conjoint analysis

Jin Kweon

3/5/2018

Group

Zotero: https://www.zotero.org/groups/2095516/conjoint_analysis

Github: https://github.com/pbstark/Conjoint

Reference

https://www.sawtoothsoftware.com/products/advanced-analytical-tools/cbc-lc#technical-paper

http://www.informit.com/articles/article.aspx?p=2350028

https://rpubs.com/haj3/conjoint

http://keii.ue.wroc.pl/pracownicy/tb/Bak_A_and_Bartlomowicz_T_Conjoint_analysis_method_and_its_implementation_in_conjoint_R_package.pdf

https://www.youtube.com/watch?v=PgFhhZhipVc

https://en.wikipedia.org/wiki/Theory_of_conjoint_measurement

Summary

Conjoint analysis is a popular marketing research technique that marketers use to determine what features a new product should have and how it should be priced.

These three steps–collecting trade-offs, estimating buyer value systems, and making choice predictions– form the basics of conjoint analysis.

Goal:

Our goal is to find why Conjoint analysis does not make sense in real world. Many companies including Sawtooth is trying to sell this nonsense software to public and companies (and they are actually believing this softwares make sense). But, as I can see many examples from Primer document professor Stark made, many rational customers can choose options that do not meet conjoint analysis (conjoint analysis axioms - single and double cancellations & sovability and archimedian axioms).

For example, let's say Apple has the patent for the new phones that has a new screen that can respond to users' behaviors/finger movements; however, Samsung copies that idea and sell the phones that have the technology. So, Apple tries to sue Samsung. In here, Apple argues to go to the mall and ask randomly chosen 150 (this number is arbitrary) "rational" people whether they are actually going to buy this phone (assume this phone is $100 more expense since it has this new thing) since this phone has the new technology. And, Apple collects many data and runs them on the conjoint software, and argues that Samsung owes 100 for each phone based on this conjoint analysis, which does not make sense.... Why??? because this is more complicated than that... As I the examples from Primer show, rational customers can choose other options that conjoint analysis give out... Also, when customers buy a phone, they do consider more than this single

new technology... So, in real world, there are many lawsuits going on whether conjoint analysis is useful or not.

So, anyway, our goal is to argue conjoint analysis is really not working properly... William is trying to fix the software that we believe working similiarly as what Sawtooth (the most famous company that is providing conjoint software) does...

So, I want to come up with more examples, and prove although axioms hold, rational customers can choose other options than what conjoint analysis software give... And, we are going to run these examples on "conjoint" library in R and William's software, and try to find out whether these results (based on conjoint) will have different answers with my examples (rational people's choice without using conjoint analysis but still satisfy axioms...) (then, it means they are counter examples, and we can say that conjoint analysis is not really a good way...)

Library

```
library(cjoint)
```

```
## Loading required package: sandwich
## Loading required package: lmtest
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package: ggplot2
## Loading required package: survey
## Warning: package 'survey' was built under R version 3.4.4
## Loading required package: grid
## Loading required package: Matrix
## Loading required package: survival
##
## Attaching package: 'survey'
## The following object is masked from 'package:graphics':
##
##     dotchart
## cjoint: AMCE Estimator for Conjoint Experiments
## Version: 2.0.6
## Authors: Soubhik Barari, Elissa Berwick, Jens Hainmueller, Daniel Hopkins, Sean Liu, Anton Strezhnev
```

```
library(bayesm)

library(conjoint)
```

```
##
## This is package 'modeest' written by P. PONCET.
## For a complete list of functions, use 'library(help = "modeest")' or 'help.start()'.
```

```r
library(faisalconjoint)

library(support.CEs)  # package for survey construction
```

```
## Loading required package: DoE.base
## Loading required package: conf.design
##
## Attaching package: 'DoE.base'
## The following objects are masked from 'package:stats':
##
##     aov, lm
## The following object is masked from 'package:graphics':
##
##     plot.design
## The following object is masked from 'package:base':
##
##     lengths
## Loading required package: MASS
## Loading required package: simex
## Loading required package: RCurl
## Loading required package: bitops
##
## Attaching package: 'RCurl'
## The following object is masked from 'package:lmtest':
##
##     reset
## Loading required package: XML
```

```r
library(R.utils) #source directory
```

```
## Loading required package: R.oo
## Loading required package: R.methodsS3
## R.methodsS3 v1.7.1 (2016-02-15) successfully loaded. See ?R.methodsS3 for help.
## R.oo v1.21.0 (2016-10-30) successfully loaded. See ?R.oo for help.
##
## Attaching package: 'R.oo'
## The following object is masked from 'package:RCurl':
##
##     clone
## The following objects are masked from 'package:methods':
##
##     getClasses, getMethods
## The following objects are masked from 'package:base':
##
##     attach, detach, gc, load, save
```

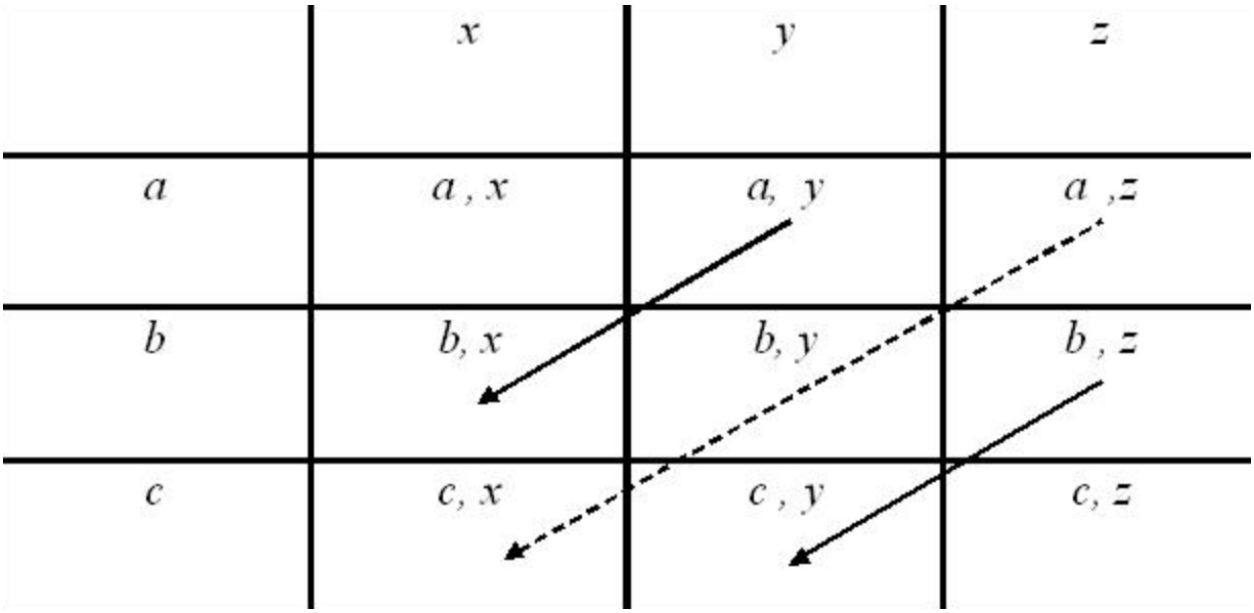

Figure 1: Double cancellation

```
## R.utils v2.6.0 (2017-11-04) successfully loaded. See ?R.utils for help.

##
## Attaching package: 'R.utils'

## The following object is masked from 'package:RCurl':
##
##     reset

## The following object is masked from 'package:lmtest':
##
##     reset

## The following object is masked from 'package:utils':
##
##     timestamp

## The following objects are masked from 'package:base':
##
##     cat, commandArgs, getOption, inherits, isOpen, parse, warnings
```

Building up the non-conjoint examples with conjoint axioms - “Phone”

Reference: https://en.wikipedia.org/wiki/Theory_of_conjoint_measurement

I am going to build up the codes with the graphical (matrix) representation, so we could always reproduce and generate as many examples as I can.

Our goal is to generate and reproduce examples that fail single cancellation (independent axiom) and double cancellation.

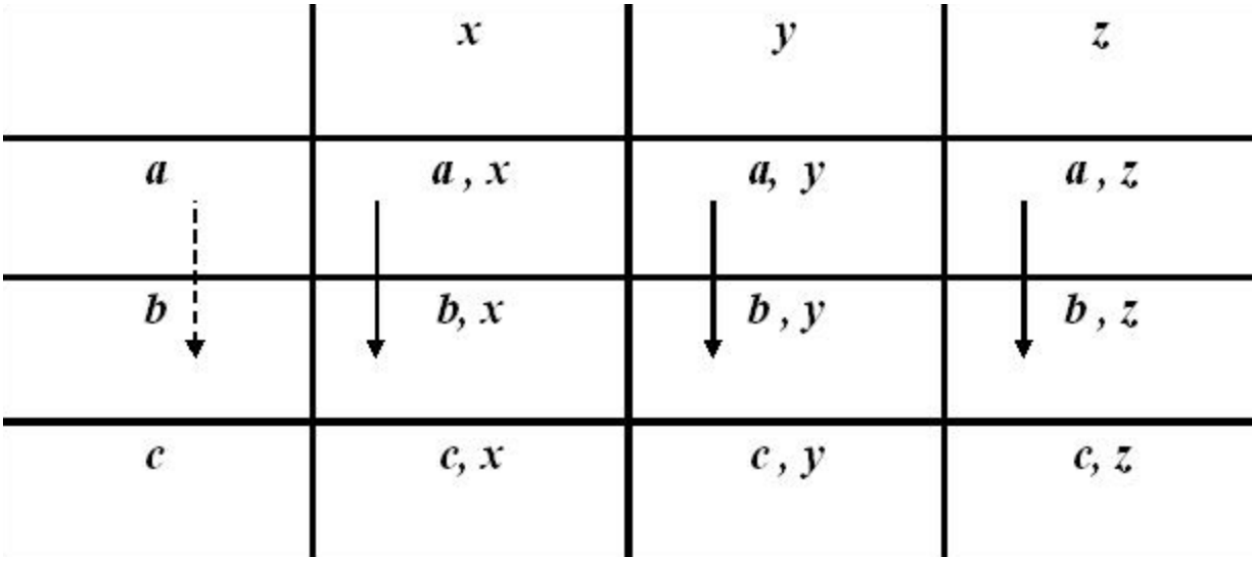

Figure 2: Single cancellation

A Luce - Tukey instance of double cancellation, in which the consequent inequality (broken line arrow) does not contradict the direction of both antecedent inequalities (solid line arrows), so supporting the axiom.

Graphical representation of the single cancellation axiom. It can be seen that a > b because (a, x) > (b, x), (a, y) > (b, y) and (a, z) > (b, z).

Try to build up 2 by 3 examples (2 attributes and 3 levels/attribute values)

```
#Test string random generator
# myFun <- function(n = 5000) {
#   a <- do.call(paste0, replicate(5, sample(letters, n, TRUE), FALSE))
#   paste0(a, sprintf("%04d", sample(999, n, TRUE)), sample(letters, n, TRUE))
# }

#First example
data <- matrix(0, 2, 3)

rownames(data) <- c("brand", "color")
data
```

```
##       [,1] [,2] [,3]
## brand    0    0    0
## color    0    0    0
```

```
data[1,] <- c("samsung", "nokia", "apple")
data[2,] <- c("yellow", "blue", "red")


#Second example
data2 <- matrix(0, 2, 3)
rownames(data2) <- c("sandwich type", "condiment")

data2[1, ] <- c("peanut butter", "turkey", "ham")
data2[2, ] <- c("mustard", "grape jelly", "cranberry sauce")
```

```r
conjoining <- function(matrix){
  print(paste("I am going to ask you to input two pairs of", rownames(matrix)[1], "and", rownames(matri
  print(paste("Here is the data you have"))
  print(matrix)


  #First comparison
  print(paste("Choose the pair you like better"))
  ff <- readline(prompt = paste0("What do you like from ", rownames(matrix)[1], ": "))
  fs <- readline(prompt = paste0("What do you like from", rownames(matrix)[2], ": "))

  print(paste("Choose the pair you like worse"))
  sf <- readline(prompt = paste0("What do you not like from", rownames(matrix)[1], ": "))
  ss <- readline(prompt = paste0("What do you not like from", rownames(matrix)[2], ": "))

  combine <- c(ff, fs, sf, ss)

  first_match <- 0
  second_match <- 0
  for(i in 1:2){
    first_match[i] <- match(combine[i], matrix)
    second_match[i] <- match(combine[2+i], matrix)
  }

  #Second comparison
  print(paste("We are going to do it once again."))
  print(paste("Choose the pair you like better"))
  ff2 <- readline(prompt = paste0("What do you like from", rownames(matrix)[1], ": "))
  fs2 <- readline(prompt = paste0("What do you like from", rownames(matrix)[2], ": "))

  print(paste("Choose the pair you like worse"))
  sf2 <- readline(prompt = paste0("What do you not like from", rownames(matrix)[1], ": "))
  ss2 <- readline(prompt = paste0("What do you not like from", rownames(matrix)[2], ": "))

  combine2 <- c(ff2, fs2, sf2, ss2)

  first_match2 <- 0
  second_match2 <- 0
  for(i in 1:2){
    first_match2[i] <- match(combine2[i], matrix)
    second_match2[i] <- match(combine2[2+i], matrix)
  }

  #Find intersections of values
  exclude1 <- intersect(first_match, second_match2)
  exclude2 <- intersect(first_match2, second_match)

  first_comb <- c(first_match, first_match2)
  second_comb <- c(second_match, second_match2)

  rm1 <- match(c(exclude1, exclude2), first_comb) #Match gives an index, not a value.
  rm2 <- match(c(exclude1, exclude2), second_comb)
```

```
  output1 <- first_comb[-rm1]
  output2 <- second_comb[-rm2]

  print(paste0("Thus, ", matrix[output1[1]], " and ", matrix[output1[2]],
               " is a preferred choice over ", matrix[output2[1]], " and ", matrix[output2[2]]))
}

conjoining(data)
```

```
## [1] "I am going to ask you to input two pairs of brand and color"
## [1] "Here is the data you have"
##       [,1]      [,2]    [,3]
## brand "samsung" "nokia" "apple"
## color "yellow"  "blue"  "red"
## [1] "Choose the pair you like better"
## What do you like from brand:
## What do you like fromcolor:
## [1] "Choose the pair you like worse"
## What do you not like frombrand:
## What do you not like fromcolor:
## [1] "We are going to do it once again."
## [1] "Choose the pair you like better"
## What do you like frombrand:
## What do you like fromcolor:
## [1] "Choose the pair you like worse"
## What do you not like frombrand:
## What do you not like fromcolor:
## [1] "Thus, NA and NA is a preferred choice over NA and NA"
```

```
conjoining(data2)
```

```
## [1] "I am going to ask you to input two pairs of sandwich type and condiment"
## [1] "Here is the data you have"
##               [,1]            [,2]          [,3]
## sandwich type "peanut butter" "turkey"      "ham"
## condiment     "mustard"       "grape jelly" "cranberry sauce"
## [1] "Choose the pair you like better"
## What do you like from sandwich type:
## What do you like fromcondiment:
## [1] "Choose the pair you like worse"
## What do you not like fromsandwich type:
## What do you not like fromcondiment:
## [1] "We are going to do it once again."
## [1] "Choose the pair you like better"
## What do you like fromsandwich type:
## What do you like fromcondiment:
## [1] "Choose the pair you like worse"
## What do you not like fromsandwich type:
## What do you not like fromcondiment:
## [1] "Thus, NA and NA is a preferred choice over NA and NA"
```

Try to build up 3 by 3 examples

Try to build up 2 by 4 examples

Try to build up 3 by 4 examples

Try to build up 4 by 4 examples

Faisalconjoint

```
data(mobile_data)
mobile_data
```

```
##    Profile Brand FM.Radio Camera Price Rank
## 1        1     4        2      2     1    4
## 2        2     2        1      2     4   12
## 3        3     1        2      2     4   14
## 4        4     4        1      1    10   11
## 5        5     1        1      2     5    7
## 6        6     2        1      1     9   15
## 7        7     5        1      2     2    2
## 8        8     5        2      1     6    5
## 9        9     2        2      2     3    8
## 10      10     2        2      1     8    6
## 11      11     3        2      1     7    3
## 12      12     1        2      1    11    1
## 13      13     1        1      1    12   10
## 14      14     3        2      2     3    9
## 15      15     3        1      1     9   16
## 16      16     3        1      2     4   13
```

```
data(mobile_levels)
mobile_levels
```

```
##           Levels
## 1          NOKIA
## 2  Sony Erricson
## 3        Samsung
```

```
## 4        Motrola
## 5             LG
## 6            Yes
## 7             No
## 8            Yes
## 9             No
## 10          2500
## 11          2700
## 12          2800
## 13          3000
## 14          3300
## 15          3600
## 16          4000
## 17          4200
## 18          5300
## 19          5400
## 20          5500
## 21          6300
```

Format:

A data frame with 16 observations on the following 6 variables. Profile a numeric vector containing profile number Brand 1= Nokia, 2 = Sony Ericsson, 3 = Samsung, 4 = LG, 5 = Motorola FM.Radio 1 = yes, 2 = No Camera 1 = yes, 2= No Price Different price level from market Rank customer preference to purchase, Greatest to Least