Topic 6:
Product attributes and choice-based conjoint

Takeaki Sunada[1]

[1]Simon Business School
University of Rochester

Introduction to product attributes

$$Pr(y = j \mid P) = \frac{\exp(\beta_0^j + \beta_1 P^j)}{1 + \sum_{j'=1}^{J} \exp(\beta_0^{j'} + \beta_1 P^{j'})},$$

- β_0^j in logit models represents each consumer's valuation of product j - for each j in our choice data, we can estimate β_0^j, with which we can predict demand for j.
- However, this means we need to *observe j in the choice data* to predict demand for j.
- Observing j in choice data means that product is already in the market, and hence what we are studying is always pricing *adjustment*.

Introduction to product attributes

- If we aim to set a price for a brand new product, we don't know β_0^j of that product, and hence logit models are useless.
- This is quite inconvenient - optimal pricing is particularly important at product launch.
- Any way to get around this issue? Specifically, can we predict β_0^j for a new product from consumer's choices across *other* j's already available in the market?

Introduction to product attributes

- For all product j' already available, we can estimate its $\beta_0^{j'}$ from choice data. If our new product j is similar to one of them, we may expect that β_0^j is similar to $\beta_0^{j'}$ of that product.
- To formalize this idea, we need to first define a metric of "how similar those products are".
- One possible way is to look at the proximity of product "attributes" - any observable characteristics, such as design, functionality and ingredients.
- Two products that share similar attributes should share a similar β_0^j. Consider an example of smartwatch (finally we are leaving the soft drink market).

Introduction to product attributes

- Suppose that a smartwatch can be characterized by three main attributes - "battery life (days)", "memory size (storage space, GB)" and "android or iOS (binary)" (three for now to keep our example simple).
- Many products with various level of those attributes are already out in this market.

product	battery	memory	android	price
1	2	1	0	199
2	0.5	4	1	149
3	1	4	1	249
4	2	2	0	149
5	0.5	1	1	199
6	4	1	0	199

Choice data with product attributes

id	scenario	choice	const.1	battery.1	memory.1	android.1	price.1	const.2	battery.2	memory.2	android
1	1	3	1	2	1	0	199	1	0.5	4	
1	2	0	1	2	2	0	149	1	0.5	1	
1	3	1	1	4	8	1	149	1	0.5	4	
1	4	3	1	4	8	0	199	1	0.5	4	
1	5	3	1	0.5	4	1	149	1	2	2	
1	6	0	1	2	8	0	99	1	0.5	2	
1	7	0	1	0.5	2	0	149	1	2	1	
1	8	0	1	0.5	2	0	149	1	1	4	

- We have choice data, now with attributes of each product.

Logit models with product attributes

- Let's apply logit framework to this environment.

$$Pr(y = j \mid P) = \frac{\exp(\beta_0^j + \beta_1 P^j)}{1 + \sum_{j'=1}^{J} \exp(\beta_0^{j'} + \beta_1 P^{j'})},$$

- Instead of estimating β_0^j product-by-product, we assume that β_0^j is determined by a collection of product attributes.

$$\beta_0^j = \beta_{00} + \beta_{0,ba} battery^j + \beta_{0,me} memory^j + \beta_{0,an} android^j.$$

Logit models with product attributes

$$\beta_0^j = \beta_{00} + \beta_{0,ba} battery^j + \beta_{0,me} memory^j + \beta_{0,an} android^j.$$

- Originally, we directly estimated β_0^j and didn't care *why* each j receives the β_0^j it receives.
- Now we impose a reason - β_0^j differs across products because attributes differ. *Consumers value products because they value attributes*. Model parameters, $\{\beta_{00}, \beta_{0,ba}, \beta_{0,me}, \beta_{0,an}\}$ represent consumer preference for each attribute.
- When we estimate this model, what we estimate now is $\{\beta_{00}, \beta_{0,ba}, \beta_{0,me}, \beta_{0,an}\}$ instead of β_0^j.

Logit models with product attributes

$$\beta_0^j = \beta_{00} + \beta_{0,ba}\,battery^j + \beta_{0,me}\,memory^j + \beta_{0,an}\,android^j.$$

- Suppose that we estimated this model with products (and attributes) available in the market.
- If we launch a product with a particular battery, memory and OS, we can predict β_0^j for that product using this relationship - we don't need observation of choices involving that product.

Imposing a model (assumptions) to complement missing data

- Recall that our models are a collection of assumptions to complement missing data.
- Originally, j is observed in the data - in that sense, there is no data incompleteness. Hence we just estimated β_0^j from the frequency that j gets selected, and we don't ask *why* it gets selected.
- Now we don't see j in the data. Data is incomplete. Hence we need to impose a model to complement a missing data. "Consumers receive values from attributes" is the extra assumption we add.
- We then predict β_0^j for the product we don't see in the data because we know how consumers value attributes.

Implementation in R

- Let's implement this model in R. Unlike the case of directly estimating β_0^j, we need to (1) change some syntax of gmnl function, and (2) configure data set differently.
- Earlier in gmnl, we simply ran "choice $\sim$ price" and it provided β_0^j for each j and β_1. However we don't want β_0^j reported this way anymore. Hence we need to change the syntax.

```
#Run MLE.
mle= gmnl(choice ~  price, data = mlogitdata)
```

Coding gmnl

- With the current specification, the overall choice probability looks as follows.

$$Pr(y = j \mid P) = \frac{\exp(\beta_{00} + \beta_{0,ba} battery^j + \beta_{0,me} memory^j + \beta_{0,an} android^j + \beta_1 P^j)}{1 + \sum_{j'=1}^{J} \exp(\beta_{00} + \beta_{0,ba} battery^{j'} + \beta_{0,me} memory^{j'} + \beta_{0,an} android^{j'} + \beta_1 P^{j'})},$$

- This looks like that there's no β_0^j anymore, but we have a bunch of j-specific attributes *including* P^j, and an intercept β_{00} *that is common across all products*.
- Hence in the syntax, we eliminate β_0^j, include a bunch of attributes just like P^j, and include an intercept β_{00}.

Coding gmnl

```
#Run MLE.
mle= gmnl(choice ~  -1+const+battery+memory+android+price, data = mlogitdata)
```

- "-1" at the beginning eliminates product-specific β_0^j.
- We then add all the product attributes in the same way as the price.
- "Constant" is a variable that allows us to estimate β_{00}.

Coding gmnl

$$\beta_0^j = \beta_{00} + \beta_{0,ba} battery^j + \beta_{0,me} memory^j + \beta_{0,an} android^j.$$

- β_{00} roughly corresponds to the constant term of the regression. In the context of running a regression, constant term usually auto-populates and hence we don't have to explicitly include it.
- However, with gmnl, a constant term *that is common across all products* won't auto-populate. What auto-populates is a constant term *specific to each product*. Hence, to estimate β_{00}, we need to manually include the constant term for estimation.

Editing data set

id	scenario	choice	const.1	battery.1	memory.1	android.1	price.1	const.2	battery.2	memory.2	android
1	1	3	1	2	1	0	199	1	0.5	4	
1	2	0	1	2	2	0	149	1	0.5	1	
1	3	1	1	4	8	1	149	1	0.5	4	
1	4	3	1	4	8	0	199	1	0.5	4	
1	5	3	1	0.5	4	1	149	1	2	2	
1	6	0	1	2	8	0	99	1	0.5	2	
1	7	0	1	0.5	2	0	149	1	2	1	
1	8	0	1	0.5	2	0	149	1	1	4	

- "const.(product number)" is always one, except for the last alternative (chopped off to the right), which corresponds to the zero-option (not buying any).
- Otherwise, we have "battery", "memory", "android" and "price" for each product. Again, for zero-option, all of them are zero.

Editing data set

```
#Convert the data to mlogit form. This time assign all columns that
#contain all product attributes.
mlogitdata=mlogit.data(data,id="id",varying=4:23,choice="choice",shape="wide")
```

- We convert this original data into mlogit form. This time we assign all the product attributes (including "const") as "varying".
- Then we run MLE using gmnl.

Estimation results

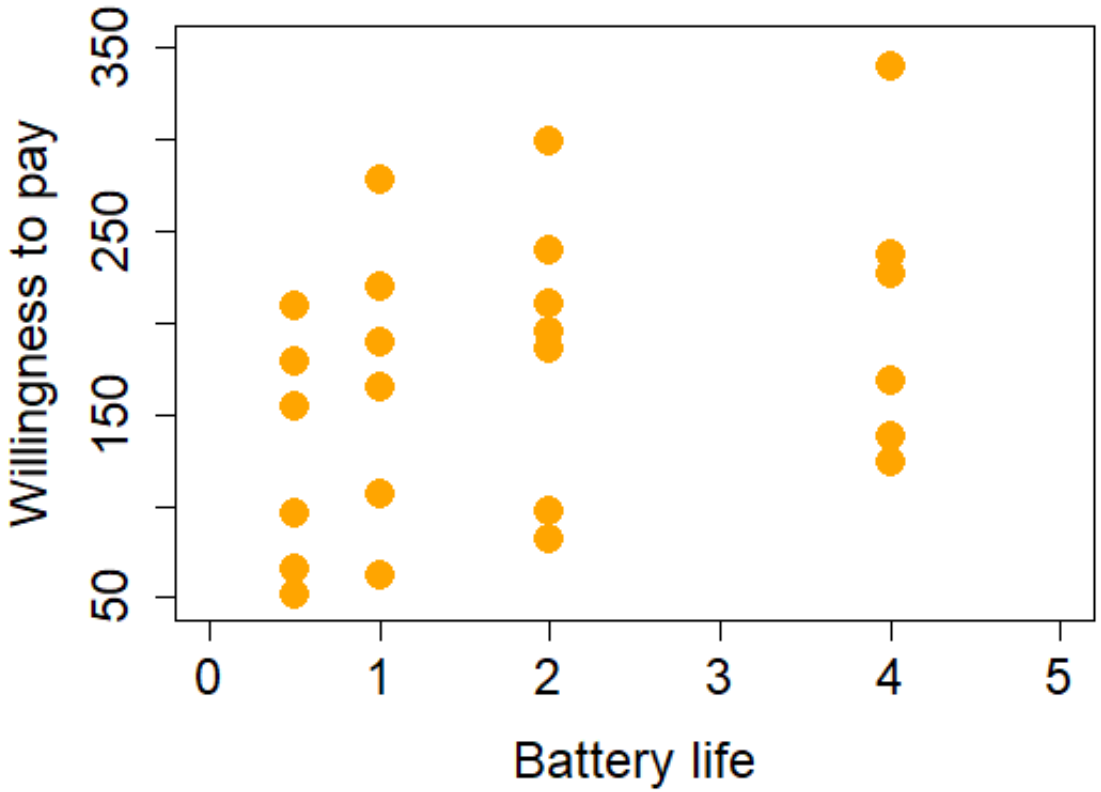
```
Coefficients:
          Estimate Std. Error  z-value  Pr(>|z|)
const    2.2288393  0.2639409   8.4445 < 2.2e-16 ***
battery  0.3294069  0.0359922   9.1522 < 2.2e-16 ***
memory   0.2361343  0.0220826  10.6933 < 2.2e-16 ***
android -1.8104201  0.1314951 -13.7680 < 2.2e-16 ***
price   -0.0159719  0.0013286 -12.0213 < 2.2e-16 ***
```

- gmnl provides parameter estimates, which we can interpret in the same way as in the regression.
- For example, $\beta_{0,ba} = 0.33$ implies that one extra day of battery life increases β_0^j by 0.33 (note that β_0^j is not measured in dollars).
- All the parameters have expected sign - the sniff test passed.

Demand prediction

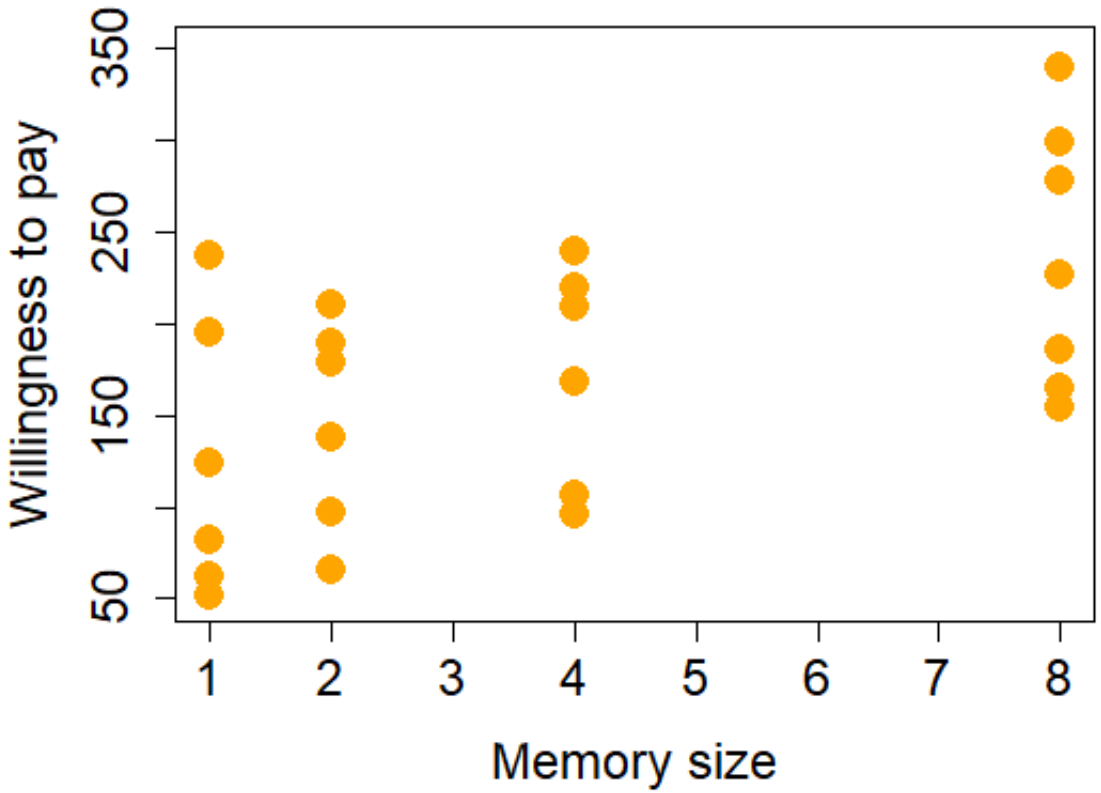
- Let's predict demand using the estimated parameters.
- First, let's calculate WTP for products used in the estimation. If we plug in the attributes of the products in the data, we can get β_0^j. As we also know β_1, we can calculate the willingness to pay for the product.

Demand prediction



- Consumers are willing to pay more for products with longer battery life.

Demand prediction



- Consumers are willing to pay more for products with larger memory.
- Note that none of the products currently out there have 6 GB of memory. With the attribute-based model, we can still predict consumer WTP for 6GB of memory (which was not possible before).

Predicting consumer WTP for a new product

- What if we launch a new product, with 4 days of battery life, 6 GB of memory and operating on Android OS?
- We can calculate β_0^j for our new product following the formula for β_0^j.

$$\beta_0^j = 2.23 + 0.33 \times \mathit{battery}^j + 0.24 \times \mathit{memory}^j - 1.81 \times \mathit{android}^j.$$

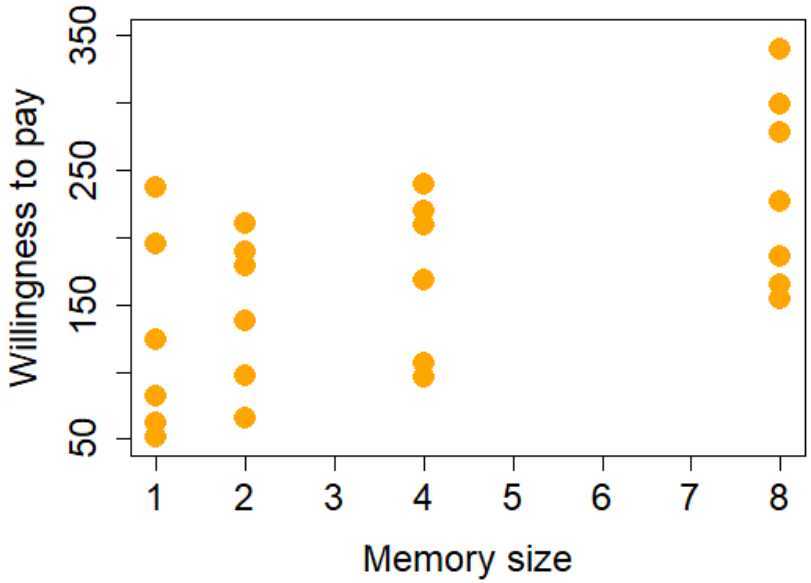
- Plugging in the values, we have $\beta_0^j = 3.15$. Hence consumer WTP for this product is 197.4 dollars.
- As we know β_0^j for the new product, we can predict choice probability, aggregate demand and solve the profit maximization problem. I will skip this part as we are already familiar with the process.

One caveat

$$\beta_0^j = 2.23 + 0.33 \times battery^j + 0.24 \times memory^j - 1.81 \times android^j.$$

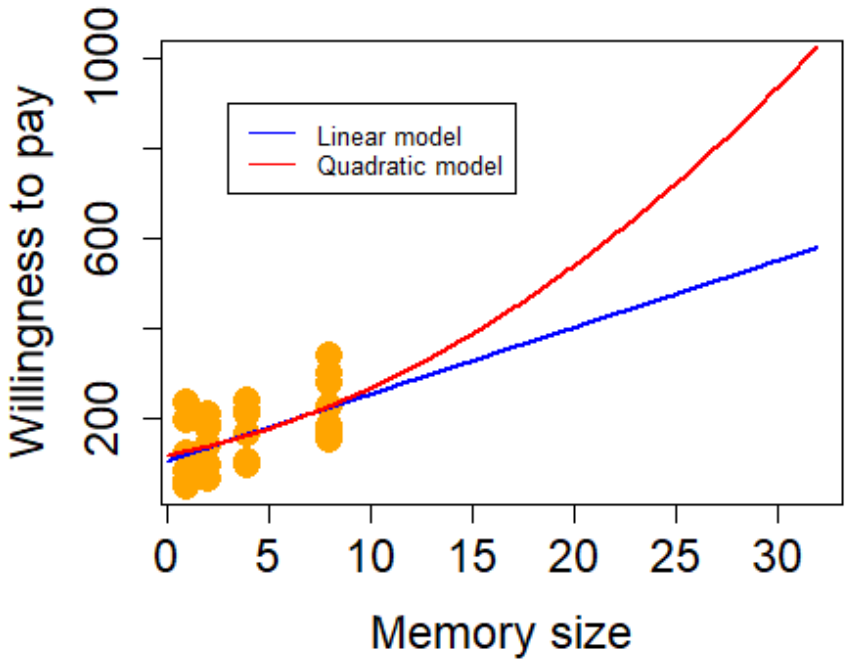
- Recall that we fit the choice probability of existing products to get this relationship - this is an interpolating exercise.
- If our new product is completely different from the ones in the data (available product), we likely cannot get estimate of WTP for our product.
- One easy example is that our new product contains a brand new attribute that no other product carries.
- The other, a bit harder to notice but equally important one, is that our new product has the same attribute, but at a completely different magnitude.

One caveat



- This model is estimated with smartwatches with up to 8GB of memory.
- What if we develop a smartwatch with 32GB of memory? Following the definition of β_0^j, we do get a number from our model. However, it's a big extrapolation beyond the sample range. The predicted number is likely way off the reality.

One caveat



- This model is estimated with smartwatches with up to 8GB of memory.
- What if we develop a smartwatch with 32GB of memory? Following the definition of β_0^j, we do get a number from our model. However, it's a big extrapolation beyond the sample range. The predicted number is likely way off the reality.

Consumer WTP for each attribute

- As we know, β_0^j is a measure of consumer's valuation for product j and we can obtain WTP by dividing it with $-\beta_1$.
- We just asserted that β_0^j is described as the sum of each consumer's valuation for each attribute.

$$\beta_0^j = \beta_{00} + \beta_{0,ba} battery^j + \beta_{0,me} memory^j + \beta_{0,an} android^j.$$

- Hence, for example, $\beta_{0,ba}$ is a measure of consumer's valuation of one extra day of battery life. If we divide it by $-\beta_1$, we can calculate the consumer's WTP for an extra day of battery life.

Consumer WTP for each attribute

```
Coefficients:
             Estimate Std. Error  z-value  Pr(>|z|)
const      2.2288393  0.2639409   8.4445 < 2.2e-16 ***
battery    0.3294069  0.0359922   9.1522 < 2.2e-16 ***
memory     0.2361343  0.0220826  10.6933 < 2.2e-16 ***
android   -1.8104201  0.1314951 -13.7680 < 2.2e-16 ***
price     -0.0159719  0.0013286 -12.0213 < 2.2e-16 ***
```

- In the smartwatch example, we predict that consumer values:
  - One extra day of battery life by 20.6 dollars
  - 1GB of extra memory by 14.8 dollars
  - Operating on iOS by 113.35 dollars
- This not only helps our optimal pricing, but also product development.

Extensions

- Any extensions of the logit model we covered earlier carry over to this model with product attributes.
- We can cluster consumers with kmeans function and estimate different attribute coefficients ($\beta_{00k}, \beta_{0,ba,k}$, etc.) across different clustered segments.
- We can define attribute coefficients as a function of demographics, just like regression-in-logit (include interaction terms of attributes with demographics in gmnl).
- We can include latent types and estimate different attribute coefficients, as well as proportion of each type of consumers, using gmnl (add |1|0|0|1 after we list all the attributes and assign "lc" as a model).

Extensions

- In practice, we may have many product attributes. Just like a regression case, we can include them as necessary. The specification choice should be done using our usual metric (BIC, standard error of the estimates, etc).
- Note that we may not want a product fixed effect in this specification - product FE is j-specific by construction. Because we don't know the value of product FE for the brand new product, we cannot predict β_0^j.

Attribute-based models and choice-based conjoint

- So far: if we assume that consumers value attributes, we can predict demand for a new product if it shares the same attributes with the existing products.
- By now, some of you must have noted that the concepts we discussed here overlaps with those of conjoint analysis. Because many of you have taken a course about conjoint, let's try to find an association between the two approaches.
- Indeed, our methodologies can be applied to a conjoint data. So if you know how to best collect conjoint data, you can put it together with our analytic models to achieve an even better outcome.

Rating-based conjoint

- Conjoint analysis is a data-collection method to elicit consumer preferences for each product attribute.
- We conduct survey, where we list multiple hypothetical products with different levels of attributes. We ask consumers to rate each option.
- We choose the attribute combinations, so that no product is dominant (i.e. highest quality and lowest price) - consumers need to make a trade-off, as they always do in real product selection.
- We then obtain data of consumer ratings over products with different levels of attributes and hence can predict WTP. This data-collection method is called "Rating-based conjoint".

Choice-based conjoint

- "Choice-based conjoint" is another conjoint approach to elicit information about consumer preferences.
- Just like a rating-based conjoint, we run a survey and list multiple products with different levels of attributes.
- However, instead of asking consumers to rate each option, we ask *which one they buy (or none)*.
- Hence the data from choice-based conjoint is the attributes and price of each hypothetical product, and each consumer's choice (and not the rating).
- The conjoint choice data have the exact same structure as the real choice data - hence we can apply our framework.

Conjoint choice data

id	scenario	choice	const.1	battery.1	memory.1	android.1	price.1	const.2	battery.2	memory.2	android
1	1	3	1	2	1	0	199	1	0.5	4	
1	2	0	1	2	2	0	149	1	0.5	1	
1	3	1	1	4	8	1	149	1	0.5	4	
1	4	3	1	4	8	0	199	1	0.5	4	
1	5	3	1	0.5	4	1	149	1	2	2	
1	6	0	1	2	8	0	99	1	0.5	2	
1	7	0	1	0.5	2	0	149	1	2	1	
1	8	0	1	0.5	2	0	149	1	1	4	

- Indeed, the data set I used earlier is not real, but is obtained from a conjoint survey. Note that at each row, available product attributes are completely different. This shouldn't be the case in real data ("product 1" should always have the same attributes).
- "id" is each conjoint survey recipient, and "scenario" is each conjoint scenario given to her. At each scenario, she chooses one from three possible options (or zero option of not buying). The choice is recorded as "choice".

Conjoint choice data

- We can use this data set in the same way as a panel data set of real consumer choices - "id" is each consumer's id, and "scenario" corresponds to that consumer's each shopping trip.
- We estimate β_0^j as a function of attributes and β_1, and all the following analyses work in the same way as before.
- If we have each survey recipient's demographics, we can estimate demographic-specific coefficients. To the extent that each consumer provides her choice in multiple hypothetical scenario, we observe history of consumer purchases. Hence we can try latent segment models.

Conjoint data or real data?

- Remember our objective: we want to develop a new product and we need to predict its β_0^j to evaluate its profitability.
- We saw two approaches: either using real choice data of existing product, or to run a hypothetical conjoint survey to create data. Either way, by applying the attribute-based model, we can predict β_0^j.
- Then an obvious question - when should we use real choice data vs conjoint choice data?

Advantages and disadvantages of real data

- Real data, by definition, records consumers' *real* choice. This is a big advantage over *stated* choice. Predictions are generally *way* better.
- With real data, we may only have sparse observations of attributes (unable to include flexible functional form). Existing products may not be located close enough to your planned product (extrapolation issue).
- As a rule of thumb, if the product you plan to launch is too far from any of the existing product, real choice data is not useful.

Advantages and disadvantages of conjoint data

- Conjoint data can assign any hypothetical product attributes and elicit consumer preference about them. Location of existing products doesn't matter.
- Conjoint data record *stated preference*. Consumers often behave differently from what they say they would do. So predictions based on conjoint data are usually much less accurate.
- Conjoint survey usually consists of up to 10-15 scenarios. Such panel size can be longer or shorter than real data - implications to estimating latent segment models.

Advanced: Comparing choice-based vs rating-based conjoint

- In a rating-based conjoint, consumers report ratings. We use regression models to derive consumer WTP.
- In a choice-based conjoint, consumers report choice. We use choice models to derive consumer WTP.
- When should we use choice vs rating conjoint?

Advanced: Comparing choice-based vs rating-based conjoint

- Because we know how to deal with choice data, a choice-based conjoint is a preferred option compared to a rating-based conjoint. The advantage of rating-based conjoint is that we only need a regression to study conjoint data, but that's not an advantage for us anymore (we know how to study choice data!).
- To make the data fit in the regression framework, rating-based conjoint needs to impose two more assumptions than choice-based conjoint.

Advanced: Comparing choice-based vs rating-based conjoint

- In a rating-based conjoint, respondents need to report *rating* of *all* the options in the survey. Hence they need to (1) express *how much* they value products by a number (rating), and (2) know their values for all the options, *including the less preferred ones*.
- (1) says if a consumer chooses find brand A milk best, she needs to be able to articulate *how much* she likes it better than the other brand milk. Can you articulate this?
- (2) says if she chooses brand A, she still needs to provide rank ordering between brand B and C. Can you compare *correctly* the two options you don't like the best?

Comparing choice-based vs rating-based conjoint

- In choice-based conjoint, respondents only report the option they choose. They don't have to articulate how much they like the selected option. They also don't need to report the relative preference between non-selected alternatives.
- Hence respondents need to provide less information to us. In particular, the information they need to provide is the one that they process in their daily life - they don't have to think through in order to fill out the survey, and hence less errors.
- Note that we still assume that respondents' stated preference is the respondents' true preference (an assumption also imposed in rating-based conjoint).

Comparing choice-based vs rating-based conjoint

- We supplement the missing pieces of information with choice models.
- With choice data, we recover β_0^j for each product from "how often each j is selected in the survey (across consumers)". As we can pool information across consumers, we don't need to know everything about each individual consumer.

Summary: product attributes and choice-based conjoint

- In order to predict demand for new products, we need to estimate β_0^j for that product *in the absence of that product*. To this end, we include product attributes in our logit framework.
- In order to estimate the model, we may use data of existing product with various attributes, or we conduct a choice-based conjoint survey. There are pros and cons: real data only provide limited coverage on the attribute dimensions. Conjoint data cannot provide long panel and stated preference often differs from actual preference.
- If we follow the conjoint approach, it is a preferred alternative than the rating-based approach, as we utilize our modeling assumptions better to reduce informational requirement.