choice_based_conjoint.nb.pdf

- Show All Code
- Hide All Code
-
- Download Rmd

R Notebook

Packages

To perform the analysis, you need to have the following packages installed:

1. ggplot2
2. ggthemes
3. xtable
4. knitr
5. mlogit
6. caret
7. e1071

Choice-based conjoint analysis

The purpose of choice-based conjoint is to estimate how different levels of different attributes (e.g., for tablets – levels of price, brand names, screen size, etc.) affect consumer's demand for the product. It is usually performed by repeatedly asking an individual to select one alternative from a small set – where characteristics of available options are varied from one choice set to another.

Once the choice model based on such data has been estimated, we can use the parameter estimates to assess relative importance of different attributes of a product.

Data

We will work with *conjoint_tablet_data.csv* file. The file contains data on choices made by 137 subjects. Each subject evaluated 15 choice sets. Thus, the file contains data on 137 * 15 = 2055 choice sets. Each choice set had three alternatives.

A subject’s task was to choose one alternative from a choice set. Each alternative was described using the following attributes: brand name, screen size, size of hard drive, RAM, battery life, and price.

Column 1 (consumer id) identifies each of the 137 subjects. Column 2 (choice set id) identifies each of 2055 choice sets. Column 3 (alternative id in set) identifies the three alternatives in a choice set. Column 4 identifies the id of the alternative chosen from the choice set. The remaining columns contain attributes for each alternative.

```
library("xtable") # processing of regression output
library("knitr") # used for report compilation and table display
library("ggplot2") # very popular plotting library ggplot2
library("ggthemes") # themes for ggplot2
suppressMessages(library("mlogit")) # multinomial logit
library("caret") # confusion matrix
# loading data
data <- read.csv(file = "conjoint_tablet_data.csv")
# defining reference levels - used when estimating model
data$Brand <- relevel(data$Brand, ref = 'Nexus')
data$Size <- relevel(data$Size, ref = 'sz7inch')
data$Storage <- relevel(data$Storage, ref = 'st16gb')
data$Ram <- relevel(data$Ram, ref = 'r1gb')
data$Battery <- relevel(data$Battery, ref = 'b7h')
#data$Price <- relevel(data$Price, ref = 'p169')
# we treat price as continuous (even though it was evaluated at discrete levels when the data was collected)
# we first do preprocessig to remove character 'p' from price variable, and then turn the variable into numeric class
data$Price <- as.numeric(gsub("p","",data$Price))
kable(head(data,6))
```

ConsumerId	ChoiceSetId	AlternativeIdInSet	Choice	Brand	Size	Storage	Ram	Battery	Price
1	1	1	1	iPad	sz7inch	st32gb	r4gb	b7h	499
1	1	2	0	Surface	sz10inch	st64gb	r2gb	b9h	399
1	1	3	0	Kindle	sz9inch	st16gb	r2gb	b8h	499
1	2	1	1	iPad	sz8inch	st32gb	r1gb	b8h	399
1	2	2	0	Surface	sz10inch	st128gb	r4gb	b7h	299
1	2	3	0	Nexus	sz7inch	st64gb	r1gb	b9h	199

In the table above, you can see two choice sets displayed for consumer #1. Each row corresponds to an alternative and characteristics that describe it. Each choice set has three alternatives and the individual selected one alternative in each set – which happened to be alternative #1 in both cases. The ordering of alternatives within a choice set has no special meaning.

Multinomial conjoint model estimation

We use mlogit package to train a conjoint multinomial choice model. First, we use provided function *mlogit.data()* to create a specially formatted data object that will be used in estimation.

```
mdata <- mlogit.data(data=data,
                     choice='Choice', # variable that contains choice
                     shape='long', # tells mlogit how data is structured (every row is alternative)
```

```
                      varying=5:10, # columns that contain variables that vary across alternatives
                      alt.levels = paste('alternative',1:3), # levels of the alternatives
                      id.var='ConsumerId') # consumer id
kable(head(mdata,6))
```

Property	ConsumerId	ChoiceSetId	AlternativeIdInSet	Choice	Brand	Size	Storage	Ram	Battery	Price
1.alternative 1	1	1	1	TRUE	iPad	sz7inch	st32gb	r4gb	b7h	499
1.alternative 2	1	1	2	FALSE	Surface	sz10inch	st64gb	r2gb	b9h	399
1.alternative 3	1	1	3	FALSE	Kindle	sz9inch	st16gb	r2gb	b8h	499
2.alternative 1	1	2	1	TRUE	iPad	sz8inch	st32gb	r1gb	b8h	399
2.alternative 2	1	2	2	FALSE	Surface	sz10inch	st128gb	r4gb	b7h	299
2.alternative 3	1	2	3	FALSE	Nexus	sz7inch	st64gb	r1gb	b9h	199

When we run the model, it selects the *reference* level for each *discrete* attribute. The utility of the reference level is normalized to zero. We specified a reference level for each discrete attribute at the data-loading stage. These reference levels are Nexus, 7" screen, 16GB HD, 1GB RAM, 7-hour battery. We treat price as a continuous variable, so we do not need to specify a reference level.

The model assumes the utility of alternative \(j\) without an error term is expressed as follows

$$\begin{aligned}
V_j = & \beta_{11} 1\,[\text{Brand=Galaxy}] + \beta_{12} 1\,[\text{Brand=iPad}] + \beta_{13} 1\,[\text{Brand=Kindle}] + \beta_{14} 1\,[\text{Brand=Surface}] + \\
& \beta_{21} 1\,[\text{Screen=10inch}] + \beta_{22} 1\,[\text{Screen=9inch}] + \beta_{23} 1\,[\text{Screen=8inch}] + \\
& \beta_{31} 1\,[\text{Storage=128gb}] + \beta_{32} 1\,[\text{Storage=64gb}] + \beta_{33} 1\,[\text{Storage=32gb}] + \\
& \beta_{41} 1\,[\text{RAM=4gb}] + \beta_{42} 1\,[\text{RAM=2gb}] + \\
& \beta_{51} 1\,[\text{Battery=9h}] + \beta_{52} 1\,[\text{Battery=8h}] + \\
& \beta_6 \text{Price}
\end{aligned}$$

where

$$U_j = V_j + \text{error}$$

That is, there are 15 parameters \(\beta\) to estimate.

Assuming independent extreme value error distribution, consumer chooses alternative \(j\) from the choice set of three alternatives with probability

$$p_j = \frac{\exp(V_j)}{\exp(V_1) + \exp(V_2) + \exp(V_3)}, \quad j \in \{1, 2, 3\}$$

Clearly,

$$p_1 + p_2 + p_3 = 1$$

And now we actually estimate the model.

```
set.seed(999) # remember to set the random seed to ensure replicability
model <- mlogit(Choice~0+Brand+Size+Storage+Ram+Battery+Price,data=mdata) # 0+ tells model to exclude intercept
#summary(model)
kable(xtable(summary(model)$CoefTable))
```

Property	Estimate	Std. Error	t-value	Pr(>|t|)
BrandGalaxy	0.3378857	0.0925056	3.652596	0.0002596
BrandiPad	0.9780287	0.0937336	10.434136	0
BrandKindle	0.2630105	0.0996254	2.639995	0.0082907
BrandSurface	0.1450365	0.0938521	1.545373	0.122256
Sizesz10inch	0.3240632	0.0841953	3.848949	0.0001186
Sizesz8inch	0.1890775	0.0829232	2.280151	0.0225987
Sizesz9inch	0.4355415	0.0808408	5.387644	0.0000001
Storagest128gb	0.5897703	0.0870533	6.774822	0
Storagest32gb	0.2168719	0.0829213	2.615395	0.0089124
Storagest64gb	0.5782183	0.0808259	7.153877	0
Ramr2gb	0.3189348	0.0672579	4.74197	0.0000021
Ramr4gb	0.6357438	0.0645225	9.853053	0
Batteryb8h	0.1299599	0.0651501	1.994777	0.0460672
Batteryb9h	0.1253824	0.0650588	1.927216	0.0539528
Price	-0.0050888	0.0002752	-18.488626	0

Meaning of parameters

After estimation, we obtain a coefficient estimate for each level (except the reference one) of every discrete attribute. Such a coefficient captures relative utility or *partworth* of the level of attribute compared to the reference. For example, in case of brand attribute, *BrandiPad* coefficient gives us an estimate of iPad's brand relative utility compared to Nexus (reference brand).

In case of the continuous price, we get a single coefficient, which captures how utility of the alternative changes when price goes up by one unit ($1), holding all other characteristics of the alternative fixed.

Prediction

We can also use the estimated parameters to predict the probabilities of the choice for different alternatives in the data. Here we print the prediction for the first five choice sets in the data.

```
head(predict(model,mdata),5)
```

```
     alternative 1 alternative 2 alternative 3
[1,]     0.3717263     0.4405521     0.1877216
[2,]     0.2367797     0.4718620     0.2913583
[3,]     0.4760867     0.2974319     0.2264814
[4,]     0.3730366     0.4456505     0.1813129
[5,]     0.3984632     0.1618560     0.4396807
```

And now we can measure the accuracy of prediction across all data.

```
predicted_alternative <- apply(predict(model,mdata),1,which.max)
selected_alternative <- data$AlternativeIdInSet[data$Choice>0]
confusionMatrix(predicted_alternative,selected_alternative,positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   1   2   3
         1 362 158 130
         2 164 449 149
         3 136 160 347

Overall Statistics

               Accuracy : 0.5635
                 95% CI : (0.5417, 0.5851)
    No Information Rate : 0.3732
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.343
 Mcnemar's Test P-Value : 0.8875

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.5468   0.5854   0.5543
Specificity            0.7933   0.7570   0.7929
Pos Pred Value         0.5569   0.5892   0.5397
Neg Pred Value         0.7865   0.7541   0.8024
Prevalence             0.3221   0.3732   0.3046
Detection Rate         0.1762   0.2185   0.1689
Detection Prevalence   0.3163   0.3708   0.3129
Balanced Accuracy      0.6700   0.6712   0.6736
```

Note that if the predictions were random, the accuracy would be 33.3% (for three alternatives). Our simple model is doing much better than
that – although it is not perfect.

Conjoint simulator

And now let us see how we can use model parameters to predict market shares under hypothetical market scenarios for an arbitrary set of products.

```
# function to predict market share for an arbitrary set of alternatives available in dataset d.
predict.share <- function(model, d) {
  temp <- model.matrix(update(model$formula, 0 ~ .), data = d)[,-1] # generate dummy matrix
  u <- temp%*%model$coef[colnames(temp)] # calculate utilities
  probs <- t(exp(u)/sum(exp(u))) # calculate probabilities
  colnames(probs) <- paste("alternative", colnames(probs))
  return(probs)
}
# hypothetical base market structure with 4 alternatives in the market
d.base <- data[c(44,34,33,40),c("Brand","Size","Storage","Ram", "Battery","Price")]
d.base <- cbind(d.base,as.vector(predict.share(model,d.base)))
colnames(d.base)[7] <- 'Predicted.Share'
rownames(d.base) <- c()
kable(d.base)
```

Brand	Size	Storage	Ram	Battery	Price	Predicted.Share
iPad	sz7inch	st64gb	r2gb	b8h	399	0.3423928
Galaxy	sz10inch	st32gb	r2gb	b7h	299	0.2540301
Surface	sz10inch	st64gb	r1gb	b7h	399	0.1313854
Kindle	sz7inch	st32gb	r1gb	b9h	169	0.2721917

```
# hypothetical market structure after Galaxy gets a RAM upgrade
d.new <- d.base
d.new[2,'Ram'] <- "r4gb"
d.new$Predicted.Share <- as.vector(predict.share(model,d.new))
kable(d.new)
```

Brand	Size	Storage	Ram	Battery	Price	Predicted.Share
iPad	sz7inch	st64gb	r2gb	b8h	399	0.3127768
Galaxy	sz10inch	st32gb	r4gb	b7h	299	0.3185544
Surface	sz10inch	st64gb	r1gb	b7h	399	0.1200209
Kindle	sz7inch	st32gb	r1gb	b9h	169	0.2486479

Willingness to pay

Very importantly, using parameter estimates, we can calculate how much a consumer would be willing to pay for the selected level of an attribute by dividing coefficient for that level by the coefficient for the price. In other words, we estimate what change in price would cause shift in utility equivalent to that due to change in the level of the attribute in question from the reference level.

For example, we see that an average consumer would be indifferent between getting a Galaxy vs. paying $125.8 more and getting an iPad. Phrasing this differently, an average consumer would be willing to pay up to $125.8 to get an iPad instead of a Nexus, holding all other characteristics fixed.

```
# brand equity - dollar value of an upgrade from Galaxy to iPad
-(coef(model)['BrandiPad']-coef(model)['BrandGalaxy']) / coef(model)['Price']
```

```
BrandiPad
 125.7944
```

```
# dollar value of an upgrade from 1gb to 4gb ram  (1gb is reference level, hence its coeff is 0)
-coef(model)['Ramr4gb'] / coef(model)['Price']
```

```
 Ramr4gb
124.9299
```

```
# dollar value of an upgrade from 7 inch to 9 inch screen (7 inch is reference level)
-coef(model)['Sizesz9inch'] / coef(model)['Price']
```

```
Sizesz9inch
    85.5882
```