

# The cbcTools Package

Tools for Designing and Testing  
Choice-Based Conjoint Surveys in 

by John Paul Helveston

Sawtooth Software Conference

May 06, 2022



# Designing a Choice-Based Conjoint Survey is Hard

# Designing a Choice-Based Conjoint Survey is Hard

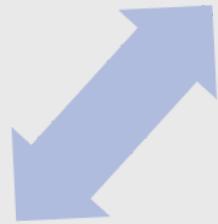
## Design Parameters

- 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

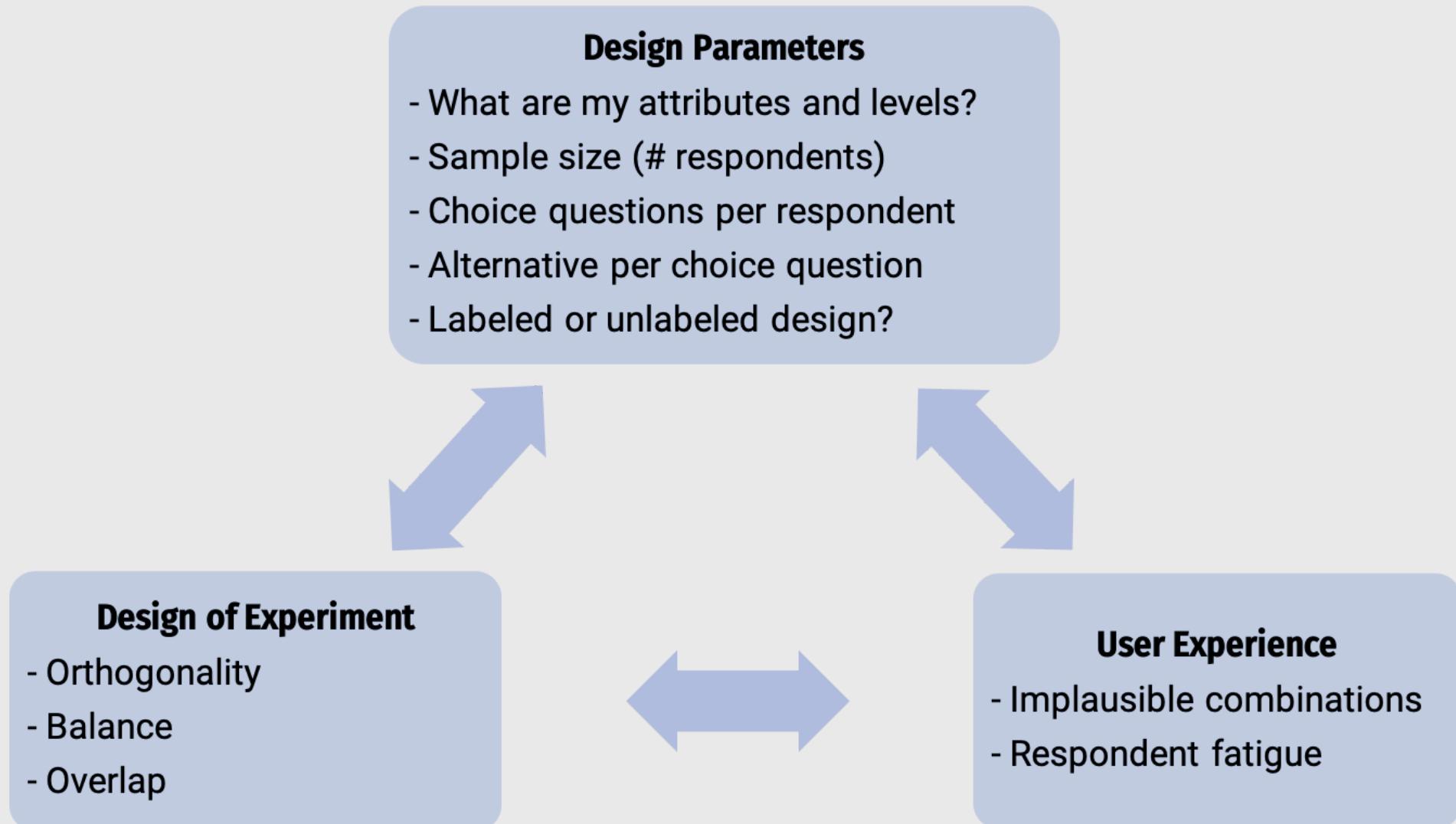
- What are my attributes and levels?
- Sample size (# respondents)
- Choice questions per respondent
- Alternative per choice question
- Labeled or unlabeled design?



## Design of Experiment

- Orthogonality
- Balance
- Overlap

# Designing a Choice-Based Conjoint Survey is Hard



# A simple conjoint experiment about cars

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

**Design: 9 choice sets, 3 alternatives each**

# A simple conjoint experiment about cars

Attribute	Levels
Brand	GM, 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

# A simple conjoint experiment about cars

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

**Design: 90 choice sets, 3 alternatives each**

# A simple conjoint experiment about cars

Attribute	Levels
Brand	GM, 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

# A simple conjoint experiment about cars

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

# Bayesian D-efficient designs

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

# Bayesian D-efficient designs

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:

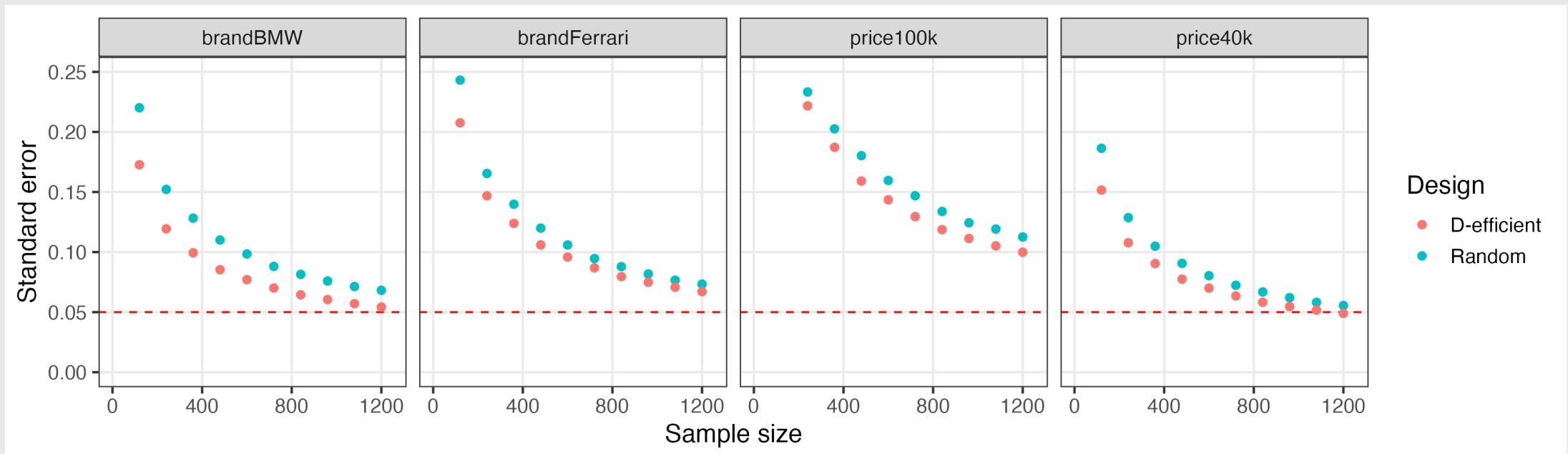
	20k	40k	100k
GM	52	41	0
BMW	30	30	30
Ferrari	15	22	49

# Bayesian D-efficient designs

Attempts to maximize information on **Main Effects**

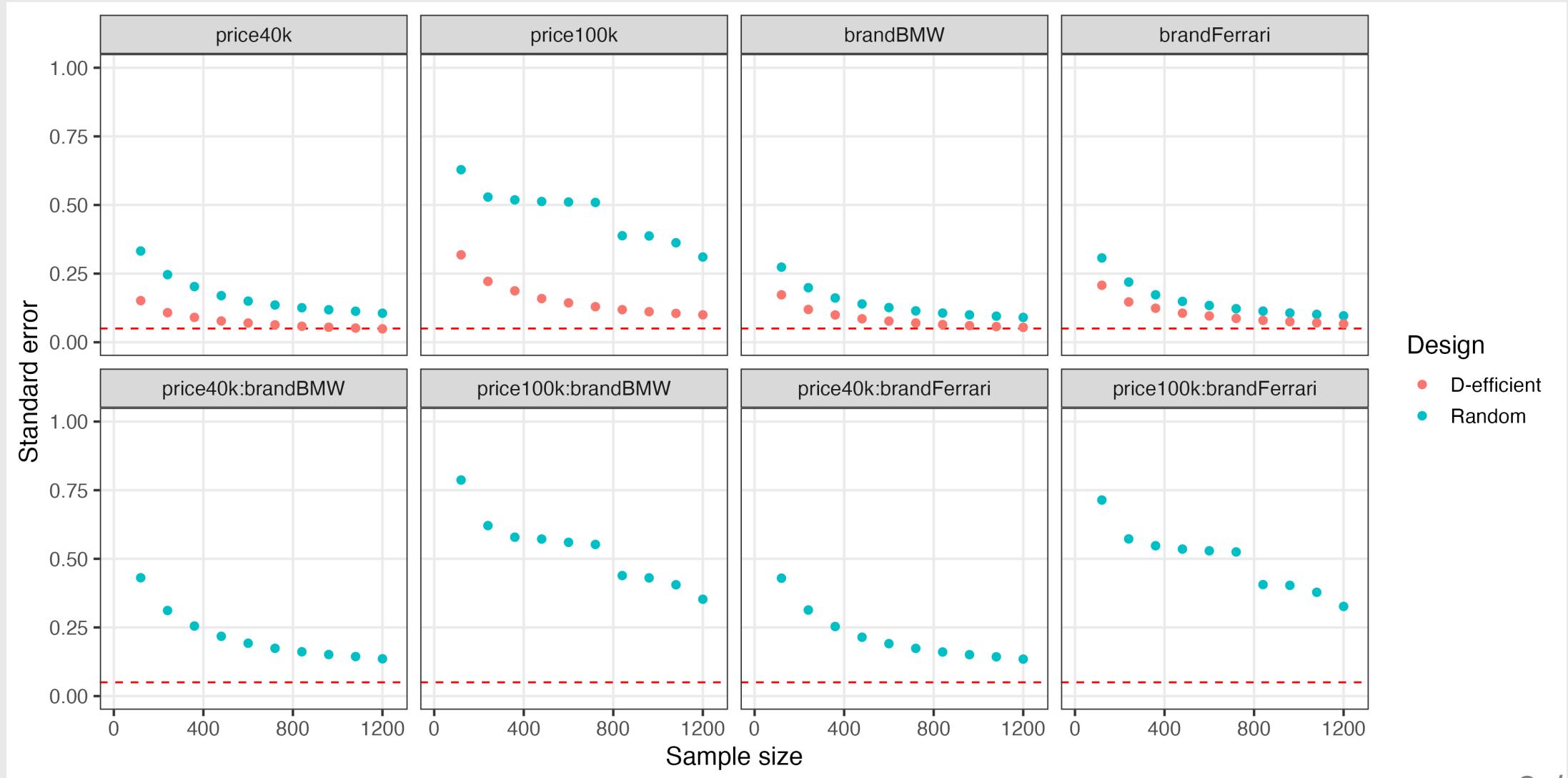
# Bayesian D-efficient designs

Attempts to maximize information on **Main Effects**



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

# ...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?

# 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?

# 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")?

# 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**

**Attributes  
& Levels**

**Generate  
Profiles**

**Generate  
Design**

**Inspect  
Design**

**Simulate  
Choices**

**Assess  
Power**

`cbc_profiles()`

`cbc_design()`

`cbc_balance()`  
`cbc_overlap()`

`cbc_choices()`

`cbc_power()`

## Attribut & Level

```
1 library(cbcTools)  
2  
3 cbc_|
```

- ◆ `cbc_balance` {cbcTools}
- ◆ `cbc_choices` {cbcTools}
- ◆ `cbc_design` {cbcTools}
- ◆ `cbc_overlap` {cbcTools}
- ◆ `cbc_power` {cbcTools}
- ◆ `cbc_profiles` {cbcTools}

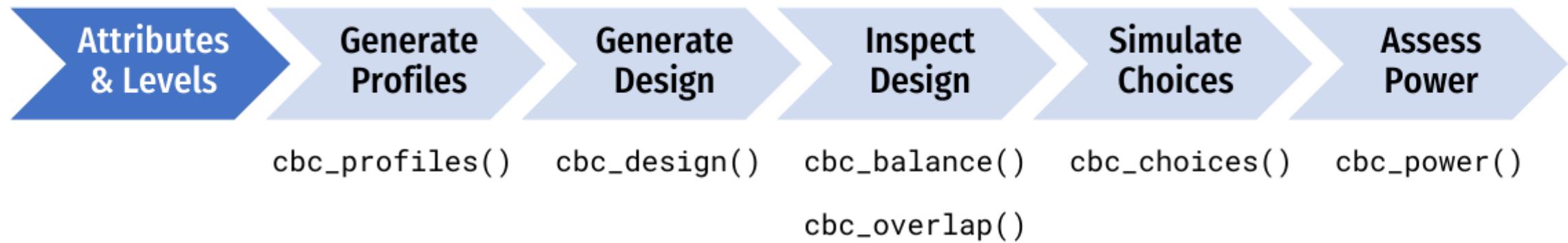
`cbc_balance(design, attrs = NULL)`

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()`



# Define the attributes and levels

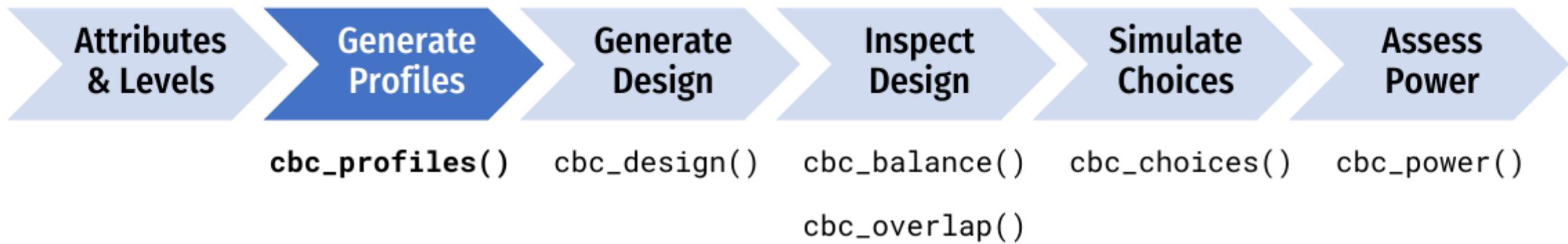
```
levels <- list(  
  price      = c(1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00), # $ per pound  
  type       = c("Fuji", "Gala", "Honeycrisp"),  
  freshness = c("Excellent", "Average", "Poor"))
```

# 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"
```



# Generate all possible profiles

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

# 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
```

# 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")
    )
  )
)
```

# Generate restricted set of profiles

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

# Generate restricted set of profiles

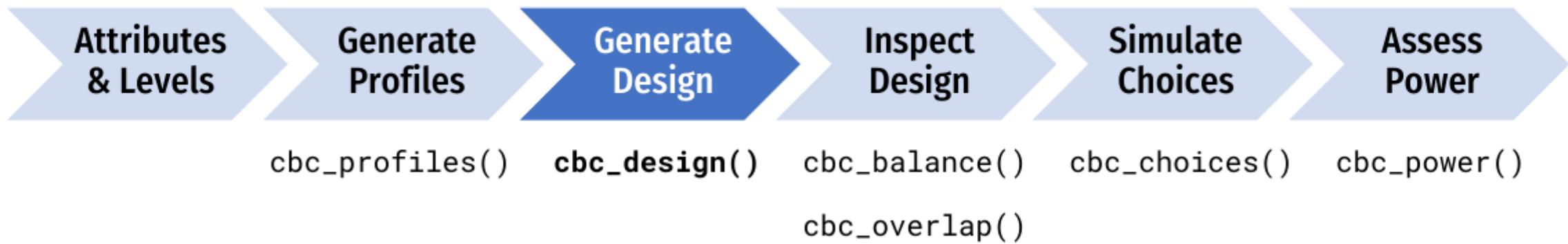
```
profiles <- cbc_profiles(levels)
```

```
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
```



# 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  
)
```

# 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        39    2.5 Honeycrisp Average  
#> 3     1   1     3     1        37    1.5 Honeycrisp Average  
#> 4     1   2     1     2        58    1.5 Honeycrisp      Poor  
#> 5     1   2     2     2        3    2.0    Fuji Excellent  
#> 6     1   2     3     2        38    2.0 Honeycrisp Average
```

# 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  
)
```

# 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 type_Honeycrisp freshness_  
#> 1     1   1     1    1        3    2.0         1         0            0  
#> 2     1   1     2    1       38    2.0         0         0            1  
#> 3     1   1     3    1       19    3.0         0         0            1  
#> 4     1   1     4    1        0    0.0         0         0            0  
#> 5     1   2     1    2       30    1.5         0         1            0  
#> 6     1   2     2    2       53    2.5         0         1            0
```

# Make a labeled design

(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"  
)
```

# Make a labeled design

(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           3    2.0    Fuji Excellent  
#> 2     1    1     2    1          53    2.5    Gala    Poor  
#> 3     1    1     3    1          38    2.0 Honeycrisp Average  
#> 4     1    2     1    2           3    2.0    Fuji Excellent  
#> 5     1    2     2    2          50    1.0    Gala    Poor  
#> 6     1    2     3    2          42    4.0 Honeycrisp Average
```

# 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

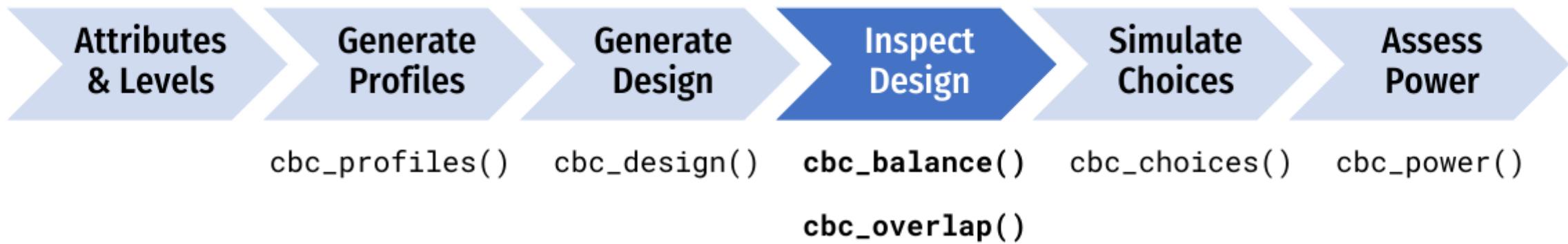
(coming soon!)

- Check out the `idefix` package

# Make a Bayesian D-efficient design

(coming soon!)

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



# Check design **balance**

```
cbc_balance(design)
```

# 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

# 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)
```

# Check design **overlap**

```
cbc_overlap(design)
```

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

price:

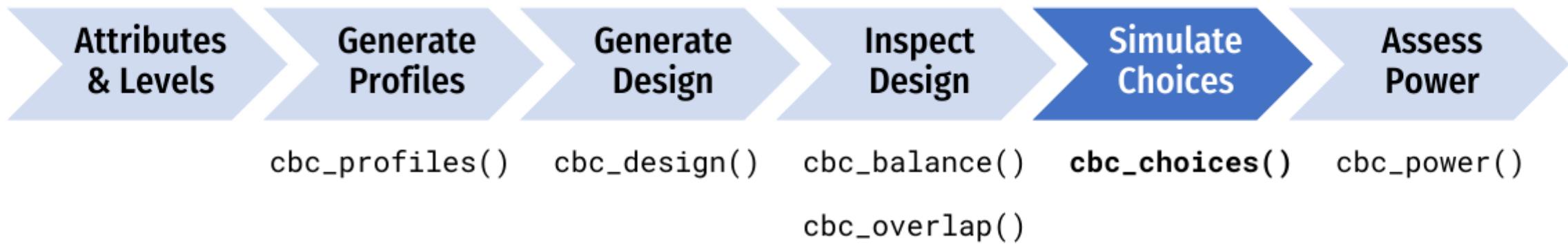
1	2	3
31	630	1139

type:

1	2	3
156	1248	396

freshness:

1	2	3
175	1189	436



# Simulate random choices

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

# 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           3    2.0    Fuji Excellent  0  
#> 2     1    1     2     1          53    2.5    Gala    Poor  0  
#> 3     1    1     3     1          38    2.0 Honeycrisp Average  1  
#> 4     1    2     1     2           3    2.0    Fuji Excellent  1  
#> 5     1    2     2     2          50    1.0    Gala    Poor  0  
#> 6     1    2     3     2          42    4.0 Honeycrisp Average  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
<b>Price</b>	Continuous	-0.1
<b>Type</b>	Fuji	0
	Gala	0.1
	Honeycrisp	0.2
<b>Freshness</b>	Average	0
	Excellent	0.1
	Poor	-0.2

# 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	Utility
<b>Price</b>	Continuous	-0.1
<b>Type</b>	Fuji	0
	Gala	N(0.1, 0.5)
<b>Freshness</b>	Honeycrisp	N(0.2, 1)
	Average	0
	Excellent	0.1
	Poor	-0.2

# 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	Utility
<b>Price</b>	Continuous	-0.1
<b>Type</b>	Fuji	0
	Gala	0.1
	Honeycrisp	0.2
<b>Freshness</b>	Average	0
	Excellent	0.1
	Poor	-0.2
<b>Price x Type</b>	Fuji	0
	Gala	0.1
	Honeycrisp	0.5

**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

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

# Conduct a power analysis

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

```
head(power)
```

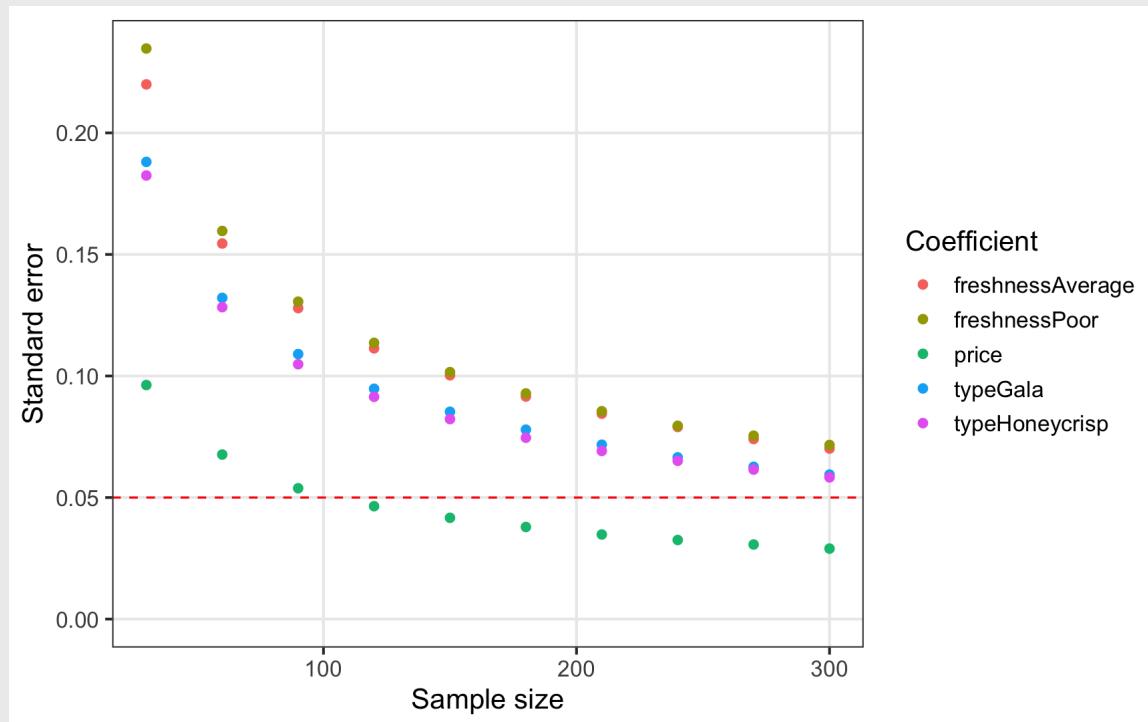
```
#>   sampleSize          coef       est  
#> 1        30      price -0.18558034  
#> 2        30    typeGala -0.11287630  
#> 3        30  typeHoneycrisp -0.00373311  
#> 4        30 freshnessAverage -0.23740384  
#> 5        30  freshnessPoor -0.58571733  
#> 6        60      price -0.13680799
```

```
tail(power)
```

```
#>   sampleSize          coef       es  
#> 45        270  freshnessPoor -0.1834163  
#> 46        300      price -0.1105467  
#> 47        300    typeGala  0.0954400  
#> 48        300  typeHoneycrisp  0.1836931  
#> 49        300 freshnessAverage  0.1352345  
#> 50        300  freshnessPoor -0.1942773
```

# Conduct a power analysis

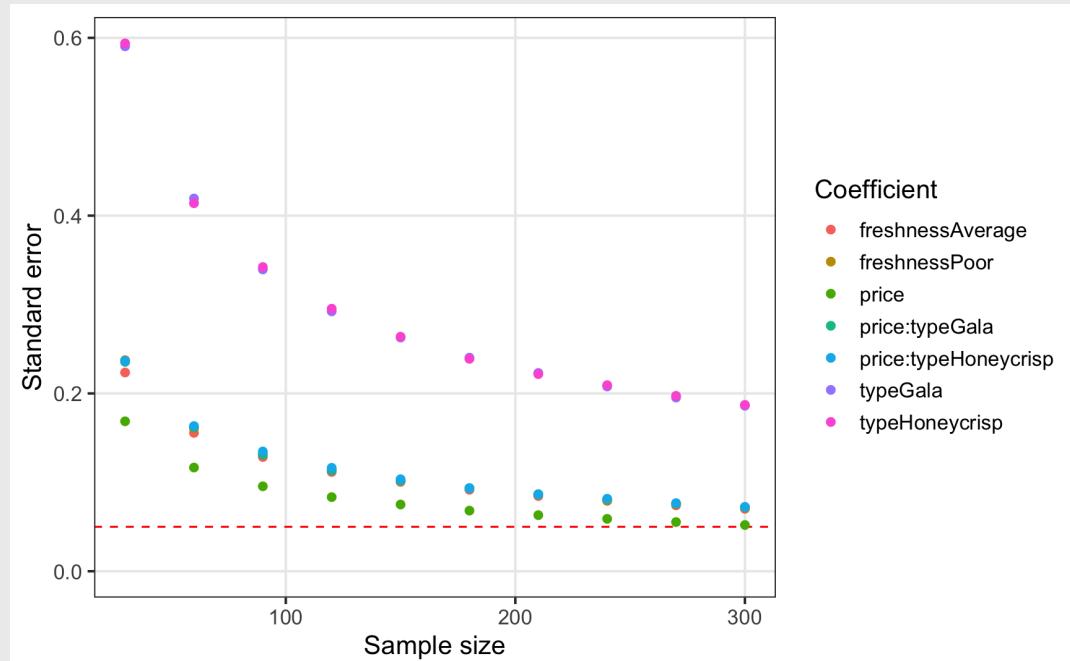
```
plot(power)
```



# Conduct a power analysis

```
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)
```





**Attributes  
& Levels**

**Generate  
Profiles**

**Generate  
Design**

**Inspect  
Design**

**Simulate  
Choices**

**Assess  
Power**

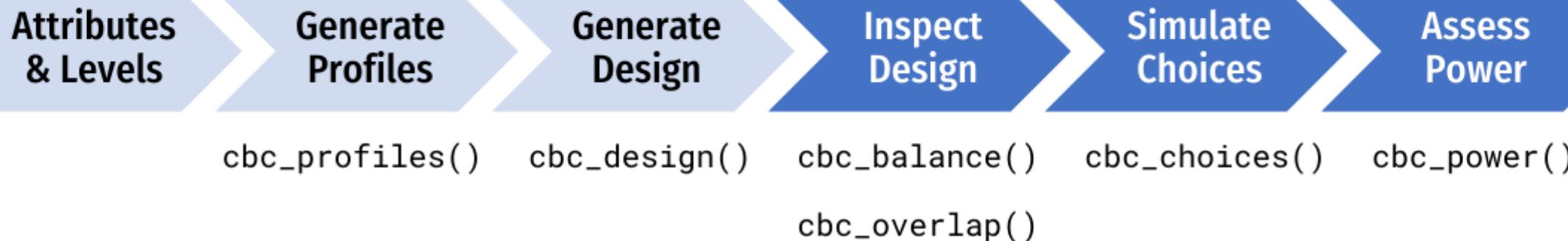
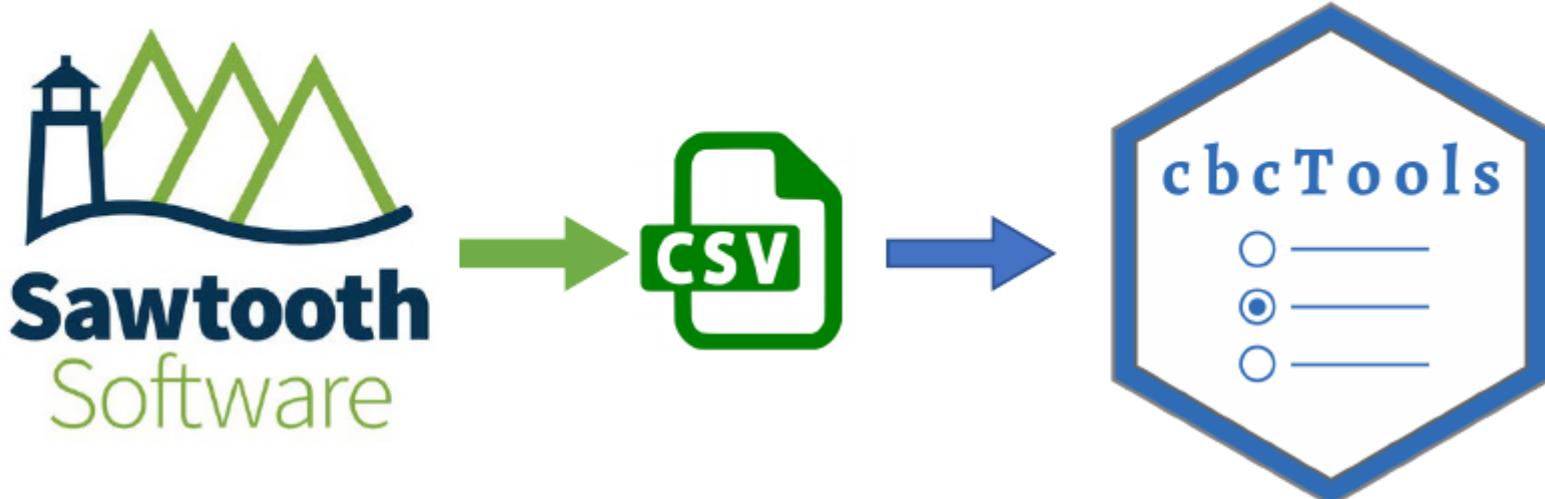
`cbc_profiles()`

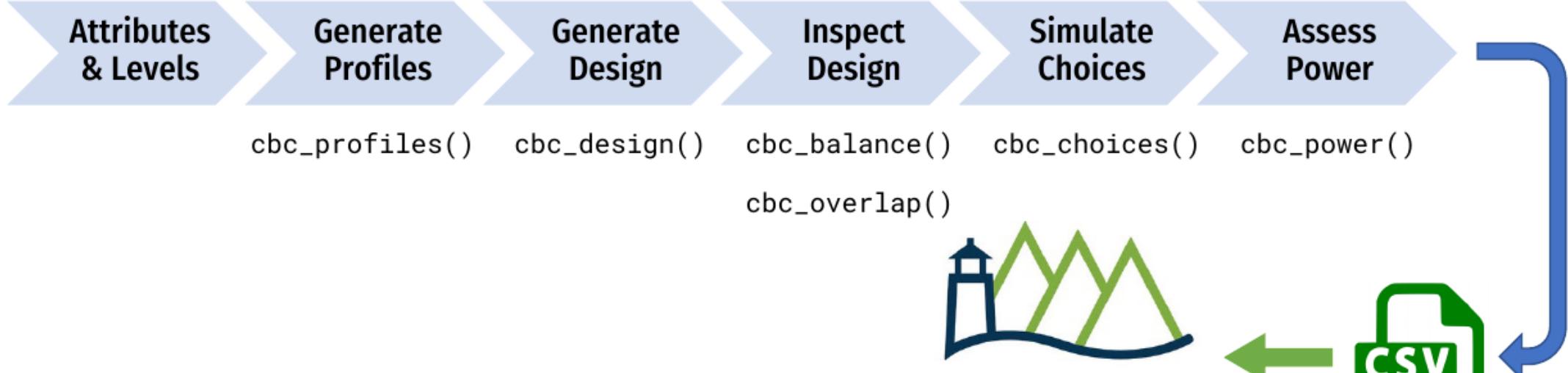
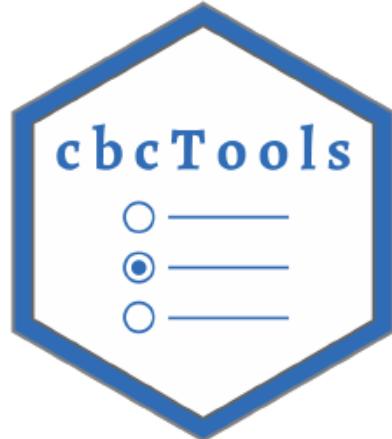
`cbc_design()`

`cbc_balance()`  
`cbc_overlap()`

`cbc_choices()`

`cbc_power()`





# 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