_0\left(c'_{k-1}\right) \\
&= \Pi_0\left(f^{-1}(c_k)\right) - \Pi_0\left(f^{-1}(c_{k-1})\right).
\end{aligned}$$

For each order-preserving function f we have two equivalent ways of reproducing the same ordinal probabilities!

We can also always transform the latent probability density function and interior cut points in complementary ways to achieve the same initial ordinal probabilities,

$$\begin{aligned} p_k &= \Pi_f\left(c'_k\right) - \Pi_f\left(c'_{k-1}\right) \\ &= \Pi_f\left(f(c_k)\right) - \Pi_f\left(f(c_{k-1})\right) \\ &= \Pi_0\left(f^{-1}(f(c_k))\right) - \Pi_0\left(f^{-1}(f(c_{k-1}))\right) \\ &= \Pi_0(c_k) - \Pi_0(c_{k-1}). \end{aligned}$$

In order to avoid these degeneracies we have to constrain, if not outright fix, any two of the three objects in the cut point construction at any given time. That said we don't have to constrain the *same* two objects across an entire analysis. Indeed activating different objects at different times is what allows us to model heterogeneity in the ordinal probabilities.

For example when modeling baseline behavior we have to define a fixed baseline latent probability density function so that any constraints on the baseline ordinal probabilities will inform the interior cut points. Because both the latent probability density function and ordinal probabilities are constrained there are no identifiability concerns.

When modeling behavior outside of the baseline we use these baseline inferences to constrain the interior cut points. Any constraints on the heterogeneous ordinal probabilities then inform variations from the baseline latent probability density function. In this case the interior cut points and ordinal probabilities are constrained, eliminating any identifiability concern for the variation of the latent probability density function.

For a specific example let's say that we want to allow the latent probability density function to be able to shift in different conditions so that the resulting ordinal probabilities coherently move towards smaller and larger ordinal values. We can achieve this with the translation family that we introduced above,

$$p_\delta(x) = p_0(x + \delta).$$

To avoid any identifiability issues we have to fix δ to a particular value when modeling the baseline probabilities. This value is arbitrary, but often it's convenient to take $\delta = 0$. Once we have fixed δ we can then use any baseline data and available domain expertise to inform the consistent behaviors of interior cut points. Those inferred interior cut point behaviors then allow us to model observations beyond the baseline context by letting δ vary to accommodate any systematic variations in the ordinal probabilities.

Altogether we end up with well-behaved joint inferences for the common interior cut points and a δ_i for each individual context.

4.2 Tail Degeneracies

Even when the latent probability density function is fixed interior cut points can suffer from another, although much more circumstantial, degeneracy.

The kth ordinal probability depends on the $(k-1)$st and kth cut points. We can quantify the strength of these dependencies with the corresponding partial derivatives,

$$\begin{aligned}
\frac{\partial p_k}{\partial c_k}(c_1, \ldots, c_K) &= \frac{\partial}{\partial c_k} \left(\Pi(c_k) - \Pi(c_{k-1}) \right) \\
&= \frac{\mathrm{d}\Pi}{\partial x}(c_k) \\
&= p(c_k)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial p_k}{\partial c_{k-1}}(c_1, \ldots, c_K) &= \frac{\partial}{\partial c_{k-1}} \left(\Pi(c_k) - \Pi(c_{k-1}) \right) \\
&= -\frac{\mathrm{d}\Pi}{\partial x}(c_{k-1}) \\
&= -p(c_{k-1}).
\end{aligned}$$

In words p_k is least sensitive to changes in c_{k-1} when $p(c_{k-1})$ is small, and it is least sensitive to changes in c_k when $p(c_k)$ is small. An immediate consequence of this is that if $p(c_k)$ is small then any constraints on p_k and p_{k+1} will only poorly inform c_k.

When the latent probability density function is unimodal, for example in the ordered logit and ordered probit models, then $p(x)$ is smallest in the tails as x approaches $-\infty$ or $+\infty$. Unfortunately it's also in the tails that these degeneracies can become particularly problematic.

Consider, for example, the first ordinal $k = 1$. If there are no observations of this particular value then are inferences will be able to suppress only larger values of the ordinal probability p_1,

$$\begin{aligned}
0 \leq \quad p_1 \quad &\lessapprox \epsilon \\
0 \leq \Pi(c_1) &\lessapprox \epsilon.
\end{aligned}$$

In theory c_1 should also be informed by p_2, but if $p(c_1)$ is small then inferences for c_1 will be limited to the consequences of this upper bound,

$$\begin{aligned}
0 \leq \Pi(c_1) &\lessapprox \epsilon \\
\Pi^{-1}(0) \leq \quad c_1 \quad &\lessapprox \Pi^{-1}(\epsilon) \\
-\infty < \quad c_1 \quad &\lessapprox \Pi^{-1}(\epsilon).
\end{aligned}$$

In this case arbitrary negative values of c_1 will be consistent with the observed data, with the likelihood function remaining non-zero even as c_1 approaches minus infinity (Figure 14). Without any compensating prior information this degeneracy will then manifest in the posterior distribution, tempting Markov chains into burning endless computation in an attempt to chase that infinity.

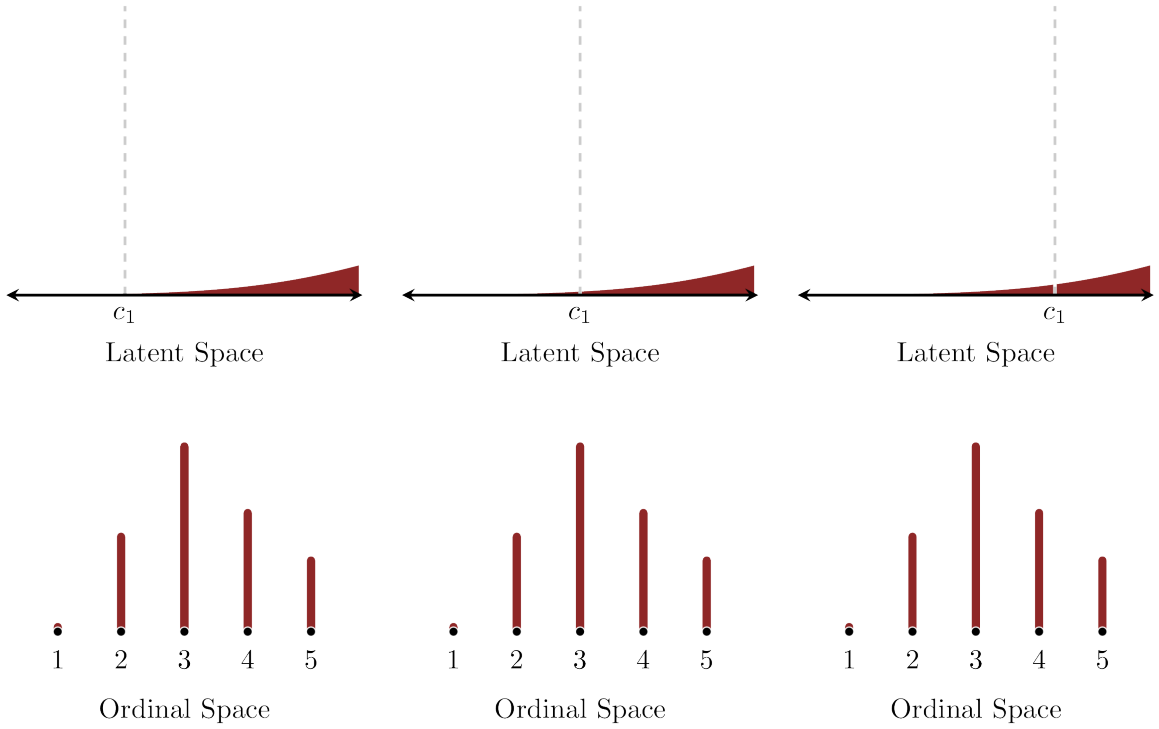


Figure 14: When the first ordinal probability p_1 is close to zero the first interior cut point p_1 will typically fall into the tail of the latent probability density function. Here even large changes to c_1 will have only a negligible impact on p_1 and the neighboring p_2. In this case arbitrarily small values of c_1 will be consistent with most observations no matter the behavior of p_2.

A similar problem can arise for the last ordinal. If we don't have any observations for $k = K$ then the observed data will be able to only upper bound p_K,

$$0 \leq p_K \lessapprox \epsilon.$$

If $p(c_{K-1})$ is small then the best constraint on c_{K-1} will be an approximate lower bound,

$$\begin{aligned} 0 \leq & \quad p_K \quad & \lessapprox \epsilon \\ 0 \leq & \; 1 - \Pi(c_{K-1}) & \lessapprox \epsilon \\ 1 - \epsilon \lessapprox & \; \Pi(c_{K-1}) & \leq 1 \\ \Pi^{-1}(1 - \epsilon) \lessapprox & \quad c_{K-1} & \leq \Pi^{-1}(1) \\ \Pi^{-1}(1 - \epsilon) \lessapprox & \quad c_{K-1} & < +\infty, \end{aligned}$$

resulting in a likelihood function that persists all the way to positive infinity.

The best way to safeguard against either of these tail degeneracies is to complement the ordinal model with an informative prior model that suppresses cut point configurations near infinity. Coincidentally we'll discuss a convenient method for this in the next section.

5 Informing Cut Points with Domain Expertise

Because interior cut points are defined only relative to a latent probability density function they are not the most interpretable objects. This lack of interpretability can limit our ability to elicit useful domain expertise which then frustrates any attempt at principled prior modeling.

Ordinal probabilities, however, enjoy a much more direct interpretation that facilitates prior modeling. For example we might engineer a Dirichlet prior

$$p(p_1, \ldots, p_K) = \text{Dirichlet}(p_1, \ldots, p_K \mid \alpha_1, \ldots, \alpha_K)$$

compatible with our domain expertise.

Taking

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

yields a uniform prior density function over all possible ordinal probabilities. On the other hand taking

$$\alpha_k = \rho_k / \tau + 1$$

gives a prior model that concentrates around the ordinal probabilities $(\rho_1, \ldots, \rho_K)$ with τ controlling the strength of that concentration.

Now given a fixed latent probability density function the ordinal probabilities completely determine the interior cut points and vice versa. Consequently once the latent probability density function is fixed we can to translate any prior model over the ordinal probabilities into an *induced* prior model over the interior cut points. More formally we can push the prior model forward from the simplex of ordinal probabilities to the space of interior cut points.

Consider a fixed latent probability density function p and cumulative distribution function

$$\Pi(x) = \int_{-\infty}^{x} \mathrm{d}x' \, p(x').$$

Given the ordinal probabilities

$$\mathbf{p} = (p_1, \ldots, p_K)$$

the interior cut points

$$\mathbf{c} = (c_1, \ldots, c_{K-1})$$

are given by the components

$$c_k = (\phi(\mathbf{p}))_k = \Pi^{-1}\left(\sum_{k'=1}^{k} p_{k'} \right)$$

with the inverse components

$$p_k = (\phi^{-1}(\mathbf{c}))_k = \Pi(c_k) - \Pi(c_{k-1}).$$

Any probability density function over the ordinal probabilities $p(\mathbf{p})$ then pushes forward to an equivalent probability density function over the interior cut points

$$p(\mathbf{c}) = p\left(\mathbf{p} = \phi^{-1}(\mathbf{c})\right) \cdot \left| \det \mathbf{J}_{\phi^{-1}}(\mathbf{c}) \right|.$$

The elements of the Jacobian matrix $\mathbf{J}_{\phi^{-1}}(\mathbf{c})$ are defined by the partial derivatives

$$J_{k,k'}(\mathbf{c}) = \frac{\partial \phi_k^{-1}}{\partial c_{k'}}(\mathbf{c}).$$

Because each p_k depends on only two neighboring cut points this results in a banded, upper-diagonal matrix. On the diagonal we have

$$J_{k,k}(\mathbf{c}) = \frac{\partial \phi_k^{-1}}{\partial c_k}(\mathbf{c}) = \frac{\mathrm{d}\Pi}{\mathrm{d}x}(c_k) = p(c_k).$$

Similarly just above the diagonal we have

$$J_{k,k+1}(\mathbf{c}) = \frac{\partial f_k^{-1}}{\partial c_{k+1}}(\mathbf{c}) = -\frac{\mathrm{d}\Pi}{\mathrm{d}x}(c_{k+1}) = -p(c_k),$$

For example if $K = 5$ then the Jacobian matrix would be the four by four matrix

$$\mathbf{J}_{\phi^{-1}}(\mathbf{c}) = \begin{pmatrix} p(c_1) & -p(c_2) & 0 & 0 \\ 0 & p(c_2) & -p(c_3) & 0 \\ 0 & 0 & p(c_3) & -p(c_4) \\ 0 & 0 & 0 & p(c_4) \end{pmatrix}.$$

Conveniently the determinant of an upper-diagonal matrix is just the product of its diagonal elements,

$$\det \mathbf{J}_{\phi^{-1}}(\mathbf{c}) = \prod_{k=1}^{K-1} p(c_k).$$

When implementing a log probability density function this simplifies even further to

$$\log \left| \det \mathbf{J} \left(\phi^{-1}(\mathbf{c}) \right) \right| = \sum_{k=1}^{K-1} \log p(c_k).$$

If our domain expertise is consistent with the Dirichlet prior model

$$\text{Dirichlet} \left(\mathbf{p} \mid \alpha_1, \ldots, \alpha_K \right)$$

then the equivalent prior model over the interior cut points is given by

$$p(\mathbf{c}) = \text{Dirichlet} \left(\phi^{-1}(\mathbf{c}) \mid \alpha_1, \ldots, \alpha_K \right) \cdot \left| \prod_{k=1}^{K-1} p(c_k) \right|.$$

I will refer to this as the **induced Dirichlet model**.

Keep in mind that any induced prior model over the interior cut points is defined relative to an appropriate latent probability density function. If that latent probability density function is not consistent with the rest of the model then the induced prior model will not capture the desired domain expertise. For instance if we're modeling heterogeneous ordinal probabilities and we have domain expertise about the baseline ordinal probabilities then we have to use the baseline latent probability density function to define the induced prior model.

Lastly if, as discussed in Section 3.4, we're parameterizing the ordinal model in terms of baseline ordinal probabilities and then deriving the interior cut points then we can just directly use any prior model for those ordinal probabilities.

6 Ordinal Pairwise Comparison Modeling

One common application of the cut point construction is modeling bipartide pairwise comparisons with ordinal outputs. That said this particular application introduces a few subtleties that warrant some careful consideration.

6.1 Homogeneous Cut Points

In these applications we are usually interested in quantifying how individual **appraiser** items express ordinal sentiments about **appraisee** items. The quality of an appraisee, γ_j, quantifies its general appeal while the quality of an appraiser, α_i, determines how austere their responses tend to be.

We can then use the difference in these item qualities to translate a latent probability function relative to fixed cut points,

$$p_{ij}(x) = p_0(x - (\gamma_j - \alpha_i)),$$

resulting in the varying ordinal probabilities

$$p_k = \Pi_0(c_k - (\gamma_j - \alpha_i)) - \Pi_0(c_{k-1} - (\gamma_j - \alpha_i)).$$

The configuration of the cut points determines the shape of the baseline ordinal probabilities which models how the appraisers translate their complex sentiments into the available ordinal values.

When the ith appraiser's austerity is larger than the jth appraisee's appeal then

$$\alpha_i > \gamma_j$$

and

$$\gamma_j - \alpha_i < 0.$$

In this case the latent probability density function, and the resulting ordinal probabilities, shift towards smaller ordinal values, consistent with a more negative response.

Conversely if the appraisee's appeal is larger than the appraiser's austerity then

$$\gamma_j > \alpha_i$$

and

$$\gamma_j - \alpha_i > 0.$$

This causes the ordinal probabilities to shift towards larger ordinal values, consistent with a more positive response.

Now as we saw in Section 4.1 we need to bind the interior cut points to a specific baseline latent probability density function in order to avoid inferential degeneracies. In this pairwise comparison setting the natural baseline is $p_0(x)$ which corresponds to a comparison between two equally matched items,

$$\gamma_{j_b} - \alpha_{i_b} = 0,$$

or

$$\alpha_{i_b} = \gamma_{j_b}.$$

Note that the items i_b and j_b do not have to correspond to any observed items. This baseline comparison can be, and in practice often is, completely hypothetical.

We're not quite out of the woods yet. To avoid a degeneracy between the two types of item qualities we also need to anchor at least one item quality to zero. For instance we might anchor the baseline appraiser austerity to zero,

$$\gamma_{j_b} = 0.$$

In tandem with the previous constraint this would also imply

$$\alpha_{i_b} = 0.$$

6.2 Heterogeneous Cut Points

All of this said, in many applications the translation of implicit, qualitative sentiment into explicit, quantitative ordinal value is not universal but rather idiosyncratic to each appraiser. Some appraisers might be more generous with their responses than others, while some might exhibit more variability in their responses than others.

If we want to model these heterogeneous baseline behaviors then we need to use a separate set of interior cut points for *each* appraiser, $\mathbf{c}_i$. This, however, results in comparisons that always consider the same appraiser austerity α_i and the same set of interior cut points $\mathbf{c}_i$, with only the appraisee appeals γ_j varying across comparisons.

This means that to avoid inferential degeneracies we have to set α_i to zero for *every* appraiser. The latent probability density function configurations for each comparison are then given by

$$p_{ij}(x) = p_0(x - \gamma_j),$$

with the corresponding the ordinal probabilities

$$p_{i,k} = \Pi_0(c_{i,k} - \gamma_j) - \Pi_0(c_{i,k-1} - \gamma_j).$$

Intuitively the appraiser austerities have been effectively "absorbed" into the appraiser interior cut points.

Many introductions to ordinal modeling start with this final result, obscuring the underlying pairwise comparison model and the baseline assumptions we have made along the way. For example without an understanding of this derivation it may not be obvious that the appraisee appeals γ_j have no absolute meaning on their own, but rather can be interpreted only relative to each other.

7 Demonstrations

Now let's put all of this theory into practice with a series of demonstrative examples.

7.1 Set up

First we need to setup the 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)
```

7.2 Homogeneous Ordinal Probabilities I

Let's start by modeling a single set of ordinal probabilities.

7.2.1 Data Exploration

Our first data set consists of 50 observations, each of which can take one of five ordinal values.

```
data <- read_rdump('data/1.data.R')
```

```
cat(sprintf('%i ordinal values', data$K))
```

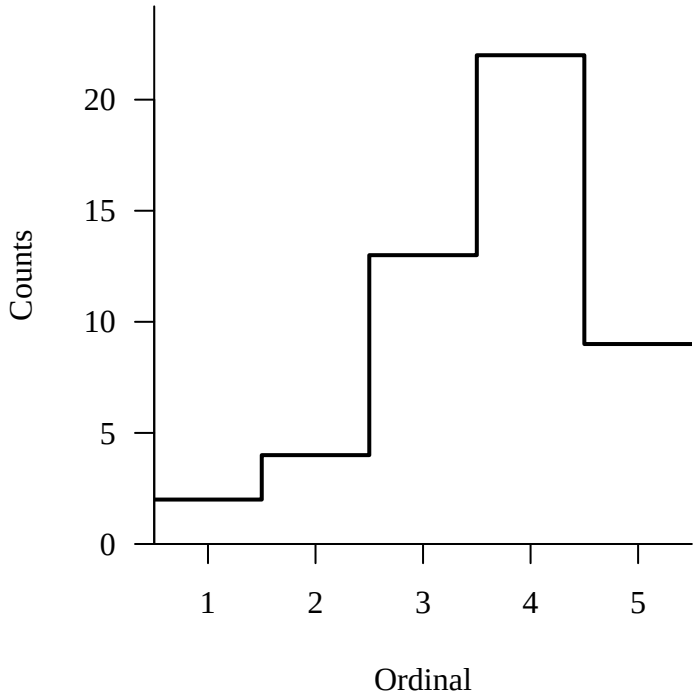
```
5 ordinal values
```

```
cat(sprintf('%i total observations', data$N))
```

```
50 total observations
```

The ordinal data are neatly summarized with a histogram.

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

util$plot_line_hist(data$y, 0.5, 5.5, 1,
                    xlab="Ordinal")
```



7.2.2 Categorical Model

We begin with a categorical model that ignores the ordering and a prior model that is uniform across the possible simplex configurations.

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

The computational diagnostics are 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.
```

Stan
Program 1 `categorical.stan`

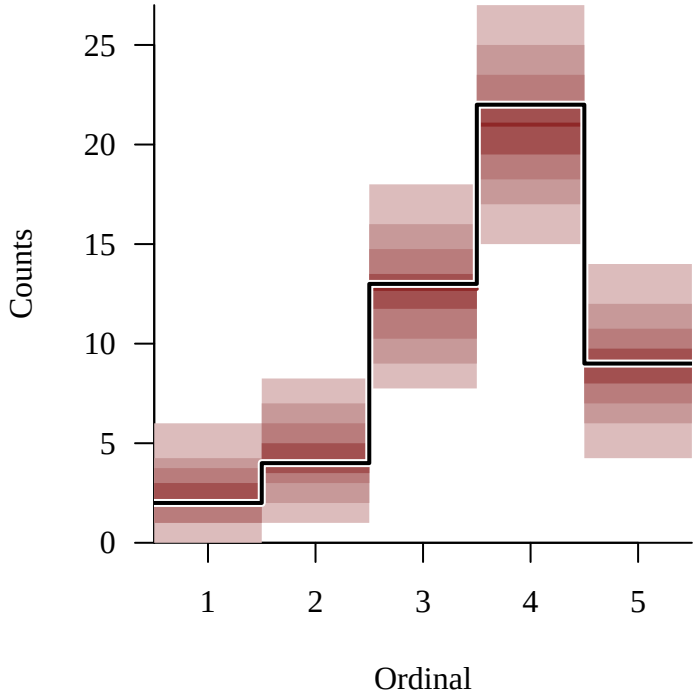
```
data {
  int<lower=1> K;                   // Number of ordinal categories
  int<lower=1> N;                   // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categoriesObserved categories
}

parameters {
  simplex[K] p; // Category probabilities
}
model {
  // Prior model
  p ~ dirichlet(rep_vector(1, K));

  // Observational model
  y ~ categorical(p);
}

generated quantities {
  array[N] int<lower=1, upper=K> y_pred;
  for (n in 1:N)
    y_pred[n] = categorical_rng(p);
}
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('p'),
                                       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.
```

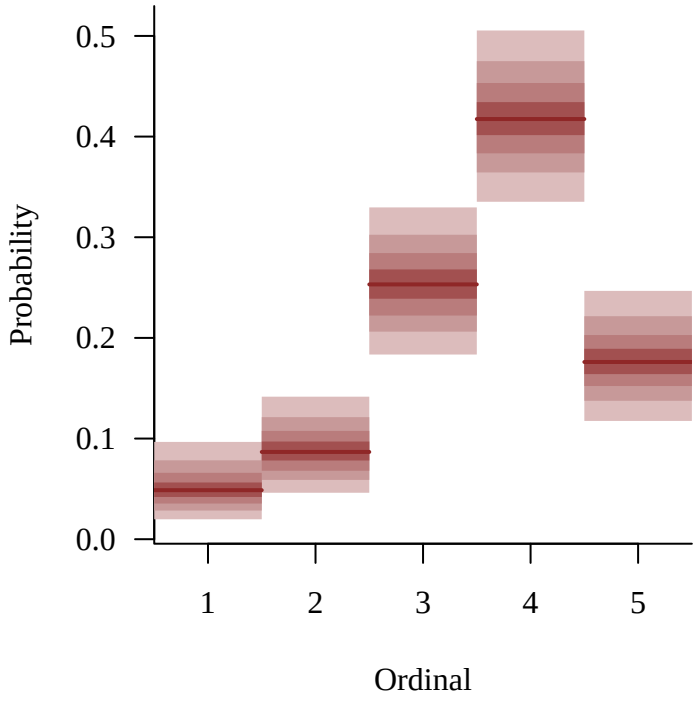
Moreover our inferred predictions don't show any sign of retrodictive tension with the observed data.

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

util$plot_hist_quantiles(samples1, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



Consequently the inferred categorical probabilities should adequately model these data.

```
names <- sapply(1:data$K, function(k) paste0('p[', k, ']'))
util$plot_disc_pushforward_quantiles(samples1, names,
                                     xlab="Ordinal",
                                     ylab="Probability")
```



7.2.3 Derived Cut Points

Equivalently we can implement the categorical model by deriving interior cut points from the categorical probabilities and then using an ordinal model. When modeling homogeneous ordinal probabilities these two steps always return the initial categorical probabilities so long as we consistently use the same logistic density function for both.

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

None of the computational diagnostics are complaining.

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

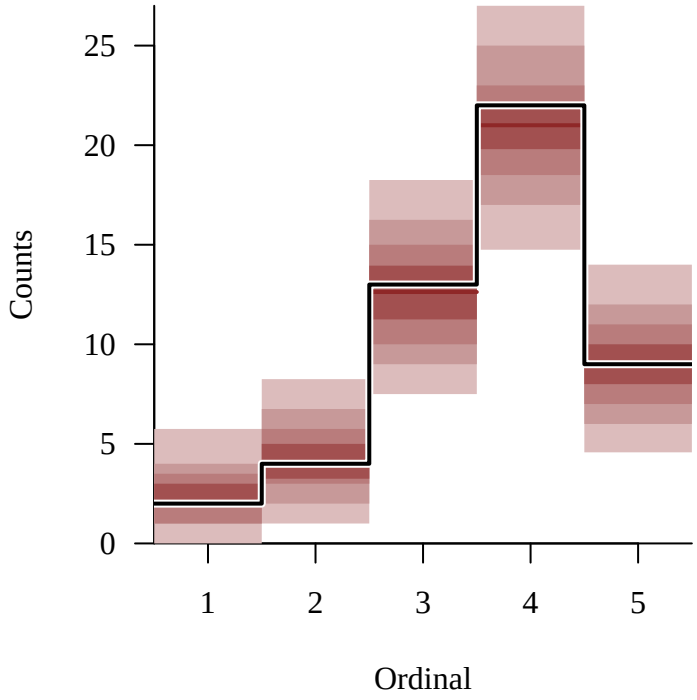
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('p'),
                                       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.
```

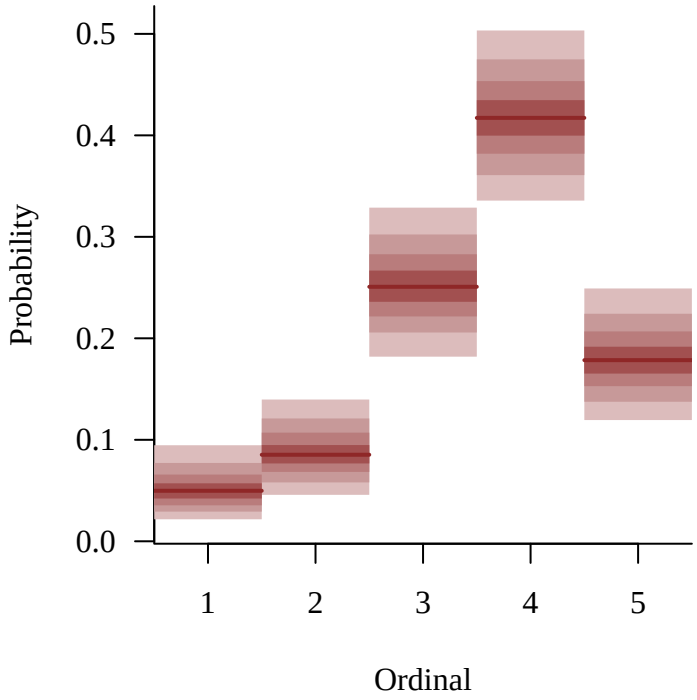
Retrodictive performance is similar to that of the previous model.

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

util$plot_hist_quantiles(samples2, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



So too are the posterior inferences.

```
names <- sapply(1:data$K, function(k) paste0('p[', k, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     xlab="Ordinal",
                                     ylab="Probability")
```

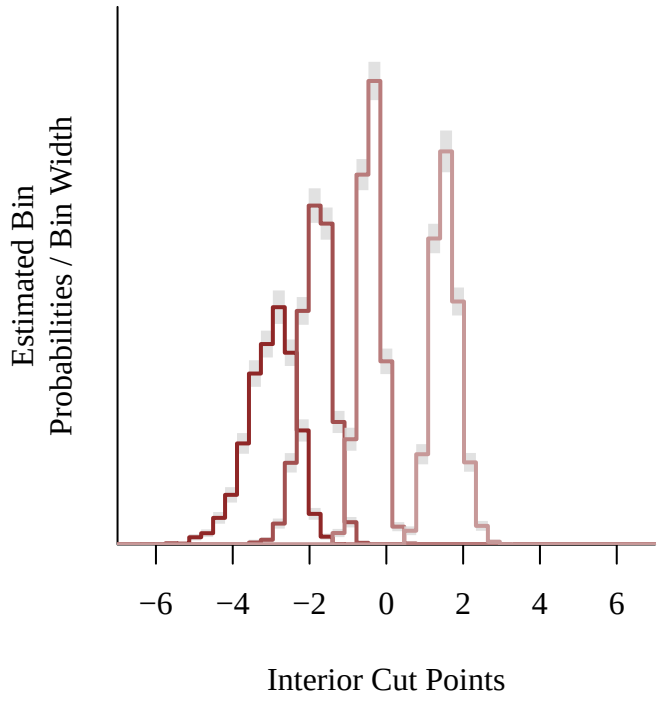


The inferred behaviors of the interior cut point behaviors can be visualized by overlaying the marginal posterior distributions over each other.

```
plot_cut_point_overlay <- function(expectand_vals_list, prefix,
                                   flim, fname, ylim, main=NULL) {
  name <- paste0(prefix, 1, ']')
  util$plot_expectand_pushforward(expectand_vals_list[[name]],
                                  45, flim=flim, display_name=fname,
                                  col=util$c_dark, border="#DDDDDDDD",
                                  ylim=ylim, main=main)
  name <- paste0(prefix, 2, ']')
  util$plot_expectand_pushforward(expectand_vals_list[[name]],
                                  45, flim=flim,
                                  col=util$c_mid_highlight,
                                  border="#DDDDDDDD",
                                  add=TRUE)
  name <- paste0(prefix, 3, ']')
  util$plot_expectand_pushforward(expectand_vals_list[[name]],
                                  45, flim=flim,
                                  col=util$c_mid,
                                  border="#DDDDDDDD",
                                  add=TRUE)
  name <- paste0(prefix, 4, ']')
```

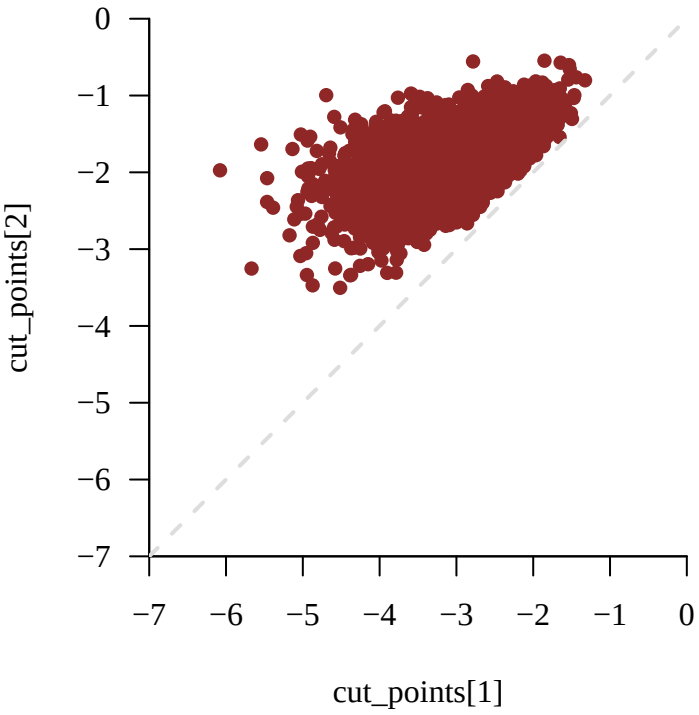
```
  util$plot_expectand_pushforward(expectand_vals_list[[name]],
                                  45, flim=flim,,
                                  col=util$c_light_highlight,
                                  border="#DDDDDDDD",
                                  add=TRUE)
}
```

```
plot_cut_point_overlay(samples2, 'cut_points[',
                       flim=c(-7, 7), fname='Interior Cut Points',
                       ylim=c(0, 1.5))
```



Note that the overlap of the marginal posterior distributions here is not a violation of the ordering constraint but rather an artifact of ignoring the coupling between the interior cut point inferences. For example pairs plots show that the ordering constraint is satisfied for all of the posterior samples.

```
plot(c(samples2[['cut_points[1]']]),
     c(samples2[['cut_points[2]']]),
     pch=16, cex=1, col=util$c_dark,
     xlim=c(-7, 0), xlab='cut_points[1]',
     ylim=c(-7, 0), ylab='cut_points[2]')
abline(a=0, b=1, lwd=2, lty=2, col="#DDDDDD")
```



7.2.4 Modeled Cut Points

Finally let's consider modeling the interior cut points directly and deriving the ordinal probabilities. We'll start by being a bit sloppy and using an ill-defined prior model that attempts to be uniform over the interior cut points. Critically this is *not* the same as the uniform simplex prior model that we used above!

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

Fortunately it looks like our sloppiness did not result in any computational problems.

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

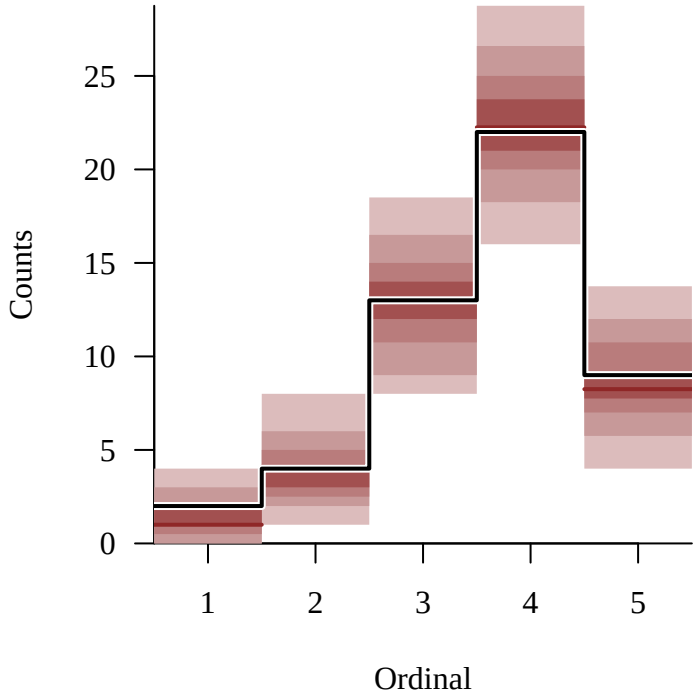
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples3 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples3,
                                       c('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.
```

Nor did it compromise the retrodictive performance.

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

util$plot_hist_quantiles(samples3, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```

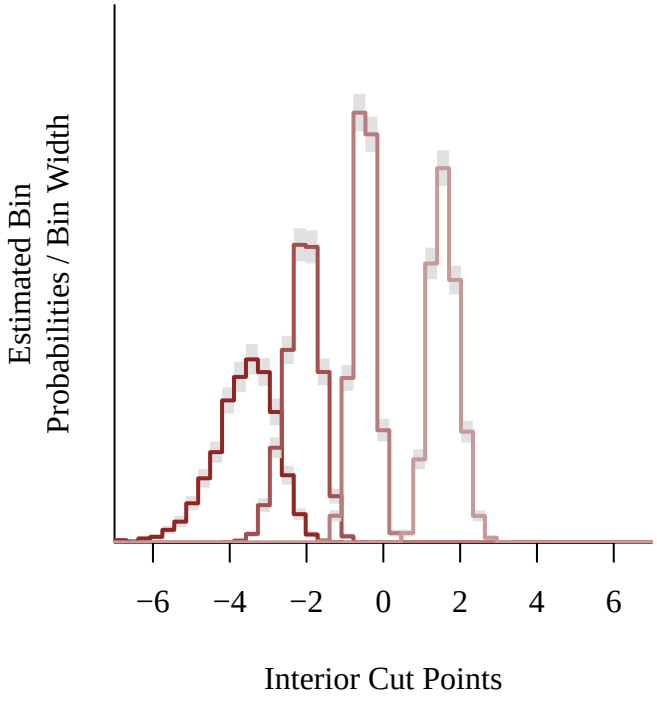


Despite the sloppy prior model the data are able to strongly constrain the interior cut points behaviors.

```
plot_cut_point_overlay(samples3, 'cut_points[',
                       flim=c(-7, 7), fname='Interior Cut Points',
                       ylim=c(0, 1.5))
```

```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 28
values (0.7%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 0
values (0.0%) fell above the histogram binning.
```



If we want to ensure exact compatibility with the first and second models that we considered above then we'll need to translate the uniform simplex prior model into an equivalent prior model over the interior cut points. Fortunately the induced Dirichlet prior model does just this.

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

The computational diagnostics are quiet.

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

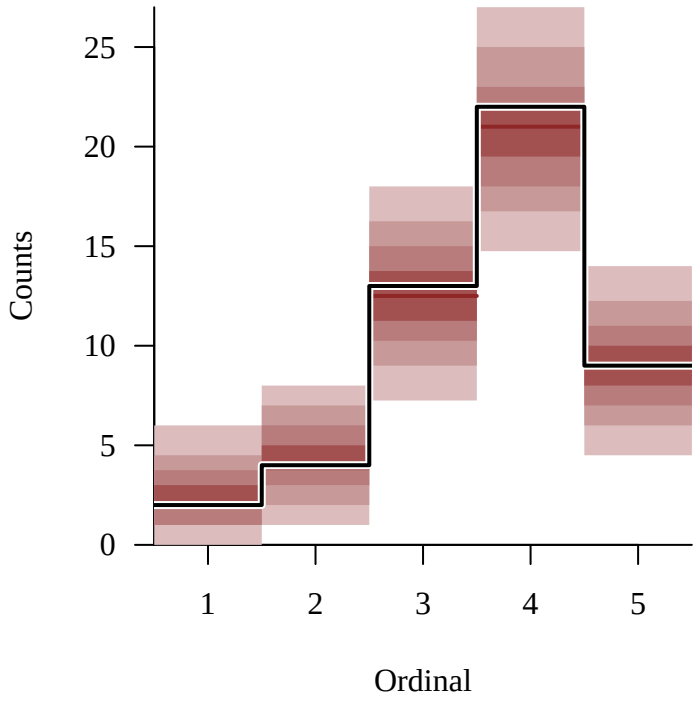
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples4 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples4,
                                       c('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.
```

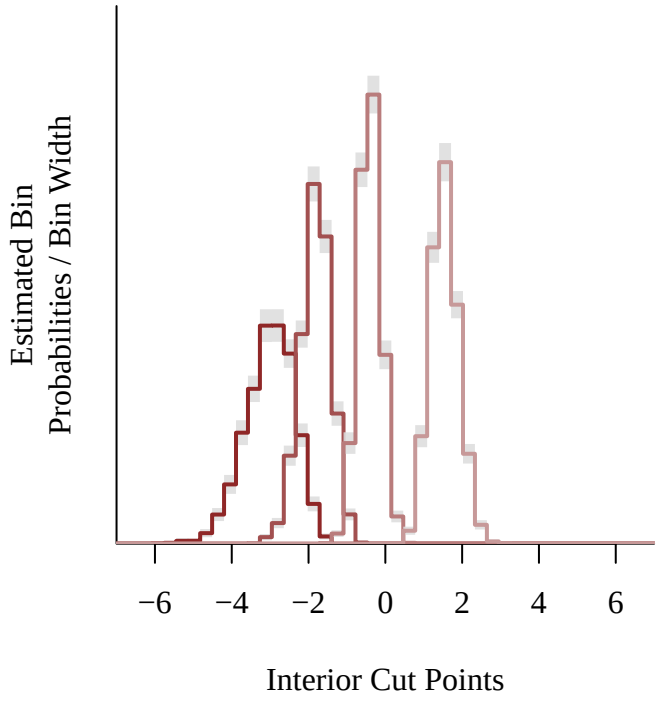
Retrodictive performance remains excellent.

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

util$plot_hist_quantiles(samples4, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



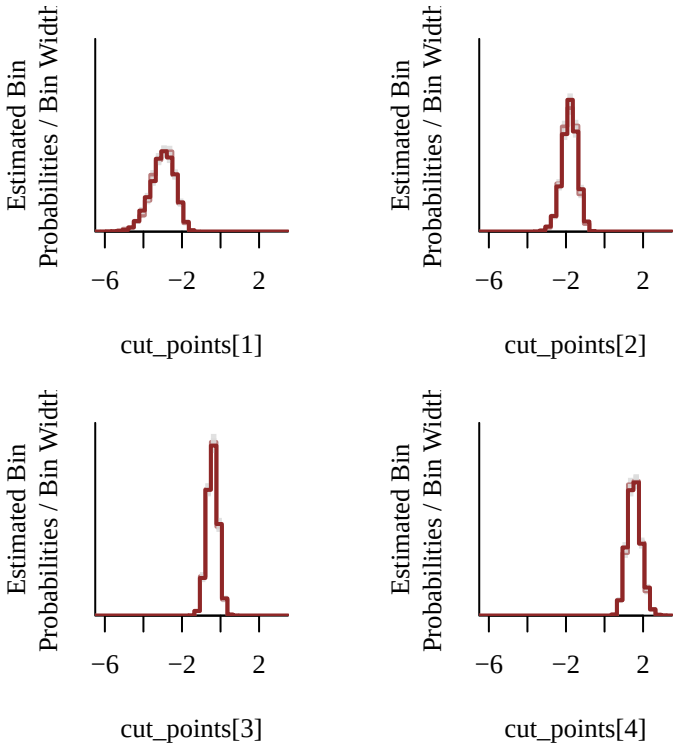
Inferences for the interior cut points are well-behaved.

```
plot_cut_point_overlay(samples4, 'cut_points[',
                       flim=c(-7, 7), fname='Interior Cut Points',
                       ylim=c(0, 1.5))
```



In fact the inferences for the interior cut points are identical to those from the `ordinal_logistic_derived` model.

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

for (k in 1:(data$K - 1)) {
  name <- paste0('cut_points[', k, ']')
  util$plot_expectand_pushforward(samples2[[name]],
                                  35, flim=c(-6.5, 3.5),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD",
                                  ylim=c(0, 1.5))
  util$plot_expectand_pushforward(samples4[[name]],
                                  35, flim=c(-6.5, 3.5),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD", add=TRUE)
}
```



7.3 Homogeneous Ordinal Probabilities II

Flush with confidence let's push our luck with a second data set.

7.3.1 Data Exploration

The structure of the data appears to be identical to the previous exercise.

```
data <- read_rdump('data/2.data.R')
```

```
cat(sprintf('%i ordinal values', data$K))
```

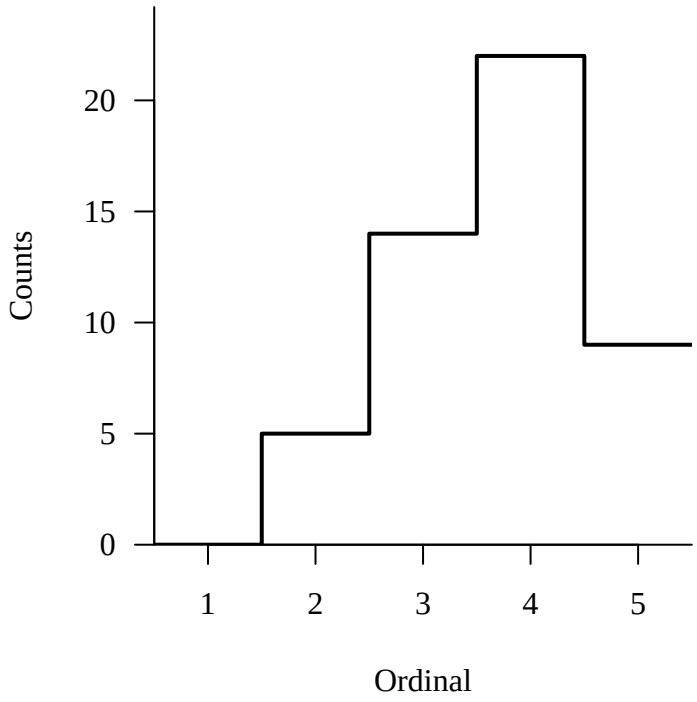
```
5 ordinal values
```

```
cat(sprintf('%i total observations', data$N))
```

```
50 total observations
```

Looking at a histogram of the data, however, we see an important difference. There are no observations for the first ordinal value.

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

util$plot_line_hist(data$y, 0.5, 5.5, 1,
                    xlab="Ordinal")
```



7.3.2 Categorical Model

Following the previous progression we start with a categorical model.

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

There are no indications of computational issues.

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

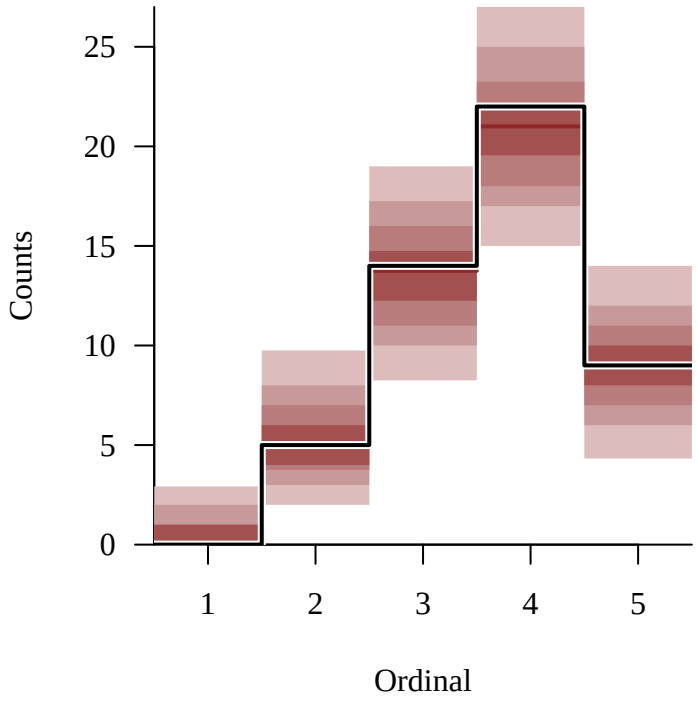
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('p'),
                                       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.
```

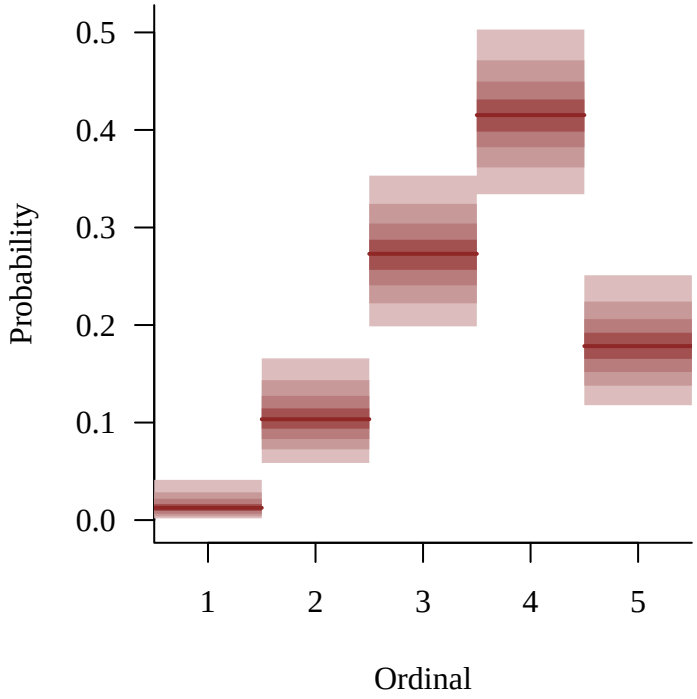
Similarly there are no indications of model inadequacies.

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

util$plot_hist_quantiles(samples1, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



In the end the inferred ordinal probabilities seem reasonable, with the consistent values of p_1 concentrating around zero.

```
names <- sapply(1:data$K, function(k) paste0('p[', k, ']'))
util$plot_disc_pushforward_quantiles(samples1, names,
                                     xlab="Ordinal",
                                     ylab="Probability")
```



7.3.3 Derived Cut Points

An ordered logit/logistic model with derived cut points works just as well.

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

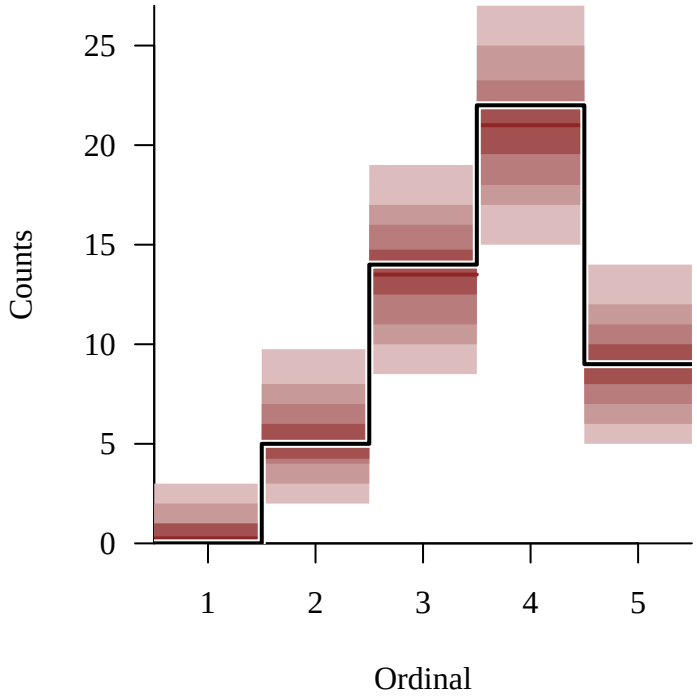
```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

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

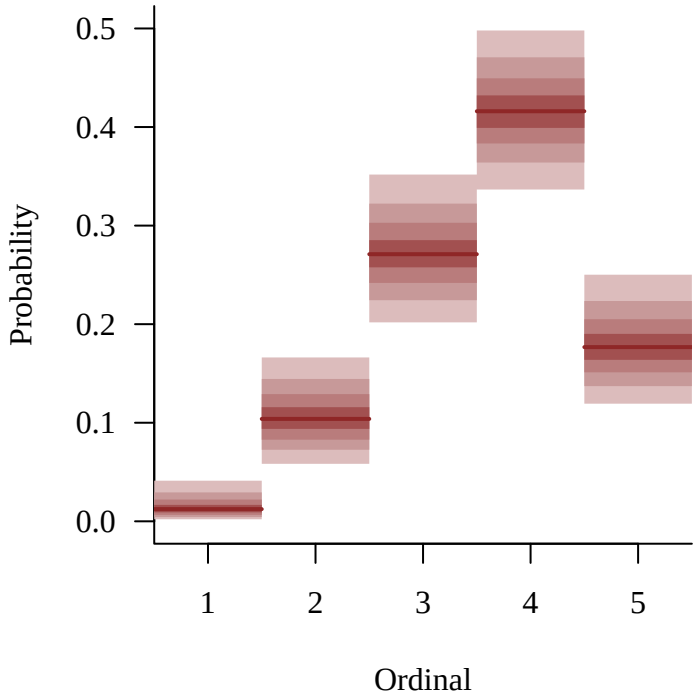
```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('p'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

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

util$plot_hist_quantiles(samples2, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



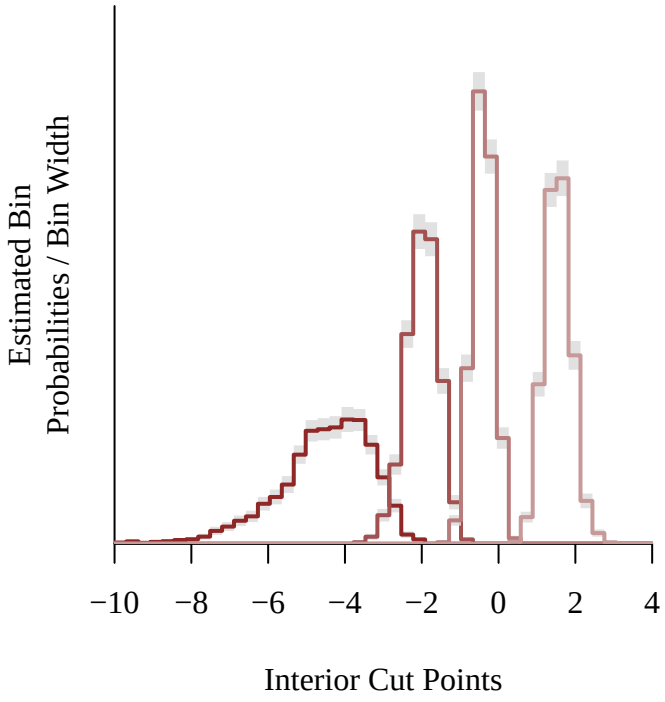

```
names <- sapply(1:data$K, function(k) paste0('p[', k, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     xlab="Ordinal",
                                     ylab="Probability")
```




```
plot_cut_point_overlay(samples2, 'cut_points[',
                       flim=c(-10, 4), fname='Interior Cut Points',
                       ylim=c(0, 1.5))
```

```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 9
values (0.2%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 0
values (0.0%) fell above the histogram binning.
```



7.3.4 Modeled Cut Points

Finally let's model the interior cut points directly and use the ill-defined uniform prior over the interior cut points that worked okay before. Will our sloppiness continue to go unpunished?

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

No, it will not. The diagnostics suggest strong inferential degeneracies and slow exploration, especially for the first interior cut point.

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

```
  Chain 1: 799 of 1024 transitions (78.03%)
           saturated the maximum treedepth of 10.

  Chain 2: 576 of 1024 transitions (56.25%)
           saturated the maximum treedepth of 10.
```

```
  Chain 2: Average proxy acceptance statistic (0.515)
           is smaller than 90% of the target (0.801).

  Chain 3: 11 of 1024 transitions (1.1%) diverged.
  Chain 3: 885 of 1024 transitions (86.43%)
           saturated the maximum treedepth of 10.

  Chain 4: 823 of 1024 transitions (80.37%)
           saturated the maximum treedepth of 10.

  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.

  Numerical trajectories that saturate the maximum treedepth have
terminated prematurely.  Increasing max_depth above 10 should result in
more expensive, but more efficient, Hamiltonian transitions.

  A small average proxy acceptance statistic indicates that the
adaptation of the numerical integrator step size failed to converge.
This is often due to discontinuous or imprecise gradients.
```

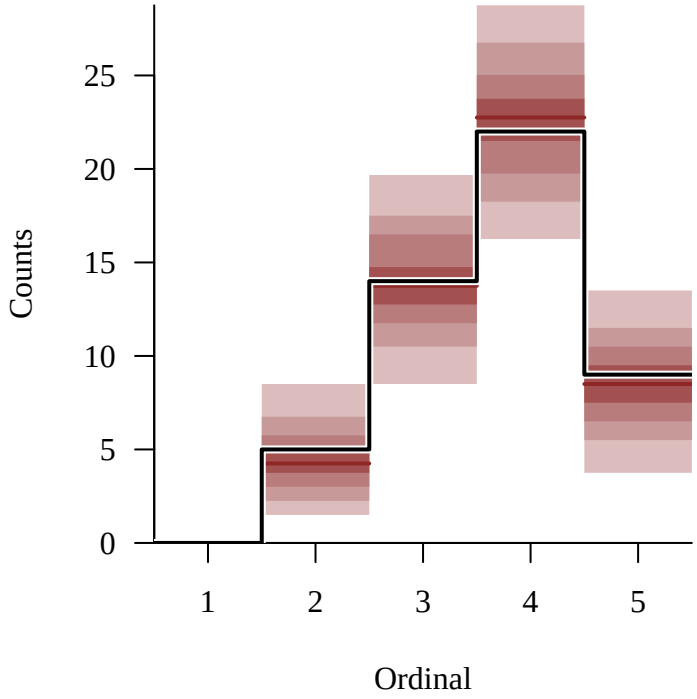
```
samples3 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples3,
                                       c('cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
cut_points[1]:
  Split hat{R} (1.539) exceeds 1.1.
  Chain 1: hat{ESS} (93.587) is smaller than desired (100).
  Chain 2: hat{ESS} (5.799) is smaller than desired (100).
  Chain 3: hat{ESS} (5.555) is smaller than desired (100).
  Chain 4: hat{ESS} (35.367) is smaller than desired (100).

cut_points[2]:
  Chain 3: hat{ESS} (48.028) is smaller than desired (100).

cut_points[4]:
```

```
  Chain 2: hat{ESS} (17.348) is smaller than desired (100).
  Chain 3: hat{ESS} (79.670) is smaller than desired (100).

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

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

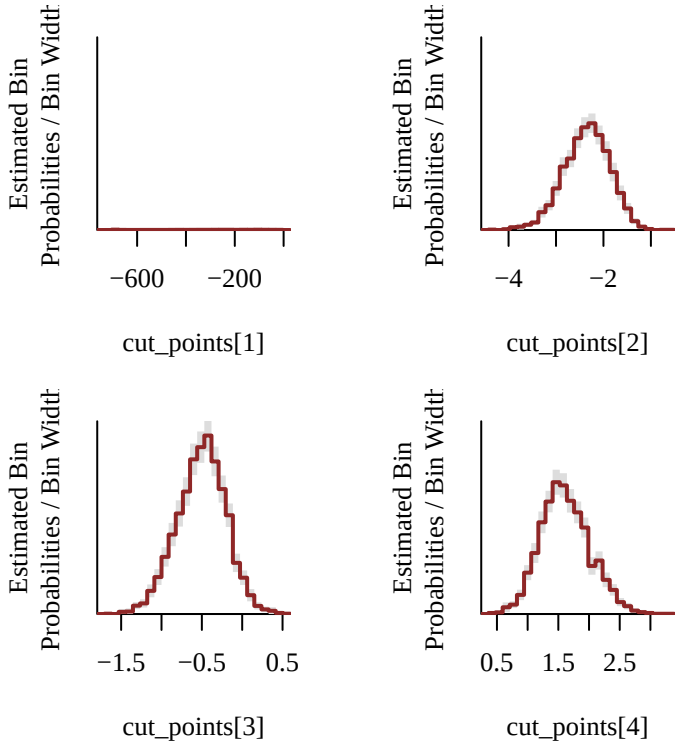
That said the retrodictive performance doesn't look bad at all.

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

util$plot_hist_quantiles(samples3, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



Once we examine our posterior inferences the problem becomes clear. Because there are no observations of the first ordinal the first interior cut point is only poorly informed by the observed data; the exact marginal posterior distribution for `cut_points[1]` stretches all the way to $-\infty$. The sampler does its best but can get only so far in finite time.

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

for (k in 1:(data$K - 1)) {
  name <- paste0('cut_points[', k, ']')
  util$plot_expectand_pushforward(samples3[[name]], 25,
                                  display_name=name,
                                  ylim=c(0, 1.5))
}
```



The reason we didn't see problems when using the `categorical` and `ordinal_logistic_derived` models is that they employed a more informative prior model. Again uniformity over the interior cut points is not equivalent to uniformity over the ordinal probabilities!

Fortunately we can incorporate the same informative prior model with the induced Dirichlet probability density function.

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

All of the computational problems have suddenly disappeared.

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

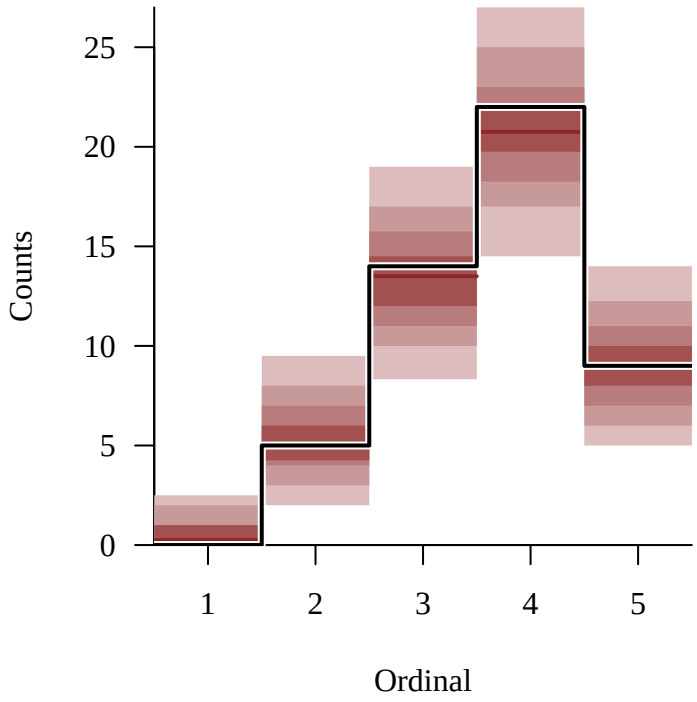
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples4 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples4,
                                       c('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.
```

The retrodictive performance is solid.

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

util$plot_hist_quantiles(samples4, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```

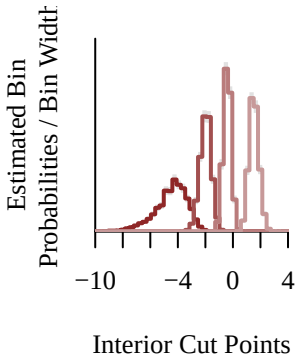


Posterior inferences for all of the interior cut points are clean.

```
plot_cut_point_overlay(samples4, 'cut_points[',
                       flim=c(-10, 4), fname='Interior Cut Points',
                       ylim=c(0, 1.5))
```

```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 4
values (0.1%) fell below the histogram binning.
```

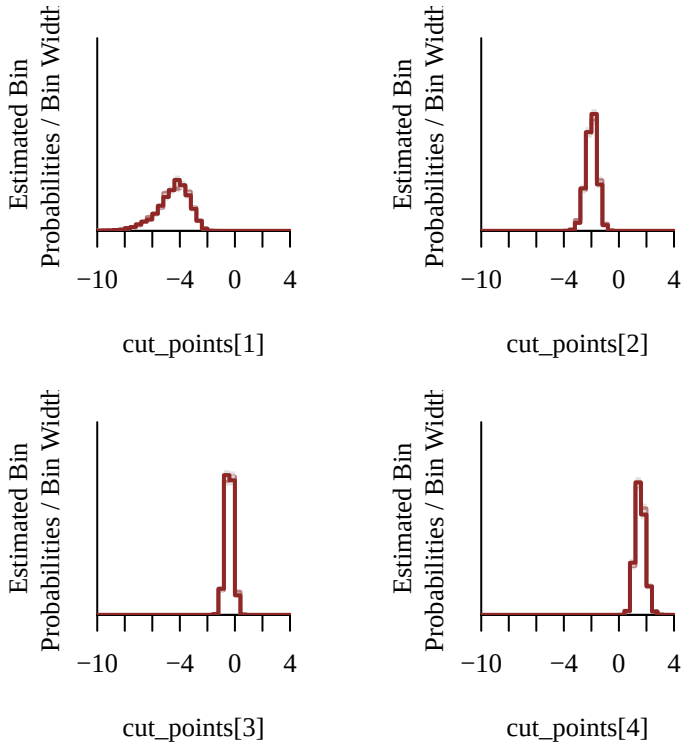
```
Warning in util$plot_expectand_pushforward(expectand_vals_list[[name]], : 0
values (0.0%) fell above the histogram binning.
```



Indeed the inferences for the interior cut points are equivalent regardless of which approach we take.

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

for (k in 1:(data$K - 1)) {
  name <- paste0('cut_points[', k, ']')
  util$plot_expectand_pushforward(samples2[[name]],
                                  35, flim=c(-10, 4),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD",
                                  ylim=c(0, 1.5))
  util$plot_expectand_pushforward(samples4[[name]],
                                  35, flim=c(-10, 4),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD", add=TRUE)
}
```

```
Warning in util$plot_expectand_pushforward(samples2[[name]], 35, flim = c(-10,
: 9 values (0.2%) fell below the histogram binning.

Warning in util$plot_expectand_pushforward(samples2[[name]], 35, flim = c(-10,
: 0 values (0.0%) fell above the histogram binning.

Warning in util$plot_expectand_pushforward(samples4[[name]], 35, flim = c(-10,
: 4 values (0.1%) fell below the histogram binning.

Warning in util$plot_expectand_pushforward(samples4[[name]], 35, flim = c(-10,
: 0 values (0.0%) fell above the histogram binning.
```



7.4 Heterogeneous Ordinal Probabilities

Having built up some familiarity with the basics of the cut point construction let's see if we can model how ordinal probabilities vary across different circumstances.

7.4.1 Data Exploration

Here our data is comprised of ordinal appraisals of nine different items.

```
data <- read_rdump('data/3.data.R')
```

```
cat(sprintf('%i ordinal values', data$K))
```

```
5 ordinal values
```

```
cat(sprintf('%i items', data$I))
```

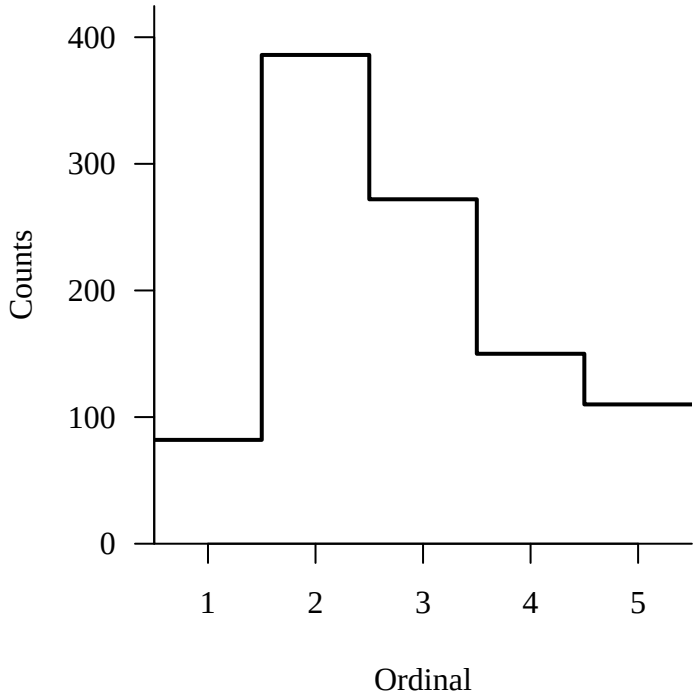
```
9 items
```

```
cat(sprintf('%i total observations', data$N))
```

```
1000 total observations
```

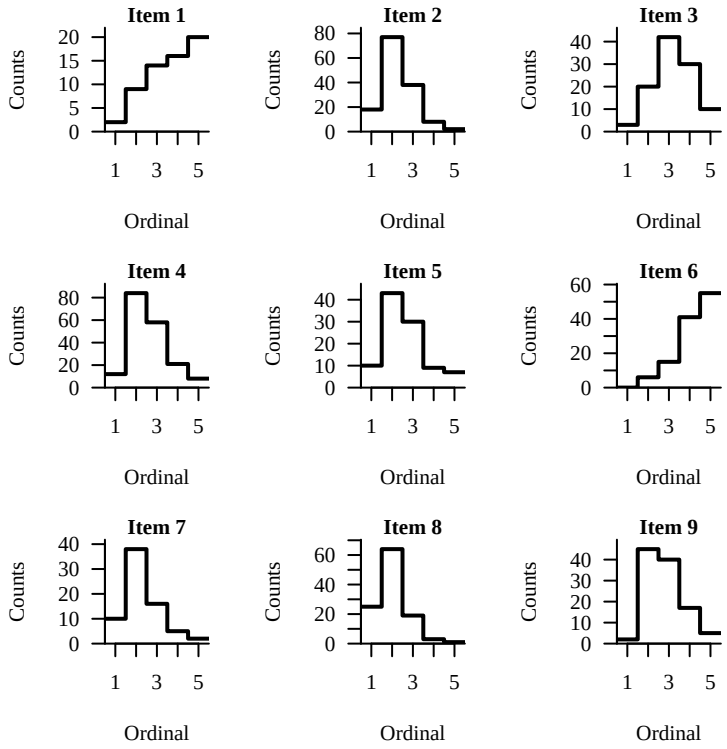
We can still aggregate all of the observations together into a single histogram.

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

util$plot_line_hist(data$y, 0.5, 5.5, 1, xlab="Ordinal")
```



Now, however, we can also stratify the histograms by item. Even the raw data exhibits clear differences in behavior across the items.

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

for (i in 1:data$I) {
  util$plot_line_hist(data$y[data$item_idx == i],
                      0.5, 5.5, 1, xlab="Ordinal",
                      main=paste("Item", i))
}
```



7.4.2 Ill-Defined Uniform Prior Model

In order to infer the individual item qualities we can use an ordinal pairwise modeling approach, with the item qualities shifting the latent logistic density function away from its baseline location. Because the shifts will generally vary across observations I have implemented the `ordinal_shifted_logistic` for one observation at a time, requiring a loop in the observational model.

Note that the line `vector[K - 1] Pi = inv_logit(c - gamma);` can be equally-well interpreted as translating a standard logistic cumulative distribution function and then evaluating it at the interior cut points, evaluating a varying-location logistic cumulative distribution function at the interior cut points, or evaluating a standard logistic cumulative distribution function at translated interior cut points.

Once again we'll start with an ill-defined uniform prior model, this time over both the interior cut points and the item qualities.

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

The diagnostics are unhappy to say the least. In particular the tree depth saturation and large empirical autocorrelations together suggest strong inferential degeneracies.

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

```
  Chain 1: 419 of 1024 transitions (40.92%)
           saturated the maximum treedepth of 10.

  Chain 2: 469 of 1024 transitions (45.80%)
           saturated the maximum treedepth of 10.

  Chain 3: 594 of 1024 transitions (58.01%)
           saturated the maximum treedepth of 10.

  Chain 4: 494 of 1024 transitions (48.24%)
           saturated the maximum treedepth of 10.

  Numerical trajectories that saturate the maximum treedepth have
terminated prematurely.  Increasing max_depth above 10 should result in
more expensive, but more efficient, Hamiltonian transitions.
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('gamma',
                                         'cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

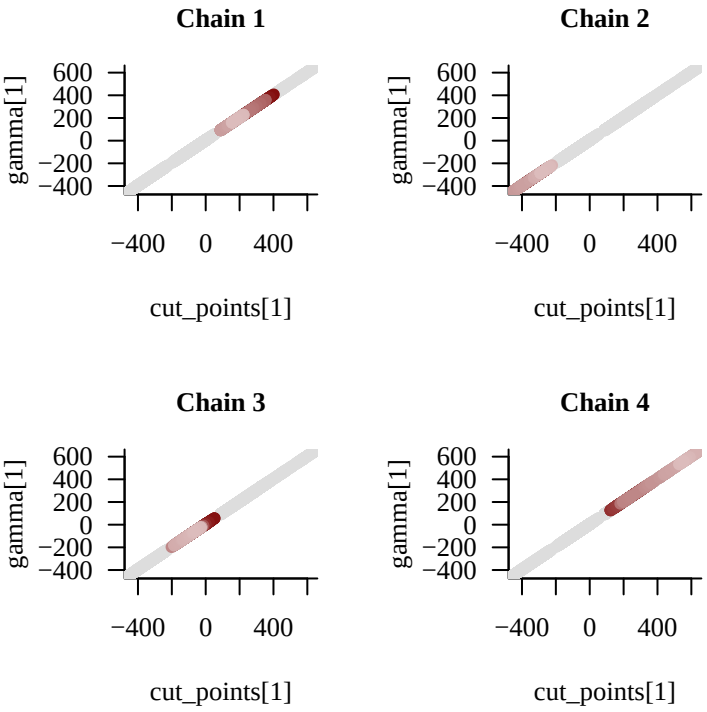
```
gamma[1]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.804) is smaller than desired (100).
  Chain 3: hat{ESS} (9.657) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).

gamma[2]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.565) is smaller than desired (100).
  Chain 2: hat{ESS} (7.804) is smaller than desired (100).
  Chain 3: hat{ESS} (9.650) is smaller than desired (100).
```

```
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).

gamma[3]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.806) is smaller than desired (100).
  Chain 3: hat{ESS} (9.656) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).

gamma[4]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.804) is smaller than desired (100).
  Chain 3: hat{ESS} (9.652) is smaller than desired (100).
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).

gamma[5]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.802) is smaller than desired (100).
  Chain 3: hat{ESS} (9.655) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).

gamma[6]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.568) is smaller than desired (100).
  Chain 2: hat{ESS} (7.806) is smaller than desired (100).
  Chain 3: hat{ESS} (9.659) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).

gamma[7]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.565) is smaller than desired (100).
  Chain 2: hat{ESS} (7.802) is smaller than desired (100).
  Chain 3: hat{ESS} (9.650) is smaller than desired (100).
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).

gamma[8]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.566) is smaller than desired (100).
  Chain 2: hat{ESS} (7.805) is smaller than desired (100).
  Chain 3: hat{ESS} (9.648) is smaller than desired (100).
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).
```

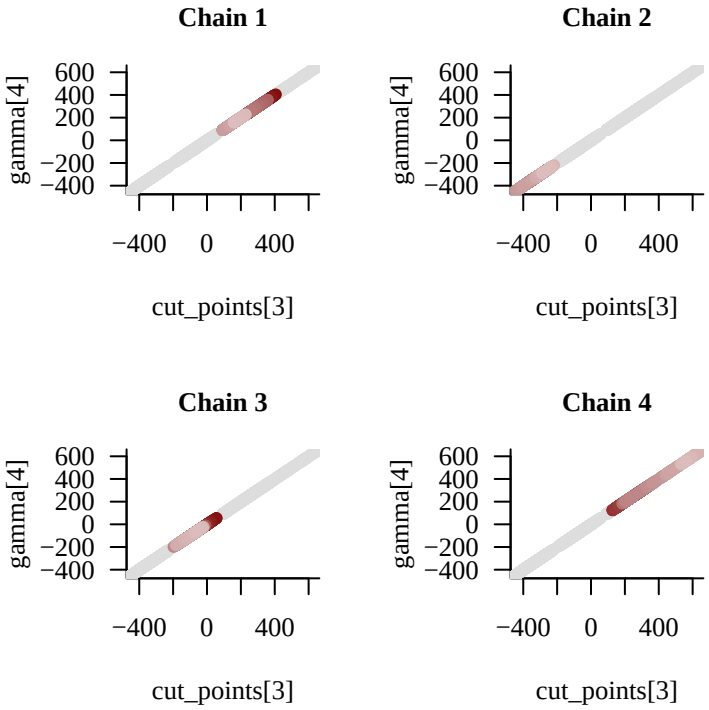
```
gamma[9]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.804) is smaller than desired (100).
  Chain 3: hat{ESS} (9.655) is smaller than desired (100).
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).

cut_points[1]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.566) is smaller than desired (100).
  Chain 2: hat{ESS} (7.802) is smaller than desired (100).
  Chain 3: hat{ESS} (9.653) is smaller than desired (100).
  Chain 4: hat{ESS} (5.223) is smaller than desired (100).

cut_points[2]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.803) is smaller than desired (100).
  Chain 3: hat{ESS} (9.651) is smaller than desired (100).
  Chain 4: hat{ESS} (5.224) is smaller than desired (100).

cut_points[3]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.568) is smaller than desired (100).
  Chain 2: hat{ESS} (7.804) is smaller than desired (100).
  Chain 3: hat{ESS} (9.652) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).

cut_points[4]:
  Split hat{R} (4.186) exceeds 1.1.
  Chain 1: hat{ESS} (5.567) is smaller than desired (100).
  Chain 2: hat{ESS} (7.806) is smaller than desired (100).
  Chain 3: hat{ESS} (9.661) is smaller than desired (100).
  Chain 4: hat{ESS} (5.225) is smaller than desired (100).


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

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

Examining a few pairs plots immediately revels the problem; the item qualities are completely degenerate with the interior cut points! The item qualities can be arbitrarily small and still be consistent with the observed data so long as the appraisers are arbitrarily generous.

```
util$plot_pairs_by_chain(samples1[['cut_points[1]']], 'cut_points[1]',
                         samples1[['gamma[1]']], 'gamma[1]')
```




```
util$plot_pairs_by_chain(samples1[['cut_points[3]']], 'cut_points[3]',
                         samples1[['gamma[4]']], 'gamma[4]')
```



7.4.3 Anchoring An Item Quality

The problem is that we have not distinguished a proper baseline, resulting in redundancy between the configuration of the latent probability density function and the interior cut points. One way to remove this redundancy is to anchor the quality of one of the observed items to zero so that $\gamma_{i_b} = 0$ defines the baseline.

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

This seems to have eliminated the computational issues.

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

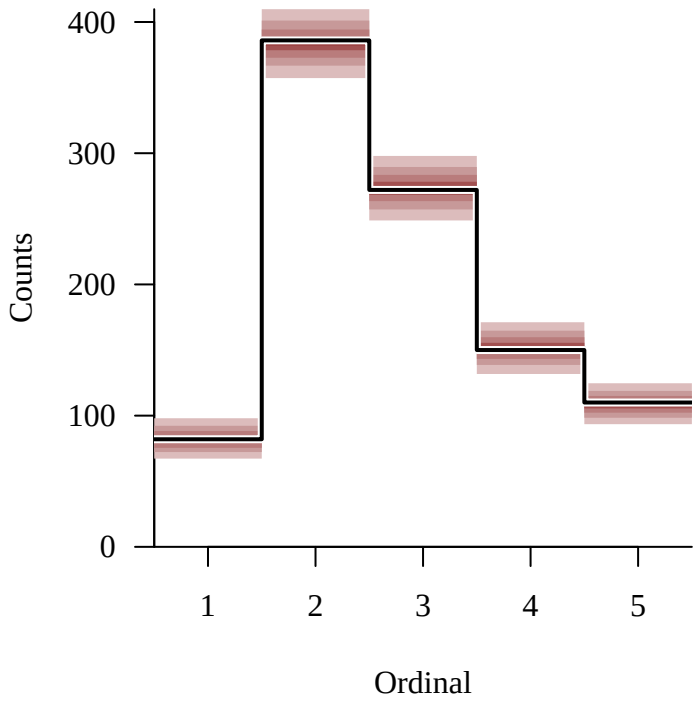
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('gamma_free',
                                         '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.
```

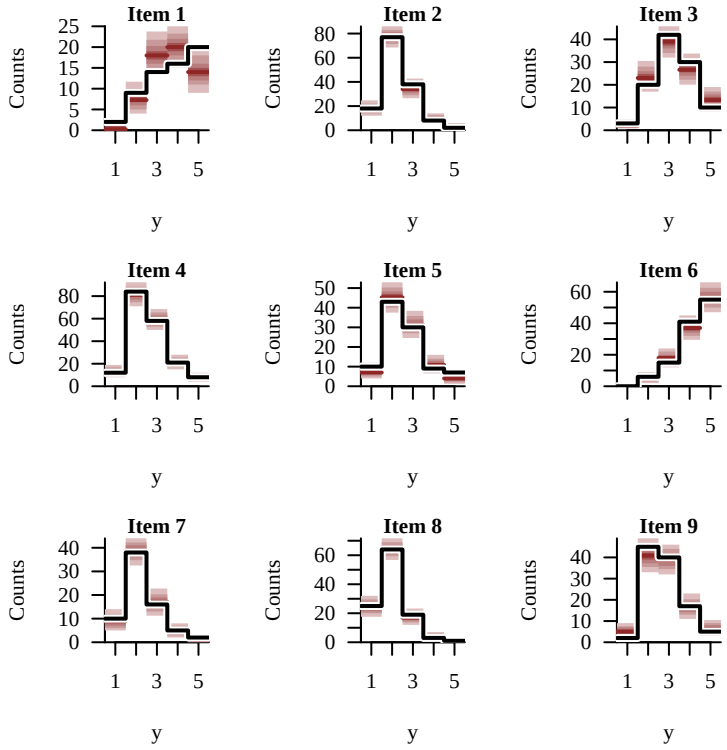
Moreover there are no indications of model inadequacy in either the aggregate or item-stratified histograms.

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

util$plot_hist_quantiles(samples2, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



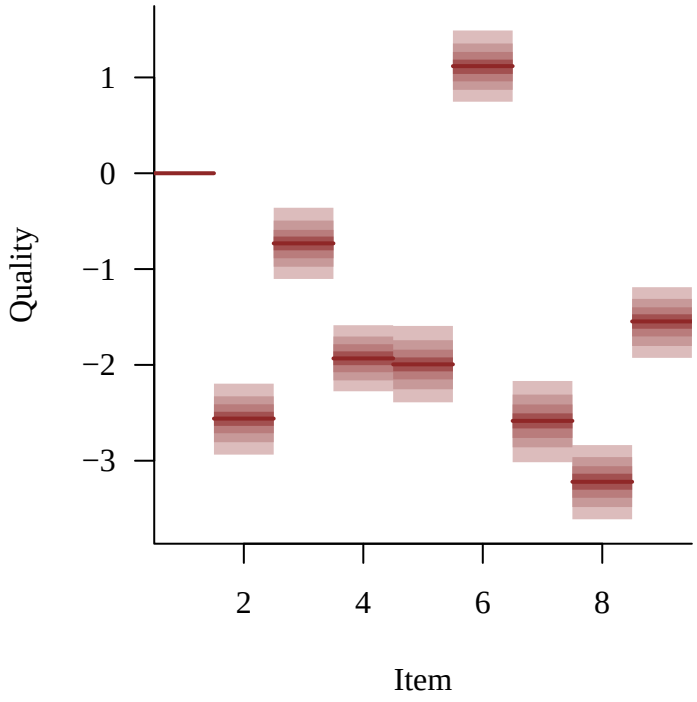

```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
```

```
  names <- sapply(which(data$item_idx == i),
                  function(n) paste0('y_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples2, names)

  item_data <- data$y[data$item_idx == i]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=item_data,
                           xlab="y", main=paste('Item', i))
}
```

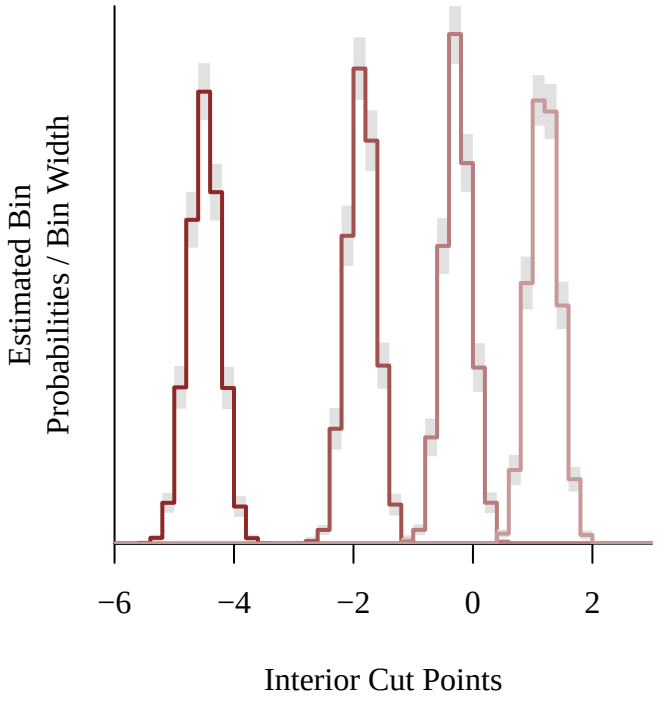


The posterior inferences are then able to cleanly inform the remaining item qualities at the same time as the interior cut points.

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

names <- sapply(1:data$I, function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     xlab="Item",
                                     ylab="Quality")
```




```
plot_cut_point_overlay(samples2, 'cut_points[',
                       flim=c(-6, 3), fname='Interior Cut Points',
                       ylim=c(0, 1.75))
```



7.4.4 Informative Prior Model

Another way to define a proper baseline and avoid this degeneracy is to exploit any domain expertise about the baseline ordinal probabilities for a hypothetical item with $\gamma = 0$. Here we will use an induced Dirichlet prior model to contain the interior cut points to reasonable behaviors.

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

If any degeneracies remain then they are not strong enough to cause any serious computational issues.

```
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.
```

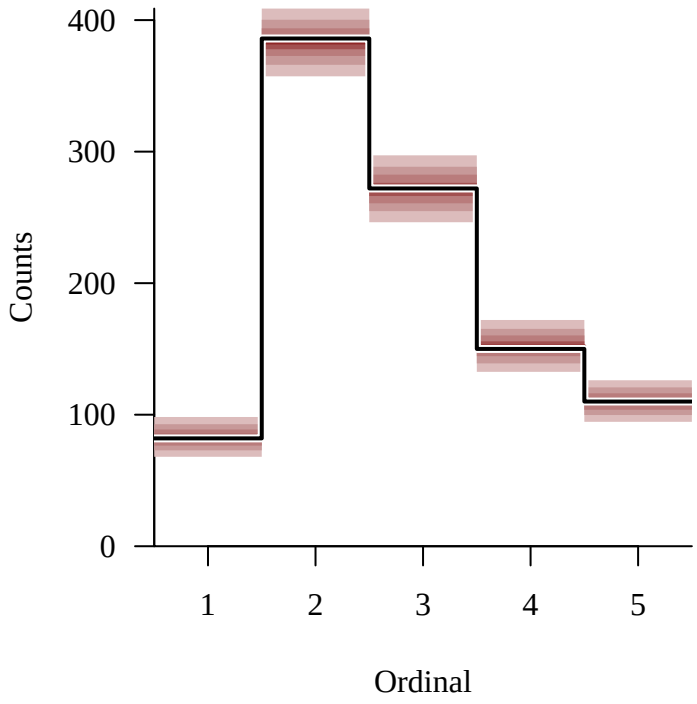
```
samples3 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples3,
                                       c('gamma',
                                         'cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)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

```
gamma[1]:
  Chain 4: hat{ESS} (96.487) is smaller than desired (100).

gamma[2]:
  Chain 4: hat{ESS} (92.607) is smaller than desired (100).

gamma[3]:
  Chain 4: hat{ESS} (97.294) is smaller than desired (100).

gamma[4]:
  Chain 4: hat{ESS} (89.670) is smaller than desired (100).

gamma[5]:
  Chain 4: hat{ESS} (92.066) is smaller than desired (100).

gamma[7]:
  Chain 4: hat{ESS} (97.492) is smaller than desired (100).

gamma[9]:
  Chain 4: hat{ESS} (92.314) is smaller than desired (100).

cut_points[1]:
  Chain 4: hat{ESS} (90.759) is smaller than desired (100).

cut_points[2]:
  Chain 4: hat{ESS} (87.662) is smaller than desired (100).

cut_points[3]:
  Chain 4: hat{ESS} (84.884) is smaller than desired (100).

cut_points[4]:
  Chain 4: hat{ESS} (87.384) is smaller than desired (100).


Small empirical effective sample sizes result in imprecise Markov chain
```

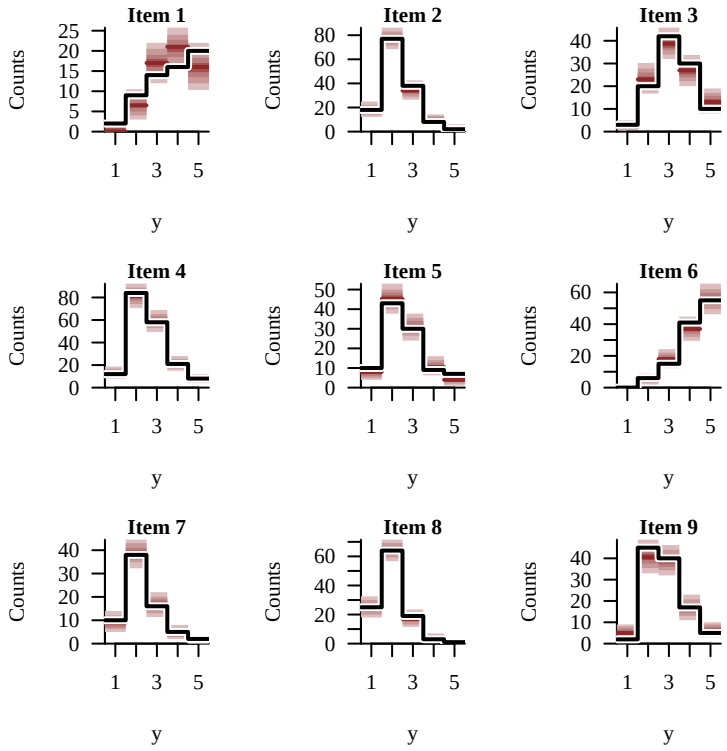
```
Monte Carlo estimators.
```

Retrodictive performance of both the aggregate and stratified histograms looks great.

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

util$plot_hist_quantiles(samples3, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```

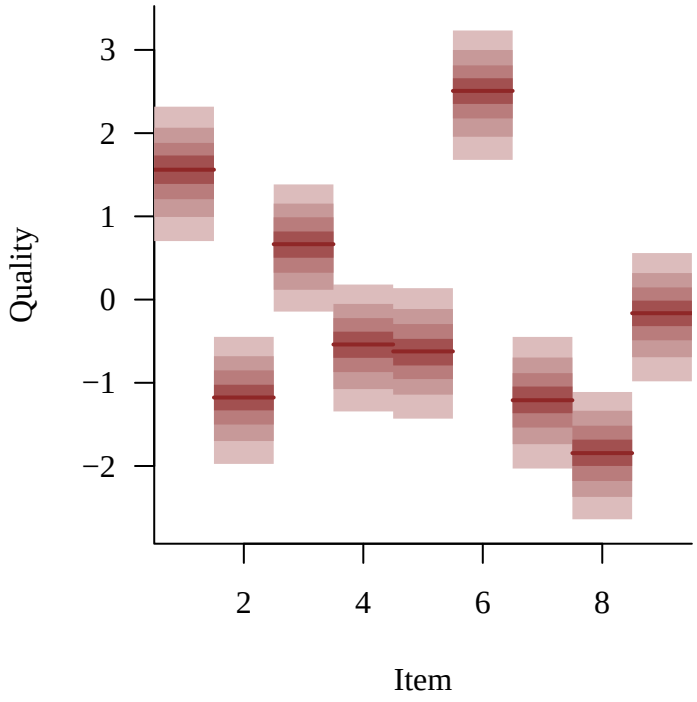



```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
  names <- sapply(which(data$item_idx == i),
                  function(n) paste0('y_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples3, names)

  item_data <- data$y[data$item_idx == i]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=item_data,
                           xlab="y", main=paste('Item', i))
}
```

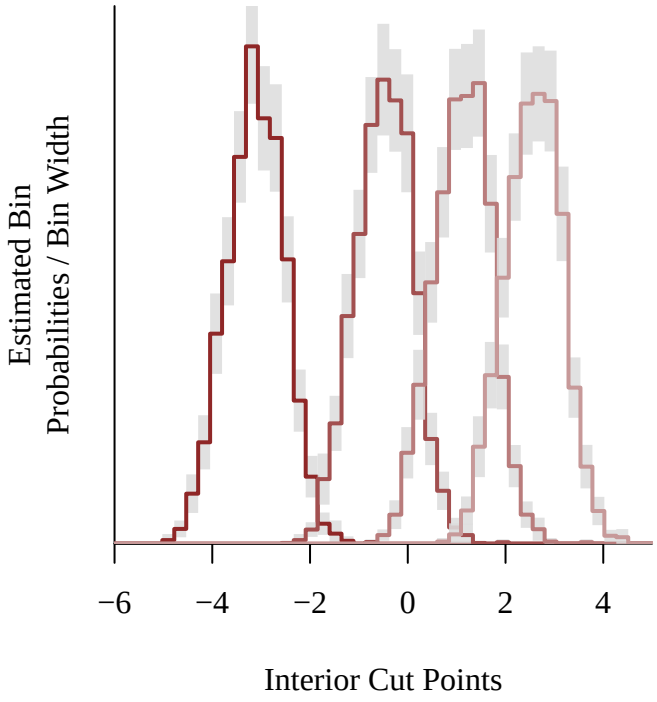


The marginal posterior inferences for both the item qualities and the interior cut points are all strongly informed.

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

names <- sapply(1:data$I, function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles(samples3, names,
                                     xlab="Item",
                                     ylab="Quality")
```




```
plot_cut_point_overlay(samples3, 'cut_points[',
                       flim=c(-6, 5), fname='Interior Cut Points',
                       ylim=c(0, 0.75))
```



Note that `Stan` provides its own implementation of a shifted logistic ordinal model under the name `ordered_logistic`. The `Stan` implementation also swaps the cut point and shift arguments relative to the convention that I've used here.

Because it is coded in `C++` the `Stan` implementation is a bit faster that what I have implemented here; it is also accepts vector arguments for the shifts so that the observational model can be defined without a loop.

Finally another mathematically equivalent implementation is to model the baseline ordinal probabilities and then derive the interior cut points assuming $\gamma = 0$. Once we've derived the cut points we can then shift the latent logistic density function to perturb the ordinal probabilities for each item.

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

Some mild autocorrelation warnings have popped up for this implementation, but this results in only some mild inefficiencies.

```
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.
```

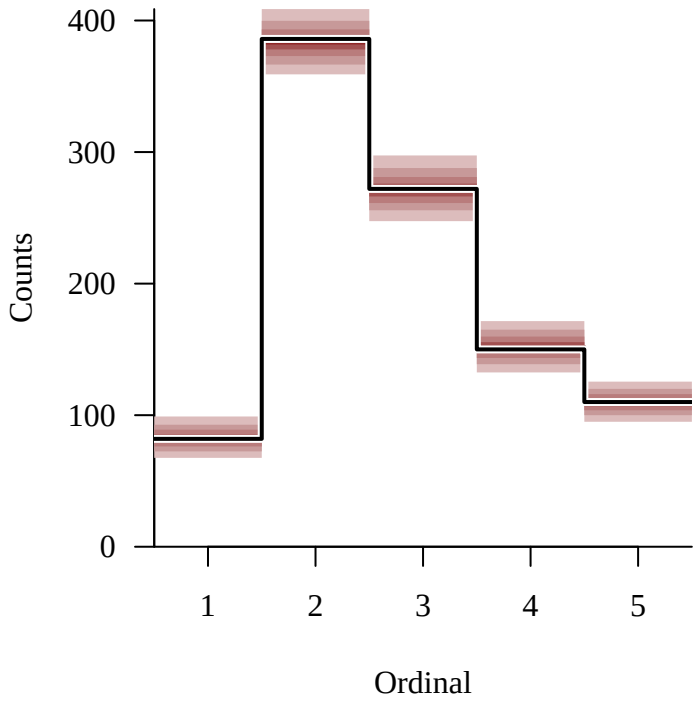
```
samples4 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples4,
                                       c('gamma', 'p'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
gamma[1]:
  Chain 1: hat{ESS} (43.754) is smaller than desired (100).

gamma[2]:
  Chain 1: hat{ESS} (39.831) is smaller than desired (100).

gamma[3]:
  Chain 1: hat{ESS} (42.573) is smaller than desired (100).

gamma[4]:
  Chain 1: hat{ESS} (36.565) is smaller than desired (100).

gamma[5]:
  Chain 1: hat{ESS} (38.798) is smaller than desired (100).

gamma[6]:
  Chain 1: hat{ESS} (40.949) is smaller than desired (100).

gamma[7]:
  Chain 1: hat{ESS} (40.586) is smaller than desired (100).

gamma[8]:
  Chain 1: hat{ESS} (41.810) is smaller than desired (100).

gamma[9]:
  Chain 1: hat{ESS} (41.289) is smaller than desired (100).

p[1]:
  Chain 1: hat{ESS} (30.114) is smaller than desired (100).
  Chain 4: hat{ESS} (81.128) is smaller than desired (100).

p[2]:
  Chain 1: hat{ESS} (39.982) is smaller than desired (100).
```

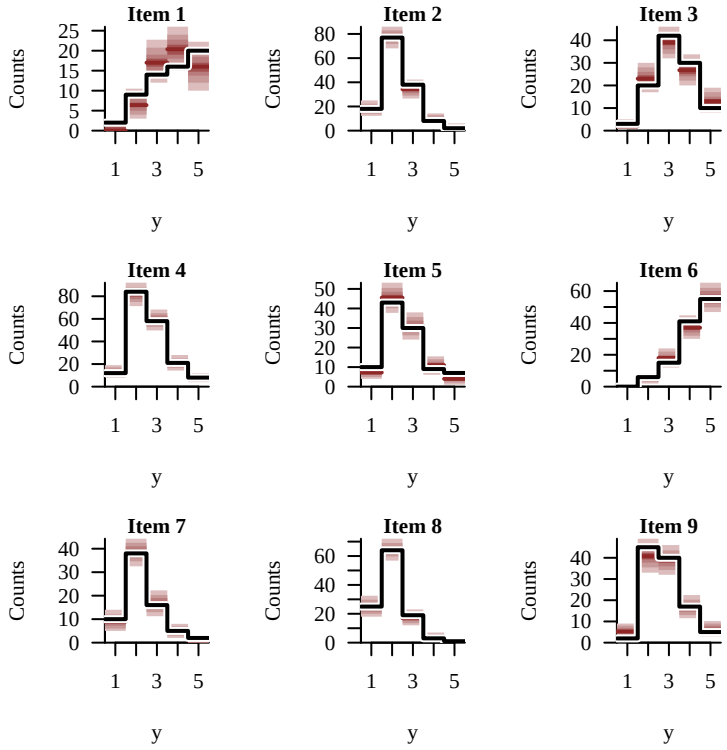
```
p[3]:
  Chain 1: hat{ESS} (47.177) is smaller than desired (100).
  Chain 4: hat{ESS} (87.203) is smaller than desired (100).

p[4]:
  Chain 1: hat{ESS} (41.954) is smaller than desired (100).

p[5]:
  Chain 1: hat{ESS} (69.353) is smaller than desired (100).


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

Otherwise the retrodictive performance is similar to what we saw with the `ordered_logistic_comp_induced` model implementation.

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

util$plot_hist_quantiles(samples4, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```



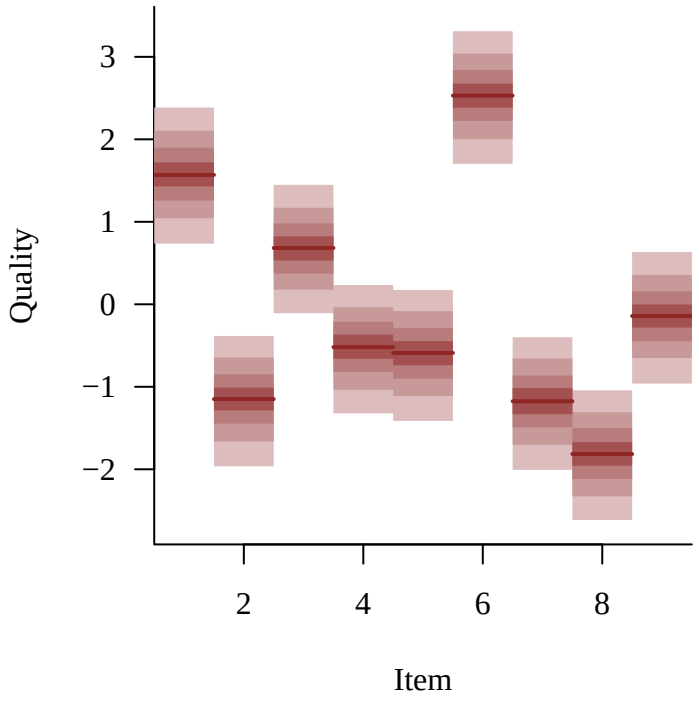
```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
  names <- sapply(which(data$item_idx == i),
                  function(n) paste0('y_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples4, names)

  item_data <- data$y[data$item_idx == i]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=item_data,
                           xlab="y", main=paste('Item', i))
}
```



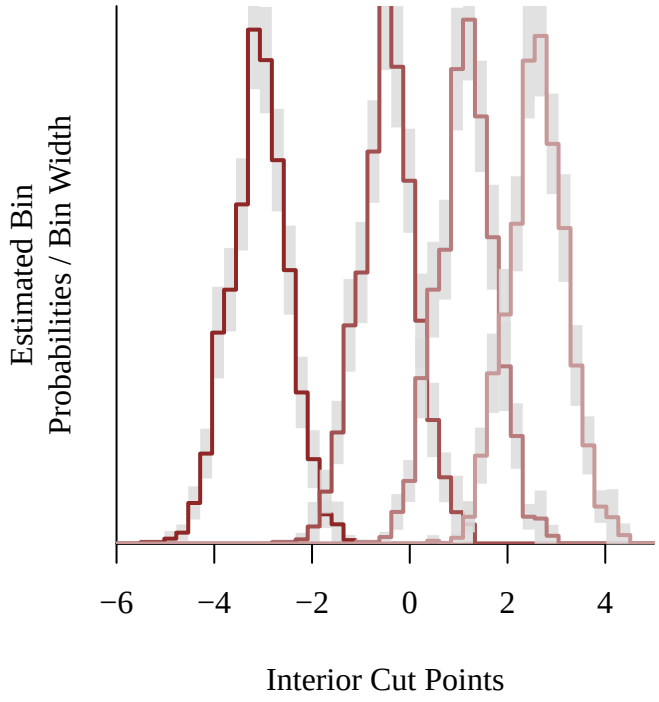
So too are the posterior inferences.

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

names <- sapply(1:data$I, function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles(samples4, names,
```

```
                                    xlab="Item",
                                    ylab="Quality")
```




```
plot_cut_point_overlay(samples4, 'cut_points[',
                       flim=c(-6, 5), fname='Interior Cut Points',
                       ylim=c(0, 0.75))
```

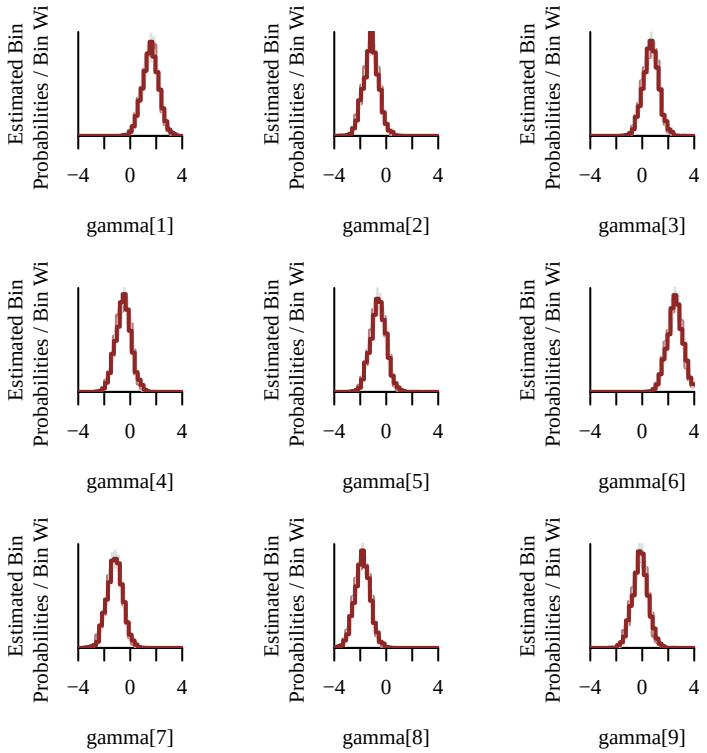


Indeed overlaying the posterior inferences from the two model implementations doesn't show any appreciable difference.

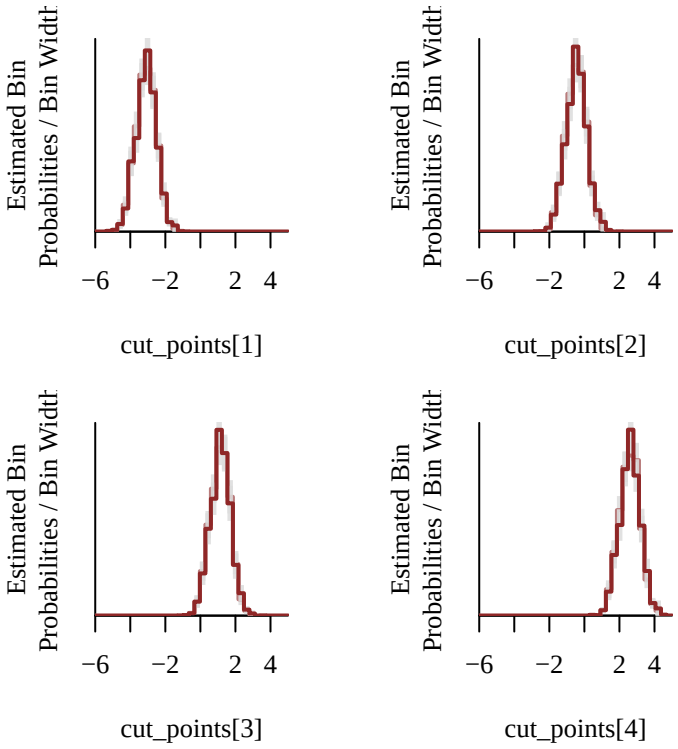
```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
  name <- paste0('gamma[', i, ']')
  util$plot_expectand_pushforward(samples3[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD",
                                  ylim=c(0, 0.75))
  util$plot_expectand_pushforward(samples4[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD", add=TRUE)
}
```

```
Warning in util$plot_expectand_pushforward(samples4[[name]], 35, flim = c(-4, :
2 values (0.0%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples4[[name]], 35, flim = c(-4, :
0 values (0.0%) fell above the histogram binning.
```




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

for (k in 1:(data$K - 1)) {
  name <- paste0('cut_points[', k, ']')
  util$plot_expectand_pushforward(samples3[[name]],
                                  35, flim=c(-6, 5),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD",
                                  ylim=c(0, 0.75))
  util$plot_expectand_pushforward(samples4[[name]],
                                  35, flim=c(-6, 5),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD", add=TRUE)
}
```



7.5 Heterogeneous Baseline Ordinal Probabilities

As a little treat let's end our demonstration by extending the cut point construction a bit.

For this example the appraisers rate items have been categorized into individual contexts, and we have reason to believe that the baseline ordinal probabilities within each context are not the same. In fact we'll not only model how the baseline behaviors vary across the contexts but do so *hierarchically*.

7.5.1 Data Exploration

The data for this analysis contains an additional piece of information: with which of the nine contexts is each observation associated.

```
data <- read_rdump('data/4.data.R')
```

```
cat(sprintf('%i ordinal values', data$K))
```

```
5 ordinal values
```

```
cat(sprintf('%i items', data$I))
```

```
9 items
```

```
cat(sprintf('%i contexts', data$K))
```

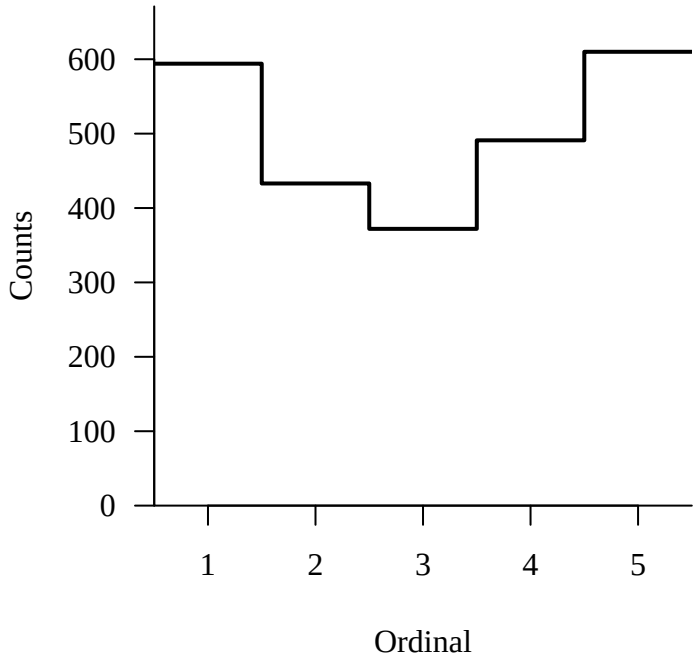
```
5 contexts
```

```
cat(sprintf('%i total observations', data$N))
```

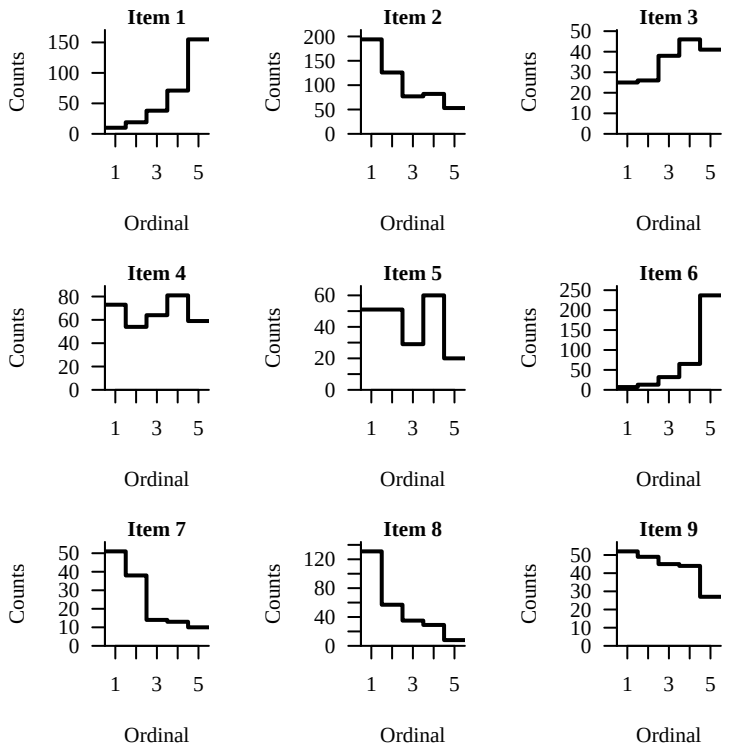
```
2500 total observations
```

This allows us to histogram not only the aggregate data but also the data stratified by item and context.

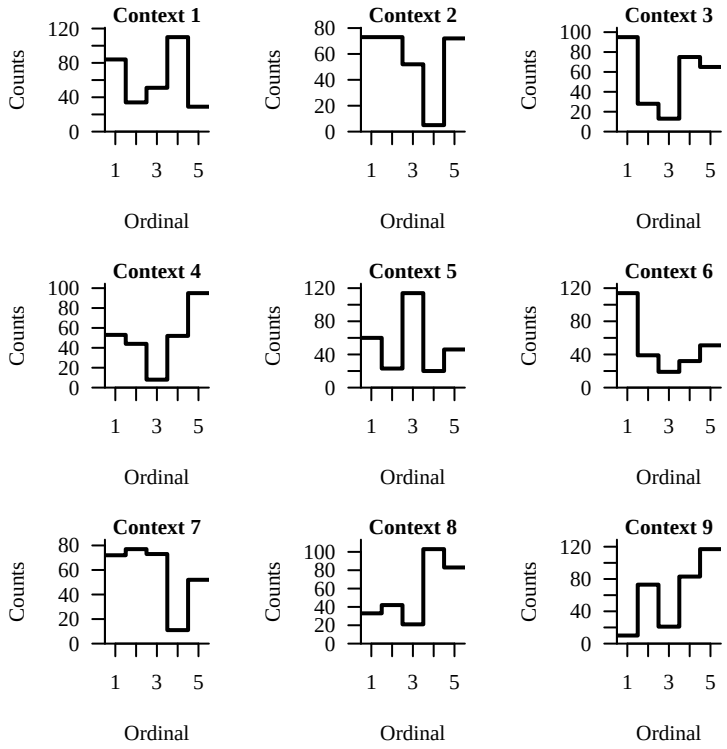
```
par(mfrow=c(1, 1), mar=c(5, 5, 2, 1))

util$plot_line_hist(data$y, 0.5, 5.5, 1,
                    xlab="Ordinal")
```



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

for (i in 1:data$I) {
  util$plot_line_hist(data$y[data$item_idx == i],
                      0.5, 5.5, 1,
                      xlab="Ordinal", main=paste('Item', i))
}
```




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

for (j in 1:data$J) {
  util$plot_line_hist(data$y[data$context_idx == j],
                      0.5, 5.5, 1,
                      xlab="Ordinal", main=paste('Context', j))
}
```



In theory we could also stratify by context and item at the time, but those summaries tend to be too variable to be insightful unless we have a lot of observations spanning all of the possible item-context pairs.

7.5.2 Hierarchical Cut Points

Replicating the baseline ordinal probabilities is straightforward; we just need to replicate the interior cut points for each context. Moreover if the contexts are exchangeable then using the induced Dirichlet prior model we can couple the interior cut points together with a Dirichlet population model.

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

In spite of the growing complexity of the model no computational problems have arisen.

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

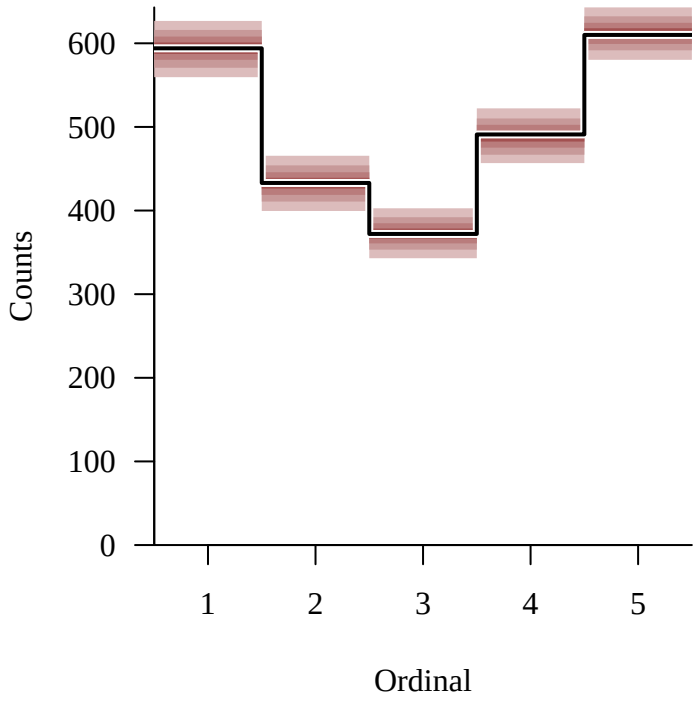
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('gamma',
                                         'mu_p',
                                         'tau_p',
                                         'cut_points'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

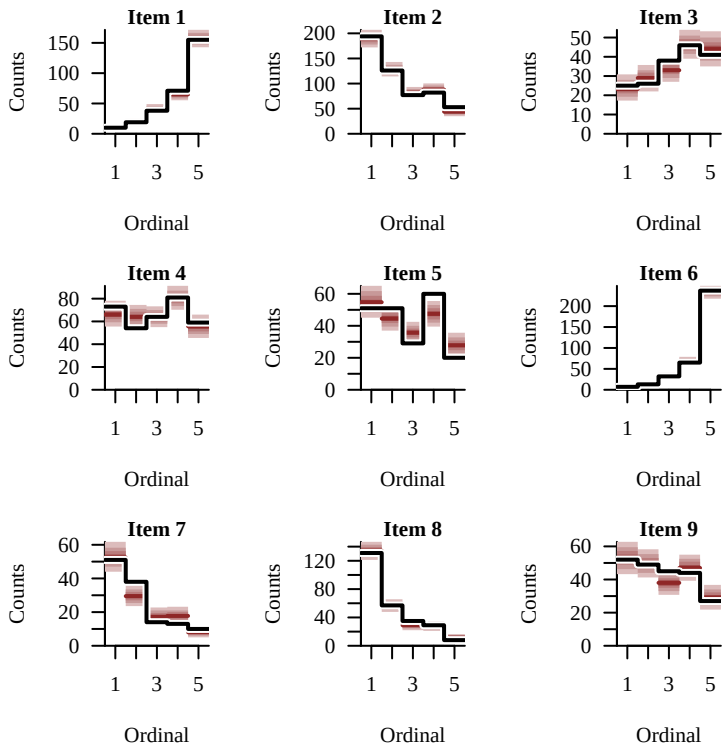
```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

The retrodictive performance of the model is great no matter how we slice and dice the observational space.

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

util$plot_hist_quantiles(samples1, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```

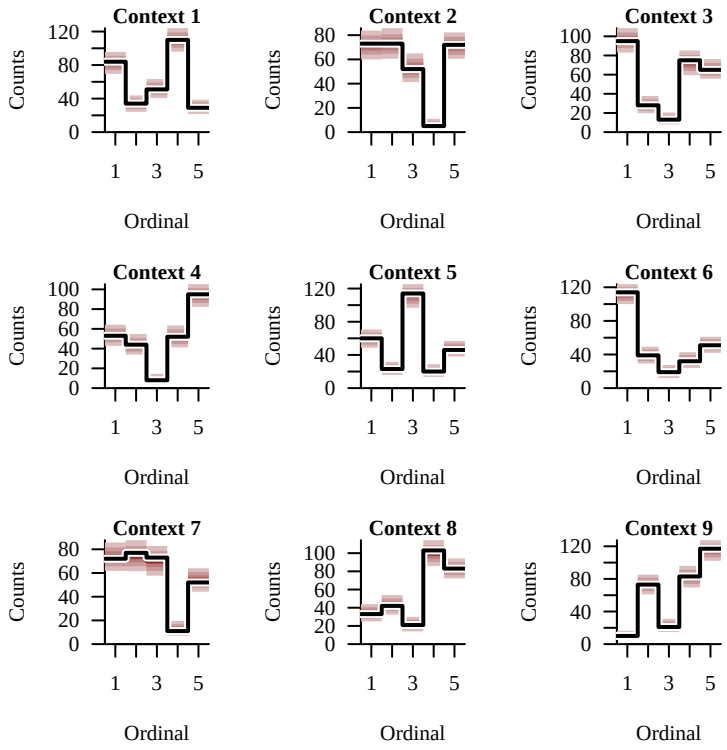


```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
  names <- sapply(which(data$item_idx == i),
                  function(n) paste0('y_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples1, names)

  item_obs <- data$y[data$item_idx == i]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=item_obs,
                           xlab="Ordinal",
                           main=paste('Item', i))
}
```



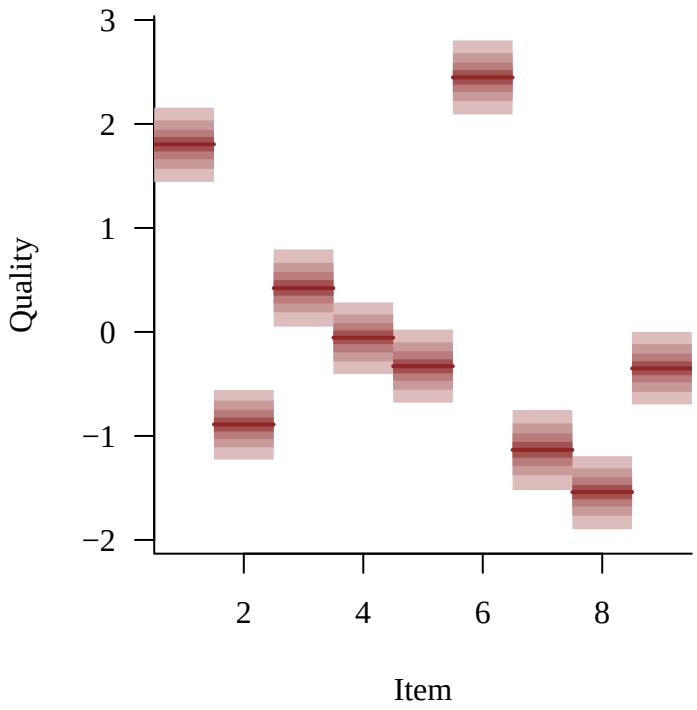

```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (j in 1:data$J) {
  names <- sapply(which(data$context_idx == j),
                  function(n) paste0('y_pred[', n, ']'))
```

```
  filtered_samples <- util$filter_expectands(samples1, names)

  context_obs <- data$y[data$context_idx == j]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=context_obs,
                           xlab="Ordinal",
                           main=paste('Context', j))
}
```



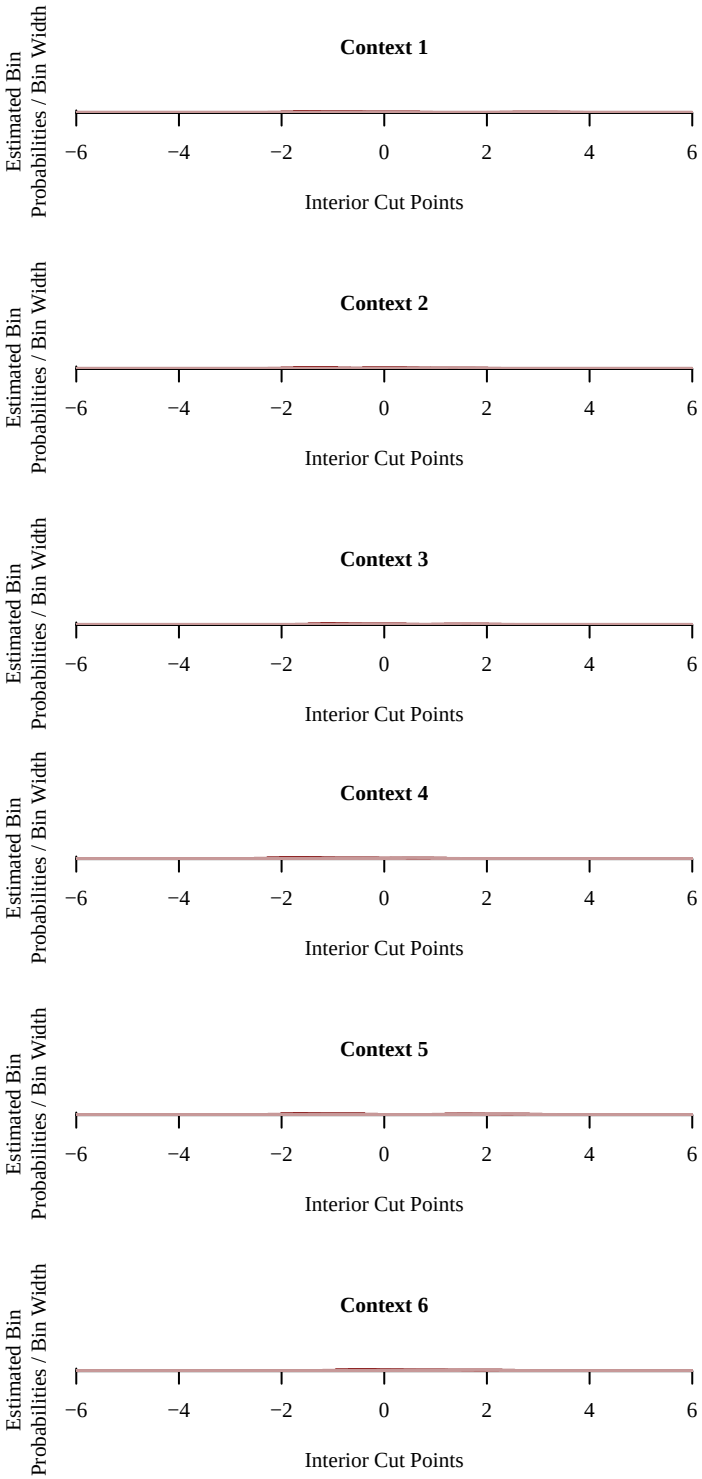
As before we can infer the quality of each item.

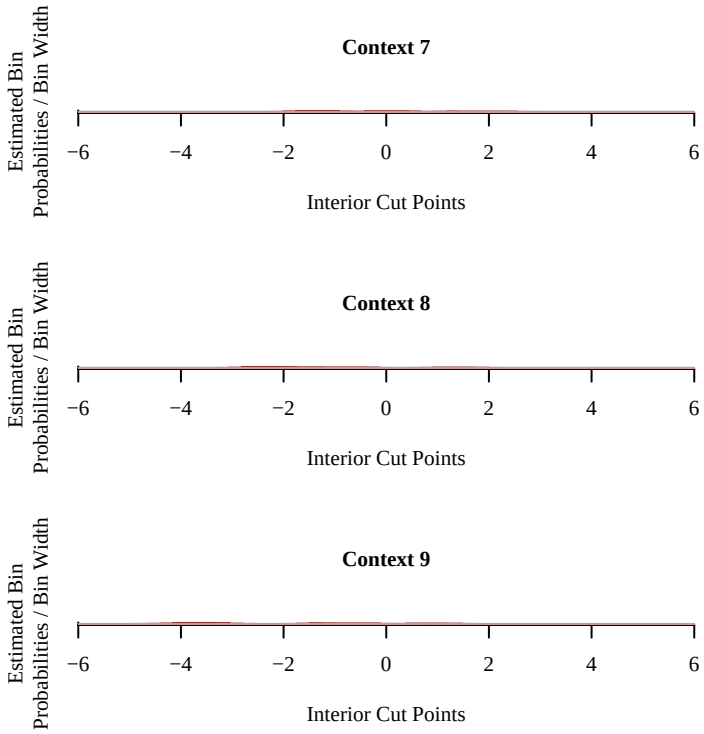
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$I, function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles(samples1, names,
                                     xlab="Item",
                                     ylab="Quality")
```



Now we can infer interior cut points for each context.

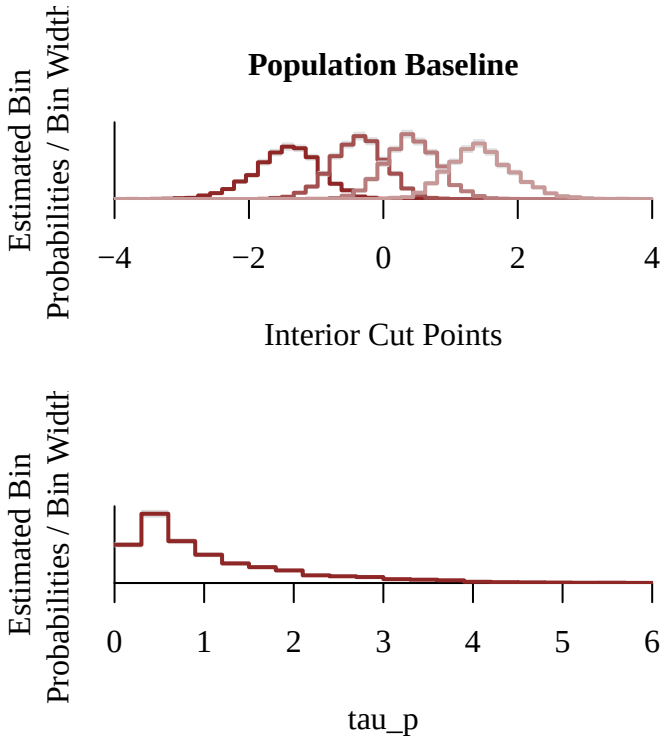
```
par(mfrow=c(3, 1), mar=c(5, 5, 5, 1))

for (j in 1:data$J) {
  prefix <- paste0('cut_points[', j, ',')
  name <- paste("Context", j)
  plot_cut_point_overlay(samples1, prefix,
                         c(-6, 6), 'Interior Cut Points', c(0, 1.5),
                         main=name)
}
```


Context 1
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points
Context 2
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points
Context 3
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points
Context 4
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points
Context 5
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points
Context 6
Estimated Bin Probabilities / Bin Width
−6 −4 −2 0 2 4 6
Interior Cut Points



Additionally we can now infer the behavior of the population that couples the context-specific interior cut points to each other. This allows us to, for example, make inferences and predictions for new contexts that we have not yet observed.

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

plot_cut_point_overlay(samples1, 'mu_cut_points[',
                       c(-4, 4), 'Interior Cut Points', c(0, 1.25),
                       main="Population Baseline")

util$plot_expectand_pushforward(samples1[['tau_p']],
                                20, flim=c(0, 6),
                                display_name='tau_p')
```



7.5.3 Hierarchical Ordinal Probabilities

Finally we can construct the same inferences by modeling the baseline ordinal probabilities in each context and then deriving the corresponding interior cut points.

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

As we saw previously the derived cut point implementation seems to be a bit more susceptible to higher Markov chain autocorrelations, but again only strong enough to reduce computational efficiency slightly.

```
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',
                                         'mu_p',
                                         'tau_p',
                                         'p'),
                                       check_arrays=TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

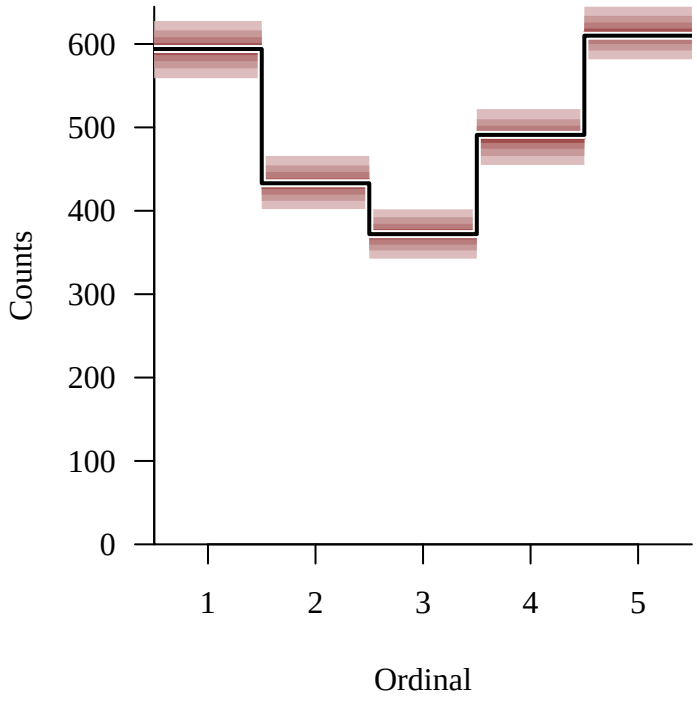
```
gamma[1]:
  Chain 4: hat{ESS} (69.426) is smaller than desired (100).

gamma[2]:
  Chain 2: hat{ESS} (92.565) is smaller than desired (100).
  Chain 4: hat{ESS} (62.679) is smaller than desired (100).

gamma[3]:
  Chain 4: hat{ESS} (76.174) is smaller than desired (100).

gamma[4]:
  Chain 4: hat{ESS} (65.889) is smaller than desired (100).

gamma[5]:
  Chain 4: hat{ESS} (69.305) is smaller than desired (100).

gamma[6]:
  Chain 4: hat{ESS} (69.709) is smaller than desired (100).

gamma[7]:
  Chain 4: hat{ESS} (75.883) is smaller than desired (100).

gamma[8]:
  Chain 4: hat{ESS} (67.523) is smaller than desired (100).

gamma[9]:
  Chain 4: hat{ESS} (67.403) is smaller than desired (100).

p[1,1]:
  Chain 4: hat{ESS} (74.821) is smaller than desired (100).

p[2,1]:
  Chain 4: hat{ESS} (76.615) is smaller than desired (100).
```

```
p[3,1]:
  Chain 4: hat{ESS} (72.310) is smaller than desired (100).

p[4,1]:
  Chain 4: hat{ESS} (87.472) is smaller than desired (100).

p[5,1]:
  Chain 4: hat{ESS} (77.825) is smaller than desired (100).

p[6,1]:
  Chain 4: hat{ESS} (75.315) is smaller than desired (100).

p[7,1]:
  Chain 4: hat{ESS} (74.741) is smaller than desired (100).

p[8,1]:
  Chain 4: hat{ESS} (94.810) is smaller than desired (100).

p[9,2]:
  Chain 4: hat{ESS} (82.693) is smaller than desired (100).

p[1,4]:
  Chain 4: hat{ESS} (86.336) is smaller than desired (100).

p[1,5]:
  Chain 4: hat{ESS} (88.966) is smaller than desired (100).

p[2,5]:
  Chain 4: hat{ESS} (73.600) is smaller than desired (100).

p[3,5]:
  Chain 4: hat{ESS} (78.068) is smaller than desired (100).

p[4,5]:
  Chain 4: hat{ESS} (82.506) is smaller than desired (100).

p[5,5]:
  Chain 4: hat{ESS} (82.858) is smaller than desired (100).

p[6,5]:
  Chain 4: hat{ESS} (87.351) is smaller than desired (100).
```

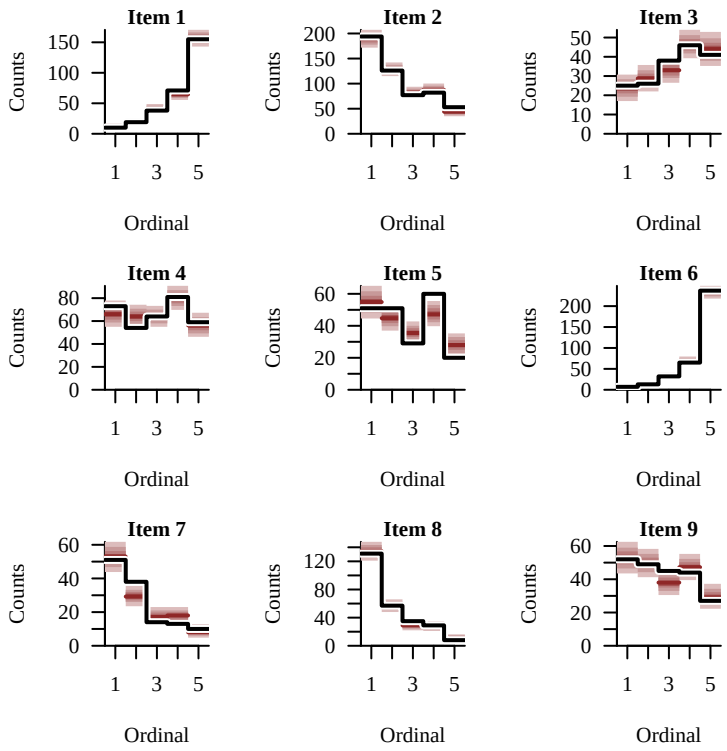
```
p[7,5]:
  Chain 4: hat{ESS} (83.040) is smaller than desired (100).

p[8,5]:
  Chain 4: hat{ESS} (73.962) is smaller than desired (100).

p[9,5]:
  Chain 4: hat{ESS} (72.169) is smaller than desired (100).


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

Otherwise the posterior predictive retrodictions are similarly well behaved.

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

util$plot_hist_quantiles(samples2, 'y_pred', 0.5, 5.5, 1,
                         baseline_values=data$y, xlab="Ordinal")
```

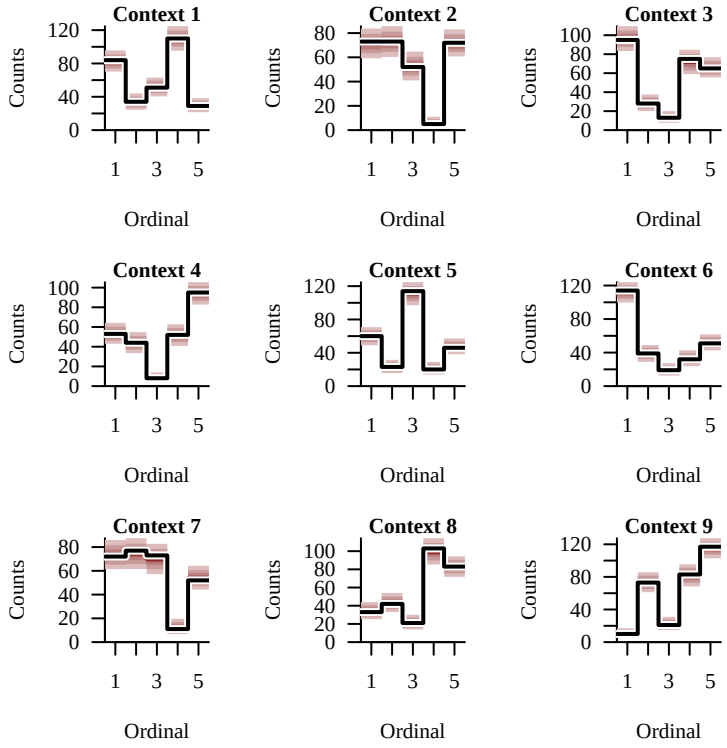


```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (i in 1:data$I) {
  names <- sapply(which(data$item_idx == i),
                  function(n) paste0('y_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples2, names)

  item_obs <- data$y[data$item_idx == i]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=item_obs,
                           xlab="Ordinal",
                           main=paste('Item', i))
}
```



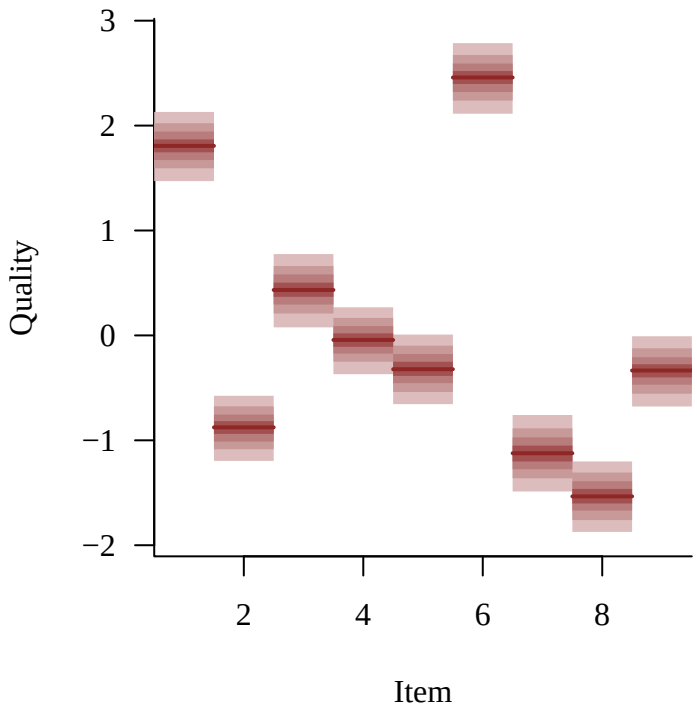

```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (j in 1:data$J) {
  names <- sapply(which(data$context_idx == j),
                  function(n) paste0('y_pred[', n, ']'))
```

```
  filtered_samples <- util$filter_expectands(samples2, names)

  context_obs <- data$y[data$context_idx == j]
  util$plot_hist_quantiles(filtered_samples, 'y_pred',
                           0.5, 5.5, 1,
                           baseline_values=context_obs,
                           xlab="Ordinal",
                           main=paste('Context', j))
}
```

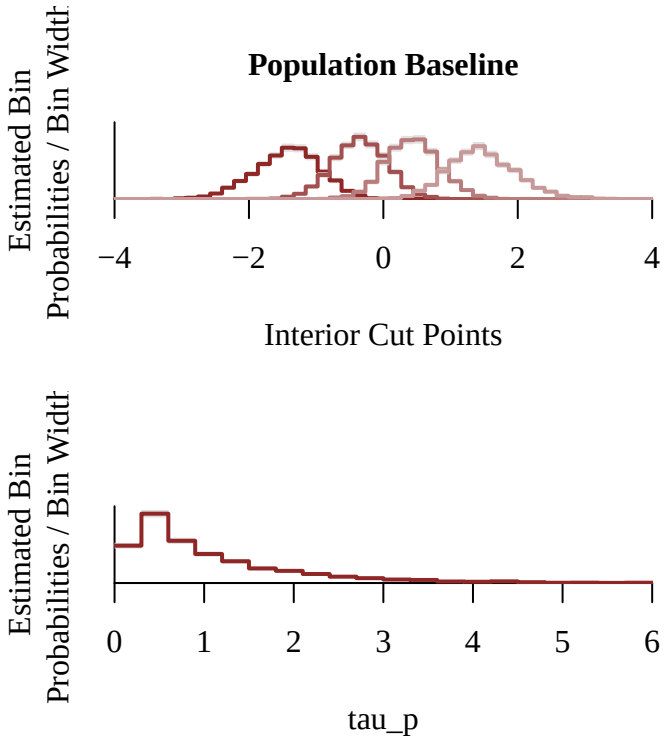


So too are the posterior inferences.

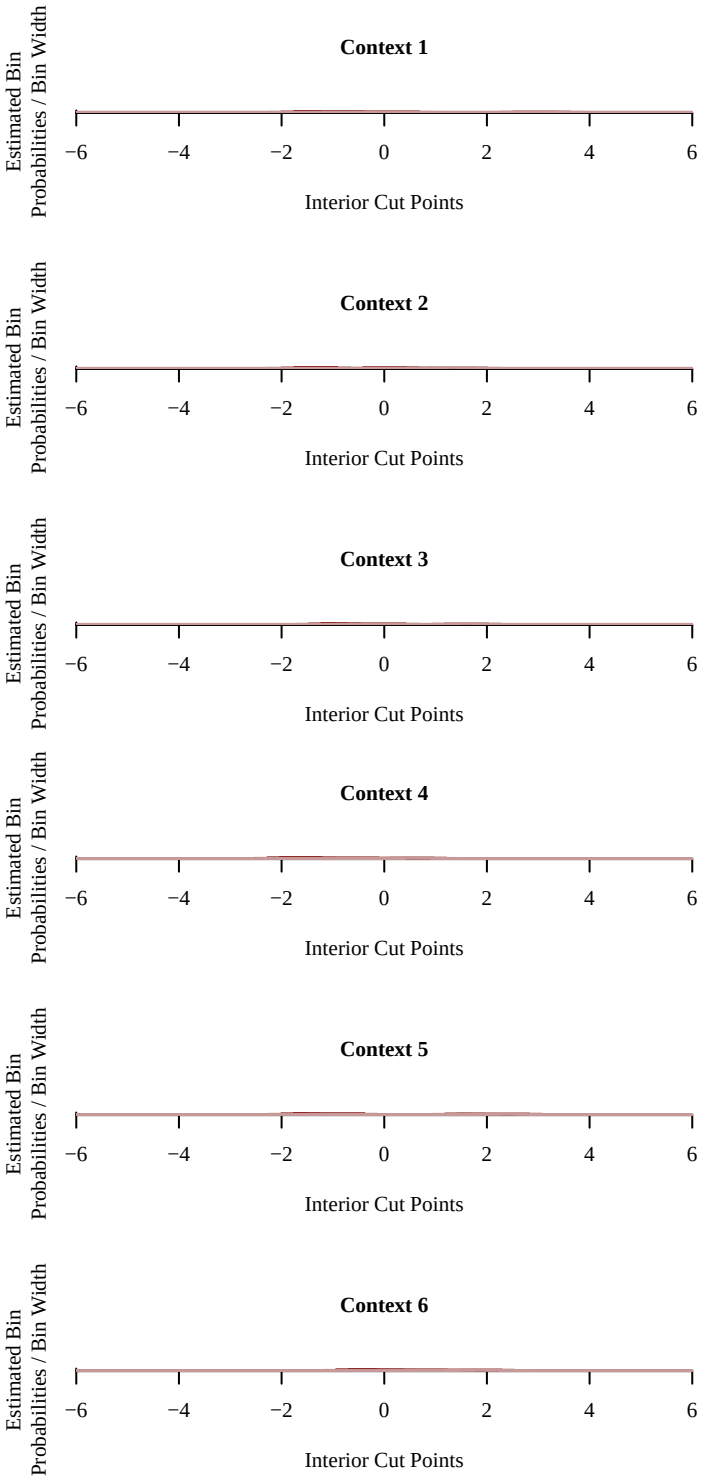
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$I, function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     xlab="Item",
                                     ylab="Quality")
```

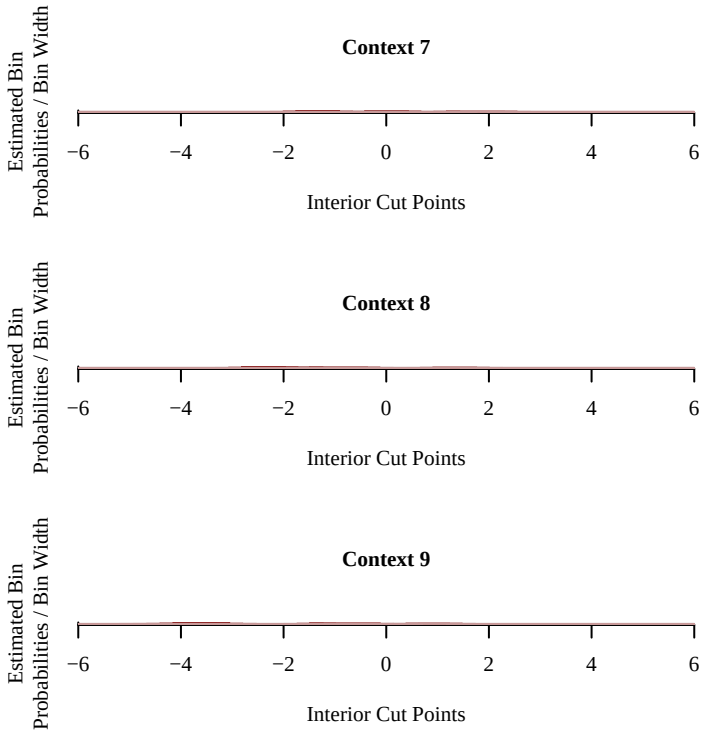



```
par(mfrow=c(2, 1), mar=c(5, 5, 3, 1))

plot_cut_point_overlay(samples2, 'mu_cut_points[',
                       c(-4, 4), 'Interior Cut Points', c(0, 1.25),
                       main="Population Baseline")

util$plot_expectand_pushforward(samples2[['tau_p']],
                                20, flim=c(0, 6),
                                display_name='tau_p')
```



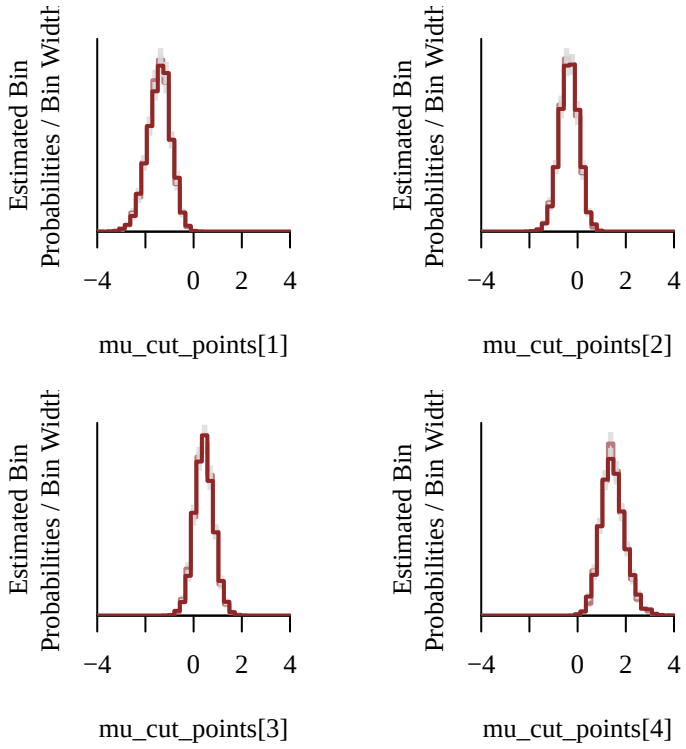

```
par(mfrow=c(3, 1), mar=c(5, 5, 5, 1))

for (j in 1:data$J) {
  prefix <- paste0('cut_points[', j, ',')
  name <- paste("Context", j)
  plot_cut_point_overlay(samples2, prefix,
                         c(-6, 6), 'Interior Cut Points', c(0, 1.5),
                         main=name)
}
```


Context 1
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points
Context 2
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points
Context 3
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points
Context 4
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points
Context 5
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points
Context 6
Estimated Bin Probabilities / Bin Width
−6
−4
−2
0
2
4
6
Interior Cut Points

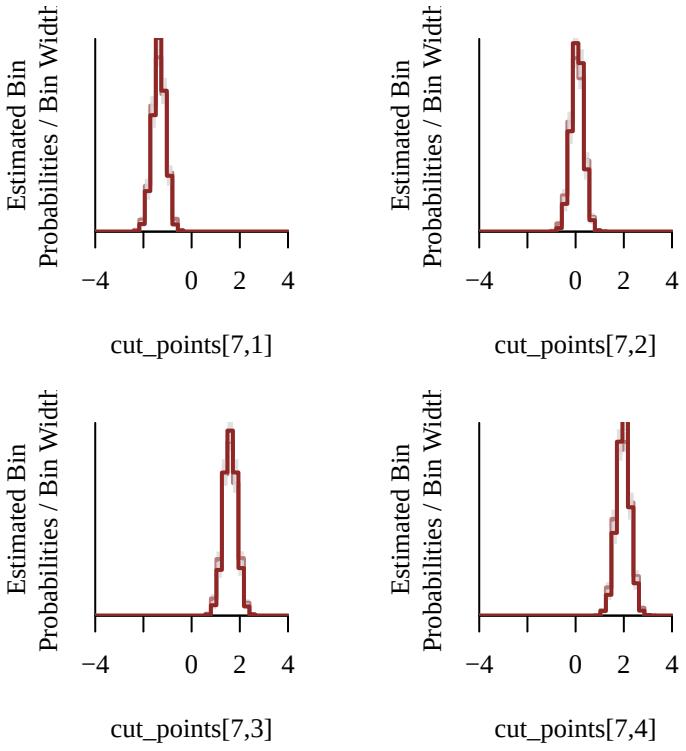


We can make a more formal comparison by overlaying the marginal posterior inferences and seeing almost no difference between the two approaches.

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

for (k in 1:(data$K - 1)) {
  name <- paste0('mu_cut_points[', k, ']')
  util$plot_expectand_pushforward(samples1[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD")
  util$plot_expectand_pushforward(samples2[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD",
                                  add=TRUE)
}
```




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

j <- 7

for (k in 1:(data$K - 1)) {
  name <- paste0('cut_points[', j, ',', k, ']')
  util$plot_expectand_pushforward(samples1[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_mid,
                                  border="#DDDDDDDD")
  util$plot_expectand_pushforward(samples2[[name]],
                                  35, flim=c(-4, 4),
                                  display_name=name,
                                  col=util$c_dark,
                                  border="#DDDDDDDD",
                                  add=TRUE)
}
```



8 Conclusion

Ordinal spaces are tricky because they add some structure but not as much structure as to which we might be accustomed in more common spaces like the integers and real lines. This makes probabilistic modeling on these spaces subtle. At the very least as we have to determine where the ordering structure is relevant and where it is not, both in theory and in practice.

Discretizing a probability distribution over a latent real line with cut points doesn't incorporate ordering into any model of baseline ordinal probabilities. It does, however, allow us to incorporate the ordering when we model how ordinal probabilities vary around that baseline. Moreover it establishes a modeling foundation on which we can incorporate heterogeneous baseline ordinal probabilities and more.

Acknowledgements

I thank jd for helpful comments as well as Max Kesin, Mark Donoghoe, shleeneu, Jan Siml, and Florian Pargent for helpful comments on a previous version of this chapter.

A very special thanks to everyone supporting me on Patreon: Adam Fleischhacker, Alejandro Navarro-Martínez, Alessandro Varacca, Alex D, Alexander Noll, Amit, Andrea Serafino, Andrew Mascioli, Andrew Rouillard, 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, Cartan, Cat Shark, 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, Danny Van Nest, Darshan Pandit, Darthmaluus , David Galley, David Wurtz, Doug Rivers, Dr. Jobo, Dr. Omri Har Shemesh, Dylan Maher, Dylan Spielman, 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, Joshua Miller, JU, 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, Name, Nathaniel Burbank, Nicholas Clark, Nicholas Cowie, Nick S, Octavio Medina, 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, Seth Axen, shira, Simon Duane, Simon Lilburn, Simone Sebben, Stefan Lorenz, Stephen Lienhard, Steve Harris, Stew Watts, Stone Chen, Susan Holmes, Svilup, Tao Ye, Tate Tunstall, Tatsuo Okubo, Teresa Ortiz, Theodore Dasher, Thomas Kealy, Thomas Siegert, Thomas Vladeck, Tobychev, Tony Wuersch, Tyler Burch, Virginia Fisher, Vitalie Spinu, 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/Edmonton
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] stringr_1.5.1      parallel_4.3.2     gridExtra_2.3      scales_1.3.0
 [9] yaml_2.3.8         fastmap_1.1.1      ggplot2_3.4.4      R6_2.5.1
[13] curl_5.2.0         knitr_1.45         tibble_3.2.1       munsell_0.5.0
[17] pillar_1.9.0       rlang_1.1.2        utf8_1.2.4         V8_4.4.1
[21] stringi_1.8.3      inline_0.3.19      xfun_0.41          RcppParallel_5.1.7
[25] cli_3.6.2          magrittr_2.0.3     digest_0.6.33      grid_4.3.2
[29] lifecycle_1.0.4    vctrs_0.6.5        evaluate_0.23      glue_1.6.2
[33] QuickJSR_1.0.8     codetools_0.2-19   stats4_4.3.2       pkgbuild_1.4.3
[37] fansi_1.0.6        colorspace_2.1-0   rmarkdown_2.25     matrixStats_1.2.0
[41] tools_4.3.2        loo_2.6.0          pkgconfig_2.0.3    htmltools_0.5.7
```

Stan
Program 2 `ordinal\_logistic\_derived.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Derive cut points from baseline probabilities
  // and latent logistic density function.
  vector derived_cut_points(vector p) {
    int K = num_elements(p);
    vector[K - 1] c;

    real cum_sum = 0;
    for (k in 1:(K - 1)) {
      cum_sum += p[k];
      c[k] = logit(cum_sum);
    }

    return c;
  }
}

data {
  int<lower=1> K;                    // Number of ordinal categories
  int<lower=1> N;                    // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  simplex[K] p; // Category probabilities
}

transformed parameters {
  // Interior cut points
  ordered[K - 1] cut_points = derived_cut_points(p);
}
```

Stan
Program 3 ordered_logistic_uniform.stan

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }
}

data {
  int<lower=1> K;                   // Number of ordinal categories
  int<lower=1> N;                   // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  ordered[K - 1] cut_points; // Interior cut points
}

model {
  // Implicit uniform prior model

  // Observational model
  y ~ ordinal_logistic(cut_points);
}

generated quantities {
  array[N] int<lower=1, upper=K> y_pred;
  for (n in 1:N)
    y_pred[n] = ordinal_logistic_rng(cut_points);
}
```

Stan
Program 4 ordinal_logistic_induced_dirichlet.stan

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Log probability density function over cut points
  // induced by a Dirichlet probability density function
  // over baseline probabilities and a 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> K;                  // Number of ordinal categories
  int<lower=1> N;                  // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  ordered[K - 1] cut_points; // Interior cut points
}
```

Stan
Program 5 `ordinal\_logistic\_derived.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Derive cut points from baseline probabilities
  // and latent logistic density function.
  vector derived_cut_points(vector p) {
    int K = num_elements(p);
    vector[K - 1] c;

    real cum_sum = 0;
    for (k in 1:(K - 1)) {
      cum_sum += p[k];
      c[k] = logit(cum_sum);
    }

    return c;
  }
}

data {
  int<lower=1> K;                    // Number of ordinal categories
  int<lower=1> N;                    // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  simplex[K] p; // Category probabilities
}

transformed parameters {
  // Interior cut points
  ordered[K - 1] cut_points = derived_cut_points(p);
}
```

Stan
Program 6 `ordinal\_logistic\_uniform.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }
}

data {
  int<lower=1> K;                   // Number of ordinal categories
  int<lower=1> N;                   // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  ordered[K - 1] cut_points; // Interior cut points
}

model {
  // Implicit uniform prior model

  // Observational model
  y ~ ordinal_logistic(cut_points);
}

generated quantities {
  array[N] int<lower=1, upper=K> y_pred;
  for (n in 1:N)
    y_pred[n] = ordinal_logistic_rng(cut_points);
}
```

Stan
Program 7 `ordinal\_logistic\_induced\_dirichlet.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent standard logistic density function.
  real ordinal_logistic_lpmf(int[] y, vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_logistic_rng(vector c) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Log probability density function over cut points
  // induced by a Dirichlet probability density function
  // over baseline probabilities and a 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> K;                  // Number of ordinal categories
  int<lower=1> N;                  // Number of observations
  array[N] int<lower=1, upper=K> y; // Observed categories
}

parameters {
  ordered[K - 1] cut_points; // Interior cut points
}
```

Stan
Program 8 `ordinal\_shifted\_logistic\_uniform.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }
}

data {
  int<lower=1> I; // Number of items
  int<lower=1> K; // Number of ordinal categories
  int<lower=1> N; // Number of observations

  array[N] int<lower=1, upper=I> item_idx; // Observed items
  array[N] int<lower=1, upper=K> y;        // Observed categories
}

parameters {
  vector[I] gamma;           // Item qualities
  ordered[K - 1] cut_points; // Interior cut points
}

model {
  // Implicit uniform prior model

  // Observational model
  for (n in 1:N)
    y[n] ~ ordinal_shifted_logistic(cut_points, gamma[item_idx[n]]);
}

generated quantities {
  array[N] int<lower=1, upper=K> y_pred;
  for (n in 1:N)
    y_pred[n] = ordinal_shifted_logistic_rng(cut_points,
                                             gamma[item_idx[n]]);
```

Stan
Program 9 `ordinal\_shifted\_logistic\_anchor.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }
}

data {
  int<lower=1> I; // Number of items
  int<lower=1> K; // Number of ordinal categories
  int<lower=1> N; // Number of observations

  array[N] int<lower=1, upper=I> item_idx; // Observed items
  array[N] int<lower=1, upper=K> y;        // Observed categories
}

parameters {
  vector[I - 1] gamma_free;  // Free item qualities
  ordered[K - 1] cut_points; // Interior cut points
}

transformed parameters {
  vector[I] gamma = append_row([0]', gamma_free);
}

model {
  // Implicit uniform prior model

  // Observational model
  for (n in 1:N)
    y[n] ~ ordinal_shifted_logistic(cut_points, gamma[item_idx[n]]);
}

generated quantities {
```

Stan
Program 10 `ordinal\_shifted\_logistic\_induced\_dirichlet.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Log probability density function over cut points
  // induced by a Dirichlet probability density function
  // over baseline probabilities and a 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> I; // Number of items
  int<lower=1> K; // Number of ordinal categories
  int<lower=1> N; // Number of observations

  array[N] int<lower=1, upper=I> item_idx; // Observed items
```

Stan
Program 11 ordered_logistic_induced_dirichlet.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> I;                    // Number of items
  int<lower=1> K;                    // Number of ordinal categories
  int<lower=1> N;                    // Number of observations

  array[N] int<lower=1, upper=I> item_idx; // Observed items
  array[N] int<lower=1, upper=K> y;        // Observed categories
}

parameters {
  vector[I] gamma;           // Item qualities
  ordered[K - 1] cut_points; // Interior cut points
}

model {
  // Prior model
  gamma ~ normal(0, 5 / 2.32);
  cut_points ~ induced_dirichlet(rep_vector(1, K));

  // Observational model
  y ~ ordered_logistic(gamma[item_idx], cut_points);
}

generated quantities {
  array[N] int<lower=1, upper=K> y_pred;
  for (n in 1:N)
    y_pred[n] = ordered_logistic_rng(gamma[item_idx[n]], cut_points);
```

Stan
Program 12 `ordinal\_shifted\_logistic\_derived.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Derive cut points from baseline probabilities
  // and latent logistic density function centered
  // at zero.
  vector derived_cut_points(vector p) {
    int K = num_elements(p);
    vector[K - 1] c;

    real cum_sum = 0;
    for (k in 1:(K - 1)) {
      cum_sum += p[k];
      c[k] = logit(cum_sum);
    }

    return c;
  }
}

data {
  int<lower=1> I; // Number of items
  int<lower=1> K; // Number of ordinal categories
  int<lower=1> N; // Number of observations

  array[N] int<lower=1, upper=I> item_idx; // Observed items
  array[N] int<lower=1, upper=K> y;        // Observed categories
}

parameters {
  vector[I] gamma; // Item qualities
```

Stan
Program 13 `ordinal\_shifted\_logistic\_induced\_dirichlet\_hier.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Log probability density function over cut points
  // induced by a Dirichlet probability density function
  // over baseline probabilities and a 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;
  }

  // Derive cut points from baseline probabilities
  // and latent logistic density function.
  vector derived_cut_points(vector p) {
    int K = num_elements(p);
    vector[K - 1] c;

    real cum_sum = 0;
```

Stan
Program 14 `ordinal\_shifted\_logistic\_derived\_hier.stan`

```
functions {
  // Ordinal probability mass function assuming a
  // latent shifted logistic density function.
  //
  // Positive gamma shifts baseline ordinal
  // probabilities towards larger values.
  real ordinal_shifted_logistic_lpmf(int y, vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_lpmf(y | p);
  }

  // Ordinal pseudo-random number generator assuming
  // a latent standard logistic density function.
  int ordinal_shifted_logistic_rng(vector c, real gamma) {
    int K = num_elements(c) + 1;
    vector[K - 1] Pi = inv_logit(c - gamma);
    vector[K] p = append_row(Pi, [1]') - append_row([0]', Pi);
    return categorical_rng(p);
  }

  // Derive cut points from baseline probabilities
  // and latent logistic density function.
  vector derived_cut_points(vector p) {
    int K = num_elements(p);
    vector[K - 1] c;

    real cum_sum = 0;
    for (k in 1:(K - 1)) {
      cum_sum += p[k];
      c[k] = logit(cum_sum);
    }

    return c;
  }
}

data {
  int<lower=1> I; // Number of items
  int<lower=1> J; // Number of contexts
  int<lower=1> K; // Number of ordinal categories
  int<lower=1> N; // Number of observations

  array[N] int<lower=1, upper=I> item_idx;    // Observed items
  array[N] int<lower=1, upper=J> context_idx; // Observed contexts
  array[N] int<lower=1, upper=K> y;           // Observed categories
}

parameters {
```