Designing Conjoint Surveys with {cbcTools}

John Paul Helveston

The George Washington University | Dept. of Engineering Management and Systems Engineering

June 15, 2023

Designing a Choice-Based Conjoint Survey is Hard

Design Parameters

- What are my attributes and levels?
- Sample size (# respondents)
- Choice questions per respondent
- Alternative per choice question
- Labeled or unlabeled design?

Designing a Choice-Based Conjoint Survey is Hard

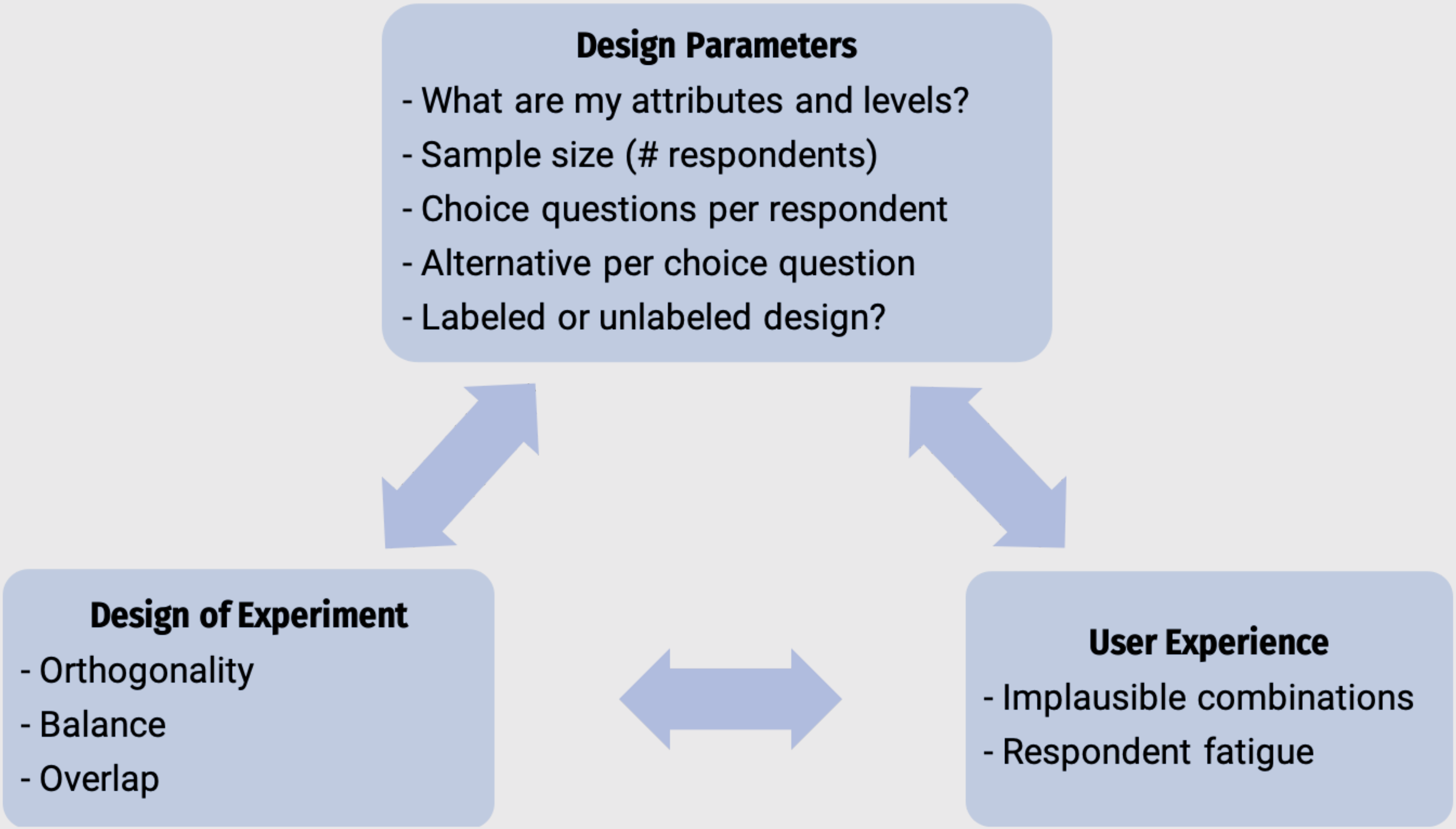
Design Parameters

- What are my attributes and levels?
- Sample size (# respondents)
- Choice questions per respondent
- Alternative per choice question
- Labeled or unlabeled design?

Design of Experiment

- Orthogonality
- Balance
- Overlap

Designing a Choice-Based Conjoint Survey is Hard



A systematic workflow for designing a CBC experiment

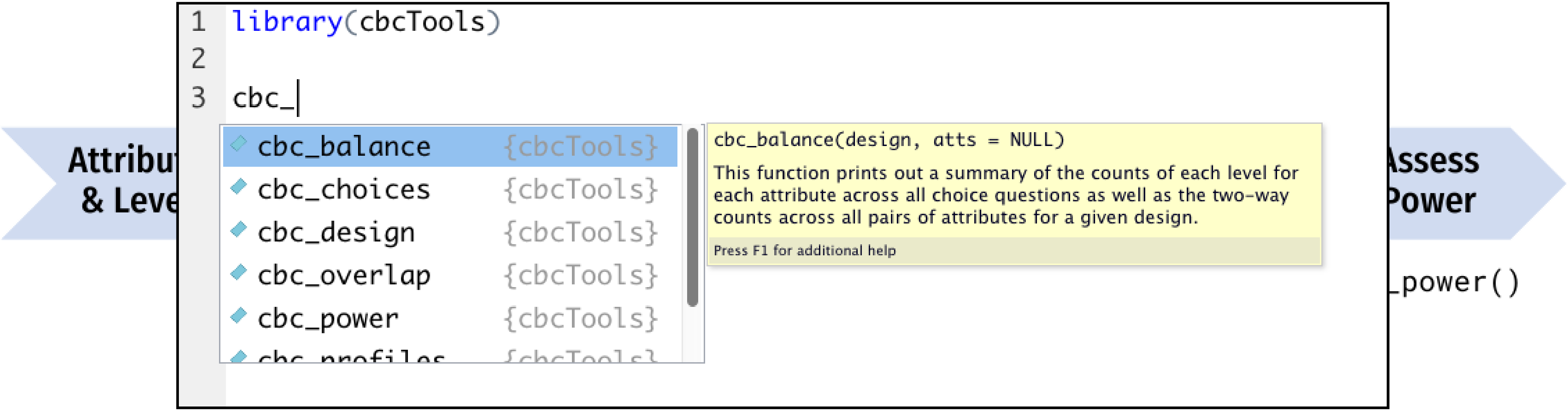
cbcTools

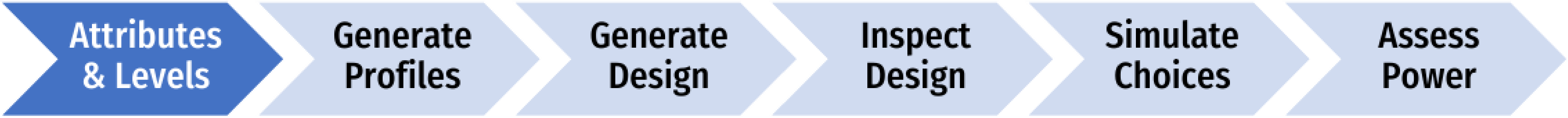
A systematic workflow for designing a CBC experiment

A systematic workflow for designing a CBC experiment

`cbc_profiles()` `cbc_design()` `cbc_balance()` `cbc_choices()` `cbc_power()`

`cbc_overlap()`

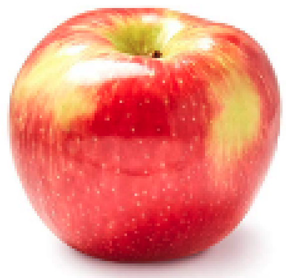

Attribu
& Leve
library(cbcTools)
cbc_
cbc_balance {cbcTools}
cbc_choices {cbcTools}
cbc_design {cbcTools}
cbc_overlap {cbcTools}
cbc_power {cbcTools}
cbc_balance(design, atts = NULL)
This function prints out a summary of the counts of each level for each attribute across all choice questions as well as the two-way counts across all pairs of attributes for a given design.
Press F1 for additional help
Assess
Power
_power()



`cbc_profiles()` `cbc_design()` `cbc_balance()` `cbc_choices()` `cbc_power()`

`cbc_overlap()`

Example CBC question about apples

Option 1	Option 2	Option 3
		
Type: Pink Lady **Price**: $ 2 / lb **Freshness**: Average	**Type**: Pink Lady **Price**: $ 1.5 / lb **Freshness**: Excellent	**Type**: Honeycrisp **Price**: $ 2 / lb **Freshness**: Average

Define the attributes and levels

- **Price ($/lb)**: 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00
- **Type**: Fuji, Gala, Honeycrisp
- **Freshness**: Excellent, Average, Poor

`cbc_profiles()` `cbc_design()` `cbc_balance()` `cbc_overlap()` `cbc_choices()` `cbc_power()`

Generate all possible profiles

```r
profiles <- cbc_profiles(
  price     = seq(1, 4, 0.5), # $ per pound
  type      = c('Fuji', 'Gala', 'Honeycrisp'),
  freshness = c('Poor', 'Average', 'Excellent')
)
```

```r
head(profiles)
```

```
#>   profileID price type freshness
#> 1         1   1.0 Fuji      Poor
#> 2         2   1.5 Fuji      Poor
#> 3         3   2.0 Fuji      Poor
#> 4         4   2.5 Fuji      Poor
#> 5         5   3.0 Fuji      Poor
#> 6         6   3.5 Fuji      Poor
```

```r
tail(profiles)
```

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

Generate a restricted set of profiles?

CAUTION: including restrictions in your designs can substantially reduce the statistical power of your design, so use them cautiously (and avoid them if possible).

```
restricted_profiles <- cbc_restrict(
    profiles,
    type == "Gala" & price %in% c(1.5, 2.5, 3.5),
    type == "Honeycrisp" & price < 2,
    type == "Fuji" & freshness == "Poor"
)

dim(restricted_profiles)
```

```
#> [1] 41  4
```

`cbc_profiles()` **`cbc_design()`** `cbc_balance()` `cbc_choices()` `cbc_power()`

`cbc_overlap()`

Generate a survey 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
)
```

```r
head(design)
```

```
#>   profileID respID qID altID obsID price       type freshness
#> 1        38      1   1     1     1   2.0 Honeycrisp   Average
#> 2        39      1   1     2     1   2.5 Honeycrisp   Average
#> 3        20      1   1     3     1   3.5 Honeycrisp      Poor
#> 4         3      1   2     1     2   2.0       Fuji      Poor
#> 5        60      1   2     2     2   2.5 Honeycrisp Excellent
#> 6        13      1   2     3     2   3.5       Gala      Poor
```

Include a "no choice" option

```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
  no_choice = TRUE
)
```

```r
head(design)
```

```
#>   profileID respID qID altID obsID price type_Fuji type_Gala type_Honeycrisp freshness_
#> 1         7      1   1     1     1   4.0         1         0               0
#> 2        46      1   1     2     1   2.5         1         0               0
#> 3        23      1   1     3     1   1.5         1         0               0
#> 4         0      1   1     4     1   0.0         0         0               0
#> 5        19      1   2     1     2   3.0         0         0               1
#> 6        34      1   2     2     2   3.5         0         1               0
```

Make a labeled design

(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)
```

```
#>   profileID respID qID altID obsID price       type freshness
#> 1        47      1   1     1     1   3.0       Fuji Excellent
#> 2        56      1   1     2     1   4.0       Gala Excellent
#> 3        40      1   1     3     1   3.0 Honeycrisp   Average
#> 4        45      1   2     1     2   2.0       Fuji Excellent
#> 5        51      1   2     2     2   1.5       Gala Excellent
#> 6        38      1   2     3     2   2.0 Honeycrisp   Average
```

Make a Bayesian D-efficient design

(Uses the `idefix` package to generate a 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
  priors = list(
    price     = -0.1, # Numeric, modeled as continuous
    type      = c(0.1, 0.2), # Reference level: "Fuji"
    freshness = c(0.1, 0.2) # Reference level: "Poor"
  )
)
```

Priors are defining the following model:

$$u_j = -0.1p_j + 0.1t_j^{Gala} + 0.2t_j^{Honeycrisp} + 0.1f_j^{Ave} + 0.2f_j^{Excellent} + \varepsilon_j$$

Import a design: Sawtooth → CSV → R

```r
library(readr)

design <- read_csv('design.csv')

head(design)
```

```
#>   respID qID altID obsID price       type freshness
#> 1      1   1     1     1   3.0       Fuji Excellent
#> 2      1   1     2     1   4.0       Gala Excellent
#> 3      1   1     3     1   3.0 Honeycrisp   Average
#> 4      1   2     1     2   2.0       Fuji Excellent
#> 5      1   2     2     2   1.5       Gala Excellent
#> 6      1   2     3     2   2.0 Honeycrisp   Average
```

Attributes & Levels	Generate Profiles	Generate Design	**Inspect Design**	Simulate Choices	Assess Power
	`cbc_profiles()`	`cbc_design()`	**`cbc_balance()`** **`cbc_overlap()`**	`cbc_choices()`	`cbc_power()`

Check design **balance**

```
cbc_balance(design)
```

```
Individual attribute level counts

price:

  1 1.5   2 2.5   3 3.5   4
784 755 759 741 776 827 758

type:

      Fuji       Gala Honeycrisp
      1800       1800       1800

freshness:

     Poor   Average Excellent
     1845      1767      1788
```

```
Pairwise attribute level counts

price x type:

         Fuji Gala Honeycrisp
      NA 1800 1800       1800
1    784  260  256        268
1.5  755  248  254        253
2    759  259  240        260
2.5  741  239  254        248
3    776  263  286        227
3.5  827  264  258        305
4    758  267  252        239
```

Check design **overlap**

```r
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
```

`cbc_profiles()` `cbc_design()` `cbc_balance()` **`cbc_choices()`** `cbc_power()`

`cbc_overlap()`

Simulate random choices

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

```r
head(data)
```

```
#>   profileID respID qID altID obsID price       type freshness choice
#> 1        47      1   1     1     1   3.0       Fuji Excellent      0
#> 2        56      1   1     2     1   4.0       Gala Excellent      1
#> 3        40      1   1     3     1   3.0 Honeycrisp   Average      0
#> 4        45      1   2     1     2   2.0       Fuji Excellent      0
#> 5        51      1   2     2     2   1.5       Gala Excellent      0
#> 6        38      1   2     3     2   2.0 Honeycrisp   Average      1
```

Simulate choices according to a prior

(Fixed coefficients)

```r
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	Utility
Price	Continuous	-0.1
Type	Fuji	0
	Gala	0.1
	Honeycrisp	0.2
Freshness	Average	0
	Excellent	0.1
	Poor	-0.2

Simulate choices according to a prior

(Random coefficients...currently supports Normal & Log-normal)

```
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)
  )
)
```

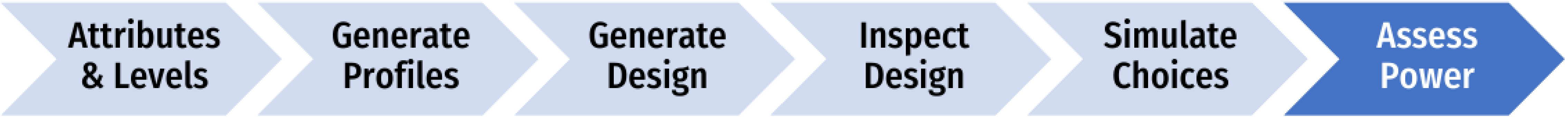
Attribute	Level	Utility
Price	Continuous	-0.1
Type	Fuji	0
	Gala	N(0.1, 0.5)
	Honeycrisp	N(0.2, 1)
Freshness	Average	0
	Excellent	0.1
	Poor	-0.2

Simulate choices according to a prior

(Models with interactions)

```r
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	Utility
Price	Continuous	-0.1
Type	Fuji	0
	Gala	0.1
	Honeycrisp	0.2
Freshness	Average	0
	Excellent	0.1
	Poor	-0.2
Price x Type	Fuji	0
	Gala	0.1
	Honeycrisp	0.5



`cbc_profiles()` `cbc_design()` `cbc_balance()` `cbc_choices()` **`cbc_power()`**

`cbc_overlap()`

Conduct a power analysis

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

```r
head(power)
```

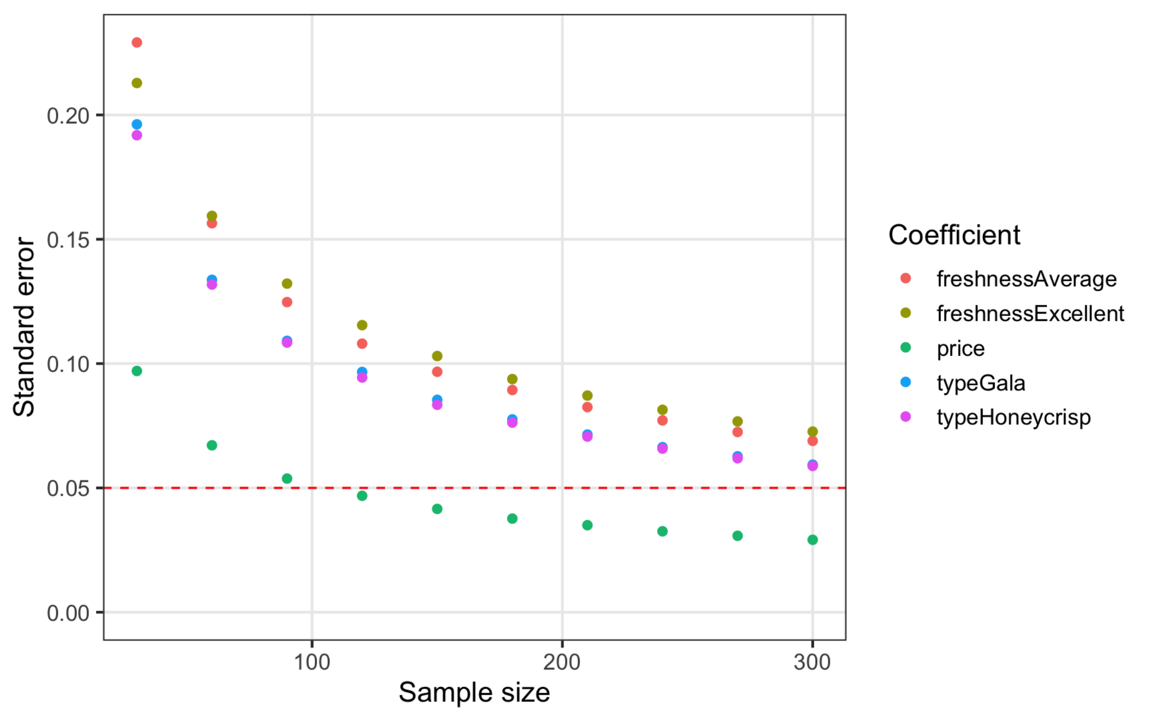
```
#>   sampleSize               coef         es
#> 1         30              price -0.128968
#> 2         30           typeGala  0.309739
#> 3         30     typeHoneycrisp  0.403983
#> 4         30    freshnessAverage -0.493620
#> 5         30  freshnessExcellent -0.277370
#> 6         60              price -0.081152
```

```r
tail(power)
```

```
#>    sampleSize               coef
#> 45        270 freshnessExcellent -0.31731
#> 46        300              price -0.08985
#> 47        300           typeGala  0.15681
#> 48        300     typeHoneycrisp  0.19697
#> 49        300   freshnessAverage -0.05403
#> 50        300 freshnessExcellent -0.31296
```

Conduct a power analysis

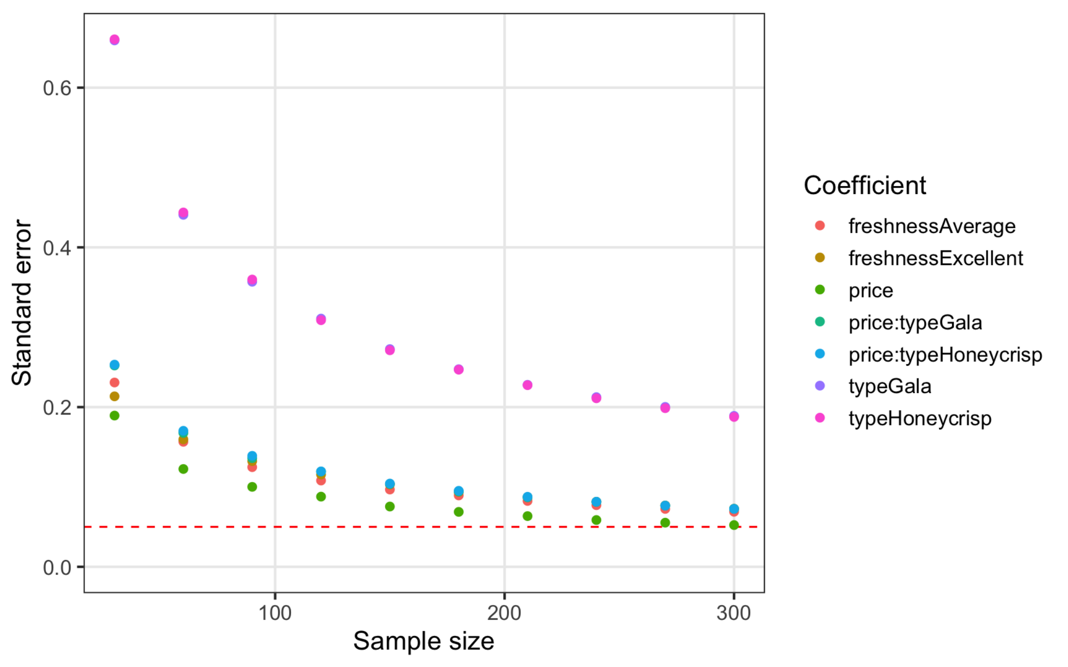
```
plot(power)
```



Conduct a power analysis

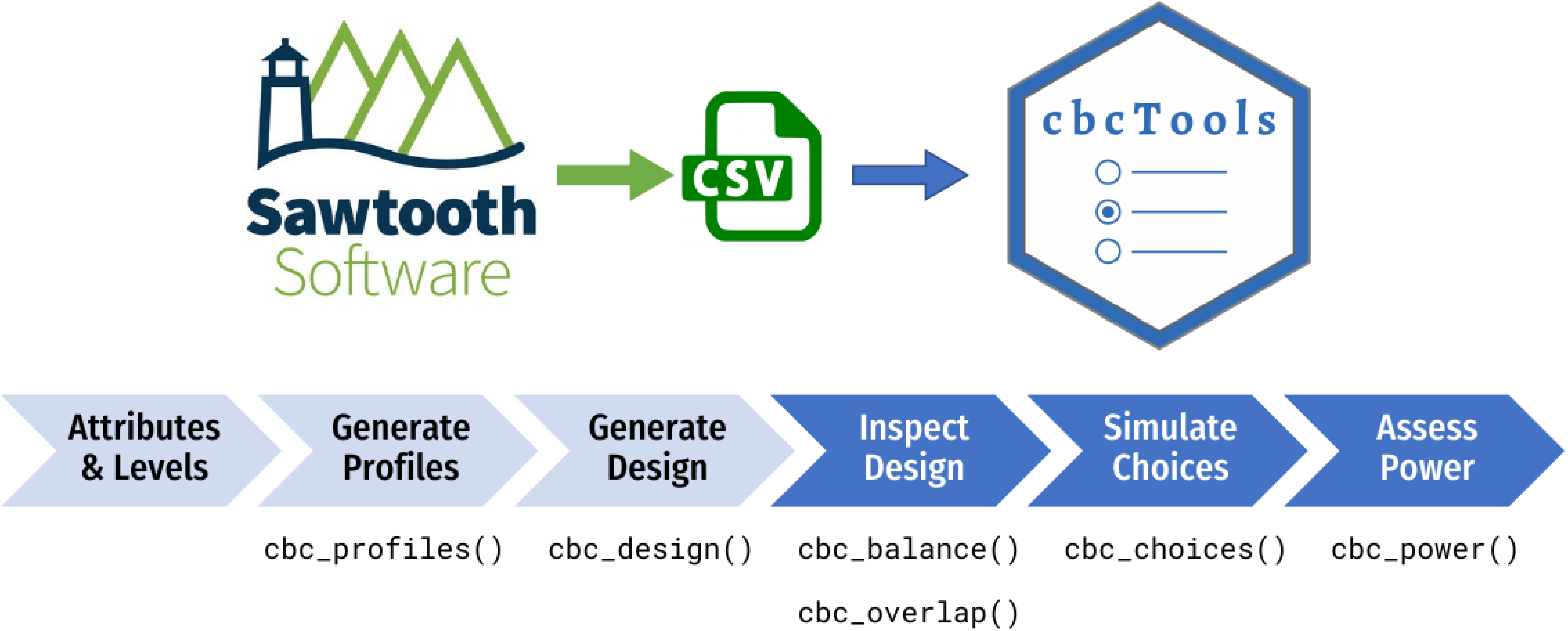
```r
power_int <- cbc_power(
    nbreaks = 10,
    n_q     = 6,
    data    = data,
    pars    = c(
      "price",
      "type",
      "freshness",
      "price*type"
    ),
    outcome = "choice",
    obsID   = "obsID"
)
```

```r
plot(power_int)
```



cbcTools

Attributes & Levels	Generate Profiles	Generate Design	Inspect Design	Simulate Choices	Assess Power
	`cbc_profiles()`	`cbc_design()`	`cbc_balance()`	`cbc_choices()`	`cbc_power()`
			`cbc_overlap()`		


Sawtooth
Software
CSV
cbcTools
Attributes & Levels
Generate Profiles
Generate Design
Inspect Design
Simulate Choices
Assess Power
cbc_profiles()
cbc_design()
cbc_balance()
cbc_overlap()
cbc_choices()
cbc_power()

cbcTools

Attributes & Levels

Generate Profiles

`cbc_profiles()`

Generate Design

`cbc_design()`

Inspect Design

`cbc_balance()`

`cbc_overlap()`

Simulate Choices

`cbc_choices()`

Assess Power

`cbc_power()`

CSV

Sawtooth Software

Your turn

- Download the practice zip file for this section.
- Open the `designing-surveys.Rproj` file to open RStudio.
- In RStudio, open the `practice.R` file.
- Experiment with different design options, then examine the power:
  - What if you modify the quesitons per respondent?
  - What if you use a labeled design?
  - What if you include a "no choice" option?
  - What if you use a Bayesian D-efficient design?

Back to workshop website:

https://jhelvy.github.io/2023-qux-conf-conjoint/

@JohnHelveston
@jhelvy
@jhelvy
jhelvy.com
jph@gwu.edu