The `cbcTools` Package

Tools for Designing and Testing Choice-Based Conjoint Surveys in R

by John Paul Helveston

Sawtooth Software Conference

May 06, 2022

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

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

User Experience

- Implausible combinations
- Respondent fatigue

A simple conjoint experiment about *cars*

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

Design: 9 choice sets, 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
```

A simple conjoint experiment about *cars*

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

Design: 90 choice sets, 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
```

Bayesian D-efficient designs

Maximize information on "Main Effects" according to priors

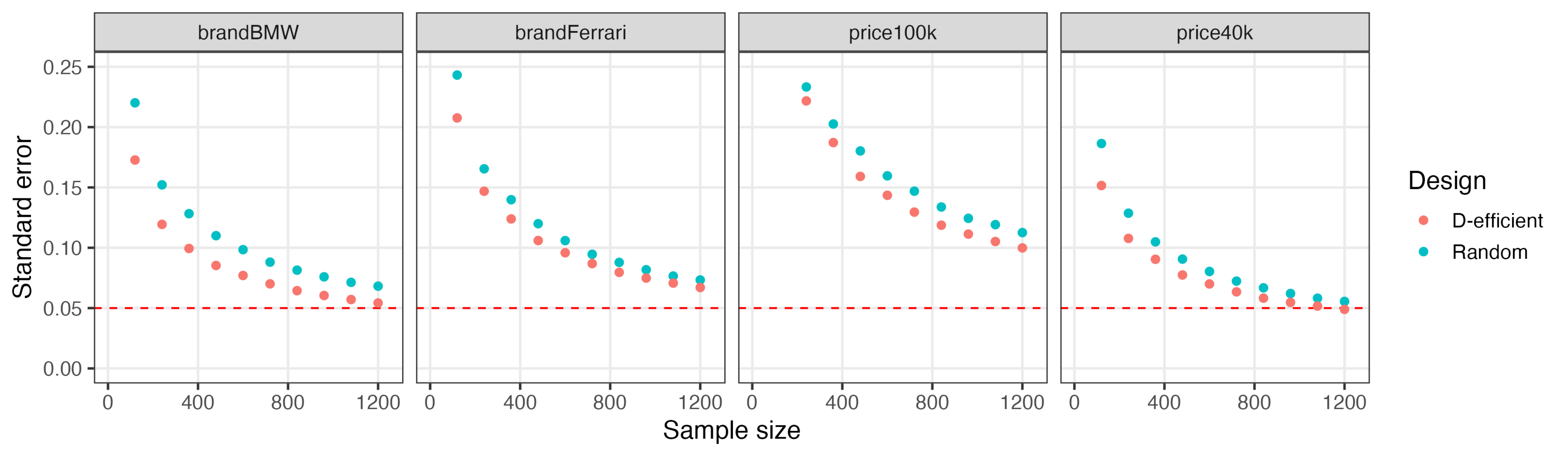
Attribute	Levels	Prior
Brand	GW, BMW, Ferrari	0, 1, 2
Price	$20k, $40k, $100k	0, -1, -4

```
Attribute counts:

brand:
  GM    BMW   Ferrari
  93    90     86

price:

  20k  40k 100k
  97   93   78
```

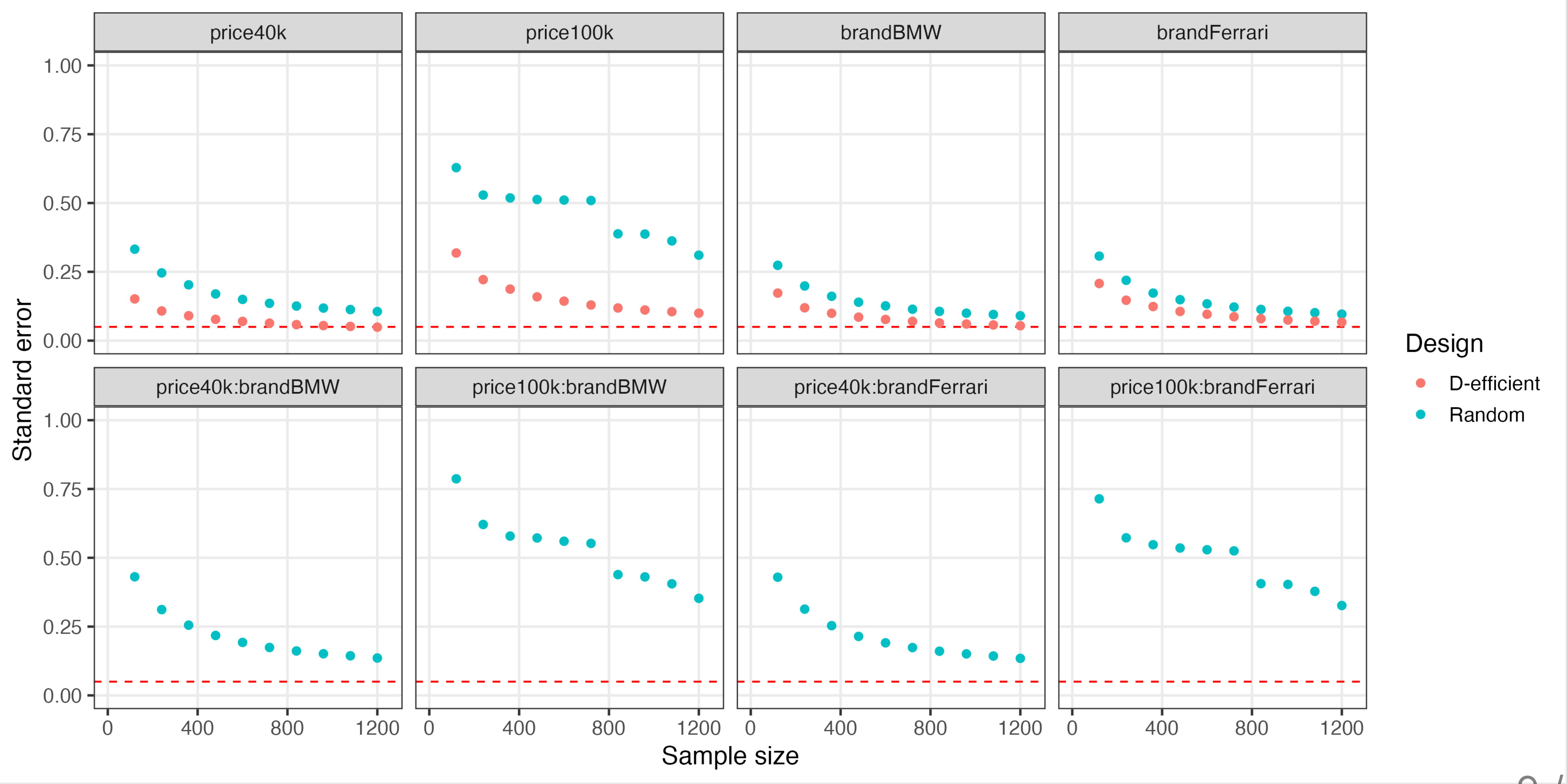
```
Pairwise attribute counts:

brand & price:

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

Bayesian D-efficient designs

Attempts to maximize information on Main Effects



...but interaction effects are confounded in D-efficient designs



But what about other factors?

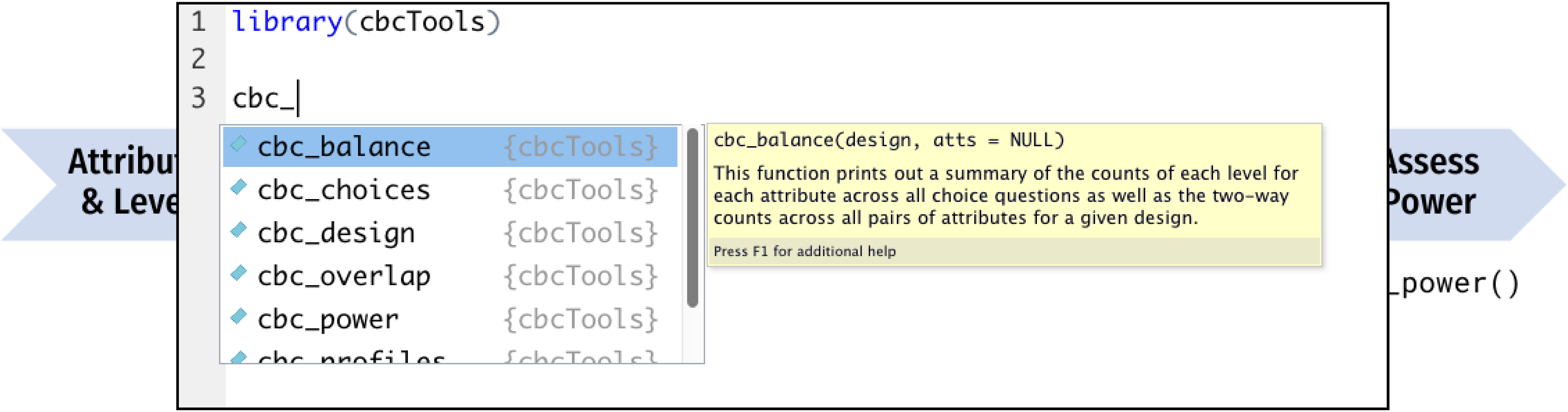
- What if I add one more choice question to each respondent?
- What if I increase the number of alternatives per choice question?
- What if I use a labeled design (aka "alternative-specific design")?
- What if there are interaction effects?

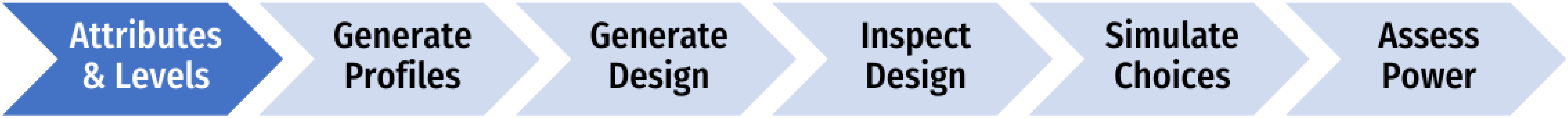
The `cbcTools` Package


Attributes & Levels
Generate Profiles
Generate Design
Inspect Design
Simulate Choices
Assess Power

`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()`

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"
```

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

`cbc_overlap()`

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
```

```r
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
```

Attribute-specific levels

```r
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")
    )
  )
)
```

Generate restricted set of profiles

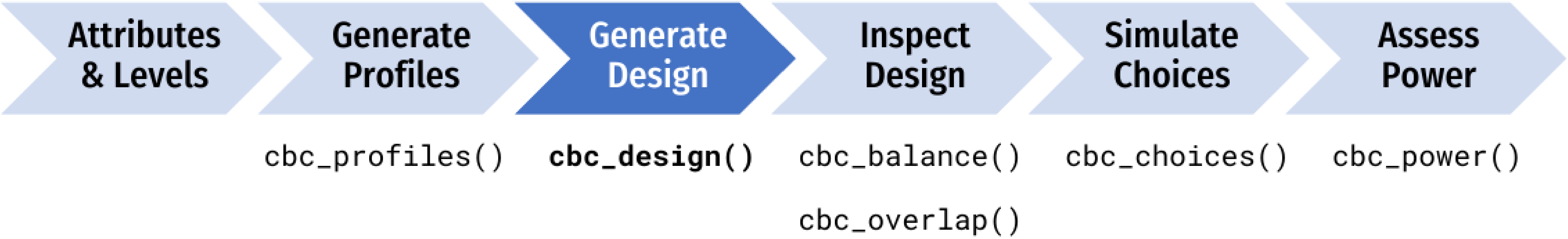
```
profiles <- cbc_profiles(levels)
```

```
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
```


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

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)
```

```
#>   respID qID altID obsID profileID price       type freshness
#> 1      1   1     1     1        16   1.5 Honeycrisp Excellent
#> 2      1   1     2     1        23   1.5       Fuji   Average
#> 3      1   1     3     1        44   1.5       Fuji      Poor
#> 4      1   2     1     2        18   2.5 Honeycrisp Excellent
#> 5      1   2     2     2        14   4.0       Gala Excellent
#> 6      1   2     3     2         8   1.0       Gala Excellent
```

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)
```

```
#>       respID qID altID obsID profileID price type_Fuji type_Gala type_Honeycrisp freshn
#> 1          1   1     1     1        37   1.5         0         0               1
#> 2          1   1     2     1        50   1.0         0         1               0
#> 3          1   1     3     1        21   4.0         0         0               1
#> 11000      1   1     4     1         0   0.0         0         0               0
#> 4          1   2     1     2         3   2.0         1         0               0
#> 5          1   2     2     2        28   4.0         1         0               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)
```

```
#>   respID qID altID obsID profileID price       type freshness
#> 1      1   1     1     1        22   1.0       Fuji   Average
#> 2      1   1     2     1        10   2.0       Gala Excellent
#> 3      1   1     3     1        58   1.5 Honeycrisp      Poor
#> 4      1   2     1     2        49   4.0       Fuji      Poor
#> 5      1   2     2     2        32   2.5       Gala   Average
#> 6      1   2     3     2        59   2.0 Honeycrisp      Poor
```

Make a Bayesian D-efficient design

(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)
  )
)
```

Make a Bayesian D-efficient design

(coming soon!)

- Check out the `idefix` package
- Import a design: Sawtooth → csv → R

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

`cbc_overlap()`

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
```

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
```

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

`cbc_overlap()`

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        22   1.0       Fuji   Average      1
#> 2      1   1     2     1        10   2.0       Gala Excellent      0
#> 3      1   1     3     1        58   1.5 Honeycrisp      Poor      0
#> 4      1   2     1     2        49   4.0       Fuji      Poor      0
#> 5      1   2     2     2        32   2.5       Gala   Average      1
#> 6      1   2     3     2        59   2.0 Honeycrisp      Poor      0
```

Simulate choices according to a prior

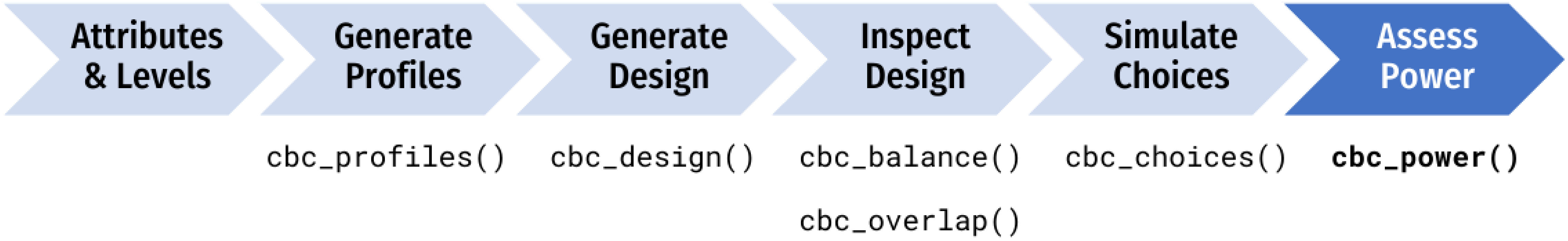
```
data <- cbc_choices(
  design = design,
  obsID = "obsID",
  priors = list(
    price     = -0.1,
    type      = c(0.1, 0.2),
    freshness = c(0.1, -0.2)
  )
)
```

Attribute	Level	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

```r
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


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

Conduct a power analysis

```r
power <- cbc_power(
    nbreaks = 10,
    n_q     = 3,
    data    = data,
    pars    = c("price", "type", "freshness"),
    outcome = "choice",
    obsID   = "obsID"
)
```

```r
head(power)
```

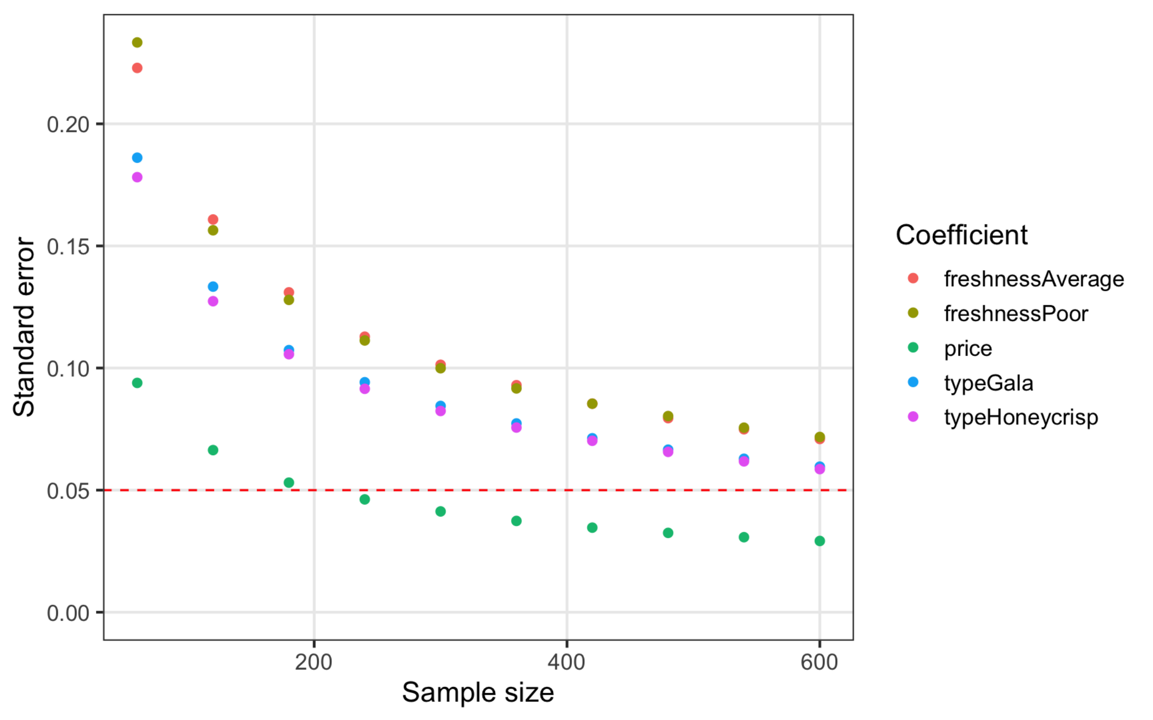
```
#>   sampleSize              coef          es
#> 1         60             price -0.00751708
#> 2         60          typeGala -0.19045129
#> 3         60   typeHoneycrisp -0.02352382
#> 4         60 freshnessAverage -0.05873052
#> 5         60     freshnessPoor -0.25428352
#> 6        120             price -0.07826151
```

```r
tail(power)
```

```
#>    sampleSize              coef          es
#> 45        540     freshnessPoor -0.1263108
#> 46        600             price -0.0854275
#> 47        600          typeGala  0.1563946
#> 48        600    typeHoneycrisp  0.2336239
#> 49        600  freshnessAverage  0.0884197
#> 50        600     freshnessPoor -0.1156453
```

Conduct a power analysis

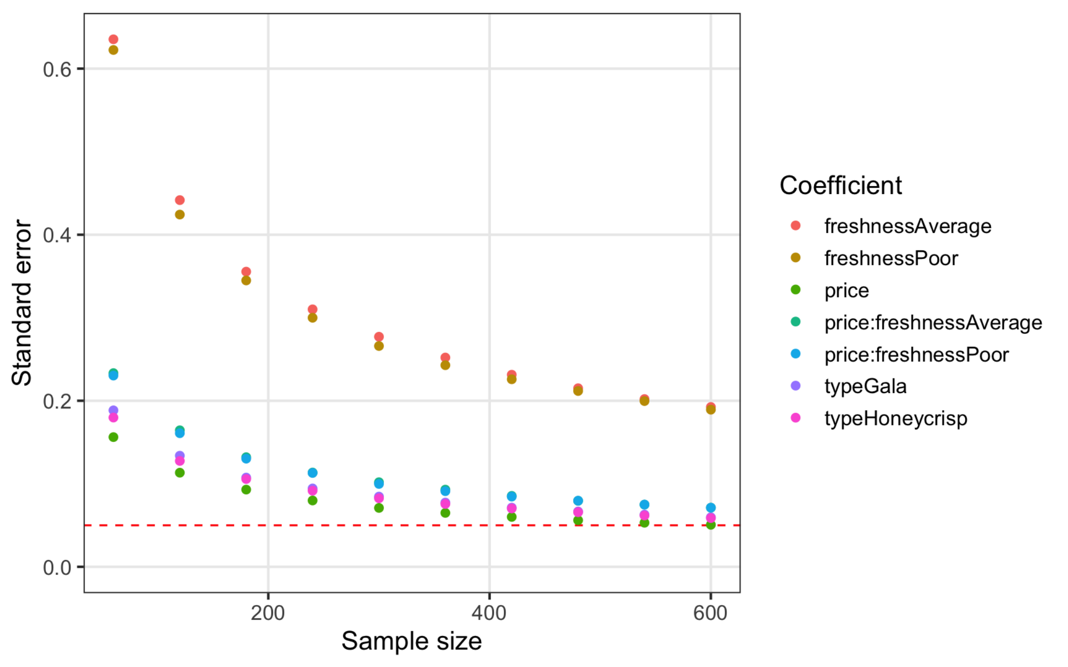
```
plot(power)
```



Conduct a power analysis

```
power_int <- cbc_power(
    nbreaks = 10,
    n_q     = 3,
    data    = data,
    pars    = c(
      "price",
      "type",
      "freshness",
      "price*freshness"
    ),
    outcome = "choice",
    obsID   = "obsID"
)
```

```
plot(power_int)
```

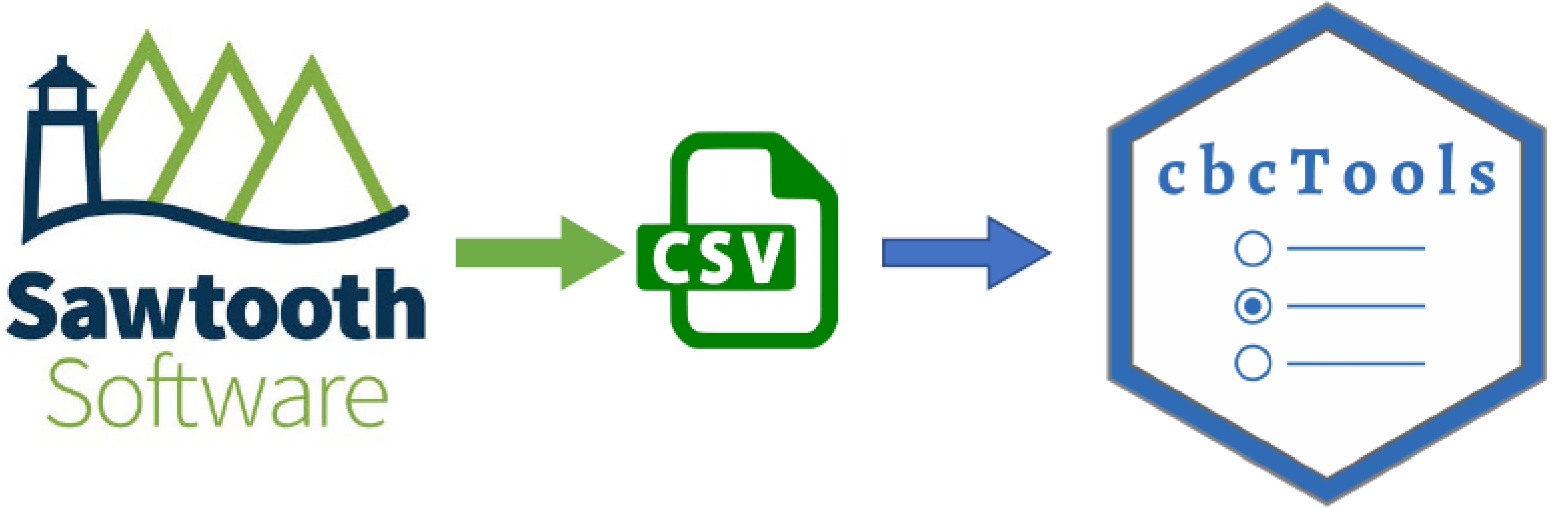


cbcTools

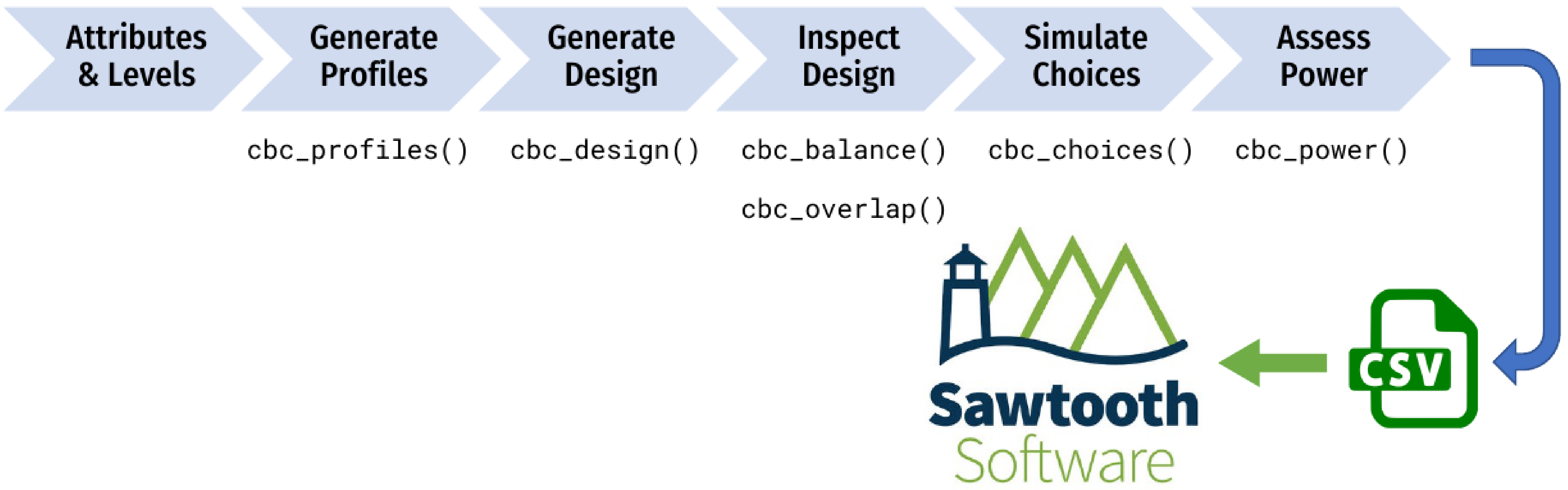
Attributes & Levels → Generate Profiles → Generate Design → Inspect Design → Simulate Choices → Assess Power

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

`cbc_overlap()`



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



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

Thanks!

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

Slides: https://jhelvy.github.io/2022-sawtooth-conf

@johnhelveston
@jhelvy
@jhelvy
jhelvy.com
jph@gwu.edu

Extra slides