


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

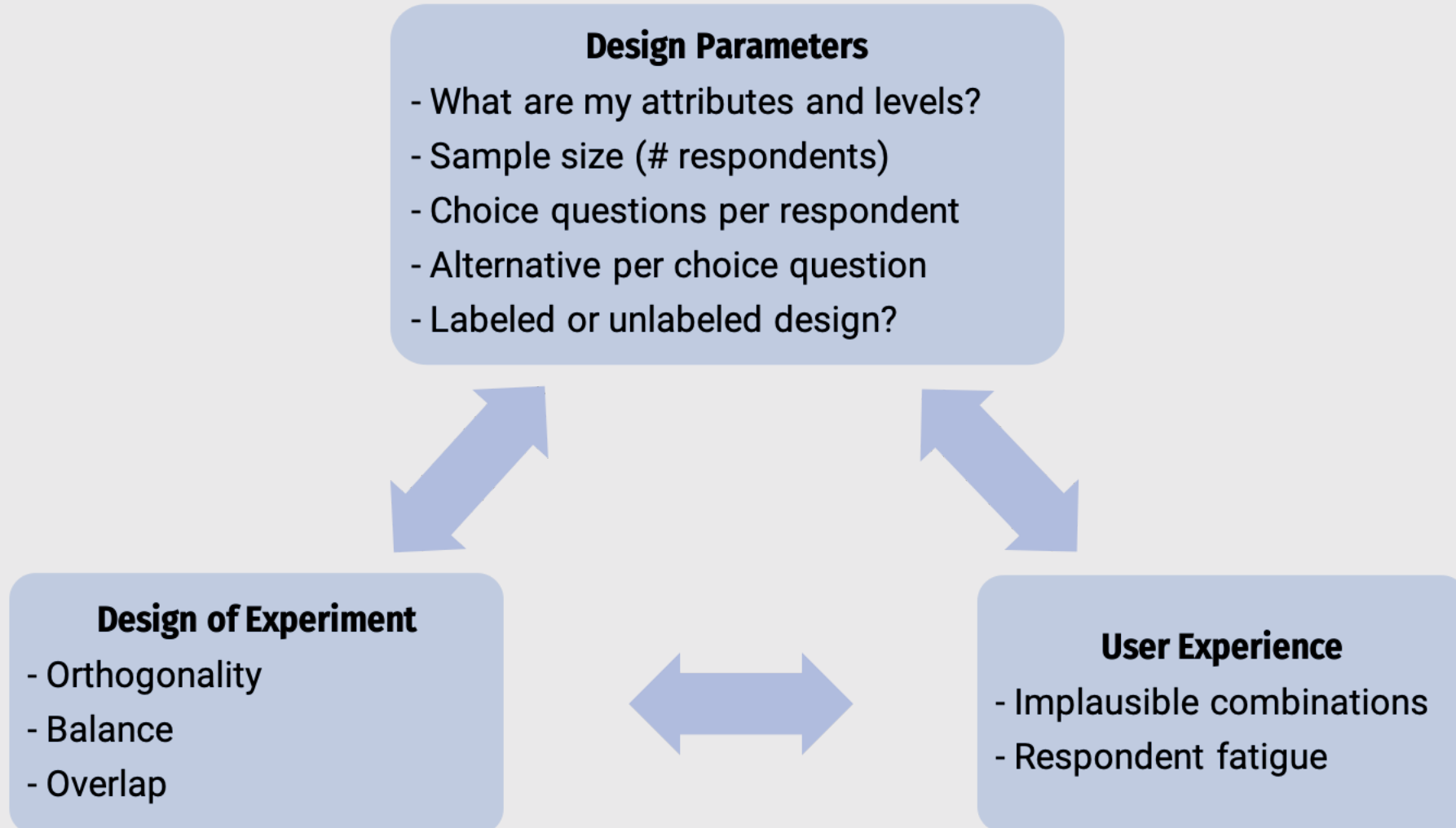
- What are my attributes and levels?
- Sample size (# respondents)
- Choice questions per respondent
- Alternative per choice question
- Labeled or unlabeled design?



Design of Experiment

- Orthogonality
- Balance
- Overlap

Designing a Choice-Based Conjoint Survey is Hard



Many R packages for design of experiment

- {cbcTools}
- {ExpertChoice}
- {support.CEs}
- {idefix}
- {choiceDes}

Many R packages for design of experiment

- {cbcTools} ← Does a lot more than just DOE!
- {ExpertChoice}
- {support.CEs}
- {idefix}
- {choiceDes}

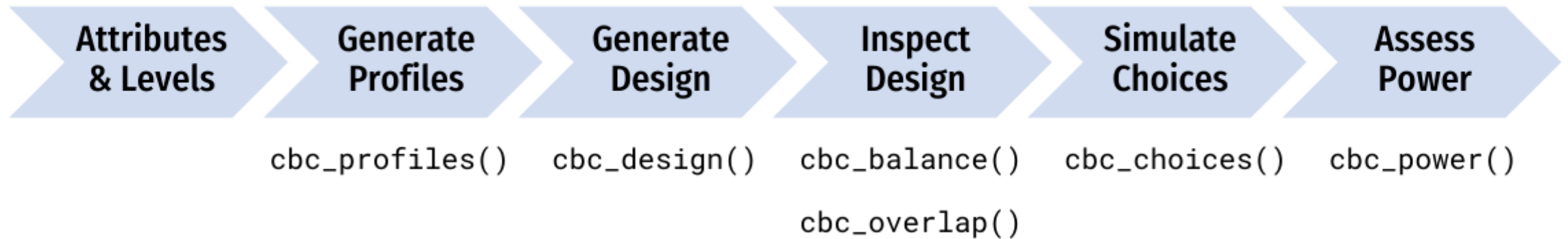
A systematic workflow for designing a CBC experiment



A systematic workflow for designing a CBC experiment



A systematic workflow for designing a CBC experiment



Attribu
& 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}

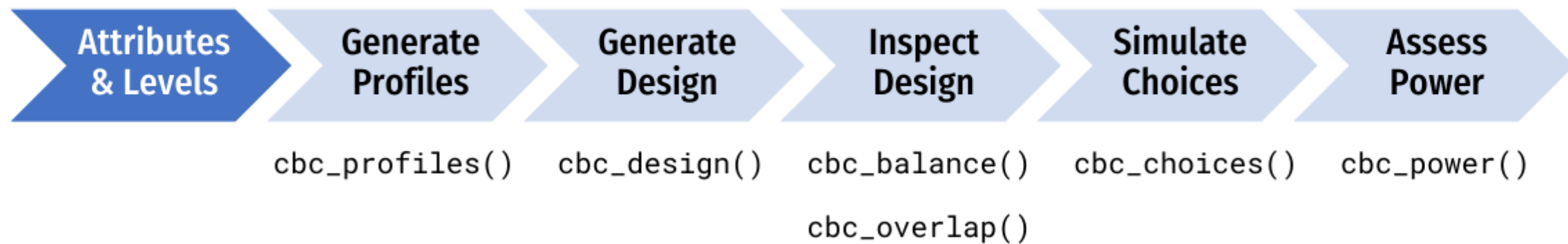
cbc_balance(design, atts = NULL)

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()



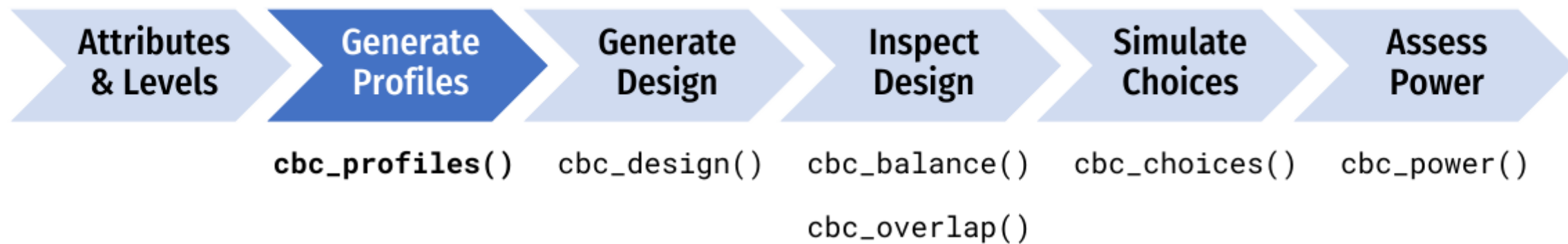
Example CBC question about apples

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



Generate all possible profiles

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

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
```

tail(profiles)

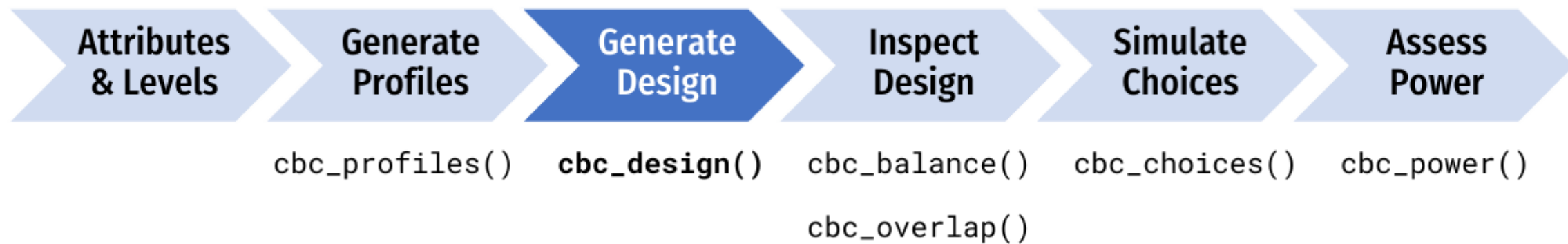
```
#>   profileID price      type freshness  
#> 58         58  1.5 Honeycrisp Excellent  
#> 59         59  2.0 Honeycrisp Excellent  
#> 60         60  2.5 Honeycrisp Excellent  
#> 61         61  3.0 Honeycrisp Excellent  
#> 62         62  3.5 Honeycrisp Excellent  
#> 63         63  4.0 Honeycrisp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

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)
```

```
#>   profileID respID qID altID obsID price      type freshness  
#> 1         55     1  1     1     1  3.5      Gala  Excellent  
#> 2         15     1  1     2     1  1.0 Honeycrisp    Poor  
#> 3         16     1  1     3     1  1.5 Honeycrisp    Poor  
#> 4          6     1  2     1     2  3.5       Fuji    Poor  
#> 5         35     1  2     2     2  4.0       Gala  Average  
#> 6         57     1  2     3     2  1.0 Honeycrisp  Excellent
```

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)
```

```
#>   profileID respID qID altID obsID price type_Fuji type_Gala type_Honeycrisp freshness_  
#> 1         33      1  1    1      1   3.0         0         1             0  
#> 2         62      1  1    2      1   3.5         0         0             1  
#> 3         48      1  1    3      1   3.5         1         0             0  
#> 4          0      1  1    4      1   0.0         0         0             0  
#> 5         15      1  2    1      2   1.0         0         0             1  
#> 6         23      1  2    2      2   1.5         1         0             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"  
)
```

```
head(design)
```

```
#>   profileID respID qID altID obsID price      type freshness  
#> 1         26      1  1     1      1     3      Fuji   Average  
#> 2         31      1  1     2      1     2       Gala   Average  
#> 3         19      1  1     3      1     3 Honeycrisp    Poor  
#> 4          1      1  2     1      2     1      Fuji    Poor  
#> 5          8      1  2     2      2     1       Gala    Poor  
#> 6         57      1  2     3      2     1 Honeycrisp Excellent
```

Make a Bayesian D-efficient design

(Uses the `idefix` package to generate a 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  
  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:

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

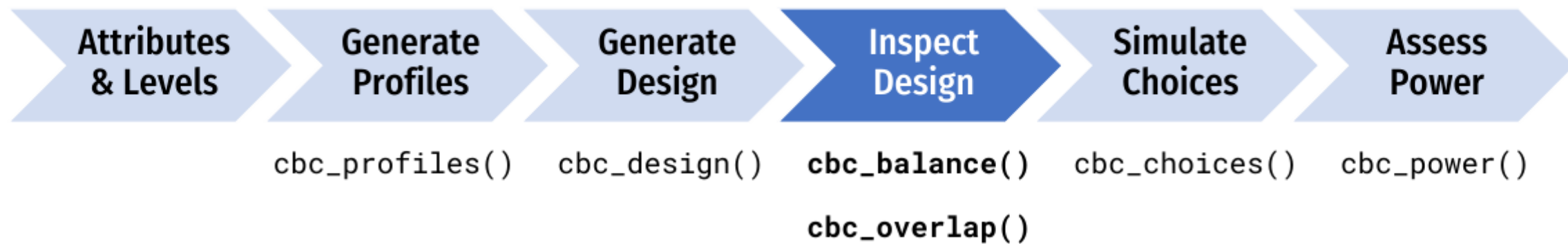
Import a design: Sawtooth → →

```
library(readr)

design <- read_csv('design.csv')

head(design)
```

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



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**

```
cbc_overlap(design)
```

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

```
price:
```

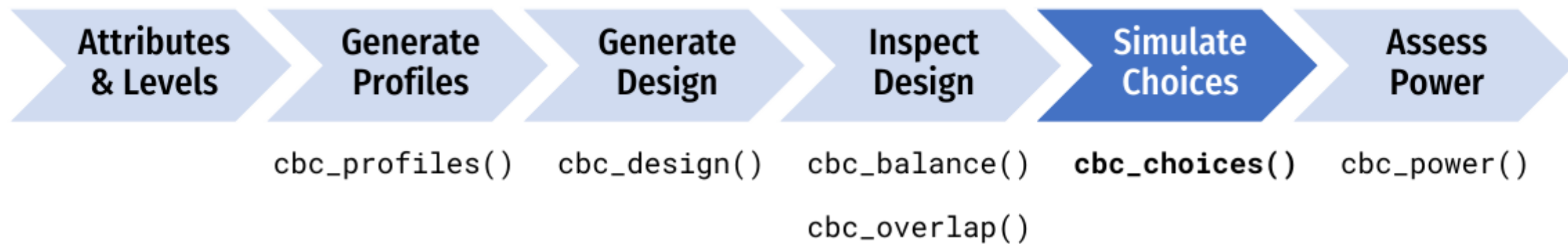
1	2	3
31	630	1139

```
type:
```

1	2	3
156	1248	396

```
freshness:
```

1	2	3
175	1189	436



Simulate random choices

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

```
head(data)
```

```
#>   profileID respID qID altID obsID price      type freshness choice  
#> 1         26      1  1    1      1     3      Fuji   Average      0  
#> 2         31      1  1    2      1     2      Gala   Average      1  
#> 3         19      1  1    3      1     3 Honeycrisp   Poor      0  
#> 4          1      1  2    1      2     1      Fuji   Poor      0  
#> 5          8      1  2    2      2     1      Gala   Poor      0  
#> 6         57      1  2    3      2     1 Honeycrisp Excellent  1
```

Simulate choices according to a prior

(Fixed coefficients)

```
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)  
  )  
)
```

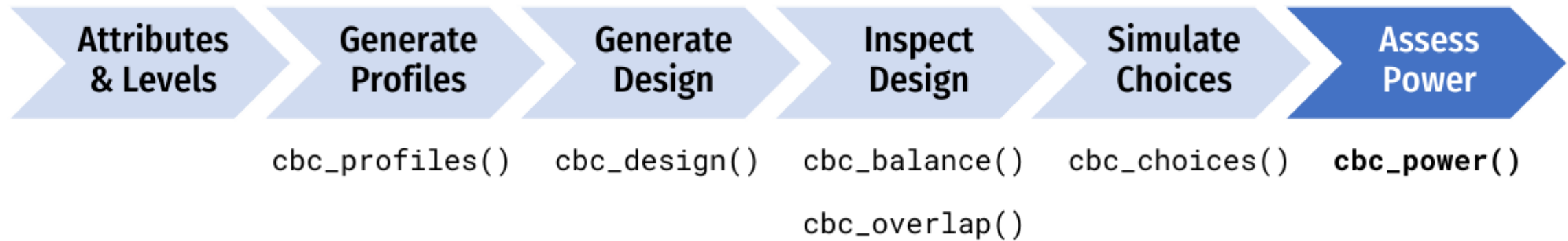
Attribute	Level	Utility
Price	Continuous	-0.1
Type	Fuji	0
	Gala	N(0.1, 0.5)
	Honeycrisp	N(0.2, 1)
Freshness	Average	0
	Excellent	0.1
	Poor	-0.2

Simulate choices according to a prior

(Models with interactions)

```
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



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)
```

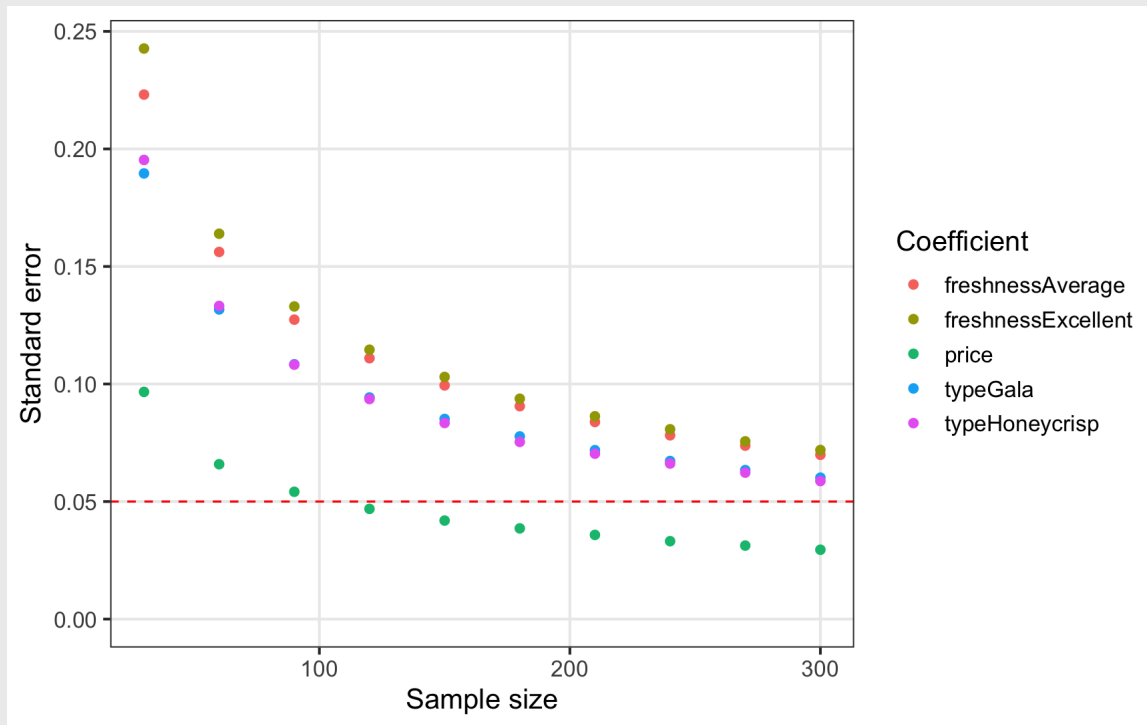
```
#>   sampleSize      coef      es  
#> 1         30      price -0.039584  
#> 2         30      typeGala  0.319803  
#> 3         30      typeHoneycrisp 0.172815  
#> 4         30      freshnessAverage 0.656496  
#> 5         30      freshnessExcellent -0.169927  
#> 6         60      price -0.118726
```

```
tail(power)
```

```
#>   sampleSize      coef      es  
#> 45         270      freshnessExcellent -0.12217  
#> 46         300      price -0.13880  
#> 47         300      typeGala  0.16386  
#> 48         300      typeHoneycrisp 0.26820  
#> 49         300      freshnessAverage 0.18943  
#> 50         300      freshnessExcellent -0.10255
```


Conduct a power analysis

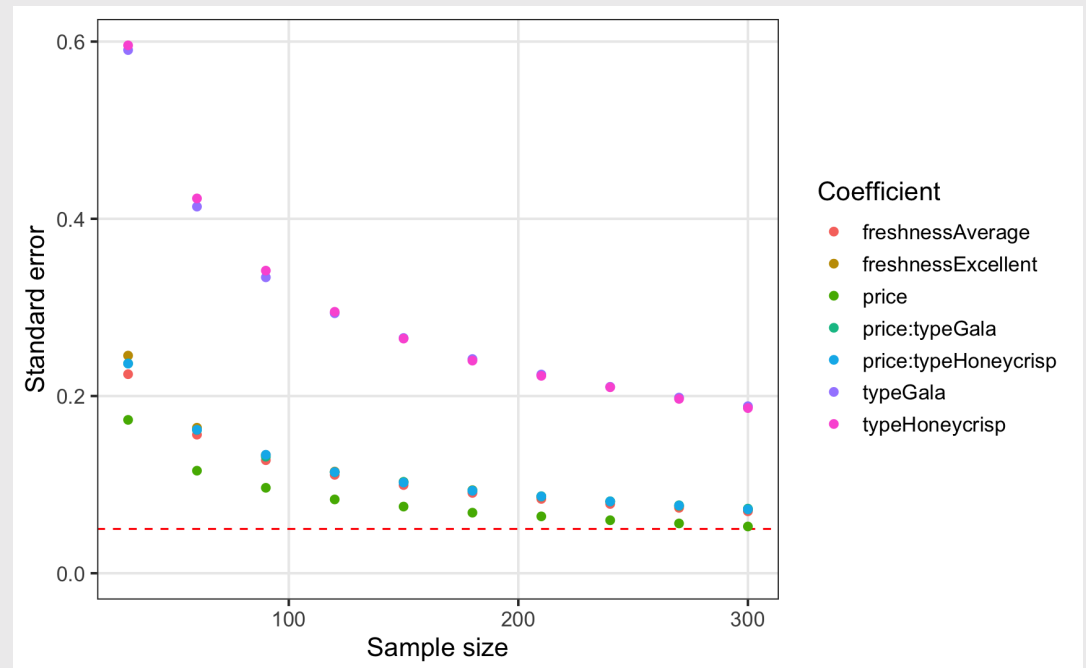
```
plot(power)
```



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_choices()`

`cbc_power()`

`cbc_overlap()`



**Attributes
& Levels**

**Generate
Profiles**

**Generate
Design**

**Inspect
Design**

**Simulate
Choices**

**Assess
Power**

`cbc_profiles()`

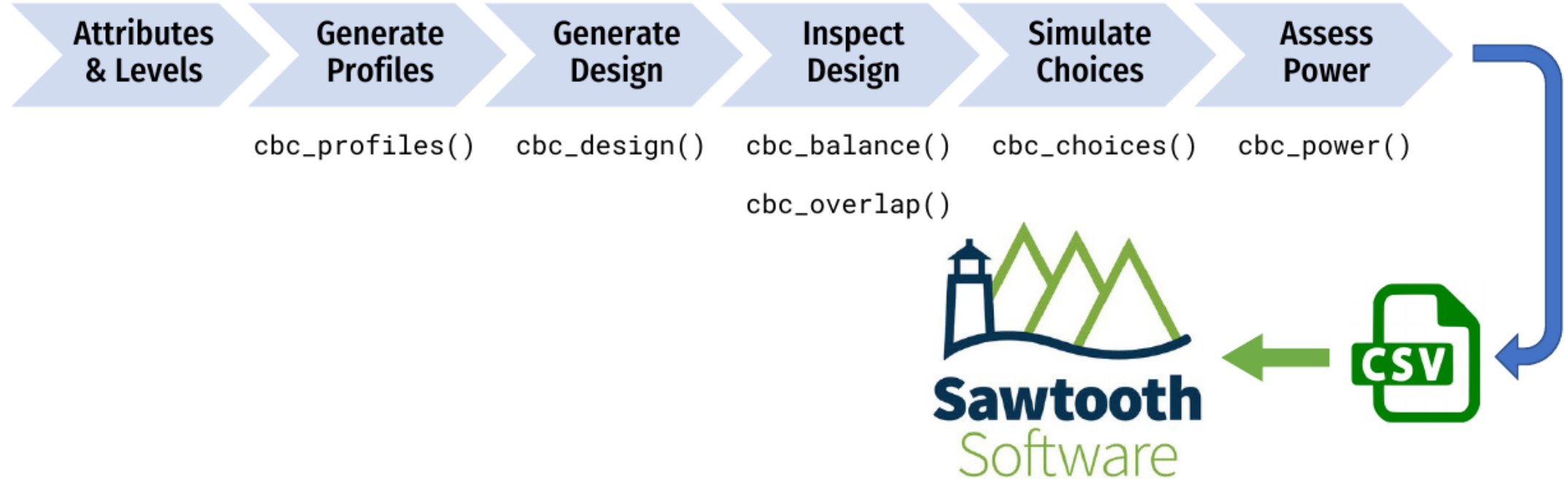
`cbc_design()`

`cbc_balance()`

`cbc_choices()`

`cbc_power()`

`cbc_overlap()`



Your turn

- Be sure to have downloaded and unzipped the [practice code](#).
- Open the `2023-qux-conf-conjoint.Rproj` file to open RStudio.
- In RStudio, open the `designing-surveys.R` file.
- Experiment with different design options, then examine the power:
 - What if you modify the questions 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 