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

John Paul Helveston, Ph.D.[[1]](#footnote-20)

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 “good” choice-based conjoint survey is almost never a simple, straightforward process.

“images/tradeoffs2.png”

# .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:  
  
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  
  
price:  
  
 20k 40k 100k   
 91 84 95

Pairwise attribute counts:  
  
brand & price:  
   
 20k 40k 100k  
 GM 31 31 30  
 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?

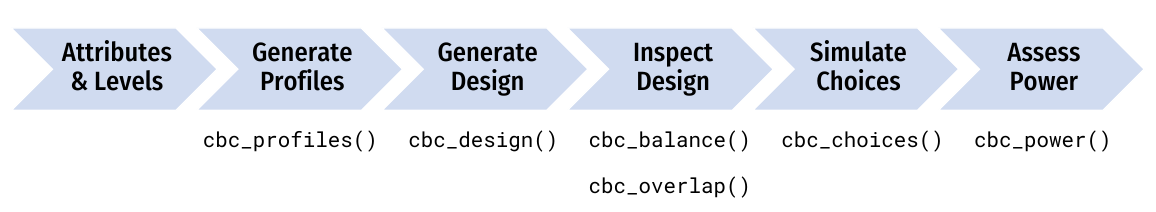


Figure 1 Caption

# .center[Define the attributes and levels]

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")  
)

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]

profiles <- cbc\_profiles(levels)

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 1.0 Excellent 2.0, 2.5, 3.0  
#> 2 2 1.5 Excellent 2.0, 2.5, 3.0  
#> 3 3 2.0 Excellent 2.0, 2.5, 3.0  
#> 4 4 2.5 Excellent 2.0, 2.5, 3.0  
#> 5 5 3.0 Excellent 2.0, 2.5, 3.0  
#> 6 6 3.5 Excellent 2.0, 2.5, 3.0

tail(profiles)

#> profileID price freshness type  
#> 58 58 1.5 Poor 2.5, 3, 3.5, 4, Excellent, Average  
#> 59 59 2.0 Poor 2.5, 3, 3.5, 4, Excellent, Average  
#> 60 60 2.5 Poor 2.5, 3, 3.5, 4, Excellent, Average  
#> 61 61 3.0 Poor 2.5, 3, 3.5, 4, Excellent, Average  
#> 62 62 3.5 Poor 2.5, 3, 3.5, 4, Excellent, Average  
#> 63 63 4.0 Poor 2.5, 3, 3.5, 4, Excellent, Average

# .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 28 4.0 Fuji Average  
#> 2 1 1 2 1 40 3.0 Honeycrisp Average  
#> 3 1 1 3 1 33 3.0 Gala Average  
#> 4 1 2 1 2 37 1.5 Honeycrisp Average  
#> 5 1 2 2 2 24 2.0 Fuji Average  
#> 6 1 2 3 2 30 1.5 Gala Average

# .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 #<<  
)

head(design)

#> respID qID altID obsID profileID price type\_Fuji type\_Gala  
#> 1 1 1 1 1 18 2.5 0 0  
#> 2 1 1 2 1 62 3.5 0 0  
#> 3 1 1 3 1 53 2.5 0 1  
#> 11000 1 1 4 1 0 0.0 0 0  
#> 4 1 2 1 2 40 3.0 0 0  
#> 5 1 2 2 2 23 1.5 1 0  
#> type\_Honeycrisp freshness\_Excellent freshness\_Average freshness\_Poor  
#> 1 1 1 0 0  
#> 2 1 0 0 1  
#> 3 0 0 0 1  
#> 11000 0 0 0 0  
#> 4 1 0 1 0  
#> 5 0 0 1 0  
#> no\_choice  
#> 1 0  
#> 2 0  
#> 3 0  
#> 11000 1  
#> 4 0  
#> 5 0

## .center[Make a labeled design]

.center[.font100[(aka “alternative-specific 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  
 label = "type" #<<  
)

head(design)

#> respID qID altID obsID profileID price type freshness  
#> 1 1 1 1 1 4 2.5 Fuji Excellent  
#> 2 1 1 2 1 55 3.5 Gala Poor  
#> 3 1 1 3 1 62 3.5 Honeycrisp Poor  
#> 4 1 2 1 2 48 3.5 Fuji Poor  
#> 5 1 2 2 2 53 2.5 Gala Poor  
#> 6 1 2 3 2 63 4.0 Honeycrisp Poor

# .center[Make a Bayesian D-efficient design]

### .center[(coming soon!)]

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](https://www.jstatsoft.org/article/view/v096i03) 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 4 2.5 Fuji Excellent 0  
#> 2 1 1 2 1 55 3.5 Gala Poor 1  
#> 3 1 1 3 1 62 3.5 Honeycrisp Poor 0  
#> 4 1 2 1 2 48 3.5 Fuji Poor 0  
#> 5 1 2 2 2 53 2.5 Gala Poor 0  
#> 6 1 2 3 2 63 4.0 Honeycrisp Poor 1

# .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

power <- cbc\_power(  
 nbreaks = 10,  
 n\_q = 6,  
 data = data,  
 obsID = "obsID",  
 outcome = "choice",  
 pars = c("price", "type", "freshness")  
)

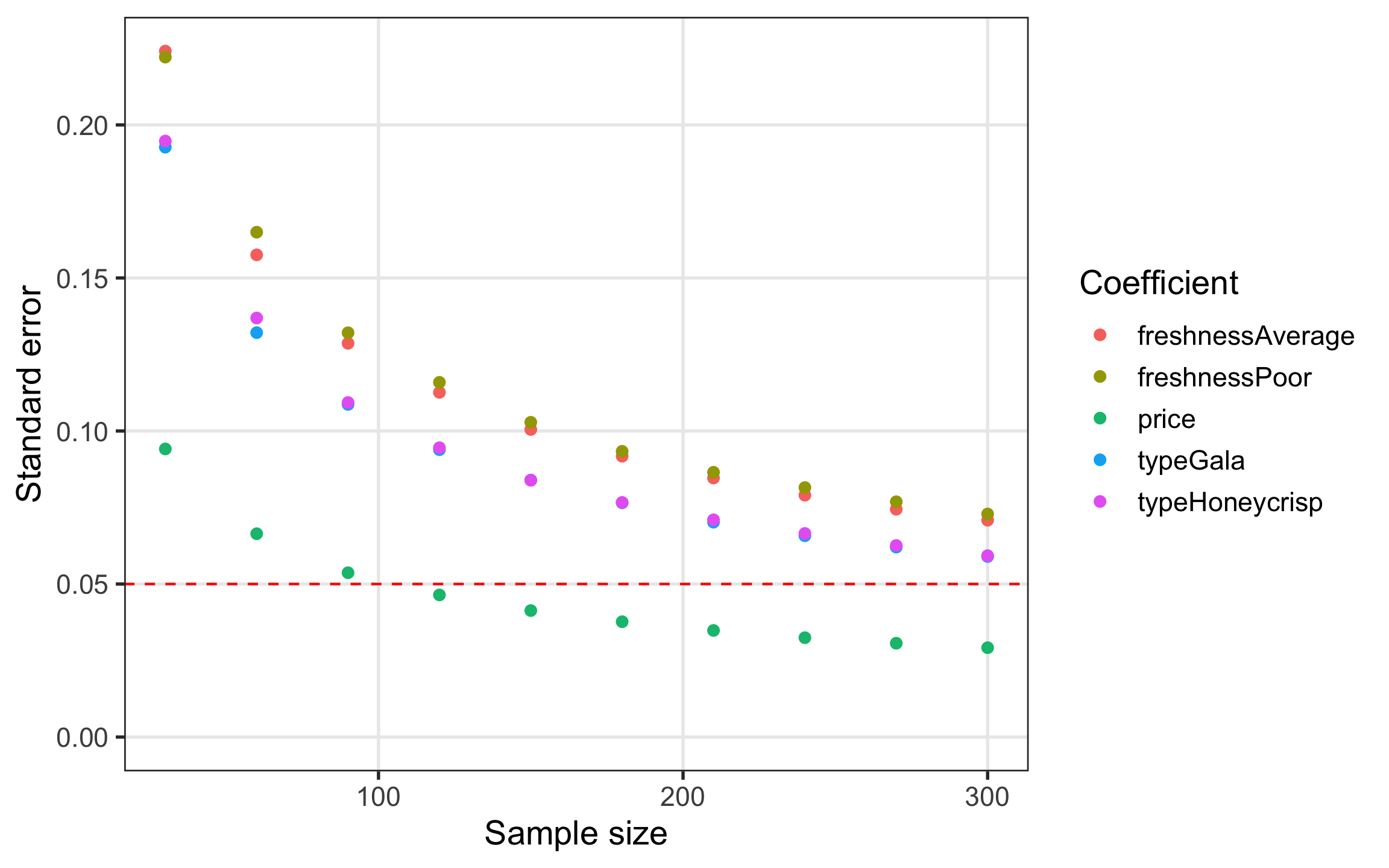
head(power)

#> sampleSize coef est se  
#> 1 30 price -0.17081907 0.09412772  
#> 2 30 typeGala 0.40420813 0.19272648  
#> 3 30 typeHoneycrisp 0.37538120 0.19472639  
#> 4 30 freshnessAverage 0.06374861 0.22410740  
#> 5 30 freshnessPoor -0.23367779 0.22215216  
#> 6 60 price -0.10399409 0.06643510

tail(power)

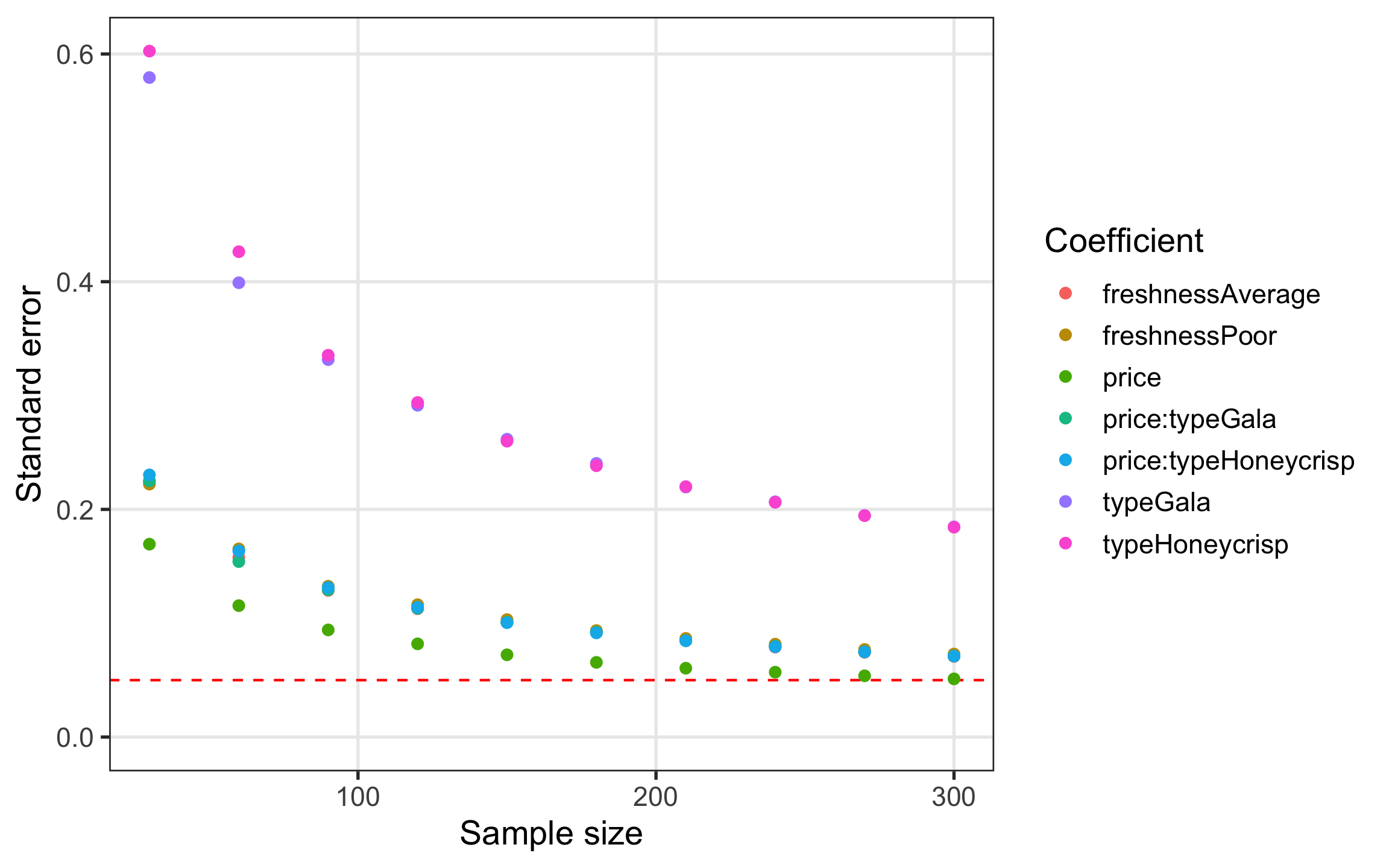
#> sampleSize coef est se  
#> 45 270 freshnessPoor -0.1405221 0.07688366  
#> 46 300 price -0.1182197 0.02919575  
#> 47 300 typeGala 0.1873587 0.05898732  
#> 48 300 typeHoneycrisp 0.1682193 0.05925969  
#> 49 300 freshnessAverage 0.1907981 0.07085259  
#> 50 300 freshnessPoor -0.1083857 0.07288734

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

1. Engineering Management and Systems Engineering, George Washington University, Washington, D.C. USA [↑](#footnote-ref-20)