# Discrete Choice Modeling, Part I

Michael Betancourt

April 2026

More modeling technique chapters are available on my website.

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

## Table of contents

**1 Utility Maximization Models** **3**
- 1.1 Agents, Alternatives, and Choice Sets . . . 3
- 1.2 Utilities . . . 4
- 1.3 Choice Probabilities . . . 5
- 1.4 Deriving Choice Probabilities From Cumulative Distribution Functions . . . 6
- 1.5 Non-Utility Choice Models . . . 8

**2 Substitution Patterns** **8**
- 2.1 Independence from Irrelevant Alternatives . . . 9
  - 2.1.1 Removing An Alternative From The Choice Set . . . 9
  - 2.1.2 Adding An Alternative To The Choice Set . . . 11
  - 2.1.3 The Blue Bus/Red Bus Problem . . . 12
- 2.2 Sensitivities and Elasticities . . . 14
- 2.3 Special Cases and Extensions . . . 16
- 2.4 Perfect Replacements . . . 17
- 2.5 Learning Substitution Patterns . . . 18

**3 Independent Utility Models** **20**
- 3.1 The Independent Gumbel Model . . . 21
  - 3.1.1 Choice Probabilities . . . 22
  - 3.1.2 Conditional Choice Probabilities . . . 23
  - 3.1.3 Implementing Choice Probabilities . . . 24
- 3.2 Redundancies of the Independent Gumbel Model . . . 25
  - 3.2.1 Translation Redundancy . . . 25

3.2.2 Scale Redundancy . . . . . 26
3.2.3 Resolving Both Redundancies Without Abusing Notation . . . . . 28
3.2.4 Prior Modeling . . . . . 29
3.3 Substitution Patterns of the Independent Gumbel Model . . . . . 31
3.4 Demonstration . . . . . 32
3.4.1 Setup . . . . . 33
3.4.2 Basics . . . . . 33
3.4.3 Independent Gumbel Substitution Patterns . . . . . 52

**4 Conclusion** **74**

**Appendix: Onerous Calculations** **75**
A.1 Independent Gumbel Model Calculations . . . . . 75
A.2 Independent Generalized Extreme Value Model Calculations . . . . . 78

**Acknowledgements** **80**

**References** **81**

**License** **81**

**Original Computing Environment** **82**

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 given 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 (Tversky 1972), marketing (McFadden 1986), and economics (Train 2009). Unfortunately, each of these applications introduces their own emphasis, terminology, and notation. In these chapters, 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.

In this first part, we will consider the general analysis of choice as decision making under uncertainty, how decision making behavior can change with the possible alternatives, and the foundational discrete choice model. This foundation will then allow us to consider more sophisticated discrete choice models, with a particular focus on the underlying assumptions and their consequences, in Part 2, Part 3, and Part 4. Throughout, I will demonstrate the accumulating theoretical insights with explicit examples.

The discrete choice literature is expansive. Instead of attempting any comprehensive bibliography, I will focus on only key references. For many more references, see for example the reviews in Train (2009), Chapman and Feit (2015), and Schwarz, Chapman, and Feit (2020).

# 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 they arrive at any particular decision.

## 1.1 Agents, Alternatives, and Choice Sets

In these chapters, 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. For example, we might examine consumers deciding on which products to purchase, herd animals deciding on which lands to graze, non-profit organizations deciding on which programs to fund, and so on. Generally, we will be interested in not a single agent, but rather an ensemble of $N$ agents.

Similarly, I will refer to the possible choices as **alternatives**. If the possible choices can be characterized by various properties, then each configuration of those properties defines a *distinct* alternative. For instance, a consumer good offered at three different prices would define three different alternatives. Be careful, however, as this is a not a universal terminological convention.

A **choice set** is a collection of $J$ alternatives from which each agent selects their choice. When modeling choices, we will assume that the choice set is both *exhaustive* and *exclusive*, so that choices are always elements of the choice set. In some experimental settings, this can be guaranteed; in more observational settings, however, it often has to be more carefully motivated. For an explicit example of these subtleties in practice, see Natan (2025).

When analyzing consumer choices, for example, we might observe what product a consumer purchased in a particular store, as well as that products available at that store. Typically, however, we cannot observe with which products the consumer was already acquainted as they entered the store, nor what products they noticed and then remembered as they shopped. In other words, the consumer in this case is not choosing from the set of all available products as we might assume, but rather a smaller subset of the available products. Some products might go unchosen because they are actually unappealing, but others might go unchosen only because the consumer was unaware of their availability.

One particularly productive way to account for agent awareness is to consider both a choice set, containing all available alternatives, and a smaller **awareness set**, containing just those alternatives known to an agent when a choice is made. The introduction of an awareness set then allows us to model choice in two stages, the selection of an awareness set from the full choice set followed by the selection of an alternative from the active awareness set.

Another potential issue with the assumption of a choice set is the possibility that agents might consider alternatives outside of the assumed choice set. Even if all of the observed choices fall

into the assumed choice set, future choices might not. Ignoring these possibilities then overly restricts inferences and predictions. For instance, behaviors observed when forcing agents to choose between subpar alternatives in an experiment are unlikely to generalize to more realistic circumstances.

One way to avoid these biases is to include an **outside option** into the assumed choice set. An outside option is an explicit alternative that represents a preference for anything but the other alternatives.

Finally, a choice set that is adequate for one decision is not necessarily adequate for *all* decisions. In many applications, the possible alternatives can vary from not only one decision to the next, both within and across agents. For example, the price and supply of consumer goods can change across both stores and time; a customer making repeated purchases at the same store and customers making contemporaneous purchases at different stores will generally encounter varying possibilities.

When the available alternatives are not fixed, I will use choice set to refer to the *complete* set of *all* possible alternatives across *all* decisions, and **choice subset** to refer to any particular subset of available options for any single decision. This convention allows the full choice set to contain not only all observable alternatives, but also any future or even relevant hypothetical alternatives.

## 1.2 Utilities

If decisions between the alternatives in a choice set are made *purposefully* and *consistently*, in other words if they are *rational*, then we can model those decisions by quantifying each agent's preference for each available alternative. 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, 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.

One way to calibrate stated preferences and revealed preferences to each other is to jointly model observed choices across different contexts. See, for example, the analyses in Ben-Akiva et al. (1994) and Feit, Beltramo, and Feinberg (2010).

## 1.3 Choice Probabilities

When the utilities that an agent assigns to each alternative are known exactly, the resulting decisions become completely deterministic. Agent $n$ 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 might be uncertain in a given analysis, we can always consistently model that uncertainty with a probability theory. Probabilistic utility models, defined by a joint probability distribution over all of the alternative utilities, are often referred to as **random utility models** (Manski 1977).

Because the utilities are all real-valued, we can always specify a probabilistic utility model with a joint Lebesgue 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 probabilistic utilities, decisions are no longer deterministic. Agent $n$ will choose alternative $j$ for any configuration of the utilities that fall into the rectangle subset

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

The probability that agent $n$ chooses alternative $j$ is then given by the probability allocated to these utility configurations,

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

These derived 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 is no longer valid.

To avoid any potential confusion, 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

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

By construction, discrete choice models are defined relative to the assumed choice subset. If we want to be precise we should really write $p(y_m \mid \mathsf{c})$, where $\mathsf{c}$ denotes an assumed choice subset, instead of $p(y_m)$. When the choice subset from which an agent is choosing is uncertain, for example when there is divergence between the awareness set and the choice subset, then can jointly model the discrete choice along with the choice subset,

$$p(y_m, \mathsf{c}) = p(y_m \mid \mathsf{c})\ p(\mathsf{c}).$$

## 1.4 Deriving Choice Probabilities From Cumulative Distribution Functions

Because $\mathsf{u}_{nj}$ is a product subset, we can write the corresponding indicator function as a product of component indicator functions,

$$I_{\mathsf{u}_{nj}}(u_{n1}, \ldots, u_{nJ}) = \prod_{j' \neq j} I_{\{u_{nj} > u_{nj'}\}}(u_{nj}).$$

This allows us to substantially simplify the derivation of 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 a 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 $j$th 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 derived choice probabilities!

Consequently, we can always 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 a probabilistic utility model with a joint cumulative distribution function, bypassing a joint probability density function representation entirely.

### 1.5 Non-Utility Choice Models

Given an agent and a choice set, we can always model alternative probabilities directly, instead of deriving them from a probabilistic utility model. That said, not every simplex of alternative probabilities can be derived from a probabilistic utility model.

Interesting, we can characterize which choice probability models are consistent with a latent probabilistic utility model, although the compatibility conditions are a bit awkward (Falmagne 1978). In theory, these conditions are also constructive, allowing one to construct a compatible probability utility model from a given choice probability model in a process known as **rationalization**. That said, these deduced latent models are not so much defined over real-valued utilities so much as *rankings* of the alternatives in the choice set.

## 2 Substitution Patterns

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

If, for example, the $j$th 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?

Taking $q_{nj}$ all the way to zero effectively removes the $j$th item from the choice set. This raises the question of 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**. In order to inform inferences and predictions that generalize beyond a particular choice subset, we need discrete choice models that manifest substitution patterns appropriate to a given application. Consequently, understanding the substitution patterns implied by a given probabilistic utility model is critical to the robust use of these techniques.

Unfortunately, quantifying substitution patterns through the behavior of choice probabilities 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 the other alternatives.

For instance, when the $j_3$th choice probability $q_{nj_3}$ decreases, 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 or decrease in $q_{nj_1}$ is complemented by the same proportional increase or 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** (Luce 1979) 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.

All of this said, the phrase "independence from irrelevant alternatives" is sometimes used to describe the behavior of not every alternative but rather only some alternatives. Usually, we have enough contextual information to avoid any ambiguity, but, as always, it's best to read carefully.

### 2.1.1 Removing An Alternative From The Choice Set

Independence from irrelevant alternatives 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 is such a strong condition that we can actually derive the choice probabilities for *any* choice subset from the fixed ratios.

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 all of the choice probabilities for this new choice subset.

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

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

At this point, 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 the assumed probabilistic utility model*. In practice, this can 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,

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

Lastly, 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 undesirable in some applications.

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

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

By construction, the ratio of these two choice probabilities is just

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

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,

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

Moreover, 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 is fundamentally incompatible with independence from irrelevant alternatives.

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, or if some commuters just really hate the color blue. 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, discrete choice models that exhibit independence from irrelevant alternatives perform poorly when some 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 the 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 fixed, external covariate.

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,

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

These families then yield 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 partial 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. Differentiating the summation constraint, for example, shows that the sensitivities across all alternatives in a choice set must 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**,

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

One advantage of elasticities is that they are independent of the choice of units for $z$. Another is that they identify invariances of both choice probabilities and ratios of choice probabilities at the same time.

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

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

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 Special Cases and Extensions

Sensitivities and elasticities are so general that they can come across as too abstract. In many fields, specific instances and combinations of these quantities are much more familiar.

Consider, for example, a model whose behavior depends on the price of the alternatives,

$$z = (p_1, \ldots, p_J).$$

Price elasticities quantify how sensitive agent preferences are to changing prices.

The elasticity of the $j_1$st alternative probability with respect to the $j_1$st price, for instance, is known as an **own-price elasticity**,

$$\epsilon_{nj_1}(p_1, \dots, p_J) = \frac{p_{j_1}}{q_{nj_1}} \frac{\partial q_{nj_1}}{\partial p_{j_1}}(p_1, \dots, p_J).$$

At the same time, the elasticity of the $j_1$st alternative probability with respect to the $j_2$st price is denoted a **cross-price elasticity**,

$$\epsilon_{nj_1j_2}(p_1, \dots, p_J) = \frac{p_{j_2}}{q_{nj_1}} \frac{\partial q_{nj_1}}{\partial p_{j_2}}(p_1, \dots, p_J).$$

In marketing, the **diversion ratio** (Conlon and Mortimer 2021) is the ratio of the sensitivities for any two pairs of alternatives,

$$D_{n,j_1 \to j_2}(z) = -\frac{\frac{\partial q_{nj_2}}{\partial z}(z)}{\frac{\partial q_{nj_1}}{\partial z}(z)}.$$

The larger $D$ is, the more increasing $z$ will siphon choice probability from $j_1$ to $j_2$. Diversion ratios are especially useful when $j_2$ denotes an outside option.

## 2.4 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,

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

or, equivalently,

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

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 instance, 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 $j$th alternative, $q_{nj}$.

Intuitively, we can interpret these two model configurations as defining two distinct $j$th alternatives that are equivalent from the perspective of the given agent.

## 2.5 Learning Substitution Patterns

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

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

or equivalently

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

different configurations of a parametric utility model will, in general, 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 infer the consistent behaviors of $\theta$ from observed data. We can then propagate inferred model configurations to corresponding choice probabilities and, ultimately, substitution patterns.

Bayesian inference uses the observations

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

to construct a posterior distribution for $\theta$,

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

This posterior distribution for $\theta$ 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.

At the point, we can use the inferred choice probabilities to study the range of substitution patterns consistent with the observed data. In particular, we can propagate inferential uncertainties into posterior distributions over sensitivities and elasticities, or even predictive distributions for new or hypothetical choice subsets.

All of this said, inferring substitution patterns is not always straightforward in practice. For instance, we learn nothing about substitution patterns from observations drawn from only a single choice set. The only way to discriminate between different substitution patterns is to consider observations that span multiple choice subsets. The more choice subsets we observe, the more precisely we will be able to infer the underlying substitution patterns. Exactly how many choice subsets we need to observe for inferences to be well-behaved, however, depends on the flexibility of the specific discrete choice model we are using.

Bayesian inference is most powerful when we always propagate inferential uncertainties through to any quantity of interest. Deriving quantities of interest from premature posterior expectation values loses information and muddles more nuanced behaviors.

Consider, for example, averaging over the consistent model configurations to construct posterior expectations of the choice probabilities,

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

Because these expected choice probabilities define a valid simplex,

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

they can be used to define substitution patterns.

These substitution patterns, however, will in general be different from the substitution patterns exhibited by any individual choice probability configuration. Even if every value of $\theta$ results in choice probabilities that exhibit independence from irrelevant alternatives, the posterior expected choice probabilities typically will not.

The deviations between these behaviors will not always be substantial, but in practice we won't know how deviate they are without computing them both. To avoid any risk of inferential misunderstanding, the safest and most robust use of Bayesian inference is to avoid premature posterior expectation values entirely.

# 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 identical,

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

With these restrictions, 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 univariate utility model is to consider *extremes*. Let's say, for instance, 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 persist. All of these asymptotic behaviors are quantified with a corresponding family of univariate probability density functions.

Unfortunately, most of the models motivated by extreme value theory are so awkward 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 $\mu$ roughly characterizing the location of the peak and the scale parameter $\sigma$ roughly characterizing the width of the peak (Figure 1).

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.

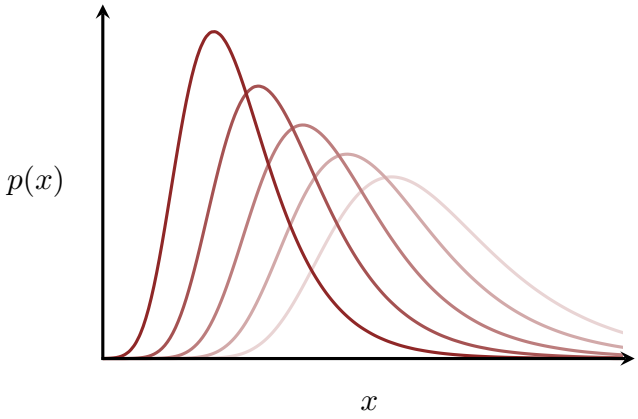



Figure 1: Each probability density function in the Gumbel family is unimodal and skewed towards larger values. The location of the peak and overall width vary across the family.

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 $\mu_{nj}$ as the **baseline utility** for each alternative, although in some fields these terms are referred to with other terms like systematic utilities and deterministic utilities.

### 3.1.1 Choice Probabilities

Under an independent Gumbel model, 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:

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

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

### 3.1.2 Conditional Choice Probabilities

One nice feature of these choice probabilities is that they are especially straightforward to decompose into conditional and marginal probabilities.

Consider a subset of $K < J$ alternatives

$$\mathsf{c} = \{j_1, \dots, j_K\}.$$

The marginal probability of choosing *any* of the alternatives in $\mathsf{c}$ is

$$\begin{aligned} p(\mathsf{c}) &= \sum_{j'' \in \mathsf{c}} q_{nj''} \\ &= \sum_{j'' \in \mathsf{c}} \frac{\exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \\ &= \frac{\sum_{j'' \in \mathsf{c}} \exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)}. \end{aligned}$$

Similarly, the conditional probability of choosing a particular alternative from $\mathsf{c}$ is given by

$$\begin{aligned} p(j_k \mid \mathsf{c}) &= \frac{p(j_k)}{p(\mathsf{c})} \\ &= \frac{q_{nj_k}}{\sum_{j'' \in \mathsf{c}} q_{nj''}} \\ &= \frac{\exp\left(\frac{\mu_{nj}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \frac{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)}{\sum_{j'' \in \mathsf{c}} \exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)} \\ &= \frac{\exp\left(\frac{\mu_{nj}}{\sigma_n}\right)}{\sum_{j'' \in \mathsf{c}} \exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)}. \end{aligned}$$

Conveniently, we can verify this decomposition with some quick algebra. For $j_k \in \mathsf{c}$, we have

$$\begin{aligned} p(j_k, \mathsf{c}) &= \frac{\exp\left(\frac{\mu_{nj}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \\ &= \frac{\exp\left(\frac{\mu_{nj}}{\sigma_n}\right)}{\sum_{j'' \in \mathsf{c}} \exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)} \frac{\sum_{j'' \in \mathsf{c}} \exp\left(\frac{\mu_{nj''}}{\sigma_n}\right)}{\sum_{j'=1}^{J} \exp\left(\frac{\mu_{nj'}}{\sigma_n}\right)} \\ &= p(j_k \mid \mathsf{c})\, p(\mathsf{c}). \end{aligned}$$

Decomposing choice probabilities is particularly useful for modeling selection within a choice set. While we can always model an outside option jointly with the other alternatives, for example, we can also model it in two stages: a marginal choice between the inside options and the outside option, followed by a conditional choice among the inside options.

### 3.1.3 Implementing Choice Probabilities

Conveniently, the choice probabilities in the independent Gumbel model 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 multinomial logit, with the multinomial logit of the choice probabilities being given by the scaled baseline utilities,

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

Moreover, the "multinomial logit" is often shortened to "multilogit" and even just "logit", not to be confused with the less general, one-dimensional logit function

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

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 terminology confusing, not only because of the sloppiness with the multilogit and logit function names, but also because it focuses on a derived behavior and not the initial assumptions that we actually make. I will stick with "independent Gumbel model", but beware when comparing to the 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. Specifically, any *translation* of the inputs leaves the output unchanged.

From the 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. Specifically, we need 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 some distinguished alternative $j'$. The distinguished alternative is often referred to as a **reference** or **anchor** alternative.

If $\delta_{nj} > 0$, then the baseline utility of the $j$th 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 freely translate these relative inputs, however, because $\delta_{nj'}$ is fixed to zero.

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.

In an independent Gumbel model, the inputs to the softmax function are all *ratios* of baseline utilities $\mu_{nj}$ and the agent scale $\sigma_n$,

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

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,

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

In other words, inferences of the choice probabilities cannot distinguish $\mu_{nj}$ from $\sigma_n$.

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

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

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

In this case, we have to distinguish a particular reference agent, and hence a particular reference agent scale, $\sigma_{n'}$. We can then use this reference scale to define the 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 reference agent is always fixed to one,

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

eliminating another degree of freedom from the model. If $\tau_n > 1$, then the choice probabilities for the $n$th agent,

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

will be more uniform than the choice probabilities for the reference agent. On the other hand, if $\tau_n < 1$ then the heterogeneity in the choice probabilities for the $n$th agent will be amplified relative to the reference agent.

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

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

Because the reference scale $\tau_{n'}$ is fixed to one, we cannot scale the relative scales $\tau_n$ to compensate for changes in the scaled baseline utilities $\zeta_{nj}$.

### 3.2.3 Resolving Both Redundancies Without Abusing Notation

To completely eliminate redundancies from the independent Gumbel discrete choice model we need to incorporate 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,

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

and relative scales,

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

With these model configuration variables, the choice probabilities are 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.

Many presentations of discrete choice models, especially those that limit inferences to point estimates, overload variable names and use $\mu_{nj}$ and $\sigma_n$ to refer to *both* the absolute and relative model configuration variables *at the same time*. Redundancies are managed by arbitrarily fixing one $\mu_{nj}$ to 0 and one $\sigma_n$ to 1, while 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 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 $\mu_{nj}$ and $\sigma_n$ when discussing theoretical properties and the relative variables $\omega_{nj}$ and $\tau_n$ when discussing implementations.

### 3.2.4 Prior Modeling

The change in interpretation as we move from absolute model configuration variables to relative model configuration variables is especially important in Bayesian analyses, as it directly impacts the prior model.

Because the absolute model configuration variables are defined up to only translations and scalings, eliciting a principled prior model is awkward at best. On the other hand, the relative model configuration variables correspond to explicit comparisons to the reference alternative and agent, which is much easier to compare to available domain expertise.

Consider, for example, the anchor agent $n'$, an arbitrary alternative $j$, and the anchor alternative $j'$. The ratio of choice probabilities between the two alternatives is given by

$$\begin{aligned}
\frac{q_{n'j}}{q_{n'j'}} &= \frac{e^{\frac{\omega_{n'j}}{\tau_{n'}}}}{\sum_{j''=1}^{J} e^{\frac{\omega_{n'j''}}{\tau_{n'}}}} \frac{\sum_{j''=1}^{J} e^{\frac{\omega_{n'j''}}{\tau_{n'}}}}{e^{\frac{\omega_{n'j'}}{\tau_{n'}}}} \\
&= \frac{e^{\frac{\omega_{n'j}}{\tau_{n'}}}}{e^{\frac{\omega_{n'j'}}{\tau_{n'}}}} \\
&= \exp\left(\frac{\omega_{n'j} - \omega_{n'j'}}{\tau_{n'}}\right) \\
&= \exp\left(\frac{\omega_{n'j}}{\tau_{n'}}\right) \\
&= \exp\left(\omega_{n'j}\right),
\end{aligned}$$

or

$$\log \frac{q_{n'j}}{q_{n'j'}} = \omega_{n'j}.$$

Consequently, any domain expertise about an alternative probability *relative* to the anchor alternative immediately constrains the reasonable values of $\omega_{n'j}$,

$$\begin{aligned} \frac{1}{\delta} &\lessapprox \frac{q_{n'j}}{q_{n'j'}} &&\lessapprox \delta \\ -\log\delta &\lessapprox \log\frac{q_{n'j}}{q_{n'j'}} &&\lessapprox +\log\delta \\ -\log\delta &\lessapprox \omega_{n'j} &&\lessapprox +\log\delta. \end{aligned}$$

We can enforce this constraint with a soft containment prior model that concentrates most, but not all, of the prior probability within these constraints,

$$p\left(\omega_{n'j}\right) = \text{normal}\left(\omega_{n'j} \,\middle|\, 0, \frac{\log\delta}{2.32}\right)$$

If we don't have any domain expertise that distinguishes between the agents, then our prior model for each agent's preferences should be symmetric. In other words, the prior model for the $\omega_{n'j}$ immediately translates to the $\omega_{nj}$,

$$p\left(\omega_{nj}\right) \approx \text{normal}\left(\omega_{nj} \,\middle|\, 0, \frac{\log\delta}{2.32}\right).$$

Somewhat weak domain expertise might, for instance, allow for a full order of magnitude proportional variation in the reference choice probabilities, $\delta = 10$. This would result in the prior model

$$\begin{aligned} p\left(\omega_{n'j}\right) &= \text{normal}\left(\omega_{n'j} \,\middle|\, 0, \frac{\log 10}{2.32}\right) \\ &\approx \text{normal}(\omega_{n'j} \mid 0, 1), \end{aligned}$$

which then translates to

$$p\left(\omega_{nj}\right) \approx \text{normal}(\omega_{nj} \mid 0, 1).$$

A symmetric prior model for the relative, scaled baseline utilities does not imply symmetric ratios of choice probabilities for all agents. That heterogeneity is moderated by the relative agent scales. One, somewhat crude, way to characterize the potential heterogeneity is to compare ratios of the logarithm of ratios of choice probabilities,

$$\begin{aligned} \frac{1}{\epsilon} &\lessapprox \frac{\left|\log\frac{q_{nj}}{q_{nj'}}\right|}{\left|\log\frac{q_{n'j}}{q_{n'j'}}\right|} &&\lessapprox \epsilon \\ \frac{1}{\epsilon} &\lessapprox \tau &&\lessapprox \epsilon \end{aligned}$$

This containment can then be implemented with, for example, a log normal prior model,

$$p\left(\tau_n\right) = \text{log-normal}\left(\tau_n \,\middle|\, 0, \frac{\epsilon}{2.32}\right).$$

When in doubt about an elicited threshold $\epsilon$, we can always implement a prior check and examine the induced behavior of the choice probabilities for any tension with the available domain expertise.

## 3.3 Substitution Patterns of the Independent Gumbel Model

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

Because the normalization that couples all of the choice probabilities together cancels exactly, 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 which two alternatives we consider, this ratio will only ever depend on the two corresponding baseline utilities; the ratios are always independent of changes to any of the other baseline utilities. Decreasing $\mu_{nj_3}$ 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 these rigid substitution patterns.

We can also demonstrate independence from irrelevant alternatives using elasticities. The

sensitivity of the $j_1$th choice probability with respect to the $j_3$th 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 always cancel so that 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 in a practical Bayesian analysis.

### 3.4.1 Setup

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

```
import matplotlib
import matplotlib.pyplot as plot
plot.rcParams['figure.figsize'] = [6, 4]
plot.rcParams['figure.dpi'] = 100
plot.rcParams['font.family'] = "Serif"
plot.rcParams['axes.spines.top'] = False
plot.rcParams['axes.spines.right'] = False

import math
import numpy

import stan

# Needed to run pystan through a jupyter kernel
import nest_asyncio
nest_asyncio.apply()
```

```
import sys
sys.path.append('support')

import mcmc_analysis_tools_pystan3 as util
import mcmc_visualization_tools as putil
```

### 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 will need some data to fit. For this demonstration, we'll simulate 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]));
  }
}
```

```
with open('stan_programs/simu_ig1.stan', 'r') as file:
  stan_program = file.read()
simu_model = stan.build(stan_program, random_seed=8438338 + 1,
                        data={'N': N, 'J': J, 'M': M})
simu = simu_model.fixed_param(num_chains=1, num_samples=1)

samples = util.extract_expectand_vals(simu)
```

```
Building...
```

```
agent = [ samples[f'agent[{m + 1}]'][0][0].astype(numpy.int64)
          for m in range(M) ]
```

```
choice = [ samples[f'choice[{m + 1}]'][0][0].astype(numpy.int64)
           for m in range(M) ]

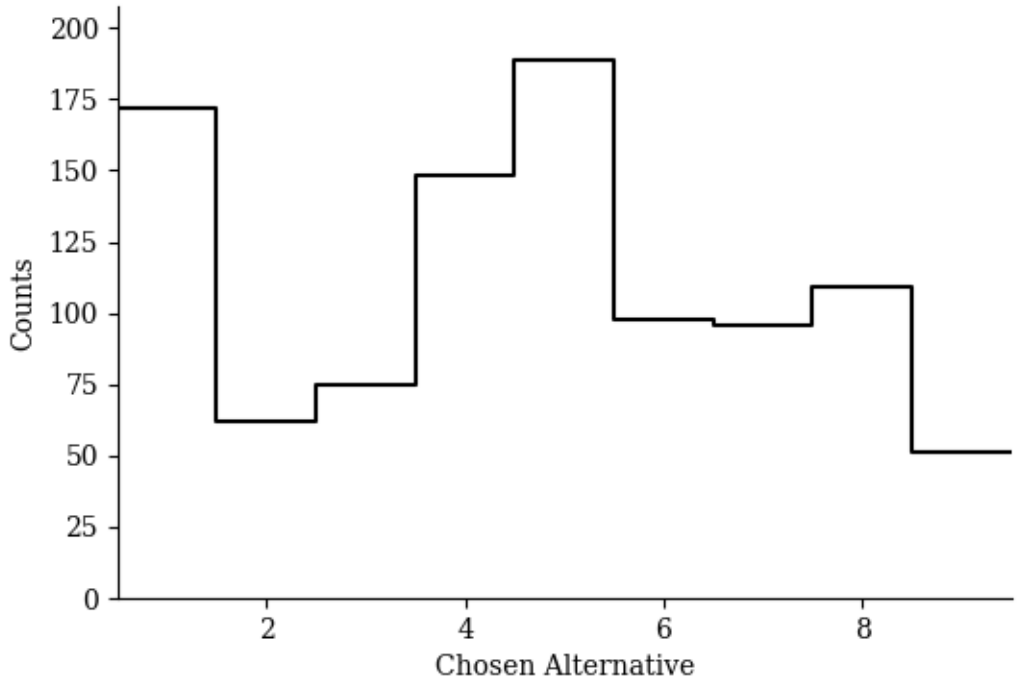
data = {'N': N, 'J': J, 'M': M,
        'agent': agent, 'choice': choice}

mu_true = [ samples[f'mu[{j + 1}]'][0][0]
            for j in range(J) ]
sigma_true = [ samples[f'sigma[{n + 1}]'][0][0]
               for n in range(N) ]
```

#### 3.4.2.2 Explore Data

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

```
putil.plot_line_hist(plot.gca(), data['choice'],
                     0.5, data['J'] + 0.5, 1,
                     xlabel="Chosen Alternative")
plot.show()
```



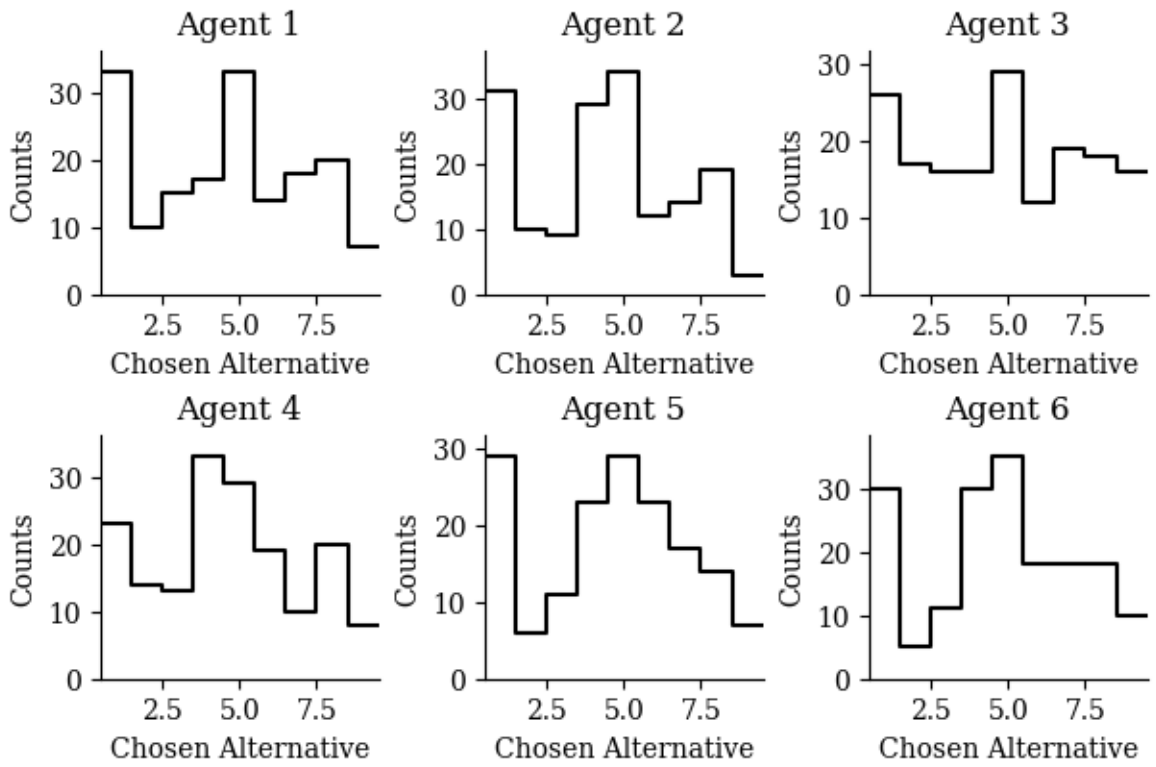
We can also stratify by agents, binning just the observed choices made one individual.

```
f, axarr = plot.subplots(2, 3, layout="constrained")

for n in range(data['N']):
  n1 = n // 3
  n2 = n % 3

  putil.plot_line_hist(
    axarr[n1, n2],
    [ c for c, a in
      zip(data['choice'], data['agent'])
      if a == n + 1                        ],
    0.5, data['J'] + 0.5, 1,
    xlabel="Chosen Alternative",
    title=f'Agent {n + 1}'
  )

plot.show()
```



#### 3.4.2.3 Redundant Implementation

To draw inferences from these simulated observations, we need to implement 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. Let's see just how bad these problems actually are.

For now, we'll avoid the challenge of developing a principled prior model for the absolute baseline utilities and agent scales. Instead, we'll use a *completely arbitrary* prior model and hope for the best.

```
with open('stan_programs/ig1.stan', 'r') as file:
  stan_program = file.read()
model = stan.build(stan_program, random_seed=8438338, data=data)
fit = model.sample(num_samples=1024, refresh=0)
```

```
Building...
```

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)
```

```
samples = util.extract_expectand_vals(fit)
base_samples = util.filter_expectands(samples,
                                      ['mu', 'sigma'],
                                      True)
util.check_all_expectand_diagnostics(base_samples)
```

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

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

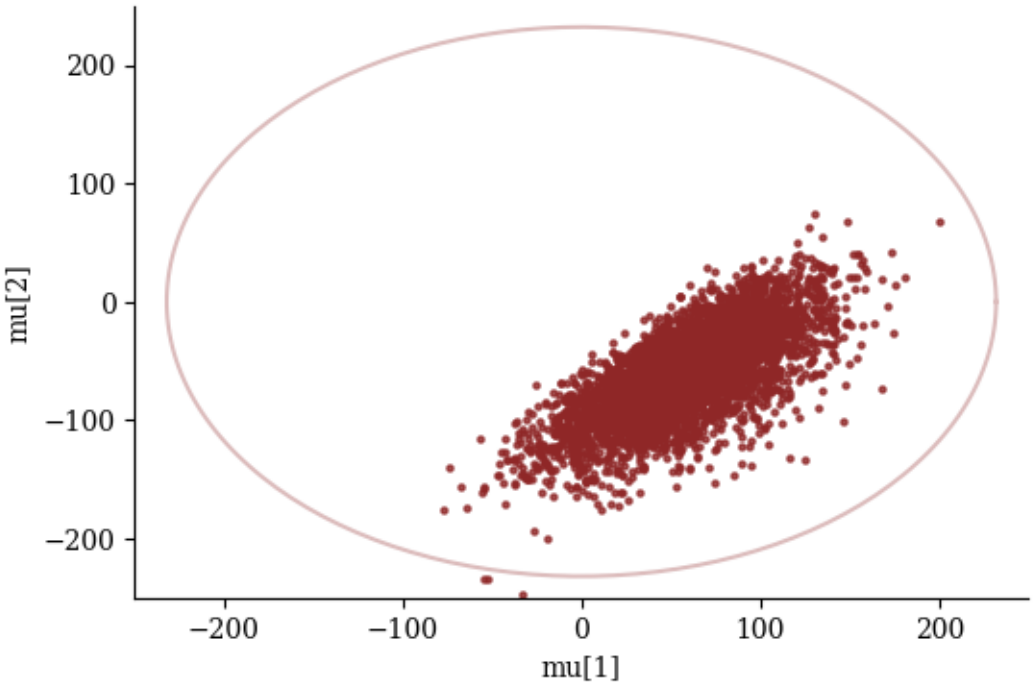
That said, the arbitrary prior model is doing a lot of work here to keep our inferences contained to a finite neighborhood where the computational burden is more manageable. Ultimately, we don't actually learn much about each absolute baseline utility.

```
plot.plot([ 232 * math.cos(x) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          [ 232 * math.sin(x) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          color = putil.light)

plot.scatter(samples["mu[1]"], samples["mu[2]"],
             color = putil.dark, s=5, alpha=0.8)

plot.gca().set_xlabel("mu[1]")
plot.gca().set_xlim([-250, 250])
plot.gca().set_ylabel("mu[2]")
plot.gca().set_ylim([-250, 250])

plot.show()
```




Our inferences for the agent scales are also mostly informed by the arbitrary prior model.

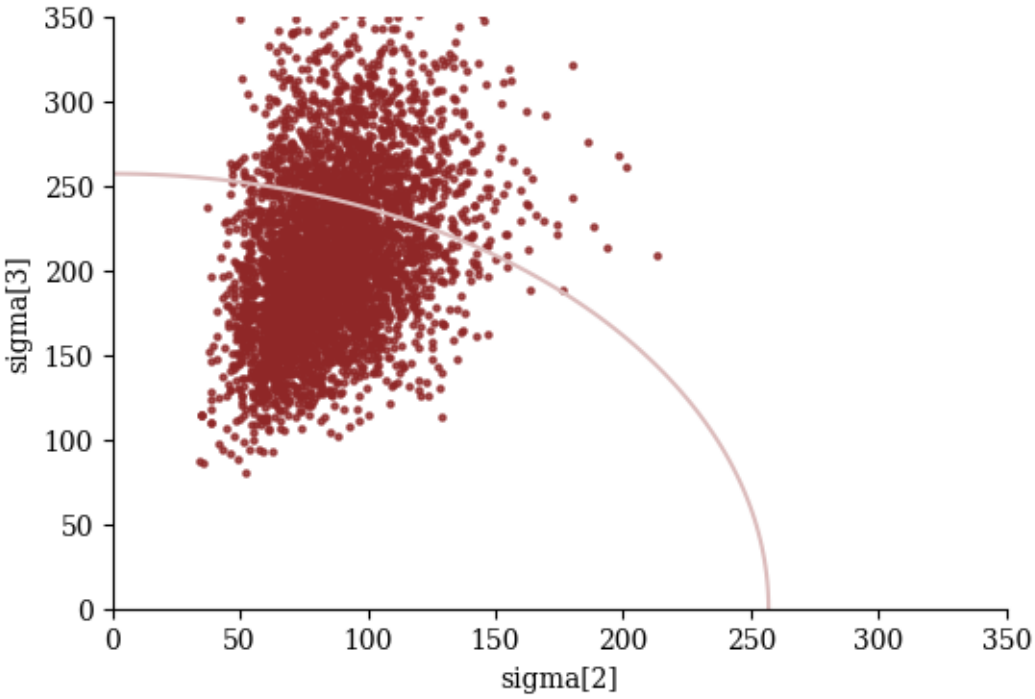
```
plot.plot([ 257 * math.cos(x) for x in
            numpy.linspace(0, 0.5 * math.pi, 100) ],
          [ 257 * math.sin(x) for x in
```

```
            numpy.linspace(0, 0.5 * math.pi, 100) ],
          color = putil.light)

plot.scatter(samples["sigma[2]"], samples["sigma[3]"],
             color = putil.dark, s=5, alpha=0.8)

plot.gca().set_xlabel("sigma[2]")
plot.gca().set_xlim([0, 350])
plot.gca().set_ylabel("sigma[3]")
plot.gca().set_ylim([0, 350])

plot.show()
```




#### 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 a reference 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 efficient as possible.

Because the observations are relatively uniform across agents, let's just anchor the first agent for simplicity.

```
print('\n'.join([ '\t'.join([str(x) for x in line])
                  for line in
                  numpy.unique(data['agent'],
                               return_counts=True)]))
```

```
1   2   3   4   5   6
167 161 169 169 159 175
```

The observed choices, however, are much more heterogeneous. Let's anchor the alternative that has been chosen the most often.

```
print('\n'.join([ '\t'.join([str(x) for x in line])
                  for line in
                  numpy.unique(data['choice'],
                               return_counts=True)]))

data['anchor'] = 5
```

```
1   2   3   4   5   6   7   8   9
172 62  75  148 189 98  96  109 51
```

Now that we're considering relative model configurations, prior modeling becomes much more straightforward. Because we don't have any actual domain expertise here, however, we'll have to use arbitrary prior containment thresholds,

$$\begin{aligned} p\left(\omega_{nj}\right) &= \text{normal}\left(\omega_{nj} \,\middle|\, 0, \frac{\log 100}{2.32}\right) \\ &\approx \text{normal}(\omega_{nj} \mid 0, 2) \end{aligned}$$

and

$$\begin{aligned} p\left(\tau_{n}\right) &= \text{log-normal}\left(\tau_{n} \,\middle|\, 0, \frac{\log 10}{2.32}\right) \\ &\approx \text{log-normal}(\tau_{n} \mid 0, 1). \end{aligned}$$

With all of this preparation, we are finally ready to implement our model. Note that we directly model only the `N - 1` unanchored scales and the `J - 1` unanchored alternatives. The `add_anchor` function inserts the missing zero back into the relative baseline utilities in the appropriate position.

```
with open('stan_programs/ig2.stan', 'r') as file:
  stan_program = file.read()
model = stan.build(stan_program, random_seed=8438338, data=data)
fit = model.sample(num_samples=1024, refresh=0)
```

```
Building...
```

None of the diagnostics suggest inaccurate posterior quantification.

```
diagnostics = util.extract_hmc_diagnostics(fit)
util.check_all_hmc_diagnostics(diagnostics)
```

```
samples = util.extract_expectand_vals(fit)
base_samples = util.filter_expectands(samples,
                                      ['omega_free',
                                       'tau_free'],
                                      True)
util.check_all_expectand_diagnostics(base_samples)
```

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

tau_free[2]:
  Chain 1: Right tail hat{xi} (0.267) exceeds 0.25.


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

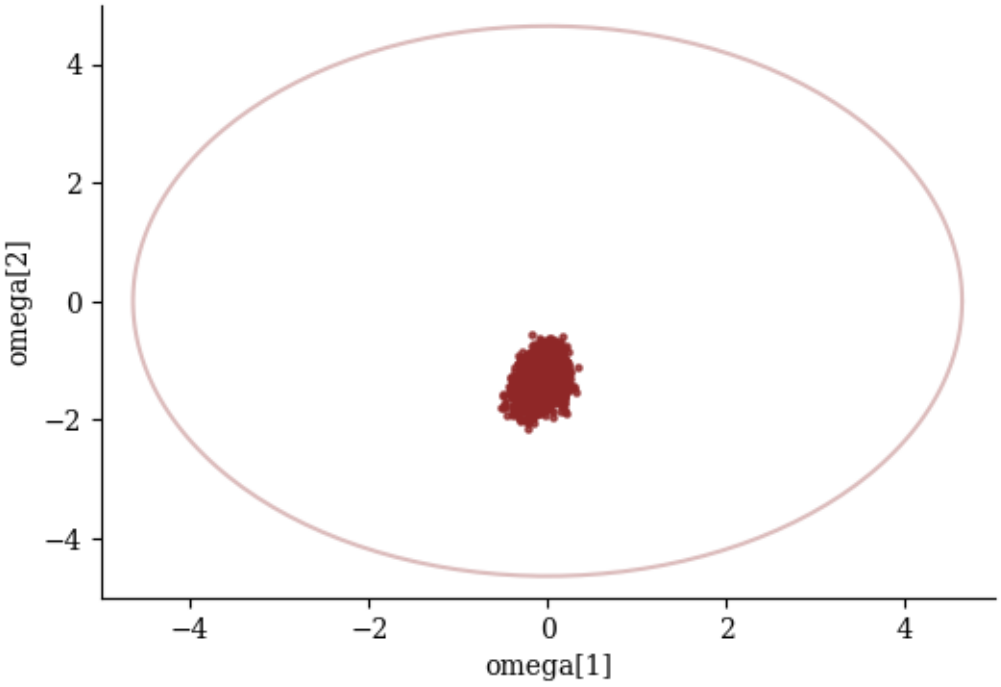
Free of redundancies, the likelihood function for the relative model configurations strongly informs our posterior inferences, even relative to the more informative prior model.

```
plot.plot([ 2 * 2.32 * math.cos(x) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          [ 2 * 2.32 * math.sin(x) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          color = putil.light)
```

```
plot.scatter(samples["omega[1]"], samples["omega[2]"],
             color = putil.dark, s=5, alpha=0.8)

plot.gca().set_xlabel("omega[1]")
plot.gca().set_xlim([-5, 5])
plot.gca().set_ylabel("omega[2]")
plot.gca().set_ylim([-5, 5])

plot.show()
```



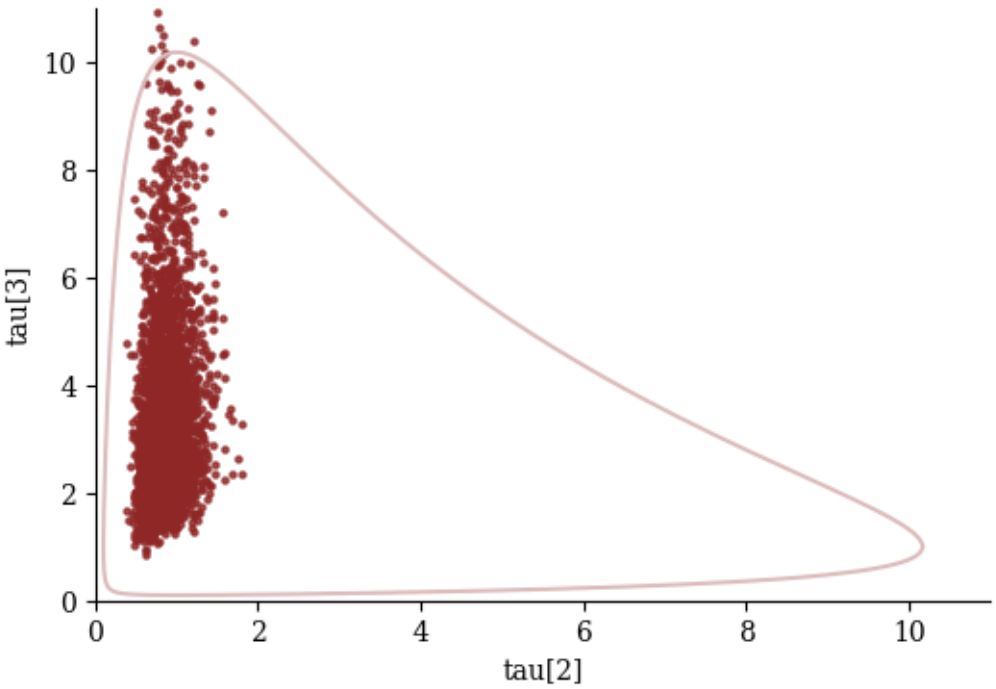

```
plot.plot([ math.exp(2.32 * math.cos(x)) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          [ math.exp(2.32 * math.sin(x)) for x in
            numpy.linspace(0, 2 * math.pi, 250) ],
          color = putil.light)

plot.scatter(samples["tau[2]"], samples["tau[3]"],
             color = putil.dark, s=5, alpha=0.8)

plot.gca().set_xlabel("tau[2]")
plot.gca().set_xlim([0, 11])
```

```
plot.gca().set_ylabel("tau[3]")
plot.gca().set_ylim([0, 11])

plot.show()
```

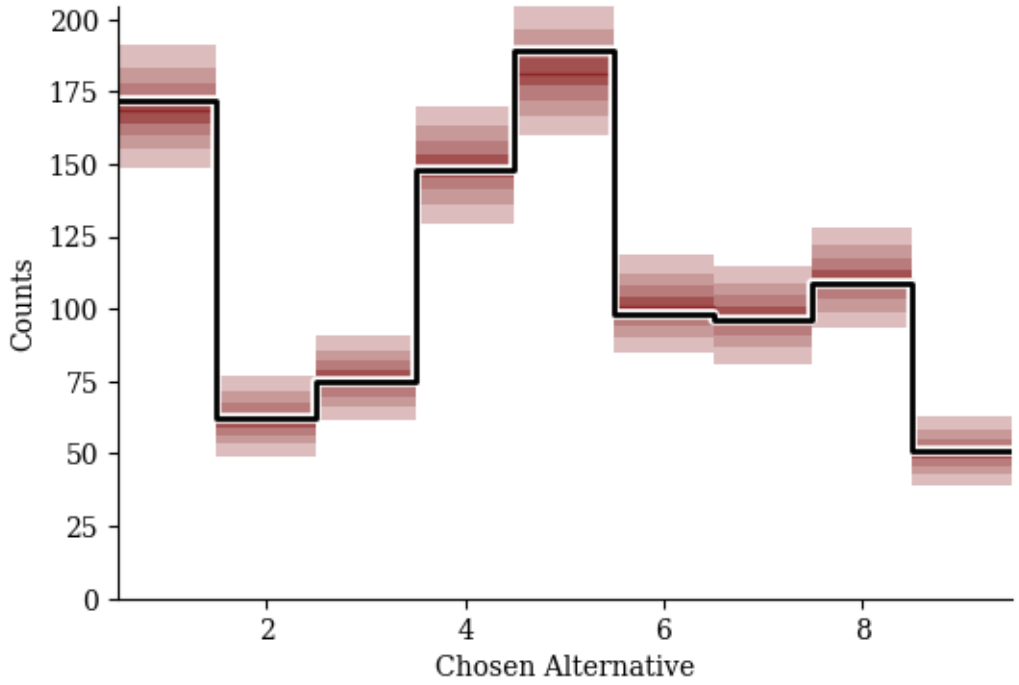



Because we're working with simulated data here, we know that the independent Gumbel model adequately captures the relevant features of the true data generating process. In practice, however, we would have to rely on posterior retrodictive checks to identify any inadequacies in our modeling assumptions.

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

```
putil.plot_hist_quantiles(plot.gca(), samples, 'choice_pred',
                          0.5, data['J'] + 0.5, 1,
                          baseline_values=data['choice'],
                          xlabel='Chosen Alternative')

plot.show()
```



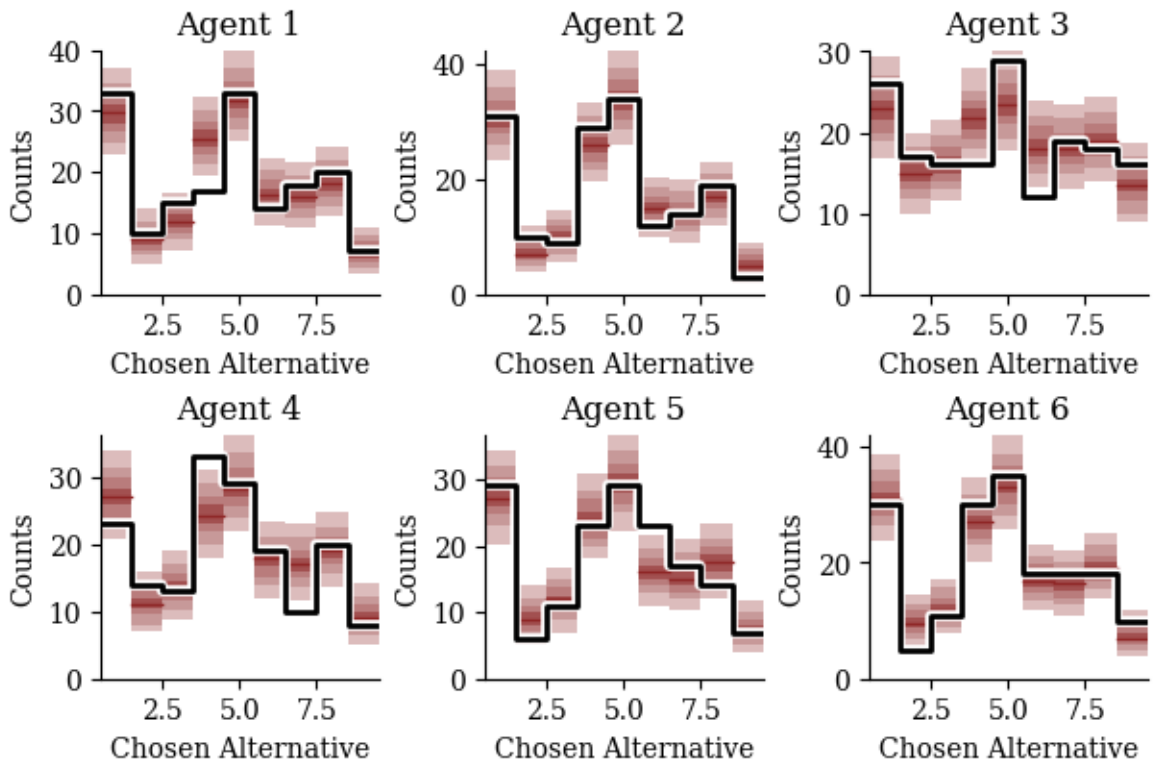

```
f, axarr = plot.subplots(2, 3, layout="constrained")

for n in range(data['N']):
  obs = [ c for c, a in zip(data['choice'], data['agent'])
          if a == n + 1 ]

  names = [ f'choice_pred[{m + 1}]'
            for m, a in enumerate(data['agent'])
            if a == n + 1 ]
  filtered_samples = util.filter_expectands(samples, names)

  n1 = n // 3
  n2 = n % 3
  putil.plot_hist_quantiles(
    axarr[n1, n2], filtered_samples, 'choice_pred',
    0.5, data['J'] + 0.5, 1,
    baseline_values=obs,
    xlabel='Chosen Alternative',
    title=f'Agent {n + 1}'
  )

plot.show()
```

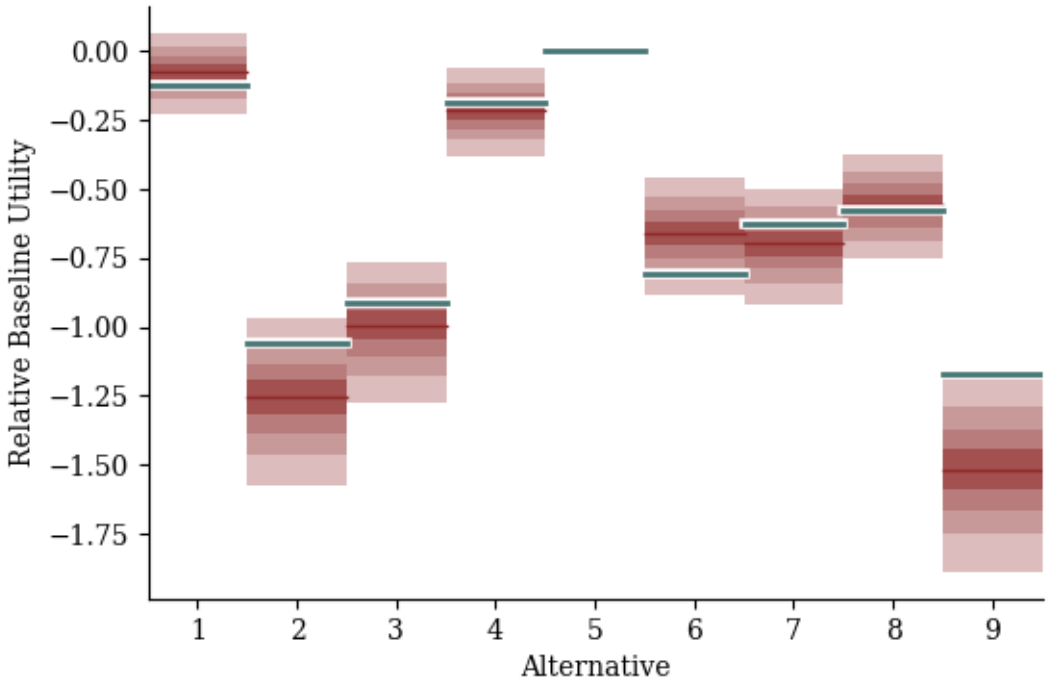



Having confirmed an adequate model, we can be confident that our posterior inferences are faithfully quantifying the behavior of the underlying data generating process, including the relative baseline utilities for each alternative and the relative scales for each agent.

```
omega_true = (mu_true - mu_true[data['anchor'] - 1]) / sigma_true[0]

names = [ f'omega[{j + 1}]' for j in range(data['J']) ]
putil.plot_disc_pushforward_quantiles(
  plot.gca(), samples, names,
  baseline_values=omega_true,
  baseline_color=putil.mid_teal,
  xlabel="Alternative",
  xticklabels=[ j + 1 for j in range(data['J']) ],
  ylabel="Relative Baseline Utility"
)

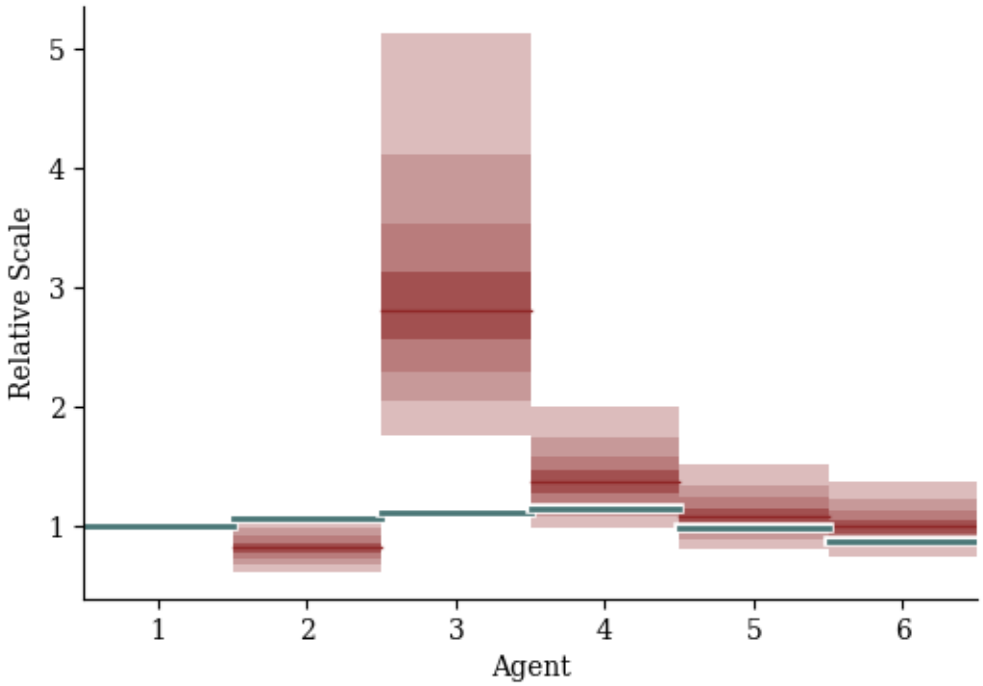
plot.show()
```




```
tau_true = sigma_true / sigma_true[0]

names = [ f'tau[{n + 1}]' for n in range(data['N']) ]
putil.plot_disc_pushforward_quantiles(
  plot.gca(), samples, names,
  baseline_values=tau_true,
  baseline_color=putil.mid_teal,
  xlabel="Agent",
  xticklabels=[ n + 1 for n in range(data['N']) ],
  ylabel="Relative Scale"
)

plot.show()
```




#### 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 model much less expensive to evaluate.

```
choice_counts = numpy.array(
  [ numpy.histogram([ c for c, a in
                      zip(data['choice'], data['agent'])
                      if a == n + 1 ],
                    bins=numpy.arange(0.5, data['J'] + 1, 1))[0]
    for n in range(data['N']) ]
)

data['choice_counts'] = choice_counts
```

```
with open('stan_programs/ig3.stan', 'r') as file:
  stan_program = file.read()
model = stan.build(stan_program, random_seed=8438338, data=data)
fit = model.sample(num_samples=1024, refresh=0)
```

```
Building...
```

The only diagnostic warnings that we see here are a few $\hat{xi}$ warnings. These suggest heavy-tailed posteriors, and some potentially ill-defined 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)


samples = util.extract_expectand_vals(fit)
base_samples = util.filter_expectands(samples,
                                      ['omega_free',
                                       'tau_free'],
                                      True)
util.check_all_expectand_diagnostics(base_samples)
```

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

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

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.

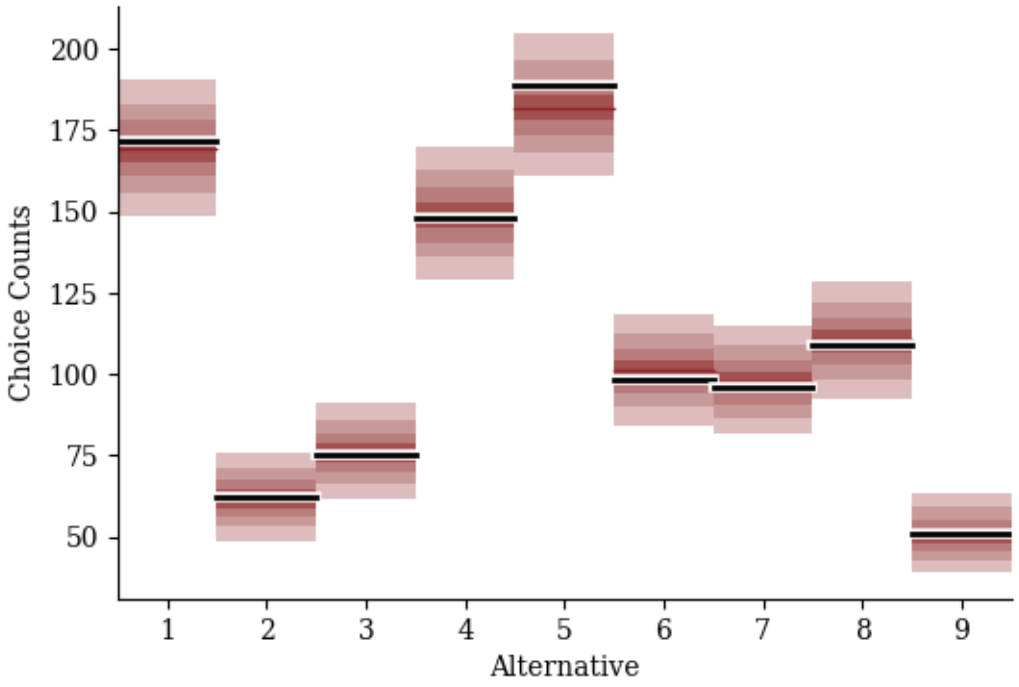
```
obs_counts = numpy.sum(data['choice_counts'], axis=0)

names = [ f'agg_choice_counts_pred[{j + 1}]'
          for j in range(data['J']) ]

putil.plot_disc_pushforward_quantiles(
  plot.gca(), samples, names,
```

```
  baseline_values=obs_counts,
  xlabel="Alternative",
  xticklabels=[ j + 1 for j in range(data['J']) ],
  ylabel="Choice Counts"
)

plot.show()
```




```
f, axarr = plot.subplots(2, 3, layout="constrained")

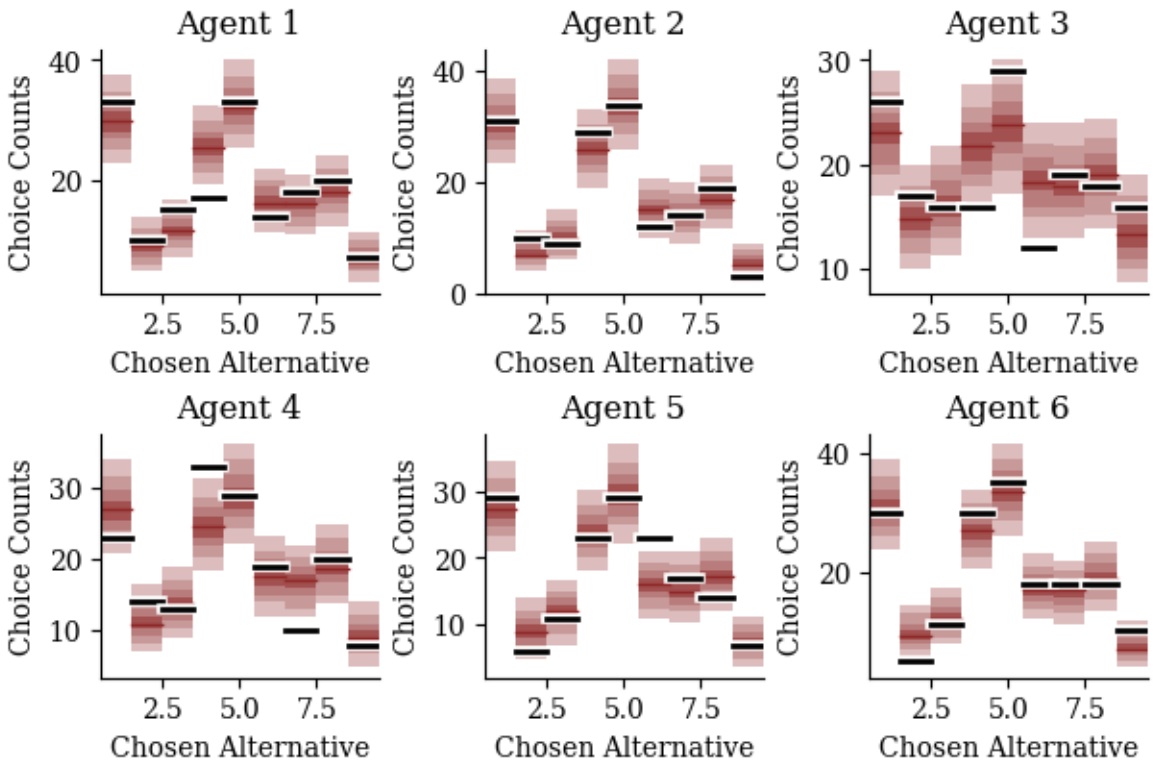
for n in range(data['N']):
  obs_counts = data['choice_counts'][n,:]

  names = [ f'choice_counts_pred[{n + 1},{j + 1}]'
            for j in range(data['J']) ]

  n1 = n // 3
  n2 = n % 3
  putil.plot_disc_pushforward_quantiles(
    axarr[n1, n2], samples, names,
    baseline_values=obs_counts,
```

```
    xlabel='Chosen Alternative',
    ylabel='Choice Counts',
    title=f'Agent {n + 1}'
  )

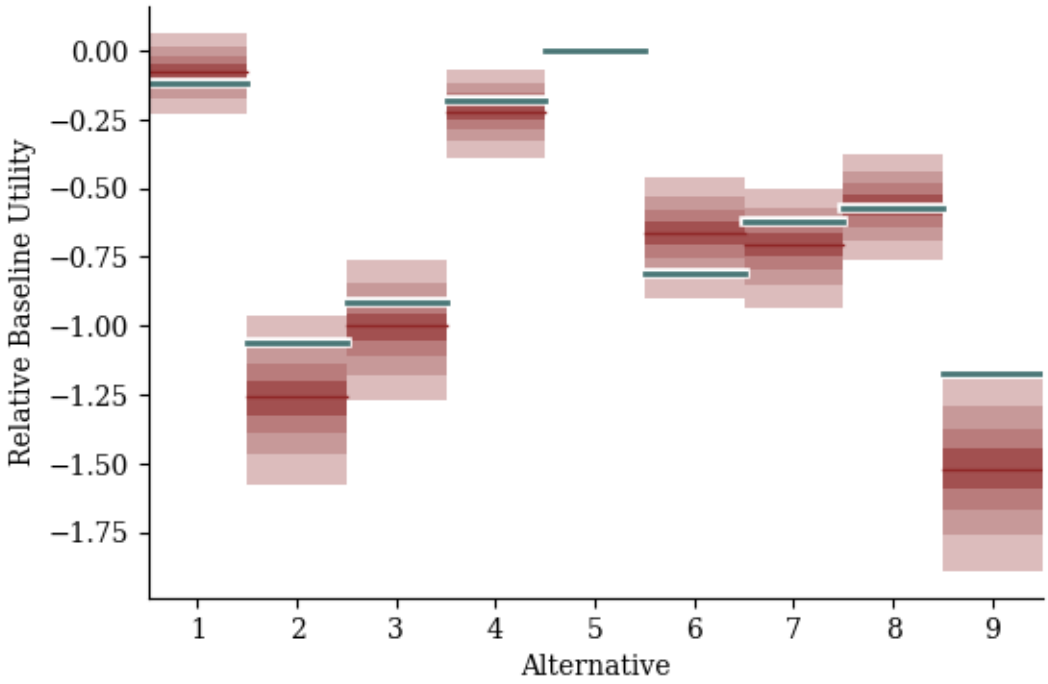
plot.show()
```




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

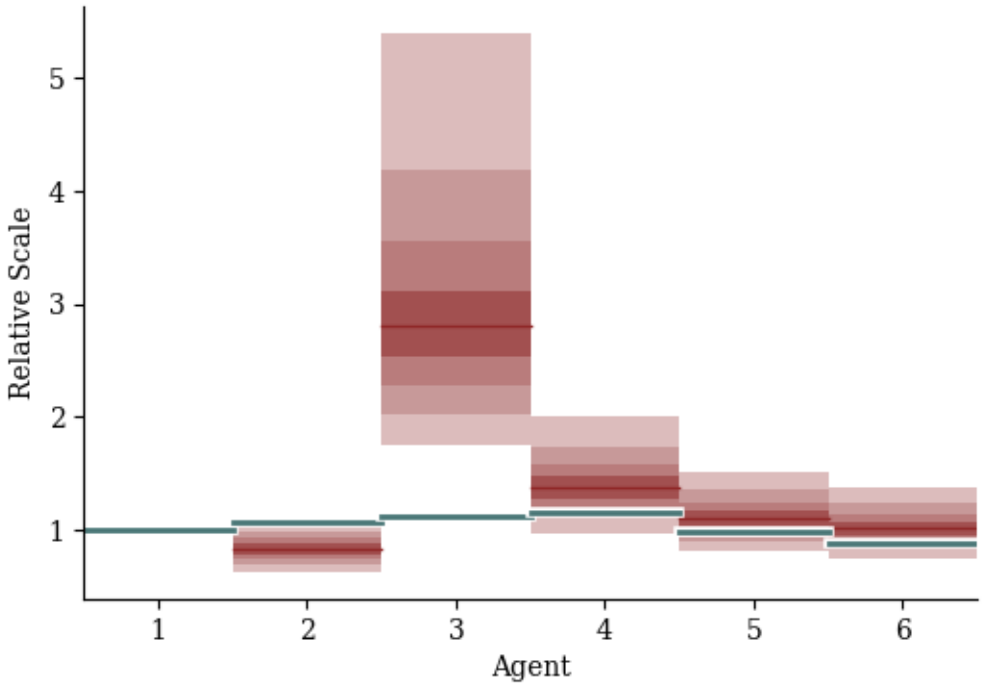
```
names = [ f'omega[{j + 1}]' for j in range(data['J']) ]
putil.plot_disc_pushforward_quantiles(
  plot.gca(), samples, names,
  baseline_values=omega_true,
  baseline_color=putil.mid_teal,
  xlabel="Alternative",
  xticklabels=[ j + 1 for j in range(data['J']) ],
  ylabel="Relative Baseline Utility"
)
```

```
plot.show()
```




```
names = [ f'tau[{n + 1}]' for n in range(data['N']) ]
putil.plot_disc_pushforward_quantiles(
  plot.gca(), samples, names,
  baseline_values=tau_true,
  baseline_color=putil.mid_teal,
  xlabel="Agent",
  xticklabels=[ n + 1 for n in range(data['N']) ],
  ylabel="Relative Scale"
)

plot.show()
```




### 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 it 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 from all of the possible alternatives, and then individual choice subsets from which we'll simulate observations.

Let's consider 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 here includes all but the eighth alternative. The second choice subset also removes the fifth alternative, while the third choice subset adds back in both missing alternatives. In other words, we can think of the second and third choice subsets as the result of removing and adding an alternative from the first choice subset, respectively.

```
subset_alts = \
  [ [1, 2, 3, 4, 5, 6, 7,    9, 10], # Nominal
    [1, 2, 3, 4,    6, 7,    9, 10], # Remove alternative 5
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  # Add alternative 8
  ]

S = len(subset_alts)
```

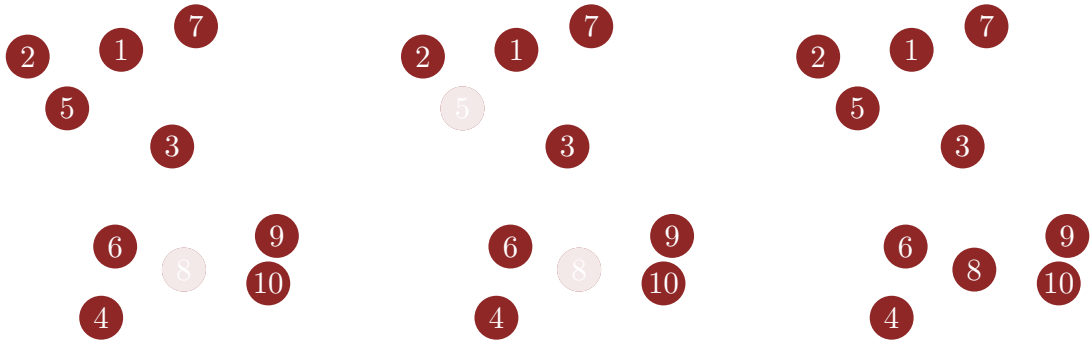



Figure 2: To study substitutions patterns for the independent Gumbel model, we'll consider three choice subsets. The first choice subset contains nine of the ten possible alternatives, and the second choice subset removes one more alternative. Lastly, the third choice subset is given by including all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

```
alts = [ j + 1 for j in range(J) ]

to_subset_idx = [ [ subset_alt.index(j) + 1
                    if j in subset_alt else None
                    for j in alts                ]
                  for subset_alt in subset_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, and ragged arrays cannot be implemented directly in the `Stan` modeling language.

One way to accommodate this ragged structure is to 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 slice through the monolithic array and retrieve the appropriate segment.

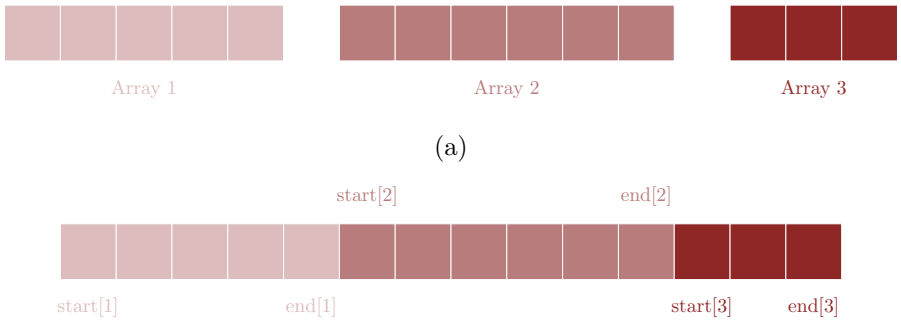



Figure 3: (a) Ragged arrays combine multiple one-dimensional arrays of different sizes. Unfortunately, they are not immediately accommodated by the rectangular arrays in the `Stan` modeling language. (b) That said, we can always implement a ragged array by concatenated the individual component arrays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

```
SJ = 0
alts = []
subset_start = []
subset_end = []

for s in range(S):
  subset_start.append(SJ + 1)
  SJ += len(subset_alts[s])
  subset_end.append(SJ)

  alts += subset_alts[s]
```

With the organization of the choice subsets handled, we can specify the remaining 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.

```
with open('stan_programs/simu_ig2.stan', 'r') as file:
  stan_program = file.read()
simu_model = stan.build(stan_program, random_seed=8438338 + 1,
                        data={'N': N, 'J': J, 'M': M,
                              'S': S, 'SJ': SJ,
                              'subset_start': subset_start,
                              'subset_end': subset_end,
                              'alts': alts})
simu = simu_model.fixed_param(num_chains=1, num_samples=1)

samples = util.extract_expectand_vals(simu)
```

```
Building...
```

```
mu_true = [ samples[f'mu[{j + 1}]'][0][0]
            for j in range(J) ]
sigma_true = [ samples[f'sigma[{n + 1}]'][0][0]
               for n in range(N) ]
```

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

```
agent = [ samples[f'agent[{m + 1}]'][0][0].astype(numpy.int64)
          for m in range(M) ]
choice = [ samples[f'choice[{m + 1}]'][0][0].astype(numpy.int64)
           for m in range(M) ]
subset = [ samples[f'subset[{m + 1}]'][0][0].astype(numpy.int64)
           for m in range(M) ]

data = []
```

```
for s in range(S):
  JJ = subset_end[s] - subset_start[s] + 1
  choice_counts = numpy.empty([N, JJ])
  for n in range(N):
    choice_counts[n,:] = \
      numpy.histogram([ c for c, aa, ss in
                        zip(choice, agent, subset)
                        if aa == n + 1 and ss == s + 1 ],
                      bins=numpy.arange(0.5, JJ + 1, 1)  )[0]

  data.append({'N': N, 'J': JJ,
               'choice_counts': choice_counts.astype(numpy.int64) })
```

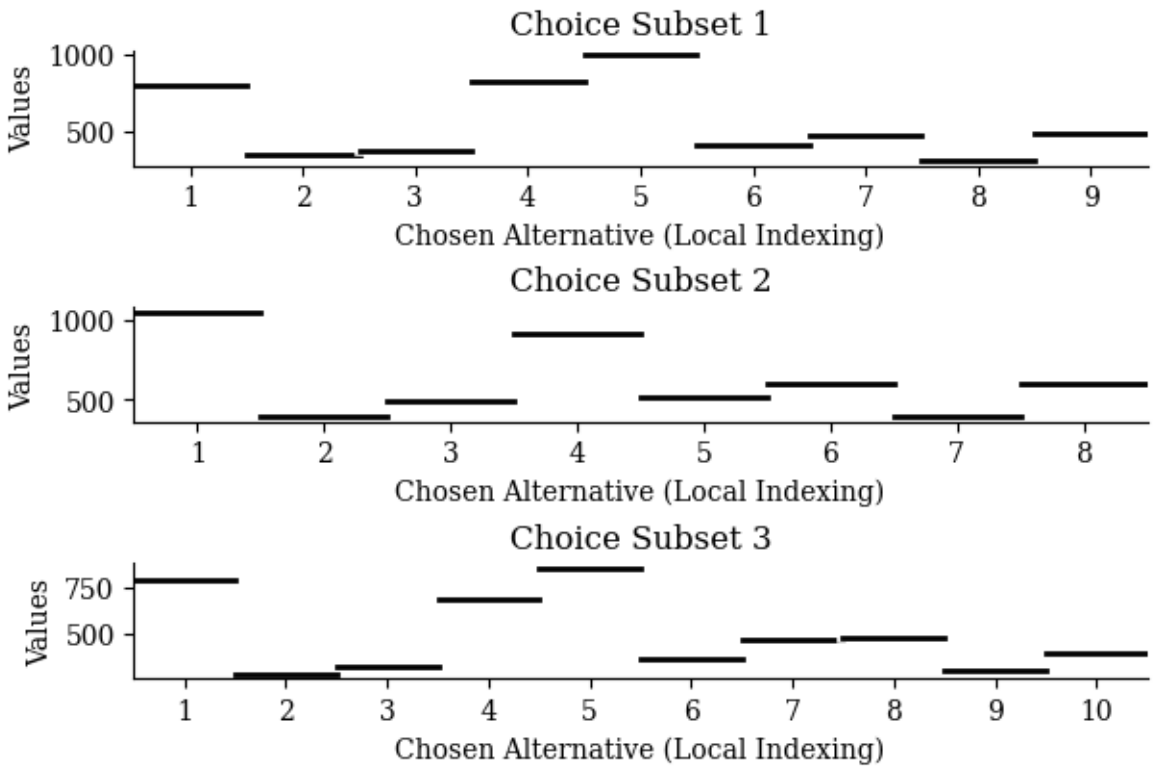
#### 3.4.3.2 Explore Data

In theory, we could aggregate all of the observed choices into a histogram with respect to the global choice set alternatives. This crude aggregation, however, would also be difficult to interpret. Because we are considering only a few choice subsets here, we can just as easily examine separate histograms for each choice subset.

```
f, axarr = plot.subplots(3, 1, layout="constrained")

for s in range(S):
  putil.plot_disc_vals(
    axarr[s],
    numpy.sum(data[s]['choice_counts'], axis=0),
    xlabel="Chosen Alternative (Local Indexing)",
    xticklabels=[ j + 1 for j in range(data[s]['J']) ],
    title=f'Choice Subset {s + 1}'
  )

plot.show()
```




We could also stratify each of these choice subset histograms by the corresponding agent if useful. If the number of choice subsets becomes unmanageable, we might also consider reducing each histogram to one-dimensional summary statistics, such as empirical means and empirical variances, or even empirical entropies, and then histogramming the summaries across all choice subsets.

#### 3.4.3.3 Individual Posterior Quantification

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

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

```
common_alts = set.intersection(
  *map(set, [ alts[(subset_start[s] - 1):subset_end[s]]
              for s in range(S) ])
)
```

```
for alt in common_alts:
  print(f'Alternative {alt}:')
  for s in range(S):
    local_idx = to_subset_idx[s][alt - 1] - 1
    MM = int(sum(data[s]['choice_counts'][:, local_idx]))
    print(f'  {MM} observations in choice subset {s + 1}')
```

```
Alternative 1:
  801 observations in choice subset 1
  1053 observations in choice subset 2
  793 observations in choice subset 3
Alternative 2:
  359 observations in choice subset 1
  403 observations in choice subset 2
  284 observations in choice subset 3
Alternative 3:
  375 observations in choice subset 1
  491 observations in choice subset 2
  318 observations in choice subset 3
Alternative 4:
  826 observations in choice subset 1
  912 observations in choice subset 2
  686 observations in choice subset 3
Alternative 6:
  422 observations in choice subset 1
  516 observations in choice subset 2
  367 observations in choice subset 3
Alternative 7:
  485 observations in choice subset 1
  603 observations in choice subset 2
  468 observations in choice subset 3
Alternative 9:
  318 observations in choice subset 1
  394 observations in choice subset 2
  297 observations in choice subset 3
Alternative 10:
  487 observations in choice subset 1
  603 observations in choice subset 2
  400 observations in choice subset 3
```

For this exercise, we'll anchor the ninth alternative, which has the most observations among

those that appear in all of the choice subsets.

```
anchor = 1
```

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

```
for s in range(S):
  data[s]['anchor'] = to_subset_idx[s][anchor - 1]
```

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

```
samples = [ None ] * S

for s in range(S):
  with open('stan_programs/ig3.stan', 'r') as file:
    stan_program = file.read()
  model = stan.build(stan_program,
                     random_seed=8438338,
                     data=data[s])
  fit = model.sample(num_samples=1024, refresh=0)

  print(f'Analyzing choice subset {s + 1}\n')

  diagnostics = util.extract_hmc_diagnostics(fit)
  util.check_all_hmc_diagnostics(diagnostics)


  samples[s] = util.extract_expectand_vals(fit)
  base_samples = util.filter_expectands(samples[s],
                                        ['omega_free',
                                         'tau_free'],
                                        True)
  util.check_all_expectand_diagnostics(base_samples)

  print('\n\n')
```

```
Building...
Analyzing choice subset 1

All Hamiltonian Monte Carlo diagnostics are consistent with accurate
```

```
Markov chain Monte Carlo.

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

```
Building...
Analyzing choice subset 2

All Hamiltonian Monte Carlo diagnostics are consistent with accurate
Markov chain Monte Carlo.

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

```
Building...
Analyzing choice subset 3

All Hamiltonian Monte Carlo diagnostics are consistent with accurate
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 = [ None ] * S
j2_probs = [ None ] * S

for s in range(S):
  var_repl = {'x': f'log_probs[{n},{to_subset_idx[s][j1 - 1]}]'}

  j1_probs[s] = \
    util.eval_expectand_pushforward(samples[s],
                                    lambda x: math.exp(x),
                                    var_repl)

  var_repl = {'x': f'log_probs[{n},{to_subset_idx[s][j2 - 1]}]'}

  j2_probs[s] = \
    util.eval_expectand_pushforward(samples[s],
                                    lambda x: math.exp(x),
                                    var_repl)
```

The initial behavior is given by the choice probabilities $q_{5,3}$ and $q_{5,10}$ in the first choice subset.

```
B = 50
flim = [0.045, 0.145]

f, axarr = plot.subplots(2, 1, layout="constrained")

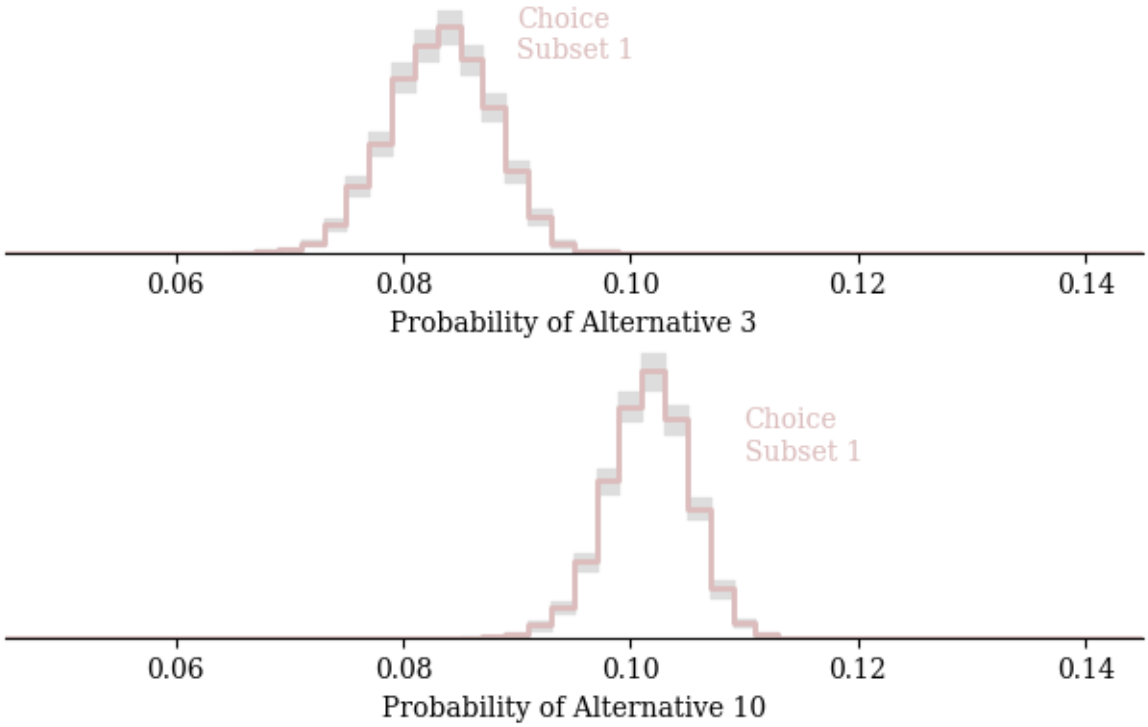
display_name = f'Probability of Alternative {j1}'
util.plot_expectand_pushforward(axarr[0], j1_probs[0], B,
                                flim=flim, ylim=[0, 110],
                                display_name=display_name,
                                color=putil.light)
axarr[0].text(0.09, 75, 'Choice\nSubset 1', color=putil.light)



display_name = f'Probability of Alternative {j2}'
util.plot_expectand_pushforward(axarr[1], j2_probs[0], B,
                                flim=flim, ylim=[0, 120],
                                display_name=display_name,
                                color=putil.light)
```

```
axarr[1].text(0.11, 75, 'Choice\nSubset 1', color=putil.light)


plot.show()
```




Moving from the first to the second choice subset requires removing an alternative. The probability initially allocated to that alternative has to be redistributed among the remaining alternatives, increasing their choice probabilities. As expected, both $q_{5,3}$ and $q_{5,10}$ *increase*.

```
B = 50
flim = [0.045, 0.145]

f, axarr = plot.subplots(2, 1, layout="constrained")

display_name = f'Probability of Alternative {j1}'
util.plot_expectand_pushforward(axarr[0], j1_probs[0], B,
                                flim=flim, ylim=[0, 110],
                                display_name=display_name,
                                color=putil.light)
```

```
axarr[0].text(0.09, 75, 'Choice\nSubset 1', color=putil.light)

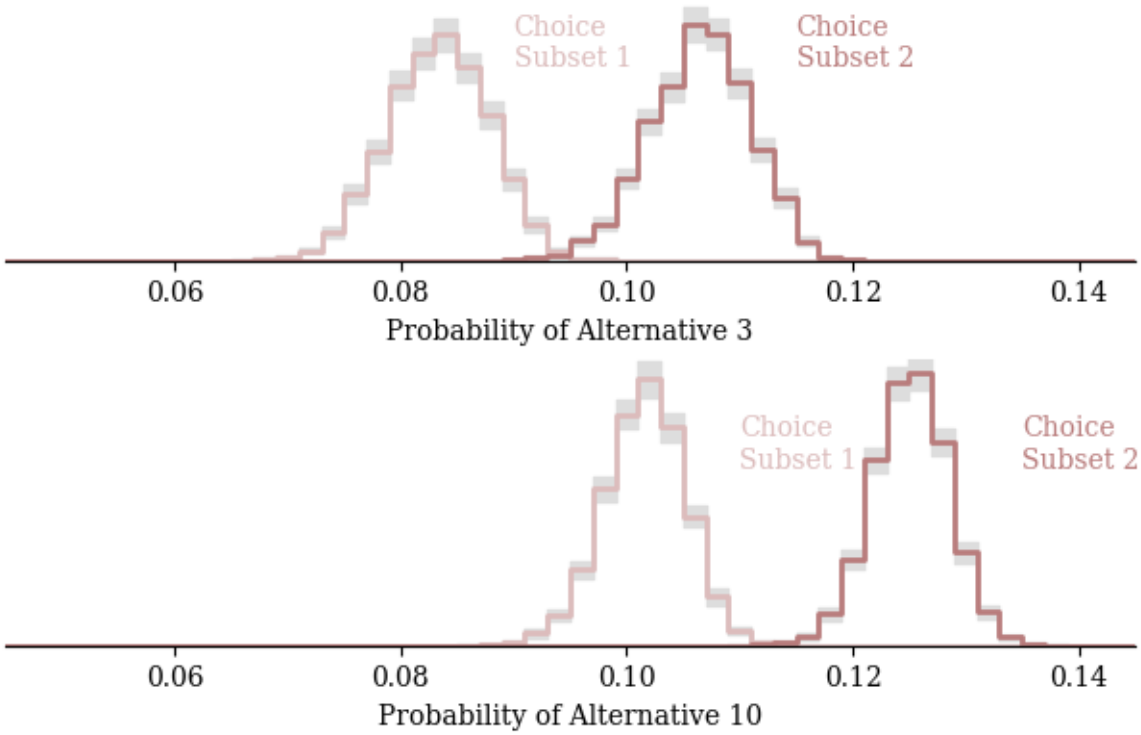
util.plot_expectand_pushforward(axarr[0], j1_probs[1], B,
                                flim=flim,
                                color=putil.mid,
                                border="#DDDDDD88",
                                add=True)
axarr[0].text(0.115, 75, 'Choice\nSubset 2', color=putil.mid)



display_name = f'Probability of Alternative {j2}'
util.plot_expectand_pushforward(axarr[1], j2_probs[0], B,
                                flim=flim, ylim=[0, 120],
                                display_name=display_name,
                                color=putil.light)
axarr[1].text(0.11, 75, 'Choice\nSubset 1', color=putil.light)

util.plot_expectand_pushforward(axarr[1], j2_probs[1], B,
                                flim=flim,
                                color=putil.mid,
                                border="#DDDDDD88",
                                add=True)
axarr[1].text(0.135, 75, 'Choice\nSubset 2', color=putil.mid)


plot.show()
```




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*.

```
B = 50
flim = [0.045, 0.145]

f, axarr = plot.subplots(2, 1, layout="constrained")

display_name = f'Probability of Alternative {j1}'
util.plot_expectand_pushforward(axarr[0], j1_probs[0], B,
                                flim=flim, ylim=[0, 110],
                                display_name=display_name,
                                color=putil.light)
axarr[0].text(0.09, 75, 'Choice\nSubset 1', color=putil.light)

util.plot_expectand_pushforward(axarr[0], j1_probs[1], B,
                                flim=flim,
                                color=putil.mid,
                                border="#DDDDDD88",
```

```
                                add=True)
axarr[0].text(0.115, 75, 'Choice\nSubset 2', color=putil.mid)

util.plot_expectand_pushforward(axarr[0], j1_probs[2], B,
                                flim=flim,
                                color=putil.dark,
                                border="#DDDDDD88",
                                add=True)
axarr[0].text(0.055, 75, 'Choice\nSubset 3', color=putil.dark)

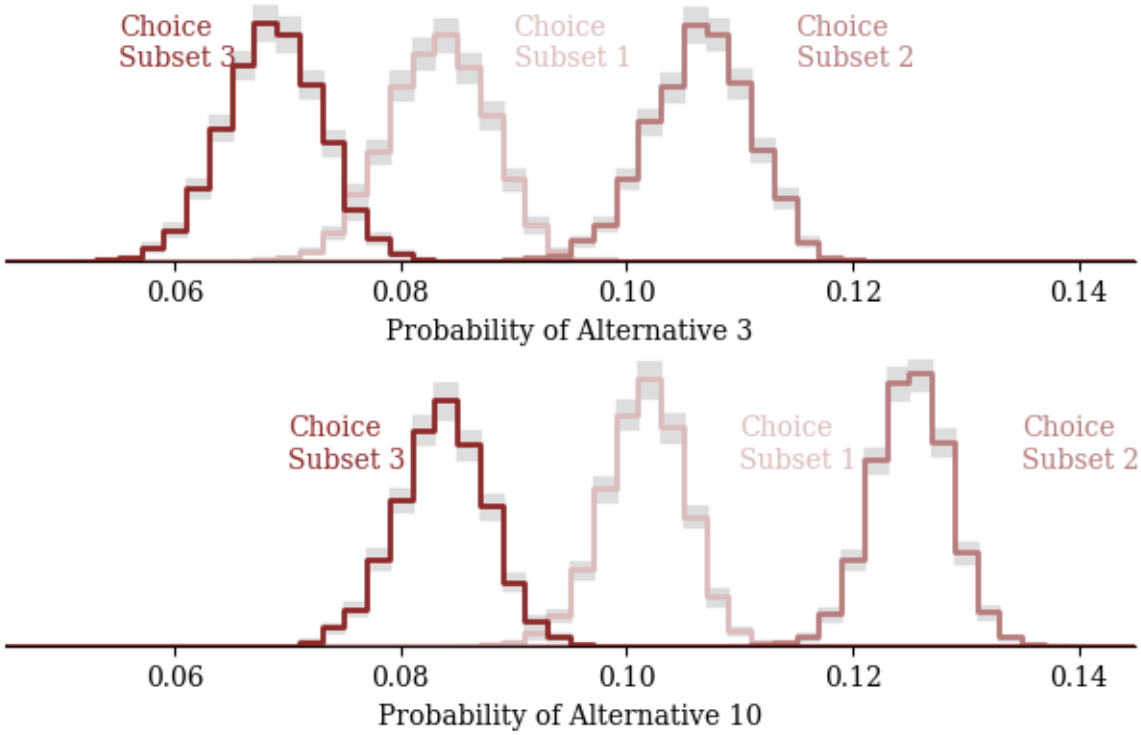


display_name = f'Probability of Alternative {j2}'
util.plot_expectand_pushforward(axarr[1], j2_probs[0], B,
                                flim=flim, ylim=[0, 120],
                                display_name=display_name,
                                color=putil.light)
axarr[1].text(0.11, 75, 'Choice\nSubset 1', color=putil.light)

util.plot_expectand_pushforward(axarr[1], j2_probs[1], B,
                                flim=flim,
                                color=putil.mid,
                                border="#DDDDDD88",
                                add=True)
axarr[1].text(0.135, 75, 'Choice\nSubset 2', color=putil.mid)

util.plot_expectand_pushforward(axarr[1], j2_probs[2], B,
                                flim=flim,
                                color=putil.dark,
                                border="#DDDDDD88",
                                add=True)
axarr[1].text(0.07, 75, 'Choice\nSubset 3', color=putil.dark)


plot.show()
```




Because the independent Gumbel model exhibits independence from irrelevant alternatives, the *ratios* of choice probabilities should be consistent across all three choice subsets.

```
ratios = [ None ] * S

for s in range(S):
  name1 = f'log_probs[{n},{to_subset_idx[s][j1 - 1]}]'
  name2 = f'log_probs[{n},{to_subset_idx[s][j2 - 1]}]'

  ratios[s] = \
    util.eval_expectand_pushforward(samples[s],
                                    lambda x1, x2: math.exp(x1 - x2),
                                    {'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*.

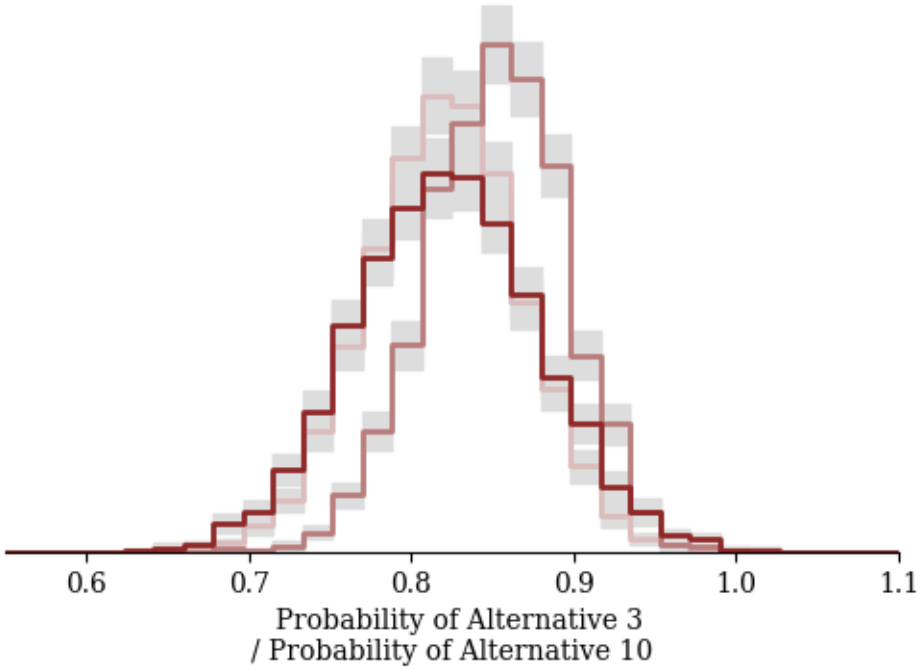
```
B = 30
flim = [0.55, 1.1]

display_name = (f'  Probability of Alternative {j1}'
                f'\n/ Probability of Alternative {j2}')
util.plot_expectand_pushforward(plot.gca(), ratios[0], B,
                                flim=flim, ylim=[0, 11],
                                display_name=display_name,
                                color=putil.light)
axarr[0].text(0.09, 75, 'Choice\nSubset 1', color=putil.light)

util.plot_expectand_pushforward(plot.gca(), ratios[1], B,
                                flim=flim,
                                color=putil.mid,
                                border="#DDDDDD88",
                                add=True)
axarr[0].text(0.115, 75, 'Choice\nSubset 2', color=putil.mid)

util.plot_expectand_pushforward(plot.gca(), ratios[2], B,
                                flim=flim,
                                color=putil.dark,
                                border="#DDDDDD88",
                                add=True)
axarr[0].text(0.055, 75, 'Choice\nSubset 3', color=putil.dark)


plot.show()
```




#### 3.4.3.5 Joint Posterior Quantification

When our model captures the appropriate substitution patterns, we don't have to limit ourselves to fitting the observations from each choice subset separately. 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 = numpy.empty([N, SJ])

for s in range(S):
  for n in range(N):
    choice_counts[n, (subset_start[s] - 1):subset_end[s]] = \
      data[s]['choice_counts'][n,:]
```

```
joint_data = { 'N': N, 'J': J,
               'S': S, 'SJ': SJ,
               'subset_start': subset_start,
               'subset_end': subset_end,
               'alts': alts,
```

```
                'choice_counts': choice_counts.astype(numpy.int64),
                'anchor': anchor }
```

One nice benefit of including multiple choice subsets at the same time is that making predictions for the behavior in *any* choice subset is straightforward. To demonstrate, we'll specifically look at predictions for the fifth agent choosing from the full choice set.

```
joint_data['n_pred'] = 5
joint_data['J_pred'] = J
joint_data['pred_alts'] = [ j + 1 for j in range(J) ]
```

Now we let Hamiltonian Monte Carlo go to work.

```
with open('stan_programs/ig_joint.stan', 'r') as file:
  stan_program = file.read()
model = stan.build(stan_program,
                   random_seed=8438338,
                   data=joint_data)
fit = model.sample(num_samples=1024, refresh=0)
```

```
Building...
```

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)


joint_samples = util.extract_expectand_vals(fit)
base_samples = util.filter_expectands(joint_samples,
                                      ['omega_free',
                                       'tau_free'],
                                      True)
util.check_all_expectand_diagnostics(base_samples)
```

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

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

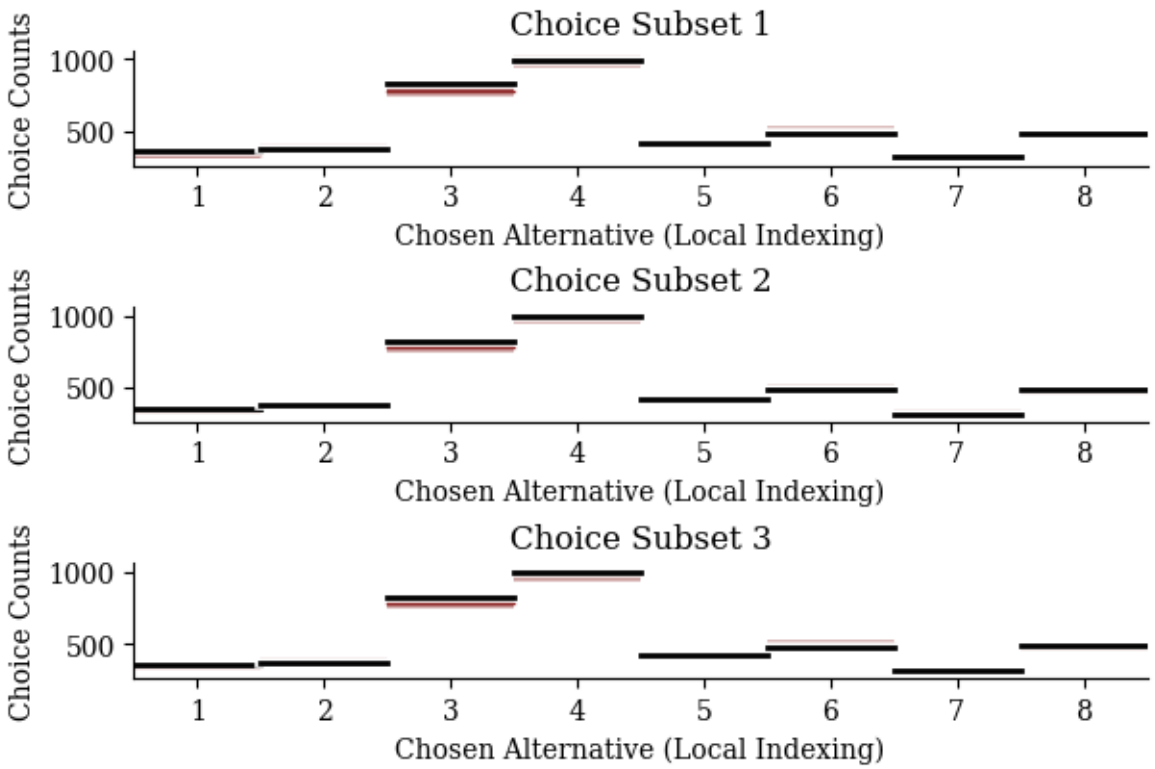
There is no appreciable retrodictive tension in any of the choice subsets.

```
f, axarr = plot.subplots(3, 1, layout="constrained")

for s in range(S):
  idxs = range(subset_start[0], subset_end[0])
  agg_choice_counts_obs = \
    numpy.sum(joint_data['choice_counts'][:,idxs], axis=0)

  names = [ f'agg_choice_counts_pred[{j + 1}]'
            for j in idxs ]
  putil.plot_disc_pushforward_quantiles(
    axarr[s], joint_samples, names,
    baseline_values=agg_choice_counts_obs,
    xlabel="Chosen Alternative (Local Indexing)",
    xticklabels=[ j + 1 for j in range(len(idxs)) ],
    ylabel='Choice Counts',
    title=f'Choice Subset {s + 1}'
  )

plot.show()
```




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 enjoy smaller uncertainties.

```
B = 50
flim = [-1.4, -0.3]

f, axarr = plot.subplots(2, 1, layout="constrained")

display_name = f'omega[{j1}]'

name = f'omega[{to_subset_idx[0][j1 - 1]}]'
util.plot_expectand_pushforward(axarr[0], samples[0][name], B,
                                flim=flim, ylim=[0, 10],
                                display_name=display_name,
                                color=putil.light)

name = f'omega[{to_subset_idx[1][j1 - 1]}]'
util.plot_expectand_pushforward(axarr[0], samples[1][name], B,
                                flim=flim,
```

```
                                color=putil.light,
                                border="#DDDDDD88",
                                add=True)

name = f'omega[{to_subset_idx[2][j1 - 1]}]'
util.plot_expectand_pushforward(axarr[0], samples[2][name], B,
                                flim=flim,
                                color=putil.light,
                                border="#DDDDDD88",
                                add=True)

name = f'omega[{j1}]'
util.plot_expectand_pushforward(axarr[0], joint_samples[name], B,
                                flim=flim,
                                color=putil.dark,
                                border="#DDDDDD88",
                                add=True)

display_name = f'omega[{j2}]'

name = f'omega[{to_subset_idx[0][j2 - 1]}]'
util.plot_expectand_pushforward(axarr[1], samples[0][name], B,
                                flim=flim, ylim=[0, 10],
                                display_name=display_name,
                                color=putil.light)

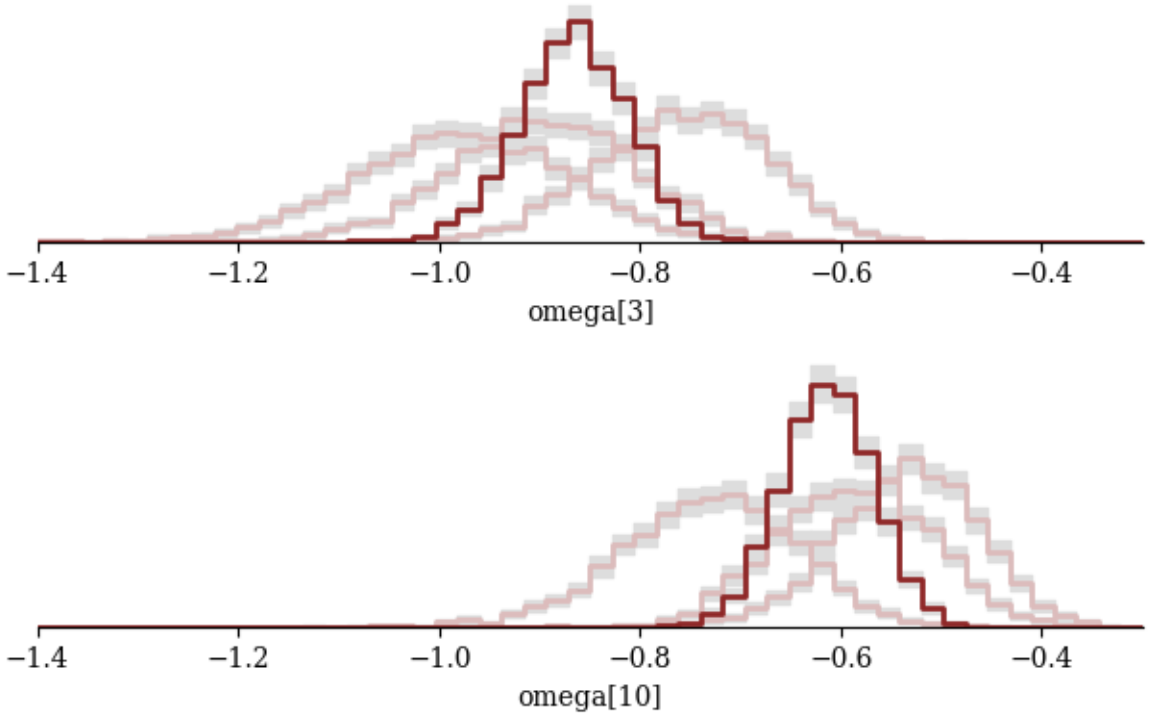
name = f'omega[{to_subset_idx[1][j2 - 1]}]'
util.plot_expectand_pushforward(axarr[1], samples[1][name], B,
                                flim=flim,
                                color=putil.light,
                                border="#DDDDDD88",
                                add=True)

name = f'omega[{to_subset_idx[2][j2 - 1]}]'
util.plot_expectand_pushforward(axarr[1], samples[2][name], B,
                                flim=flim,
                                color=putil.light,
                                border="#DDDDDD88",
                                add=True)

name = f'omega[{j2}]'
util.plot_expectand_pushforward(axarr[1], joint_samples[name], B,
```

```
                                  flim=flim,
                                  color=putil.dark,
                                  border="#DDDDDD88",
                                  add=True)

plot.show()
```



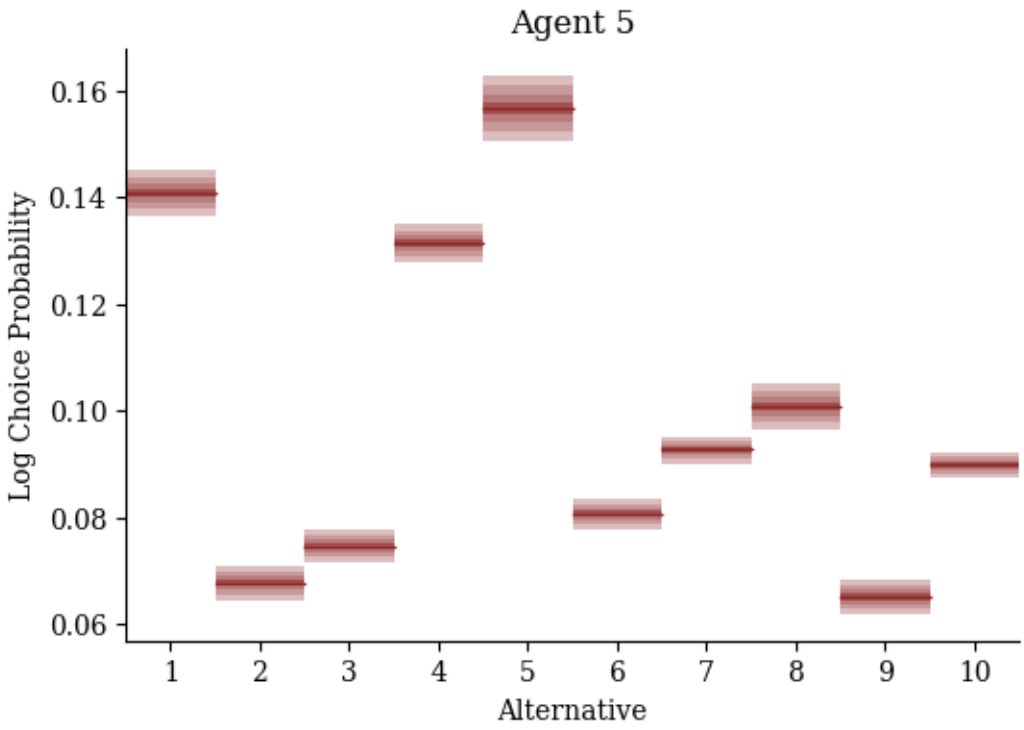

We can use these more precise inferences to make predictions that generalize to any choice subset.

```
names = [ f'log_prob_pred[{j}]'
          for j in joint_data['pred_alts'] ]

putil.plot_disc_pushforward_quantiles(
  plot.gca(), joint_samples, names,
  xlabel="Alternative",
  xticklabels=joint_data['pred_alts'],
  ylabel='Log Choice Probability',
  title=f'Agent {joint_data['n_pred']}'
```

```
)

plot.show()
```




## 4 Conclusion

Discrete choice modeling is a general technique for analyzing decisions made by individual agents. In this chapter, we have laid a strong foundation, reviewing the basic construction of these models and investigating the details of independent Gumbel models. This initial investigation, however, scratches only the surface of the possible applications of this technique.

The next three chapters diver further into discrete choice modeling. Next, **Part II** presents a few probabilistic utility models that couple the utilities together, allowing for richer substitution patterns. Then, **Part III** considers a general technique for elaborating an initial probabilistic utility model, and its implied substitution patterns. Finally, **Part IV** discusses techniques that don't model baseline utilities directly, but rather derive them from functional models.

# Appendix: Onerous Calculations

To keep the main text from being overwhelmed by mathematical derivations, I have sequestered 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

$$\begin{aligned}\Pi_{\text{Gumbel}}(x \mid \mu, \sigma) &= \int_{-\infty}^{x} \mathrm{d}x' \frac{1}{\sigma} \exp\left(-\frac{x'-\mu}{\sigma} - e^{-\frac{x'-\mu}{\sigma}}\right), \\ &= \int_{-\infty}^{x} \mathrm{d}x' \frac{1}{\sigma} e^{-\frac{x'-\mu}{\sigma}} \exp\left(-e^{-\frac{x'-\mu}{\sigma}}\right).\end{aligned}$$

At this point, 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 isolating the corresponding terms as much as possible gives

$$\begin{aligned}
\frac{\partial \Pi}{\partial u_{nj}}(u_{nj}, \ldots, u_{nj}) &= \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj} - \mu_{nj'}}{\sigma_n}}\right) e^{-\frac{u_{nj} - \mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} \\
&= \exp\left(-\sum_{j'=1}^{J} e^{-\frac{u_{nj}}{\sigma_n}} e^{\frac{\mu_{nj'}}{\sigma_n}}\right) e^{-\frac{u_{nj}}{\sigma_n}} e^{\frac{\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} \\
&= e^{\frac{\mu_{nj}}{\sigma_n}} \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}} \sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}\right).
\end{aligned}$$

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

$$\begin{aligned}
q_{nj} &= e^{\frac{\mu_{nj}}{\sigma_n}} \int_{-\infty}^{+\infty} \mathrm{d}u_{nj}\, \frac{1}{\sigma_n} e^{-\frac{u_{nj}}{\sigma_n}} \exp\left(-e^{-\frac{u_{nj}}{\sigma_n}}\, C\right) \\
&= e^{\frac{\mu_{nj}}{\sigma_n}}\, \frac{1}{C} \int_{-\infty}^{0} \mathrm{d}z\, \exp(z) \\
&= e^{\frac{\mu_{nj}}{\sigma_n}}\, \frac{1}{C} \Big[\exp(0) - \exp(-\infty)\Big] \\
&= e^{\frac{\mu_{nj}}{\sigma_n}}\, \frac{1}{C} \\
&= \frac{e^{\frac{\mu_{nj}}{\sigma_n}}}{\sum_{j'=1}^{J} e^{\frac{\mu_{nj'}}{\sigma_n}}},
\end{aligned}$$

which is exactly the softmax function applied to the scaled baseline utilities,

$$(q_{n1}, \dots, q_{nJ}) = \mathrm{softmax}\left(\frac{\mu_{n1}}{\sigma_n}, \dots, \frac{\mu_{nJ}}{\sigma_n}\right).$$

## 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**, which includes not only the Gumbel family but also all other extreme value families as special cases. This comprehensive family is defined by the probability density functions

$$\text{GEV}(x \mid \mu, \sigma, \xi) = \frac{1}{\sigma} t(x)^{\xi+1} \exp\left(-t(x)\right),$$

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 the generalized model reduces to a Gumbel model, does the support extend out to the entire real line $(-\infty, +\infty)$.

Despite this 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 $\xi$, we have

$$\begin{aligned}
\Pi_{\text{GEV}}(x \mid \mu, \sigma, \xi) &= \int \mathrm{d}x' \frac{1}{\sigma} \, t(x')^{\xi+1} \, \exp\left(-t(x')\right) \\
&= \int \mathrm{d}z \, \exp\left(z\right) \\
&= \exp\left(-t(x)\right).
\end{aligned}$$

An independent generalized extreme value model for alternative utilities is then given by

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

or, equivalently,

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

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 among all generalized extreme value models. Because of these issues, other one-dimensional generalized extreme value models are rarely considered for discrete choice models.

## Acknowledgements

I thank Andrés Castro Araújo, jd, and Fred Feinberg for helpful comments.

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

Ben-Akiva, M., M. Bradley, T. Morikawa, J. Benjamin, T. Novak, H. Oppewal, and V. Rao. 1994. “Combining Revealed and Stated Preferences Data.” *Marketing Letters* 5 (4): 335–49.

Chapman, Chris, and Elea McDonnell Feit. 2015. *R for Marketing Research and Analytics*. Edited by 1. Use r! Springer Cham.

Conlon, Christopher, and Julie Holland Mortimer. 2021. “Empirical Properties of Diversion Ratios.” *The RAND Journal of Economics* 52 (4): 693–726.

de Carvalho, M., R. Huser, P. Naveau, and B. J. Reich. 2026. *Handbook of Statistics of Extremes*. Boca Raton, FL: Chapman & Hall/CRC.

Falmagne, J. C. 1978. “A Representation Theorem for Finite Random Scale Systems.” *Journal of Mathematical Psychology* 18 (1): 52–72.

Feit, Eleanor McDonnell, Mark A. Beltramo, and Fred M. Feinberg. 2010. “Reality Check: Combining Choice Experiments with Market Data to Estimate the Importance of Product Attributes.” *Management Science* 56 (5): 785–800.

Luce, R. Duncan. 1979. *Individual Choice Behavior: A Theoretical Analysis*. Westport, Conn: Greenwood Press.

Manski, Charles F. 1977. “The Structure of Random Utility Models.” *Theory and Decision* 8 (3): 229–54.

McFadden, Daniel. 1986. “The Choice Theory Approach to Market Research.” *Marketing Science* 5 (4): 275–97.

Natan, Olivia R. 2025. “Choice Frictions in Large Assortments.” *Marketing Science* 44 (3): 593–625.

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.

Tversky, Amos. 1972. “Elimination by Aspects: A Theory of Choice.” *Psychological Review* 79 (4): 281–99.

## License

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

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

The text and figures in this 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

```
from watermark import watermark
print(watermark())
```

```
Last updated: 2026-11-09T19:01:14.610490-05:00

Python implementation: CPython
Python version       : 3.12.10
IPython version      : 9.6.0

Compiler    : Clang 13.0.0 (clang-1300.0.29.30)
OS          : Darwin
Release     : 24.6.0
Machine     : x86_64
Processor   : i386
CPU cores   : 16
Architecture: 64bit
```

```
print(watermark(packages="matplotlib,numpy,stan,scipy"))
```

```
matplotlib: 3.10.7
numpy     : 2.3.4
stan      : 3.10.0
scipy     : 1.16.3
```

**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, 2);
  target += lognormal_lpdf(tau_free | 0, 1);
```

**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);
```