The cbcTools Package: Tools for Designing and Testing Choice-Based Conjoint Surveys in R

John Paul Helveston, Ph.D.*

Abstract

Traditional tools for designing choice-based conjoint survey experiments focus on optimizing the design of experiment for statistical power under ideal conditions. But these tools rarely provide guidance on important design decisions for less ideal conditions, such as when preference heterogeneity may be expected in respondent choices or when strong interactions may be expected between certain attributes. The `cbcTools` R package was developed to provide researchers tools for creating and assessing experiment designs and sample size requirements under a variety of different conditions prior to fielding an experiment. The package contains functions for generating experiment designs and surveys as well as functions for simulating choice data and conducting power analyses. Since the package data format matches that of designs exported from Sawtooth Software, it should integrate into the Sawtooth workflow. Detailed package documentation can be found at https://jhelvy.github.io/cbcTools/.

Designing a choice-based conjoint survey is almost never a simple, straightforward process. Designers must consider a wide variety of trade offs between design parameters (e.g., which attributes and levels to include, how many choice questions to ask each respondent, and how many alternatives per choice question) and the design outcomes in terms of the user experience and the statistical power available for identifying effects.

.center[A simple conjoint experiment about *cars*]

Attribute	Levels
Brand	GM, BMW, Ferrari
Price	$20k, $40k, $100k

.center[**Design: .red[9] choice sets, .blue[3] alternatives each**]

`Attribute counts:`

*Engineering Management and Systems Engineering, George Washington University, Washington, D.C. USA

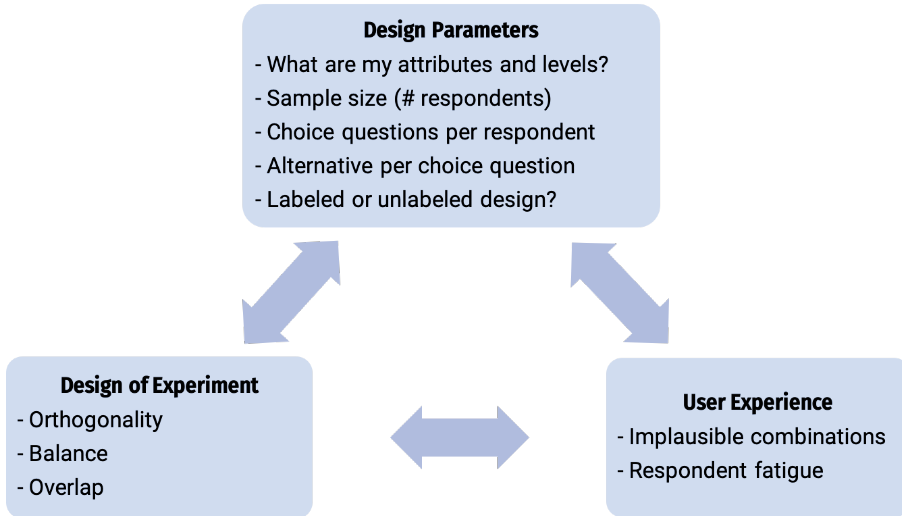


Figure 1: Caption

```
brand:
  GM   BMW  Ferrari
  10    11    6

price:

 20k  40k 100k
  9    9   9

Pairwise attribute counts:

brand & price:

          20k 40k 100k
  GM        3   0    7
  BMW       4   5    2
  Ferrari   2   4    0
```

.center[A simple conjoint experiment about *cars*]

Attribute	Levels
Brand	GM, BMW, Ferrari
Price	$20k, $40k, $100k

.center[**Design: .red[90] choice sets, .blue[3] alternatives each**]

```
Attribute counts:
```



```
brand:
  GM    BMW   Ferrari
  92    80     98
```

```
  BMW     25  25  30
  Ferrari 35  28  35
```

.center[Bayesian D-efficient designs]

.center[Maximize information on "Main Effects" according to priors]

Attribute	Levels	Prior
Brand	GM, BMW, Ferrari	0, 1, 2
Price	$20k, $40k, $100k	0, -1, -4

```
Attribute counts:

brand:
  GM    BMW   Ferrari
  93    90     86

price:

  20k  40k 100k
  97   93   78

Pairwise attribute counts:

brand & price:

          20k 40k 100k
  GM      52  41  0
  BMW     30  30  30
  Ferrari 15  22  49
```

.center[Bayesian D-efficient designs]

.center[Attempts to maximize information on .red[Main Effects]]

"images/design_compare.png"

.center[... but .red[interaction effects] are confounded in D-efficient designs]

"images/design_compare_int.png"

.center[But what about other factors?]

- What if I add one more choice question to each respondent?
- What if I increase the number of alternatives per choice question?
- What if I use a labeled design (aka "alternative-specific design")?
- What if there are interaction effects?

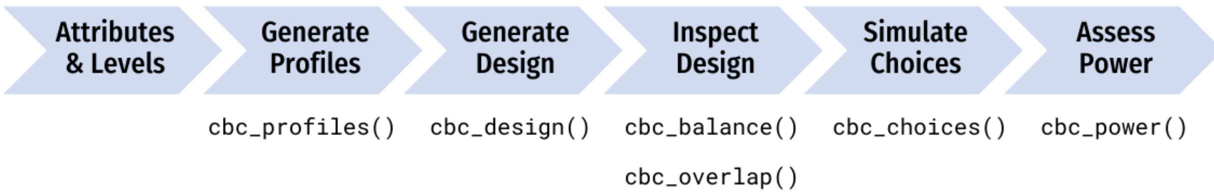


Figure 2: Caption

.center[Define the attributes and levels]

```r
levels <- list(
  price      = c(1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00), # $ per pound
  type       = c("Fuji", "Gala", "Honeycrisp"),
  freshness = c("Excellent", "Average", "Poor")
)
```

```r
levels
```

```
#> $price
#> [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
#>
#> $type
#> [1] "Fuji"       "Gala"       "Honeycrisp"
#>
#> $freshness
#> [1] "Excellent" "Average"   "Poor"
```

.center[Generate all possible profiles]

```r
profiles <- cbc_profiles(levels)
```

```r
head(profiles)
```

```
#>   profileID price type freshness
#> 1         1   1.0 Fuji Excellent
#> 2         2   1.5 Fuji Excellent
```

```
#> 3          3   2.0 Fuji Excellent
#> 4          4   2.5 Fuji Excellent
#> 5          5   3.0 Fuji Excellent
#> 6          6   3.5 Fuji Excellent
```

```
tail(profiles)
```

```
#>    profileID price       type freshness
#> 58        58   1.5 Honeycrisp      Poor
#> 59        59   2.0 Honeycrisp      Poor
#> 60        60   2.5 Honeycrisp      Poor
#> 61        61   3.0 Honeycrisp      Poor
#> 62        62   3.5 Honeycrisp      Poor
#> 63        63   4.0 Honeycrisp      Poor
```

.center[Attribute-specific levels]

```
levels <- list(
  price = c(1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00),
  freshness = c("Excellent", "Average", "Poor"),
  type = list(
    "Fuji" = list(
        price = c(2.00, 2.50, 3.00)
    ),
    "Gala" = list(
        price = c(1.00, 1.50, 2.00)
    ),
    "Honeycrisp" = list(
        price = c(2.50, 3.00, 3.50, 4.00),
        freshness = c("Excellent", "Average")
    )
  )
)
```

.center[Generate restricted set of profiles]

```
profiles <- cbc_profiles(levels)
```

```
head(profiles)
```

```
#>   profileID price freshness type
#> 1         1   2.0 Excellent Fuji
#> 2         2   2.5 Excellent Fuji
#> 3         3   3.0 Excellent Fuji
```

```
#> 4          4   2.0   Average Fuji
#> 5          5   2.5   Average Fuji
#> 6          6   3.0   Average Fuji
```

```
tail(profiles)
```

```
#>    profileID price freshness       type
#> 21        21   3.5 Excellent Honeycrisp
#> 22        22   4.0 Excellent Honeycrisp
#> 23        23   2.5   Average Honeycrisp
#> 24        24   3.0   Average Honeycrisp
#> 25        25   3.5   Average Honeycrisp
#> 26        26   4.0   Average Honeycrisp
```

.center[Generate a survey design]

```
design <- cbc_design(
  profiles = profiles,
  n_resp   = 300, # Number of respondents
  n_alts   = 3,   # Number of alternatives per question
  n_q      = 6    # Number of questions per respondent
)
```

```
head(design)
```

```
#>   respID qID altID obsID profileID price       type freshness
#> 1      1   1     1     1        60   2.5 Honeycrisp      Poor
#> 2      1   1     2     1        44   1.5       Fuji      Poor
#> 3      1   1     3     1        15   1.0 Honeycrisp Excellent
#> 4      1   2     1     2        16   1.5 Honeycrisp Excellent
#> 5      1   2     2     2        54   3.0       Gala      Poor
#> 6      1   2     3     2         1   1.0       Fuji Excellent
```

.center[Include a “no choice” option]

```
design <- cbc_design(
  profiles  = profiles,
  n_resp    = 300, # Number of respondents
  n_alts    = 3,   # Number of alternatives per question
  n_q       = 6,    # Number of questions per respondent
  no_choice = TRUE #<<
)
```

```r
head(design)
```

```
#>   respID qID altID obsID profileID price type_Fuji type_Gala type_Honeycrisp
#> 1      1   1     1     1        29   1.0         0         1               0
#> 2      1   1     2     1        45   2.0         1         0               0
#> 3      1   1     3     1        61   3.0         0         0               1
#> 4      1   1     4     1         0   0.0         0         0               0
#> 5      1   2     1     2         6   3.5         1         0               0
#> 6      1   2     2     2        44   1.5         1         0               0
#>   freshness_Excellent freshness_Average freshness_Poor no_choice
#> 1                   0                 1              0         0
#> 2                   0                 0              1         0
#> 3                   0                 0              1         0
#> 4                   0                 0              0         1
#> 5                   1                 0              0         0
#> 6                   0                 0              1         0
```

.center[Make a labeled design]

.center[.font100[(aka “alternative-specific design”)]]

```r
design <- cbc_design(
  profiles = profiles,
  n_resp   = 300, # Number of respondents
  n_alts   = 3,   # Number of alternatives per question
  n_q      = 6,    # Number of questions per respondent
  label    = "type" #<<
)
```

```r
head(design)
```

```
#>   respID qID altID obsID profileID price       type freshness
#> 1      1   1     1     1        48   3.5       Fuji      Poor
#> 2      1   1     2     1        56   4.0       Gala      Poor
#> 3      1   1     3     1        63   4.0 Honeycrisp      Poor
#> 4      1   2     1     2        23   1.5       Fuji   Average
#> 5      1   2     2     2        10   2.0       Gala Excellent
#> 6      1   2     3     2        63   4.0 Honeycrisp      Poor
```

.center[Make a Bayesian D-efficient design]

.center[(coming soon!)]

```r
design <- cbc_design(
  profiles = profiles,
```

```
  n_resp   = 300, # Number of respondents
  n_alts   = 3,   # Number of alternatives per question
  n_q      = 6,   # Number of questions per respondent
  priors = list( #<<
    price     = -0.1, #<<
    type      = c(0.1, 0.2), #<<
    freshness = c(0.1, -0.2) #<<
  ) #<<
)
```

.center[Make a Bayesian D-efficient design]

.center[(coming soon!)]

- Check out the `idefix` package

- Import a design: .blue[Sawtooth]

.center[Check design balance]

```
cbc_balance(design)
```

```
Attribute counts:

price:

   1 1.5   2 2.5   3 3.5   4
 825 797 743 743 767 779 746

type:

 Fuji       Gala Honeycrisp
 1842       1769       1789

freshness:

 Excellent   Average      Poor
      1813      1775      1812

Pairwise attribute counts:

price & type:

   Fuji Gala Honeycrisp
```

```
  1     304  252         269
  1.5   274  251         272
  2     257  254         232
  2.5   240  254         249
  3     249  263         255
  3.5   257  250         272
  4     261  245         240
```

.center[Check design overlap]

```
cbc_overlap(design)
```

```
Counts of attribute overlap:
(# of questions with N unique levels)

price:

     1    2    3
    31  630 1139

type:

     1    2    3
   156 1248  396

freshness:

     1    2    3
   175 1189  436
```

.center[Simulate random choices]

```
data <- cbc_choices(
  design = design,
  obsID  = "obsID"
)
```

```
head(data)
```

```
#>   respID qID altID obsID profileID price       type freshness choice
#> 1      1   1     1     1        48   3.5       Fuji      Poor      1
#> 2      1   1     2     1        56   4.0       Gala      Poor      0
#> 3      1   1     3     1        63   4.0 Honeycrisp      Poor      0
```

```
#> 4       1   2     1     2         23   1.5        Fuji   Average       0
#> 5       1   2     2     2         10   2.0        Gala Excellent       1
#> 6       1   2     3     2         63   4.0 Honeycrisp       Poor       0
```

.center[Simulate choices according to a prior]

```
data <- cbc_choices(
  design = design,
  obsID = "obsID",
  priors = list( #<<
    price     = -0.1, #<<
    type      = c(0.1, 0.2), #<<
    freshness = c(0.1, -0.2) #<<
  ) #<<
)
```

Attribute	Level
Price	Continuous
Type	Fuji
Gala	0.1
Honeycrisp	0.2
Freshness	Average
Excellent	0.1
Poor	-0.2

]

.center[Simulate choices according to a prior]

```
data <- cbc_choices(
  design = design,
  obsID = "obsID",
  priors = list(
    price = -0.1,
    type = randN(  #<<
      mu    = c(0.1, 0.2), #<<
      sigma = c(0.5, 1) #<<
    ), #<<
    freshness = c(0.1, -0.2)
  )
)
```

Attribute	Level
Price	Continuous
Type	Fuji
Gala	N(0.1, 0.5)
Honeycrisp	N(0.2, 1)
Freshness	Average
Excellent	0.1
Poor	-0.2

]

.center[Simulate choices according to a prior]

```
data <- cbc_choices(
  design = design,
  obsID = "obsID",
  priors = list(
    price      = -0.1,
    type       = c(0.1, 0.2),
    freshness = c(0.1, -0.2),
    "price*type" = c(0.1, 0.5) #<<
  )
)
```

Attribute	Level
Price	Continuous
Type	Fuji
Gala	0.1
Honeycrisp	0.2
Freshness	Average
Excellent	0.1
Poor	-0.2
Price x Type	Fuji
Gala	0.1
Honeycrisp	0.5

]

Power analyses

```r
power <- cbc_power(
    nbreaks = 10,
    n_q     = 6,
    data    = data,
    obsID   = "obsID",
    outcome = "choice",
    pars    = c("price", "type", "freshness")
)
```

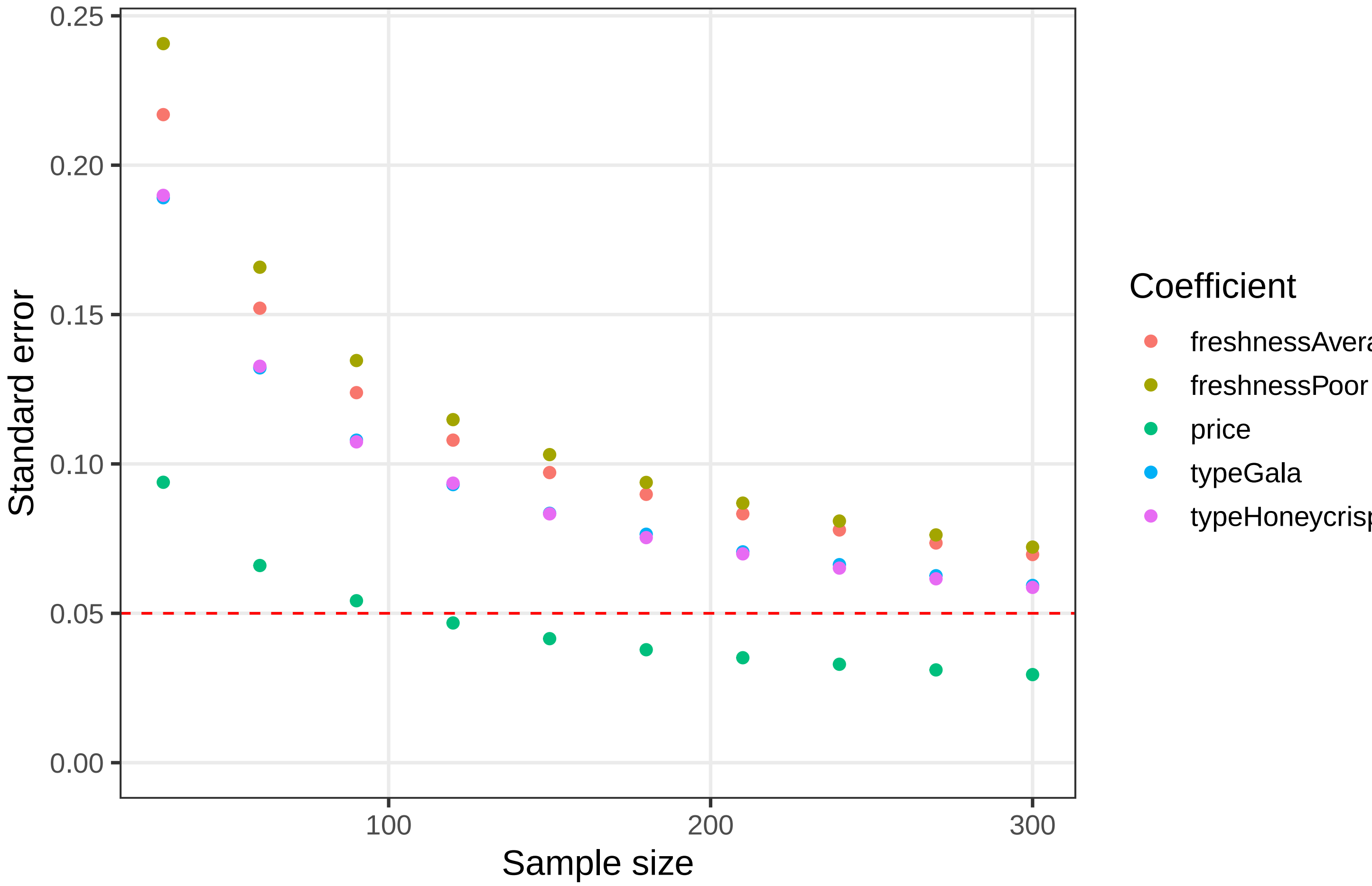
```r
head(power)
```

```
#>   sampleSize             coef        est         se
#> 1         30            price -0.1936219 0.09386325
#> 2         30         typeGala  0.1286319 0.18910349
#> 3         30   typeHoneycrisp  0.1483335 0.18989859
#> 4         30 freshnessAverage  0.1058520 0.21692308
#> 5         30    freshnessPoor -0.4049243 0.24068743
#> 6         60            price -0.1784406 0.06599736
```

```r
tail(power)
```

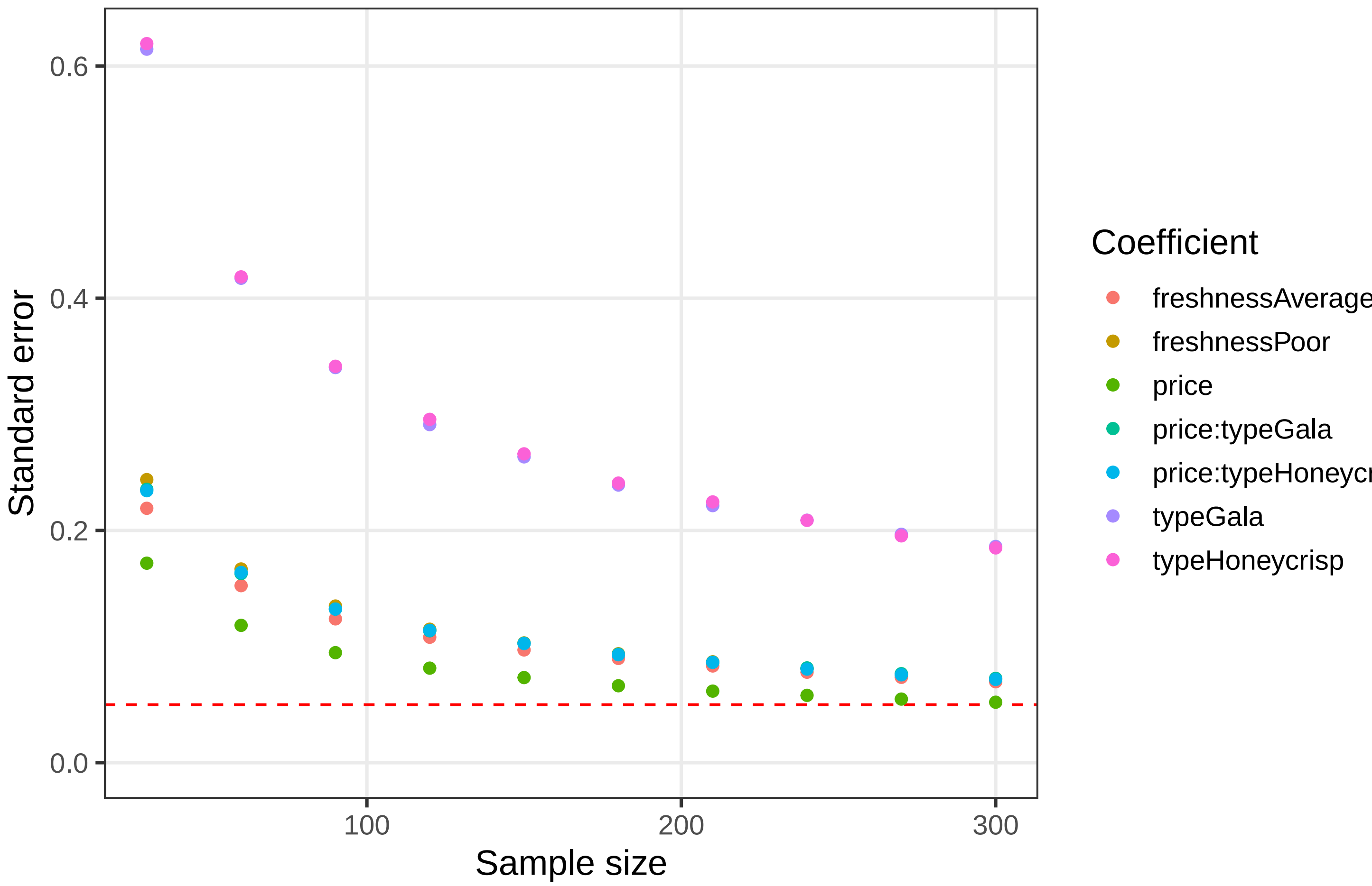
```
#>    sampleSize             coef         est         se
#> 45        270    freshnessPoor -0.20558097 0.07624275
#> 46        300            price -0.13617149 0.02949081
#> 47        300         typeGala  0.13650079 0.05933542
#> 48        300   typeHoneycrisp  0.19142805 0.05868091
#> 49        300 freshnessAverage  0.08825258 0.06967591
#> 50        300    freshnessPoor -0.17087775 0.07217023
```

```r
plot(power)
```




```
power_int <- cbc_power(
    nbreaks = 10,
    n_q     = 6,
    data    = data,
    pars    = c(
      "price",
      "type",
      "freshness",
      "price*type" #<<
    ),
    outcome = "choice",
    obsID   = "obsID"
)
```

```
plot(power_int)
```



`cbcTools` documentation: https://jhelvy.github.io/cbcTools/

References