Discrete Choice Modeling

Michael Betancourt

February 2026

Table of contents

1 Utility Maximization Models **1**
- 1.1 Agents, Alternatives, and Choice Sets . . . 2
- 1.2 Utilities . . . 2
- 1.3 Choice Probabilities . . . 2
- 1.4 Deriving Choice Probabilities From Cumulative Distribution Functions . . . 4

2 Substitution Patterns **5**
- 2.1 Independence from Irrelevant Alternatives . . . 6
  - 2.1.1 Removing An Alternative From The Choice Set . . . 6
  - 2.1.2 Adding An Alternative To The Choice Set . . . 8
  - 2.1.3 The Blue Bus/Red Bus Problem . . . 9
- 2.2 Sensitivities and Elasticities . . . 11
- 2.3 Perfect Replacements . . . 13
- 2.4 Expected Substitution Patterns . . . 15
- 2.5 Inferring Substitution Patterns . . . 16

3 Independent Utility Models **17**
- 3.1 The Independent Gumbel Model . . . 18
- 3.2 Redundancies of the Independent Gumbel Model . . . 20
  - 3.2.1 Translation Redundancy . . . 20
  - 3.2.2 Scale Redundancy . . . 21
  - 3.2.3 Resolving Both Redundancies Without Abusing Notation . . . 23
- 3.3 Substitution Patterns of the Independent Gumbel Model . . . 24
- 3.4 Demonstration . . . 25
  - 3.4.1 Setup . . . 25
  - 3.4.2 Basics . . . 26
  - 3.4.3 Independent Gumbel Substitution Patterns . . . 41

4 Coupled Utility Models **61**
4.1 Nested Gumbel Model . . . 62
4.1.1 Nested Gumbel As A Generalization of Independent Gumbel . . . 62
4.1.2 Choice Probabilities . . . 63
4.1.3 Substitution Patterns . . . 67
4.2 Generalized Nested Gumbel Model . . . 72
4.3 The Probit Model . . . 74
4.4 Demonstration . . . 75
4.4.1 Basics . . . 75
4.4.2 Nested Gumbel Substitution Patterns . . . 95

5 Mixed Discrete Choice Models **119**
5.1 Mixed Choice Probabilities . . . 120
5.2 Mixed Substitution Patterns . . . 121
5.2.1 Ratios of Choice Probabilities . . . 122
5.2.2 Elasticities . . . 123
5.2.3 Strong Coupling Makes Good Substitutes . . . 124
5.3 The Multivariate Normal Baseline Mixing Kernel . . . 126
5.3.1 General Construction . . . 126
5.3.2 Special Cases . . . 127
5.4 Relative Implementations . . . 128
5.4.1 Deriving The Relative Baseline Utility Model . . . 129
5.4.2 Eliminating Redundancies In The Relative Baseline Utilities . . . 130
5.4.3 Eliminating Latent Redundancies . . . 132
5.4.4 Cleaning Up . . . 133
5.4.5 Demonstration . . . 134
5.4.6 Introducing Relative Agent Scales . . . 140
5.5 Practical Implementation . . . 141
5.6 Demonstration . . . 143

6 Derived Baseline Utility Models **155**
6.1 Factor Models . . . 155
6.1.1 Agent and Alternative Factors . . . 156
6.1.2 Conjoint Models . . . 157
6.1.3 Relative Factor Level Contributions . . . 158
6.2 Conjoint Model Demonstration . . . 159
6.2.1 Explore Data . . . 159
6.2.2 Specifying An Observational Model . . . 161
6.2.3 Specifying A Prior Model . . . 163
6.2.4 Model Implementation . . . 164
6.2.5 Posterior Quantification . . . 164
6.2.6 Consumer Appeal . . . 168

6.3 Regression Models . . . 173
6.3.1 Variate-Covariate Modeling . . . 173
6.3.2 Agent, Alternative, and Agent-Alternative Covariates . . . 174
6.3.3 Baseline Utility Functional Models . . . 175
6.3.4 Relative Baseline Utility Functional Models . . . 176
6.4 Perfect Replacements . . . 178
6.5 Regression Model Demonstration . . . 181
6.5.1 Explore Data . . . 181
6.5.2 Specifying An Observational Model . . . 184
6.5.3 Specifying A Prior Model . . . 185
6.5.4 Posterior Quantification . . . 186
6.5.5 Posterior Retrodictive Checks . . . 187
6.5.6 Posterior Inferences . . . 189

7 Conclusion **193**

A: Onerous Calculations **194**
A.1 Independent Gumbel Model Calculations . . . 194
A.2 Independent Generalized Extreme Value Model Calculations . . . 197
A.3 Nested Gumbel Model Calculations . . . 199
A.3.1 Choice Probabilities . . . 200
A.3.2 Sensitivities . . . 203
A.4 Generalized Nested Gumbel Model Calculations . . . 207

Acknowledgements **210**

References **211**

License **212**

Original Computing Environment **212**

Analyzing observed choices from a finite set is a common task in many fields. Survey data, for example, often takes the form of individual respondents answering questions by choosing from a set of possible answers. Similarly, sales data is often derived from the purchasing choices of individual consumers in a market. Moreover, choice observations are not limited to humans: some ecological analysis study animal behavior by observing choices in consumption, movement, and more.

Modeling observed choices has long been a topic of interest in psychology, marketing Schwarz, Chapman, and Feit (2020), and economics (Train 2009), each of which introduces their own emphasis, terminology, and notation. In this chapter I present choice modeling from a more general probabilistic modeling perspective. This perspective helps to contextualize conventional modeling techniques while also clarifying their full generality.

We will first consider the general analysis of choice as decision making under uncertainty, and how decision making behavior can change with the possible alternatives. Then we will review some of the more common discrete choice models with a particular focus on the underlying assumptions and their consequences. Along the way I will demonstrate the more theoretical insights with explicit examples.

1 Utility Maximization Models

If we want to treat choices as the result of a decision making process then we need to specify who is making decisions, what decisions they can make, and how the arrive at any particular decision.

1.1 Agents, Alternatives, and Choice Sets

In this chapter I will refer to any entity responsible for making a decision as an **agent**. Depending on the application, agents can be individuals, such as people or organisms, or even collectives, like companies and communities. Generally we will be interested in not one single agent but rather an ensemble of N agents.

A **choice set** is a collection of J **alternatives** from which each agent can select their preferred option. We will assume that the choice set is both **exclusive** and **exhaustive** so that each agent always chooses one, but only one, preferred alternative at a time.

The choice set is not always fixed; in many settings the possible alternatives vary from one decision to the next. In this case I will use choice set to refer to the complete set of all possible alternatives and **choice subset** to refer to any particular subset of available options in a given decision.

1.2 Utilities

If decisions between the alternatives in a choice set are made **purposefully** then can model those decisions by quantifying each agent's preference for the alternatives. The quantification of individual preferences into a one-dimensional real number, taking into account all relevant benefits and costs, is known as a **utility**. Here I will denote the utility agent n gains by choosing alternative J by $u_{nj} \in \mathbb{R}$.

Utilities are not generally universal but rather can be influenced by the context of a particular decision. For example, utilities can in theory depend on number of alternatives in each choice subset, the number of choice subsets considered, the order in which those choice subsets are presented to a given agent, and so on. Because of this we have to be especially careful when extrapolating utilities from one context to another.

In particular the utilities agents allocate, and the decisions that those utilities motivate, can be different if an agent is asked to make a hypothetical choice and if they are making an actual choice. These inconsistent utilities complicate analyses that attempt to infer consistent behavior from multiple sources. For instance, the outcome of surveys and focus groups, which propose hypothetical consumer decisions, may be inconsistent with sales data, which is derived from actual consumer decisions. In marketing and economics these varying behaviors are known as **stated preferences** and **revealed preferences**, respectively.

1.3 Choice Probabilities

When the utilities that an agent assigns to each alternative are known exactly, the resulting decisions become completely deterministic. Agent will choose alternative j if

$$u_{nj} > u_{nj'}$$

for all $j' \neq j$. Because utilities are real-valued, exact ties are exceedingly improbable and can be safely ignored.

Utilities, however, are rarely known exactly in practical applications. The understanding of utilities might, for example, be limited by incomplete or untrustworthy elicitation of the agents, varying or otherwise inconsistent preferences, or any number of other unknown or unquantifiable factors.

Regardless of why utilities are uncertain in a given analysis, we can always consistently model that uncertainty with a probability distribution over all of the utilities. Probabilistic utility models are sometimes referred to as **behavioral models**.

Because the utilities are all real-valued, we can always specify a probabilistic utility model with a joint probability density function,

$$p(u_{11}, \ldots, u_{1J}, \ldots, u_{N1}, \ldots, u_{NJ}).$$

In applications where agents make decisions independently from each other, we can model the utilities for each agent with an independent product distribution,

$$p(u_{11}, \ldots, u_{1J}, \ldots, u_{N1}, \ldots, u_{NJ}) = \prod_{n=1}^{N} p(u_{n1}, \ldots, u_{nJ}).$$

With uncertainty utilities decisions are no longer deterministic but rather probabilistic. Agent n will choose alternative j for any configurations of the utilities that fall into the rectangular subset

$$\mathsf{u}_{nj} = \bigcap_{j' \neq j} \{u_{nj} > u_{nj'}\}.$$

The probability that agent n chooses alternative j is given by the probability allocated to this subset,

$$\begin{aligned} q_{nj} &= \pi(\mathsf{u}_{nj}) \\ &= \int \prod_{j'=1}^{J} \mathrm{d}u_{nj'} \, p(u_{n1}, \ldots, u_{nJ}) \, I_{\mathsf{u}_{nj}}(u_{n1}, \ldots, u_{nJ}). \end{aligned}$$

These alternative probabilities then define a **discrete choice model** for an observed choice y made by agent n,

$$p(y) = \text{categorical}(y \mid q_{n1}, \ldots, q_{nJ}) = q_{ny}.$$

Many presentations of discrete choice models implicitly assume that each agent chooses from a given choice set only once. In this case each observed choice can be uniquely labeled by the agent who made it, y_n. When agents can make multiple choices, however, this labeling convention becomes ungainly.

To avoid any potential confusion in this chapter, I will label individual choice observations with the separate index

$$m \in 1, \ldots M.$$

We can then denote the agent making that choice as n_m, so that the discrete choice model becomes

$$p(y_m) = \text{categorical}(y_m \mid q_{n_m 1}, \ldots, q_{n_m J}) = q_{n(m) y_m}.$$

This notation also allows us to explicitly denote varying choice subsets, for example

$$p(y_m) = \text{categorical}(y_m \mid q_{n_m 1}, \ldots, q_{n_m J_m}) = q_{n_m y_m}.$$

1.4 Deriving Choice Probabilities From Cumulative Distribution Functions

Because the joint subset u_{nj} is defined by the intersection of component subsets, we can write its indicator function as a product of component indicator functions,

$$I_{\{u_{nj} > u_{nj'}\}}(u_{nj}) = \prod_{j' \neq j} I_{\{u_{nj} > u_{nj'}\}}(u_{nj}).$$

This then allows us to substantially simplify the choice probabilities,

$$\begin{aligned}
q_{nj} &= \int \left[\prod_{j'=1}^{J} \mathrm{d}u_{nj'} \right] \, p(u_{n1}, \ldots, u_{nJ}) \, I_{\mathsf{u}_{nj}}(u_{n1}, \ldots, u_{nJ}) \\
&= \int \left[\prod_{j'=1}^{J} \mathrm{d}u_{nj'} \right] \, p(u_{n1}, \ldots, u_{nJ}) \, \prod_{j' \neq j} I_{\{u_{nj} > u_{nj'}\}}(u_{nj}) \\
&= \int \mathrm{d}u_{nj} \, \left[\prod_{j' \neq j} \int \mathrm{d}u_{nj'} \, I_{\{u_{nj} > u_{nj'}\}}(u_{nj}) \right] \, p(u_{n1}, \ldots, u_{nJ}) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \, \left[\prod_{j' \neq j} \int_{-\infty}^{u_{nj}} \mathrm{d}u_{nj'} \right] \, p(u_{n1}, \ldots, u_{nJ}).
\end{aligned}$$

Conveniently, this integral can always be written in terms of the cumulative distribution function of the given probabilistic utility model.

The joint probability density function

$$p(x_1, \ldots, x_J)$$

defines the joint cumulative distribution function

$$\Pi(x_1, \ldots, x_J) = \left[\prod_{j'=1}^{J} \int_{-\infty}^{x_j} \mathrm{d}x'_j \right] \, p(x'_1, \ldots, x'_J).$$

Taking a partial derivative with respect to the jth input gives

$$\frac{\partial \Pi}{\partial x_j}(x_1, \ldots, x_J) = \left[\prod_{j' \neq j} \int_{-\infty}^{x_j} \mathrm{d}x'_j \right] \, p(x'_1, \ldots, x'_J).$$

If we set all of the inputs to u_{nj} then this is exactly the integrand defining the choice probabilities! Consequently we can write the choice probabilities as

$$\begin{aligned}
q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \, \left[\prod_{j' \neq j} \int_{-\infty}^{u_{nj}} \mathrm{d}u_{nj'} \right] \, p(u_{n1}, \ldots, u_{nJ}) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \, \frac{\partial \Pi}{\partial x_j}(u_{nj}, \ldots, u_{nj}),
\end{aligned}$$

or, a bit more compactly,

$$q_{nj} = \int_{-\infty}^{+\infty} \mathrm{d}u \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u).$$

In addition to potentially simplifying calculations, these formulas also allow us to define probabilistic utility model not in terms of a joint probability density function but rather with a joint cumulative distribution function.

2 Substitution Patterns

Once we can derive choice probabilities, we can consider how choice probabilities change when the behavior of one or more alternatives changes.

For example, if the jth alternative is replaced with a worse option then we would expect that q_{nj} will decrease. How, however, should the excess probability be redistributed amongst the other alternatives? Do some alternatives benefit more than others, or do they all increase in a similar manner?

If q_{nj} is taken all the way to zero, then the jth item is effectively removed form the choice set. In many applications a critical question is how choice probabilities change when alternatives are removed or added to the choice set or, more generally, how choice probabilities vary across different choice subsets.

All of these behaviors are examples of **substitution patterns**. Without the right substitution patterns, a discrete choice model will not be able to make inferences and predictions that generalize beyond a particular choice subset. Consequently, understanding the substitution patterns implied by a given probabilistic utility model is critical.

Quantifying substitution patterns through the behavior of choice probabilities directly is complicated by the summation constraint

$$\sum_{j=1}^{J} q_{nj} = 1.$$

Any change in the choice probability for one alternative *always* results in a corresponding change to at least one other choice probability.

Instead substitution patterns are often better quantified by *proportional* changes to choice probabilities. In particular, ratios of choice probabilities are often much more interpretable.

2.1 Independence from Irrelevant Alternatives

One prototypical substitution pattern arises when the *relative* preference between two alternatives is unaffected by changes in preferences of other alternatives.

For example, if the j_3th choice probability q_{nj_3} decreases then we are left with an excess of probability that has to be redistributed amongst the other alternatives. If q_{nj_1} increases more than q_{nj_2} then the relative preference for alternative j_1 over alternative j_2 will increase.

The relative preference for alternatives j_1 and j_2 will be unaffected by a change in preference for alternative j_3 if, and only if, any increase/decrease in q_{nj_1} is complemented by the same proportional increase/decrease in q_{nj_2}.

In other words the ratio of choice probabilities,

$$\frac{q_{nj_1}(z)}{q_{nj_2}(z)},$$

must be independent of q_{nj_3}. In this case the behavior alternative j_3 is **irrelevant** to the relative behavior of alternatives j_1 and j_2.

A discrete choice model exhibits **independence from irrelevant alternatives** when the ratio of choice probabilities for *every* pair of alternatives is invariant to changes in any other choice probabilities. Independence from irrelevant alternatives results in very particular substitution patterns that can be both conveniently and overly restrictive.

2.1.1 Removing An Alternative From The Choice Set

Independence from irrelevant alternatives, for example, implies that the ratio of choice probabilities between each pair of alternatives will be the same even if the choice probability for a remaining alternative goes to zero and we eliminate it from the choice set entirely. This behavior allows us to derive the choice probabilities for *any* choice subset.

To demonstrate, let's consider four alternatives with the choice probability ratios

$$\begin{aligned}
&\frac{q_{n1}}{q_{n2}} = r_{12} \qquad &&\frac{q_{n1}}{q_{n3}} = r_{13} \qquad &&\frac{q_{n1}}{q_{n4}} = r_{14} \\
&\frac{q_{n2}}{q_{n3}} = r_{23} \qquad &&\frac{q_{n2}}{q_{n4}} = r_{24} \qquad &&\frac{q_{n3}}{q_{n4}} = r_{34}.
\end{aligned}$$

Under independence from irrelevant alternatives, the ratios of choice probabilities will be the same if we remove the second alternative from the choice set,

$$\frac{q'_{n1}}{q'_{n3}} = r_{13} \qquad \frac{q'_{n1}}{q'_{n4}} = r_{14} \qquad \frac{q'_{n3}}{q'_{n4}} = r_{34}.$$

Along with the summation constraint

$$q'_{n1} + q'_{n3} + q'_{n4} = 1$$

these ratios are enough to completely determine the choice probabilities for this choice subset.

Using the first two ratios we can isolate the first choice probability,

$$\begin{aligned} q'_{n1} + q'_{n3} + q'_{n4} &= 1 \\ q'_{n1} + \frac{1}{r_{13}} q'_{n1} + \frac{1}{r_{14}} q'_{n1} &= 1 \\ q'_{n1} \left(1 + \frac{1}{r_{13}} + \frac{1}{r_{14}} \right) &= 1 \\ q'_{n1} &= \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}}. \end{aligned}$$

Then we can isolate the next choice probability,

$$\begin{aligned} q'_{n1} + q'_{n3} + q'_{n4} &= 1 \\ \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}} + q'_{n3} + \frac{1}{r_{34}} q'_{n3} & \\ &= 1 \\ q'_{n3} \left(1 + \frac{1}{r_{34}} \right) &= 1 - \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}} \\ q'_{n3} &= \frac{1}{1 + \frac{1}{r_{34}}} \left(1 - \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}} \right). \end{aligned}$$

The final choice probability follows from the summation constraint,

$$\begin{aligned} q'_{n1} + q'_{n3} + q'_{n4} &= 1 \\ q'_{n4} &= 1 - q'_{n1} - q'_{n3} \\ &= 1 - \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}} - \frac{1}{1 + \frac{1}{r_{34}}} \left(1 - \frac{1}{1 + \frac{1}{r_{13}} + \frac{1}{r_{14}}} \right). \end{aligned}$$

All of this is to say that, under independence from irrelevant alternatives, we can reconstruct choice probabilities for any choice subset from the choice probabilities of the full choice set *without having to rederive from them a probabilistic utility model*. In practice this can also facilitate the aggregation of inferences from observations that span many different choice subsets.

2.1.2 Adding An Alternative To The Choice Set

Independence from irrelevant alternatives also constrains the choice probabilities when we add new alternatives to the choice set. To completely determine the updated choice probabilities, we just need to specify the ratios between the choice probability of the new alternative and

the existing J choice probabilities. While this might sound burdensome, it is a much less demanding task then having to specify all

$$\binom{J+1}{2} = \frac{(J+1)\,J}{2}$$

possible ratios across the expanded choice set.

To illustrate this behavior let's consider an initial choice subset with two alternatives and the choice probability ratio

$$\frac{q_{n1}}{q_{n2}} = r_{12}.$$

Assuming independence from irrelevant alternatives, the choice probabilities after adding a new, third alternative would satisfy the ratios

$$\frac{q'_{n1}}{q'_{n2}} = r_{12} \qquad \frac{q'_{n1}}{q'_{n3}} = r_{13} \qquad \frac{q'_{n2}}{q'_{n3}} = r_{23}.$$

Once we provide r_{13} and r_{23} we can derive all three choice probabilities using calculations similar to those in the previous section.

Starting with the first alternative we have

$$\begin{aligned}
q'_{n1} + q'_{n2} + q'_{n3} &= 1 \\
q'_{n1} + \frac{1}{r_{12}} q'_{n1} + \frac{1}{r_{13}} q'_{n1} &= 1 \\
q'_{n1} \left(1 + \frac{1}{r_{12}} + \frac{1}{r_{13}}\right) &= 1 \\
q'_{n1} &= \frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}}.
\end{aligned}$$

At this point we can isolate the next choice probability,

$$\begin{aligned}
q'_{n1} + q'_{n2} + q'_{n3} &= 1 \\
\frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}} + q'_{n2} + \frac{1}{r_{23}} q'_{n2} &= 1 \\
q'_{n2} \left(1 + \frac{1}{r_{23}}\right) &= 1 - \frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}} \\
q'_{n2} &= \frac{1}{1 + \frac{1}{r_{23}}} \left(1 - \frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}}\right).
\end{aligned}$$

Then we can use the summation constraint to derive the final choice probability,

$$\begin{aligned} q'_{n1} + q'_{n2} + q'_{n3} &= 1 \\ q'_{n3} &= 1 - q'_{n1} - q'_{n2} \\ &= 1 - \frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}} \frac{1}{1 + \frac{1}{r_{23}}} \left(1 - \frac{1}{1 + \frac{1}{r_{23}} + \frac{1}{r_{13}}} \right). \end{aligned}$$

2.1.3 The Blue Bus/Red Bus Problem

While independence from irrelevant alternatives facilitates calculations involving the reduction or expansion of choice subsets, it also implies behaviors that can be less desirable in some applications.

Consider, for example, analyzing discrete choice data for two commuter alternatives, driving a car verses taking a blue bus, with the choice probabilities

$$q_{n1} = \frac{r}{1 + r} \qquad q_{n2} = \frac{1}{1 + r}$$

and the corresponding ratio

$$\frac{q_{n1}}{q_{n2}} = r.$$

At this point we introduce a third alternative: the same bus as before only with a shiny new coat of red paint. If the aesthetics have negligible impact on commuter behavior then the two bus options in the expanded choice set should have the same probability,

$$\frac{q'_{n2}}{q'_{n3}} = 1,$$

and the ratio of choice probabilities between driving a car and taking a new bus should be the same as the ratio between driving a car and taking a blue bus,

$$\frac{q'_{n1}}{q'_{n2}} = \frac{q'_{n1}}{q'_{n3}}.$$

Assuming independence from irrelevant alternatives, the latter two ratios are completely determined by the initial behavior,

$$\frac{q'_{n1}}{q'_{n2}} = \frac{q'_{n1}}{q'_{n3}} = r.$$

The only choice probabilities that satisfy all three of these ratio constraints while also still summing to one are

$$q_{n1} = \frac{r}{r + 2} \qquad q_{n2} = \frac{1}{r + 2} \qquad q_{n3} = \frac{1}{r + 2}.$$

Under independence from irrelevant alternatives, the introduction of an equivalent bus cannibalizes demand not just from the existing bus option but also from the existing car option!

In some applications this might be the desired behavior. For instance this would be reasonable if an additional bus reduced wait times or opened up more seats.

On the other hand, if the two buses offered the exact same commuter experience then the new bus would not offer car commuters anything new to entice them towards public transportation. In this case we would expect the introduction of an equivalent bus option to cannibalize from only the existing bus option and not the existing car option, a behavior that independence from irrelevant alternatives cannot accommodate.

Independence from irrelevant alternatives can also cause problems when we remove one alternative from an initial choice set. Consider, for example, an initial choice set consisting of the driving, blue bus, and red bus commuting alternatives with the choice probabilities

$$q_{n1} = \frac{r}{r+1} \qquad q_{n2} = \frac{1/2}{r+1} \qquad q_{n3} = \frac{1/2}{r+2}$$

and the ratios

$$\frac{q_{n1}}{q_{n2}} = 2\,r \qquad \frac{q_{n1}}{q_{n3}} = 2\,r \qquad \frac{q_{n2}}{q_{n3}} = 1.$$

Under independence from irrelevant alternatives, removing the red bus option results in the choice probabilities

$$q'_{n1} = \frac{2\,r}{2\,r+1}, q'_{n2} = \frac{1}{2\,r+1}.$$

In order to maintain the ratio

$$\frac{q'_{n1}}{q'_{n2}} = \frac{q_{n1}}{q_{n2}} = 2\,r$$

the probability for the red bus has to be reallocated to *both* of the remaining commuting alternatives, with some red bus riders choosing to start driving instead of taking the blue bus.

This might make sense if the removal of the red bus degrades the experience of taking the blue bus. If, on the other hand, the experience of taking the blue bus is unchanged then we would expect that red bus riders would all move to the equivalent blue bus alternative. In this case we would expect to see

$$q'_{n2} = q_{n2} + q_{n3} = \frac{1}{r+1}$$

with

$$q'_{n1} = 1 - q'_{n2} = \frac{r}{r+1}.$$

This implies the new choice probability ratio

$$\frac{q'_{n1}}{q'_{n2}} = r \neq \frac{q_{n1}}{q_{n2}},$$

which violates independence from irrelevant alternatives!

Ultimately, models that exhibit independence from irrelevant alternatives perform poorly when certain pairs of alternatives are better substitutes for each other than other alternatives. One way to avoid these awkward behaviors in practice is to exclude any alternatives that are not sufficiently distinct from the other alternatives. Another is to consider models that don't necessarily exhibit independence from irrelevant alternatives.

2.2 Sensitivities and Elasticities

Another way to quantify substitution patterns is to examine infinitesimal changes to the choice probabilities that arise from infinitesimal changes to the probabilistic utility model.

To formalize this we'll need to consider a discrete choice model defined by a parametric utility model configured with a real-valued variable z. This variable might, for example, represent some unknown model configuration or some external covariate that is known exactly.

In this case the probabilistic utility model is defined by an entire family of joint probability density functions,

$$p(u_{n1}, \ldots, u_{nJ} \mid z),$$

or, equivalently, a family of joint cumulative distribution functions,

$$\Pi(u_{n1}, \ldots, u_{nJ} \mid z).$$

This families then give parametric choice probabilities,

$$q_{n1}(z), \ldots q_{nJ}(z).$$

If this dependence is smooth then we can quantify how much each choice probability varies with infinitesimal changes to z with the derivatives

$$\frac{\partial q_{nj}}{\partial z}(z).$$

Because they quantify how much each choice probability is influenced by changes in z, these derivatives are also known as **sensitivities**.

One immediate complication with interpreting sensitivities, however, is that they are always coupled together. Indeed if we differentiate the summation constraint then we see that the

sensitivities across all alternatives in a choice set must always sum to zero,

$$\begin{aligned}
\sum_{j=1}^{J} q_{nj} &= 1 \\
\frac{\partial}{\partial z} \left(\sum_{j=1}^{J} q_{nj} \right) &= \frac{\partial}{\partial z} \left(1 \right) \\
\sum_{j=1}^{J} \frac{\partial q_{nj}}{\partial z}(z) &= 0.
\end{aligned}$$

As we saw in the previous section, ratios of two choice probabilities are often easier to interpret. The influence of z on the ratio of any two choice probabilities is given by

$$\begin{aligned}
\frac{\partial}{\partial z} \left(\frac{q_{nj}}{q_{nj'}} \right) &= \frac{1}{q_{nj'}} \frac{\partial q_{nj}}{\partial z}(z) - \frac{q_{nj}}{q_{nj'}^2} \frac{\partial q_{nj'}}{\partial z}(z) \\
&= \frac{z}{z} \frac{q_{nj}}{q_{nj}} \frac{1}{q_{nj'}} \frac{\partial q_{nj}}{\partial z}(z) - \frac{z}{z} \frac{q_{nj}}{q_{nj'}^2} \frac{\partial q_{nj'}}{\partial z}(z) \\
&= \frac{1}{z} \frac{q_{nj}}{q_{nj'}} \left[\frac{z}{q_{nj}} \frac{\partial q_{nj}}{\partial z}(z) - \frac{z}{q_{nj'}} \frac{\partial q_{nj'}}{\partial z}(z) \right].
\end{aligned}$$

The normalized sensitivities that arise in this calculation are known as **elasticities**,

$$\epsilon_{nj}(z) = \frac{z}{q_{nj}} \frac{\partial q_{nj}}{\partial z}(z).$$

Besides being independent of the choice of units for z, elasticities immediately identify invariances of both choice probabilities and ratios of choice probabilities.

For example, if $z \neq 0$ then a vanishing elasticity implies a vanishing sensitivity,

$$\begin{aligned}
0 &= \epsilon_{nj}(z) \\
0 &= \frac{z}{q_{nj}} \frac{\partial q_{nj}}{\partial z}(z) \\
&= \frac{\partial q_{nj}}{\partial z}(z).
\end{aligned}$$

This, in turn, implies that the choice probability q_{nj} is unaffected by changes in z.

On the other hand, if $z \neq 0$ and two elasticities are the same then the corresponding ratio of

choice probabilities is independent of z,

$$\begin{aligned}
\epsilon_{nj}(z) &= \epsilon_{nj'}(z) \\
0 &= \epsilon_{nj}(z) - \epsilon_{nj'}(z) \\
0 &= \frac{1}{z}\frac{q_{nj}}{q_{nj'}}\left[\epsilon_{nj}(z) - \epsilon_{nj'}(z)\right] \\
0 &= \frac{1}{z}\frac{q_{nj}}{q_{nj'}}\left[\frac{z}{q_{nj}}\frac{\partial q_{nj}}{\partial z}(z) - \frac{z}{q_{nj'}}\frac{\partial q_{nj'}}{\partial z}(z)\right] \\
0 &= \frac{\partial}{\partial z}\left(\frac{q_{nj}}{q_{nj'}}\right).
\end{aligned}$$

In particular we can use elasticities to identify discrete choice models that exhibit independence from irrelevant alternatives. A discrete choice model will exhibit this property if and only if the elasticities for all of the alternatives in the choice set are the same,

$$\epsilon_{nj}(z) = \epsilon_{nj'}(z)$$

for all possible values of z.

2.3 Perfect Replacements

In some applications the most relevant feature of a discrete choice model is how the behavior of alternatives can change *without* affecting specific choice probabilities. In other words, we might be interested in what hypothetical alternatives define **perfect replacements** for a given alternative.

Consider, for example, a probabilistic utility model with not one but two configuration variables

$$p(u_{n1}, \ldots, u_{nJ} \mid z_1, z_2),$$

or, equivalently,

$$\Pi(u_{n1}, \ldots, u_{nJ} \mid z_1, z_2).$$

The infinitesimal change in z_1 needed to keep a choice probability q_{nj} constant under an infinitesimal change in z_2 is determined by the total derivative

$$\frac{\mathrm{d}z_1}{\mathrm{d}z_2}$$

along the surface implicitly defined by the constraint

$$q_{nj}(z_1, z_2) = \text{constant}.$$

We can compute this constrained derivative by differentiating the constraint with respect to a parameter t that coordinates motion along the constraint surface,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(q_{nj}(z_1(t), z_2(t))) &= \frac{\mathrm{d}}{\mathrm{d}t}\,(\text{const}) \\
\frac{\partial q_{nj}}{\partial z_1}\,\frac{\mathrm{d}z_1}{\mathrm{d}t} + \frac{\partial q_{nj}}{\partial z_2}\,\frac{\mathrm{d}z_2}{\mathrm{d}t} &= 0 \\
\frac{\partial q_{nj}}{\partial z_1}\,\frac{\mathrm{d}z_1}{\mathrm{d}t} &= -\frac{\partial q_{nj}}{\partial z_2}\,\frac{\mathrm{d}z_2}{\mathrm{d}t} \\
\frac{\mathrm{d}z_1}{\mathrm{d}t}\left(\frac{\mathrm{d}z_2}{\mathrm{d}t}\right)^{-1} &= -\frac{\partial q_{nj}}{\partial z_2}\,\left(\frac{\partial q_{nj}}{\partial z_1}\right)^{-1} \\
\frac{\mathrm{d}z_1}{\mathrm{d}z_2} &= -\frac{\partial q_{nj}}{\partial z_2}\,\left(\frac{\partial q_{nj}}{\partial z_1}\right)^{-1}.
\end{aligned}$$

For example, if both partial derivatives are constant,

$$\begin{aligned}
\frac{\partial q_{nj}}{\partial z_1} &= \gamma_1 \\
\frac{\partial q_{nj}}{\partial z_2} &= \gamma_2,
\end{aligned}$$

then so too is the total derivative,

$$\frac{\mathrm{d}z_1}{\mathrm{d}z_2} = -\frac{\gamma_2}{\gamma_1}.$$

In this case the utility model configurations

$$(z_1, z_2)$$

and

$$\left(z_1 + \delta, z_2 - \frac{\gamma_2}{\gamma_1}\,\delta\right)$$

result in exactly the same choice probability for the jth alternative, q_{nj}.

Intuitively we can interpret these two model configurations as defining two distinct jth alternatives that are equivalent from the perspective of the agents.

2.4 Expected Substitution Patterns

When a probabilistic utility model is configured by unknown model configuration variables,

$$p(u_{n1}, \ldots, u_{nJ} \mid \theta)$$

or equivalently

$$\Pi(u_{n1}, \ldots, u_{nJ} \mid \theta),$$

then the resulting choice probabilities, and substitution patterns, will also be unknown. In practice we need to infer the consistent behaviors of θ from observed data and then propagate the inferential uncertainty to the derived choice behaviors.

Bayesian inference uses the observations

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

to construct a posterior distribution for θ,

$$\begin{aligned} p(\theta \mid \tilde{y}_1, \ldots, \tilde{y}_M) &\propto p(\tilde{y}_1, \ldots, \tilde{y}_M \mid \theta)\, p(\theta) \\ &\propto \left[\prod_{m=1}^{M} \text{categorical}(y_m \mid q_{n(m)1}(\theta), \ldots, q_{n(m)J}(\theta)) \right] p(\theta). \end{aligned}$$

This posterior distribution for θ can then be pushed forward to an induced posterior distribution for the choice probabilities,

$$p(q_{n1}, \ldots, q_{nJ} \mid \tilde{y}_1, \ldots, \tilde{y}_M),$$

quantifying our uncertainty about agent behaviors.

We can use this induced posterior distribution to study the range of substitution patterns consistent with the observed data. For example in practice we might use posterior samples of the model configuration variables,

$$\tilde{\theta}_s \sim p(\theta \mid \tilde{y}_1, \ldots, \tilde{y}_M),$$

to derive posterior choice samples,

$$\tilde{q}_{n1,s}, \ldots, \tilde{q}_{nJ,s} = q_{n1}(\tilde{\theta}_s), \ldots, q_{nJ}(\tilde{\theta}_s),$$

and then investigate the varying substitution patterns across the samples.

That said, we can also average over the consistent model configurations to construct point estimates of the choice probabilities. More formally we might consider the posterior expectation value of each choice probability,

$$\mathbb{E}[q_{nj}] = \int \mathrm{d}q_{nj}\, p(q_{n1}, \ldots, q_{nJ} \mid \tilde{y}_1, \ldots, \tilde{y}_M)\, q_{nj}.$$

Interestingly the expected choice probabilities define a valid simplex,

$$\begin{aligned} \sum_{j=1}^{J} \mathbb{E}\left[q_{nj} \right] &= \mathbb{E}\left[\sum_{j=1}^{J} q_{nj} \right] \\ &= \mathbb{E}[1] \\ &= 1, \end{aligned}$$

which allows them to define *expected substitution patterns*.

Expected choice probabilities, however, will in general exhibit *different* substitution patterns than the substitution patterns from any individual choice probability configuration. For example, even if every value of θ results in choice probabilities that exhibit independence from irrelevant alternatives, the posterior expected choice probabilities typically will not. That said, the deviations between these behaviors will not always be substantial.

Ultimately the most productive use of posterior inferences is to first derive whatever behavior might be of interest, such as ratios of choice probabilities, and only then propagate posterior uncertainties. Using posterior expectations of individual model configuration variables to derive behaviors of interest not only loses information but also is not guaranteed to preserve all of the nuances in the resulting behaviors.

2.5 Inferring Substitution Patterns

In general different configurations of a parametric utility models will exhibit different substitution patterns. This flexibility can be useful when we don't know what substitution patterns are appropriate in a given application. To really leverage that flexibility, however, we need to be able to infer the appropriate substitution patterns from observed data.

Unfortunately inferring substitution patterns is not always straightforward. In particular, we cannot infer substitution patterns from observations drawn from a single choice set. The only way to discriminate between different substitution patterns is to have observations drawn from multiple choice subsets.

The more choice subsets we observe, the more precisely we will be able to infer the substitution patterns. Exactly how many choice subsets we need to observe for inferences to be well-behaved depends on the flexibility of the specific discrete choice model we are using.

Bayesian analysis of discrete choice data is particularly useful. Once we quantify these uncertainties, no matter how large or complex they might be, we can propagate them into inferences and predictions for decision behavior in new choice subsets.

3 Independent Utility Models

The construction of a discrete choice model is greatly simplified if we assume that the utilities for each alternative are independent of each other,

$$p(u_{n1}, \ldots, u_{nJ}) = \prod_{j=1}^{J} p_j(u_{nj}).$$

Beyond just independence, we might also assume that the individual utility models are also the same,

$$p(u_{n1}, \ldots, u_{nJ}) = \prod_{j=1}^{J} p(u_{nj}).$$

In this case the joint model is completely determined by a single, univariate utility model with potentially varying configurations.

Under these assumptions the joint cumulative distribution function also simplifies into a product of univariate cumulative distribution functions,

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \prod_{j=1}^{J} \Pi(u_{nj}).$$

The better behaved $\Pi(x)$ is under products, the easier $\Pi(u_{n1}, \ldots, u_{nJ})$ will be to differentiate, and then integrate, as we derive choice probabilities.

One way to motivate a useful component utility model is to consider *extremes*. Let's say that the utilities that ultimately drive agent choice are largely determined by the best case behavior over a wide range of possibilities. In this case the model for each $p(u_{nj})$ would be derived from the *maximum* utility considered.

The behavior of the maximum of an ensemble of probabilistic behaviors is the subject of **extreme value theory** in statistics (de Carvalho et al. 2026). One of the key insights from extreme value theory is the Fisher-Tippett-Gnedenko extreme value theorem, which states that over a sufficiently large ensemble only a few coherent extreme behaviors can arise. All of these asymptotic behaviors are quantified with a corresponding family of probability density functions.

Unfortunately, most of the models motivated by extreme value theory are sufficiently complex that we cannot derive choice probabilities in closed form; for some details see Appendix A.2. Fortunately, there is one exceptional subset of extreme value theory models that do allow for convenient analytic results: the Gumbel model.

3.1 The Independent Gumbel Model

One of the emergent behaviors given by the extreme value theorem is defined by the Gumbel family of probability density functions over a real line,

$$\text{Gumbel}(x \mid \mu, \sigma) = \frac{1}{\sigma} \exp\left(-\frac{x - \mu}{\sigma} - e^{-\frac{x-\mu}{\sigma}}\right).$$

Each probability density function in the Gumbel family is unimodal, with the location parameter μ roughly characterizing the location of the peak and the scale parameter σ roughly characterizing the width of the peak (Figure 1).

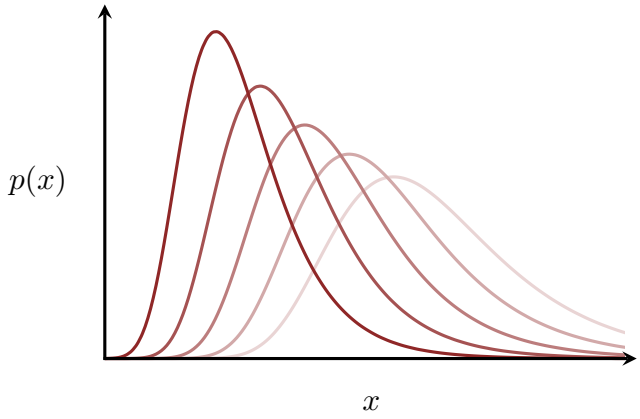


Figure 1: Each probability density function in the Gumbel family is unimodal and skewed towards larger values. The location of the peak and overall width vary across the family.

One nice feature of the Gumbel family is that the cumulative distribution functions are particularly well-behaved,

$$\Pi_{\text{Gumbel}}(x \mid \mu, \sigma) = \exp\left(-e^{-\frac{x' - \mu}{\sigma}}\right).$$

For a derivation of these cumulative distribution functions see Appendix A.1.

If we model the utilities that an agent assigns to each alternative with independent Gumbel models configured with separate location parameters and a common scale parameter,

$$\theta = (\mu_{n1}, \ldots, \mu_{nJ}, \sigma_n),$$

then the joint utility model becomes

$$p(u_{n1}, \ldots, u_{nJ} \mid \theta) = \prod_{j=1}^{J} \text{Gumbel}(u_{nj} \mid \mu_{nj}, \sigma_n).$$

I will refer to each μ_{nj} as the **baseline utility** for each alternative.

In this case the joint cumulative distribution function simplifies into an exponential of the sum

of contributions from each alternative,

$$\begin{aligned}\Pi(u_{n1}, \ldots, u_{nJ} \mid \theta) &= \prod_{j=1}^{J} \Pi_{\text{Gumbel}}(u_{nj} \mid \mu_{nj}, \sigma_n), \\ &= \prod_{j=1}^{J} \exp\left(-e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}}\right) \\ &= \exp\left(-\sum_{j=1}^{J} e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}}\right).\end{aligned}$$

Fortunately, all of the derivatives and integrals needed to derive the choice probabilities from this model can be performed analytically. I will sequester the actual calculations to Appendix A.1 referring here to only the results:

$$\begin{aligned}q_{nj}(\theta) &= \int_{-\infty}^{+\infty} \mathrm{d}u \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u \mid \theta) \\ &= \frac{\exp\left(\frac{\mu_{nj}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)}.\end{aligned}$$

The larger σ_n is, the more each μ_{nj}/σ_n will shrink towards zero. This, in turn, pushes the choice probabilities towards more uniform configurations. On the other hand, smaller values of σ_n amplify the baseline utilities, pushing the choice probabilities towards more heterogeneous configurations.

Conveniently these choice probabilities can be readily calculated with the **softmax function**,

$$\text{softmax}(x_1, \ldots, x_J) = \left(\frac{e^{x_1}}{\sum_{j'=1}^{J} e^{x_{j'}}}, \ldots, \frac{e^{x_J}}{\sum_{j'=1}^{J} e^{x_{j'}}}\right)$$

All we have to do is apply the softmax function to the scaled baseline utilities,

$$(q_{n1}, \ldots, q_{nJ}) = \text{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n}\right)!$$

If we introduce an explicit variable for the direct inputs to the softmax function,

$$\nu_{nj} = \frac{\mu_{nj}}{\sigma_n},$$

then we can write the choice probabilities in the even more compact form,

$$(q_{n1}, \ldots, q_{nJ}) = \text{softmax}\left(\nu_{n1}, \ldots, \nu_{nJ}\right).$$

The softmax function is also known as the *inverse multilogit function*, with the multilogit of the choice probabilities being given by the scaled baseline utilities. Moreover, the multilogit function is often shorted to just the logit function, not to be confused with the less general, one-dimensional logit function

$$\text{logit}(p) = \log \frac{p}{1 - p}.$$

Because of this sequence of terminological jumps, the independent Gumbel discrete choice model is often referred to as just the **logit model**. Personally I find this confusing, not only because of the sloppiness with the multilogit and logit functions but also because it focuses on a derived behavior and not the initial assumptions that we actually make. In this chapter I will stick with "independent Gumbel model", but beware when comparing to the more conventional literature in some fields.

3.2 Redundancies of the Independent Gumbel Model

With the softmax function, the independent Gumbel discrete choice model is particularly straightforward to implement in practice. At the same time the structure of the softmax function also provides insight into the inherit redundancies of this model.

3.2.1 Translation Redundancy

As I discuss in my general Taylor modeling chapter and die fairness case study, the softmax function is not injective: many distinct inputs map to the same output probabilities. Specifically, any *translation* of the inputs leaves the output unchanged.

From a discrete choice perspective, any translation of the scaled baseline utilities does not affect the derived choice probabilities,

$$\begin{aligned}(q_{n1}, \ldots, q_{nJ}) &= \text{softmax}\left(\nu_{n1}, \ldots, \nu_{nJ}\right) \\ &= \text{softmax}\left(\nu_{n1} + \alpha, \ldots, \nu_{nJ} + \alpha\right).\end{aligned}$$

An immediate consequence of this redundancy is that, no matter how precisely we infer the choice probabilities, we will only ever be able to learn the baseline utilities up to translations. This results in problematic likelihood functions that concentrate not around a point in the model configuration space but rather entire planes.

In order to avoid these inferential degeneracies we need to eliminate this redundancy from the model. This requires some way of obstructing translations of the baseline utilities.

One particularly convenient way to accomplish this is to parameterize the model in terms of *relative baseline utilities*,

$$\delta_{nj} = \mu_{nj} - \mu_{nj'}$$

for any distinguished alternative j'. If $\delta_{nj} > 0$ then the baseline utility of the jth alternative is larger than the baseline utility of the distinguished j'th alternative, and vice versa. Critically, the relative baseline utility of the distinguished alternative is fixed to zero,

$$\delta_{nj'} = \mu_{nj'} - \mu_{nj'} = 0,$$

eliminating a degree of freedom from the model.

Because of the translation invariance of the softmax function, we can always write the choice probabilities directly in terms of these relative baseline utilities:

$$\begin{aligned}
(q_{n1}, \ldots, q_{nJ}) &= \text{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n}\right) \\
&= \text{softmax}\left(\frac{\mu_{n1}}{\sigma_n} - \frac{\mu_{nj'}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n} - \frac{\mu_{nj'}}{\sigma_n}\right) \\
&= \text{softmax}\left(\frac{\mu_{n1} - \mu_{nj'}}{\sigma_n}, \ldots, \frac{\mu_{nJ} - \mu_{nj'}}{\sigma_n}\right) \\
&= \text{softmax}\left(\frac{\delta_{n1}}{\sigma_n}, \ldots, \frac{\delta_{nJ}}{\sigma_n}\right).
\end{aligned}$$

We cannot translate the relative inputs, however, without moving

More formally, fixing one of the inputs to zero makes the softmax function bijective, with each configuration of the scaled, relative baseline utilities corresponding to a unique configuration of the output choice probabilities.

3.2.2 Scale Redundancy

Although we have resolved the inherent redundancy of the softmax function, we are not quite out of the woods yet. The *inputs* to the softmax function are themselves redundant.

The inputs to the softmax function are all *ratios* of baseline utilities μ_{nj} and the agent scale σ_n,

$$\nu_{nj} = \frac{\mu_{nj}}{\sigma_n}.$$

Scaling the baseline utilities and the agent scale in the same way doesn't change these ratios,

$$\begin{aligned}
\nu'_{nj} &= \frac{\mu'_{nj}}{\sigma'_n} \\
&= \frac{\alpha\,\mu_{nj}}{\alpha\,\sigma_n} \\
&= \frac{\mu_{nj}}{\sigma_n} \\
&= \nu_{nj}.
\end{aligned}$$

With the inputs to the softmax function unchanged,the derived choice probabilities are the same,

$$\begin{aligned}
(q'_{n1}, \ldots, q'_{nJ}) &= \text{softmax}\left(\frac{\mu'_{n1}}{\sigma'_n}, \ldots, \frac{\mu'_{nJ}}{\sigma'_n}\right) \\
&= \text{softmax}\left(\frac{\alpha\,\mu_{n1}}{\alpha\,\sigma_n}, \ldots, \frac{\alpha\,\mu_{nJ}}{\alpha\,\sigma_n}\right) \\
&= \text{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n}\right) \\
&= (q_{n1}, \ldots, q_{nJ}).
\end{aligned}$$

Consequently inferences of the choice probabilities cannot discriminate between μ_{nj} and σ_n.

If we're considering only a single agent, then we can eliminate this scale redundancy by directly parameterizing the model in terms of scaled baseline utilities

$$\eta_{nj} = \frac{\mu_{nj}}{\sigma_n}.$$

When we want to analyze choice observations across multiple agents, however, we have to account for the fact that σ_n can vary from one agent to another.

In this case we have to distinguish a particular agent scale, $\sigma_{n'}$, and use it to define scaled baseline utilities

$$\zeta_{nj} = \frac{\mu_{nj}}{\sigma_{n'}},$$

and relative scales,

$$\tau_n = \frac{\sigma_n}{\sigma_{n'}}.$$

By construction the relative scale of the distinguished agent is fixed to one,

$$\tau_{n'} = 1,$$

eliminating a degree of freedom from the model.

We can now write the choice probabilities for any particular agent as

$$\begin{aligned}
(q_{n1}, \ldots, q_{nJ}) &= \text{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n}\right) \\
&= \text{softmax}\left(\frac{\mu_{n1}/\sigma_{n'}}{\sigma_n/\sigma_{n'}}, \ldots, \frac{\mu_{nJ}/\sigma_{n'}}{\sigma_n/\sigma_{n'}}\right) \\
&= \text{softmax}\left(\frac{\zeta_{n1}}{\tau_n}, \ldots, \frac{\zeta_{nJ}}{\tau_n}\right)
\end{aligned}$$

3.2.3 Resolving Both Redundancies Without Abusing Notation

To completely eliminate redundancies from the independent Gumbel discrete choice model we need to integrate both of these strategies at the same time.

Given a distinguished alternative j' and agent n' we can parameterize the model in terms of scaled, relative baseline utilities

$$\omega_{nj} = \frac{\delta_{nj}}{\sigma_{n'}}$$

and relative scales

$$\tau_n = \frac{\sigma_n}{\sigma_{n'}}.$$

The choice probabilities are then given by

$$(q'_{n1}, \ldots, q'_{nJ}) = \text{softmax}\left(\frac{\omega_{n1}}{\tau_n}, \ldots, \frac{\omega_{nJ}}{\tau_n}\right)$$

Which alternative and agent we distinguish does not affect the behavior or implementation of the choice probabilities. It does, however, change the interpretation and inferential behavior of the model configuration variables we use.

The change in interpretation is especially important in Bayesian analyses, as it changes the form of the prior model. For example, an independent prior model over the baseline utilities does not correspond to an independent prior model over the scaled, relative baseline utilities and vice versa. In practice eliciting a prior model for the relative model configuration variables is more straightforward because we can contextualize our domain expertise through explicit comparisons to the reference alternative and agent.

Many presentations of discrete choice models, especially those that limit inferences to point estimates, overload variable names and use μ_{nj} and σ_n to refer to *both* the absolute and relative model configuration variables at *at the same time*. Redundancies are managed by fixing one μ_{nj} to 0 and one σ_n to 1, while largely ignoring any change in interpretation.

To avoid any ambiguity, and clarify the robust implementation of discrete choice models, I will be pedantic as possible with my notation in this chapter. In particular, I will use as many greek letters as are needed to ensure that different transformations of the absolute model configurations are always denoted with unique symbols. For the independent Gumbel model I will largely use the absolute variables μ_{nj} and σ_n when discussing theoretical properties and the relative variables ω_{nj} and τ_n when discussing implementations.

3.3 Substitution Patterns of the Independent Gumbel Model

The analytic form of the choice probabilities in the independent Gumbel model makes the substitution patterns particularly straightforward to investigate.

Because the normalization that couples all of the choice probabilities together cancels, ratios of choice probabilities simplify substantially,

$$\begin{aligned}
\frac{q_{nj_1}}{q_{nj_2}} &= \frac{\exp\left(\frac{\mu_{nj_1}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \frac{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)}{\exp\left(\frac{\mu_{nj_2}}{\sigma_n}\right)} \\
&= \frac{\exp\left(\frac{\mu_{nj_1}}{\sigma_n}\right)}{\exp\left(\frac{\mu_{nj_2}}{\sigma_n}\right)} \\
&= \exp\left(\frac{\mu_{nj_1}}{\sigma_n} - \frac{\mu_{nj_2}}{\sigma_n}\right).
\end{aligned}$$

Regardless of the two alternatives we consider, this ratio will only ever depend on the two corresponding baseline utilities. In other words the ratio is independent of changes to any of the other baseline utilities. Decreasing μ_{nj_3} may increase both q_{nj_1} and q_{nj_2}, but in exactly the same proportion so that the ratio between them remains the same.

In other words the independent Gumbel model exhibits independence from irrelevant alternatives, and all of the consequences of that property that we discussed in Section 2.1. This makes the independent Gumbel model relatively exceptional, as most discrete choice models do not exhibit independence from irrelevant alternatives.

We can also demonstrate independence from irrelevant alternatives using elasticities. The sensitivity of the j_1th choice probability with respect to the j_3th baseline utility is given by

$$\begin{aligned}
\frac{\partial q_{nj_1}}{\partial \mu_{nj_3}} &= \frac{\partial}{\partial \mu_{nj_3}} \left[\frac{\exp\left(\frac{\mu_{nj_1}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \right] \\
&= \exp\left(\frac{\mu_{nj_1}}{\sigma_n}\right) \frac{-\frac{1}{\sigma_n} \exp\left(\frac{\mu_{nj_3}}{\sigma_n}\right)}{\left(\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)\right)^2} \\
&= -\frac{1}{\sigma_n} \frac{\exp\left(\frac{\mu_{nj_1}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \frac{\exp\left(\frac{\mu_{nj_3}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \\
&= -\frac{1}{\sigma_n} q_{nj_1} q_{nj_3}.
\end{aligned}$$

The corresponding elasticity is then

$$\begin{aligned}\frac{\mu_{nj_3}}{q_{nj_1}} \frac{\partial q_{nj_1}}{\partial \mu_{nj_3}} &= \frac{\mu_{nj_3}}{q_{nj_1}} \left(-\frac{1}{\sigma_n} \, q_{nj_1} \, q_{nj_3} \right) \\ &= -\frac{\mu_{nj_3}}{\sigma_n} \, q_{nj_3}.\end{aligned}$$

Notice that this elasticity is the same for all alternatives j_1!

Because the elasticities are all the same, differences in elasticities cancel, and ratios of choice probabilities are invariant,

$$\begin{aligned}\frac{\partial}{\partial \mu_{nj_3}} \left(\frac{q_{nj_1}}{q_{nj_2}} \right) &= \frac{1}{\mu_{nj_3}} \frac{q_{nj_1}}{q_{nj_2}} \left[-\frac{\mu_{nj_3}}{\sigma_n} \, q_{nj_3} + \frac{\mu_{nj_3}}{\sigma_n} \, q_{nj_3} \right] \\ &= \frac{1}{\mu_{nj_3}} \frac{q_{nj_1}}{q_{nj_3}} \left[\qquad 0 \qquad \right] \\ &= 0\end{aligned}$$

for any distinct j_1, j_2, and j_3. This is consistent with the invariance of ratios of choice probabilities that we identified above from direct inspection.

3.4 Demonstration

Having worked through all of the theory, let's now see how we can implement the independent Gumbel model into a practical Bayesian analysis.

3.4.1 Setup

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

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

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)
```

3.4.2 Basics

Let's start with the basic implementation of the independent Gumbel model.

3.4.2.1 Simulate Data

First and foremost, we'll need some data to fit. Here let's consider many observations, each of which pair one of six agents with a fixed choice set of nine alternatives.

```
N <- 6
J <- 9
M <- 1000
```

Stan
Program 1 `simu\_ig1.stan`

```
data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set
}

generated quantities {
  vector[J] mu;                 // Baseline utilities
  array[N] real<lower=0> sigma; // Scales

  array[M] int<lower=1, upper=N> agent;  // Observed agent
  array[M] int<lower=1, upper=J> choice; // Observed choice

  // Sample true data generating process behavior
  for (j in 1:J) mu[j] = normal_rng(0, 1);
  for (n in 1:N) sigma[n] = gamma_rng(31.0, 15.5);

  // Simulate observations
  for (m in 1:M) {
    int n = categorical_rng(uniform_simplex(N));
    agent[m] = n;
    choice[m] = categorical_logit_rng(log_softmax(mu / sigma[n]));
  }
}
```

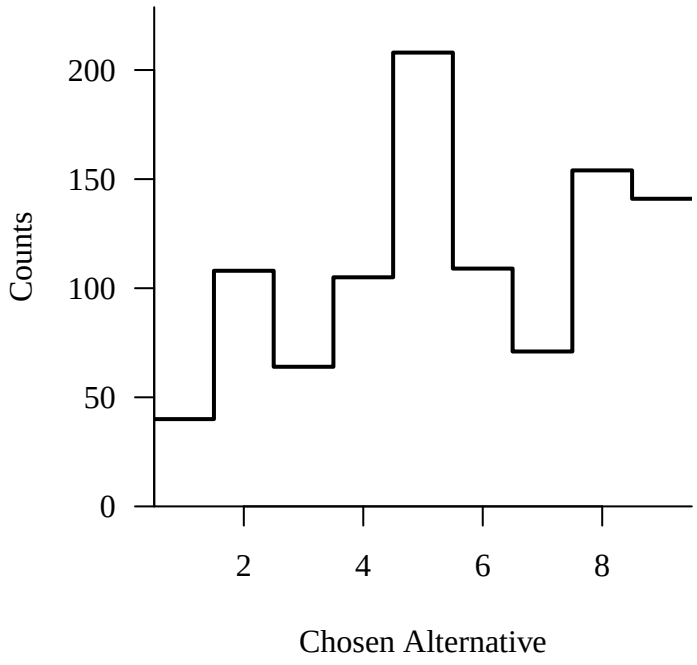
```
simu <- stan(file="stan_programs/simu_ig1.stan",
             algorithm="Fixed_param", seed=8438338,
             data=list('N' = N, 'J' = J, 'M' = M),
             warmup=0, iter=1, chains=1, refresh=0)

simu_fit <- extract(simu)
```

```
data <- list('N' = N, 'J' = J, 'M' = M,
             'agent' = simu_fit$agent[1,],
             'choice' = simu_fit$choice[1,])

mu_true <- simu_fit$mu[1,]
sigma_true <- simu_fit$sigma[1,]
```

3.4.2.2 Explore Data

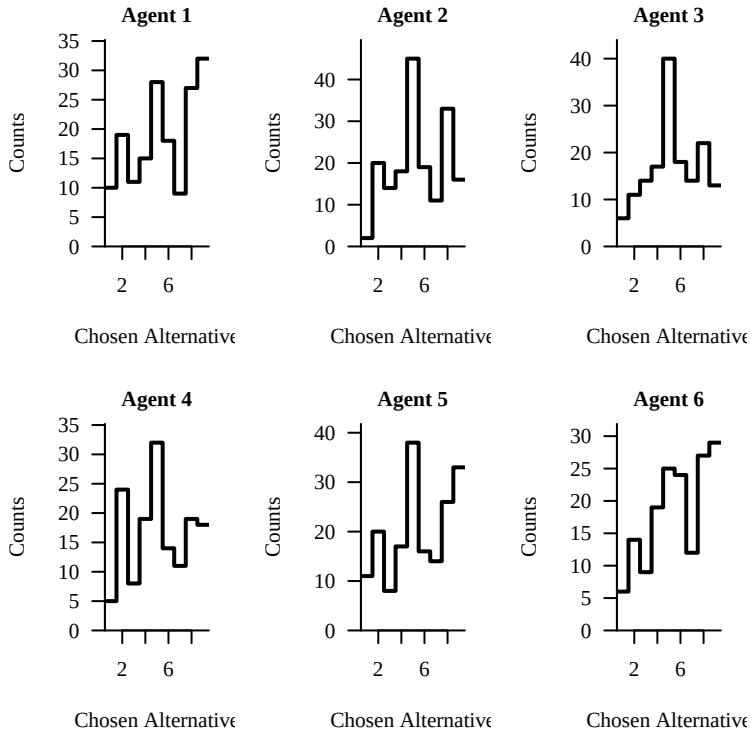
There are a few natural ways that we can summarize observations over a fixed choice set. For example, we can aggregate all of the observed choices into a single histogram.

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

util$plot_line_hist(data$choice, 0.5, data$J + 0.5, 1,
                    xlab='Chosen Alternative')
```



We can also stratify by agents, histogramming just the observed choices made by one agent at a time.

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

for (n in 1:data$N)
  util$plot_line_hist(data$choice[data$agent == n],
                      0.5, data$J + 0.5, 1,
                      xlab='Chosen Alternative',
                      main=paste('Agent', n))
```



3.4.2.3 Redundant Implementation

With these simulated data we can now draw inferences from an independent Gumbel discrete choice model. As we discussed in Section 3.2, implementing this model with the absolute baseline utilities and agent scales should lead to problems.

The prior model here is completely arbitrary.

```
fit <- stan(file='stan_programs/ig1.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Interestingly Hamiltonian Monte Carlo doesn't seem to have any trouble exploring the posterior distribution over the absolute model configurations.

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

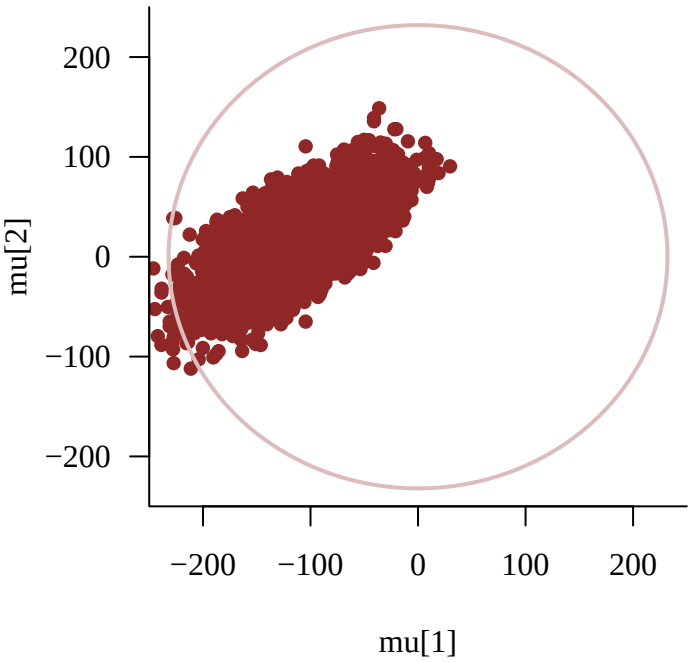
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('mu', 'sigma'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

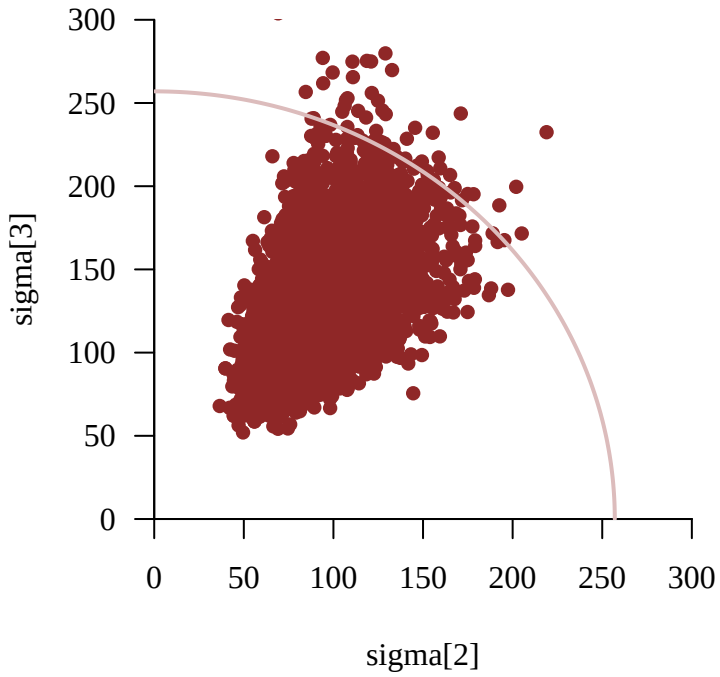
That said, we don't actually learn much about each absolute baseline utilities. The arbitrary prior model is doing a lot of work here to keep our inferences contained to a finite neighborhood.

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

plot(samples[['mu[1]']], samples[['mu[2]']],
     cex=1, pch=16, col=util$c_dark,
     xlim=c(-250, 250), xlab='mu[1]',
     ylim=c(-250, 250), ylab='mu[2]')

thetas <- seq(0, 2 * pi, 0.01)
lines(232 * cos(thetas), 232 * sin(thetas),
      lwd=2, col=util$c_light)
```



Similarly, our inferences for the agent scales are largely informed by the prior model.

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

plot(samples[['sigma[2]']], samples[['sigma[3]']],
     cex=1, pch=16, col=util$c_dark,
     xlim=c(0, 300), xlab='sigma[2]',
     ylim=c(0, 300), ylab='sigma[3]')

thetas <- seq(0, 0.5 * pi, 0.01)
lines(257 * cos(thetas), 257 * sin(thetas),
      lwd=2, col=util$c_light)
```



3.4.2.4 Non-Redundant Implementation

We should be able to do better by implementing the independent Gumbel model with scaled, relative baseline utilities and relative agent scales. This, however, requires distinguishing an anchor alternative and anchor agent.

In theory the choice of anchors should not matter, but in practice the more we observe the anchors the more well-behaved our inferences for the relative model configurations will be. This, in turn, allows our posterior computation to be as fast as possible.

Because the observations are relatively uniform across agents, I will anchor the first agent for simplicity.

```
table(data$agent)
```

```
  1   2   3   4   5   6
169 178 155 150 183 165
```

The observed choices, however, are much more heterogeneous. Here I will anchor the alternative that has been chosen the most often.

```
table(data$choice)
```

```
  1   2   3   4   5   6   7   8   9
 40 108  64 105 208 109  71 154 141
```

```
data$anchor <- 5
```

To avoid redundancy, we directly model only the `J - 1` unanchored alternatives. The `add_anchor` function inserts the missing zero back into the relative baseline utilities.

Note that the prior model for the relative model configurations is once again arbitrary, but it is *not* consistent with the prior model for the absolute model configurations that we used previously. In practical applications it is often more straightforward to elicit domain expertise about the relative model configurations directly.

```
fit <- stan(file='stan_programs/ig2.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

None of the diagnostics suggest inaccurate posterior quantification.

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

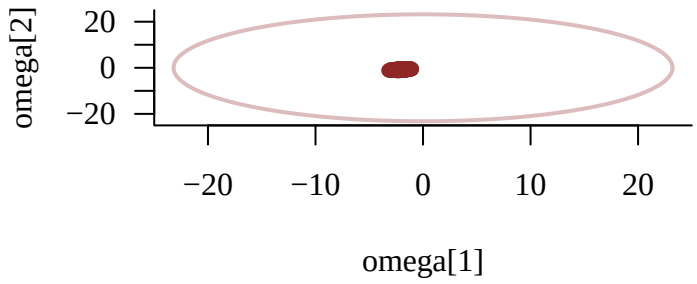
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

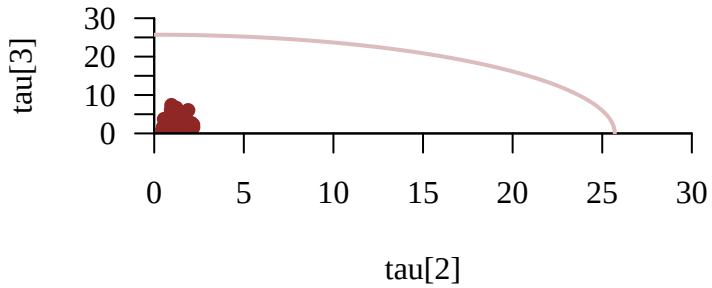
```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

Free of redundancies, the likelihood function for the relative model configurations strongly informs our posterior inferences.

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

plot(samples[['omega[1]']], samples[['omega[2]']],
     cex=1, pch=16, col=util$c_dark,
     xlim=c(-25, 25), xlab='omega[1]',
     ylim=c(-25, 25), ylab='omega[2]')

thetas <- seq(0, 2 * pi, 0.01)
lines(23.2 * cos(thetas), 23.2 * sin(thetas),
      lwd=2, col=util$c_light)

plot(samples[['tau[2]']], samples[['tau[3]']],
     cex=1, pch=16, col=util$c_dark,
     xlim=c(0, 30), xlab='tau[2]',
     ylim=c(0, 30), ylab='tau[3]')

thetas <- seq(0, 0.5 * pi, 0.01)
lines(25.7 * cos(thetas), 25.7 * sin(thetas),
      lwd=2, col=util$c_light)
```



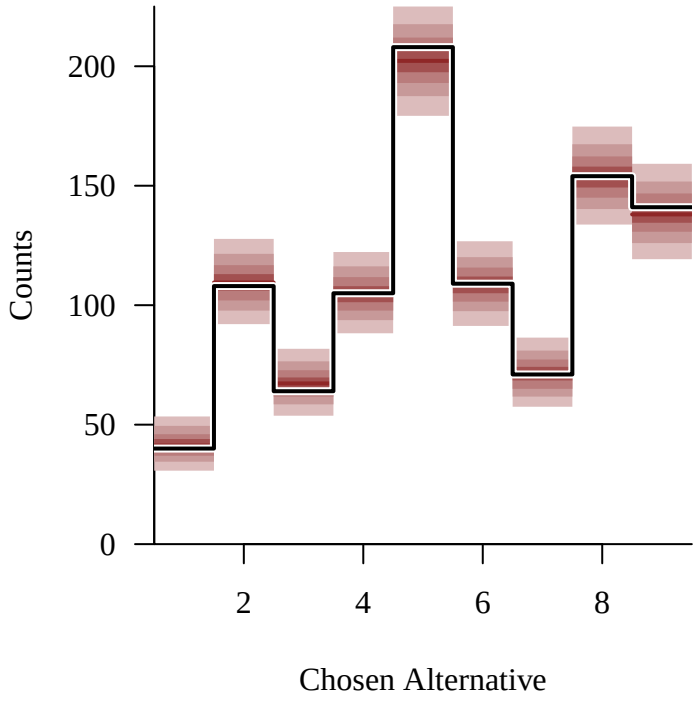


Because we're working with simulated data, we know that the independent Gumbel model adequately captures the relevant features of the true data generating process. In practice we would have to rely on posterior retrodictive checks to identify any inadequacies in our modeling

assumptions.

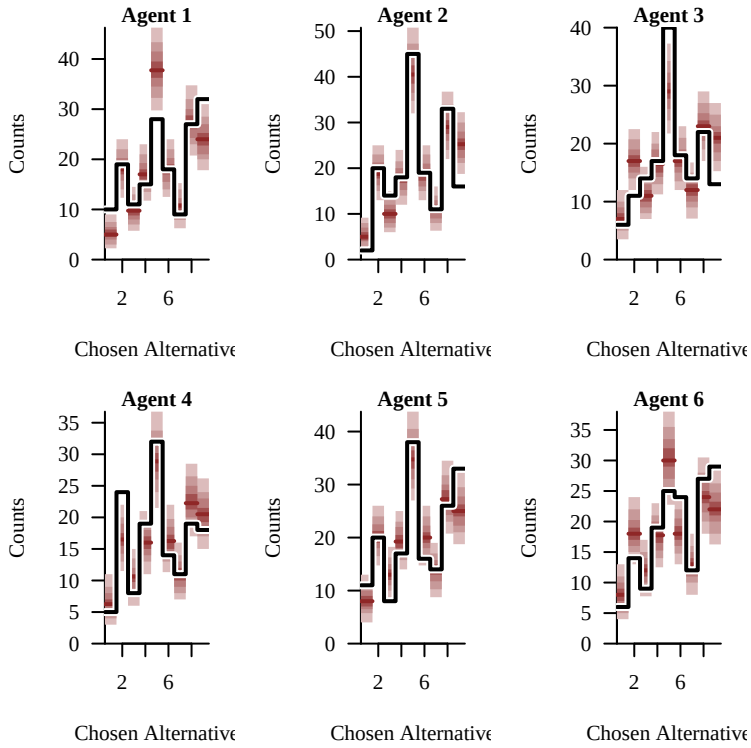
Conveniently, the histograms that we used to explore the data also make for productive visual retrodictive checks. Here we don't see any signs of model in adequacy.

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

util$plot_hist_quantiles(samples, 'choice_pred',
                         0.5, data$J + 0.5, 1,
                         baseline_values=data$choice,
                         xlab='Chosen Alternative')
```



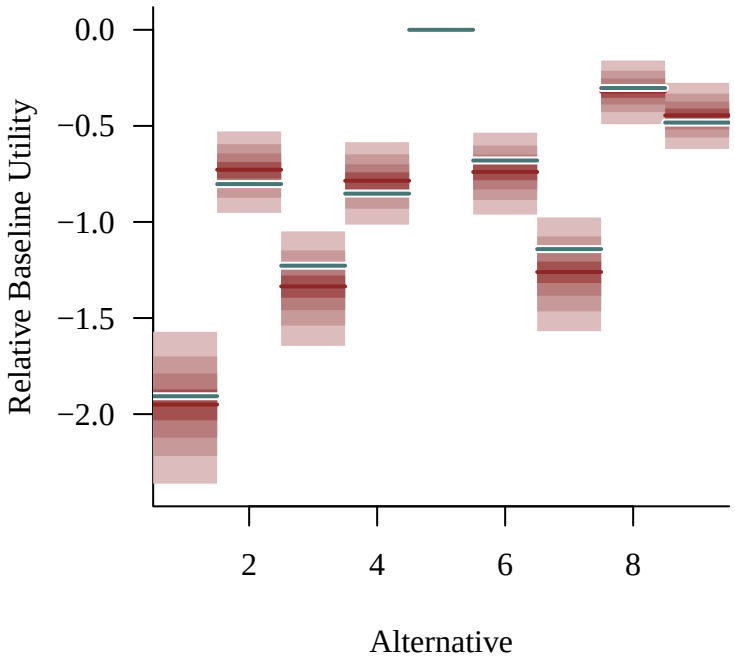

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (n in 1:data$N) {
  names <- sapply(which(data$agent == n),
                  function(n) paste0('choice_pred[', n, ']'))
  filtered_samples <- util$filter_expectands(samples, names)

  util$plot_hist_quantiles(
    filtered_samples, 'choice_pred',
    0.5, data$J + 0.5, 1,
```

```
    baseline_values=data$choice[data$agent == n],
    xlab='Chosen Alternative',
    main=paste('Agent', n)
  )
}
```

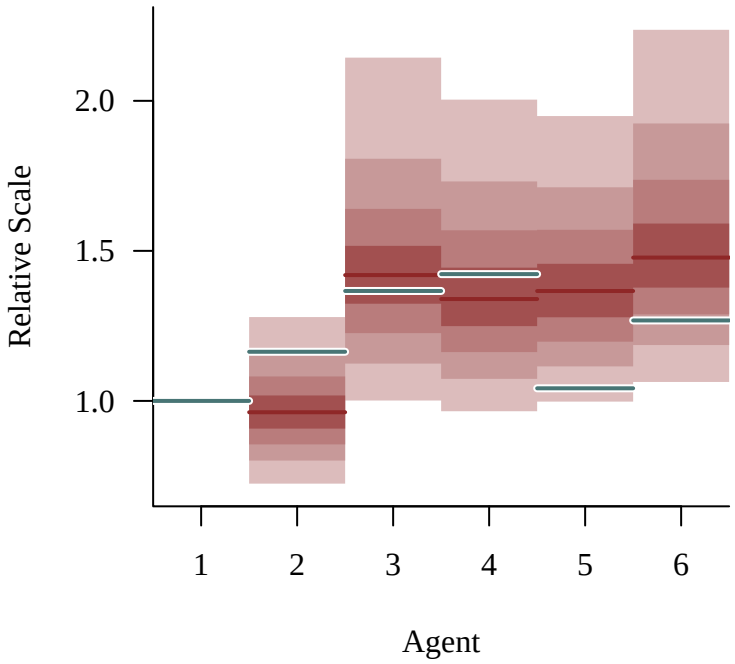


With an adequate model we can be confident that our posterior inferences are faithfully quantifying the behavior of the underlying data generating process, including the relative baseline utilities for each alternative and the relative scales for each agent.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J, function(j) paste0('omega[', j, ']'))
util$plot_disc_pushforward_quantiles(samples, names,
                                     baseline_values=omega_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Relative Baseline Utility")
```




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

tau_true <- sigma_true / sigma_true[1]

names <- sapply(1:data$N, function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(samples, names,
                                     baseline_values=tau_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Agent",
                                     ylab="Relative Scale")
```



3.4.2.5 Aggregated Implementation

When agents choose from the same choice set multiple times, we end up querying each set of choice probabilities repeatedly. Our previous model implementation ends up recalculating the choice probabilities for each observation, unnecessarily wasting computation.

By aggregating observations for each agent-choice set pairing, we can collapse the repeated categorical models into multinomial models. This allows us to compute the choice probabilities only once, making the implementation as efficient as possible.

```
choice_counts <- matrix(NA, data$N, data$J)
for (n in 1:data$N)
  choice_counts[n,] <- hist(data$choice[data$agent == n],
                            seq(0.5, data$J + 0.5, 1),
                            plot=FALSE)$counts

data$choice_counts <- choice_counts
```

```
fit <- stan(file='stan_programs/ig3.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

The only diagnostic warnings that we see here are a few $\hat{xi}$ warnings. These indicate heavy-tailed posteriors, and potentially unreliable expectation values, but don't cast any doubt on the overall posterior quantification.

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

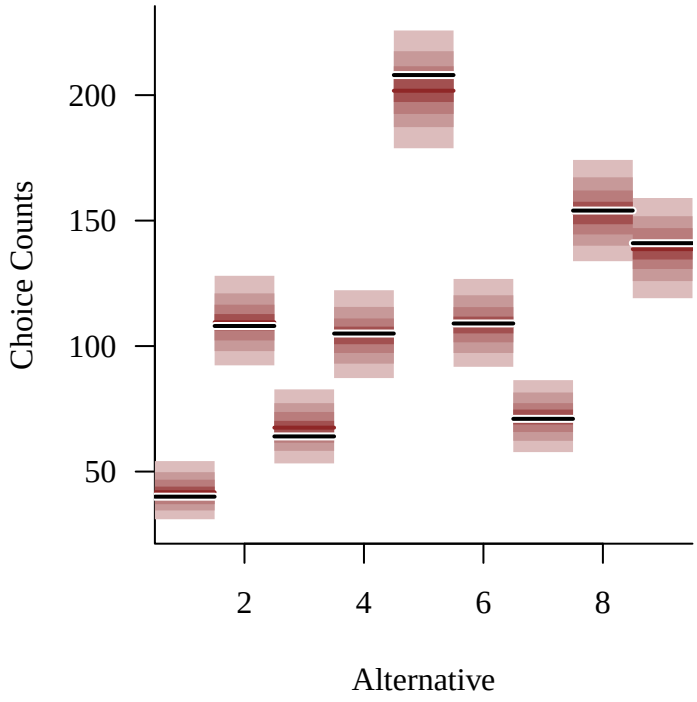
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

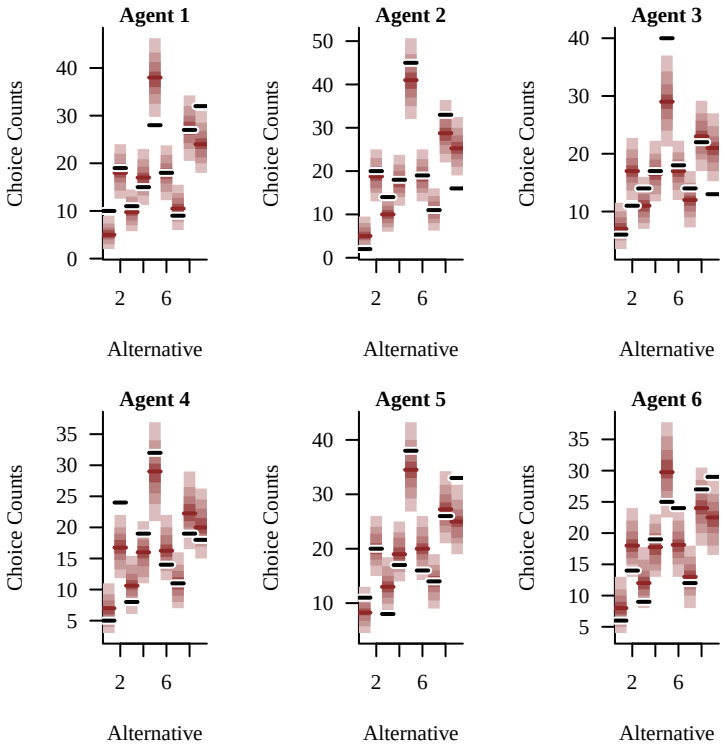
```
tau_free[5]:
  Chain 4: Right tail hat{xi} (0.391) exceeds 0.25.


Large tail hat{xi}s suggest that the expectand might not be
sufficiently integrable.
```

The posterior retrodictive checks behave identically to those we derived from `ig2.stan`. Note that, because we are working with aggregate counts and not individual observations, we have to visualize the histograms slightly differently here.

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

obs_counts <- colSums(data$choice_counts)

names <- sapply(1:data$J,
                function(j)
                paste0('agg_choice_counts_pred[', j, ']'))

util$plot_disc_pushforward_quantiles(samples, names,
                                     baseline_values=obs_counts,
                                     xlab="Alternative",
                                     ylab="Choice Counts")
```



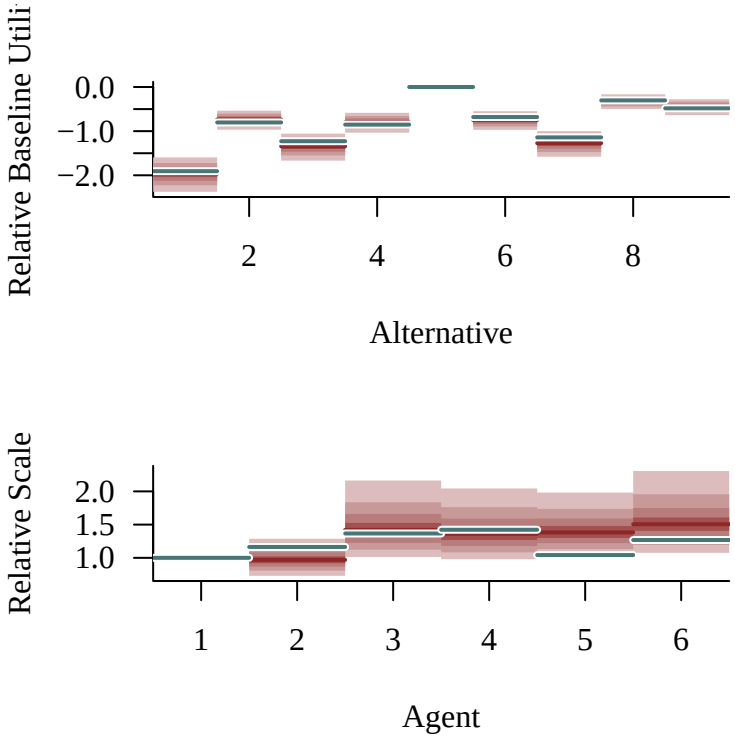

```
par(mfrow=c(2, 3), mar=c(5, 5, 1, 1))

for (n in 1:data$N) {
  obs_counts <- data$choice_counts[n,]

  names <- sapply(1:data$J,
                  function(j)
                  paste0('choice_counts_pred[', n, ',', j, ']'))
  util$plot_disc_pushforward_quantiles(samples, names,
                                       baseline_values=obs_counts,
                                       xlab="Alternative",
                                       ylab="Choice Counts",
                                       main=paste('Agent', n))
}
```



Similarly, posterior inferences for the relative model configurations are equivalent to the unaggregated model.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J,
                function(j) paste0('omega[', j, ']'))
util$plot_disc_pushforward_quantiles(
  samples, names,
  baseline_values=omega_true,
  baseline_col=util$c_mid_teal,
  xlab="Alternative",
  ylab="Relative Baseline Utility"
)

tau_true <- sigma_true / sigma_true[1]

names <- sapply(1:data$N,
                function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(
```

```
  samples, names,
  baseline_values=tau_true,
  baseline_col=util$c_mid_teal,
  xlab="Agent",
  ylab="Relative Scale"
)
```



3.4.3 Independent Gumbel Substitution Patterns

Now that we're comfortable with implementing an independent Gumbel model over a single choice set, we can explore the substitution patterns if manifests across multiple choice subsets.

3.4.3.1 Simulate Data

To infer substitution patterns we need to consider observations across different choice subsets. In practice this requires defining a global choice set of all of the possible alternatives, and then choice subsets from which we'll simulate observations.

Here let's take a choice set consisting of 10 alternatives, each of which are contiguously indexed.

```
J <- 10
```

We can then define choice subsets by the index of the included alternatives (Figure 2) Our first choice subset includes all but the eighth alternatives. The second choice subset removes the fifth alternative, while the third choice subset adds back in both missing alternatives. In particular we can think of the second and third choice subsets as the result of removing and adding, respectively, an alternative from the first choice subset.

```
subset_alts <-
  list(c(1, 2, 3, 4, 5, 6, 7,    9, 10),  # Nominal
       c(1, 2, 3, 4,    6, 7,    9, 10),  # Remove alternative 5
       c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))  # Add alternative 8

S <- length(subset_alts)
```

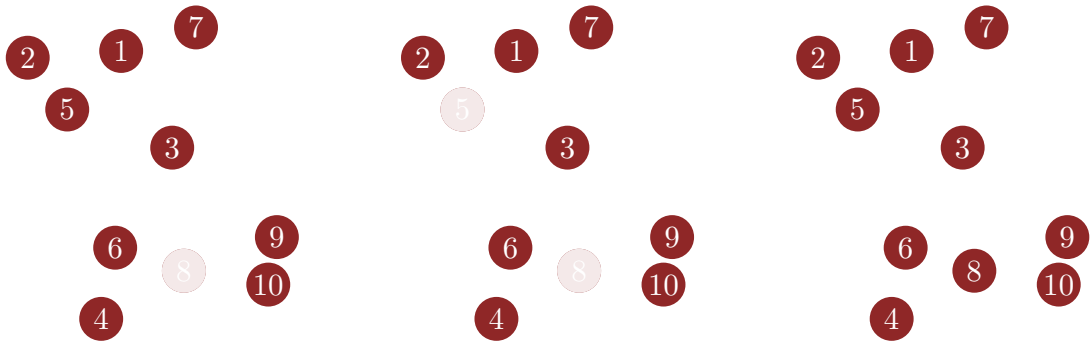


Figure 2: To study substitutions patterns for the independent Gumbel model we'll consider three choice subsets. The first choice subset contains nine of the ten possible alternatives while the second choice subset is given by removing one alternative. Finally the third choice subset is given by including all ten of the possible alternatives.

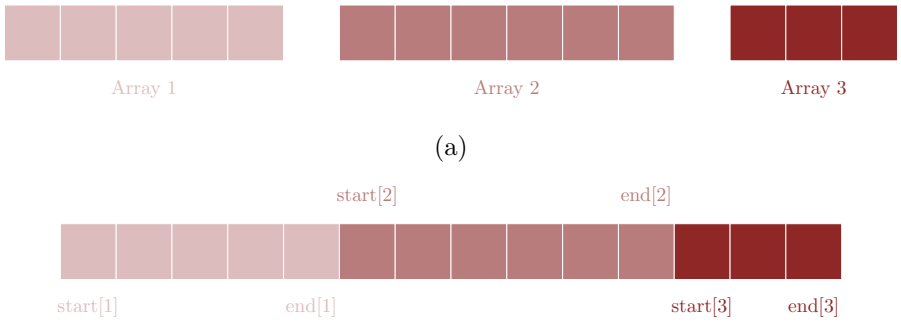
Perhaps the most difficult task when implementing discrete choice models over multiple choice subsets is keeping track of the indices. Within a choice subset we will need to index the included alternatives contiguously. This, however, means that the local indexing will not be the same as the global indexing of the full choice set!

Fortunately, the `subset_alts` arrays already map from local to global indexing. Even more, it's not too hard to convert this to an inverse map from global to local indexing.

```
to_subset_idx <-
  lapply(subset_alts,
         function(alts) sapply(1:J,
                               function(j) match(j, alts)))
```

The other awkward implementation issue concerns the encoding of `subset_alts`. Because each choice subset is a different size, `subset_alts` has the structure of a ragged array which we cannot implement directly in the `Stan` modeling language.

Here I'll concatenate all of the choice subset information into a single monolithic array (Figure 3). In order to retrieve the information for a particular subset, we use the start and end indices to segment the monolithic array.



(a)

(b)

Figure 3: (a) Ragged arrays combine multiple one-dimensional arrays of different sizes. Unfortunately they are not immediately accommodated by the rectangular arrays in the `Stan` modeling language. (b) That said, we can always implement a ragged array by concatenated the individual component arrays together into one long array and keeping track of the indices where each component starts and ends. In particular we can retrieve any component array by appropriately subsetting this concatenated array.

```
SJ <- 0
alts <- c()
subset_start <- c()
subset_end <- c()

for (s in 1:S) {
  subset_start <- c(subset_start, SJ + 1)
  SJ <- SJ + length(subset_alts[[s]])
  subset_end <- c(subset_end, SJ)
```

```
  alts <- c(alts, subset_alts[[s]])
}
```

With the organization of the choice subsets handled we can specify the final details. Here we'll consider the same six agents that we did previously.

```
N <- 6
```

To tease out any subtle substitution patterns, however, we'll simulate many more choice observations.

```
M <- 15000
```

Now we are finally ready to simulate some data. Note that the observed choices are defined in terms of the local indexing within choice subset.

```
simu <- stan(file="stan_programs/simu_ig2.stan",
             algorithm="Fixed_param", seed=8438338,
             data=list('N' = N, 'J' = J, 'M' = M,
                       'S' = S, 'SJ' = SJ,
                       'subset_start' = subset_start,
                       'subset_end' = subset_end,
                       'alts' = alts),
             warmup=0, iter=1, chains=1, refresh=0)

simu_fit <- extract(simu)
```

```
mu_true <- simu_fit$mu[1,]
sigma_true <- simu_fit$sigma[1,]
```

For maximum efficiency we'll use the aggregated model implementation, which requires calculating the observed counts for the alternatives in each choice subset.

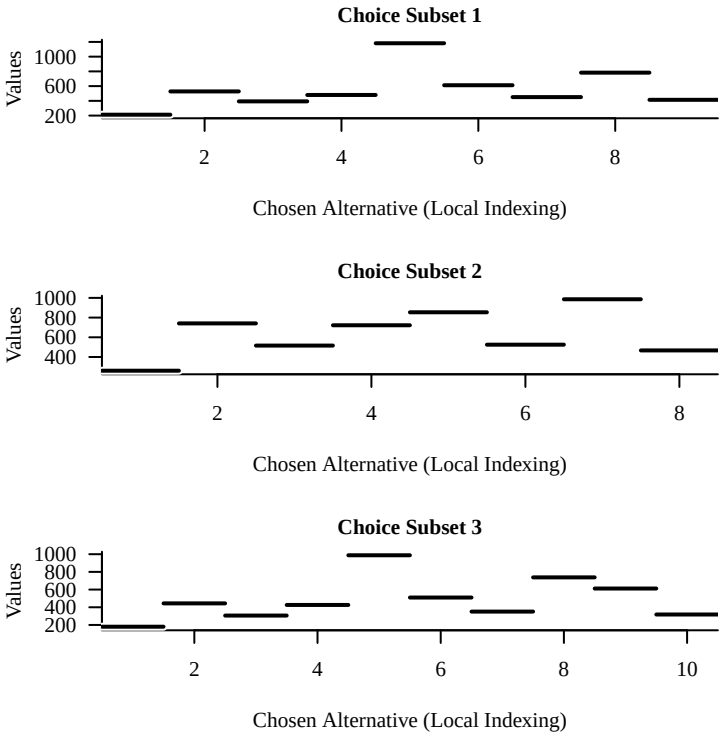
```
data <- list()

for (s in 1:S) {
  JJ <- subset_end[s] - subset_start[s] + 1
  choice_counts <- matrix(NA, N, JJ)
  for (n in 1:N)
    choice_counts[n,] <- hist(simu_fit$choice[  simu_fit$subset == s
```

```
                                                & simu_fit$agent  == n],
                                seq(0.5, JJ + 0.5, 1),
                                plot=FALSE)$counts

  data[[s]] <- list('N' = N, 'J' = JJ,
                    'choice_counts' = choice_counts)
}
```

3.4.3.2 Explore Data

In theory we could aggregate all of the observed choices into a histogram with respect to the global choice set alternatives. That said, this would convolve all of the choice subsets together making interpretation tricky at best. Because we are considering only a few choice subsets here, we can just as easily examine separate histograms for each choice subset.

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

for (s in 1:S) {
  util$plot_disc_vals(colSums(data[[s]]$choice_counts),
                      xlab='Chosen Alternative (Local Indexing)',
                      main=paste('Choice Subset', s))
}
```



We could also stratify each of these choice subset histograms by the corresponding agent if useful.

3.4.3.3 Individual Posterior Quantification

Let's start by fitting the observations of each choice subset separately.

In theory we can set different alternative anchors for each choice subset, but different anchors makes the resulting relative baseline utilities awkward to compare. To make comparisons more straightforward, we'll anchor the same alternative in all three choice subsets so that the relative baseline utilities are compatible. Any alternative that appears in all three choice subsets is a potential candidate.

```
common_alts <-
  Reduce(intersect,
         lapply(1:S,
                function(s) alts[subset_start[s]:subset_end[s]]))

for (alt in common_alts) {
  print(paste0('Alternative ', alt, ':'))
  for (s in 1:S) {
    MM <- sum(data[[s]]$choice_counts[,to_subset_idx[[s]][alt]])
```

```
    print(paste0('  ', MM, ' observations in choice subset ', s))
  }
}
```

```
[1] "Alternative 1:"
[1] "  212 observations in choice subset 1"
[1] "  261 observations in choice subset 2"
[1] "  180 observations in choice subset 3"
[1] "Alternative 2:"
[1] "  528 observations in choice subset 1"
[1] "  741 observations in choice subset 2"
[1] "  444 observations in choice subset 3"
[1] "Alternative 3:"
[1] "  393 observations in choice subset 1"
[1] "  516 observations in choice subset 2"
[1] "  306 observations in choice subset 3"
[1] "Alternative 4:"
[1] "  480 observations in choice subset 1"
[1] "  722 observations in choice subset 2"
[1] "  426 observations in choice subset 3"
[1] "Alternative 6:"
[1] "  612 observations in choice subset 1"
[1] "  854 observations in choice subset 2"
[1] "  510 observations in choice subset 3"
[1] "Alternative 7:"
[1] "  451 observations in choice subset 1"
[1] "  525 observations in choice subset 2"
[1] "  351 observations in choice subset 3"
[1] "Alternative 9:"
[1] "  783 observations in choice subset 1"
[1] "  986 observations in choice subset 2"
[1] "  612 observations in choice subset 3"
[1] "Alternative 10:"
[1] "  414 observations in choice subset 1"
[1] "  467 observations in choice subset 2"
[1] "  318 observations in choice subset 3"
```

We'll anchor the ninth alternative, which has the most observations amongst those that appear in all of the choice subsets.

```
anchor <- 9
```

That said, we have to be careful to translate this global alternative index to the local indices within each choice subset.

```
for (s in 1:S)
  data[[s]]$anchor <- to_subset_idx[[s]][anchor]
```

Now we can quantify separate posterior distributions for each of the data sets.

```
samples <- list()

for (s in 1:S) {
  fit <- stan(file='stan_programs/ig3.stan',
              data=data[[s]], seed=8438338,
              warmup=1000, iter=2024, refresh=0)

  cat(paste('Analyzing choice subset', s, ':\n'))

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

  cat('\n')

  samples[[s]] <- util$extract_expectand_vals(fit)
  base_samples <- util$filter_expectands(samples[[s]],
                                         c('omega_free',
                                           'tau_free'),
                                         TRUE)
  util$check_all_expectand_diagnostics(base_samples)

  cat('\n\n')
}
```

```
Analyzing choice subset 1 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 2 :
```

```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 3 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

3.4.3.4 Inferential Comparisons

For demonstration purposes let's focus on the substitution patterns between the third and tenth alternatives by studying how the corresponding choice probabilities vary across the three choice subsets. Because the choice probabilities will vary across agents, we'll also need to pick a specific agent.

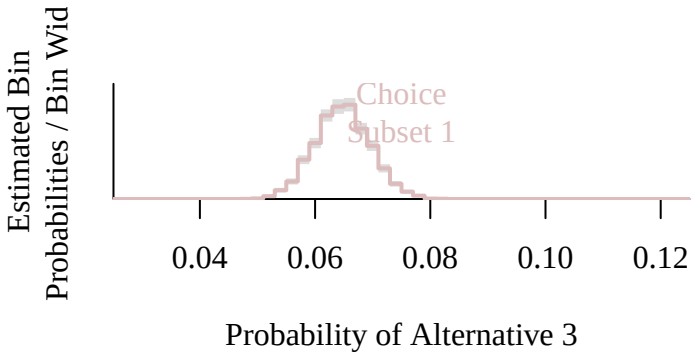
```
n <- 5
j1 <- 3
j2 <- 10
```

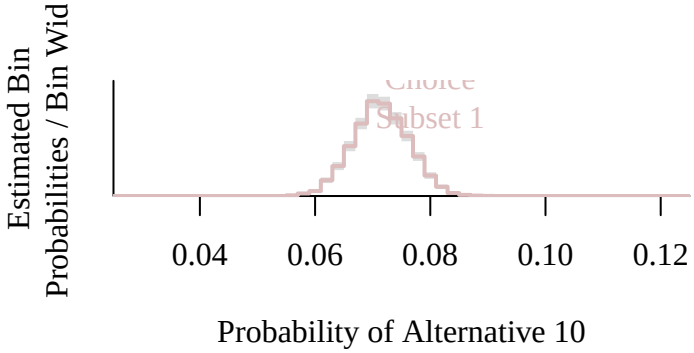
While log choice probabilities are more useful computationally, the untransformed choice probabilities are a bit easier to interpret.

```
j1_probs <- list()
j2_probs <- list()

for (s in 1:S) {
  var_repl <- list('x'=paste0('log_probs[', n, ',',
                              to_subset_idx[[s]][j1], ']'))
  j1_probs[[s]] <-
    util$eval_expectand_pushforward(samples[[s]],
                                    function(x) exp(x),
                                    var_repl)

  var_repl <- list('x'=paste0('log_probs[', n, ',',
                              to_subset_idx[[s]][j2], ']'))
  j2_probs[[s]] <-
    util$eval_expectand_pushforward(samples[[s]],
                                    function(x) exp(x),
                                    var_repl)
}
```

The baseline behavior is defined by the choice probabilities $q_{5,3}$ and $q_{5,10}$ in the nominal choice subset.

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

B <- 50
flim <- c(0.025, 0.125)

display_name <- paste('Probability of Alternative', j1)
util$plot_expectand_pushforward(j1_probs[[1]], B,
                                flim=flim, ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.075, 75, 'Choice\nSubset 1', col=util$c_light)

display_name <- paste('Probability of Alternative', j2)
util$plot_expectand_pushforward(j2_probs[[1]], B,
                                flim=flim,
                                ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.08, 85, 'Choice\nSubset 1', col=util$c_light)
```

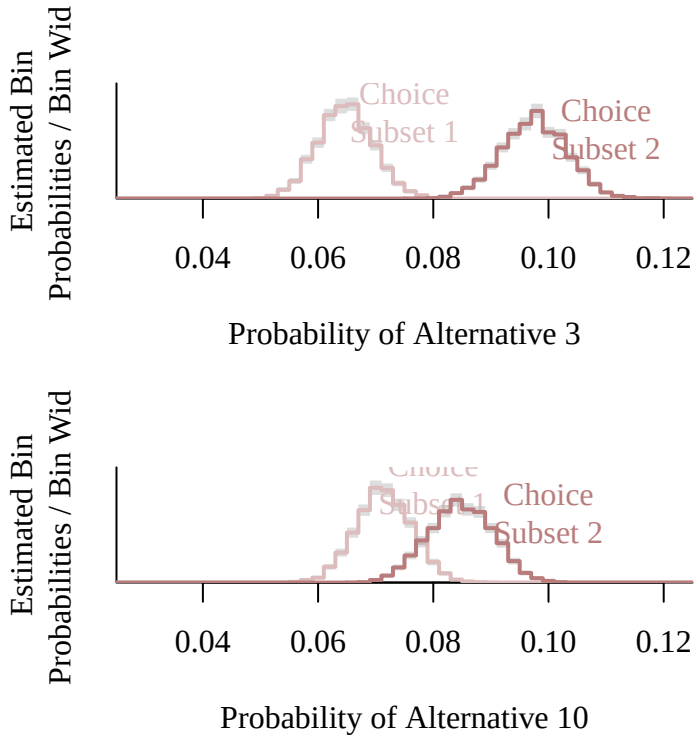




Moving from the first to the second choice subset requires removing an alternative. The probability initially allocated to that alternative has to then be redistributed amongst the remaining alternatives, increasing their choice probabilities.

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

B <- 50
flim <- c(0.025, 0.125)

display_name <- paste('Probability of Alternative', j1)
util$plot_expectand_pushforward(j1_probs[[1]], B,
                                flim=flim, ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.075, 75, 'Choice\nSubset 1', col=util$c_light)

util$plot_expectand_pushforward(j1_probs[[2]], B,
                                flim=flim,
                                col=util$c_mid,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.11, 60, 'Choice\nSubset 2', col=util$c_mid)
```

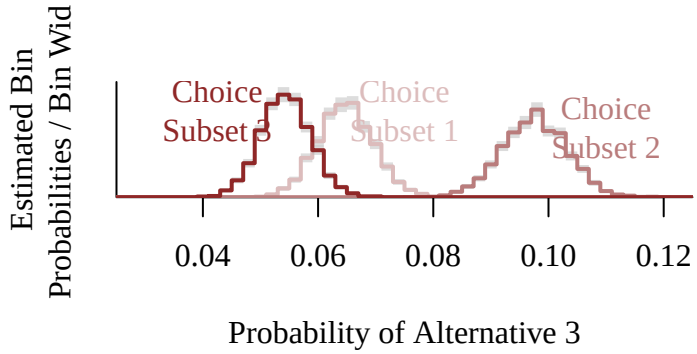
```
display_name <- paste('Probability of Alternative', j2)
util$plot_expectand_pushforward(j2_probs[[1]], B,
                                flim=flim,
                                ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.08, 85, 'Choice\nSubset 1', col=util$c_light)

util$plot_expectand_pushforward(j2_probs[[2]], B,
                                flim=flim,
                                col=util$c_mid,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.10, 60, 'Choice\nSubset 2', col=util$c_mid)
```

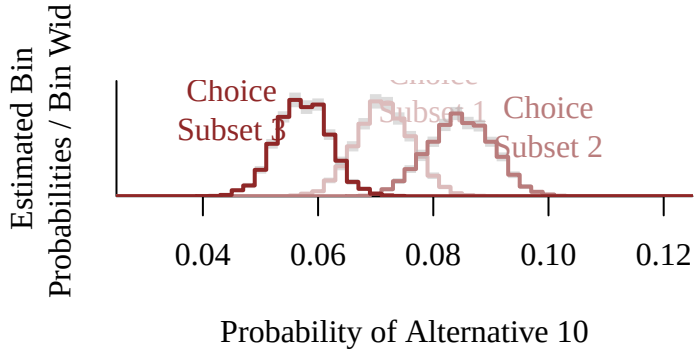


At the same time, moving from the first to the third choice subset requires adding a new alternative. The probability allocated to this new alternative has to siphoned from the probability allocated to the initial alternatives. Consequently both $q_{5,3}$ and $q_{5,10}$ *decrease*.

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

B <- 50
flim <- c(0.025, 0.125)

display_name <- paste('Probability of Alternative', j1)
util$plot_expectand_pushforward(j1_probs[[1]], B,
                                flim=flim, ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.075, 75, 'Choice\nSubset 1', col=util$c_light)

util$plot_expectand_pushforward(j1_probs[[2]], B,
                                flim=flim,
                                col=util$c_mid,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.11, 60, 'Choice\nSubset 2', col=util$c_mid)

util$plot_expectand_pushforward(j1_probs[[3]], B,
                                flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.0425, 75, 'Choice\nSubset 3', col=util$c_dark)

display_name <- paste('Probability of Alternative', j2)
util$plot_expectand_pushforward(j2_probs[[1]], B,
                                flim=flim,
                                ylim=c(0, 100),
                                display_name=display_name,
                                col=util$c_light)
text(0.08, 90, 'Choice\nSubset 1', col=util$c_light)

util$plot_expectand_pushforward(j2_probs[[2]], B,
                                flim=flim,
                                col=util$c_mid,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.10, 60, 'Choice\nSubset 2', col=util$c_mid)

util$plot_expectand_pushforward(j2_probs[[3]], B,
```

```
                                  flim=flim,
                                  col=util$c_dark,
                                  border="#BBBBBB88",
                                  add=TRUE)
text(0.045, 75, 'Choice\nSubset 3', col=util$c_dark)
```



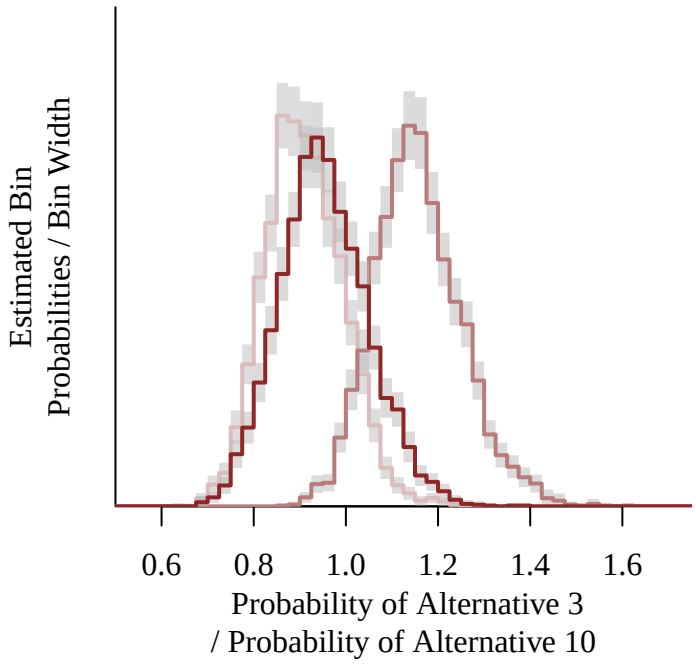


Because the independent Gumbel model exhibits independence from irrelevant alternatives, however, the *ratios* of choice probabilities should be consistent across any choice subset.

```
ratios <- list()

for (s in 1:S) {
  name1 <- paste0('log_probs[', n, ',',
                  to_subset_idx[[s]][j1], ']')
  name2 <- paste0('log_probs[', n, ',',
                  to_subset_idx[[s]][j2], ']')
  ratios[[s]] <-
    util$eval_expectand_pushforward(samples[[s]],
                                    function(x1, x2) exp(x1 - x2),
                                    list('x1'=name1, 'x2'=name2))
}
```

Indeed the posterior distributions for $q_{5,3}/q_{5,10}$ are consistent across all three of our choice subsets here. Note that they are not *exactly* the same because each posterior distribution is informed by a different set of observations.

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

B <- 50
flim <- c(0.5, 1.75)

display_name <- paste('  Probability of Alternative', j1, '\n/',
                      'Probability of Alternative', j2)
util$plot_expectand_pushforward(ratios[[1]],
                                B, flim=flim,
                                ylim=c(0, 6),
                                display_name=display_name,
                                col=util$c_light)

util$plot_expectand_pushforward(ratios[[2]],
                                B, flim=flim,
                                col=util$c_mid,
                                border="#BBBBBB88",
                                add=TRUE)

util$plot_expectand_pushforward(ratios[[3]],
                                B, flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)
```



3.4.3.5 Joint Posterior Quantification

By modeling the appropriate substitution patterns we don't have to limit ourselves to separate fits for each choice subset. We can equally well analyze all of the data at the same time using a single, joint discrete choice model.

To keep the organization consistent I will store these choice counts using the same ragged array pattern that I used for the choice subset alternatives.

```
choice_counts <- matrix(NA, N, SJ)
for (s in 1:S) {
  for (n in 1:N) {
    choice_counts[n, subset_start[s]:subset_end[s]] <-
      data[[s]]$choice_counts[n,]
  }
}
```

```
joint_data <- list('N'=N, 'J'=J,
                   'S'=S, 'SJ'=SJ,
                   'subset_start'=subset_start,
                   'subset_end'=subset_end,
                   'alts'=alts,
                   'choice_counts'=choice_counts,
                   'anchor'=anchor)
```

For demonstration purposes we'll specifically look at predictions for the fifth agent choosing from the full choice set. That said, we could equally just as easily make predictions for any agent choosing from any choice subset.

```
joint_data$n_pred <- 5
joint_data$J_pred <- J
joint_data$pred_alts <- 1:J
```

Now we let Hamiltonian Monte Carlo go to work.

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

The diagnostics are clean, giving us confidence in the accuracy of our posterior quantification.

```
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.
```

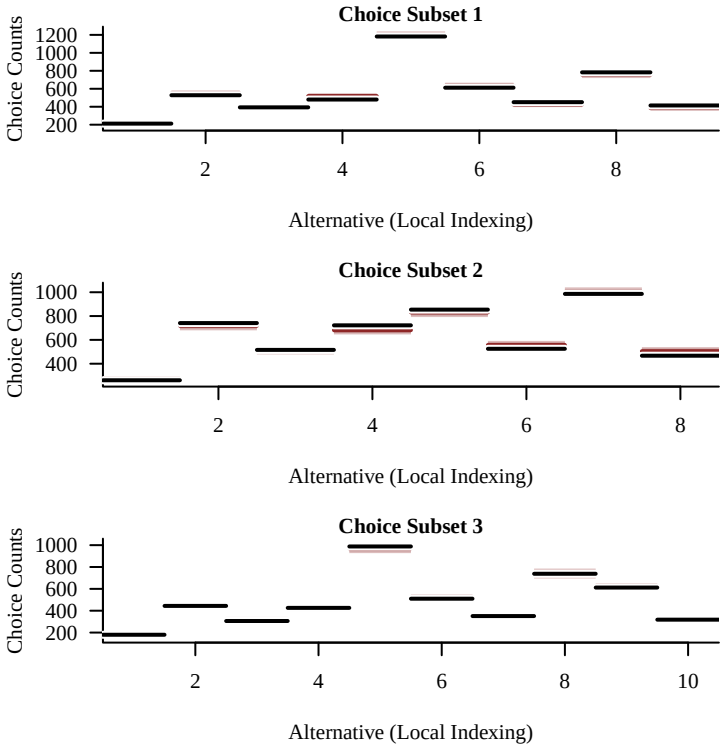
```
joint_samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(joint_samples,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

There is no appreciable retrodictive tension in any of the choice subsets.

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

for (s in 1:S) {
  agg_choice_counts_obs <-
    colSums(joint_data$choice_counts[,subset_start[s]:subset_end[s]])
```

```
  names <- sapply(subset_start[s]:subset_end[s],
                  function(j)
                  paste0('agg_choice_counts_pred[', j, ']'))
  util$plot_disc_pushforward_quantiles(
    joint_samples, names,
    baseline_values=agg_choice_counts_obs,
    xlab="Alternative (Local Indexing)",
    ylab="Choice Counts",
    main=paste("Choice Subset", s)
  )
}
```

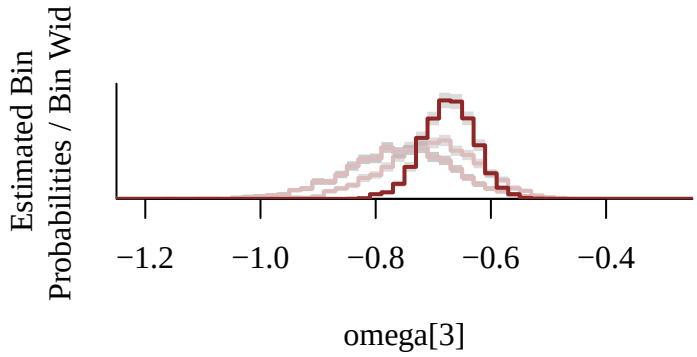


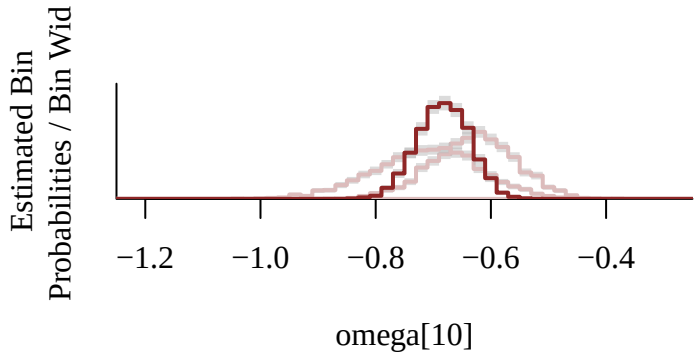
Posterior inferences of the relative baseline utilities drawn from the joint model are consistent with posterior inferences drawn from each of the individual models. By leveraging all of the data at the same time, however, the joint inferences enjoys much smaller uncertainties.

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

B <- 50
flim <- c(-1.25, -0.25)
```

```
display_name <- paste0('omega[', j1, ']')

name <- paste0('omega[', to_subset_idx[[1]][j1], ']')
util$plot_expectand_pushforward(samples[[1]][[name]],
                                B, flim=flim,
                                ylim=c(0, 10),
                                display_name=display_name,
                                col=util$c_light)

name <- paste0('omega[', to_subset_idx[[2]][j1], ']')
util$plot_expectand_pushforward(samples[[s]][[name]],
                                B, flim=flim,
                                col=util$c_light,
                                border="#BBBBBB88",
                                add=TRUE)

name <- paste0('omega[', to_subset_idx[[3]][j1], ']')
util$plot_expectand_pushforward(samples[[s]][[name]],
                                B, flim=flim,
                                col=util$c_light,
                                border="#BBBBBB88",
                                add=TRUE)

name <- paste0('omega[', j1, ']')
util$plot_expectand_pushforward(joint_samples[[name]],
                                B, flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)



flim <- c(-1.25, -0.25)

display_name <- paste0('omega[', j2, ']')

name <- paste0('omega[', to_subset_idx[[1]][j2], ']')
util$plot_expectand_pushforward(samples[[1]][[name]],
                                B, flim=flim,
                                ylim=c(0, 10),
                                display_name=display_name,
                                col=util$c_light)
```

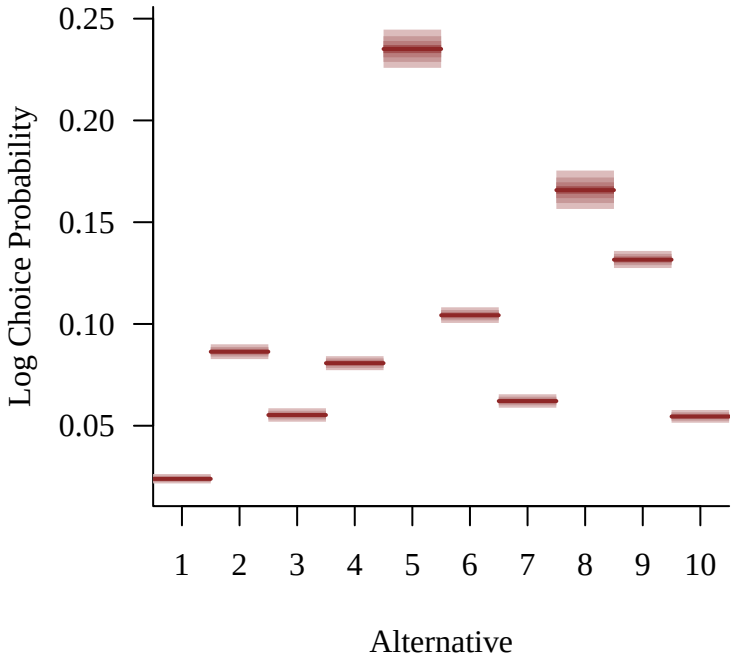
```
name <- paste0('omega[', to_subset_idx[[2]][j2], ']')
util$plot_expectand_pushforward(samples[[s]][[name]],
                                B, flim=flim,
                                col=util$c_light,
                                border="#BBBBBB88",
                                add=TRUE)

name <- paste0('omega[', to_subset_idx[[3]][j2], ']')
util$plot_expectand_pushforward(samples[[s]][[name]],
                                B, flim=flim,
                                col=util$c_light,
                                border="#BBBBBB88",
                                add=TRUE)

name <- paste0('omega[', j2, ']')
util$plot_expectand_pushforward(joint_samples[[name]],
                                B, flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)
```





We can then use these precise inferences to make predictions that generalize to any choice subset.

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

names <- sapply(1:joint_data$J_pred,
                function(j) paste0('log_prob_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(joint_samples, names,
                                     xlab="Alternative",
                                     xticklabs=joint_data$pred_alts,
                                     ylab="Log Choice Probability")
```



4 Coupled Utility Models

One of the most productive ways to enable more sophisticated substitution patterns in discrete models, in particular to transcend independence from irrelevant alternatives, is to couple the components of the probabilistic utility model together. The more coupled the utilities of two alternatives are, the better substitutions those two alternatives are for each other than other alternatives. In this section we'll review some of the more common discrete choice models that are derived from coupled utility models.

4.1 Nested Gumbel Model

The **nested Gumbel model** partitions a choice set into subsets known as **nests**, with alternatives in the same nest more strongly coupled together than those across different nests.

More formally, consider partitioning a given choice set into K subsets. We denote each subset as B_k so that if $j \in B_k$ then the jth alternative is in the kth group. When implementing these models in practice, it will also useful to be able to quickly refer to the nest containing the jth alternative. Here I will use the notation $k(j)$.

Given a partition of the choice set, the nested Gumbel model is most easily defined not with a joint probability density function but rather with the a joint cumulative distribution function,

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{k=1}^{K} \left(\sum_{j \in B_k} e^{-\frac{1}{\lambda_k} \frac{u_{nj} - \mu_{nj}}{\sigma_n}} \right)^{\lambda_k} \right).$$

The λ_k are positive variables that quantify how strongly the utilities within each nest are coupled together, with larger values corresponding to weaker couplings.

When $\lambda_k = 1$ then the behavior of the nested Gumbel model in the kth nest reduces to the behavior of the independent Gumbel, at least up to a scale parameter. Values of λ_k larger than one can lead to self-consistent model configurations, but not always. To simplify the use of this model here, I will consider only

$$0 \leq \lambda_k \leq 1.$$

In theory we could differentiate this joint cumulative distribution function with respect to each argument to derive a corresponding joint probability density function. The results, however, are not particularly insightful and not needed to derive choice probabilities. Consequently we will not consider the nested Gumbel probability density function any further.

4.1.1 Nested Gumbel As A Generalization of Independent Gumbel

We can also derive the nested Gumbel model as a generalization of the independent Gumbel model. Recall that the cumulative distribution function is given by

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{j=1}^{J} e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}} \right).$$

Given a nesting structure, we can always split the sum in the exponential up into a sum over nests and then a sum over the alternatives in each nest,

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{k=1}^{K} \left(\sum_{j \in B_k} e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}} \right) \right).$$

At this point we embrace a vector space perspective (Horn and Johnson 1985) and recognize that the inner sums can be interpreted as 1-norms over each nest,

$$\| x_1, \ldots, x_I \|_1 = \sum_{i=1}^{I} x_i.$$

The immediate generalization of 1-norms are p-norms,

$$\| x_1, \ldots, x_I \|_p = \left(\sum_{i=1}^{I} x_i^p \right)^{\frac{1}{p}}.$$

When $p = 1$, all of elements contribute to the norm equally. As p increases, however, the norm becomes dominated more and more by the one element with the largest value.

Allowing p to vary across nests then gives the generalized utility model

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{k=1}^{K} \left(\sum_{j \in B_k} e^{-p_k \frac{u_{nj} - \mu_{nj}}{\sigma_n}} \right)^{\frac{1}{p_k}} \right).$$

Defining

$$\lambda_k = \frac{1}{p_k}$$

immediately gives the nested Gumbel model.

Even if we don't derive the nested Gumbel model in this way, the connection to p-norms provides intuition for the behavior in each nest. When $\lambda_k = 1$, for example, all of the alternatives in the nest contribute equally. As λ_k decreases, however, the the one alternative with the largest utility residual $u_{nj} - \mu_{nj}$ begins to dominate (Figure 4).

4.1.2 Choice Probabilities

Following our general construction, the choice probabilities are given by differentiating the joint cumulative distribution function, evaluating all arguments on the target utility, and then integrating. After a few pages of calculations, which I have reserved for Appendix A.3, this gives

$$\begin{aligned} q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u) \\ &= \frac{e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj}}{\sigma_n}} \left(\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k(j)} - 1}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{1}{\lambda_{k'}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k'}}}. \end{aligned}$$

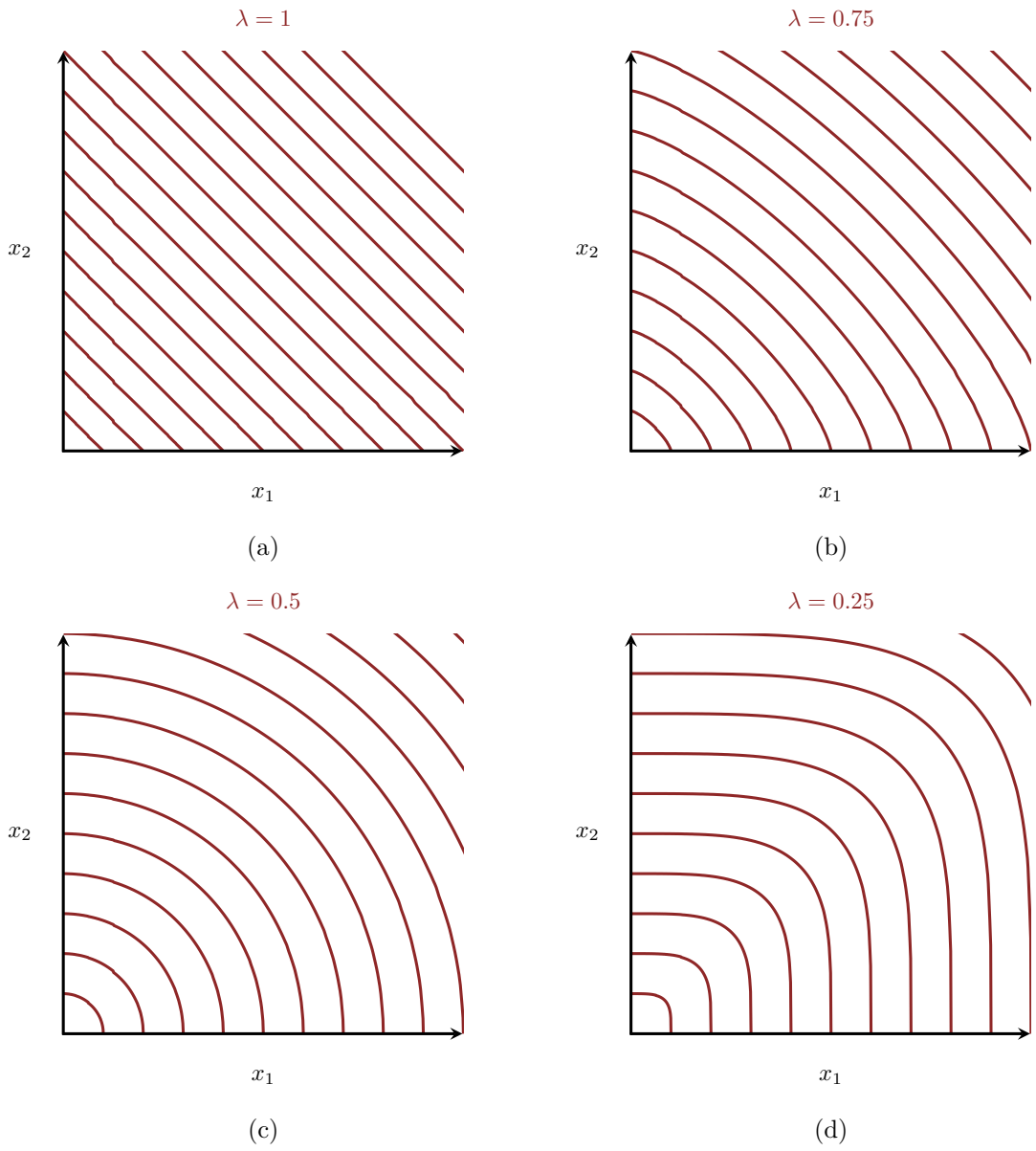


Figure 4: The contribution of each nest in the nested Gumbel model is given by a p-norm from linear algebra, $\left(\sum_{j \in B_k} x_j^{p_k}\right)^{1/p_k}$, with $p_k = 1/\lambda_k$ and $x_j = \exp\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n}\right)$. (a) When $\lambda_k = 1$ the contribution is given by the 1-norm – as known as the taxicab norm, Manhattan norm, and ℓ^1-norm – where each x_j contributes equally. (b, c, d) As \$p_{k} increases, and λ_k decreases, the nest k nest sum becomes more and more dominated by the largest x_j.

4.1.2.1 Conditional and Marginal Choice Probabilities

The choice probabilities become substantially more interpretable with a little bit of mathematical manipulation. In particular, we can manipulate them into a product of two softmax functions,

$$\begin{aligned} q_{nj} &= \frac{e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj}}{\sigma_n}} \left(\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k(j)} - 1}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{1}{\lambda_{k'}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k'}}} \\ &= \frac{e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj}}{\sigma_n}}}{\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}}} \cdot \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{1}{\lambda_{k'}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k'}}}. \end{aligned}$$

The first term,

$$q_{nj|k(j)} = \frac{e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj}}{\sigma_n}}}{\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}}}$$

quantifies the *conditional probability* of choosing the jth alternative from the $k(j)$th nest, while the second term

$$q_{nk(j)} = \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{1}{\lambda_{k'}} \frac{\mu_{nj'}}{\sigma_n}} \right)^{\lambda_{k'}}}$$

quantifies the *marginal probability* of choosing the $k(j)$th nest from the partition. In other words, the nesting structure of the nested Gumbel model induces a natural conditional decomposition of the choice probabilities,

$$q_{nj} = q_{nj|k(j)} \, q_{nk(j)}.$$

4.1.2.1.1 Efficient Calculation

Using the softmax function we can calculate all of the conditional probabilities for the alternatives within a particular nest,

$$(j_1, \ldots, j_{J_k}) \in B_k,$$

at the same time,

$$(q_{nj_1|k}, \ldots, q_{nj_{J_k}|k}) = \text{softmax}\left(\frac{1}{\lambda_k} \frac{\mu_{nj_1}}{\sigma_n}, \ldots, \frac{1}{\lambda_k} \frac{\mu_{nj_{J_k}}}{\sigma_n} \right).$$

With a little bit of work we can also calculate all of the marginal probabilities for the nests with a single softmax function evaluation,

$$\begin{aligned}
&(q_{n1}, \ldots, q_{nK}) \\
&\quad = \text{softmax}\left(\lambda_1 \log \sum_{j' \in B_1} e^{\frac{1}{\lambda_1} \frac{\mu_{nj'}}{\sigma_n}}, \right. \\
&\qquad \ldots, \\
&\qquad \left. \lambda_K \log \sum_{j' \in B_K} e^{\frac{1}{\lambda_K} \frac{\mu_{nj'}}{\sigma_n}} \right).
\end{aligned}$$

The marginal probabilities simplify even further with the use of a log-sum-exp function,

$$\text{log-sum-exp}(x_1, \ldots, x_L) = \log \sum_{l=1}^{L} e^{x_l}.$$

This allows us to write

$$(q_{n1}, \ldots, q_{nK}) = \text{softmax}(\lambda_1 C_1, \ldots, \lambda_K C_K)$$

with

$$C_k = \text{log-sum-exp}\left(\frac{1}{\lambda_k} \frac{\mu_{nj_1}}{\sigma_n}, \ldots, \frac{1}{\lambda_k} \frac{\mu_{nj_{J_k}}}{\sigma_n} \right)$$

4.1.2.2 Redundancies

The form of the conditional choice probabilities within each nest is identical to the choice probabilities that we would derived from the independent Gumbel model over that nest. This immediately indicates that the baseline utilities within each nest are subject to their own translation redundancy.

Independently translating the baseline utilities within each nest, however, does not result in the same marginal probabilities. Instead there is only a global translation redundancy across all of the baseline utilities. Writing

$$\nu_{nj} = \frac{\mu_{nj}}{\sigma_n},$$

we have

$$
\begin{aligned}
q_{nk(j)}(\nu_{n1}, \ldots, \nu_{nJ}) &= \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{\nu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{\nu_{nj'}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
&= \frac{e^{\alpha}}{e^{\alpha}} \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{\nu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_{k'}} e^{\frac{\nu_{nj'}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
&= \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{\alpha}{\lambda_{k(j)}}} e^{\frac{\nu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} e^{\alpha} \left(\sum_{j' \in B_{k'}} e^{\frac{\alpha}{\lambda_{k'}}} e^{\frac{\nu_{nj'}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
&= \frac{\left(\sum_{j' \in B_{k(j)}} e^{\frac{\nu_{nj'}+\alpha}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}}}{\sum_{k'=1}^{K} e^{\alpha} \left(\sum_{j' \in B_{k'}} e^{\frac{\nu_{nj'}+\alpha}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
&= q_{nk(j)}(\nu_{n1} + \alpha, \ldots, \nu_{nJ} + \alpha)
\end{aligned}
$$

All of this is to say that the same relative baseline utility and relative scale approach that we considered in Section 3.2 will eliminate the redundancies of the nested Gumbel model as well!

4.1.3 Substitution Patterns

The nested Gumbel model couples the alternative utilities within each nest together, making alternatives within a nest better substitutes for each other than alternatives in other nests. These substitution patterns can be quantified in various ways.

4.1.3.1 Ratios of Choice Probabilities

In a nested Gumbel model the ratios of choice probabilities for any two alternatives is given by

$$
\begin{aligned}
\frac{q_{nj_1}}{q_{nj_2}} &= \frac{q_{nj_1|k(j_1)}\, q_{nk(j_1)}}{q_{nj_2|k(j_2)}\, q_{nk(j_2)}} \\
&= \frac{q_{nj_1|k(j_1)}}{q_{nj_2|k(j_2)}} \frac{q_{nk(j_1)}}{q_{nk(j_2)}}.
\end{aligned}
$$

If two alternatives are in the same nest,

$$k(j_1) = k(j_2) = k,$$

then the marginal probabilities cancel,

$$\begin{aligned}
\frac{q_{nj_1}}{q_{nj_2}} &= \frac{q_{nj_1|k(j_1)}\, q_{nk(j_1)}}{q_{nj_2|k(j_2)}\, q_{nk(j_2)}} \\
&= \frac{q_{nj_1|k}\, q_{nk}}{q_{nj_2|k}\, q_{nk}} \\
&= \frac{q_{nj_1|k}}{q_{nj_2|k}},
\end{aligned}$$

leaving only the ratio of conditonal probabilities.

This ratio, however, depends on only the two corresponding baseline utilities,

$$\begin{aligned}
\frac{q_{nj_1}}{q_{nj_2}} &= \frac{q_{nj_1|k}}{q_{nj_2|k}} \\
&= \frac{e^{\frac{1}{\lambda_k}\frac{\mu_{nj_1}}{\sigma_n}}}{\sum_{j' \in B_k} e^{\frac{1}{\lambda_k}\frac{\mu_{nj'}}{\sigma_n}}} \frac{\sum_{j' \in B_k} e^{\frac{\mu_{nj'}}{\lambda_k}}}{e^{\frac{1}{\lambda_k}\frac{\mu_{nj_2}}{\sigma_n}}} \\
&= \exp\left(\frac{1}{\lambda_k}\frac{\mu_{nj_1} - \mu_{nj_2}}{\sigma_n}\right).
\end{aligned}$$

In other words the nested Gumbel model exhibits independent of irrelevant alternatives *within each nest*.

When comparing alternatives across nests, however, the marginal probabilities no longer cancel. Instead we have

$$\begin{aligned}
\frac{q_{nj_1}}{q_{nj_2}} &= \frac{q_{nj_1|k(j_1)}}{q_{nj_2|k(j_2)}} \frac{q_{nk(j_1)}}{q_{nk(j_2)}} \\
&= \frac{e^{\frac{1}{\lambda_k}\frac{\mu_{nj_1}}{\sigma_n}}}{e^{\frac{1}{\lambda_k}\frac{\mu_{nj_2}}{\sigma_n}}} \frac{\left(\sum_{j' \in B_{k(j_1)}} e^{\frac{1}{\lambda_{k(j_1)}}\frac{\mu_{nj'}}{\sigma_n}}\right)^{\lambda_{k(j_1)}}}{\left(\sum_{j' \in B_{k(j_2)}} e^{\frac{1}{\lambda_{k(j_2)}}\frac{\mu_{nj'}}{\sigma_n}}\right)^{\lambda_{k(j_2)}}}.
\end{aligned}$$

The ratio of the choice probabilities now depend on *all* of the baseline utilities across the two nests. Whenever the behavior of any alternatives in either of the nests change, so two will the substitution pattern between j_1 and j_2.

In general the coupling of the nested Gumbel model allows alternatives in one nest to be better substitutes for each other than any alternative in another nest. If both $\lambda_{k(j_1)}$ and $\lambda_{k(j_2)}$ are strictly less than one and if

$$\sum_{j' \in B_{k(j_1)}} e^{\frac{1}{\lambda_{k(j_1)}}\frac{\mu_{nj'}}{\sigma_n}}$$

decreases relative

$$\sum_{j' \in B_{k(j_2)}} e^{\frac{1}{\lambda_{k(j_2)}} \frac{\mu_{nj'}}{\sigma_n}}$$

then the choice probability q_{nj_1} will *increase* relative to q_{nj_2}, and vice versa.

In other words any choice probability removed from one alternative within a nest are preferentially transferred to the choice probabilities of the other alternatives in that same nest. Taking this to the limit, removing an item from $B_{k(j_1)}$ but keeping everything else fixed decreases

$$\sum_{j' \in B_{k(j_1)}} e^{\frac{1}{\lambda_{k(j_1)}} \frac{\mu_{nj'}}{\sigma_n}}$$

relative to the corresponding sum for the $k(j_2)$ nest. This in turn increases q_{nj_1} relative to q_{nj_2} (Figure 5).

4.1.3.2 Elasticities

The substitution patterns are also mirrored in the behavior of the elasticities. That said, the analytic form of the elasticities become substantially more ungainly. After a long calculation Appendix A.1 we have

$$\begin{aligned}
\epsilon_{njj'} &= \frac{\partial q_{nj}}{\partial \mu_{nj'}} \\
&= \frac{\mu_{nj'}}{q_{nj}} \frac{\partial}{\partial \mu_{nj'}} \left[\frac{e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj}}{\sigma_n}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{1}{\lambda_{k(j)}} \frac{\mu_{nj''}}{\sigma_n}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{1}{\lambda_{k'}} \frac{\mu_{nj''}}{\sigma_n}} \right)^{\lambda_{k'}}} \right] \\
&= \left(\quad \delta_{j'j} \frac{1}{\lambda_{k(j)}} \right. \\
&\quad + \delta_{k(j')k(j)} \frac{\lambda_{k(j)} - 1}{\lambda_{k(j)}} q_{nj'|k(j)} \\
&\quad \left. - q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{1}{\lambda_{k(j')}} \frac{\mu_{nj''}}{\sigma_n}} \right) \frac{\mu_{nj'}}{\sigma_n},
\end{aligned}$$

where δ denotes the Kronecker delta function,

$$\delta_{ii'} = \begin{cases} 0, & i \neq i' \\ 1, & i = i' \end{cases} .$$

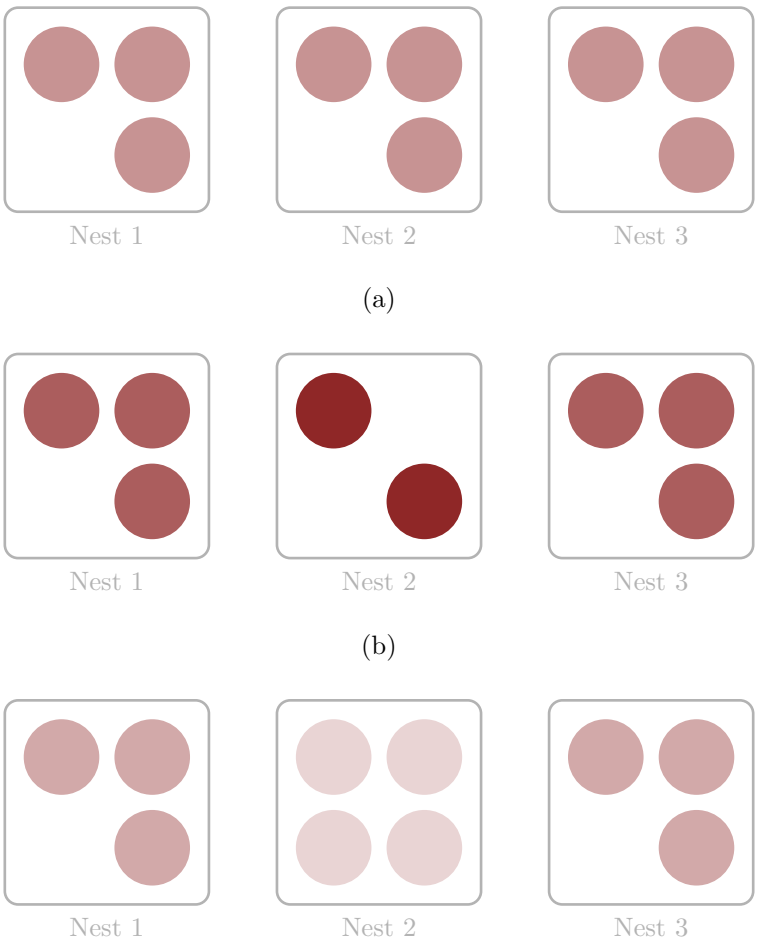


Figure 5: The nested Gumbel model couples the alternative utilities within each nest together, making alternatives within a nest better substitutes for each other than alternatives in other nests. This immediately impacts how the choice probabilities change across different choice subsets. (a) Consider, for example, an initial choice subset with nine alternatives partitioned into three nests. (b) Removing an alternative from the second nest frees up excess choice probability. In general the choice probabilities of all of the alternatives will increase, but the choice probabilities of the alternatives in the second nest will increase more as they better replace the removed alternative. (c) In order to add an alternative to the second nest, we need to siphon choice probability from initial alternatives. While all of the choice probabilities will typically decrease a little, the choice probabilities in the second nest will decrease the most as the new alternative cannibalizes agent demand.

In particular we can identify when independence of irrelevant alternatives arises by comparing the elasticities defined by three distinct alternatives, j_1, j_2, and j', in different nesting scenarios.

For example if j_1 and j' are in different nests,

$$k(j_1) \neq k(j'),$$

then only the third term of the elasticity remains,

$$\epsilon_{nj_1j'} = \frac{\mu_{nj'}\, q_{nj'}}{\sigma_n} \sum_{j'' \in B_{k(j')}} e^{\frac{1}{\lambda_{k(j')}} \frac{\mu_{nj''}}{\sigma_n}},$$

which is independent of j_1 and its nest. Consequently if j_2 and is also in a different nest than j', but not necessarily in a different nest than j_1,

$$k(j_1) \neq k(j') \text{ and } k(j_2) \neq k(j'),$$

then the elasticities are equal,

$$\epsilon_{nj_1j'} = \epsilon_{nj_2j'}.$$

Consequently the ratio of q_{nj_1}/q_{nj_2} will be invariant to changes in the behavior of the j'th alternative.

At the same time if all three alternatives are in the same nest,

$$k(j_1) = k(j_2) = k(j') \equiv k$$

then the elasticities become

$$\epsilon_{nj_1j'} = \frac{\mu_{nj'}}{\sigma_n} \left(q_{nj'|k} \frac{\lambda_k - 1}{\lambda_k} + q_{nj'} \sum_{j'' \in B_k} e^{\frac{1}{\lambda_k} \frac{\mu_{nj''}}{\sigma_n}} \right)$$

$$\epsilon_{nj_2j'} = \frac{\mu_{nj'}\, q_{nj'}}{\sigma_n} \left(q_{nj'|k} \frac{\lambda_k - 1}{\lambda_k} + q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{1}{\lambda_k} \frac{\mu_{nj''}}{\sigma_n}} \right).$$

The two elasticities are exactly equal once again, and the ratio of q_{nj_1}/q_{nj_2} will be invariant to changes in the behavior of the j'th alternative.

This behavior changes, however, when j' is in the same nest as j_1 or j_2. For example if $k(j') = k(j_1) \neq k(j_2)$ then

$$\epsilon_{nj_1j'} = \frac{\mu_{nj'}}{\sigma_n} \left(q_{nj'|k(j_1)} \frac{\lambda_{k(j_1)} - 1}{\lambda_{k(j_1)}} + q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{1}{\lambda_{k(j')}} \frac{\mu_{nj''}}{\sigma_n}} \right)$$

but

$$\epsilon_{nj_2j'} = \frac{\mu_{nj'}}{\sigma_n} \left(q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{1}{\lambda_{k(j')}} \frac{\mu_{nj''}}{\sigma_n}} \right)$$

In this case the elasticities are not equal and changes in the behavior of the j' alternative will influence q_{nj_1}/q_{nj_2}.

Altogether these results imply that the nested Gumbel model exhibits independence of irrelevant alternatives within each individual nest. Any changes outside of the nests containing any alternatives being considered do not change the corresponding substitution patterns.

4.2 Generalized Nested Gumbel Model

Conveniently the nested Gumbel model can be generalized without compromising analytic tractability by removing the constraint that the nests must be disjoint.

To allow nests to overlap we introduce for each alternative a set of parameters that quantify the partial association to each nest,

$$(\alpha_{j1}, \ldots, \alpha_{jK}),$$

that form a simplex,

$$0 \leq \alpha_{jk} \leq 1, \quad \sum_{k=1}^{K} \alpha_{jk} = 1.$$

The possible configurations of these partial simplices is not arbitrary but rather is restricted by the structure of the nests. In particular α_{jk} can be non-zero if and only if $j \in B_k$.

With partial associations we can generalize the probabilistic utility model to the joint cumulative distribution function,

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{k=1}^{K} \left(\sum_{j \in B_k} \left(\alpha_{jk} \, e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k} \right).$$

If for each alternative j we have $\alpha_{jk} = 1$ for one, and only one, k then the nests become disjoint and this model reduces to the nested Gumbel model.

Deriving the choice probabilities becomes more straightforward if we first define the subset of nests that contain a given alternative, A_j. In other words $k \in A_j$ if and only if $j \in B_k$.

After a long calculation Appendix A.3, this organization allows us to write the choice probabilities as

$$
\begin{aligned}
q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, \frac{\partial \Pi}{\partial x_j}(u_{nj}, \dots, u_{nj}) \\
&= \frac{\sum_{k \in A_j} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\frac{\mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k}}.
\end{aligned}
$$

Similar the choice probabilities in the nested Gumbel model, the choice probabilities in the generalized nested Gumbel model admit a convenient conditional decomposition,

$$
\begin{aligned}
q_{nj} &= \frac{\sum_{k \in A_j} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\frac{\mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k}} \\
&= \sum_{k \in A_j} \frac{\left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\frac{\mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k'}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \\
&= \sum_{k \in A_j} \frac{\left(\alpha_{jk}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}} \frac{\left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k'}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \\
&\equiv \sum_{k \in A_j} q_{nj|k}\, q_{nk}.
\end{aligned}
$$

Here

$$
q_{nj|B_k} = \frac{\left(\alpha_{jk}\, e^{\frac{\mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}
$$

quantifies the conditional probabilities of the alternative within each nest, while

$$q_{nk} = \frac{\left(\sum_{j' \in B_k} \left(\alpha_{j'k} \, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}} \right)^{\lambda_k}}{\sum_{k'=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k'} \, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_{k'}}} \right)^{\lambda_{k'}}}$$

quantifies the marginal probability of each nest.

With some careful manipulation we can also write these probabilities as the output of softmax functions, which is particularly useful for practical implementations. For example

$$\begin{aligned} q_{nj|B_k} &= \frac{\left(\alpha_{jk} \, e^{\frac{\mu_{nj}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}}{\sum_{j' \in B_k} \left(\alpha_{j'k} \, e^{\frac{\mu_{nj'}}{\sigma_n}} \right)^{\frac{1}{\lambda_k}}} \\ &= \frac{\exp\left(\frac{1}{\lambda_k} \left(\log \alpha_{jk} + \frac{\mu_{nj}}{\sigma_n} \right) \right)}{\sum_{j' \in B_k} \exp\left(\frac{1}{\lambda_k} \left(\log \alpha_{j'k} + \frac{\mu_{nj}}{\sigma_n} \right) \right)} \end{aligned}$$

with

$$(q_{n1|k}, \ldots, q_{nJ|k}) = \text{softmax}\left(\frac{1}{\lambda_k} \left(\log \alpha_{1k} + \frac{\mu_{n1}}{\sigma_n} \right), \ldots, \frac{1}{\lambda_k} \left(\log \alpha_{Jk} + \frac{\mu_{nJ}}{\sigma_n} \right) \right).$$

The substitution patterns exhibited by these generalized choice probabilities are, unsurprisingly, more complicated than those of the nested Gumbel model. Specifically the summation over all nests that contain a given alternative immediately obstructs the cancellation we would need for the ratios of choice probabilities to be independent of the behavior of other alternatives. I will leave the exploration of these substitution patterns as an exercise for the enthusiastic reader…

4.3 The Probit Model

Another popular discrete choice modeling technique with coupled utilities is the **probit model** which assumes a multivariate normal utility model for each agent,

$$p(u_{n1}, \ldots, u_{nJ}) = \text{multinormal}\left(u_{n1}, \ldots, u_{nJ} \mid \mu_n, \Sigma_n \right).$$

In order to calculate the choice probabilities in this model,

$$q_{nj} = \pi \left(\bigcap_{j' \neq j} \{ u_{nj} > u_{nj'} \} \right),$$

we have to be able to integrate the multivariate normal probability density function over these non-rectangular subsets.

Unfortunately these integrals are not available in closed form. At the same time, numerical integration is either too inaccurate or too expensive to be productive when we are working with more than a few alternatives.

Because of these challenges the probit model is typically implemented not by calculating the choice probabilities directly but rather by explicitly modeling the latent utilities. Specifically, the utilities for each agent

$$u_{n1}, \ldots, u_{nJ}$$

are elevated to parameters with the observational model

$$p(\tilde{y}_m \mid u_{n(m)1}, \ldots, u_{n(m)J}) = \begin{cases} 1, & \bigcap_{j' \neq \tilde{y}_m} \{u_{n(m)\tilde{y}_m} > u_{n(m)j'}\} \\ 0, & \text{else.} \end{cases}$$

This observational model, however, presents some immediate practical challenges. In particular the likelihood functions, and then posterior distributions, derived from this observational model will always be *discontinuous*, with some configurations of the utilities equally consistent with any observed data and some completely disallowed.

Because of this discontinuous behavior, the gradients of the derived likelihood functions are not informative. This, in turn, obstructs not only optimization-based maximum likelihood estimators but also scalable Markov chain Monte Carlo tools like Hamiltonian Monte Carlo.

Historically, probit models have largely been limited to Bayesian analyses using Markov chain Monte Carlo with relatively inefficient Gibbs samplers. Because the methodology required for efficient implementation and diagnostics of these tools is pretty distinct from Hamiltonian Monte Carlo, I will not consider probit models further in this chapter.

4.4 Demonstration

Let's put all of this theory into context by analyzing data simulated from a model with more sophisticated substitution patterns than the model from which we simulated in Section 3.5.

4.4.1 Basics

As we did previously, let's start with analyzing data arising from a single, fixed choice set.

4.4.1.1 Simulate Data

In this case we'll use a nested Gumbel model for our simulation truth. This means configuring the nests and the intra-nest couplings within the given choice set.

```
K <- 3
nests <- c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3)

lambda_true <- c(0.65, 0.4, 0.8)
```

The baseline utilities and agent scales will behave similar to before.

To implement the nested Gumbel model we need to be able to sum over all of the alternatives in each nest. This, in, turn, requires knowing which alternatives are in each nest. There are a few different ways to do this, but here I'll use the `num_which_int` and `which_int` functions to dynamically lookup the appropriate alternatives.

```
simu <- stan(file="stan_programs/simu_ng1.stan",
             algorithm="Fixed_param", seed=8438338,
             data=list('N' = N, 'J' = J, 'M' = M,
                       'K' = K, 'nests' = nests,
                       'lambda' = lambda_true),
             warmup=0, iter=1, chains=1, refresh=0)

simu_fit <- extract(simu)
```

Once the individual choices have been simulated we can aggregate them across agent-alternative pairs again.

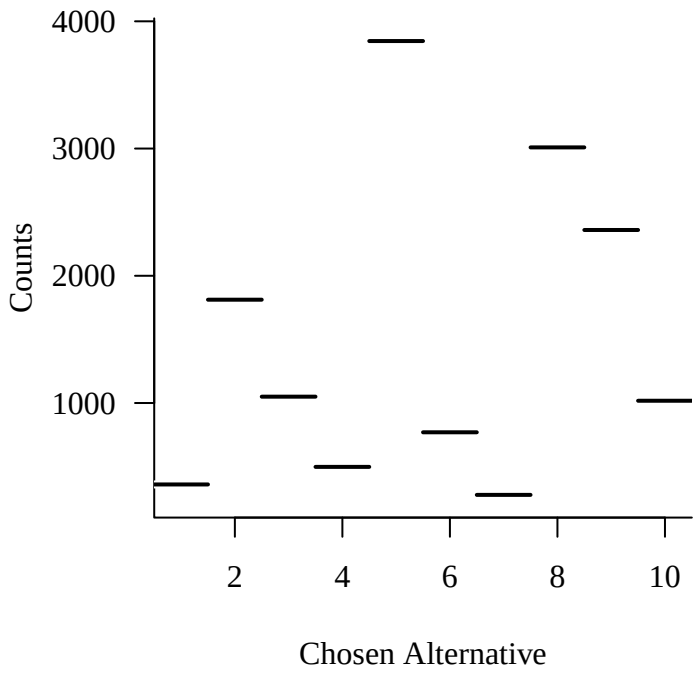
```
mu_true <- simu_fit$mu[1,]
sigma_true <- simu_fit$sigma[1,]
log_prob_true <- simu_fit$log_prob[1,,]

data <- list('N' = N, 'J' = J,
             'K' = K, 'nests' = nests,
             'agent' = simu_fit$agent[1,],
             'choice' = simu_fit$choice[1,])

choice_counts <- matrix(NA, data$N, data$J)
for (n in 1:data$N)
  choice_counts[n,] <- hist(data$choice[data$agent == n],
                            seq(0.5, data$J + 0.5, 1),
                            plot=FALSE)$counts
```

```
data$choice_counts <- choice_counts
```

4.4.1.2 Explore Data

The aggregate histogram of choices across all agents still proves informative.

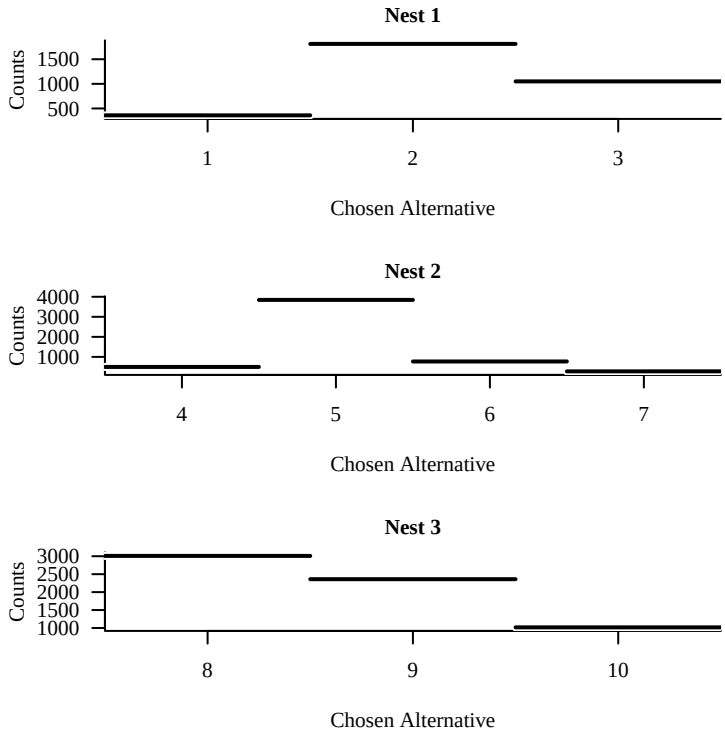
```
par(mfrow=c(1, 1), mar = c(5, 5, 2, 1))

agg_choice_counts <- colSums(data$choice_counts)

util$plot_disc_vals(agg_choice_counts,
                    xlab='Chosen Alternative',
                    ylab='Counts')
```



Now, however, we can also stratify these histograms by nest to summarize the behavior within each nest.

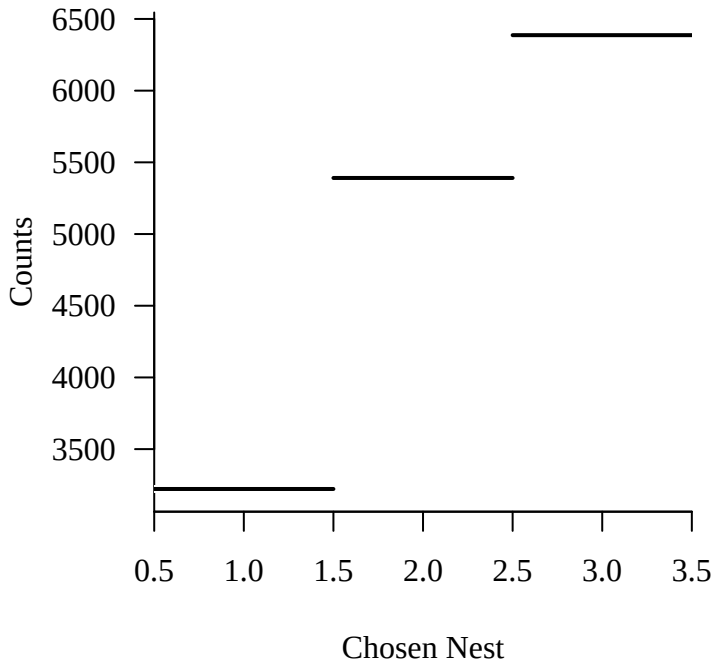
```
par(mfrow=c(3, 1), mar = c(5, 5, 2, 1))

for (k in 1:data$K) {
  alts <- which(data$nests == k)
  nest_alt_choice_counts <- agg_choice_counts[alts]
```

```
  util$plot_disc_vals(nest_alt_choice_counts,
                      xlab='Chosen Alternative',
                      xticklabs=alts,
                      ylab='Counts',
                      main=paste0('Nest ', k))
}
```



At the same time, we can summarize the relative abundance of choices across nests by aggregating counts over all of the alternatives in each nest. Here we see that the third nest contains the most attractive alternatives.

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

nest_choice_counts <-
  sapply(1:K,
         function(k)
         sum(agg_choice_counts[which(data$nests == k)]))

util$plot_disc_vals(nest_choice_counts,
                    xlab='Chosen Nest',
                    ylab='Counts')
```



4.4.1.3 Independent Gumbel Model

Although we know that the true data generating process couples the baseline utilities, let's see what happens when we assume the simpler independent Gumbel model.

Once again we'll choose the anchor alternative based on the total number of observed choices.

```
colSums(data$choice_counts)
```

```
 [1]  360 1812 1050  498 3845  770  278 3009 2360 1018
```

```
data$anchor <- 5
```

```
# Posterior quantification
fit <- stan(file='stan_programs/ig3.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

All of the diagnostics are clean, indicting that our posterior computation is reasonably accurate.

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

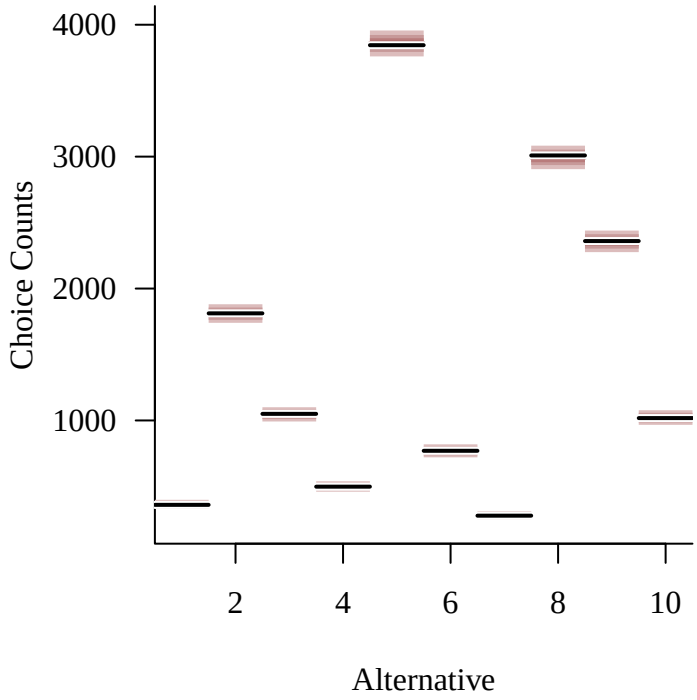
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

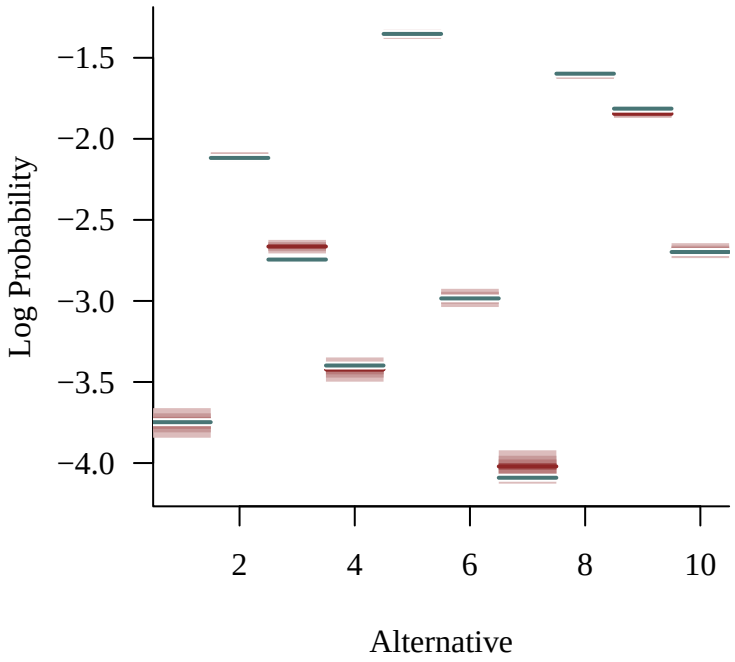
Moreover, despite the oversimplified assumptions the aggregate posterior retrodictive behavior is great.

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

names <- sapply(1:data$J,
                function(j)
                paste0('agg_choice_counts_pred[', j, ']'))

util$plot_disc_pushforward_quantiles(
  samples, names,
  baseline_values=agg_choice_counts,
  xlab="Alternative",
  ylab="Choice Counts"
)
```



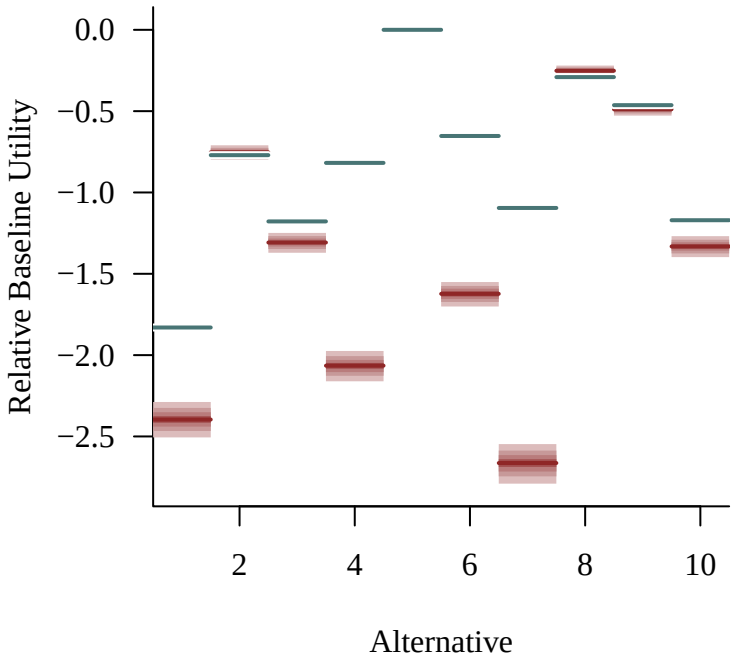
The inferred log choice probabilities are also well-behaved, matching the true behaviors from the simulation.

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

names <- sapply(1:data$J,
                function(j)
                paste0('log_probs[1,', j, ']'))
util$plot_disc_pushforward_quantiles(
  samples, names,
  baseline=log_prob_true[1,],
  baseline_col=util$c_mid_teal,
  xlab="Alternative",
  ylab="Log Probability"
)
```



Inferences for the baseline utilities, however, are less satisfying, as they concentrate pretty far away from the true values.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J, function(j) paste0('omega[', j, ']'))
util$plot_disc_pushforward_quantiles(samples, names,
                                     baseline_values=omega_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Relative Baseline Utility")
```



Regardless of the behavior of the true data generating process, the independent Gumbel model is perfectly adequate for capturing the choice probabilities *within a fixed choice set*. The limitations of the independent Gumbel model will manifest only when we consider observations across *multiple* choice subsets at the same time. We'll investigate that in Section 4.4.2.

4.4.1.4 Nested Gumbel Model

Before trying to fit data from multiple choice subsets, let's spend some time implementing the nested Gumbel model for this single choice set. Here we'll also use the `num_which_int` and `which_int` functions to make the choice probability calculations a bit cleaner.

```
fit <- stan(file='stan_programs/ng1.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Despite the fact that we're now using the correct model, the posterior computation has faltered. Specifically we now see divergences that indicate stunted exploration.

```
diagnostics1 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics1)
```

```
  Chain 2: 27 of 1024 transitions (2.6%) diverged.
```

```
  Chain 3: 1 of 1024 transitions (0.1%) diverged.

  Divergent Hamiltonian transitions result from unstable numerical
trajectories.  These instabilities are often due to degenerate target
geometry, especially "pinches".  If there are only a small number of
divergences then running with adept_delta larger than 0.801 may reduce
the instabilities at the cost of more expensive Hamiltonian
transitions.
```

```
samples1 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples1,
                                       c('omega_free',
                                         'tau_free',
                                         'lambda'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

To root out the underlying problem we'll need to find where the divergences at concentrating, which means examining scatter plots of pairs of parameters. We can try to brute force our way through *all* of the possible scatter plots, but let's take a second to consider the structure of the model and motivate particularly suspect pairs.

Going through the mathematical structure of the nested Gumbel model we see that the relative baseline utilities within each nest always appear with the corresponding intra-nest coupling parameter,

$$\exp\left(\frac{1}{\lambda_{k(j')}}\frac{\omega_{nj''}}{\tau_n}\right).$$

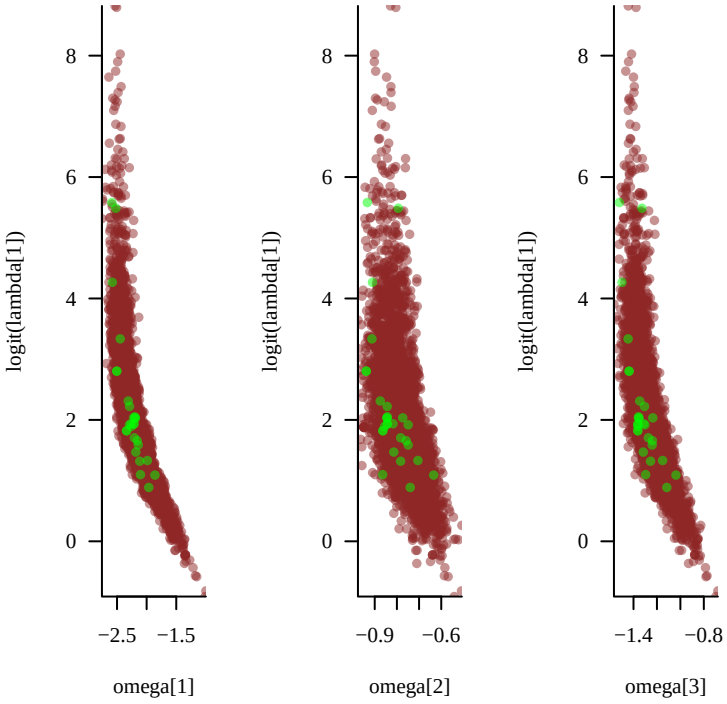
If the data mostly inform the ratio

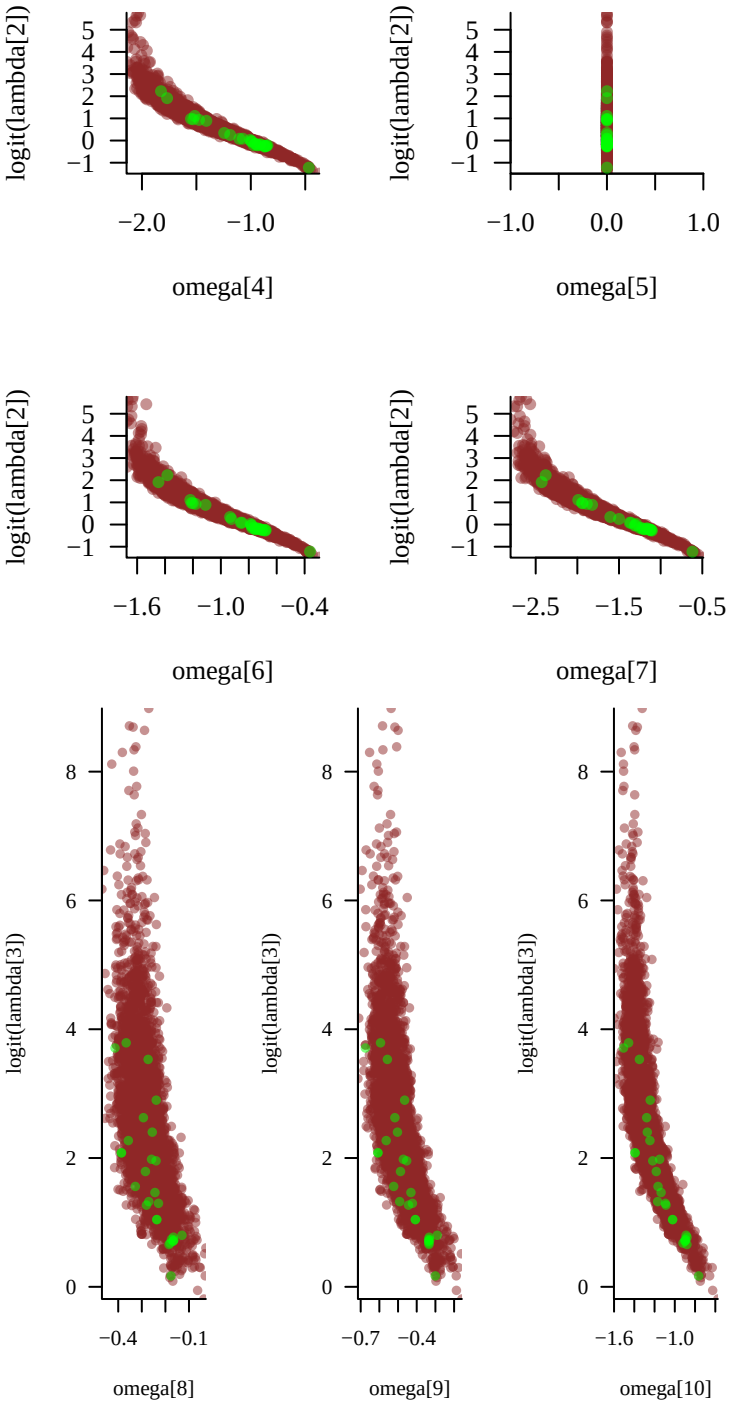
$$\frac{\omega_{nj''}}{\lambda_{k(j')}}$$

and $\omega_{nj''}$ and $\lambda_{k(j')}$ individually, then any inferences for these parameters will exhibit a problematic degeneracy. Consequently let's look at these pairs first.

```
lambda_names <- sapply(1:data$K,
                       function(k) paste0('lambda[', k, ']'))
transforms <- setNames(as.list(rep(2, data$K)), lambda_names)

for (k in 1:K) {
```

```
  alt_names <- sapply(which(data$nest == k),
                      function(j) paste0('omega[', j, ']'))

  util$plot_div_pairs(alt_names, lambda_names[k],
                      samples1, diagnostics1,
                      transforms=transforms)
}
```





Looks like our suspicions were spot on. As each $\omega_{nj''}$ increases the corresponding λ_k decreases to keep the ratio between the two approximately constant. That is, however, until we see the divergences start to concentrate, suggesting that the Hamiltonian Monte Carlo sampler has

stalled prematurely.

Now that we've identified a likely culprit we can consider possible improvements to our model implementation. Here let's reparameterize our model using the ratios that the data seem to be directly informing,

$$\zeta_{nj''} = \frac{\omega_{nj''}}{\lambda_{k(j')}}.$$

We can then recover inferences for $\omega_{nj''}$ by multiplying $\zeta_{nj''}$ and $\lambda_{k(j')}$.

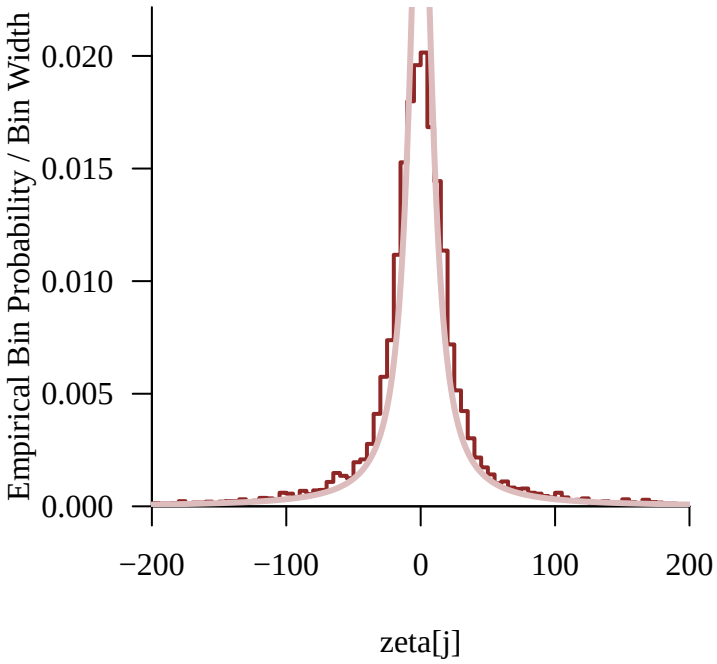
Once immediate difficulty with this reparameterization is updating the prior model appropriately. An analytic transformation of the prior model is technically possible, but it is not particularly straightforward to implement in practice. Fortunately, however, a Cauchy prior model approximates the exact transformation reasonably well.

```
S <- 10000
zetas <- rnorm(S, 0, 10) / runif(S, 0, 1)

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

util$plot_line_hist(zetas, -200, 200, 5,
                    xlab='zeta[j]', prob=TRUE,
                    col=util$c_dark)
```

```
Warning in check_bin_containment(bin_min, bin_max, values): 199 values (2.0%)
fell below the binning.
```

```
Warning in check_bin_containment(bin_min, bin_max, values): 204 values (2.0%)
fell above the binning.
```

```
xs <- seq(-200, 200, 0.1)
ys <- dcauchy(xs, 0, 10)
lines(xs, ys, lwd=3, col=util$c_light)
```



This gives us everything that we need.

```
fit <- stan(file='stan_programs/ng2.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

The diagnostics are a little bit better – the divergences are gone – but that are some new warnings to parse. Specifically the realized Markov chains exhibit heavier tails and stronger autocorrelations than ideal. Because these warnings indicate only inefficient and not unreliable posterior quantification, however, we'll press on here.

```
diagnostics2 <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics2)
```

```
  Chain 2: 2 of 1024 transitions (0.20%)
           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.
```

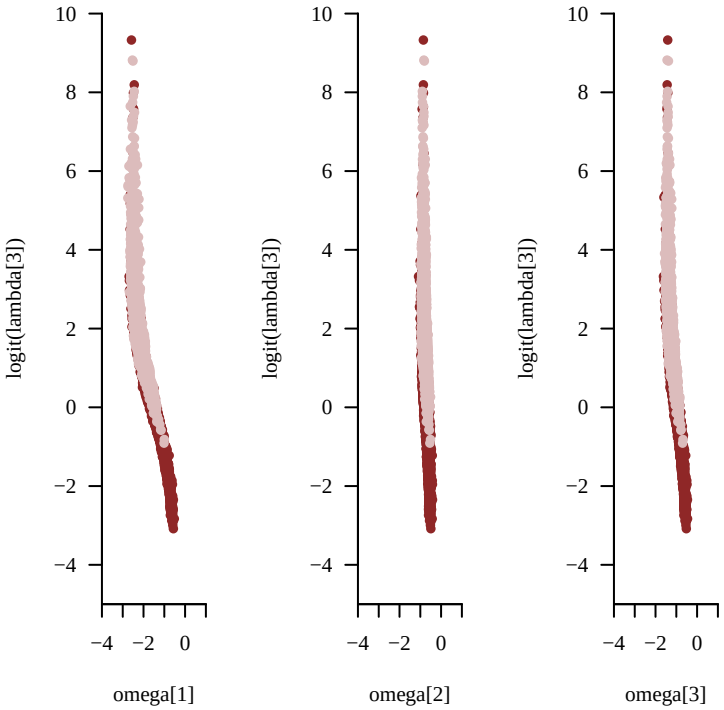
```
samples2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples2,
                                       c('zeta_free',
                                         'tau_free',
                                         'lambda'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
zeta_free[1]:
  Chain 2: hat{ESS} (94.414) is smaller than desired (100).

zeta_free[2]:
  Chain 2: hat{ESS} (93.271) is smaller than desired (100).

zeta_free[3]:
  Chain 2: hat{ESS} (93.245) is smaller than desired (100).

zeta_free[7]:
  Chain 2: Right tail hat{xi} (0.462) exceeds 0.25.
  Chain 2: hat{ESS} (79.439) is smaller than desired (100).

zeta_free[8]:
  Chain 2: Right tail hat{xi} (0.485) exceeds 0.25.
  Chain 2: hat{ESS} (79.507) is smaller than desired (100).

zeta_free[9]:
  Chain 2: Right tail hat{xi} (0.454) exceeds 0.25.
  Chain 2: hat{ESS} (79.806) is smaller than desired (100).


Large tail hat{xi}s suggest that the expectand might not be
sufficiently integrable.

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

Comparing the inferences for $\omega_{nj''}$ and $\lambda_{k(j')}$ between `ng1.stan` and `ng2.stan`, we see more expansive exploration in the latter. The divergences really were indicating a serious problem!

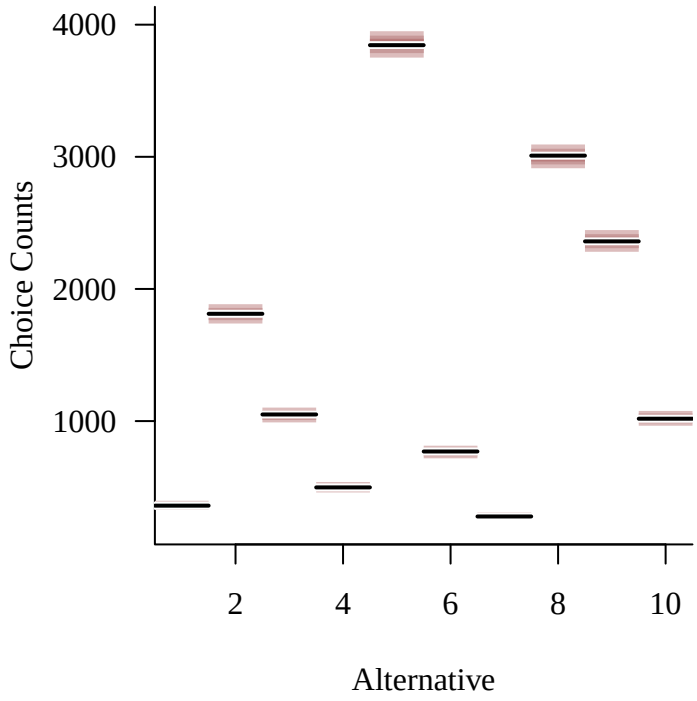
```
logit <- function(p) {
  log(p / (1 - p))
```

```
}

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

k <- 1
alt_names <- sapply(which(data$nest == k),
                    function(j) paste0('omega[', j, ']'))

for (name in alt_names) {
  plot(samples2[[name]], logit(samples2[[lambda_names[k]]]),
       cex=1, pch=16, col=util$c_dark,
       xlim=c(-4, 1), xlab=name,
       ylim=c(-5, 10), ylab='logit(lambda[3])')

  points(samples1[[name]], logit(samples1[[lambda_names[k]]]),
       cex=1, pch=16, col=util$c_light)
}
```



With reasonably accurate posterior quantification we can examine the posterior retrodictive behavior for the nested Gumbel model. Unsurprisingly the aggregate behavior doesn't show any signs of retrodictive tension.

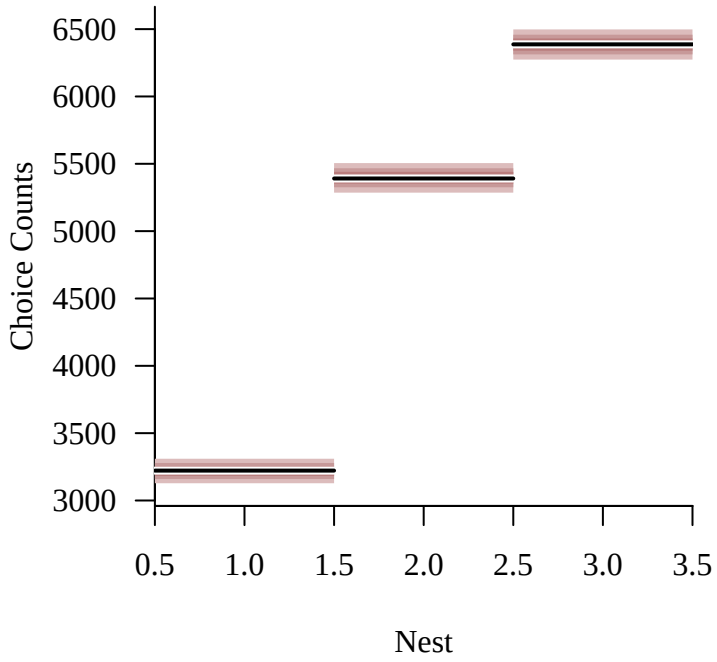
```
par(mfrow=c(1, 1), mar=c(5, 5, 1, 1))

names <- sapply(1:data$J,
                function(j)
                paste0('agg_choice_counts_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(
  samples2, names,
  baseline_values=agg_choice_counts,
  xlab="Alternative",
  ylab="Choice Counts"
)
```



Moreover, the retrodictive performance within each nest is also excellent.

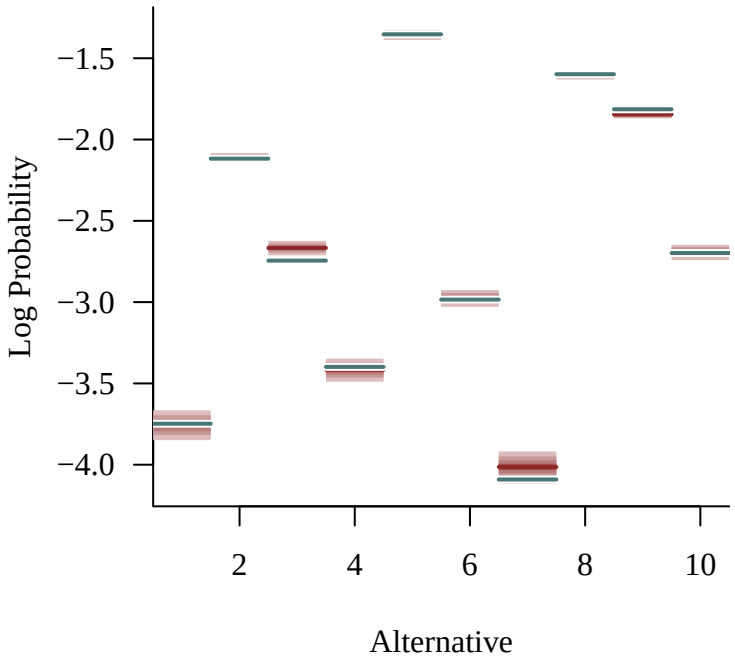
```
par(mfrow=c(1, 1), mar = c(5, 5, 2, 1))

names <- sapply(1:data$K,
                function(k)
                paste0('nest_choice_counts_pred[', k, ']'))
util$plot_disc_pushforward_quantiles(
  samples2, names,
  baseline_values=nest_choice_counts,
```

```
  xlab="Nest",
  ylab="Choice Counts"
)
```



As with those derived from the independent Gumbel model, the inferred choice probabilities derived from the nested Gumbel model are consistent with the true choice probabilities.

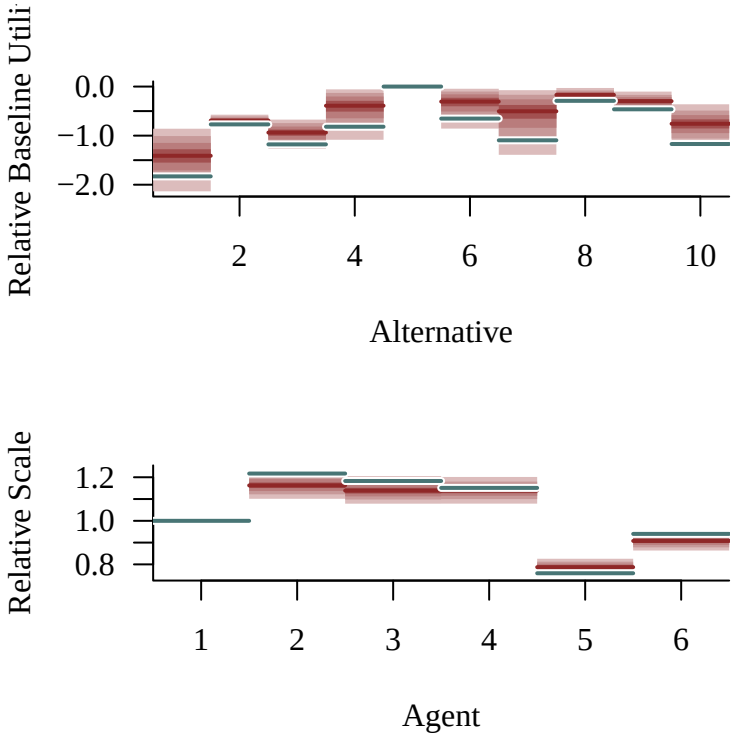
```
par(mfrow=c(1, 1), mar = c(5, 4, 2, 1))

names <- sapply(1:data$J, function(j) paste0('log_probs[1,', j, ']'))
util$plot_disc_pushforward_quantiles(samples2, names,
                                     baseline=log_prob_true[1,],
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Log Probability")
```



Now, however, we also recover the true baseline utilities and agent scales.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J,
                function(j)
                paste0('omega[', j, ']'))
util$plot_disc_pushforward_quantiles(
   samples2, names,
   baseline_values=omega_true,
   baseline_col=util$c_mid_teal,
   xlab="Alternative",
   ylab="Relative Baseline Utility"
)

tau_true <- sigma_true / sigma_true[1]

names <- sapply(1:data$N,
                function(n)
                paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(
  samples2, names,
```

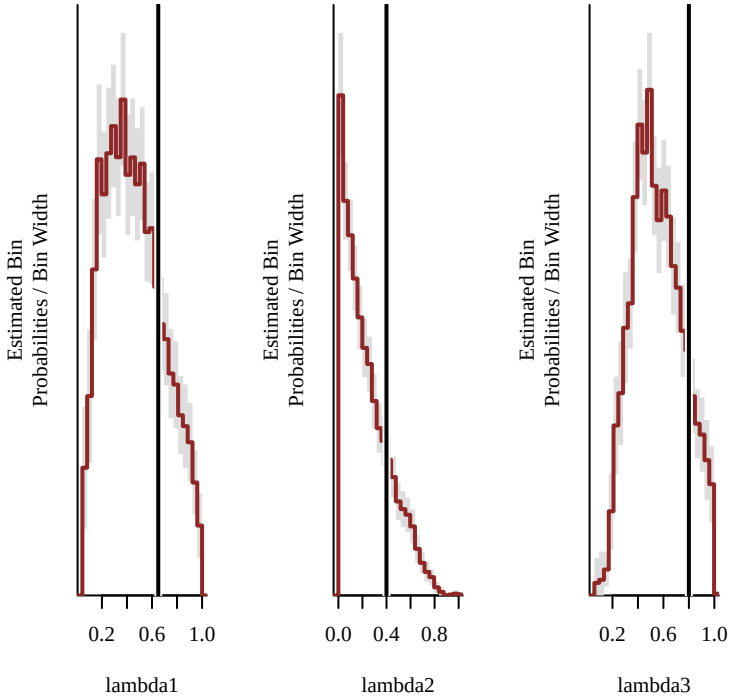
```
  baseline_values=tau_true,
  baseline_col=util$c_mid_teal,
  xlab="Agent",
  ylab="Relative Scale"
)
```



The posterior inferences for the intra-nest coupling parameters are also consistent with the simulation data generating process, but the uncertainties are massive. Indeed these large uncertainties amplify the degeneracy between the relative baseline utilities and the intra-nest coupling parameters, which is why the reparameterization was so critical.

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

util$plot_expectand_pushforward(samples2[['lambda[1]']], 25,
                                display_name='lambda1',
                                baseline=lambda_true[1])

util$plot_expectand_pushforward(samples2[['lambda[2]']], 25,
                                display_name='lambda2',
                                baseline=lambda_true[2])
```

```
util$plot_expectand_pushforward(samples2[['lambda[3]']], 25,
                                display_name='lambda3',
                                baseline=lambda_true[3])
```



Reflecting for a second, these large uncertainties should not have been surprising. The configuration of the λ_k determine substitution patterns, but with observations spanning only a single choice subset we can't learn much about those substitution patterns. Consequently there's not much these observations can do to constrain the λ_k.

4.4.2 Nested Gumbel Substitution Patterns

In order to evaluate how well the independent and nested Gumbel models capture the true substitution patterns, and generalize from one choice subset to another, we need to observe different choice subsets. The more the nest occupancies vary across the choice subsets, the better we'll be able to resolve the substitution patterns.

4.4.2.1 Simulate Data

Let's use the same nesting structure.

```
K <- 3
nests <- c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3)

lambda_true <- c(0.65, 0.4, 0.8)
```

Because the size of the choice subsets can vary we'll need to concatenate all of the choice subsets together like we did previously. Here we'll generate seven choice subsets randomly and append two choice subsets with very specific alternatives.

```
set.seed(24148538)

S <- 9
subset_alts <- lapply(1:(S - 2),
                      function(s)
                      sample(1:J, sample((J - 5):(J - 1), 1),
                             replace=FALSE))
subset_alts[[8]] <- c(1, 2, 3, 4,    6, 7, 8,    10)
subset_alts[[9]] <- c(1, 2, 3, 4, 5, 6, 7, 8,    10)

to_subset_idx <-
  lapply(subset_alts,
         function(alts) sapply(1:J,
                               function(j) match(j, alts)))
```

```
SJ <- 0
alts <- c()
subset_start <- c()
subset_end <- c()

for (s in 1:S) {
  subset_start <- c(subset_start, SJ + 1)
  SJ <- SJ + length(subset_alts[[s]])
  subset_end <- c(subset_end, SJ)

  alts <- c(alts, subset_alts[[s]])
}
```

To avoid being overwhelmed by all of the possibilities, let's investigate the substitution patterns of only three alternatives.

```
j1 <- 4
j2 <- 7
j3 <- 2
```

```
nests[c(j1, j2, j3)]
```

```
[1] 2 2 1
```

Because j_1 and j_2 are in the same nest, the ratios

$$\frac{q_{nj_1}}{q_{nj_2}}$$

should be consist across all of the choice subsets that include both alternatives. On the other hand, because j_1 and j_3 are in the different nests we expect that the ratios

$$\frac{q_{nj_1}}{q_{nj_2}}$$

will vary across choice subsets.

For example, when moving from the eight choice subset to the ninth choice subset we add an alternative to the second nest. This should preferentially leech probability from the other alternatives in that nest relative to the alternatives in other nests. Specifically this should result in

$$\frac{q_{nj_1}}{q_{nj_3}}$$

decreasing in the ninth choice subset relative to the eighth choice subset.

With all of the details set we can simulate data.

```
simu <- stan(file="stan_programs/simu_ng2.stan",
             algorithm="Fixed_param", seed=8438338,
             data=list('M' = M, 'N' = N, 'J' = J,
                       'K' = K, 'nests' = nests,
                       'S' = S, 'SJ' = SJ,
                       'subset_start' = subset_start,
                       'subset_end' = subset_end,
                       'alts' = alts,
                       'lambda' = lambda_true),
             warmup=0, iter=1, chains=1, refresh=0)

simu_fit <- extract(simu)
```

```
subset <- simu_fit$subset[1,]
agent <- simu_fit$agent[1,]
choice <- simu_fit$choice[1,]

choice_counts <- matrix(NA, N, SJ)
for (n in 1:N) {
  for (s in 1:S) {
    JJ <- subset_end[s] - subset_start[s] + 1
    choice_counts[n, subset_start[s]:subset_end[s]] <-
      hist(choice[agent == n & subset == s],
           seq(0.5, JJ + 0.5, 1),
           plot=FALSE)$counts
  }
}
```

In addition to saving the true configuration of the data generating process, we can also calculate the true choice probabilities across the full choice set for a particular agent.

```
mu_true <- simu_fit$mu[1,]
sigma_true <- simu_fit$sigma[1,]
```

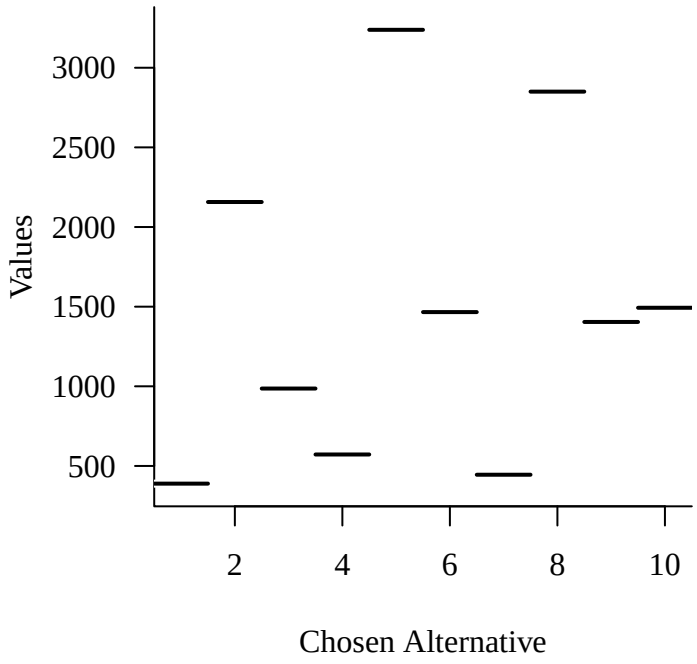
```
n_pred <- 5

log_probs_true <- rep(NA, J)
nest_lses <- rep(NA, K)
log_marg_probs <- rep(NA, K)

for (k in 1:K) {
  nest_alts <- which(nests == k)
  nest_lses[k] <-
    log(sum(exp(  mu_true[nest_alts] /
                (sigma_true[n_pred] * lambda_true[k]) )))
}

x <- lambda_true * nest_lses
log_marg_probs <- x - log(sum(exp(x)));

for (k in 1:K) {
  nest_alts <- which(nests == k)
  log_probs_true[nest_alts] <-
    log_marg_probs[k] +
    mu_true[nest_alts] /
```

```
      (sigma_true[n_pred] * lambda_true[k]) -
    nest_lses[k];
}
```

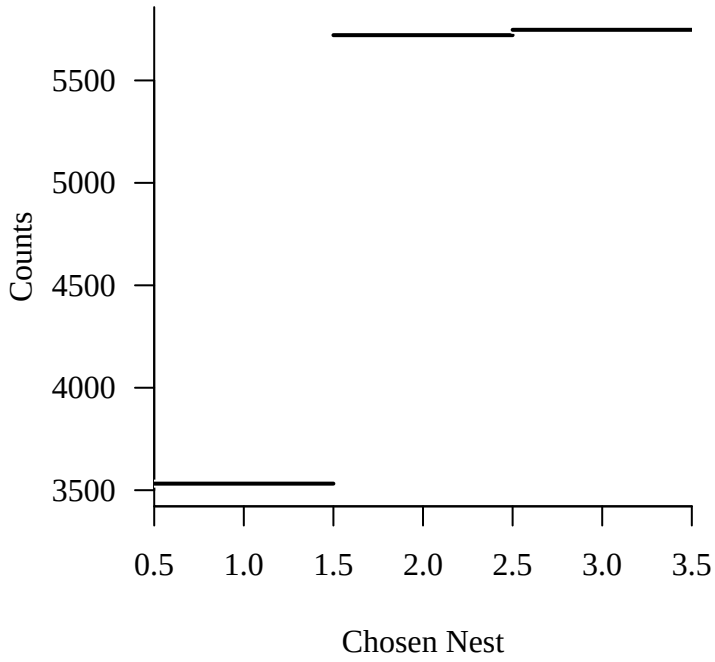
4.4.2.2 Explore Data

By aggregating over all of the choice subsets, we can use the same histogram summaries that we used previously. The only subtlety is making sure to aggregate using the *global* alternative indices.

```
agg_choice_counts <- rep(0, J)

for (s in 1:S) {
  subset_agg_choice_counts <-
    colSums(choice_counts[,subset_start[s]:subset_end[s]])

  subset_alts <- alts[subset_start[s]:subset_end[s]]
  agg_choice_counts[subset_alts] <-
    agg_choice_counts[subset_alts] + subset_agg_choice_counts
}

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

util$plot_disc_vals(agg_choice_counts,
                    xlab='Chosen Alternative')
```

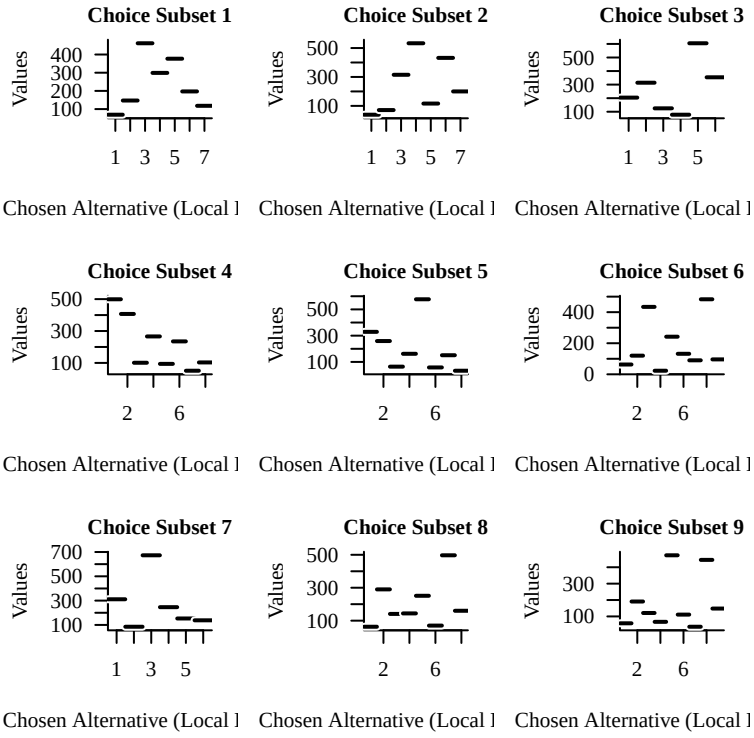



```
par(mfrow=c(1, 1), mar = c(5, 5, 2, 1))

nest_choice_counts <-
  sapply(1:K,
         function(k)
         sum(agg_choice_counts[which(data$nests == k)]))

util$plot_disc_vals(nest_choice_counts,
                    xlab='Chosen Nest',
                    ylab='Counts')
```



Now, however, we can also stratify these summaries by the individual choice subsets. Here it's a bit more convenient to use the local alternative indexing within each choice subset to avoid too many gaps in the histograms.

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

for (s in 1:S) {
  subset_agg_choice_counts <-
     colSums(choice_counts[,subset_start[s]:subset_end[s]])
  util$plot_disc_vals(subset_agg_choice_counts,
                      xlab='Chosen Alternative (Local Indexing)',
                      main=paste('Choice Subset', s))
}
```



4.4.2.3 Independent Gumbel Model

The independent Gumbel model fared well when analyzing data from only a single choice set, even when that data was simulated from a more sophisticated model. Will that performance persist to data across multiple choice subsets?

If we want to be able to compare parameters across fits then we'll need to choose a consistent anchor. The ideal anchor is the most observed alternative that also appears in all of the choice subsets. Here we don't have many options so we'll anchor the only alternative that appears in all nine of the choice subsets.

```
Reduce(intersect,
       lapply(1:S,
              function(s) alts[subset_start[s]:subset_end[s]]))
```

```
[1] 10
```

```
anchor <- 10
```

At this point we have the option of fitting the data from each choice subset independently or jointly. Let's start by fitting them independently.

Note that we have to convert the anchor from global alternative indexing to local alternative indexing in each fit.

```
samples <- list()

for (s in 1:S) {
  JJ <- subset_end[s] - subset_start[s] + 1
  subset_choice_counts <-
    choice_counts[,subset_start[s]:subset_end[s]]
  data <- list('N'=N, 'J'=JJ,
               'choice_counts'=subset_choice_counts,
               'anchor'=to_subset_idx[[s]][anchor])

  fit <- stan(file='stan_programs/ig3.stan',
              data=data, seed=8438338,
              warmup=1000, iter=2024, refresh=0)

  cat(paste('Analyzing choice subset', s, ':\n'))

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

  cat('\n')

  samples[[s]] <- util$extract_expectand_vals(fit)
  base_samples <- util$filter_expectands(samples[[s]],
                                         c('omega_free',
                                           'tau_free'),
                                         TRUE)
  util$check_all_expectand_diagnostics(base_samples)

  cat('\n\n')
}
```

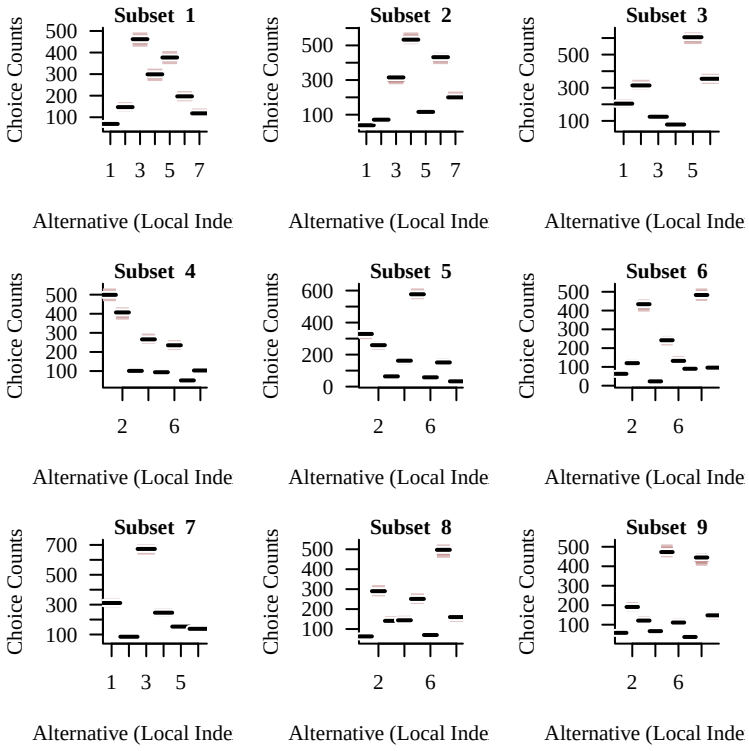
```
Analyzing choice subset 1 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 2 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
```

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

Analyzing choice subset 3 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 4 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 5 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 6 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 7 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 8 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.

Analyzing choice subset 9 :
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

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

Each fit exhibits excellent retrodictive performance individually.

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

for (s in 1:S) {
  JJ <- subset_end[s] - subset_start[s] + 1
  subset_agg_choice_counts <-
    colSums(choice_counts[,subset_start[s]:subset_end[s]])

  names <- sapply(1:JJ,
                  function(j)
                  paste0('agg_choice_counts_pred[', j, ']'))
  util$plot_disc_pushforward_quantiles(
    samples[[s]], names,
    baseline_values=subset_agg_choice_counts,
    xlab="Alternative (Local Indexing)",
    ylab="Choice Counts",
    main=paste('Subset ', s)
  )
}
```



That said, the relative baseline utilities needed to achieve these retrodictive performances is all over the place! The inferred behaviors are not only inconsistent across fits but also largely inconsistent with the true configuration of the simulation data generating process.

```
omega_true <- (mu_true - mu_true[anchor]) / sigma_true[1]

js <- c(j1, j2, j3)
Bs <- c(75, 75, 50)
flims <- list(c(-2, 0.5), c(-3.5, 0.5), c(-0.25, 1.5))
ylims <- list(c(0, 5), c(0, 5), c(0, 5))

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

for (i in seq_along(js)) {
  valid_subsets <-
    na.omit(sapply(1:S,
           function(s)
           ifelse(is.na(to_subset_idx[[s]][js[i]]), NA, s)))

  sv <- valid_subsets[1]
  name <- paste0('omega[', to_subset_idx[[sv]][js[i]], ']')
```

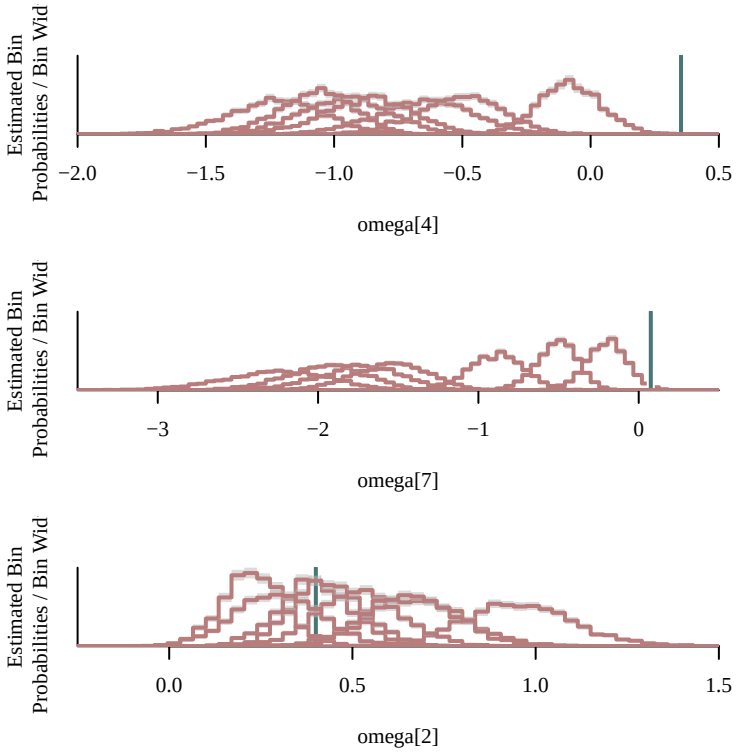
```
  display_name <- paste0('omega[', js[i], ']')
  util$plot_expectand_pushforward(samples[[sv]][[name]],
                                  Bs[i], flim=flims[[i]],
                                  ylim=ylims[[i]],
                                  display_name=display_name,
                                  col=util$c_mid,
                                  baseline=omega_true[js[i]],
                                  baseline_col=util$c_mid_teal)

  for (sv in valid_subsets[-1]) {
    name <- paste0('omega[', to_subset_idx[[sv]][js[i]], ']')
    util$plot_expectand_pushforward(samples[[sv]][[name]],
                                    Bs[i], flim=flims[[i]],
                                    col=util$c_mid,
                                    border="#BBBBBB88",
                                    add=TRUE)
  }
}
```

```
Warning in util$plot_expectand_pushforward(samples[[sv]][[name]], Bs[i], : 1
value (0.0%) fell below the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples[[sv]][[name]], Bs[i], : 0
value (0.0%) fell above the histogram binning.
```

```
Warning in util$plot_expectand_pushforward(samples[[sv]][[name]], Bs[i], : 5
values (0.1%) fell below the histogram binning.
```

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



While the independent Gumbel model can reproduce the observed behaviors within each choice subset individually, it cannot reproduce those all of those behaviors at the same time. This inconsistency across choice subsets then results in poor inferences and predictions for new choice subsets.

We can force the configuration of the independent Gumbel model to be consistent across choice subsets but analyzing all of the data jointly.

```
data <- list('N'=N, 'J'=J,
             'S'=S, 'SJ'=SJ,
             'subset_start'=subset_start,
             'subset_end'=subset_end,
             'alts'=alts,
             'choice_counts'=choice_counts,
             'anchor'=anchor)
```

To study particular substitution patterns we can make predictions for the choice behavior in unobserved choice subsets. Here we'll consider predictions for the full choice set.

```
data$n_pred <- n_pred
data$J_pred <- data$J
data$pred_alts <- 1:data$J
```

How does the joint model fare?

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

Well there don't appear to be any 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.
```

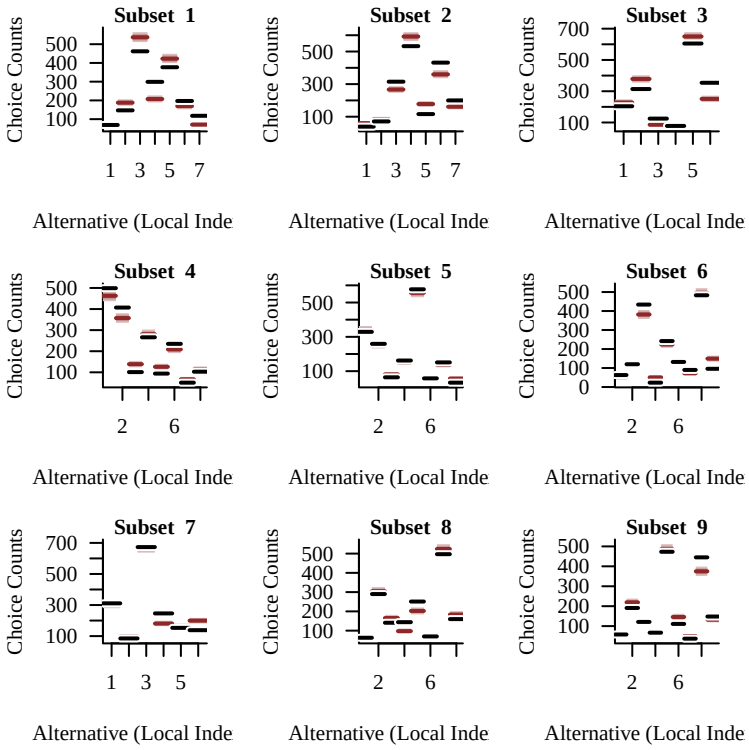
```
samples_ig <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples_ig,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

Unfortunately the retrodictive performance is not so well behaved. When forcing a consistent configuration of the independent Gumbel model the retrodictive performance in most of the choice subsets is terrible.

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

for (s in 1:S) {
  subset_agg_choice_counts <-
    colSums(data$choice_counts[,subset_start[s]:subset_end[s]])
  names <- sapply(subset_start[s]:subset_end[s],
                  function(j)
                  paste0('agg_choice_counts_pred[', j, ']'))
  util$plot_disc_pushforward_quantiles(
    samples_ig, names,
    baseline_values=subset_agg_choice_counts,
    xlab="Alternative (Local Indexing)",
```

```
    ylab="Choice Counts",
    main=paste('Subset ', s)
  )
}
```



Because the independent Gumbel model is not sophisticated enough to accommodate the substitution patterns of the true data generating process it performs poorly in one way or another when analyzing data from multiple choice subsets. The only way to ensure consistent inferences and performant predictions is to use a sufficiently flexible model.

Because we simulated from a nested Gumbel model, analyzing the simulated data with a nested Gumbel model should should yield better results. Fortunately we don't have to speculate: we can study this directly.

```
data$K <- K
data$nests <- nests

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

We don't see any signs of computational problems, even though we're using the nominal parameterization of the nested Gumbel model that exhibits problems before. This suggests that analyzing data from multiple choice subsets is able to inform the intra-nest coupling parameters much more strongly, although we'll have to examine those posterior inferences to confirm.

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

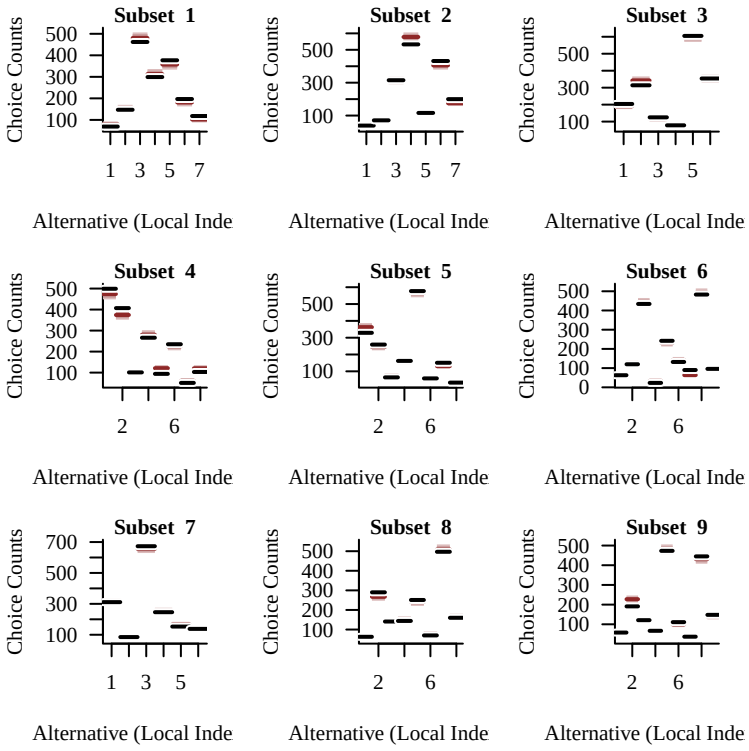
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples_ng <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples_ng,
                                       c('omega_free',
                                         'tau_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

More good news: the retrodictive performance is good across all of the choice subsets.

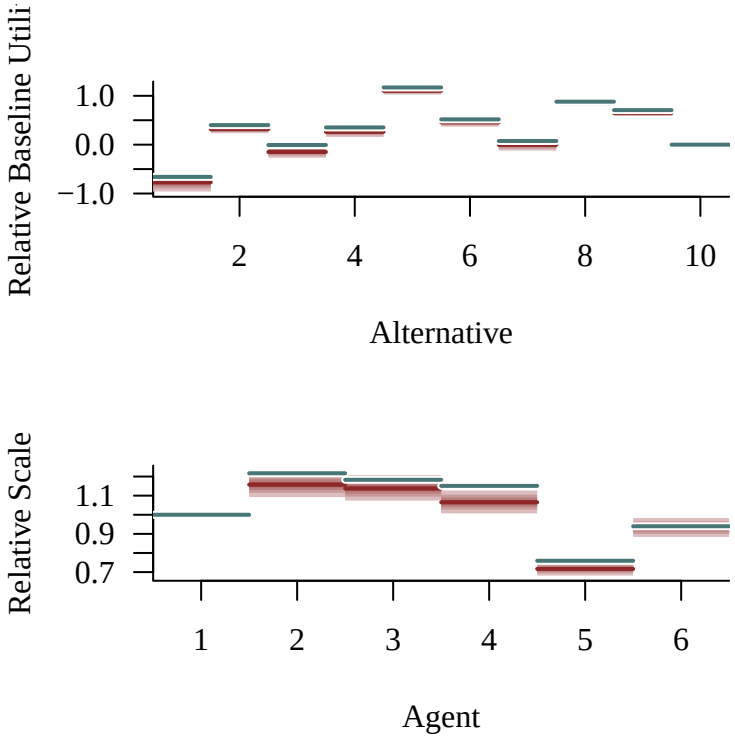
```
par(mfrow=c(3, 3), mar=c(5, 5, 1, 1))

for (s in 1:S) {
  subset_agg_choice_counts <-
    colSums(data$choice_counts[,subset_start[s]:subset_end[s]])
  names <- sapply(subset_start[s]:subset_end[s],
                  function(j)
                  paste0('agg_choice_counts_pred[', j, ']'))
  util$plot_disc_pushforward_quantiles(
    samples_ng, names,
    baseline_values=subset_agg_choice_counts,
    xlab="Alternative (Local Indexing)",
    ylab="Choice Counts",
    main=paste('Subset ', s)
  )
}
```



This broad retrodictive performance is matched by inferences that concentrate around the configuration of the simulation data generating process.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J, function(j) paste0('omega[', j, ']'))
util$plot_disc_pushforward_quantiles(samples_ng, names,
                                     baseline_values=omega_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Relative Baseline Utility")


tau_true <- sigma_true / sigma_true[1]

names <- sapply(1:data$N, function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(samples_ng, names,
                                     baseline_values=tau_true,
                                     baseline_col=util$c_mid_teal,
```

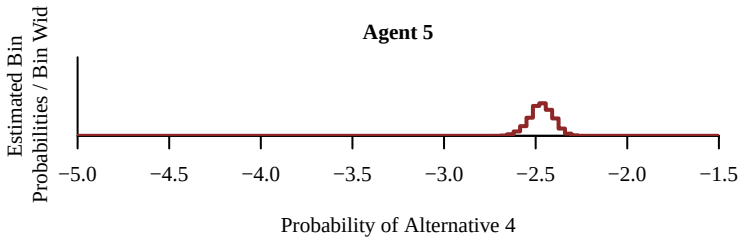
```
                                    xlab="Agent",
                                    ylab="Relative Scale")
```

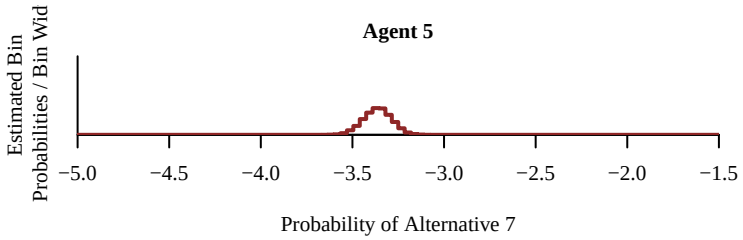


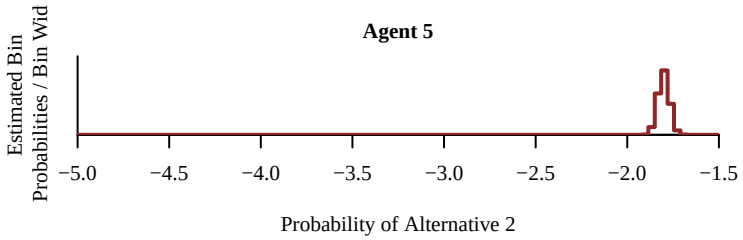
Confident that our model is learning the behavior of the true data generating process, let's investigate some particular substitution patterns for the fifth agent. We begin by examining the inferred choice probabilities for our three distinguished alternatives in the eight choice subset.

```
B <- 100
flim=c(-5, -1.5)

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

for (j in c(j1, j2, j3)) {
  s <- 8
  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j]
  name <- paste0('log_probs[', n_pred, ',', idx, ']')
  display_name <- paste0('Probability of Alternative ', j)

  util$plot_expectand_pushforward(samples_ng[[name]],
                                  B, flim=flim,
```

```
                                    ylim=c(0, 15),
                                    display_name=display_name,
                                    col=util$c_dark,
                                    main=paste0('Agent ', n_pred))
}
```







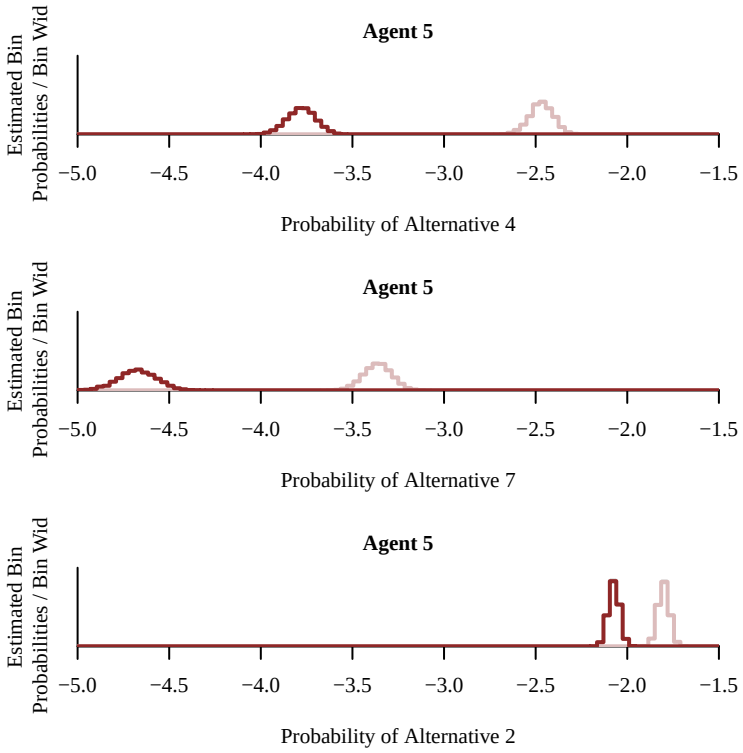
The ninth choice subset is given by adding a new alternative to the eight choice subset. Any choice probability allocated to this new alternative has to taken from the existing alternatives. Indeed we see that the choice probabilities for all three distinguished alternatives decrease.

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

for (j in c(j1, j2, j3)) {
  s <- 8
  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j]
  name <- paste0('log_probs[', n_pred, ',', idx, ']')
  display_name <- paste0('Probability of Alternative ', j)

  util$plot_expectand_pushforward(samples_ng[[name]],
                                  B, flim=flim,
                                  ylim=c(0, 15),
```

```
                                    display_name=display_name,
                                    col=util$c_light,
                                    main=paste0('Agent ', n_pred))

  s <- 9
  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j]
  name <- paste0('log_probs[', n_pred, ',', idx, ']')

  util$plot_expectand_pushforward(samples_ng[[name]],
                                    B, flim=flim,
                                    col=util$c_dark,
                                    border="#BBBBBB88",
                                    add=TRUE)

}
```

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

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



That said, the choice probabilities don't all decrease by the same amount. In particular the choice probabilities allocated to j_1 and j_2, which are in the same nest as the new alternative, decrease more than the choice probability allocated to j_3.

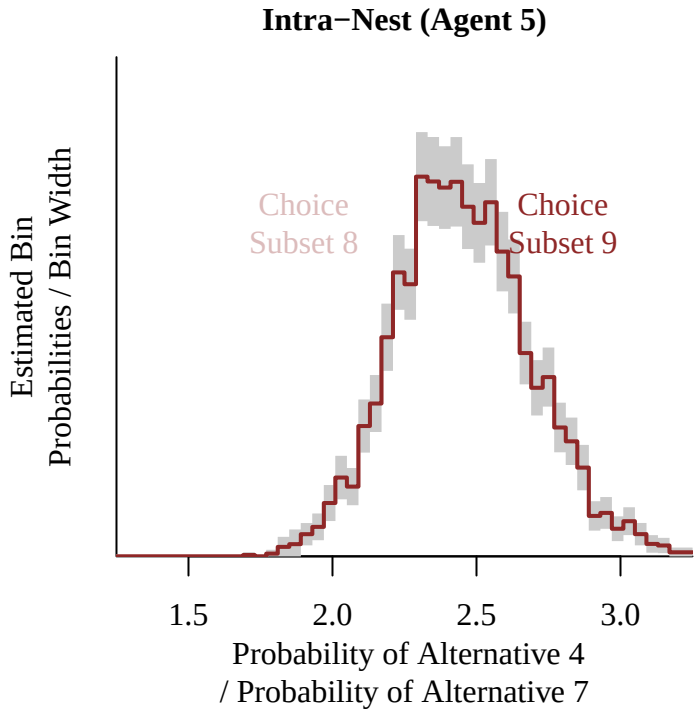
As we have seen over and over again at this point, these specific substitution patterns are a bit more evident in ratios of choice probabilities.

```
ratios <- list()

for (s in 8:9) {
  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j1]
  name1 <- paste0('log_probs[', n_pred, ',', idx, ']')

  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j2]
  name2 <- paste0('log_probs[', n_pred, ',', idx, ']')

  idx <- subset_start[s] - 1 + to_subset_idx[[s]][j3]
  name3 <- paste0('log_probs[', n_pred, ',', idx, ']')

  ratios[[paste0(s, '_12')]] <-
    util$eval_expectand_pushforward(samples_ng,
                                    function(x1, x2) exp(x1 - x2),
```

```
                                    list('x1'=name1, 'x2'=name2))

  ratios[[paste0(s, '_13')]] <-
    util$eval_expectand_pushforward(samples_ng,
                                    function(x1, x2) exp(x1 - x2),
                                    list('x1'=name1, 'x2'=name3))
}
```

Because alternatives j_1 and j_2 are in the same nest, the ratio of their choice probabilities are consistent across choice subsets. In fact they are exactly equal, with the corresponding marginal posterior distributions perfectly overlapping.

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

B <- 50
flim <- c(1.25, 3.25)

display_name <- paste('  Probability of Alternative', j1, '\n/',
                      'Probability of Alternative', j2)
main <- paste0("Intra-Nest (Agent ", n_pred, ")")
util$plot_expectand_pushforward(ratios[['8_12']],
                                B, flim=flim,
                                ylim=c(0, 2.25),
                                display_name=display_name,
                                col=util$c_light,
                                main=main)
text(1.9, 1.5, 'Choice\nSubset 8', col=util$c_light)

util$plot_expectand_pushforward(ratios[['9_12']],
                                B, flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)
text(2.8, 1.5, 'Choice\nSubset 9', col=util$c_dark)
```



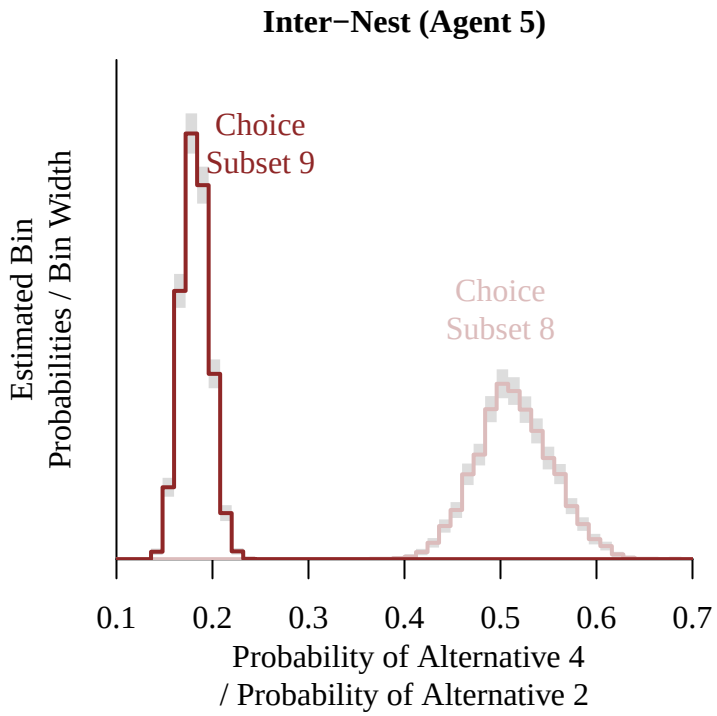
The introduction of the new alternative should siphon more choice probability from alternatives in the same nest, including j_1, than alternatives in other nests, including j_3. Consequently

$$\frac{q_{5j_1}}{q_{5j_3}}$$

should decrease when we move to the larger choice subset. Indeed this is exactly the behavior that the nested Gumbel model infers.

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

B <- 50
flim <- c(0.1, 0.7)

display_name <- paste('  Probability of Alternative', j1, '\n/',
                      'Probability of Alternative', j3)
main <- paste0("Inter-Nest (Agent ", n_pred, ")")
util$plot_expectand_pushforward(ratios[['8_13']],
                                B, flim=flim,
                                ylim=c(0, 30),
                                display_name=display_name,
                                col=util$c_light,
                                main=main)
```

```
text(0.50, 15, 'Choice\nSubset 8', col=util$c_light)

util$plot_expectand_pushforward(ratios[['9_13']],
                                B, flim=flim,
                                col=util$c_dark,
                                border="#BBBBBB88",
                                add=TRUE)
text(0.25, 25, 'Choice\nSubset 9', col=util$c_dark)
```



5 Mixed Discrete Choice Models

Probabilistic utility models that both capture sophisticated couplings and lead to clean, analytic results are few and far between. Often a more productive route to constructing useful discrete choice models is to *modify* an existing discrete choice model.

For example given a parameterized **base utility model**,

$$p(u_{n1}, \ldots, u_{nJ} \mid \phi)$$

and a **mixture kernel**

$$p(\phi \mid \psi),$$

we can derive a new utility model as a continuous mixture,

$$p(u_{n1}, \ldots, u_{nJ} \mid \psi) = \int \mathrm{d}\phi \, p(\phi \mid \psi) \, p(u_{n1}, \ldots, u_{nJ} \mid \phi).$$

The mixed probabilistic utility model defines new choice probabilities that will, in general, exhibit different substitution patterns than the choice probabilities derived from the base model.

If the base utility model and mixture kernel are sufficiently well-behaved, then we can freely exchange the mixture integral with other integrals and derivatives. This allow us to define new discrete choice models by directly mixing the joint cumulative distribution function,

$$\begin{aligned}
\Pi(u_{n1}, \ldots, u_{nJ} \mid \psi) &= \left[\prod_{j=1}^{J} \int_{-\infty}^{u_{nj}} \mathrm{d}x_j \right] \, p(x_{n1}, \ldots, x_{nJ} \mid \psi) \\
&= \left[\prod_{j=1}^{J} \int_{-\infty}^{u_{nj}} \mathrm{d}x_j \right] \, \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \Pi(x_1, \ldots, x_J \mid \phi) \\
&= \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \left[\prod_{j=1}^{J} \int_{-\infty}^{u_{nj}} \mathrm{d}x_j \right] \, \Pi(x_1, \ldots, x_J \mid \phi) \\
&= \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \Pi(u_{n1}, \ldots, u_{nJ} \mid \phi).
\end{aligned}$$

By appropriately engineering the mixture kernel $p(\phi \mid \psi)$ we can, in theory, ensure that the resulting mixture model exhibits desired substitution patterns that the base model does not. Actually designing useful mixture kernels in practice, however, is by no means trivial.

In the econometric and marketing communities, discrete choice models derived from continuous mixtures are often referred to as **mixed models**. The term **marketing mixed model**, or even just **MMM** is also common, perhaps only for its alliterative appeal.

5.1 Mixed Choice Probabilities

Because choice probabilities are defined through a sequence of derivatives and integrals of the joint cumulative distribution function, deriving choice probabilities is much easier when we can be a bit cavalier about exchanging these operations with the mixture integral. In practice we are almost always working with base utility models and mixture kernels that are sufficiently well-behaved to allow this, and I will assume as much moving forwards.

In this case we can write not only

$$\Pi(u_{n1}, \ldots, u_{nJ} \mid \psi) = \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \Pi(u_{n1}, \ldots, u_{nJ} \mid \phi)$$

but also

$$\frac{\partial \Pi}{\partial x_j}(u_{n1}, \ldots, u_{nJ} \mid \psi) = \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{\partial \Pi}{\partial x_j}(u_{n1}, \ldots, u_{nJ} \mid \phi).$$

This, in turn, allows us to write the new choice probabilities as continuous mixtures of the base choice probabilities,

$$\begin{aligned}
q_{nj}(\psi) &= \pi(\mathsf{u}_{nj}) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u \mid \psi) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u \mid \phi) \\
&= \int \mathrm{d}\phi \, p(\phi \mid \psi) \int_{-\infty}^{+\infty} \mathrm{d}u \, \frac{\partial \Pi}{\partial x_j}(u, \ldots, u \mid \phi) \\
&= \int \mathrm{d}\phi \, p(\phi \mid \psi) \, q_{nj}(\phi).
\end{aligned}$$

To avoid any confusion I will refer to the $q_{nj}(\psi)$ as **marginal choice probabilities** and the $q_{nj}(\phi)$ as **base choice probabilities**. Similarly I will refer to ψ as a global parameter and ϕ as an auxiliary parameter.

For example, given an initial independent Gumbel model with the base choice probabilities,

$$q_{nj}(\mu_{n1}, \ldots, \mu_{nJ}, \sigma_n) = \frac{e^{\frac{\mu_{nj}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}}$$

we could define a new discrete choice model with the "mixed" choice probabilities

$$\begin{aligned}
q_{nj}(\psi) &= \int \mathrm{d}\mu_{n1} \ldots \mathrm{d}\mu_{nJ} \, p(\mu_{n1}, \ldots, \mu_{nJ} \mid \psi) \, q_{nj}(\mu_{n1}, \ldots, \mu_{nJ}, \sigma_n) \\
&= \int \mathrm{d}\mu_{n1} \ldots \mathrm{d}\mu_{nJ} \, p(\mu_{n1}, \ldots, \mu_{nJ} \mid \psi) \, \frac{e^{\frac{\mu_{nj}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}}.
\end{aligned}$$

That said we can also define mixed models with respect to *any* of the parameters. For instance we could consider mixtures over the agent scales σ_n, scaled baseline utilities $\nu_{nj} = \mu_{nj} / \sigma_n$, relative baseline utilities $\delta_{nj} = \mu_{nj} - \mu_{nj'}$, and so on.

5.2 Mixed Substitution Patterns

One of the main benefits of mixing a base discrete choice model is that the procedure tends to obstruct simpler substitution patterns exhibited by that model, such as independence from

irrelevant alternatives. In this section we'll investigate how many simpler substitution patterns are mechanically obstructed by the introduction of a continuous mixture.

For an explicit example we will consider mixing the baseline utilities in the independent Gumbel model, resulting in the choice probabilities

$$q_{nj} = \int \mathrm{d}\mu_{n1} \ldots \mathrm{d}\mu_{nJ} \, p(\mu_{n1}, \ldots, \mu_{nJ} \mid \psi) \, e^{\frac{\mu_{nj}}{\sigma_n}} \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1}.$$

While mathematically straightforward, this in a bit ungainly in terms of notation. To allow for more compact equations let's introduce the notation

$$\mu_n = (\mu_{n1}, \ldots, \mu_{nJ})$$

and

$$\mathrm{d}\mu_n = \mathrm{d}\mu_{n1} \ldots \mathrm{d}\mu_{nJ}$$

so that we can instead write

$$q_{nj} = \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, e^{\frac{\mu_{nj}}{\sigma_n}} \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1}.$$

Discrete choice models derived from mixing baseline utilities are common referred to as **random utility models**, especially in the econometrics literature.

5.2.1 Ratios of Choice Probabilities

Let's first consider the substitution patterns quantified by ratios of choice probabilities. Mechanically, continuous mixtures obstruct the cancellations that can make ratios of choice probabilities derived from the base model particularly simple. More casually, the ratio of integrals is not equal to the integral the ratio.

The independent Gumbel model, for instance, exhibits independence from irrelevant alternatives because every ratio of choice probabilities is independent of the other alternatives,

$$\frac{q_{nj_1}}{q_{nj_2}} = \frac{e^{\frac{\mu_{nj_1}}{\sigma_n}}}{e^{\frac{\mu_{nj_2}}{\sigma_n}}} = \exp\left(\frac{\mu_{nj_1} - \mu_{nj_2}}{\sigma_n} \right).$$

When we mix the baseline utilities, however, ratios of choice probabilities become

$$\frac{q_{nj_1}}{q_{nj_2}} = \frac{\int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, e^{\frac{\mu_{nj_1}}{\sigma_n}} \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1}}{\int \mu_n \, p(\mu_n \mid \psi) \, e^{\frac{\mu_{nj_2}}{\sigma_n}} \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1}}.$$

The common denominators,

$$\left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1},$$

no longer cancel because they cannot be factored out of the integrals,

$$\begin{aligned} &\int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, e^{\frac{\mu_{nj_1}}{\sigma_n}} \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1} \\ &\quad \neq \left[\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}} \right]^{-1} \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, e^{\frac{\mu_{nj_1}}{\sigma_n}}. \end{aligned}$$

Specifically, the ratio of mixed choice probabilities is *not* equal to the mixed ratio of independent Gumbel choice probabilities,

$$\frac{q_{nj_1}}{q_{nj_2}} \neq \int \mu_n \, p(\mu_n \mid \psi) \, \frac{e^{\frac{\mu_{nj_1}}{\sigma_n}}}{e^{\frac{\mu_{nj_2}}{\sigma_n}}}.$$

5.2.2 Elasticities

A similar obstruction arises when computing elasticities.

Assuming all of the niceties needed to freely exchange integrals and derivates, the sensitivities of a mixed discrete choice model can be written as

$$\frac{\partial q_{nj}}{\partial z}(\psi) = \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{\partial q_{nj}}{\partial z}(\phi).$$

The corresponding elasticities then become

$$\begin{aligned} \frac{z}{q_{nj}(\psi)} \frac{\partial q_{nj}}{\partial z}(\psi) &= \frac{z}{q_{nj}(\psi)} \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{\partial q_{nj}}{\partial z}(\phi) \\ &= z \int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{1}{q_{nj}(\psi)} \frac{\partial q_{nj}}{\partial z}(\phi) \\ &= z \, \frac{\int \mathrm{d}\phi \, p(\phi \mid \psi) \, \frac{\partial q_{nj}}{\partial z}(\phi)}{\int \mathrm{d}\phi \, p(\phi \mid \psi) \, q_{nj}(\phi)}. \end{aligned}$$

In the second equation the marginal choice probability $q_{nj}(\psi)$ does not cancel with the base choice probability $q_{nj}(\phi)$. The last equation is written entirely in terms of base choice probabilities, but the terms in the numerator and denominator are blocked from canceling with each other due to the integrals.

These obstructions can be a bit easier to understand in an explicit example. For instance, in an independent Gumbel model we have

$$\frac{\partial q_{nj}}{\partial z}(\mu_n) = -\frac{q_{nj}(\mu_n)}{\sigma_n} \sum_{j'=1}^{J} q_{nj'}(\mu_n) \frac{\partial \mu_{nj'}}{\partial z}.$$

Here the choice probability $q_{nj}(\mu_n)$ cancels when evaluating the corresponding elasticity,

$$\frac{z}{q_{nj}(\mu_n)} \frac{\partial q_{nj}}{\partial z}(\mu_n) = -\frac{1}{\sigma_n} \sum_{j'=1}^{J} q_{nj'}(\mu_n) \frac{\partial \mu_{nj'}}{\partial z}.$$

Because of this cancellation, the elasticities are the same for all alternatives j.

If we mix the baseline utilities, however, then the elasticities become

$$\begin{aligned}
&\frac{z}{q_{nj}(\psi)} \frac{\partial q_{nj}}{\partial z}(\psi) \\
&\quad = \frac{z}{q_{nj}(\psi)} \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, \frac{\partial q_{nj}}{\partial z}(\mu_n) \\
&\quad = -\frac{z}{\sigma_n} \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, \left[\frac{q_{nj}(\mu_n)}{q_{nj}(\psi)} \right] \sum_{j'=1}^{J} q_{nj'}(\mu_n) \frac{\partial \mu_{nj'}}{\partial z}.
\end{aligned}$$

The term

$$\frac{q_{nj}(\mu_n)}{q_{nj}(\psi)} \neq 1$$

will, in general, vary across alternatives. Consequently the sensitivities will no longer be homogeneous across all alternatives, violating independence from irrelevant alternatives.

5.2.3 Strong Coupling Makes Good Substitutes

Ideally we would be able to say a bit more about the specific substitution patterns that arise when we mix the baseline utilities of the independent Gumbel model beyond just the violation of independence of irrelevant alternatives. While making quantitative statements is difficult, we can make some qualitative remarks.

Consider two choice subsets, one consisting of J alternatives and the second given by removing the j_2th alternative from the first. The mixed choice probabilities over the initial choice set

S_1 are given by

$$\begin{aligned} q_{nj_1}^{S_1}(\psi) &= \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, \frac{e^{\frac{\mu_{nj_1}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}} \\ &= \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, \frac{1}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'} - \mu_{nj_1}}{\sigma_n}}} \\ &= \int \mathrm{d}\mu_n \, p(\mu_n \mid \psi) \, \frac{1}{\sum_{j' \neq j_2}^{J} e^{\frac{\mu_{nj'} - \mu_{nj_1}}{\sigma_n}} + e^{\frac{\mu_{nj_2} - \mu_{nj_1}}{\sigma_n}}}. \end{aligned}$$

Similarly, the mixed choice probabilities over the second choice subset S_2 are given by

$$\begin{aligned} q_{nj_1}^{S_2}(\psi) &= \int \mathrm{d}\mu_{n,\,j_2} \, p(\mu_{n,\,j_2} \mid \psi) \, \frac{e^{\frac{\mu_{nj_1}}{\sigma_n}}}{\sum_{j' \neq j_2} e^{\frac{\mu_{nj'}}{\sigma_n}}} \\ &= \int \mathrm{d}\mu_{n,\,j_2} \, p(\mu_{n,\,j_2} \mid \psi) \, \frac{1}{\sum_{j' \neq j_2} e^{\frac{\mu_{nj'} - \mu_{nj_1}}{\sigma_n}}} \end{aligned}$$

where

$$\mu_{n,\,j_2} = (\mu_{n1}, \ldots, \mu_{n(j_2-1)}, \mu_{n(j_2+1)}, \mu_{nJ}).$$

The more strongly $p(\mu_n \mid \psi)$ couples together μ_{nj_1} and μ_{nj_2}, the more the extra term

$$e^{\frac{\mu_{nj_2} - \mu_{nj_1}}{\sigma_n}}$$

will concentrate around one. This concentration then causes $q_{nj_1}^{S_1}(\psi)$ to *decrease* relative to $q_{nj_1}^{S_2}(\psi)$ faster than what we would see from other alternatives whose baseline utilities are more weakly coupled to μ_{nj_2}.

Adding j_2 to the choice set leeches more probability from the alternatives whose baseline utilities behave similarly to μ_{nj_2} than the utilities whose baseline utilities behave differently. In other words, the more strongly coupled two baseline utilities are in $p(\mu_n \mid \psi)$, the better substitutes the two corresponding alternatives will be for each other as we expand or contract the choice set.

Consequently, the goal in developing a productive mixed independent Gumbel model is to engineer a mixture kernel $p(\mu_n \mid \psi)$ so any desired substitution patterns are mirrored in the couplings of the baseline utilities.

5.3 The Multivariate Normal Baseline Mixing Kernel

Continuous mixture modeling is an extremely general technique that can be used to construct an infinite diversity of discrete choice models. One particular family of continuous mixture models, however, dominates applications of discrete choice modeling in practice.

Because it uses a base independent Gumbel model, this family is often referred to simply as the "mixed logit" model in the literature. I will avoid using that term here for two main reasons. Firstly, as I discussed at the end of Section 3.1, I'm not a fan of the term "logit" in the context of discrete choice models. Secondly, the family is only one of many ways that we might mix a base independent Gumbel model.

5.3.1 General Construction

The ubiquitous approach starts with a base independent Gumbel model and then mixes the baseline utilities with a multivariate normal kernel,

$$p(\mu_n \mid \kappa_n, \Sigma) = \text{multinormal}(\mu_n \mid \kappa_n, \Sigma).$$

In this chapter I will refer to the resulting family of discrete choice models as **multivariate normal baseline** discrete choice models, although this is not standard terminology. A more common description that one might encounter in other literature is a **normal error component** model.

The components of κ_n define a sort of baseline, baseline utility for each alternative. In practice they're usually interpreted in the same way as the baseline utilities in the base independent Gumbel model: the larger κ_{nj_1} is relative to κ_{nj_2} the more agent n prefers alternative j_1 to alternative j_2 in a fixed choice set.

Unfortunately, many discussions of mixed discrete choice modeling take these similarities a bit too far and treat probabilistic utilities, u_{nj}, baseline utilities, μ_{nj}, and baseline-baseline utilities, κ_{nj}, as equivalent objects. Many go as far to use the same variables to denote all three, leaving it up to the reader to distinguish when each quantity is being considered. To avoid any ambiguity in this chapter, I will be diligent in using different notation for each distinct object.

What really defines the substitution patterns of this mixed discrete choice model is the structure of Σ. The larger $\Sigma_{j_1 j_2}$ is, the better substitutes the j_1th and j_2th alternatives will be for each other.

To facilitate a wide range of covariance structures, including even singular structures, the covariance is often written as

$$\Sigma = Z \cdot \Gamma \cdot Z^T,$$

where Γ is, in general, a positive-definite, symmetric $(L \times L)$-matrix and Z is a rectangular $(J \times L)$-matrix.

If $L < J$ then Σ will not be positive-definite, but the mixed discrete choice model will still be well-defined. To avoid having to deal with any singular behavior in Σ when implementing these models, we can instead model latent deviations,

$$p(\theta_n \mid \Gamma) = \text{multinormal}(\theta_n \mid 0, \Gamma),$$

that are then pushed forward through a deterministic linear transformation,

$$\mu_n = \kappa_n + Z \cdot \theta_n$$

to give

$$p(\mu_n \mid \kappa_n, Z, \Gamma) = \text{multinormal}(\mu_n \mid \kappa_n, Z \cdot \Gamma \cdot Z^T).$$

One nice benefit of this implementation is that it clarifies how Z mediates the coupling between each alternative. The larger $Z_{j_1 l}$ and $Z_{j_2 l}$ are, the more θ_{nl} contributes to both μ_{nj_1} and μ_{nj_2} which coupling the two baseline utilities together. If there is only one l for with $Z_{j_1 l}$ and $Z_{j_2 l}$ are both non-zero, then μ_{nj_1} and μ_{nj_2} will be *exactly* coupled to each other in the mixed model. At the same time, if there is no l for which $Z_{j_1 l}$ and $Z_{j_2 l}$ are both non-zero, then μ_{nj_1} and μ_{nj_2} will be independent of each other in the mixed model.

This model can also be implemented with *non-centered* deviations η_n,

$$p(\eta_n) = \text{multinormal}(\eta_n \mid 0, I)$$

with

$$\mu_n = \kappa_n + Z \cdot L_\Gamma \cdot \eta_n$$

where L is the Choleksy decomposition of Γ,

$$L_\Gamma \cdot L_\Gamma^T = \Gamma.$$

I will refer to the implementation of this mixed model with θ_n as a **centered** implementation and the implementation with η_n as a **non-centered** implementation. Note that, in general, the deviations for *each* agent can be implemented differently from each other. This flexibility will be useful when trying to optimize the performance of these models in practice.

5.3.2 Special Cases

Many common discrete choice models are defined by a mixed independent Gumbel model with specific choices for the structure of Z and Γ. Even more discrete choice models can be *well-approximated* by an appropriate mixed independent Gumbel model. In this section I'll discuss a few; for a much more extensive presentation see Walker, Ben-Akiva, and Bolduc (2007) and the references therein.

For example, the **heteroskedastic logit model** is defined by taking $L = J$ with Z the J-dimensional identify matrix and Γ a diagonal matrix quantifying variances for each alternative.

This model simply introduces variation into each baseline utility, complicating the resulting substitution patterns.

The **analytic factor model** is given by fixing $L < J$ but allowing Z to be an otherwise arbitrary matrix and Γ to be an arbitrary positive-definite, symmetric matrix. This lack of restrictions makes the analytic factor model extremely flexible, but that flexibility also tends to manifest in frustrating inferential degeneracies in practice.

The mixed independent Gumbel model can not exactly reproduce the behavior of the nested Gumbel model, but it can approximate it well. To do so we take $L = K$ and then fix the elements of the coupling matrix to be

$$Z_{jk} = \begin{cases} 1, & j \in B_k \\ 0, & j \notin B_k \end{cases}.$$

Because each alternative falls into one and only one nest, each row of Z contains only a single non-zero value.

Finally we restrict Γ to a diagonal matrix with K elements

$$\Gamma_{kk} = \gamma_k^2.$$

As γ_k increases, the alternatives in the kth nest become better substitutes for each other relative to the alternatives in other nests. Qualitatively, each γ_k behaves similarly to $1 - \lambda_k$ in the nested Gumbel model, with smaller values corresponding to weaker couplings within the kth nest and larger values corresponding to larger couplings.

Generalizing this model to approximate the generalized nested Gumbel model is straightforward. We can allow the jth alternative to have partial membership in an arbitrary number of nests by generalizing Z so that each row of zeros and ones is replaced by a $(K-1)$-simplex of positive values summing to one.

5.4 Relative Implementations

Although mixed discrete choice models exhibit substitution patterns that are distinct from those of their base models, they often exhibit the same redundancies as their base models. In particular, all discrete choice models derived by mixing the baseline utilities in an independent Gumbel model inherit the translation and scale redundancies of that base model. Fortunately, we can eliminate these redundancies using techniques to those we discussed in Section 3.2.

5.4.1 Deriving The Relative Baseline Utility Model

To account for the translation redundancy in a mixed independent Gumbel model, we need to transform the baseline utilities to relative baseline utilities,

$$\delta_{nj} = \mu_{nj} - \mu_{nj'}.$$

When working with a mixed model, however, this transformation is no longer *deterministic* but rather *probabilistic*. In this case we have to push forward the initial probabilistic model for the baseline utilities, $p(\mu_n \mid \kappa_n, Z, \Gamma)$, into a probabilistic model for the relative baseline utilities, $p(\delta_n \mid \kappa_n, Z, \Gamma)$.

The implementation of this operation is a bit more straightforward if we adopt linear algebra for the anchoring transformation. Collecting the baseline utilities μ_{nj} into the vector μ_n and the relative baseline utilities δ_{nj} into the vector δ_n allows us to anchor the j'th baseline utility to zero with the matrix operation

$$\delta_n = \mathbf{A} \cdot \mu_n,$$

where the elements of A are given by

$$A_{ii'} = \delta_{ii'} - \delta_{i'j'}.$$

Here δ_{ij} denotes the Kronecker delta function,

$$\delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}.$$

In other words A is formed by taking the $J \times J$ identify matrix and then subtracting one from all of the elements in the j'th column.

Under this transformation the multivariate normal model for the baseline utilities,

$$p(\mu_n \mid \kappa_n, Z, \Gamma) = \text{multinormal}(\mu_n \mid \kappa_n, Z \cdot \Gamma \cdot Z^T),$$

conveniently pushes forward to another multivariate normal model for the relative baseline utilities,

$$p(\delta_n \mid \kappa_n, Z, \Gamma) = \text{multinormal}(\nu_n \mid \chi_n, \Omega),$$

only with the transformed location vector

$$\chi_n = A \cdot \kappa_n$$

and the transformed covariance matrix

$$\begin{aligned} \Omega &= A \cdot Z \cdot \Gamma \cdot Z^T \cdot A^T \\ &= (A \cdot Z) \cdot \Gamma \cdot (A \cdot Z)^T. \end{aligned}$$

Equivalently we can avoid any singular behavior by writing this model for the relative baseline utilities as

$$\delta_n = A \cdot \kappa_n + (A \cdot Z) \cdot \theta_n$$

with

$$p(\theta_n \mid \Gamma) = \text{multinormal}(\theta_n \mid 0, \Gamma).$$

The corresponding non-centered implementation is given by

$$\delta_n = A \cdot \kappa_n + A \cdot Z \cdot L \cdot \eta_n.$$

with

$$p(\eta_n) = \text{multinormal}(\eta_n \mid 0, I).$$

and

$$L \cdot L^T = \Gamma.$$

5.4.2 Eliminating Redundancies In The Relative Baseline Utilities

At this point we have to reconcile the anchoring with the covariance structure of the baseline utilities. In particular, if the transformed covariance matrix,

$$\Omega = (A \cdot Z) \cdot \Gamma \cdot (A \cdot Z)^T,$$

is singular, then some of the elements of ν_n will be redundant.

Conveniently, we can quantify this singularity analytically using the rank R of the combined matrix

$$W = A \cdot Z.$$

If W is rank-deficient,

$$R < L \neq J,$$

then then many different configurations of the deviations θ_n, or η_n in the non-centered implementation, will yield exactly the same configuration of the relative utilities ν_n. In this case no amount of data will ever be able to distinguish between these equivalent possibilities.

How do we actually avoid these redundancies when W is singular? Fortunately, we can use the same linear algebra tools to identify singular behavior and rectify it.

The systematic procedure starts with the combined transformation

$$W = A \cdot Z.$$

and then performs a singular value decomposition (Train 2009),

$$W = U \cdot S \cdot V^T.$$

Note that, in theory, we could also use a rank-revealing QR decomposition here. The only challenge is that, in practice, it's not always straightforward to determine whether or not an available QR decomposition is actually rank-revealing.

This singular value decomposition isolates the singular behavior of W into the central matrix S. Specifically, if the rank of W is less than L then all of the elements of S will be zero except for the diagonals in the upper-left $R \times R$ block,

$$S = \begin{pmatrix} \overbrace{\widetilde{S_{11}}}^{R \times R} & \overbrace{\widehat{0}}^{R \times (L-R)} \\ \underbrace{0}_{(J-R) \times R} & \underbrace{0}_{(J-R) \times (L-R)} \end{pmatrix}.$$

Next we decompose the U and V matrices to match the block structure of S,

$$U = \begin{pmatrix} \overbrace{\widetilde{U_{11}}}^{R \times R} & \overbrace{\widetilde{U_{12}}}^{R \times (J-R)} \\ \underbrace{U_{21}}_{(J-R) \times R} & \underbrace{U_{22}}_{(J-R) \times (J-R)} \end{pmatrix}$$

$$V = \begin{pmatrix} \overbrace{\widetilde{V_{11}}}^{R \times R} & \overbrace{\widetilde{V_{12}}}^{R \times (L-R)} \\ \underbrace{V_{21}}_{(L-R) \times R} & \underbrace{V_{22}}_{(L-R) \times (L-R)} \end{pmatrix}$$

This allows us to reduce W to only it's non-singular behavior,

$$\begin{aligned} W &= U \cdot S \cdot V^T \\ &= \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \cdot \begin{pmatrix} S_{11} & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V_{11}^T & V_{21}^T \\ V_{12}^T & V_{22}^T \end{pmatrix} \\ &= \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \cdot \begin{pmatrix} S_{11} \cdot V_{11}^T & S_{11} \cdot V_{21}^T \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} \cdot \begin{pmatrix} S_{11} \cdot V_{11}^T & S_{11} \cdot V_{21}^T \end{pmatrix} \\ &= B \cdot C, \end{aligned}$$

where

$$B = \underbrace{\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}}_{J \times R}$$

and

$$C = \underbrace{\begin{pmatrix} S_{11} \cdot V_{11}^T & S_{11} \cdot V_{21}^T \end{pmatrix}}_{R \times L}.$$

We can now completely isolate the non-singular covariations of the relative baseline utilities into the non-singular covariance matrix

$$\Xi = \underbrace{C \cdot \Gamma \cdot C^T}_{R \times R}.$$

The centered implementation of this non-redundant relative baseline utilities model is given by

$$\delta_n = A \cdot \kappa_n + B \cdot \zeta_n,$$

where

$$p(\zeta_n) = \text{multinormal}(\zeta_n \mid 0, \Xi).$$

Note the vector ζ_n consists of not L elements here but rather only $R < L$ elements.

At the same time we also have a corresponding non-centered implementation,

$$\delta_n = A \cdot \kappa_n + B \cdot L_\Xi \cdot \xi_n.$$

where

$$p(\xi_n) = \text{multinormal}(\xi_n \mid 0, I).$$

and

$$L_\Xi \cdot L_\Xi^T = \Xi = C \cdot \Gamma \cdot C^T.$$

Once again the vector ξ_n consists of only $R < L$ elements.

5.4.3 Eliminating Latent Redundancies

Even after we eliminate the redundancies in the relative baseline utilities, these mixed discrete choice models can still exhibit redundancies in the latent location and covariance parameters.

For example, the J components of the original baseline-baseline utilities κ are no longer distinguishable once we move to relative baseline utilities. Fortunately we can immediately avoid this redundancy by directly modeling the $J - 1$ components of relative baseline-baseline utilities,

$$\chi = A \cdot \kappa,$$

instead of the J components of κ.

The potential redundancies in the covariance parameters are a bit more subtle.

By construction the initial covariance matrix Γ contains up to $L(L+1)/2$ individual elements, each of which could be modeled with separate parameters. That said, in some implementations we might constrain the structure of Γ, for example by fixing the off-diagonal elements to zero to leave only L free parameters. In general let's say that we're modeling Γ with

$$H \leq \frac{L\,(L+1)}{2}$$

total parameters.

On the other hand, the non-singular covariance matrix

$$\Xi = C \cdot \Gamma \cdot C^T,$$

which can actually be informed by observed data, contains only $R(R+1)/2$ individual elements.

If we ever have

$$H > \frac{R\,(R+1)}{2}$$

then we will not able to able to distinguish all H of the parameters comprising Γ. The only way to avoid this redundancy is to reduce parameterization of Γ to any set of $R(R+1)/2$ parameters that completely determine Ξ. Often this is best done by hand to ensure that the reduced parameterization is as interpretable as possible.

5.4.4 Cleaning Up

One limitation of this systematic procedure is that the resulting matrices X and U matrices are not always the most interpretable objects. In many cases it can be useful to massage these matrices into more interpretable forms with the introduction of some auxiliary matrices.

Specifically, consider two $R \times R$ matrices M_1 and M_2 that are inverses of each other,

$$\mathbf{M}_1 \cdot \mathbf{M}_2 = \mathbf{M}_2^T \cdot \mathbf{M}_1^T = \mathbf{I}.$$

Inserting the identify matrix into the relative covariance matrix,

$$\begin{aligned} \Omega &= B \cdot \left(C \cdot \Gamma \cdot B^T \right) \cdot B^T \\ &= B \cdot \mathbf{I} \cdot \left(C \cdot \Gamma \cdot C^T \right) \cdot \mathbf{I} \cdot B^T \end{aligned}$$

allows us to insert these matrices,

$$\begin{aligned} \Omega &= B \cdot \mathbf{I} \cdot \left(C \cdot \Gamma \cdot C^T \right) \cdot \mathbf{I} \cdot B^T \\ &= B \cdot (\mathbf{M}_1 \cdot \mathbf{M}_2) \cdot \left(C \cdot \Gamma \cdot C^T \right) \cdot (\mathbf{M}_2^T \cdot \mathbf{M}_1^T) \cdot B^T \\ &= (B \cdot \mathbf{M}_1)\,(\mathbf{M}_2 \cdot C) \cdot \Gamma \cdot \left(C^T \cdot \mathbf{M}_2^T \right) \cdot \left(\mathbf{M}_1^T \cdot B^T \right) \\ &= (B \cdot \mathbf{M}_1)\,(\mathbf{M}_2 \cdot C) \cdot \Gamma \cdot (M_2 \cdot \mathbf{C})^T \cdot (B \cdot \mathbf{M}_1)^T. \end{aligned}$$

At this point we can then engineer M_1 and M_2 so that

$$B \cdot \mathbf{M}_1$$

and

$$\mathbf{M}_2 \cdot C$$

take on more interpretable forms.

To be clear this step is entirely optional, and finding useful matrices is much more of an art than a systematic process. That said, this procedure can be extremely convenient when trying to simplify the more general mixed model construction in the context of particular special cases.

Specifically this kind of manipulation is often helpful for reconciling the structure of discrete choice models derived from the general construction and discrete choice models derived from special cases. For example, if we know the desired form of $B \cdot \mathbf{M}_1$ then we can solve for $\mathbf{M}_1$ and then $\mathbf{M}_2 = \mathbf{M}_1^{-1}$.

5.4.5 Demonstration

To demonstrate the construction of non-redundant, relative baseline utilities let's consider a mixed independent Gumbel model that approximates the nested Gumbel model. We start with the nesting configuration, which we'll take from the previous exercise.

```
J <- 10
K <- 3
nests <- c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3)
```

Next we construct a matrix that encodes this nesting structure. Multiplying this matrix by the deviation vector will add the deviation associated with each nest to the baseline utilities of the corresponding alternatives.

```
Z <- matrix(0, J, K)
for (k in 1:K)
  Z[which(nests == k), k] <- 1

Z
```

```
      [,1] [,2] [,3]
 [1,]    1    0    0
 [2,]    1    0    0
 [3,]    1    0    0
 [4,]    0    1    0
 [5,]    0    1    0
 [6,]    0    1    0
 [7,]    0    1    0
```

```
 [8,]    0    0    1
 [9,]    0    0    1
[10,]    0    0    1
```

Lastly we construct a matrix that encodes the anchoring transformation taking baseline utilities to relative baseline utilities.

```
anchor <- 10
```

```
A <- diag(nrow=J) -
     cbind(sapply(1:J,
                  function(j) rep(as.numeric(j == anchor), J)))

A
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    0    0    0    0    0    0    0    0    -1
 [2,]    0    1    0    0    0    0    0    0    0    -1
 [3,]    0    0    1    0    0    0    0    0    0    -1
 [4,]    0    0    0    1    0    0    0    0    0    -1
 [5,]    0    0    0    0    1    0    0    0    0    -1
 [6,]    0    0    0    0    0    1    0    0    0    -1
 [7,]    0    0    0    0    0    0    1    0    0    -1
 [8,]    0    0    0    0    0    0    0    1    0    -1
 [9,]    0    0    0    0    0    0    0    0    1    -1
[10,]    0    0    0    0    0    0    0    0    0     0
```

To eliminate the redundancy in this model we combine A and A into a single transformation.

```
W <- A %*% Z
```

We then isolate the non-singular part of this combined transformation by apply a singular value decomposition.

```
SVD <- svd(W)
```

The non-zero diagonal elements of $\mathbf{S}$ immediately indicate that the rank of $\mathbf{W}$ is not $K = 3$ but rather only 2.

```
SVD$d
```

```
[1] 3.256617 1.842403 0.000000
```

```
R <- sum(SVD$d != 0)
```

Finally we construct the $\mathbf{B}$ and $\mathbf{C}$ matrices by reducing the components of the singular value decomposition.

```
B <- SVD$u[,1:R]

1e-3 * round(1e3 * B)
```

```
        [,1]   [,2]
 [1,] -0.347  0.461
 [2,] -0.347  0.461
 [3,] -0.347  0.461
 [4,] -0.400 -0.301
 [5,] -0.400 -0.301
 [6,] -0.400 -0.301
 [7,] -0.400 -0.301
 [8,]  0.000  0.000
 [9,]  0.000  0.000
[10,]  0.000  0.000
```

```
C <- cbind(diag(SVD$d[1:R]) %*%
           t(SVD$v[1:R, 1:R]),
           diag(SVD$d[1:R]) %*%
           t(matrix(SVD$v[(R + 1):K, 1:R], nrow=K - R, ncol=R)))

1e-3 * round(1e3 * C)
```

```
       [,1]   [,2]   [,3]
[1,] -1.041 -1.598  2.639
[2,]  1.384 -1.202 -0.182
```

As expected, the product $\mathbf{B}\cdot\mathbf{C}$ recovers $\mathbf{W}$ without having to deal with any singular behavior.

```
1e-3 * round(1e3 * (B %*% C - W))
```

```
      [,1] [,2] [,3]
 [1,]    0    0    0
 [2,]    0    0    0
 [3,]    0    0    0
 [4,]    0    0    0
 [5,]    0    0    0
 [6,]    0    0    0
 [7,]    0    0    0
 [8,]    0    0    0
 [9,]    0    0    0
[10,]    0    0    0
```

The only downside to this procedure is that $\mathbf{B}$ and $\mathbf{C}$ are not the most interpretable objects. To clean them up a bit consider the matrices $\mathbf{M}_1$ and $\mathbf{M}_2$.

```
M1 <- matrix(c(-1.041141, 1.384206, -1.598343, -1.202206),
             2, 2)
M2 <- matrix(c(-0.3470470, -0.3995857,  0.4614019, -0.3005516),
             2, 2)
```

Critically, these matrices are inverses of each other.

```
1e-3 * round(1e3 * (M1 %*% M2))
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Consequently we can use $\mathbf{B} \cdot \mathbf{M}_1$ instead of $\mathbf{B}$ and $\mathbf{M}_2 \cdot \mathbf{C}$ instead of $\mathbf{C}$ when constructing the relative baseline utilities.

```
1e-3 * round(1e3 * (B %*% M1))
```

```
      [,1] [,2]
 [1,]    1    0
 [2,]    1    0
 [3,]    1    0
 [4,]    0    1
```

```
 [5,]    0    1
 [6,]    0    1
 [7,]    0    1
 [8,]    0    0
 [9,]    0    0
[10,]    0    0
```

```
1e-3 * round(1e3 * (M2 %*% C))
```

```
     [,1] [,2] [,3]
[1,]    1    0   -1
[2,]    0    1   -1
```

What's so special about these matrices? Well $\mathbf{M}_2 \cdot \mathbf{C}$ defines a transformation taking the nest-wise deviations to *relative* nest-wise deviations,

$$\theta_{n,k} - \theta_{n,k'},$$

where the anchor nest k' is the nest containing the anchored alternative, in this case the third nest. At the same time $\mathbf{B} \cdot \mathbf{M}_1$ defines a transformation that adds these relative nest-wise deviations to the corresponding alternatives.

All of this is to say that we can derive the non-redundant nested mixed discrete choice model in two ways. On one hand we can map the nest-wise deviations into relative nest-wise deviations to eliminate the redundancy and then add the relative nest-wise deviations to the appropriate relative baseline utilities. At the same time we can construct the relative baseline utilities first and then systematically eliminate the resulting redundancy using the singular value decomposition. Because both approaches implement the same model, we are free to use the one that is most convenient in any given application.

After eliminating the redundancy in the relative baseline utilities we have to consider the potential redundancy in the covariance parameters.

In the nested mixed discrete choice model we take the covariance matrix Γ to be diagonal, resulting in

$$H = K$$

free parameters. The non-redundant implementation of this model informs

$$\begin{aligned} G &= \frac{R\,(R + 1)}{2} \\ &= \frac{(K - 1)\,(K - 1 + 1)}{2} \\ &= \frac{(K - 1)\,K}{2} \end{aligned}$$

covariance parameters. If $K > 2$ then $H < G$ and all of the non-zero elements of Γ will be informed by the observed data.

What happens, however, if $K = 2$? Well let's consider an explicit example.

```
J <- 10
K <- 2
nests <- c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)
```

```
Z <- matrix(0, J, K)
for (k in 1:K)
  Z[which(nests == k), k] <- 1

A <- diag(nrow=J) -
  cbind(sapply(1:J,
               function(j) rep(as.numeric(j == anchor), J)))

W <- A %*% Z
```

```
SVD <- svd(W)
R <- sum(SVD$d != 0)
C <- diag(SVD$d[1:R]) %*% t(SVD$v[1:R, 1:R])

1e-3 * round(1e3 * C)
```

```
       [,1]  [,2]
[1,] -2.236 2.236
[2,]  0.000 0.000
```

In this case the matrix $\mathbf{C}$ has the form

$$\mathbf{C} = \begin{pmatrix} a & -a \\ b & b \end{pmatrix}$$

and the non-singular covariance has the form

$$\begin{aligned} &= \mathbf{C} \cdot \Gamma \cdot \mathbf{C}^T \\ &= \begin{pmatrix} a & -a \\ b & b \end{pmatrix} \cdot \begin{pmatrix} \gamma_1^2 & 0 \\ 0 & \gamma_2^2 \end{pmatrix} \cdot \begin{pmatrix} a & b \\ -a & b \end{pmatrix} \\ &= (\gamma_1^2 + \gamma_2^2) \begin{pmatrix} a^2 & -a\,b \\ -a\,b & b^2 \end{pmatrix}. \end{aligned}$$

Because γ_1 and γ_2 appear only as a squared sum, any observations will only ever inform that sum and not the individual values. The only way to avoid redundancy in this case is to model the squared sum and then derive the consistent values for γ_1 and γ_2.

5.4.6 Introducing Relative Agent Scales

Once we've derived a non-redundant probabilistic model for the relative baseline utilities we deal with the redundancy of the agent scales.

Our first step is exactly the same as in the unmixed independent Gumbel model: we distinguish an anchor agent n' and then define relative scales

$$\tau_n = \frac{\sigma_n}{\sigma_{n'}}.$$

At this point we need to absorb $\sigma_{n'}$ into the relative baseline utilities. In the unmixed independent Gumbel model this was a deterministic operation where we define

$$\omega_n = \frac{\delta_n}{\sigma_{n'}}.$$

In the mixed model, however, we need to specify an appropriate probabilistic model for ω_n.

Fortunately this is a relatively – how could this *not* be an intentional pun – straightforward operation,

$$\omega_n = \rho_n + B \cdot \iota_n,$$

with

$$p(\iota_n) = \text{multinormal}(\iota_n \mid 0, \Upsilon).$$

Here

$$\rho_n = \frac{1}{\sigma_{n'}} \chi_n = \frac{1}{\sigma_{n'}} A \cdot \kappa_n$$

and

$$\begin{aligned} \Upsilon &= \frac{1}{\sigma_{n'}} \Xi \\ &= C \cdot \left(\frac{1}{\sigma_{n'}} \Gamma \right) \cdot C^T \\ &\equiv C \cdot \left(\Psi \right) \cdot C^T \end{aligned}$$

In other words, instead of modeling χ_n and Γ we instead model ρ_n and Ψ. The resulting implementation looks the same superficially but the interpretation of these objects, and consequently any principled prior model, will change.

5.5 Practical Implementation

While mixed discrete choice models are straightforward to construct in theory, they can be a challenge to implement in practice. The main difficulty is that directly implementing a mixed discrete choice model for an observed choice y_m requires being able to evaluate a marginal choice probability,

$$\begin{aligned} p(y_m \mid \psi) &= \text{categorical}(y_m \mid q_{n(m)1}(\psi), \ldots, q_{n(m)J}(\psi)) \\ &= q_{n(m)y_m}(\psi) \\ &= \int \mathrm{d}\phi \, p(\phi \mid \psi) \, q_{n(m)y_m}(\phi). \end{aligned}$$

Unfortunately analytic integration is not possible for even for the simplest base choice probability $q_{nj}(\phi)$. Numerical integration is available, but it requires care to implement effectively and its cost can quickly become infeasible.

That said, we don't actually have to evaluate the integral at all. Instead of working with the marginal discrete choice model,

$$p(y_m, \psi) = p(y_m \mid \psi) \, p(\psi),$$

we can also jointly model ϕ and ψ together,

$$p(y_m, \phi, \psi) = p(y_m \mid \phi) \, p(\phi \mid \psi) \, p(\psi),$$

without the need for any explicit marginalization.

To see why, recall that all well-defined Bayesian inferences take the form of posterior expectation values. Moreover, for mixed discrete choice models the relevant expectands are functions of only the global variables ψ,

$$\begin{aligned} \mathbb{E}[f] &= \int \mathrm{d}\psi \, p(\psi \mid y_m) f(\psi) \\ &= \int \mathrm{d}\psi \, \frac{p(y_m \mid \psi) \, p(\psi)}{\int \mathrm{d}\psi \, p(y_m \mid \psi) \, p(\psi)} f(\psi) \\ &= \frac{\int \mathrm{d}\psi \, p(y_m \mid \psi) \, p(\psi) \, f(\psi)}{\int \mathrm{d}\psi \, p(y_m \mid \psi) \, p(\psi)}. \end{aligned}$$

Expanding the marginal model as a continuous mixture allows us to write these expectation

values as

$$\begin{aligned}
\mathbb{E}[f] &= \frac{\int \mathrm{d}\psi \, p(y_m \mid \psi) \, p(\psi) \, f(\psi)}{\int \mathrm{d}\psi \, p(y_m \mid \psi) \, p(\psi)} \\
&= \frac{\int \mathrm{d}\psi \left[\int \mathrm{d}\phi \, p(y_m \mid \phi) \, p(\phi \mid \psi) \right] p(\psi) \, f(\psi)}{\int \mathrm{d}\psi \left[\int \mathrm{d}\phi \, p(y_m \mid \phi) \, p(\phi \mid \psi) \right] p(\psi)} \\
&= \frac{\int \mathrm{d}\psi \, \mathrm{d}\phi \, p(y_m \mid \phi) \, p(\phi \mid \psi) \, p(\psi) \, f(\psi)}{\int \mathrm{d}\psi \, \mathrm{d}\phi \, p(y_m \mid \phi) \, p(\phi \mid \psi) \, p(\psi)} \\
&= \int \mathrm{d}\psi \, \mathrm{d}\phi \, \frac{p(y_m \mid \phi) \, p(\phi \mid \psi) \, p(\psi)}{\int \mathrm{d}\psi \, \mathrm{d}\phi \, p(y_m \mid \phi) \, p(\phi \mid \psi) \, p(\psi)} f(\psi) \\
&= \int \mathrm{d}\psi \, \mathrm{d}\phi \, p(\phi, \psi \mid y_m) \, f(\psi).
\end{aligned}$$

In other words, we will always arrive at the *exact same* Bayesian inferences regardless of whether we work with the marginal model $p(y_m, \psi)$ or the joint model $p(y_m, \phi, \psi)$. Intuitively, the auxiliary ϕ parameter is implicitly marginalized whenever we compute joint expectation values for functions $f(\psi)$. This includes, for example, the mixed choice probability, predictive choices, and so on.

All of this, however, concerns only a single observation and its corresponding choice probability. When modeling multiple observations we generally have to consider multiple choice probabilities. In this case the proper joint model requires *distinct* auxiliary parameters for each distinct choice probability.

For example, if each agent-alternative pairing results in a distinct choice probability $q_{nj}(\psi)$ then the joint model would require one auxiliary parameter for every observed configuration of n and j,

$$\begin{aligned}
p(y_1, \ldots, y_M, &\phi_{11}, \ldots, \phi_{NJ}, \psi) \\
&= \prod_{m=1}^{M} p(y_m \mid \phi_{n(m)y_m}) \cdot \prod_{n=1}^{N} \prod_{j=1}^{J} p(\phi_{nj} \mid \psi) \cdot p(\psi) \\
&= \prod_{m=1}^{M} q_{n(m)y_m}(\phi_{n(m)y_m}) \cdot \prod_{n=1}^{N} \prod_{j=1}^{J} p(\phi_{nj} \mid \psi) \cdot p(\psi).
\end{aligned}$$

If, on the other hand, the base choice probabilities also depend on external covariates, $q_{nj}(x_m, \phi)$, then we would need one auxiliary parameter for *every* observation with a unique covariate value,

$$\begin{aligned}
p(y_1, \ldots, y_M, &\phi_{11}, \ldots, \phi_{NJ}, \psi \mid x_1, \ldots, x_M) \\
&= \prod_{m=1}^{M} p(y_m \mid \phi_m) \, p(\phi_m \mid \psi) \cdot p(\psi).
\end{aligned}$$

The superficial drawback with a joint implementation of a mixed discrete choice model instead of a marginal implementation is that the additional parameters increase the computational burden of estimating posterior expectation values. Tools that scale well to high-dimensions, such as Hamiltonian Monte Carlo, substantially moderate much of this burden.

The more subtle, but often more relevant, concern with a joint implementation is not so much the total number of parameters but rather how those parameters are coupled together in the joint posterior distribution. For instance, when working with smaller data sets the coupling between the parameters in the mixing kernel $p(\phi_{nj} \mid \psi)$ can warp the posterior geometry, which then frustrates posterior computation.

In the multivariate normal baseline mixed model the posterior geometries induced by the centered and non-centered implementations exhibit complementary geometric properties. The implementation that achieves optimal performance depends on the details of each application, in particular the amount of data for each agent and the behavior of the prior model. Conveniently, this is very similar to the behavior of normal hierarchical models. Consequently we can carry over experience implementing those models directly to the implementation of mixed discrete choice models with multivariate normal baseline utilities.

5.6 Demonstration

Admittedly, that was *a lot*. Let's take some time to anchor these concepts by analyzing the data from the previous exercise on the nested Gumbel model with a nested mixed independent Gumbel model.

To avoid any redundancies we need to construct the reduced transformation. Fortunately we already worked through this in Section 5.4.5.

```
Z <- matrix(0, data$J, data$K)
for (k in 1:data$K)
  Z[which(data$nests == k), k] <- 1

A <- diag(nrow=data$J) -
     cbind(sapply(1:data$J,
                  function(j)
                  rep(as.numeric(j == data$anchor), data$J)))

W <- A %*% Z
SVD <- svd(W)

data$R <- sum(SVD$d != 0)
data$B <- SVD$u[,1:data$R]
data$C <- cbind(diag(SVD$d[1:data$R]) %*%
```

```
                t(SVD$v[1:data$R, 1:data$R]),
                diag(SVD$d[1:data$R]) %*%
                t(matrix(SVD$v[(data$R + 1):data$K, 1:data$R],
                         nrow=data$K - data$R, ncol=data$R)))
```

```
data$R
```

```
[1] 2
```

We could, in theory, implement these calculations directly in a `Stan` program. The main difficulty with this approach is that checking for exactly zero singular values can be problematic when working with floating point arithmetic. I prefer to implement these calculations externally and pass them to the `Stan` program so that I can first review the singular values and the calculated rank.

Each of the $N \cdot S$ unique combinations of agents and choice subsets defines a unique set of choice probabilities. In order to implement the mixed independent Gumbel model we will then need $N \cdot S$ distinct vectors of auxiliary deviations, each which can implemented as centered or non-centered deviations. Instead of hard-coding a centered or non-centered implementation for each of these vectors, we'll use a more general `Stan` program where the precise implementation can be tuned externally.

Let's start by configuring the `Stan` program so that all of the deviations are centered.

```
data$I_cp <- data$N * data$S
data$cp_idxs <- matrix(NA, data$I_cp, 2)
i <- 1
for (n in 1:data$N) {
  for (s in 1:data$S) {
    data$cp_idxs[i, 1] <- n
    data$cp_idxs[i, 2] <- s
    i <- i + 1
  }
}

data$I_ncp <- 0
data$ncp_idxs <- matrix(NA, 0, 2)
```

```
fit <- stan(file='stan_programs/mig_joint1.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Unfortunately, this results in divergences.

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

```
  Chain 2: 1 of 1024 transitions (0.1%) diverged.

  Chain 3: 37 of 1024 transitions (3.6%) diverged.

  Chain 4: 1 of 1024 transitions (0.1%) diverged.

  Divergent Hamiltonian transitions result from unstable numerical
trajectories.  These instabilities are often due to degenerate target
geometry, especially "pinches".  If there are only a small number of
divergences then running with adept_delta larger than 0.801 may reduce
the instabilities at the cost of more expensive Hamiltonian
transitions.
```

```
samples_mig <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples_mig,
                                       c('rho_free',
                                         'tau_free',
                                         'psi',
                                         'iota_cp'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
rho_free[4]:
  Chain 3: hat{ESS} (31.470) is smaller than desired (100).

rho_free[6]:
  Chain 1: hat{ESS} (90.666) is smaller than desired (100).
  Chain 3: hat{ESS} (36.760) is smaller than desired (100).

rho_free[7]:
  Chain 3: hat{ESS} (39.587) is smaller than desired (100).

tau_free[3]:
  Chain 3: hat{ESS} (67.111) is smaller than desired (100).

psi[1]:
```

```
  Chain 3: hat{ESS} (76.123) is smaller than desired (100).

psi[3]:
  Chain 3: hat{ESS} (32.195) is smaller than desired (100).

iota_cp[13,1]:
  Chain 3: hat{ESS} (56.734) is smaller than desired (100).

iota_cp[16,1]:
  Chain 3: hat{ESS} (86.432) is smaller than desired (100).

iota_cp[27,1]:
  Chain 3: hat{ESS} (87.343) is smaller than desired (100).

iota_cp[38,1]:
  Chain 3: hat{ESS} (51.759) is smaller than desired (100).

iota_cp[46,1]:
  Chain 3: hat{ESS} (91.095) is smaller than desired (100).

iota_cp[47,1]:
  Chain 3: hat{ESS} (62.573) is smaller than desired (100).

iota_cp[51,1]:
  Chain 3: hat{ESS} (56.289) is smaller than desired (100).

iota_cp[53,1]:
  Chain 3: hat{ESS} (54.125) is smaller than desired (100).

iota_cp[9,2]:
  Chain 3: hat{ESS} (96.013) is smaller than desired (100).

iota_cp[10,2]:
  Chain 3: hat{ESS} (31.715) is smaller than desired (100).

iota_cp[13,2]:
  Chain 3: hat{ESS} (46.272) is smaller than desired (100).

iota_cp[22,2]:
  Chain 3: hat{ESS} (79.291) is smaller than desired (100).

iota_cp[25,2]:
  Chain 3: hat{ESS} (45.067) is smaller than desired (100).
```

```
iota_cp[26,2]:
  Chain 3: hat{ESS} (72.290) is smaller than desired (100).

iota_cp[27,2]:
  Chain 3: hat{ESS} (65.931) is smaller than desired (100).

iota_cp[31,2]:
  Chain 3: hat{ESS} (83.740) is smaller than desired (100).

iota_cp[32,2]:
  Chain 3: hat{ESS} (96.773) is smaller than desired (100).

iota_cp[37,2]:
  Chain 3: hat{ESS} (67.078) is smaller than desired (100).

iota_cp[44,2]:
  Chain 3: hat{ESS} (43.102) is smaller than desired (100).

iota_cp[46,2]:
  Chain 3: hat{ESS} (60.079) is smaller than desired (100).

iota_cp[53,2]:
  Chain 3: hat{ESS} (79.744) is smaller than desired (100).


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

Fortunately, we don't have to go searching through a myriad of scatter plots to try to diagnose this problem. If the centered deviations result in a problematic posterior geometry then the non-centered deviations should be better behaved.

In general we might consider non-centering only those deviations for agent-choice subset pairs with the most observations. Here we'll just go all the way an non-center all of the deviations.

```
data$I_ncp <- data$N * data$S
data$ncp_idxs <- matrix(NA, data$I_ncp, 2)
i <- 1
for (n in 1:data$N) {
  for (s in 1:data$S) {
    data$ncp_idxs[i, 1] <- n
    data$ncp_idxs[i, 2] <- s
```

```
    i <- i + 1
  }
}

data$I_cp <- 0
data$cp_idxs <- matrix(NA, 0, 2)
```

```
fit <- stan(file='stan_programs/mig_joint1.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Looks like this did the trick! There are now no signs of incomplete posterior quantification.

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

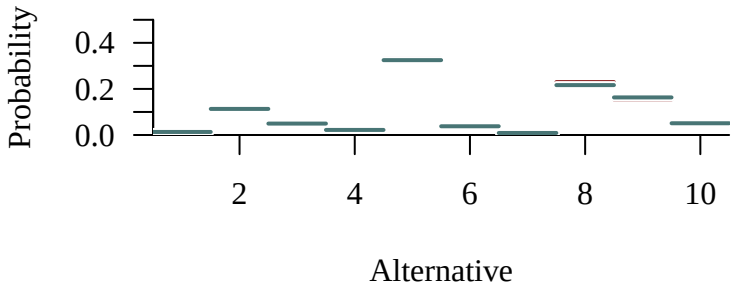
```
  Chain 1: 1 of 1024 transitions (0.1%) diverged.

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

  Chain 3: 3 of 1024 transitions (0.3%) diverged.

  Chain 4: 1 of 1024 transitions (0.1%) diverged.

  Divergent Hamiltonian transitions result from unstable numerical
trajectories.  These instabilities are often due to degenerate target
geometry, especially "pinches".  If there are only a small number of
divergences then running with adept_delta larger than 0.801 may reduce
the instabilities at the cost of more expensive Hamiltonian
transitions.
```

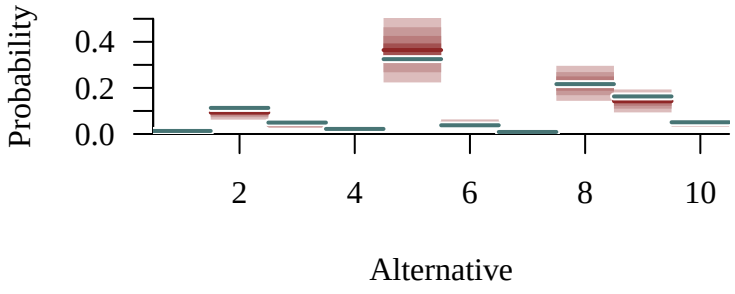
```
samples_mig <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples_mig,
                                       c('rho_free',
                                         'tau_free',
                                         'psi',
                                         'iota_ncp'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

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

How do the inferences from this model compare to those from the exact nested Gumbel model? The inferred probabilities for the full choice set are consistent, both with each other and the true values. That said, the uncertainties in the mixed model are substantially larger.

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

probs_true <- exp(log_probs_true)

names <- sapply(1:data$J,
                function(j) paste0('probs_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(samples_ng, names,
                                     baseline_values=probs_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Probability",
                                     display_ylim=c(0, 0.5))

names <- sapply(1:data$J,
                function(j) paste0('probs_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(samples_mig, names,
                                     baseline_values=probs_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Probability",
                                     display_ylim=c(0, 0.5))
```

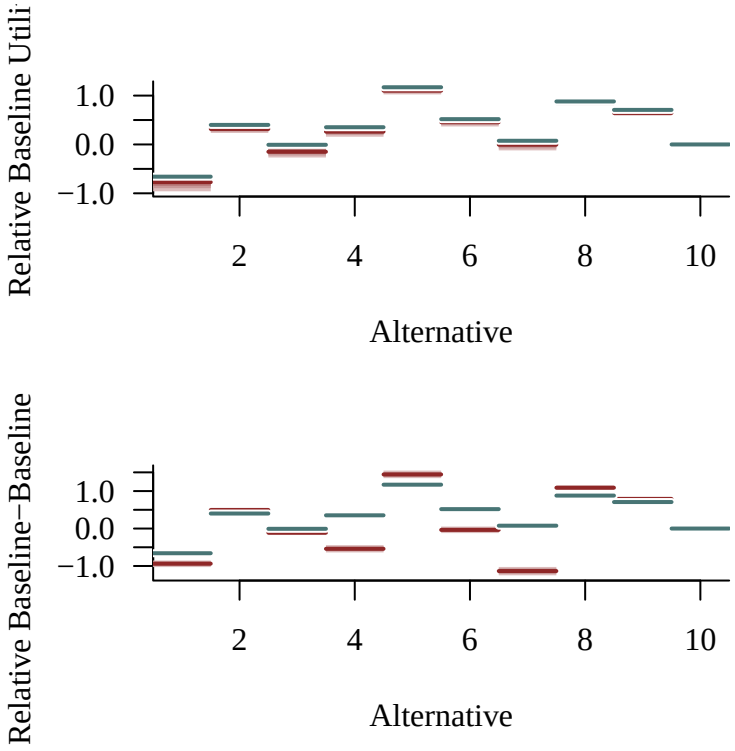




We have to be careful comparing the parameters in the two models. Many of the parameters are similar but not exactly equivalent. For example the relative baseline utilities in the nested Gumbel model and the relative baseline-baseline utilities in the mixed independent Gumbel model exhibit qualitatively similar behaviors, but the quantitative details don't quite match.

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

omega_true <- (mu_true - mu_true[data$anchor]) / sigma_true[1]

names <- sapply(1:data$J,
                function(j) paste0('omega[', j, ']'))
ylab <- "Relative Baseline Utility"
util$plot_disc_pushforward_quantiles(samples_ng, names,
                                     baseline_values=omega_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab=ylab)

names <- sapply(1:data$J,
                function(j) paste0('rho[', j, ']'))
ylab <- "Relative Baseline-Baseline Utility"
util$plot_disc_pushforward_quantiles(samples_mig, names,
                                     baseline_values=omega_true,
                                     baseline_col=util$c_mid_teal,
```

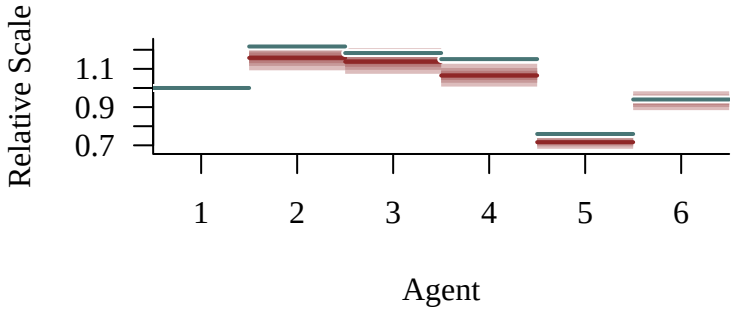
```
                                    xlab="Alternative",
                                    ylab=ylab)
```

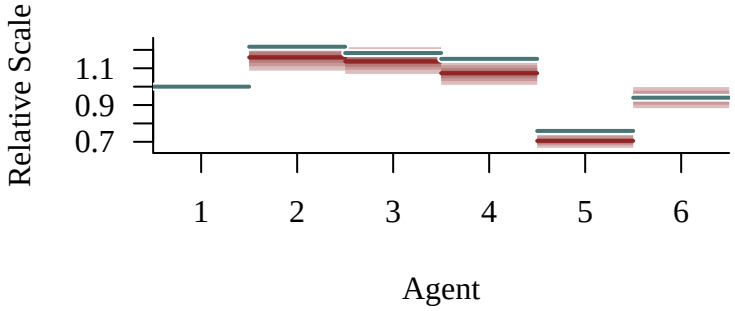


On the other hand the agent scales are equivalent in the two models, and indeed their inferences are consistent with each other.

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

tau_true <- sigma_true / sigma_true[1]

names <- sapply(1:data$N,
                function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(samples_ng, names,
                                     baseline_values=tau_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Agent",
                                     ylab="Relative Scale")

names <- sapply(1:data$N,
                function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(samples_mig, names,
```

```
                                    baseline_values=tau_true,
                                    baseline_col=util$c_mid_teal,
                                    xlab="Agent",
                                    ylab="Relative Scale")
```

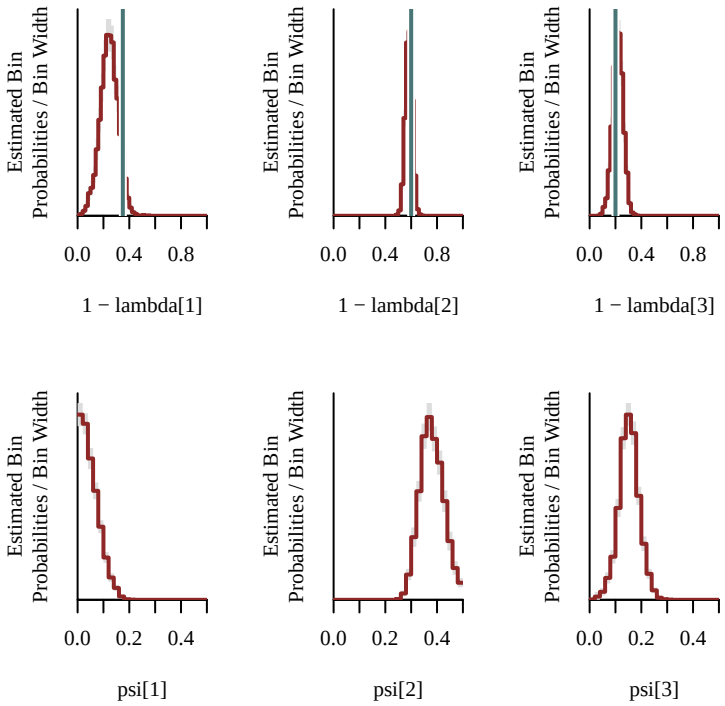




This leaves the intra-nest coupling parameters in the two models, which are similar but not identical. Once we reorient the λ_k in the nested Gumbel model they behavior qualitatively similar to the ψ_k in the mixed independent Gumbel model: the alternatives in the second next is are the most strongly coupled while the alternatives in the third nest are the most weakly coupled.

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

for (k in 1:data$K) {
  name <- paste0('lambda[', k, ']')
  util$plot_expectand_pushforward(1 - samples_ng[[name]],
                                  50, flim=c(0, 1),
                                  display_name=paste('1 -', name),
                                  baseline=1 - lambda_true[k],
                                  baseline_col=util$c_mid_teal)
}
```

```
for (k in 1:data$K) {
  name <- paste0('psi[', k, ']')
  util$plot_expectand_pushforward(samples_mig[[name]],
                                  25, flim=c(0, 0.5),
                                  display_name=name)
}
```



Lastly, in Section 5.4.5 we saw that we could also derive this mixed independent Gumbel model by introducing nest-wise deviations and then allocating them to each alternative. We can also readily implement this model in the `Stan` modeling language.

```
fit <- stan(file='stan_programs/mig_joint2.stan',
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

There are no signs of computational problems with this alternative implementation.

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

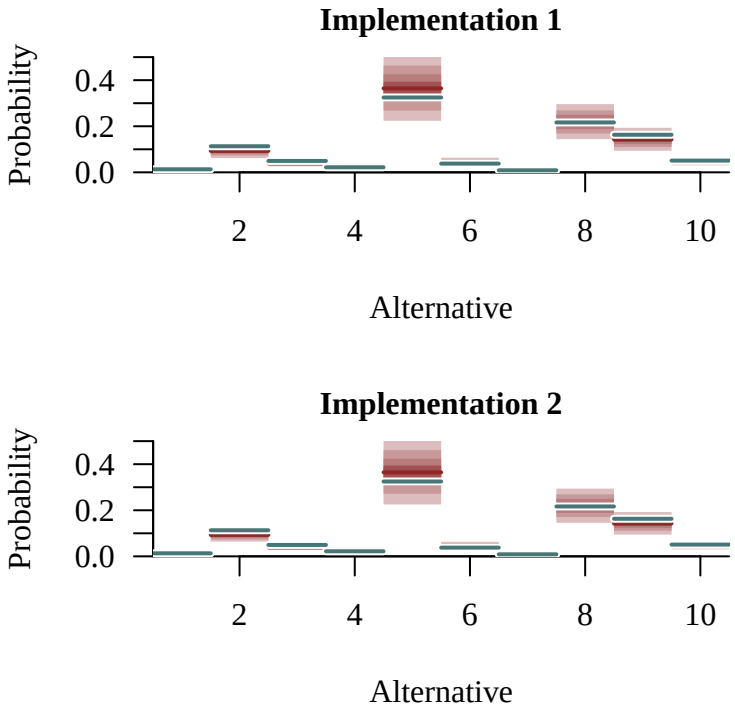
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples_mig2 <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples_mig2,
                                       c('rho_free',
                                         'tau_free',
                                         'psi',
                                         'iota_ncp'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

```
psi[3]:
  Chain 2: hat{ESS} (91.786) is smaller than desired (100).


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

Moreover, the resulting inferences are exactly the same up to the expected Markov chain Monte Carlo variation.

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

names <- sapply(1:data$J,
                function(j) paste0('probs_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(samples_mig, names,
                                     baseline_values=probs_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Probability",
                                     display_ylim=c(0, 0.5),
                                     main='Implementation 1')

names <- sapply(1:data$J,
                function(j) paste0('probs_pred[', j, ']'))
util$plot_disc_pushforward_quantiles(samples_mig2, names,
                                     baseline_values=probs_true,
                                     baseline_col=util$c_mid_teal,
                                     xlab="Alternative",
                                     ylab="Probability",
                                     display_ylim=c(0, 0.5),
                                     main='Implementation 2')
```



Ultimately we are free to derive, and then implement, these mixed models in whichever way is most convenient for ourselves and our collaborators.

6 Derived Baseline Utility Models

All of the probabilistic utility models that we have considered so far, from the independent Gumbel model to the nested Gumbel model to the probit model, are configured by baseline utilities μ_{nj} that determine the centrality of the probabilistic utilities. In all of our examples we have modeled these baseline utilities directly, but we can also *derive* them. This allows us, for example, to model how baseline utilities systematically vary with the properties of the agents, alternatives, and their interactions.

Although we will focus on deriving baseline utilities here, these techniques are equally applicable to deriving baseline-baseline utilities in mixed discrete choice models. For instance instead of modeling κ_n in a mulivariate normal baseline utility model, we could derive the κ_n from a deterministic model.

6.1 Factor Models

Factor models capture the heterogeneity of a behavior across given contexts. Mechanically, factor models expand a variable into deviations across those contexts.

Consider, for example, a variable α that models a behavior of interest and K categorical variables, or factors, each consisting of L_k individual values, or levels. If we assume that this behavior is homogeneous across those levels then the value of α is constant,

$$\alpha = \alpha_0.$$

At the same time, we can introduce independent heterogeneity across the levels in each factor with level-specific deviations,

$$\alpha = \alpha_0 + \alpha^k_{i_k}.$$

Coupled heterogeneity is then modeled by introducing pairwise deviations,

$$\alpha = \alpha_0 + \alpha^k_{i_k} + \alpha^{k_1 k_2}_{i_{k_1} i_{k_2}} + \ldots,$$

three-way deviations, and so on. Typically this expansion is truncated after only the first few contributions, although truncated factor models require care as neglecting large higher-order contributions leads to poor inferential and predictive performance.

If we think that the behavior of the baseline utilities might vary across the levels of relevant factors then we can always model that heterogeneity with factor models for each baseline utility

6.1.1 Agent and Alternative Factors

For example, we have so far considered each baseline utility as independently capturing the behavior particular to each agent-alternative pairing. In many applications, however, this behavior is coupled across common agents and alternatives.

We can model behavior shared by specific agents and alternatives by expanding the baseline utilities into a factor model where the agent and alternative assignments are treated as factors. If we include all possible interactions then this results in the derived baseline utilities

$$\mu_{nj} = \alpha_0 + \alpha_n + \alpha_j + \alpha_{nj}.$$

Here the baseline α_0 captures the behaviors common to all choices. The first-order contributions α_n and α_j capture the residual behaviors that are common to each particular agent and alternative, respectively. Finally the α_{nj} capture any remaining behaviors that are idiosyncratic to particular agent-alternative pairings.

The only problem with this model, which is typical of factor models more generally, is that we end up with too many parameters. With enough observations, the data will inform the $N\,J$ baseline utilities. On the other hand, the factor model contains

$$1 + N + J + N\,J > N\,J$$

total parameters, which cannot all be distinguished by constraints on the baseline utilities. In other words, this exhaustive factor model is redundant.

One way to avoid this redundancy is to ignore any behaviors particular to each agent-alternative pairing and simply truncate the residual expansion at

$$\mu_{nj} = \alpha_0 + \alpha_n + \alpha_j.$$

This approach works well when the agent-alternative idiosyncrasies are actually negligible in the true data generating process.

Another approach is to regularize the deviations around α_0. We could do this, for example, by introducing a hard constraint the possible value of α_n, α_j, and α_{nj}. Similarly we could introduce a softer constraint through an informative prior model or even independent hierarchical models.

6.1.2 Conjoint Models

Often the alternatives in a discrete choice application are not distinct objects but rather different configurations of some common object. Moreover, these configurations can often be specified by a collection of categorical factors. A discrete choice model that uses a factor model to quantify how the baseline utilities vary across these discrete configurations is known as a **conjoint model**.

If the configurations of the target object are defined by F categorical factors, each of which can take one of L_f possible levels, then there will be

$$L = \prod_{f=1}^{F} L_f$$

possible configurations in total. Even for a small number of factors, the total number of possible configurations will be impossible to exhaustively explore in any one analysis. Instead the data in practical conjoint analyses typically span only a subset of these possible configurations, leaving us to use our modeling assumptions to infer the behavior of unobserved configurations.

As with any factor model, conjoint models can in general consider how the baseline utilities vary across arbitrary interactions between the configuring factors. Most conjoint models, however, assume that the interactions are negligible and instead build up the baseline utilities from only a common initial value and then deviations to account for independent variations across the levels in each factor,

$$\mu_{nj} = \alpha_0 + \sum_{f=1}^{F} \alpha^f_{l_f(j)}.$$

6.1.3 Relative Factor Level Contributions

When implementing a discrete choice model that derives the baseline utilities from a factor models, we have to keep in mind that baseline utilities are often redundant. In order to avoid this redundancies we typically have to distinguish some reference alternative j' and then implement the model using relative baseline utilities for each agent,

$$\delta_{nj} = \mu_{nj} - \mu_{nj'}.$$

The transition to relative baseline utilities has an interesting consequence for baseline utility factor models. Consider, for example, a factor model defined by agent and alternative factors,

$$\mu_{nj} = \alpha_0 + \alpha_n + \alpha_j + \alpha_{nj}.$$

In this case the relative baseline utilities become

$$\begin{aligned} \delta_{nj} = &\quad \alpha_0 + \alpha_n + \alpha_j + \alpha_{nj} \\ &- \alpha_0 + \alpha_n + \alpha_{j'} + \alpha_{nj'} \\ &= (\alpha_j - \alpha_{j'}) + (\alpha_{nj} - \alpha_{nj'}). \end{aligned}$$

Because the baseline contribution α_0 and agent-specific contributions α_n are common to each agent, they cancel exactly when considering only relative behaviors. Consequently these contributions can be safely ignored!

The alternative-specific contributions persist, but now quantify the behavior of each alternative relative to the distinguished reference alternative,

$$\epsilon_j = \alpha_j - \alpha_{j'}.$$

Similarly the idiosyncratic agent-alternative contributions quantify the behavior relative to the reference pairing,

$$\epsilon_{nj} = \alpha_{nj} - \alpha_{n'j'}.$$

More generally, distinguishing an anchor alternative define distinguished levels in each factor. Relative baseline utilities are then built up from factor level contributions relative to those distinguished reference levels.

In a conjoint model, for example, a reference alternative is defined by a reference configuration specified by a particular level in each of the categorical factors. The relative baseline utility for any other configuration is defined by how much the baseline utility improves or worsens as we move away from those reference levels.

6.2 Conjoint Model Demonstration

Let's take a minute to see how a conjoint model might be implemented in practice.

In this example we're interesting in understanding what kind of statistics book would be of interest to the broad market of consumers. The potential books can be characterized by four factors, each of which contains a varying number of levels.

```
factor_names <- list('Length', 'Depth', 'Topic', 'Emphasis')

factor_levels <-
  list( list('1-25 pp', '26-100 pp', '101-250 pp',
             '251-500 pp', '500+ pp'),
        list('Beginner', 'Intermediate', 'Advanced'),
        list('Probability\nTheory', 'Modeling and\nInference',
             'Modeling\nTechniques', 'Case\nStudies'),
        list('Mathematical\nConcepts',
             'Mathematical\nDerivations',
             'Programming\nImplementations') )
```

Even with this relatively crude characterization of statistics books, the number of possibilities is a bit overwhelming.

```
L <- sapply(factor_levels, function(f) length(f))
Reduce('*', L)
```

```
[1] 180
```

To narrow down the book configurations that are mostly likely to be to popular we've surveyed individual consumers, asking each to choose their preferred configuration from a choice sets consisting of a selection of possible configurations.

6.2.1 Explore Data

We begin by reading in the data and taking a look.

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

The survey consisted of 5000 observed choices gathered from 10 individual consumers.

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

```
5000 total observations
```

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

```
10 total agents
```

In each observation one of these consumers was given four hypothetical book configurations and asked for the book that they would be most likely to read.

```
cat(sprintf('%i alternatives in each choice set', data$J))
```

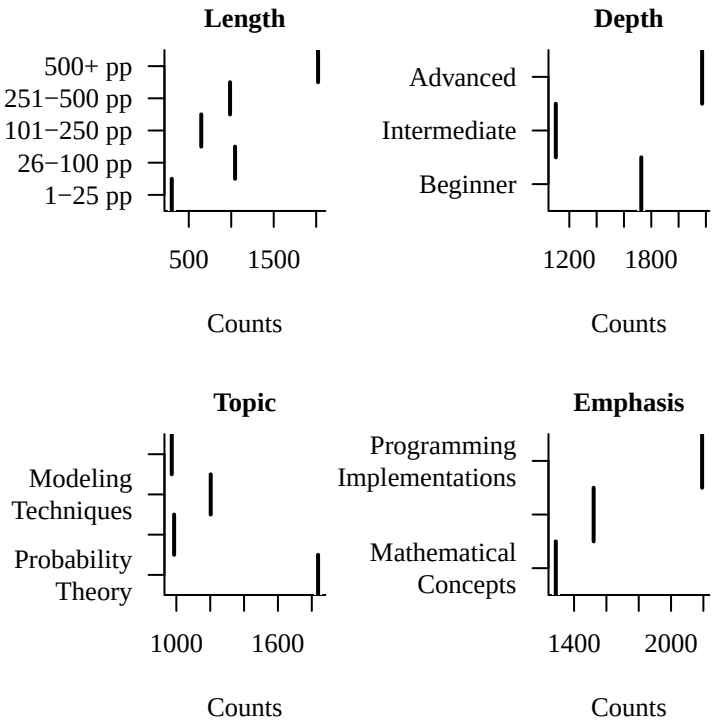
```
4 alternatives in each choice set
```

Outside of a few coincidences, the choice set in each observation consists of a unique collection of alternatives. This makes visualizing the data a bit tricky. Instead of looking at the chosen alternatives directly, we'll examine at the frequency of each factor level in the chosen alternatives.

```
chosen_level_counts <- c()
for (f in 1:data$F) {
  L <- data$factor_end[f] - data$factor_start[f] + 1

  factor_chosen_levels <-
    sapply(1:data$M,
           function(m) data$alt_levels[m, data$choice[m], f])

  factor_chosen_level_counts <-
    hist(factor_chosen_levels,
         breaks=seq(0.5, L + 0.5, 1),
         plot=FALSE)$counts

  chosen_level_counts <-  c(chosen_level_counts,
                            factor_chosen_level_counts)
}
```

Just from the raw data alone we can see strong preferences for longer lengths, deeper depths, more theoretical topics, and a programming emphasis.

```
par(mfrow=c(2, 2), mar = c(5, 6, 2, 1))

for (f in 1:data$F) {
  util$plot_disc_vals_vert(
    chosen_level_counts[data$factor_start[f]:data$factor_end[f]],
    yticklabs=factor_levels[[f]],
    xlab='Counts',
    main=factor_names[f]
  )
}
```



6.2.2 Specifying An Observational Model

For this analysis we'll assume an independent Gumbel model,

$$q_{nj} = \frac{e^{\frac{\mu_{nj}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}},$$

with baseline utilities that derived from a first-order factor model built up from our four book configuration factors,

$$\mu_{nj} = \alpha_0 + \sum_{f=1}^{F} \alpha^f_{l_f(j)}.$$

Note that this implies that the baseline utility, and ultimately the derived choice probability, for any book configuration is independent of any particular consumer. In other words we're assuming that the consumer behavior is homogeneous.

To avoid redundancies we won't actually model the α^f_l and σ_n but rather the scaled relative contributions

$$\upsilon^f_l = \frac{\alpha^f_l - \alpha^f_{l'}}{\sigma_{n'}}.$$

and relative scales

$$\tau_n = \frac{\sigma_n}{\sigma_{n'}}.$$

This, in turn, requires specifying reference levels for each configuration factor.

```
factor_anchors <- c(2, 1, 2, 3)
for (f in 1:data$F) {
  anchor <- factor_anchors[f]
  print(paste0('Factor ', f, ': Anchor ',
        factor_names[[f]], ' = ',
        factor_levels[[f]][[anchor]]))
}
```

```
[1] "Factor 1: Anchor Length = 26-100 pp"
[1] "Factor 2: Anchor Depth = Beginner"
[1] "Factor 3: Anchor Topic = Modeling and\nInference"
[1] "Factor 4: Anchor Emphasis = Programming\nImplementations"
```

```
data$factor_anchors <- factor_anchors
```

As we have done in other exercises, we'll use the first agent as our anchor agent.

6.2.3 Specifying A Prior Model

To specify a full Bayesian model we need to complement this observational model with a prior model.

Reasoning about the relative factor contributions υ_l^f is a bit more straightforward if we consider some hypothetical book configurations. Let j_1 denote the book configuration defined by all of the anchor levels and j_2 a book configuration that is exactly the same except for the level of the fth factor. The ratio of choice probabilities for these two alternatives and the anchor agent is given by

$$\begin{aligned}
\frac{q_{n' j_2}}{q_{n' j_1}} &= \frac{e^{\frac{\mu_{n' j_2}}{\sigma_{n'}}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{n' j'}}{\sigma_{n'}}}} \frac{\sum_{j'=1}^{J} e^{\frac{\mu_{n' j'}}{\sigma_{n'}}}}{e^{\frac{\mu_{n' j_1}}{\sigma_{n'}}}} \\
&= \exp\left(\frac{\mu_{n' j_2} - \mu_{n' j_2}}{\sigma_{n'}}\right) \\
&= \exp\left(\frac{\alpha^f_{l(j_2)} - \alpha^f_{l(j_1)}}{\sigma_{n'}}\right) \\
&= \exp\left(\frac{\alpha^f_{l(j_2)} - \alpha^f_{l'}}{\sigma_{n'}}\right) \\
&= \exp\left(\upsilon^f_{l(j_2)}\right),
\end{aligned}$$

or

$$\log \frac{q_{n' j_2}}{q_{n' j_1}} = \upsilon^f_{l(j_2)}.$$

Consequently any domain expertise about the reasonably variation in the choice probability as we tweak the level of the fth factor immediately constrains the reasonable values of $\upsilon^f_{l(j_2)}$. Here let's be *extremely* generous and allow for a full order of magnitude proportional variation in the choice probabilities,

$$\begin{aligned}
0.1 \lessapprox \quad & \frac{q_{n' j_2}}{q_{n' j_1}} \quad \lessapprox \quad 10 \\
-\log 10 \lessapprox \log & \frac{q_{n' j_2}}{q_{n' j_1}} \lessapprox +\log 10.
\end{aligned}$$

This then implies

$$-\log 10 \lessapprox \upsilon^f_{l(j_2)} \lessapprox +\log 10.$$

A soft containment prior model that concentrates most, but not all, of the prior probability within these constraints is given by

$$\begin{aligned} p\left(\upsilon^{f}_{l(j_2)}\right) &= \text{normal}\left(\upsilon^{f}_{l(j_2)} \;\middle|\; 0, \frac{\log 10}{2.32}\right) \\ &\approx \text{normal}\left(\upsilon^{f}_{l(j_2)} \;\middle|\; 0, 1\right). \end{aligned}$$

Similarly, we can allow for an order of magnitude proportional variation in the agent scales with the half-normal prior model

$$\begin{aligned} p\left(\tau_n\right) &= \text{normal}\left(\tau_n \;\middle|\; 0, \frac{10}{2.57}\right) \\ &\approx \text{normal}\left(\tau_n \mid 0, 4\right). \end{aligned}$$

6.2.4 Model Implementation

To account for the varying number of levels in each of the book configuration factors, we'll pack all of the factor level contributions into a single, concatenated array that we address with start/end indices. The main subtlety is that there are only $L_f - 1$ non-zero contributions in each factor, so this concatenated array consists of not

$$LL = \sum_{f=1}^{F} L_f$$

total elements but rather

$$LL_{\text{rel}} = \sum_{f=1}^{F} (L_f - 1) = \left(\sum_{f=1}^{F} L_f\right) - F$$

total elements.

6.2.5 Posterior Quantification

With our model implementation in hand let's try to quantify some posterior inferences.

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

Fortunately there don't appear to be any computational problems.

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

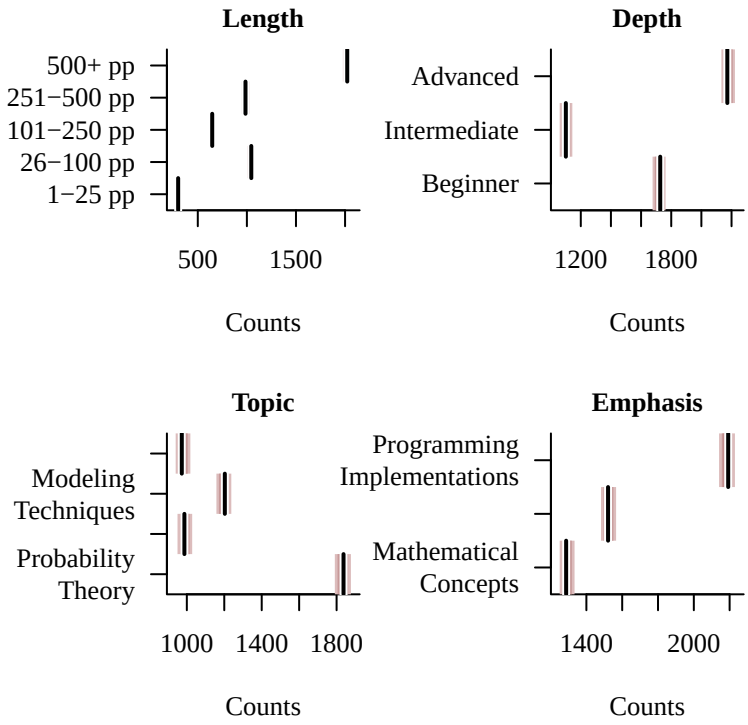
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('upsilon_free',
                                         'tau_free'),
                                       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 for the observed factor level histograms is excellent.

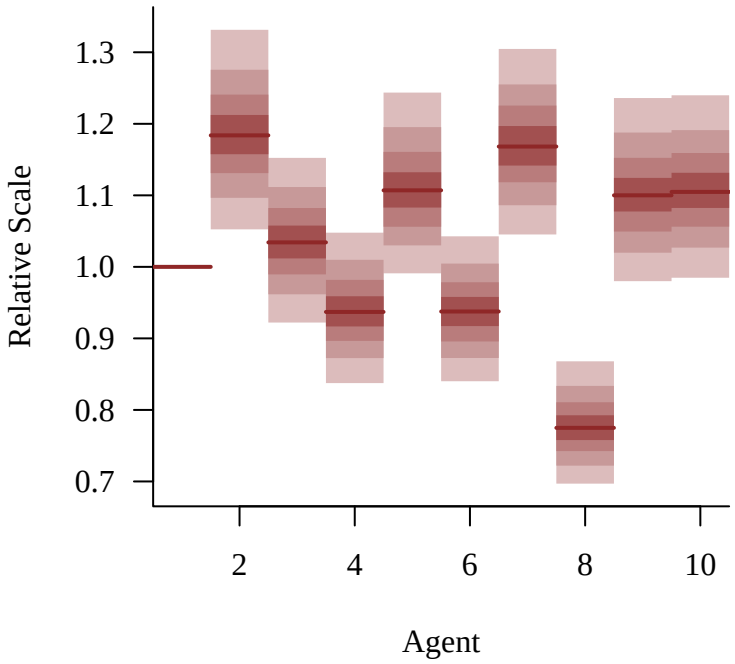
```
par(mfrow=c(2, 2), mar = c(5, 5, 2, 1))

for (f in 1:data$F) {
  obs <-
    chosen_level_counts[data$factor_start[f]:data$factor_end[f]]
  names <-
    sapply(data$factor_start[f]:data$factor_end[f],
           function(ll) paste0('chosen_level_counts_pred[',
                               ll, ']'))
  util$plot_disc_pushforward_quantiles_vert(
    samples, names,
    baseline_values=obs,
    yticklabs=factor_levels[[f]],
    xlab='Counts',
    main=factor_names[f]
  )
}
```



To really confirm that this first-order factor model is sufficient we would really want to see how well our model reproduces the observed frequency pairs of chosen factors levels, triplets, and so on. Given the pedagogical nature of this exercise, however, we'll move on.

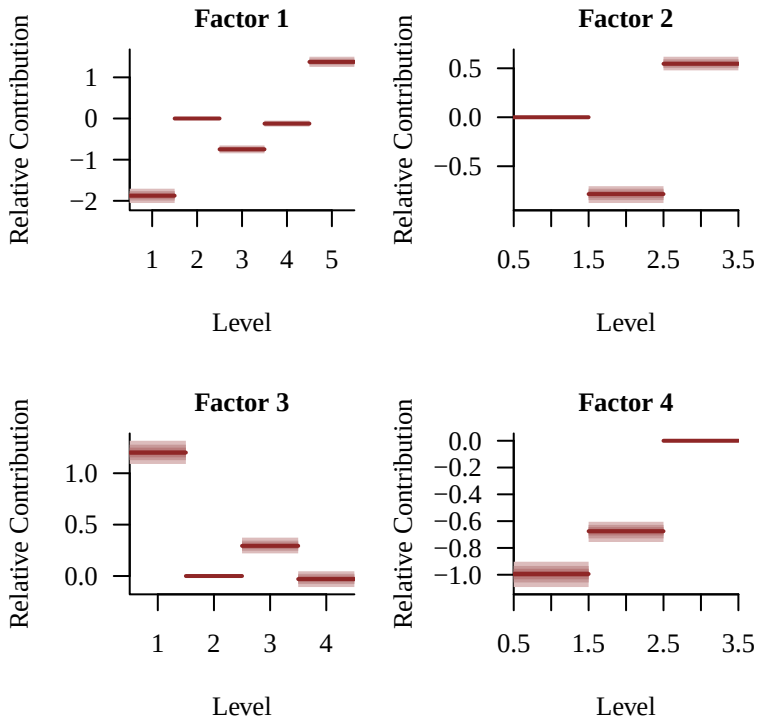
Inferences for the relative agent scales appear to be reasonable. Given the uncertainties, the agent scaling is largely uniform. That said the sixth consumer surveyed is a bit of an outlier, with more sensitivity to the baseline utilities than the other agents.

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

names <- sapply(1:data$N, function(n) paste0('tau[', n, ']'))
util$plot_disc_pushforward_quantiles(samples, names,
                                     xlab="Agent",
                                     ylab="Relative Scale")
```



More pertinent to our original question, however, are the inferences for the configuration factor level contributions.

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

for (f in 1:data$F) {
  names <- sapply(data$factor_start[f]:data$factor_end[f],
                  function(ll) paste0('upsilon[', ll, ']'))
  util$plot_disc_pushforward_quantiles(samples, names,
                                       xlab="Level",
                                       ylab="Relative Contribution",
                                       main=paste('Factor', f))
}
```



6.2.6 Consumer Appeal

For example, we can use these posterior inferences to identify by hand the most and least appealing levels of each configuration factor.

```
max_levels <- c(5, 3, 1, 3)
for (f in 1:data$F) {
  l <- max_levels[f]
  print(paste0('Best ',
               factor_names[[f]], ': ',
               factor_levels[[f]][[l]]))
}
```

```
[1] "Best Length: 500+ pp"
[1] "Best Depth: Advanced"
[1] "Best Topic: Probability\nTheory"
[1] "Best Emphasis: Programming\nImplementations"
```

```
min_levels <- c(1, 2, 4, 1)
for (f in 1:data$F) {
```

```
  l <- min_levels[f]
  print(paste0('Worst ',
               factor_names[[f]], ': ',
               factor_levels[[f]][[l]]))
}
```

```
[1] "Worst Length: 1-25 pp"
[1] "Worst Depth: Intermediate"
[1] "Worst Topic: Case\nStudies"
[1] "Worst Emphasis: Mathematical\nConcepts"
```

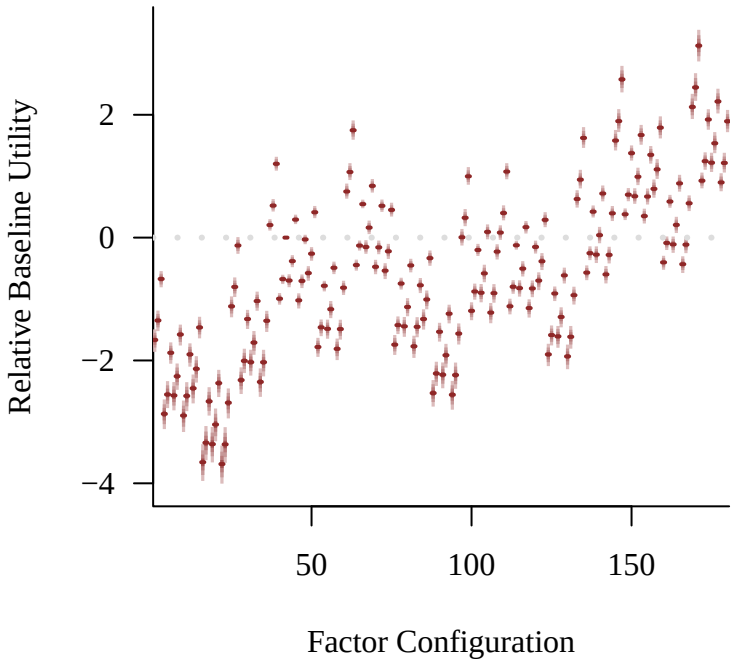
We can also programmatically calculate the relative baseline utility for *every* possible book configuration.

```
N_config <-
  Reduce('*',
         sapply(1:data$F,
                function(f)
                data$factor_end[f] - data$factor_start[f] + 1))

factor_configs <- matrix(NA, N_config, data$F)
pushforward_samples <- list()

ls <- rep(1, data$F)
for (c in 1:N_config) {
  # Save current factor configuration
  factor_configs[c,] <- ls

  # Compute relative baseline utility for factor configuration
  var_repl <-
    list('x'=sapply(1:data$F,
                    function(f)
                    paste0('upsilon[',
                           data$factor_start[f] + ls[f] - 1,
                           ']')))

  name <- paste0('config', c)
  pushforward_samples[[name]] <-
    util$eval_expectand_pushforward(samples,
                                    function(x) sum(x),
                                    var_repl)
```

```
  # Update factor configuration
  ls[data$F] <- ls[data$F] + 1
  for (f in rev(1:(data$F - 1))) {
    if (ls[f + 1] >
        data$factor_end[f + 1] - data$factor_start[f + 1] + 1)
      ls[f] <- ls[f] + 1
  }

  for (f in rev(1:data$F)) {
    if (ls[f] > data$factor_end[f] - data$factor_start[f] + 1)
      ls[f] <- 1
  }
}
```

If we directly comparing the marginal posterior distribution for the relative baseline utility for all of the possible configurations, then we can see some clear winners and losers.

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

names <- sapply(1:N_config,
                function(c) paste0('config', c))
util$plot_disc_pushforward_quantiles(
  pushforward_samples, names,
  xlab="Factor Configuration",
  ylab="Relative Baseline Utility"
)
abline(h=0, lwd=3, lty=3, col='#DDDDDD')
```



Exactly inferring ranks is challenging, but we can approximately rank each book configuration using the expectation value of its relative baseline utility.

```
expected_utility <- function(c) {
  name <- paste0('config', c)
  util$ensemble_mcmc_est(pushforward_samples[[name]])[1]
}

expected_utilities <- sapply(1:N_config,
                             function(c) expected_utility(c))

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

This allows us to isolate the best performing configurations.

```
for (c in rev(tail(post_mean_ordering, 4))) {
  for (f in 1:data$F) {
    l <- factor_configs[c,f]
    print(paste0(factor_names[[f]], ' = ',
                 factor_levels[[f]][[l]]))
  }
  print('')
}
```

```
[1] "Length = 500+ pp"
[1] "Depth = Advanced"
[1] "Topic = Probability\nTheory"
[1] "Emphasis = Programming\nImplementations"
[1] ""
[1] "Length = 500+ pp"
[1] "Depth = Beginner"
[1] "Topic = Probability\nTheory"
[1] "Emphasis = Programming\nImplementations"
[1] ""
[1] "Length = 500+ pp"
[1] "Depth = Advanced"
[1] "Topic = Probability\nTheory"
[1] "Emphasis = Mathematical\nDerivations"
[1] ""
[1] "Length = 500+ pp"
[1] "Depth = Advanced"
[1] "Topic = Modeling\nTechniques"
[1] "Emphasis = Programming\nImplementations"
[1] ""
```

As well as some of the worst.

```
for (c in head(post_mean_ordering, 4)) {
  for (f in 1:data$F) {
    l <- factor_configs[c,f]
    print(paste0(factor_names[[f]], ' = ',
                 factor_levels[[f]][[l]]))
  }
  print('')
}
```

```
[1] "Length = 1-25 pp"
[1] "Depth = Intermediate"
[1] "Topic = Case\nStudies"
[1] "Emphasis = Mathematical\nConcepts"
[1] ""
[1] "Length = 1-25 pp"
[1] "Depth = Intermediate"
[1] "Topic = Modeling and\nInference"
[1] "Emphasis = Mathematical\nConcepts"
[1] ""
```

```
[1] "Length = 1-25 pp"
[1] "Depth = Intermediate"
[1] "Topic = Case\nStudies"
[1] "Emphasis = Mathematical\nDerivations"
[1] ""
[1] "Length = 1-25 pp"
[1] "Depth = Intermediate"
[1] "Topic = Modeling\nTechniques"
[1] "Emphasis = Mathematical\nConcepts"
[1] ""
```

6.3 Regression Models

In some applications we are able to characterize the behavior of the baseline utilities in each alternative-agent pairing with a collection of real-valued variables,

$$x_{nj} = (x_{nj1}, \ldots, x_{njK}).$$

Jointly modeling the observed choices and these characterizing variables and then making a sequence of nontrivial assumptions takes us to discrete choice regression models.

6.3.1 Variate-Covariate Modeling

From a variate-covariate modeling perspective, the choices y_m are natural *variates* while the auxiliary variables are natural *covariates*. Specifically, most discrete choice applications are interested in predicting unobserved choices given observed values for the x_{nj}.

In general the probabilistic utility model can vary these with covariates,

$$p(u_{n1}, \ldots, u_{nJ} \mid x_{n1}, \ldots, x_{nJ}, \theta),$$

which then leads to covariate-dependent choice probabilities,

$$q_{nj}(x_{n1}, \ldots, x_{nJ}, \theta).$$

Ultimately, the joint variate-covariate model for a particular observation becomes

$$\begin{aligned} &p(y_m, x_{n(m)1}, \ldots, x_{n(m)J}, \theta) \\ &\quad = p(y_m \mid q_{n(m)y_m}(x_{n(m)1}, \ldots, x_{n(m)J}, \theta)) \\ &\quad\quad p(x_{n(m)1}, \ldots, x_{n(m)J}, \theta)\, p(\theta). \end{aligned}$$

To move towards a regression model we have assume that there is no confounding, so that the observational choice model and the covariate model are configured by *distinct* parameters,

$$\begin{aligned} p(y_m, x_{n(m)1}, \ldots, x_{n(m)J}, \eta, \gamma) \\ = p(y_m \mid q_{n(m)y_m}(x_{n(m)1}, \ldots, x_{n(m)J}, \eta))\, p(\eta) \\ p(x_{n(m)1}, \ldots, x_{n(m)J}, \gamma)\, p(\gamma). \end{aligned}$$

As always, we cannot take for granted this assumption of negligible confounding. If, for instance, agent choice behavior and agent characteristics jointly vary across different cohorts then any model that ignores confounding will generalize poorly across cohorts. The infamous Berry, Levinsohn & Pakes model (Berry, Levinsohn, and Pakes 1995) is an example where modeling confounding is critical.

Because we're interested in only making conditional predictions for choices given observed covariates, the parameter separation afforded by a lack of confounding allows us to ignore the covariate model entirely. In other words the relevant inferences and predictions require working with just the conditional variate model,

$$\begin{aligned} p(y_m, \eta \mid x_{n(m)1}, \ldots, x_{n(m)J}) \\ = p(y_m \mid q_{n(m)y_m}(x_{n(m)1}, \ldots, x_{n(m)J}, \eta))\, p(\eta). \end{aligned}$$

Finally we assume that the covariates configure the probabilistic utility model through only baseline utility parameters. Specifically, we assume that the relationship between the baseline utilities and the covariates is defined by deterministic baseline utility functions,

$$p(u_{n1}, \ldots, u_{nJ} \mid \mu_{n1} = f(x_{n1}, \eta_{n1}), \ldots, \mu_{nJ} = f(x_{nJ}, \eta_{nJ})).$$

This, in turn, results in the choice probabilities

$$q_{nj}(\mu_{n1} = f(x_{n1}, \eta_{n1}), \ldots, \mu_{nJ} = f(x_{nJ}, \eta_{nJ})).$$

Note that, in general, all of the baseline utilities could vary with all of the covariates. Here we are making the common assumption that the baseline utility for each agent-alternative pair is configured by only the covariates associated with that pair.

6.3.2 Agent, Alternative, and Agent-Alternative Covariates

So far we have assumed that each agent-alternative pair is characterized by distinct covariate configurations, x_{nj}. In many applications, however, some of the available covariates inform the behavior of not complete pairs but rather individual agents or individual alternatives.

To account for these possibilities, we can always write each covariate in terms of agent-specific components, alternative-specific components, and pair-specific components,

$$(x_n, x_j, x_{nj}).$$

As before, each of these components can themselves be comprised of multiple components,

$$\begin{aligned} x_n &= (x_{n1}, \ldots, x_{nK}) \\ x_j &= (x_{j1}, \ldots, x_{jK'}) \\ x_{nj} &= (x_{nj1}, \ldots, x_{njK''}). \end{aligned}$$

For example, the purchasing choices of a consumer are often informed by the income of the consumer and the price of the available products. Consequently we might consider consumer income i_n an agent covariate and product price c_j an alternative covariate. At the same time, however, consumer behavior might be more sensitive to *relative* prices. In this case we might consider instead the covariate

$$x_{nj} = \frac{c_j}{i_n}$$

which is unique to each agent-alternative pair.

The structure of the covariates often informs the structure of the baseline utility functions. For example we might assume that the baseline utility functions additively decompose into functions for each type of covariate,

$$\begin{aligned} \mu_{nj} &= f(x_{nj}, \eta_{nj}) \\ &= f_1(x_n, \eta_n) + f_2(x_j, \eta_j) + f_3(x_{nj}, \eta_{nj}). \end{aligned}$$

Then f_1, f_2, and f_3 might further decompose into functions for each component of x_n, x_j, and x_{nj}.

6.3.3 Baseline Utility Functional Models

The functional models for the baseline utilities provide ample opportunity for incorporating domain expertise about choice behavior in any given application. For example, if we believe that the benefits or costs of particular alternatives cannot increase to arbitrarily large values then we might engineer non-linear functional models that saturate for sufficiently large inputs.

When domain expertise is more sparse then we can also appeal to more flexible functional models, such as spline models or gaussian process models. That said, we typically have to introduce some regularization when implementing these models to suppress particularly unreasonable functional behaviors.

While baseline utility functions are typically nonlinear, they can often be *locally* well-approximated by linear functional models. One way to employ linear models effectively is to appeal to Taylor approximation theory. From this perspective the parameters of the linear model feature explicit interpretations, each model particular features of the exact functional behavior. The Taylor approximation perspective is particularly useful for validating the

adequacy of a linear functional model and, when necessarily, incorporate more sophisticated functional behaviors with higher-order contributions.

That said, in some applications the engineering of an adequate baseline utility functional model is more productive if we first transform the exact function, Taylor approximate the transformed functional model, and then transform back to the original baseline utility outputs. This general Taylor modeling approach allows, for example, the accommodation of constraints on the baseline utility values.

6.3.4 Relative Baseline Utility Functional Models

As have repeatedly seen, observed choice data cannot inform all of the baseline utilities in most discrete choice models. To avoid awkward redundancies we have to implement these discrete choice models using relative baseline utilities. This transition to relative baseline utilities has interesting, and often beneficial, consequences to the implementation of baseline utility functional models.

When we anchor the baseline utilities to the behavior of a distinguished alternative j', the relative baseline utilities become

$$\begin{aligned} \delta_{nj} &= \mu_{nj} - \mu_{nj'} \\ &= f(\theta_{nj}, x_{nj}) - f(\theta_{nj'}, x_{nj'}). \end{aligned}$$

For many functional models these differences

$$f(\theta_{nj}, x_{nj}) - f(\theta_{nj'}, x_{nj'}),$$

are easier to model than $f(\theta_{nj}, x_{nj})$ itself.

Moreover, the distinguished alternative does not have to be an actual alternative in a particular choice subset. We can always implicitly define a hypothetical alternative with the choice of a reference covariate configuration,

$$x_0 = (x_{01}, \ldots, x_{0K}).$$

In this case the relative baseline utilities become

$$\begin{aligned} \delta_{nj} &= \mu_{nj} - \mu_0 \\ &= f(\theta_{nj}, x_{nj}) - f(\theta_0, x_0) \\ &\equiv g(\theta_{nj}, x_{nj}). \end{aligned}$$

This relative functional model $g(\theta_{nj}, x_{nj})$ not only is often easier to engineer but also can be applied to arbitrary choice subsets.

For example, let's say that we're working with the linear model

$$\mu_{nj} = \alpha + \sum_{k=1}^{K} \beta_{1,k} \, x_{nk} + \sum_{k'=1}^{K'} \beta_{2,k'} \, x_{jk'} + \sum_{k''=1}^{K''} \beta_{3,k''} \, x_{njk''}.$$

If for each agent we define an anchor alternative by specifying the reference covariate values

$$\begin{aligned} &(x_{n1}, \ldots, x_{nK}, \\ &\, x_{0,1}, \ldots, x_{0,K'}, \\ &\, x_{0,n1}, \ldots, x_{0,nK''}), \end{aligned}$$

then the relative baseline utilities become

$$\begin{aligned} \delta_{nj} &= \mu_{nj} - \mu_{n0} \\ &= \alpha + \sum_{k=1}^{K} \beta_{1,k} \, x_{nk} + \sum_{k'=1}^{K'} \beta_{2,k'} \, x_{jk'} + \sum_{k''=1}^{K''} \beta_{3,k''} \, x_{njk''} \\ &- \alpha + \sum_{k=1}^{K} \beta_{1,k} \, x_{nk} + \sum_{k'=1}^{K'} \beta_{2,k'} \, x_{0,k'} + \sum_{k''=1}^{K''} \beta_{3,k''} \, x_{0,nk''} \\ &= \sum_{k'=1}^{K'} \beta_{2,k'} \, (x_{jk'} - x_{0,k'}) + \sum_{k''=1}^{K''} \beta_{3,k''} \, (x_{njk''} - x_{0,nk''}). \end{aligned}$$

Let's review everything that happened when we moved from baseline utilities to relative baseline utilities. Firstly, the common parameter α canceled identically, which we means that are free to ignore it when implementing the model. Because we assumed common agent covariates, the second term also cancels and hence can also be ignored. The two terms that do remain,

$$\sum_{k'=1}^{K'} \beta_{2,k'} \, (x_{jk'} - x_{0,k'})$$

and

$$\sum_{k''=1}^{K''} \beta_{3,k''} \, (x_{njk''} - x_{0,nk''}),$$

are now written in terms of *deviations* of the covariates from the reference values,

$$\delta x_{jk'} = x_{jk'} - x_{0,k'}.$$

These deviations are especially well-suited to the Taylor approximation framework. Perhaps more importantly, they also admit explicit interpretations. For example, if the appeal of the jth alternative should increase when the k'th alternative covariate increases *above the reference value*,

$$x_{jk'} > x_{0,k'},$$

then $\beta_{2,k'}$ should be positive. More quantitatively, $\beta_{2,k'}$ defines exactly how much more appealing the jth alternative becomes when

$$\delta x_{jk'} = x_{jk'} - x_{0,k'} = 1.$$

6.4 Perfect Replacements

When choice probabilities are completely determined by baseline utilities, any modification that leaves the baseline utilities unchanged also leaves the choice probabilities unchanged. In other words, perfect replacements are defined by the invariances of the baseline utilities. If we derive baseline utilities from alternative characteristics ,then perfect replacements are defined by the *level sets* of the deterministic baseline utility functions.

For example, consider a first-order factor model where the baseline utility μ_{nj} is given by a sum of factor-specific contributions specific to each alternative,

$$\mu_{nj} = \alpha_0 + \sum_{k=1}^{K} \alpha^k_{l_k(j)}.$$

Any alternative j'' with

$$\begin{aligned} \mu_{nj} &= \mu_{nj''} \\ \alpha_0 + \sum_{k=1}^{K} \alpha^k_{l_k(j)} &= \alpha_0 + \sum_{k=1}^{K} \alpha^k_{l_k(j'')} \\ \sum_{k=1}^{K} \alpha^k_{l_k(j)} &= \sum_{k=1}^{K} \alpha^k_{l_k(j'')} \end{aligned}$$

defines a perfect replacement for the jth alternative.

Note that scaled relative baseline utilities define the exact same perfect replacements,

$$\begin{aligned}
\omega_{nj} &= \omega_{nj''} \\
\sum_{k=1}^{K} \upsilon^{k}_{l_k(j)} &= \sum_{k=1}^{K} \upsilon^{k}_{l_k(j'')} \\
\sum_{k=1}^{K} \frac{\alpha^{k}_{l_k(j)} - \alpha^{k}_{l_k(j')}}{\sigma_n} &= \sum_{k=1}^{K} \frac{\alpha^{k}_{l_k(j'')} - \alpha^{k}_{l_k(j')}}{\sigma_n} \\
\frac{1}{\sigma_n} \left[\sum_{k=1}^{K} \alpha^{k}_{l_k(j)} - \sum_{k=1}^{K} \alpha^{k}_{l_k(j')} \right] &= \frac{1}{\sigma_n} \left[\sum_{k=1}^{K} \alpha^{k}_{l_k(j'')} - \sum_{k=1}^{K} \alpha^{k}_{l_k(j')} \right] \\
\sum_{k=1}^{K} \alpha^{k}_{l_k(j)} - \sum_{k=1}^{K} \alpha^{k}_{l_k(j')} &= \sum_{k=1}^{K} \alpha^{k}_{l_k(j'')} - \sum_{k=1}^{K} \alpha^{k}_{l_k(j')} \\
\sum_{k=1}^{K} \alpha^{k}_{l_k(j)} &= \sum_{k=1}^{K} \alpha^{k}_{l_k(j'')}.
\end{aligned}$$

Consequently we can always make inferences about perfect replacements directly from non-redundant model implementations.

Perfect replacements are often considered in terms of tradeoffs. If the k_1st contribution to the baseline utility for some new alternative j'' is larger than k_1st contribution to the jth alternative,

$$\alpha^{k_1}_{l_{k_1}(j'')} > \alpha^{k_1}_{l_{k_1}(j)},$$

but the k_2st contribution is smaller,

$$\alpha^{k_2}_{l_{k_2}(j'')} < \alpha^{k_2}_{l_{k_2}(j)},$$

such that

$$\alpha^{k_1}_{l_{k_1}(j)} + \alpha^{k_2}_{l_{k_2}(j)} = \alpha^{k_1}_{l_{k_1}(j'')} + \alpha^{k_2}_{l_{k_2}(j'')},$$

then the two alternatives would be perfect replacements for each other.

In a regression baseline utility model perfect replacements are defined by covariate configurations that fall into the same level set of the baseline configuration function,

$$\begin{aligned}
\mu_{nj} &= \mu_{nj''} \\
f(x_{nj}, \eta_{nj}) &= f(x_{nj''}, \eta_{nj''}).
\end{aligned}$$

Again this is often framed in terms of tradeoffs, with changes to some alternative covariates compensating for changes to other alternative covariates so that the output baseline utility is fixed.

Completely characterizing all of the perfect replacements in a regression discrete choice model requires navigating the level sets of baseline utility functions. When the geometry of the

level sets are complicated, it can be easier to reason about *infinitesimal* changes that define "nearby' ' perfect replacements.

In **Section Ref** we saw that for any two continuous configuration variables z_1 and z_2, perfect replacements are defined by the differential constraint

$$\frac{\mathrm{d}z_1}{\mathrm{d}z_2} = -\frac{\partial q_{nj}}{\partial z_2} \left(\frac{\partial q_{nj}}{\partial z_1} \right)^{-1}.$$

For regression baseline utility models, this constraint characterizes the local shape of the baseline utility function level sets. Moreover, in the regression setting we can use the chain rule to simplify this constraint in terms of partial derivatives of the baseline utility functions,

$$\frac{\mathrm{d}z_1}{\mathrm{d}z_2} = -\frac{\partial \mu_{nj}}{\partial z_2} \left(\frac{\partial \mu_{nj}}{\partial z_1} \right)^{-1}.$$

For example, let's say that we are working with a linear baseline utility functional model,

$$\mu_{nj} = \beta_n \, v_j - \gamma_n \, c_j,$$

where v_j is a covariate that quantifies some desirable property of the jth alternative and c_j is a covariate that quantifies its cost. The parameters β_n and γ_n quantify how sensitive each consumer is to each of these alternative behaviors.

Now let's say that we are interested in how much we would have to improve an alternative in order to justify increasing its cost by a certain amount. We can answer this question by applying our differential constraint,

$$\begin{aligned} \frac{\mathrm{d}v_j}{\mathrm{d}c_j} &= -\frac{\partial \mu_{nj}}{\partial c_j} \left(\frac{\partial \mu_{nj}}{\partial v_j} \right)^{-1} \\ &= -(-\gamma_n) \, (\beta_n)^{-1} \\ &= \frac{\gamma_n}{\beta_n}. \end{aligned}$$

In particular, if we wanted to double the cost of the jth consumer without affecting the interest of the nth consumer then we would need to increase v_j by

$$2 \, \frac{\gamma_n}{\beta_n}.$$

Because they quantify how much the cost of an alternative could be increased without affecting a particular consumer's behavior, quantities like

$$\mathrm{d}v_j / \mathrm{d}c_j = \gamma_n / \beta_n$$

are often referred to as an agent's *willingness to pay*.

6.5 Regression Model Demonstration

Let's apply everything that we've learned to this point to understand how people in a large city their regular local bar. Each bar is characterized by a collection of categorical factors covariates that we will treat as continuous. Two categorical factors describe whether or not a bar offer lives entertainment and then what, if any, food options it provides. The five covariates quantify distance from an individual, the beer menu, and the seating arrangements.

```
factor_names <- c('Entertainment', 'Food')

factor_levels <-
  list( list('No', 'Yes'),
        list('None', 'BYO', 'Snacks', 'Comfort', 'Elevated') )

covar_names <- c('Walking\nDistance (min)',
                 'Ave Beer Cost /\nDaily Income (%)',
                 'Number of\nBeers',
                 'Number of\nIndoor Seats',
                 'Number of\nOutdoor Seats')
```

Notice that the walking distance and relative beer cost are unique to each agent-alternative pair, where as the other three covariates vary across only alternatives. In this case there are no agent covariates.

6.5.1 Explore Data

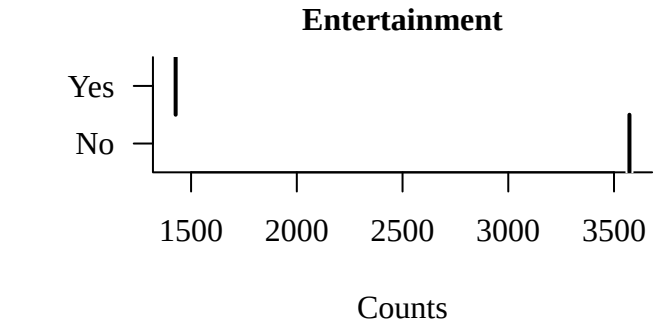
For this exercise let's assume that we have surveyed individuals in various neighborhoods across the city. Each individual was asked to list all of the local bars they know, choose the local bar they visit most regularly, and then estimate the time it takes to walk to the chosen bar from their home as well as how much a beer costs relative to their daily income. At the same time we collected information on each of the neighborhoods bars.

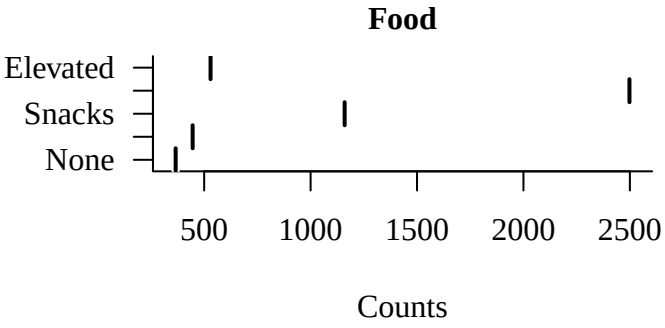
```
data <- read_rdump('data/regression.data.R')
```

There are lots of ways that we could examine all of this data, but here we'll look at the behavior of the factor levels and covariates for the chosen alternatives.

We can see, for instance, that live entertainment is relatively rare amongst the chosen bars. Note that this doesn't necessarily imply that no entertainment is preferred over live entertainment; we'd have to consider the base rates of each to make a judgement like that.

```
chosen_level_counts <- c()
for (f in 1:data$F) {
  L <- data$factor_end[f] - data$factor_start[f] + 1

  factor_chosen_levels <-
    sapply(1:data$M,
           function(m) data$alt_levels[data$alt_start[m] +
                                       data$choice[m] - 1, f])

  factor_chosen_level_counts <-
    hist(factor_chosen_levels,
         breaks=seq(0.5, L + 0.5, 1),
         plot=FALSE)$counts

  chosen_level_counts <-  c(chosen_level_counts,
                            factor_chosen_level_counts)
}


par(mfrow=c(2, 1), mar = c(5, 6, 2, 1))

for (f in 1:data$F) {
  util$plot_disc_vals_vert(
    chosen_level_counts[data$factor_start[f]:data$factor_end[f]],
    yticklabs=factor_levels[[f]],
    xlab='Counts',
    main=factor_names[f]
  )
}
```



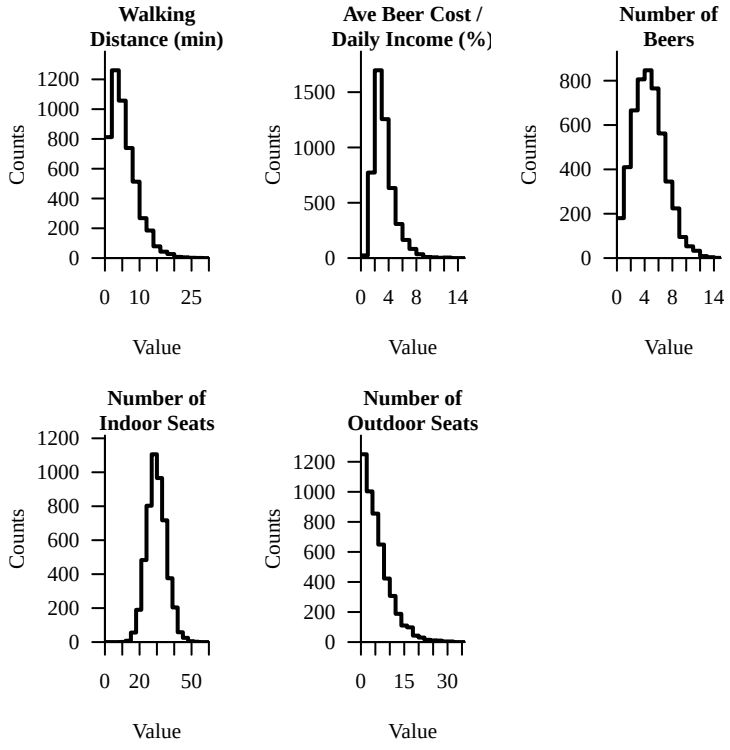


Similarly outdoor seats are uncommon compared to indoor seats in the chosen bars, but that doesn't mean that outdoor seats aren't a draw for customers.

```
chosen_x <-
  sapply(1:data$I,
         function(i)
         sapply(1:data$M,
                function(m) data$X[data$alt_start[m] +
                                   data$choice[m] - 1, i]))

bin_configs <- list( c(0, 30, 2),
                     c(0, 15, 1),
                     c(0, 15, 1),
                     c(0, 60, 3),
                     c(0, 36, 2) )

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

for (i in 1:data$I) {
  util$plot_line_hist(chosen_x[,i],
                      bin_configs[[i]][1],
                      bin_configs[[i]][2],
                      bin_configs[[i]][3],
```

```
                      xlab='Value',
                      main=covar_names[i])
}
```

```
Warning in check_bin_containment(bin_min, bin_max, values): 1 value (0.0%) fell
above the binning.
```



6.5.2 Specifying An Observational Model

Here we'll assume an independent Gumbel model with homogeneous agent scales,

$$q_{nj} = \frac{e^{\frac{\mu_{nj}}{\sigma}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}}.$$

The baseline utilities will be derived from first-order factor and covariate contributions,

$$\mu_{nj} = \alpha_0 + \sum_{f=1}^{F} \alpha^f_{l_f(j)} + \sum_{i=1}^{I} \beta_i x_{nji}.$$

To eliminate any redundancies, however, we'll actually implement the model with relative baseline utilities defined with respect to a hypothetical anchor pairing,

$$\begin{aligned}
\omega_{nj} &= \frac{\mu_{nj} - \mu_{n'j'}}{\sigma} \\
&= \frac{1}{\sigma} \left[\sum_{f=1}^{F} \left(\alpha^f_{l_f(j)} - \alpha^f_{l_f(j')} \right) + \sum_{i=1}^{I} \beta_i \left(x_{nji} - x_{n'j'i} \right) \right] \\
&= \sum_{f=1}^{F} \frac{\alpha^f_{l_f(j)} - \alpha^f_{l_f(j')}}{\sigma} + \sum_{i=1}^{I} \frac{\beta_i}{\sigma} \left(x_{nji} - x_{n'j'i} \right) \\
&\equiv \sum_{f=1}^{F} \upsilon^f_{l_f(j)} - \alpha^f_{l_f(j')} + \sum_{i=1}^{I} \gamma_i \left(x_{nji} - x_{n'j'i} \right).
\end{aligned}$$

We are free to configure this anchor configuration however we like, but in practice the more interpretable the anchor configuration is the more interpretable the rest of the model we will be. Let's say that in this case the anchor configuration is defined by a particular individual/bar pairing of interest.

```
# Anchoring
data$x0 <- c(5, 0.05, 5, 36, 8)
data$factor_anchors <- c(1, 3)
```

6.5.3 Specifying A Prior Model

We can motivate a prior model for the relative factor contributions exactly as we did in Section 6.2.3. In particular, if we once again want to allow for an order of magnitude proportional variation in the choice probabilities,

$$-\log 10 \lessapprox \upsilon^f_{l(j_2)} \lessapprox +\log 10,$$

then we can use the containment prior model

$$\begin{aligned}
p\left(\upsilon^f_{l(j_2)}\right) &= \text{normal}\left(\upsilon^f_{l(j_2)} \,\middle|\, 0, \frac{\log 10}{2.32}\right) \\
&\approx \text{normal}\left(\upsilon^f_{l(j_2)} \,\middle|\, 0, 1\right).
\end{aligned}$$

An appropriate prior model for the scaled slopes,

$$\gamma_i = \frac{\beta_i}{\sigma},$$

requires just a bit more care. In this case we have to consider the variation in the choice probabilities given a particular change in the covariate deviations,

$$-\log l \lessapprox \gamma_i \, \delta x_i \lessapprox +\log u,$$

or equivalently

$$-\frac{\log l}{\delta x_i} \lessapprox \gamma_i \lessapprox +\frac{\log u}{\delta x_i}.$$

Assuming the same order of magnitude variation, here we'll use the component prior models

$$\begin{aligned} p\left(\gamma_i\right) &= \text{normal}\left(\gamma_i \,\middle|\, 0, \frac{\log 10}{2.32}\frac{1}{\delta x_i}\right) \\ &\approx \text{normal}\left(\gamma_i \,\middle|\, 0, \frac{1}{\delta x_i}.\right). \end{aligned}$$

6.5.4 Posterior Quantification

With our Bayesian model fully specified we just have to implement it as a `Stan` program and then let Hamiltonian Monte Carlo do it's magic.

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

Fortunately the diagnostics don't show any signs of incomplete posterior quantification.

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

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

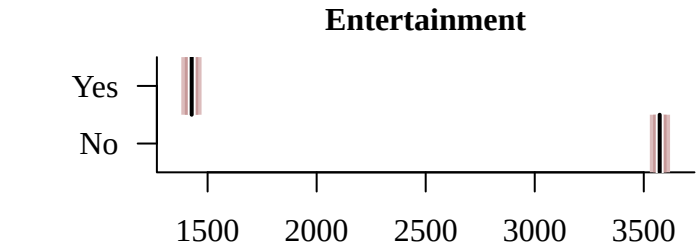
```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('gamma',
                                         'upsilon_free'),
                                       TRUE)
util$check_all_expectand_diagnostics(base_samples)
```

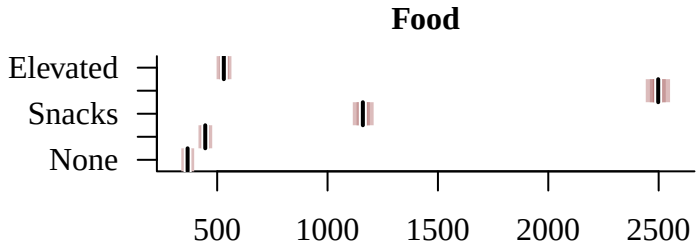
```
All expectands checked appear to be behaving well enough for reliable
Markov chain Monte Carlo estimation.
```

6.5.5 Posterior Retrodictive Checks

Scanning through all of the summaries that we considered above, it looks like the model is able to reproduce all of the corresponding features of the observed factor levels. As with our retrodictive checks in Section 6.2.5, however, these summary statistics aren't the most sensitive to higher-order interaction contributions that we might be missing.

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

for (f in 1:data$F) {
  obs <-
    chosen_level_counts[data$factor_start[f]:data$factor_end[f]]
  names <- sapply(data$factor_start[f]:data$factor_end[f],
                  function(ll)
                  paste0('chosen_level_counts_pred[', ll, ']'))
  util$plot_disc_pushforward_quantiles_vert(
    samples, names,
    baseline_values=obs,
    yticklabs=factor_levels[[f]],
    xlab='Counts',
    main=factor_names[f]
  )
}
```



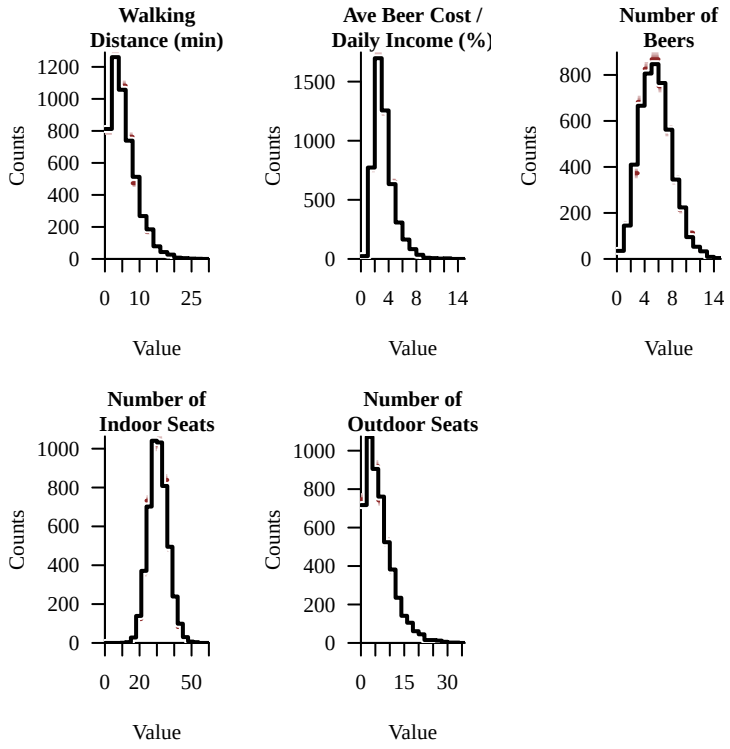


The retrodictive behavior of the chosen covariates also looks great.

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

for (i in 1:data$I) {
  names <- sapply(1:data$M,
                  function(m)
                  paste0('chosen_x_pred[', m, ',', i, ']'))
  filtered_samples <- util$filter_expectands(samples, names)

  util$plot_hist_quantiles(filtered_samples, 'chosen_x_pred',
                           bin_configs[[i]][1],
                           bin_configs[[i]][2],
                           bin_configs[[i]][3],
                           baseline_values=chosen_x[,i],
                           xlab='Value',
                           main=covar_names[i])
}
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 3148 predictive values (0.0%) fell above the binning.
```

```
Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2719 predictive values (0.0%) fell above the binning.

Warning in check_bin_containment(bin_min, bin_max, baseline_values, "observed
value"): 1 observed value (0.0%) fell above the binning.

Warning in check_bin_containment(bin_min, bin_max, collapsed_values,
"predictive value"): 2597 predictive values (0.0%) fell above the binning.
```



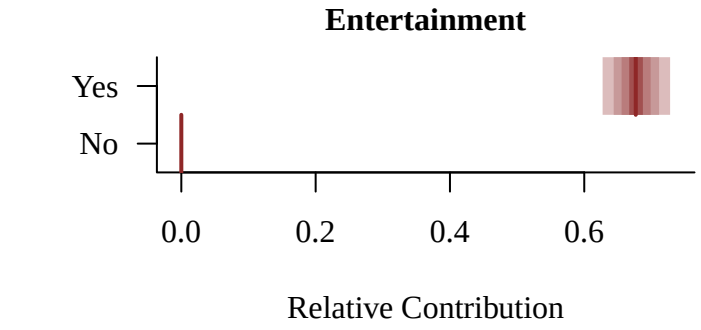
That said, it's worth a second to consider how we might interpret less agreeable retrodictive comparisons. For example, what would happen if the true x_{nji} dependence was not linear but rather quadratic? In that case the baseline utility μ_{nj} will initially increase as x_{nji} increases, but then eventually turn over and start to decrease. This results in a narrow range of x_{nji} values where μ_{nj} is large enough for j to be chosen over the alternatives. Consequently we would expect to see the observed distribution of chosen x_{nji} values to be *narrower* than the posterior predictions from the linear model.

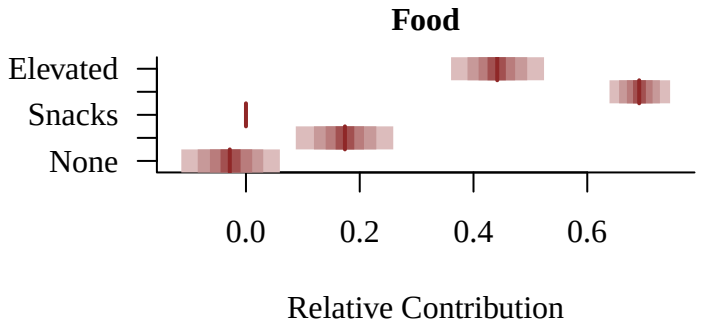
6.5.6 Posterior Inferences

With no sign of model inadequacy we have some confidence that our posterior inferences actually quantify meaningful features of the true data generating process.

Although live entertainment are elevated food offerings are rare in the observed choices, they are quite appealing to the people surveyed.

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

for (f in 1:data$F) {
  names <- sapply(data$factor_start[f]:data$factor_end[f],
                  function(ll) paste0('upsilon[', ll, ']'))
  util$plot_disc_pushforward_quantiles_vert(
    samples, names,
    yticklabs=factor_levels[[f]],
    xlab="Relative Contribution",
    main=factor_names[f]
  )
}
```

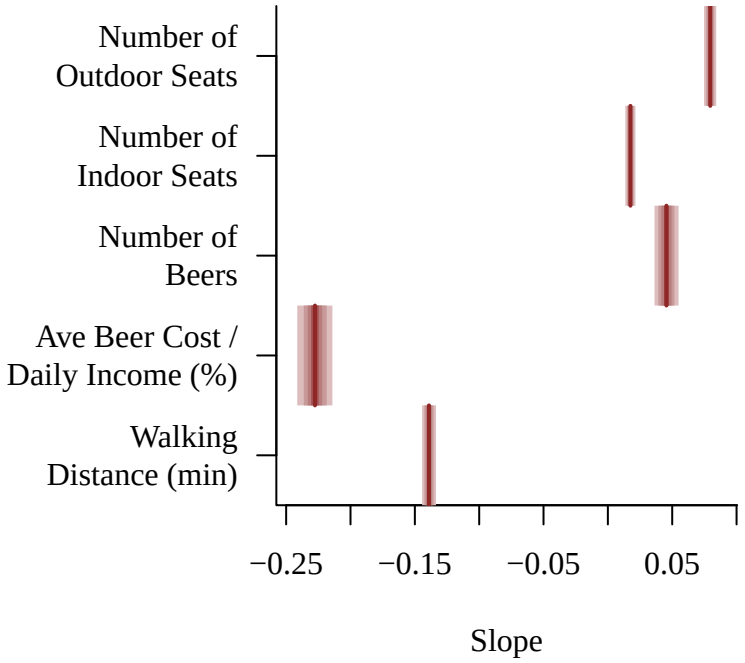




Similarly the baseline utilities are most sensitive to changes in the number of outdoor seats and, at least for the particular cohort surveyed, relative insensitive to price.

```
par(mfrow=c(1, 1), mar = c(5, 7, 2, 1))

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

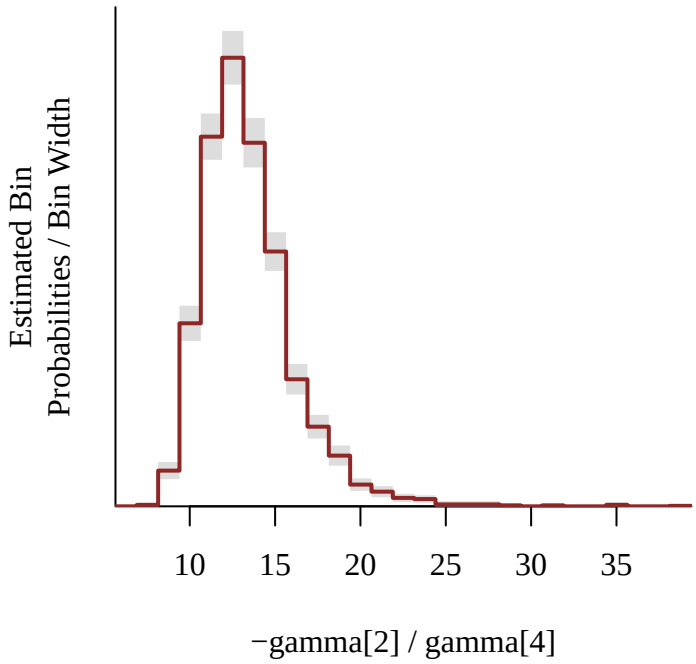
```
                function(i) paste0('gamma[', i, ']'))
util$plot_disc_pushforward_quantiles_vert(samples, names,
                                          yticklabs=covar_names,
                                          xlab="Slope")
```



Even better, we can put these posterior inferences to start answering relevant questions about alternatives. For example, we might be interested in what improvements we could make to a bar that would allow for increase in beer prices without losing customers.

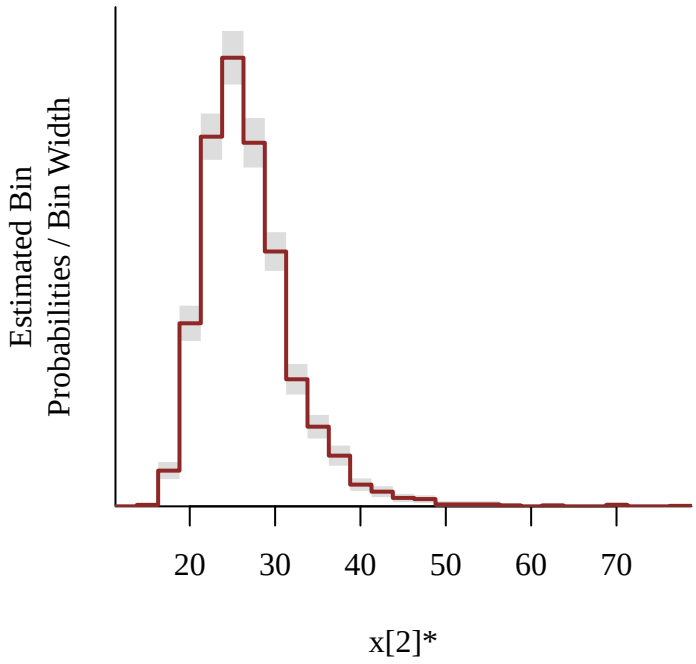
Because our model is linear we can characterize the trade-off between relative beer price and the number of indoor seats with a ratio of the corresponding slopes.

```
rs <- util$eval_expectand_pushforward(samples,
                                      function(x1, x2) -x1 / x2,
                                      list('x1'='gamma[2]',
                                           'x2'='gamma[4]'))

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

util$plot_expectand_pushforward(
  rs, 25, display_name='-gamma[2] / gamma[4]'
)
```



Specifically we might quantify how much the relative beer price can increase if we double the number of indoor seats.

```
rs <-
  util$eval_expectand_pushforward(
    samples,
    function(x1, x2) (-x1 / x2) * 2,
    list('x1'='gamma[2]', 'x2'='gamma[4]')
  )

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

util$plot_expectand_pushforward(rs, 25,
                                display_name='x[2]*')
```



In this case it looks like doubling the number of indoor seats could justify increasing the relative beer prices by about 25%.

Converting this increase in relative beer prices to an actionable increase in absolute beer prices requires knowing something about the income of the bar's customers,

$$\frac{\text{New Ave Beer Cost}}{\text{Daily Income}} - \frac{\text{Old Ave Beer Cost}}{\text{Daily Income}} \approx 0.25$$
$$\text{New Ave Beer Cost} - \text{Old Ave Beer Cost} \approx 0.25\,\text{Daily Income}.$$

Consequently we'd have to do some demographic research before attempting any menu changes.

7 Conclusion

Discrete choice modeling is a general technique for analyzing decisions made by individual agents. While the implementation of a general discrete choice model is limited by analytic tractability, there are enough exceptions are available to allow some rich modeling opportunities. This is especially true when we expand an discrete choice model into continuous mixtures.

Some of the probabilistic utility models that we considered in this chapter can be generalized further with mathematical care. The nested Gumbel model, for example, can be generalized to

multiple layers of hierarchical nests. Moreover, instead of just mixing an independent Gumbel model we can also mix an already coupled probabilistic utility model, and so on.

Much of the art of discrete choice modeling lies in the development of appropriate baseline utility models, with the best models being specific to each particular application. Ultimately we are responsible for determining what phenomena can influence decision making behavior in any given analysis.

A: Onerous Calculations

To keep the main text from being overwhelmed by mathematical derivations, I have sequestered most the more onerous calculations into this appendix. For those who trust my work, these calculations can be skipped entirely. For those interested in how the results can be derived, read on.

A.1 Independent Gumbel Model Calculations

Recall that the Gumbel family of probability density functions is given by

$$\text{Gumbel}(x \mid \mu, \sigma) = \frac{1}{\sigma} \exp\left(-\frac{x - \mu}{\sigma} - e^{-\frac{x-\mu}{\sigma}}\right),$$

where x is supported over the entire real line.

The corresponding cumulative distribution functions are given by relatively straightforward integrals. By definition we have

$$\begin{aligned}
\Pi_{\text{Gumbel}}(x \mid \mu, \sigma) &= \int_{-\infty}^{x} \mathrm{d}x' \frac{1}{\sigma} \exp\left(-\frac{x' - \mu}{\sigma} - e^{-\frac{x'-\mu}{\sigma}}\right), \\
&= \int_{-\infty}^{x} \mathrm{d}x' \frac{1}{\sigma} e^{-\frac{x'-\mu}{\sigma}} \exp\left(-e^{-\frac{x'-\mu}{\sigma}}\right).
\end{aligned}$$

Now we can make the substitution

$$z = \phi(x') = -e^{-\frac{x'-\mu}{\sigma}}$$

with

$$\mathrm{d}z = \mathrm{d}x' \, \frac{1}{\sigma} \, e^{-\frac{x'-\mu}{\sigma}}$$

and

$$\begin{aligned}
\phi(-\infty) &= -\infty \\
\phi(x) &= e^{-\frac{x-\mu}{\sigma}}.
\end{aligned}$$

This gives

$$\begin{aligned}
\Pi_{\text{Gumbel}}(x \mid \mu, \sigma) &= \int_{-\infty}^{x} \mathrm{d}x' \, \frac{1}{\sigma} \, e^{-\frac{x' - \mu}{\sigma}} \exp\left(-e^{-\frac{x' - \mu}{\sigma}}\right) \\
&= \int_{-\infty}^{-e^{-\frac{x' - \mu}{\sigma}}} \mathrm{d}z \, \exp(z) \\
&= \left[\exp\left(-e^{-\frac{x' - \mu}{\sigma}}\right) - \exp(-\infty) \right] \\
&= \exp\left(-e^{-\frac{x' - \mu}{\sigma}}\right).
\end{aligned}$$

The independent Gumbel model is then given by the probabilistic utility model

$$p(u_{n1}, \ldots, u_{nJ}) = \prod_{j=1}^{J} \text{Gumbel}(u_{nj} \mid \mu_{nj}, \sigma_n),$$

or, equivalently,

$$\begin{aligned}
\Pi(u_{n1}, \ldots, u_{nJ}) &= \prod_{j=1}^{J} \Pi_{\text{Gumbel}}(u_{nj} \mid \mu_{nj}, \sigma_n), \\
&= \prod_{j=1}^{J} \exp\left(-e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}}\right) \\
&= \exp\left(-\sum_{j=1}^{J} e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}}\right)
\end{aligned}$$

Conveniently it's not too hard to calculate the partial derivative of this joint cumulative distribution function with respect to any one utility,

$$\begin{aligned}
\frac{\partial \Pi}{\partial u_{nj}}(u_{n1}, \ldots, u_{nJ}) &= \frac{\partial}{\partial u_{nj}} \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj'} - \mu_{nj'}}{\sigma_n}}\right) \\
&= \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj'} - \mu_{nj'}}{\sigma_n}}\right) e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n}.
\end{aligned}$$

Evaluating each argument on the same component utility and then isolate the corresponding

terms as much as possible gives

$$\begin{aligned}\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \dots, u_{nj}) &= \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj}-\mu_{nj'}}{\sigma_n}}\right) e^{-\frac{u_{nj}-\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} \\ &= \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj}}{\sigma_n}} e^{\frac{\mu_{nj'}}{\sigma_n}}\right) e^{-\frac{u_{nj}}{\sigma_n}} e^{\frac{\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} \\ &= e^{\frac{\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}} \sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}\right).\end{aligned}$$

Finally, the choice probabilities are given by integrating this differential object,

$$\begin{aligned}q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \frac{\partial \Pi}{\partial u_{jn}}(u_{nj}, \dots, u_{nj}) \\ &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{\frac{\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}} \sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}\right) \\ &= e^{\frac{\mu_{nj}}{\sigma_n}} \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}} C\right),\end{aligned}$$

where

$$C = \sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}.$$

To simplify this further we make the substitution

$$z = \phi(u_{nj}) = -e^{-\frac{u_{nj}}{\sigma_n}} C$$

with

$$\mathrm{d}z = C\, \mathrm{d}u_{nj} \frac{1}{\sigma_n} \exp\left(-\frac{u_{nj}}{\sigma_n}\right)$$

and

$$\begin{aligned}\phi(-\infty) &= -\infty \\ \phi(+\infty) &= 0.\end{aligned}$$

This gives

$$\begin{aligned}
q_{nj} &= e^{\frac{\mu_{nj}}{\sigma_n}} \int_{-\infty}^{+\infty} \mathrm{d}u_{nj} \, \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}} \, C\right) \\
&= e^{\frac{\mu_{nj}}{\sigma_n}} \, \frac{1}{C} \int_{-\infty}^{0} \mathrm{d}z \, \exp\left(z\right) \\
&= e^{\frac{\mu_{nj}}{\sigma_n}} \, \frac{1}{C} \Big[\exp\left(0\right) - \exp\left(-\infty\right) \Big] \\
&= e^{\frac{\mu_{nj}}{\sigma_n}} \, \frac{1}{C} \\
&= \frac{e^{\frac{\mu_{nj}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}},
\end{aligned}$$

which is exactly the softmax function applied to the scaled baseline utilities,

$$(q_{n1}, \ldots, q_{nJ}) = \mathrm{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \ldots, \frac{\mu_{nJ}}{\sigma_n}\right).$$

A.2 Independent Generalized Extreme Value Model Calculations

The Gumbel family of probability density functions is just one extreme value model. A natural question to consider is whether or not there are other extreme value models from which we can derive choice probabilities in closed form.

We can answer this question by considering the generalized extreme value family of probability density functions, which includes not only the Gumbel family but also all other extreme value families as special cases,

$$\mathrm{GEV}(x \mid \mu, \sigma, \xi) = \frac{1}{\sigma} \, t(x)^{\xi+1} \, \exp\left(-t(x)\right),$$

where

$$t(x) = \begin{cases} \left(1 + \xi \, \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}, & \xi \neq 0 \\ \exp\left(-\frac{x-\mu}{\sigma}\right), & \xi = 0 \end{cases}.$$

Note the Gumbel family is given by fixing $\xi = 0$.

Unlike the Gumbel model, the generalized extreme value model is not generally supported over the entire real line. If $\xi > 0$ then the generalized extreme value model is well-defined for only the range

$$\left[\mu - \frac{\sigma}{\xi}, +\infty\right),$$

while for $\xi < 0$ is it well-defined for only the range

$$\left(-\infty, \mu - \frac{\sigma}{\xi}\right].$$

Only when $\xi = 0$, in particular when it reduces to a Gumbel model, does the support extend out to the entire real line $(-\infty, +\infty)$.

Despite its added complexity, the generalized extreme value model still benefits from a straightforward cumulative distribution function. We just need to take advantage of the substitution

$$z = -t(x) = -\left(1 + \xi \frac{x - \mu}{\sigma}\right)^{-\frac{1}{\xi}}$$

with

$$\begin{aligned}
\mathrm{d}z &= \mathrm{d}x' \, \frac{1}{\xi} \frac{\xi}{\sigma} \, \left(1 + \xi \frac{x - \mu}{\sigma}\right)^{-\frac{1}{\xi} - 1} \\
&= \mathrm{d}x' \, \frac{1}{\sigma} \, \left(1 + \xi \frac{x - \mu}{\sigma}\right)^{-\frac{1}{\xi}(1 + \xi)} \\
&= \mathrm{d}x' \, \frac{1}{\sigma} \, t(x)^{\xi + 1}.
\end{aligned}$$

Being a little sloppy with the bounds, which again change with ξ, we have

$$\begin{aligned}
\Pi_{\mathrm{GEV}}(x \mid \mu, \sigma, \xi) &= \int \mathrm{d}x' \frac{1}{\sigma} \, t(x')^{\xi + 1} \, \exp\left(-t(x')\right) \\
&= \int \mathrm{d}z \, \exp\left(z\right) \\
&= \exp\left(-t(x)\right).
\end{aligned}$$

An independent generalized extreme value model for alternative utilities is then given by

$$p(u_{n1}, \ldots, u_{nJ}) = \prod_{j=1}^{J} \mathrm{GEV}(u_{nj} \mid \mu_{nj}, \sigma_n, \xi)$$

or, equivalently,

$$\begin{aligned}
\Pi(u_{n1}, \ldots, u_{nJ}) &= \prod_{j=1}^{J} \mathrm{GEV}(u_{nj} \mid \mu_{nj}, \sigma_n, \xi) \\
&= \prod_{j=1}^{J} \exp\left(-t\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n}\right)\right) \\
&= \exp\left(-\sum_{j=1}^{J} t\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n}\right)\right).
\end{aligned}$$

To derive choice probabilities we first have to compute

$$\begin{aligned}
\frac{\partial \Pi}{\partial u_{nj}}(u_{n1}, \ldots, u_{nJ}) &= \frac{\partial}{\partial u_{nj}} \exp\left(-\sum_{j'=1}^{J} t\left(\frac{u_{nj'} - \mu_{nj'}}{\sigma_n} \right) \right) \\
&= \exp\left(-\sum_{j'=1}^{J} t\left(\frac{u_{nj'} - \mu_{nj'}}{\sigma_n} \right) \right) \frac{\partial t}{\partial u_{nj}}\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n} \right) \frac{\xi}{\sigma_n} \\
&= \exp\left(-\sum_{j'=1}^{J} t\left(\frac{u_{nj'} - \mu_{nj'}}{\sigma_n} \right) \right) t\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n} \right)^{\xi+1} \frac{1}{\sigma_n}.
\end{aligned}$$

At this point we start to see the problems with generalizing beyond the Gumbel model. When we evaluate this partial derivative on a particular utility,

$$\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \ldots, u_{nj}) = \exp\left(-\sum_{j'=1}^{J} t\left(\frac{u_{nj} - \mu_{nj'}}{\sigma_n} \right) \right) t\left(\frac{u_{nj} - \mu_{nj}}{\sigma_n} \right)^{\xi+1} \frac{1}{\sigma_n},$$

we can no longer isolate the u_{nj} terms. In particular unless $\xi = 0$ there is no way to factor out u_{nj} in

$$\sum_{j'=1}^{J} t\left(\frac{u_{nj} - \mu_{nj'}}{\sigma_n} \right) = \sum_{j'=1}^{J} \left(1 + \xi \frac{u_{nj} - \mu_{nj'}}{\sigma_n} \right)^{-\frac{1}{\xi}}.$$

In other words, the Gumbel model is exceptionally amenable to analytic calculations amongst all generalized extreme value models. Consequently they are rarely considered for discrete choice models.

A.3 Nested Gumbel Model Calculations

Calculations for the nested Gumbel model are relatively straightforward, they just require very careful organization to avoid being overwhelmed by all of the possible terms.

A.3.1 Choice Probabilities

The first step in calculation the choice probabilities is, as always, differentiating the joint cumulative distribution function,

$$
\begin{aligned}
&\frac{\partial \Pi}{\partial u_{nj}}(u_{n1}, \ldots, u_{nJ}) \\
&\quad = \frac{\partial}{\partial u_{nj}} \exp\left(-\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \right) \\
&\quad = -\exp\left(-\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \right) \frac{\partial}{\partial u_{nj}} \sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \\
&\quad = -\exp\left(-\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \right) \frac{\partial}{\partial u_{nj}} \left(\sum_{j' \in B_{k(j)}} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}} \\
&\quad = -\exp\left(-\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \right) \\
&\qquad \lambda_{k(j)} \left(\sum_{j' \in B_{k(j)}} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)} - 1} \frac{-1}{\lambda_{k(j)}} e^{-\frac{u_{nj} - \mu_{nj}}{\lambda_{k(j)}}} \\
&\quad = \exp\left(-\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_k}} \right)^{\lambda_k} \right) \left(\sum_{j' \in B_{k(j)}} e^{-\frac{u_{nj'} - \mu_{nj'}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)} - 1} e^{-\frac{u_{nj} - \mu_{nj}}{\lambda_{k(j)}}}.
\end{aligned}
$$

Next evaluate each argument on the target utility and simplify,

$$
\begin{aligned}
&\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \ldots, u_{nj}) \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k} e^{-\frac{u_{nj}-\mu_{nj'}}{\lambda_k}}\right)^{\lambda_k}\right)\left(\sum_{j' \in B_{k(j)}} e^{-\frac{u_{nj}-\mu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1} e^{-\frac{u_{nj}-\mu_{nj}}{\lambda_{k(j)}}} \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k} e^{-\frac{u_{nj}}{\lambda_k}} e^{\frac{\mu_{nj'}}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \left(\sum_{j' \in B_{k(j)}} e^{-\frac{u_{nj}}{\lambda_{k(j)}}} e^{\frac{\mu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1} e^{-\frac{u_{nj}}{\lambda_{k(j)}}} e^{\frac{\mu_{nj}}{\lambda_{k(j)}}} \\
&\quad = \exp\left(-e^{-u_{nj}}\sum_{k=1}^{K}\left(\sum_{j' \in B_k} e^{\frac{\mu_{nj'}}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \left(e^{-\frac{u_{nj}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1}\left(\sum_{j' \in B_{k(j)}} e^{\frac{\mu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1} e^{-\frac{u_{nj}}{\lambda_{k(j)}}} e^{\frac{\mu_{nj}}{\lambda_{k(j)}}} \\
&\quad = e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\left(e^{-\frac{u_{nj}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}}\left(\sum_{j' \in B_{k(j)}} e^{\frac{\mu_{nj'}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1} \\
&\qquad \exp\left(-e^{-u_{nj}}\sum_{k=1}^{K}\left(\sum_{j' \in B_k} e^{\frac{\mu_{nj'}}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\quad = e^{\frac{\mu_{nj}}{\lambda_{k(j)}}} C_1^{\lambda_{k(j)}-1} e^{-u_{nj}} \exp\left(-e^{-u_{nj}} C_2\right).
\end{aligned}
$$

where

$$C_1 = \sum_{j' \in B_{k(j)}} e^{\frac{\mu_{nj'}}{\lambda_{k(j)}}}$$

and

$$C_2 = \sum_{k=1}^{K}\left(\sum_{j' \in B_k} e^{\frac{\mu_{nj'}}{\lambda_k}}\right)^{\lambda_k}.$$

Now we can integrate,

$$\begin{aligned}
q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, \frac{\partial \Pi}{\partial x_j}(u_{nj}, \ldots, u_{nj}) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1}\, e^{-u_{nj}}\, \exp\left(-e^{-u_{nj}}\, C_2\right) \\
&= e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1} \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{-u_{nj}}\, \exp\left(-e^{-u_{nj}}\, C_2\right)
\end{aligned}$$

To proceed further we make the substitution

$$z = \phi(u_{nj}) = -e^{-u_{nj}}\, C_2$$

with

$$\mathrm{d}z = \mathrm{d}u_{nj}\, e^{-u_{nj}}\, C_2$$

and

$$\begin{aligned}
\phi(-\infty) &= -\infty \\
\phi(+\infty) &= 0.
\end{aligned}$$

This gives

$$\begin{aligned}
q_{nj} &= e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1} \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{-u_{nj}}\, \exp\left(-e^{-u_{nj}}\, C_2\right) \\
&= e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1} \int_{-\infty}^{0} \mathrm{d}z\, \frac{1}{C_2}\, \exp(-z) \\
&= e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1}\, \frac{1}{C_2} \Big[\exp(0) - \exp(-\infty) \Big] \\
&= e^{\frac{\mu_{nj}}{\lambda_{k(j)}}}\, C_1^{\lambda_{k(j)}-1}\, \frac{1}{C_2} \\
&= \frac{e^{\frac{\mu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j' \in B_{k(j)}} e^{\frac{\mu_{nj'}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k=1}^{K} \left(\sum_{j' \in B_k} e^{\frac{\mu_{nj'}}{\lambda_k}} \right)^{\lambda_k}}.
\end{aligned}$$

A.3.2 Sensitivities

Calculating sensitivities, and then elasticities, requires even more careful organization. First we have

$$\begin{aligned}\frac{\partial q_{nj}}{\partial \nu_{nj'}} &= \frac{\partial}{\partial \nu_{nj'}} \left[\frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \right] \\ &= C_1 + C_2 + C_3\end{aligned}$$

These terms are

$$\begin{aligned}C_1 &= \frac{\left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \frac{\partial}{\partial \nu_{nj'}} \left[e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \right] \\ &= \frac{\left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \delta_{j'j} \frac{1}{\lambda_{k(j)}} e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \\ &= \delta_{j'j} \frac{1}{\lambda_{k(j)}} \frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \\ &= \delta_{j'j} \frac{1}{\lambda_{k(j)}} q_{nj}\end{aligned}$$

and

$$
\begin{aligned}
C_2 = & \frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}}}{\sum_{k'=1}^{K}\left(\sum_{j''\in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
& \cdot \frac{\partial}{\partial \nu_{nj'}}\left[\left(\sum_{j''\in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1}\right] \\
= & \frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}}}{\sum_{k'=1}^{K}\left(\sum_{j''\in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
& \cdot \delta_{k(j')k(j)}\left(\lambda_{k(j)}-1\right)\left(\sum_{j''\in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-2} \frac{1}{\lambda_{k(j)}} e^{\frac{\nu_{nj'}}{\lambda_{k(j)}}} \\
= & \delta_{k(j')k(j)}\frac{\lambda_{k(j)}-1}{\lambda_{k(j)}}\frac{e^{\frac{\nu_{nj'}}{\lambda_{k(j)}}}}{\sum_{j''\in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}}}\frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}}\left(\sum_{j''\in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}}\right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K}\left(\sum_{j''\in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}}\right)^{\lambda_{k'}}} \\
= & \delta_{k(j')k(j)}\frac{\lambda_{k(j)}-1}{\lambda_{k(j)}} q_{nj'|k(j)}\, q_{nj}
\end{aligned}
$$

and then

$$
\begin{aligned}
C_3 = \quad & e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1} \\
& \cdot \frac{\partial}{\partial \nu_{nj'}} \left[\frac{1}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \right] \\
= \quad & e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1} \\
& \cdot (-1) \frac{1}{\left(\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}} \right)^2} \lambda_{k(j')} \left(\sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \right)^{\lambda_{k(j')}} \frac{1}{\lambda_{k(j')}} e^{\frac{\nu_{nj'}}{\lambda_{k(j')}}} \\
= - \quad & \frac{e^{\frac{\nu_{nj}}{\lambda_{k(j)}}} \left(\sum_{j'' \in B_{k(j)}} e^{\frac{\nu_{nj''}}{\lambda_{k(j)}}} \right)^{\lambda_{k(j)}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \\
& \cdot \frac{e^{\frac{\nu_{nj'}}{\lambda_{k(j')}}} \left(\sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \right)^{\lambda_{k(j')}-1}}{\sum_{k'=1}^{K} \left(\sum_{j'' \in B_{k'}} e^{\frac{\nu_{nj''}}{\lambda_{k'}}} \right)^{\lambda_{k'}}} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \\
= & -q_{nj}\, q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}}.
\end{aligned}
$$

Substituting these intermediate results back in then gives

$$\begin{aligned}
\frac{\partial q_{nj}}{\partial \nu_{nj'}} = &\quad C_1 + C_2 + C_3 \\
= &\quad \delta_{j'j} \frac{1}{\lambda_{k(j)}} q_{nj} \\
&+ \delta_{k(j')k(j)} \frac{\lambda_{k(j)} - 1}{\lambda_{k(j)}} q_{nj'|k(j)} \, q_{nj} \\
&- q_{nj} \, q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \\
= &\left(\quad \delta_{j'j} \frac{1}{\lambda_{k(j)}} \right. \\
&\quad + \delta_{k(j')k(j)} \frac{\lambda_{k(j)} - 1}{\lambda_{k(j)}} q_{nj'|k(j)} \\
&\quad \left. - q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \right) q_{nj}.
\end{aligned}$$

The elasticities immediately follow,

$$\begin{aligned}
\epsilon_{njj'} &= \frac{\mu_{nj'}}{q_{nj}} \frac{\partial q_{nj_1}}{\partial \nu_{nj'}} \\
&= \frac{\mu_{nj'}}{q_{nj}} \left(\quad \delta_{j'j} \frac{1}{\lambda_{k(j)}} \right. \\
&\quad + \delta_{k(j')k(j)} \frac{\lambda_{k(j)} - 1}{\lambda_{k(j)}} q_{nj'|k(j)} \\
&\quad \left. - q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \right) q_{nj} \\
&= \left(\quad \delta_{j'j} \frac{1}{\lambda_{k(j)}} \right. \\
&\quad + \delta_{k(j')k(j)} \frac{\lambda_{k(j)} - 1}{\lambda_{k(j)}} q_{nj'|k(j)} \\
&\quad \left. - q_{nj'} \sum_{j'' \in B_{k(j')}} e^{\frac{\nu_{nj''}}{\lambda_{k(j')}}} \right) \mu_{nj'}.
\end{aligned}$$

A.4 Generalized Nested Gumbel Model Calculations

Recall that the generalized nested Gumbel model is defined by the joint cumulative distribution function

$$\Pi(u_{n1}, \ldots, u_{nJ}) = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j \in B_k}\left(\alpha_{jk}\, e^{-u_{nj}+\mu_{nj}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right).$$

Differentiating this joint cumulative distribution function is a little bit more straightforward if we first define the subset of nests that contain each alternative, A_j so that $k \in A_j$ if and only if $j \in B_k$.

Now we can turn return to the chain rule mines,

$$\begin{aligned}
&\frac{\partial \Pi}{\partial u_{nj}}(u_{n1}, \ldots, u_{nJ}) \\
&\quad = \frac{\partial}{\partial u_{nj}} \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj'}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj'}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j} \lambda_k \left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj'}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \frac{1}{\lambda_k}\left(\alpha_{jk}\, e^{-u_{nj}+\mu_{nj}}\right)^{\frac{1}{\lambda_k}-1} \alpha_{jk}\, e^{-u_{nj'}+\mu_{nj'}} \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj'}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j} \left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj'}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(\alpha_{jk}\, e^{-u_{nj}+\mu_{nj}}\right)^{\frac{1}{\lambda_k}-1} \alpha_{jk}\, e^{-u_{nj}+\mu_{nj}}.
\end{aligned}$$

Next we evaluate each argument on the active utility and simplify as much as possible,

$$
\begin{aligned}
&\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \dots, u_{nj}) \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj}+\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(\alpha_{jk}\, e^{-u_{nj}+\mu_{nj}}\right)^{\frac{1}{\lambda_k}-1}\ \alpha_{jk}\, e^{-u_{nj}+\mu_{nj}} \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj} e^{\mu_{nj'}}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{-u_{nj} e^{\mu_{nj'}}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(\alpha_{jk}\, e^{-u_{nj} e^{\mu_{nj}}}\right)^{\frac{1}{\lambda_k}-1}\ \alpha_{jk}\, e^{-u_{nj} e^{\mu_{nj}}} \\
&\quad = \exp\left(-\sum_{k=1}^{K}\left(\left(e^{-u_{nj}}\right)^{\frac{1}{\lambda_k}}\ \sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j}\left(\left(e^{-u_{nj}}\right)^{\frac{1}{\lambda_k}}\ \sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(e^{-u_{nj}}\right)^{\frac{1}{\lambda_k}-1}\ e^{-u_{nj}}\ \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}-1}\ \alpha_{jk}\, e^{\mu_{nj}} \\
&\quad = \exp\left(-e^{-u_{nj}}\ \sum_{k=1}^{K}\left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j}\left(e^{-u_{nj}}\right)^{\frac{\lambda_k - 1}{\lambda_k}}\ \left(\sum_{j' \in B_k}\left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(e^{-u_{nj}}\right)^{\frac{1}{\lambda_k}}\ \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}}.
\end{aligned}
$$

Pushing forward with the simplifications,

$$
\begin{aligned}
&\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \dots, u_{nj}) \\
&\quad = \exp\left(-e^{-u_{nj}} \sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j} \left(e^{-u_{nj}}\right)^{\frac{\lambda_k - 1}{\lambda_k}} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(e^{-u_{nj}}\right)^{\frac{1}{\lambda_k}} \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}} \\
&\quad = \exp\left(-e^{-u_{nj}} \sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j} \left(e^{-u_{nj}}\right)^{\frac{\lambda_k - 1 + 1}{\lambda_k}} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \\
&\qquad\qquad \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}} \\
&\quad = \exp\left(-e^{-u_{nj}} \sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad \sum_{k \in A_j} e^{-u_{nj}} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}} \\
&\quad = \exp\left(-e^{-u_{nj}} \sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}\right) \\
&\qquad e^{-u_{nj}} \sum_{k \in A_j} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}} \\
&\quad \equiv \exp\left(-e^{-u_{nj}}\, C_1\right)\, e^{-u_{nj}}\, C_2
\end{aligned}
$$

At this point the choice probabilities are given by integration,

$$
\begin{aligned}
q_{nj} &= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, \frac{\partial \Pi}{\partial x_j}(u_{nj}, \dots, u_{nj}) \\
&= \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, \exp\left(-e^{-u_{nj}}\, C_1\right)\, e^{-u_{nj}}\, C_2 \\
&= C_2 \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{-u_{nj}}\, \exp\left(-e^{-u_{nj}}\, C_1\right)
\end{aligned}
$$

To evaluate this integrate we employ the substitution

$$z = \phi(u_{nj}) = -e^{-u_{nj}} C_1$$

with

$$\mathrm{d}z = \mathrm{d}u_{nj} e^{-u_{nj}} C_1$$

and

$$\begin{aligned}\phi(-\infty) &= -\infty \\ \phi(+\infty) &= 0.\end{aligned}$$

This gives

$$\begin{aligned}q_{nj} &= C_2 \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, e^{-u_{nj}}\, \exp\left(-e^{-u_{nj}} C_1\right) \\ &= \frac{C_2}{C_1} \int_{-\infty}^{0} \mathrm{d}z\, \exp\left(z\right) \\ &= \frac{C_2}{C_1} \left[\exp\left(0\right) - \exp(-\infty)\right] \\ &= \frac{C_2}{C_1} \\ &= \frac{\sum_{k \in A_j} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k - 1} \left(\alpha_{jk}\, e^{\mu_{nj}}\right)^{\frac{1}{\lambda_k}}}{\sum_{k=1}^{K} \left(\sum_{j' \in B_k} \left(\alpha_{j'k}\, e^{\mu_{nj'}}\right)^{\frac{1}{\lambda_k}}\right)^{\lambda_k}}.\end{aligned}$$

Acknowledgements

A very special thanks to everyone supporting me on Patreon: Alessandro Varacca, Alex D, Alexander Noll, Amit, Andrea Serafino, Andrew Mascioli, Andrew Rouillard, Ara Winter, Ari Holtzman, Austin Rochford, Aviv Keshet, Avraham Adler, Ben Matthews, Ben Swallow, Benoit Essiambre, boot, Brendan Galdo, Bryan Chang, Cameron Smith, Canaan Breiss, Cat Shark, Cathy Oliveri, Charles Naylor, Chase Dwelle, Chris Jones, Christina Van Heer, Christopher Mehrvarzi, Colin Carroll, Colin McAuliffe, Damien Mannion, dan mackinlay, Dan W Joyce, Dan Waxman, Dan Weitzenfeld, Daniel Edward Marthaler, Daniel Hammarström, Danny Van Nest, David Burdelski, Doug Rivers, Dr. Jobo, Dr. Omri Har Shemesh, Dylan Maher, Dylan Spielman, Ebriand, Ed Cashin, Edgar Merkle, Edoardo Marcora, Eric LaMotte, Erik Banek, Eugene O'Friel, Felipe Vaca, Fergus Chadwick, Francesco Corona, Geoff Rollins, Glenn Williams, Granville Matheson, Gregor Gorjanc, Guilherme Marthe, Håkan Johansson, Hamed Bastan-Hagh, haubur, Hector Munoz, hs, Hugo Botha, 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, Josh Knecht, JU, Julian Lee, Justin Bois, Kádár András, Karim Naguib, Karim Osman, Konstantin Shakhbazov, Kristian Gårdhus Wichmann, Lars Barquist, lizzie , LOU ODETTE, Mads Christian Hansen, Marek Kwiatkowski, Mariana Carmona, Mark Donoghoe, Markus P., Márton Vaitkus, Matěj, Matthew, Matthieu LEROY, Mattia Arsendi, Maurits van der Meer, Max, Michael Colaresi, Michael DeJesus, Michael DeWitt, Michael Dillon, Michael Lerner, Mick Cooney, MisterMentat , N Sanders, N.S. , Name, Nathaniel Burbank, Nicholas Cowie, Nick S, Nikita Karetnikov, Octavio Medina, Ole Rogeberg, Olivier Ma, Patrick Kelley, Patrick Boehnke, Pau Pereira Batlle, Pieter van den Berg , ptr, quasar, Ramiro Barrantes Reynolds, Raúl Peralta Lozada, Rex, Riccardo Fusaroli, Richard Nerland, Rob Davies, Robert Frost, Robert Goldman, Robert kohn, Robin Taylor, Ryan Gan, Ryan Grossman, Ryan Kelly, S Hong, Sean Wilson, Sergiy Protsiv, Seth Axen, shira, Simon Duane, Simon Lilburn, Simon Steiger, Simone, Spencer Boucher, sssz, Stefan Lorenz, Stephen Lienhard, Steve Harris, Steven Forrest, Stew Watts, Stone Chen, Susan Holmes, Svilup, Tate Tunstall, Tatsuo Okubo, Teresa Ortiz, Theodore Dasher, Thomas Siegert, Thomas Vladeck, Tobychev , Tomas Capretto, Tony Wuersch, Virgile Andreani, Virginia Fisher, Vitalie Spinu, Vladimir Markov, VO2 Maximus Decius, Will Farr, Will Lowe, Will Wen, Xianda Sun, yolhaj , yureq , Zach A, and Zhengchen Cai.

References

Berry, Steven, James Levinsohn, and Ariel Pakes. 1995. "Automobile Prices in Market Equilibrium." *Econometrica* 63 (4): 841–90.

Chapman, Chris, and Elea McDonnell Feit. 2015. *R for Marketing Research and Analytics*. Edited by 1. Use r! Springer Cham.

de Carvalho, M., R. Huser, P. Naveau, and B. J. Reich. 2026. *Handbook of Statistics of Extremes*. Boca Raton, FL: Chapman & Hall/CRC.

Horn, Roger A., and Charles R. Johnson. 1985. *Matrix Analysis*. Cambridge University Press.

Schwarz, Jason S., Chris Chapman, and Elea McDonnell Feit. 2020. *Python for Marketing Research and Analytics*. Edited by 1. Springer Cham.

Train, Kenneth E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge University Press.

Walker, Joan L., Moshe Ben-Akiva, and Denis Bolduc. 2007. "Identification of Parameters in Normal Error Component Logit-Mixture (NECLM) Models." *Journal of Applied Econometrics* 22 (6): 1095–125.

License

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

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

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

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

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

Original Computing Environment

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

```
CC=clang

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

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

```
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS 15.7.4

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

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

time zone: America/New_York
```

```
tzcode source: internal

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

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

loaded via a namespace (and not attached):
 [1] gtable_0.3.4       jsonlite_1.8.8     compiler_4.3.2     Rcpp_1.0.11
 [5] parallel_4.3.2     gridExtra_2.3      scales_1.3.0       yaml_2.3.8
 [9] fastmap_1.1.1      ggplot2_3.4.4      R6_2.6.1           curl_5.2.0
[13] knitr_1.45         tibble_3.2.1       munsell_0.5.0      pillar_1.9.0
[17] rlang_1.1.2        utf8_1.2.4         V8_4.4.1           inline_0.3.19
[21] xfun_0.41          RcppParallel_5.1.7 cli_3.6.2          magrittr_2.0.3
[25] digest_0.6.33      grid_4.3.2         lifecycle_1.0.4    vctrs_0.6.5
[29] evaluate_0.23      glue_1.6.2         QuickJSR_1.0.8     codetools_0.2-19
[33] stats4_4.3.2       pkgbuild_1.4.3     fansi_1.0.6        colorspace_2.1-0
[37] rmarkdown_2.25     matrixStats_1.2.0  tools_4.3.2        loo_2.6.0
[41] pkgconfig_2.0.3    htmltools_0.5.7
```

Stan
Program 2 `ig1.stan`

```
data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Individual observations
  array[M] int<lower=1, upper=N> agent;
  array[M] int<lower=1, upper=J> choice;
}

parameters {
  vector[J] mu;                 // Baseline utilities
  array[N] real<lower=0> sigma; // Scales
}

transformed parameters {
  // Alternative log probabilities
  array[N] vector[J] log_probs;
  for (n in 1:N)
    log_probs[n] = log_softmax(mu / sigma[n]);
}

model {
  // Prior model
  target += normal_lpdf(mu | 0, 100);
  target += normal_lpdf(sigma | 0, 100);

  // Observational model
  for (m in 1:M) {
    int n = agent[m];
    target += log_probs[n][choice[m]];
  }
}

generated quantities {
  array[M] int<lower=1, upper=J> choice_pred;

  for (m in 1:M) {
    int n = agent[m];
    choice_pred[m] = categorical_logit_rng(log_probs[n]);
  }
}
```

Stan
Program 3 `ig2.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }
}

data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Individual observations
  array[M] int<lower=1, upper=N> agent;
  array[M] int<lower=1, upper=J> choice;

  // Anchoring configuration
  int<lower=1, upper=J> anchor;
}

parameters {
  // Relative baseline utilities for non-anchor alternatives
  vector[J - 1] omega_free;

  // Relative scales for non-anchor agents
  array[N - 1] real<lower=0> tau_free;
}

transformed parameters {
  // Relative baseline utilities for all alternatives
  vector[J] omega = add_anchor(omega_free, anchor);

  // Relative scales for all agents
  array[N] real<lower=0> tau = append_array({1}, tau_free);

  // Alternative log probabilities
  array[N] vector[J] log_probs;
  for (n in 1:N)
    log_probs[n] = log_softmax(omega / tau[n]);
}

model {
  // Prior model
```

Stan
Program 4 `ig3.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }
}

data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Aggregated observations
  array[N, J] int<lower=0> choice_counts;

  // Anchoring configuration
  int<lower=1, upper=J> anchor;
}

parameters {
  // Relative baseline utilities for non-anchor alternatives
  vector[J - 1] omega_free;

  // Relative scales for non-anchor agents
  array[N - 1] real<lower=0> tau_free;
}

transformed parameters {
  // Relative baseline utilities for all alternatives
  vector[J] omega = add_anchor(omega_free, anchor);

  // Relative scales for all agents
  array[N] real<lower=0> tau = append_array({1}, tau_free);

  // Alternative log probabilities
  array[N] vector[J] log_probs;
  for (n in 1:N)
    log_probs[n] = log_softmax(omega / tau[n]);
}

model {
  // Prior model
  target += normal_lpdf(omega_free | 0, 10);
  target += normal_lpdf(tau_free | 0, 10);
```

Stan
Program 5 `simu\_ig2.stan`

```
data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Choice subset configurations
  int<lower=1> S;

  int<lower=1> SJ;
  array[S] int<lower=1, upper=SJ> subset_start;
  array[S] int<lower=1, upper=SJ> subset_end;
  array[SJ] int<lower=1, upper=J> alts;
}

generated quantities {
  vector[J] mu;                 // Baseline utilities
  array[N] real<lower=0> sigma; // Scales

  array[M] int<lower=1, upper=S> subset; // Observed choice subset
  array[M] int<lower=1, upper=N> agent;  // Observed agent
  array[M] int<lower=1, upper=J> choice; // Observed choice

  // Sample true data generating process behavior
  for (j in 1:J) mu[j] = normal_rng(0, 1);
  for (n in 1:N) sigma[n] = gamma_rng(31.0, 15.5);

  // Simulate observations
  for (m in 1:M) {
    int n = categorical_rng(uniform_simplex(N));
    int s = categorical_rng(uniform_simplex(S));

    int JJ = subset_end[s] - subset_start[s] + 1;
    array[JJ] int subset_alts
      = alts[subset_start[s]:subset_end[s]];

    vector[JJ] subset_log_probs
      = log_softmax(mu[subset_alts] / sigma[n]);

    agent[m] = n;
    subset[m] = s;
    choice[m] = categorical_logit_rng(subset_log_probs);
  }
}
```

Stan
Program 6 `ig\_joint.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }
}

data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Choice subset configurations
  int<lower=1> S;

  int<lower=1> SJ;
  array[S] int<lower=1, upper=SJ> subset_start;
  array[S] int<lower=1, upper=SJ> subset_end;
  array[SJ] int<lower=1, upper=J> alts;

  // Aggregated observations
  array[N, SJ] int<lower=0> choice_counts;

  // Anchoring configuration
  int<lower=1, upper=J> anchor;

  // Prediction configuration
  int<lower=1, upper=N> n_pred;

  int<lower=1> J_pred;
  array[J_pred] int<lower=1, upper=J> pred_alts;
}

parameters {
  // Relative baseline utilities for non-anchor alternatives
  vector[J - 1] omega_free;

  // Relative scales for non-anchor agents
  array[N - 1] real<lower=0> tau_free;
}

transformed parameters {
  // Relative baseline utilities for all alternatives
  vector[J] omega = add_anchor(omega_free, anchor);
```

Stan
Program 7 `simu\_ng1.stan`

```
functions {
  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}

data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Nesting configuration
  int<lower=1, upper=J> K;
  array[J] int<lower=1, upper=K> nests;

  // Intra-nest coupling parameters
  vector<lower=0, upper=1>[K] lambda;
}

generated quantities {
  vector[J] mu;                 // Baseline utilities
  array[N] real<lower=0> sigma; // Scales

  array[M] int<lower=1, upper=N> agent;  // Observed agent
```

Stan
Program 8 `ng1.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }

  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}

data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Aggregated observations
  array[N, J] int<lower=0> choice_counts;
```

Stan
Program 9 `ng2.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }

  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}
```



```
data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Aggregated observations
  array[N, J] int<lower=0> choice_counts;
```

Stan
Program 10 `simu\_ng2.stan`

```
functions {
  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}

data {
  int<lower=1> M; // Number of observations
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Total number of alternatives

  // Nesting configuration
  int<lower=1, upper=J> K;
  array[J] int<lower=1, upper=K> nests;

  // Choice subset configurations
  int<lower=1> S;

  int<lower=1> SJ;
  array[S] int<lower=1, upper=SJ> subset_start;
  array[S] int<lower=1, upper=SJ> subset_end;
  array[SJ] int<lower=1, upper=J> alts;

  // Intra-nest coupling parameters
```

Stan
Program 11 `ng\_joint.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }

  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}
```



```
data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Nesting configuration
  int<lower=1, upper=J> K;
```

Stan
Program 12 `mig\_joint1.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }

  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }
}
```



```
data {
  int<lower=1> N; // Number of agents
  int<lower=1> J; // Number of alternatives in choice set

  // Nesting configuration
  int<lower=1, upper=J> K;
```

Stan
Program 13 `mig\_joint2.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }

  // Return the number of elements in vals that equal i
  int num_which_int(array[] int vals, int i) {
    int N = num_elements(vals);
    int NN = 0;

    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
      }
    }

    return NN;
  }

  // Return the elements in vals that equal i
  array[] int which_int(array[] int vals, int i) {
    int N = num_elements(vals);

    array[N] int selected_idxs;
    int NN = 0;
    for (n in 1:N) {
      if (vals[n] == i) {
        NN += 1;
        selected_idxs[NN] = n;
      }
    }

    return selected_idxs[1:NN];
  }

  // Remove the ith row from the matrix X
  matrix thin(matrix X, int i) {
    int N = rows(X);
    array[N - 1] int idxs;
    if (i == 1)
      idxs = linspaced_int_array(N - 1, 2, N);
    else if (i == N)
```

Stan
Program 14 `conjoint.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }
}

data {
  int<lower=1> M; // Number of observations
  int<lower=1> J; // Number of alternatives in each observation
  int<lower=1> N; // Number of agents
  int<lower=1> F; // Number of factors

  // Concatenated organization for factor levels
  int LL;
  array[F] int factor_start;
  array[F] int factor_end;

  // Observation configuration
  array[M] int<lower=1, upper=N> agent;  // Agent
  array[M] int<lower=1, upper=J> choice; // Choice
  array[M, J, F] int alt_levels;         // Alternative levels

  // Anchoring configuration
  array[F] int<lower=1> factor_anchors;
}

transformed data {
  // Concatenated organization for relative factor levels
  int rel_LL = LL - F;
  array[F] int rel_factor_start;
  array[F] int rel_factor_end;

  rel_factor_start[1] = factor_start[1];
  for (f in 1:(F - 1)) {
    rel_factor_end[f] = factor_end[f] - f;
    rel_factor_start[f + 1] = factor_start[f + 1] - f;
  }
  rel_factor_end[F] = factor_end[F] - F;
}

parameters {
  // Relative scales for non-anchor agents
```

Stan
Program 15 `regression.stan`

```
functions {
  // Insert a zero into x_free at index anchor
  vector add_anchor(vector x_free, int anchor) {
    int J = size(x_free) + 1;
    if (anchor == 1)
      return append_row(0, x_free);
    else if (anchor == J)
      return append_row(x_free, 0);
    else
      return append_row(x_free[1:(anchor - 1)],
                        append_row(0, x_free[anchor:(J - 1)]));
  }
}

data {
  int<lower=1> M; // Number of observations
  int<lower=1> I; // Number of (continuous) covariates
  int<lower=1> F; // Number of (categorical) factors

  // Concatenated organization for alternative configurations
  int<lower=M> MJ; // Total number of alternatives
  array[M] int alt_start;
  array[M] int alt_end;

  // Concatenated organization for factor levels
  int<lower=1> LL;
  array[F] int factor_start;
  array[F] int factor_end;

  // Observation configuration
  matrix [MJ, I] X;             // Covariate design matrix
  array[MJ, F] int alt_levels;  // Factor levels
  array[M] int<lower=1> choice; // Choice

  // Anchoring configuration
  vector[I] x0;
  array[F] int<lower=1> factor_anchors;
}

transformed data {
  // Concatenated organization for relative factor levels
  int rel_LL = 0;
  array[F] int rel_factor_start;
  array[F] int rel_factor_end;

  // Residual design matrix
  matrix [MJ, I] deltaX;

  for (f in 1:F) {
    rel_factor_start[f] = rel_LL + 1;
```