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
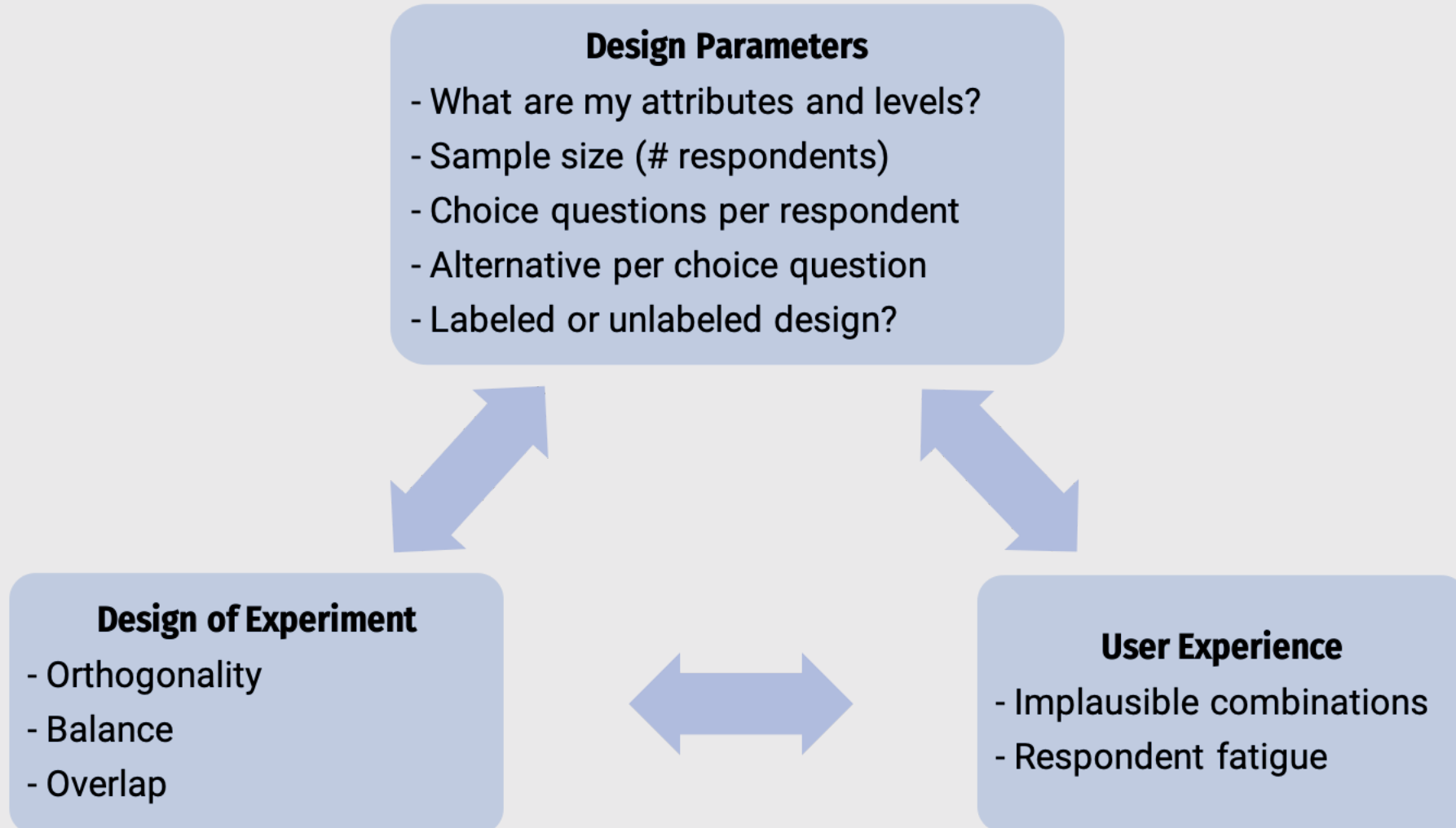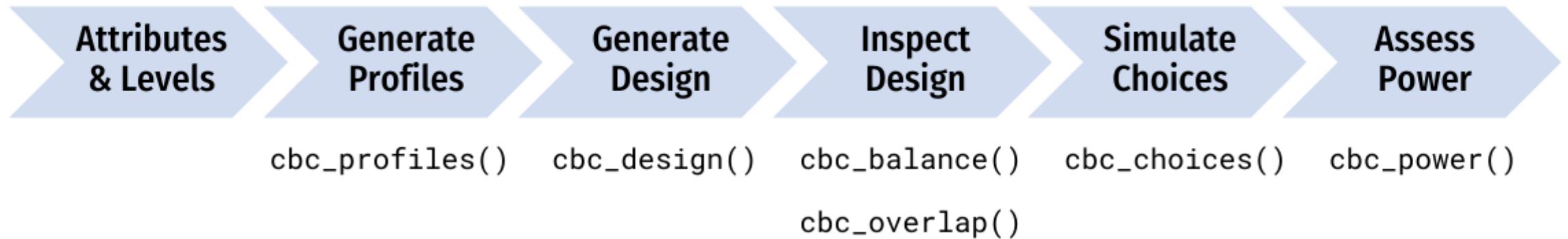
Design Parameters

- 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 systematic workflow for designing a CBC experiment

A systematic workflow for designing a CBC experiment

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

A systematic workflow for designing a CBC experiment

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()		
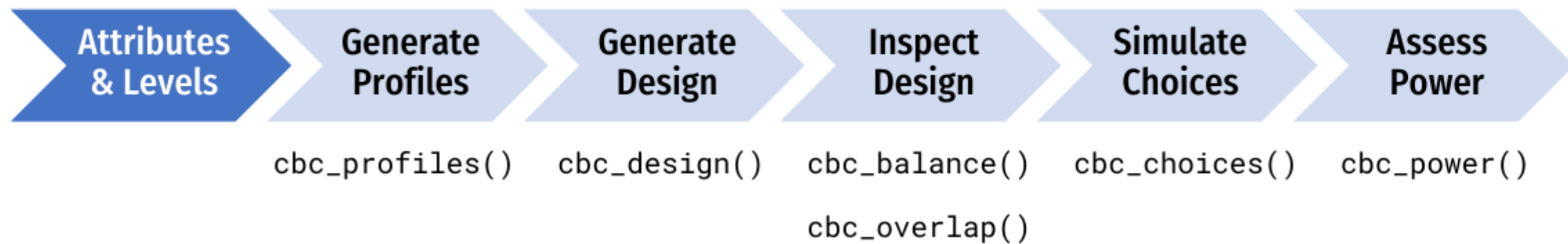
Example CBC question about apples

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

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

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

cbc_overlap()
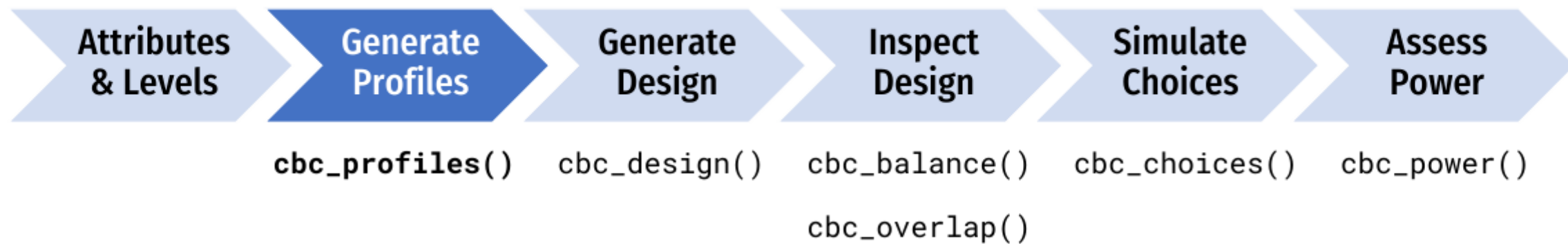
Generate all possible profiles

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

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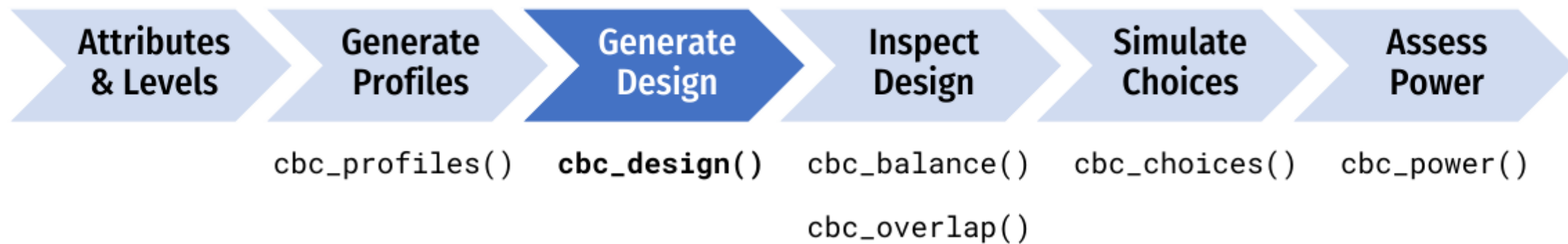
```
#>    profileID price       type freshness
#> 58        58   1.5 Honeycrisp Excellent
#> 59        59   2.0 Honeycrisp Excellent
#> 60        60   2.5 Honeycrisp Excellent
#> 61        61   3.0 Honeycrisp Excellent
#> 62        62   3.5 Honeycrisp Excellent
#> 63        63   4.0 Honeycrisp Excellent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

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).

```r
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
```

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

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

                                  cbc_overlap()
```

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        30      1   1     1     1   1.5      Gala   Average
#> 2        55      1   1     2     1   3.5      Gala Excellent
#> 3         6      1   1     3     1   3.5      Fuji      Poor
#> 4        43      1   2     1     2   1.0      Fuji Excellent
#> 5        22      1   2     2     2   1.0      Fuji   Average
#> 6        62      1   2     3     2   3.5 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         9      1   1     1     1   1.5         0         1               0
#> 2         4      1   1     2     1   2.5         1         0               0
#> 3        18      1   1     3     1   2.5         0         0               1
#> 4         0      1   1     4     1   0.0         0         0               0
#> 5        48      1   2     1     2   3.5         1         0               0
#> 6        61      1   2     2     2   3.0         0         0               1
```

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         5      1   1     1     1   3.0       Fuji      Poor
#> 2         8      1   1     2     1   1.0       Gala      Poor
#> 3        15      1   1     3     1   1.0 Honeycrisp      Poor
#> 4        46      1   2     1     2   2.5       Fuji Excellent
#> 5        10      1   2     2     2   2.0       Gala      Poor
#> 6        61      1   2     3     2   3.0 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 → [csv] → R

```r
library(readr)

design <- read_csv('design.csv')

head(design)
```
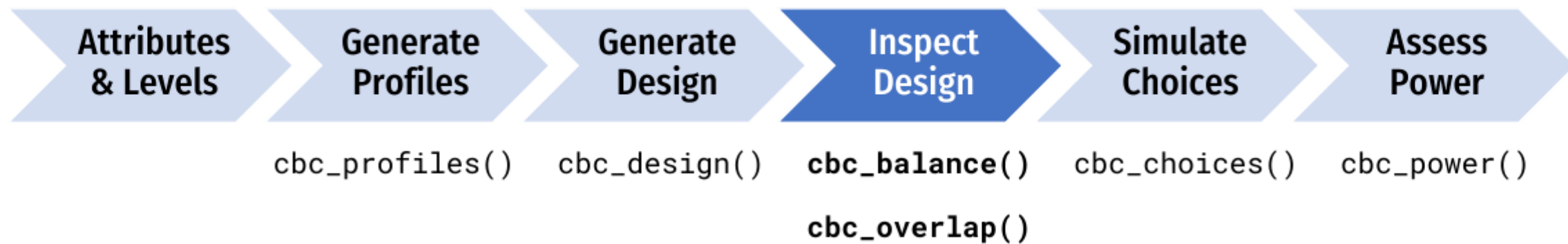
```
#>   respID qID altID obsID price       type freshness
#> 1      1   1     1     1   3.0       Fuji      Poor
#> 2      1   1     2     1   1.0       Gala      Poor
#> 3      1   1     3     1   1.0 Honeycrisp      Poor
#> 4      1   2     1     2   2.5       Fuji Excellent
#> 5      1   2     2     2   2.0       Gala      Poor
#> 6      1   2     3     2   3.0 Honeycrisp Excellent
```

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

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

cbc_overlap()

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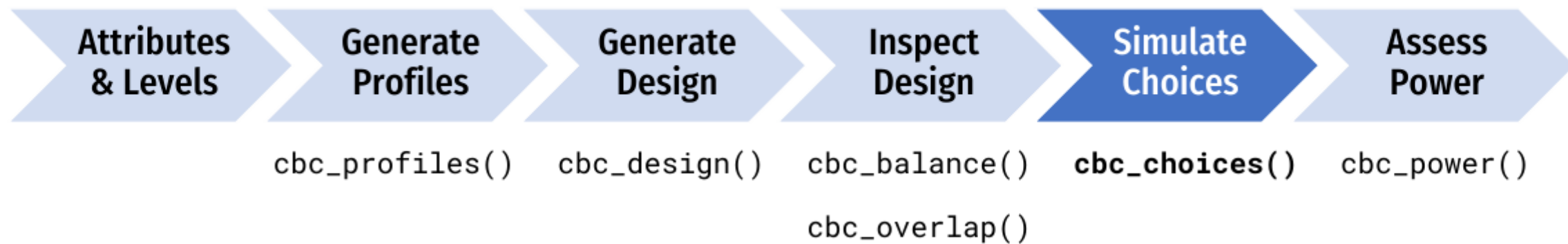
Attributes & Levels	Generate Profiles	Generate Design	Inspect Design	Simulate Choices	Assess Power
	`cbc_profiles()`	`cbc_design()`	`cbc_balance()` `cbc_overlap()`	**`cbc_choices()`**	`cbc_power()`

Simulate random choices

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

```
head(data)
```

```
#>   profileID respID qID altID obsID price       type freshness choice
#> 1         5      1   1     1     1   3.0       Fuji      Poor      0
#> 2         8      1   1     2     1   1.0       Gala      Poor      0
#> 3        15      1   1     3     1   1.0 Honeycrisp      Poor      1
#> 4        46      1   2     1     2   2.5       Fuji Excellent      0
#> 5        10      1   2     2     2   2.0       Gala      Poor      1
#> 6        61      1   2     3     2   3.0 Honeycrisp Excellent      0
```

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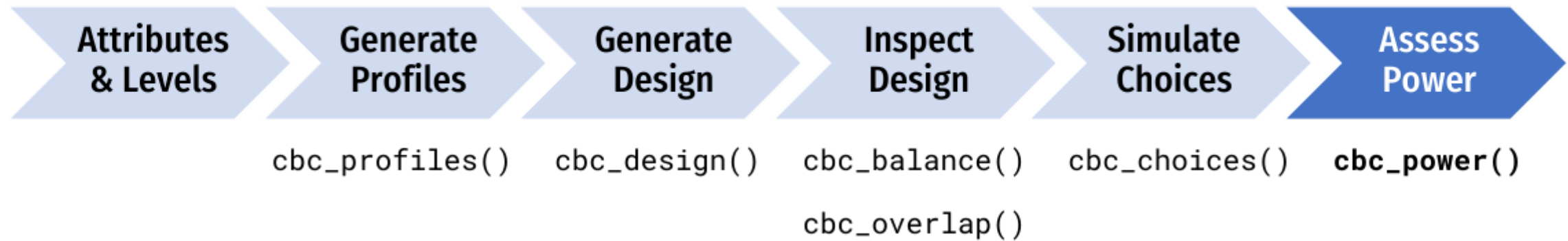
Attribute	Level	Utility
Price	Continuous	-0.1
Type	Fuji	0
	Gala	N(0.1, 0.5)
	Honeycrisp	N(0.2, 1)
Freshness	Average	0
	Excellent	0.1
	Poor	-0.2

Simulate choices according to a prior

(Models with interactions)

```
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

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

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

`cbc_overlap()`

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 -0.182544
#> 2         30             typeGala -0.255585
#> 3         30        typeHoneycrisp  0.173183
#> 4         30      freshnessAverage  0.299261
#> 5         30   freshnessExcellent -0.106366
#> 6         60                price -0.162037
```

tail(power)

```
#>    sampleSize                 coef
#> 45        270  freshnessExcellent -0.30805
#> 46        300                price -0.11574
#> 47        300             typeGala  0.08684
#> 48        300       typeHoneycrisp  0.23828
#> 49        300     freshnessAverage -0.00292
#> 50        300  freshnessExcellent -0.30257
```

Conduct a power analysis
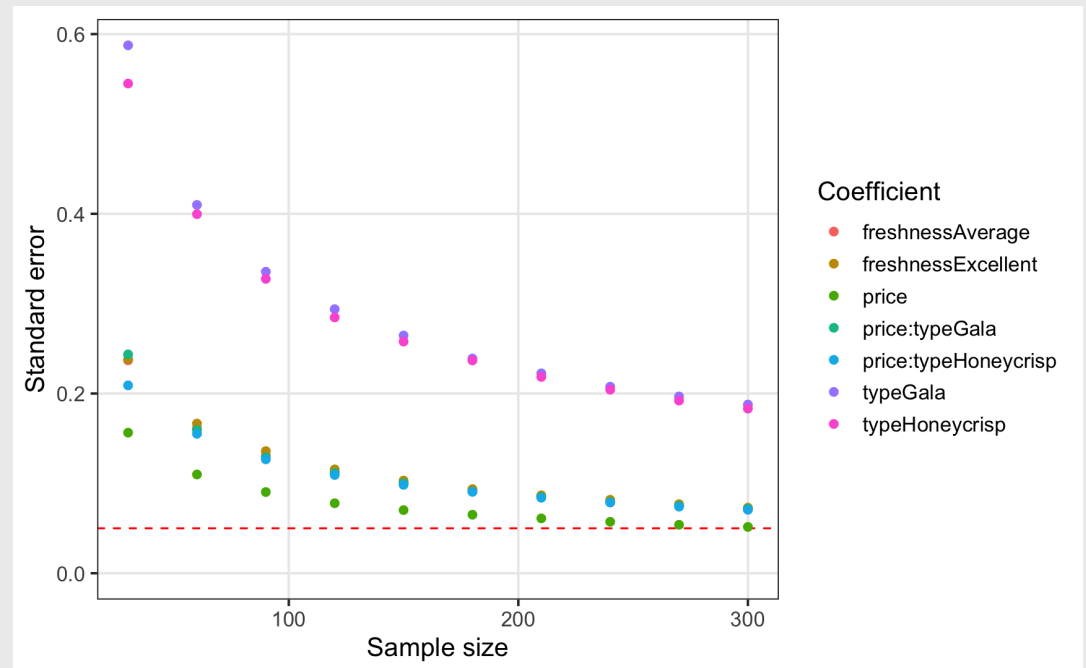
```
plot(power)
```
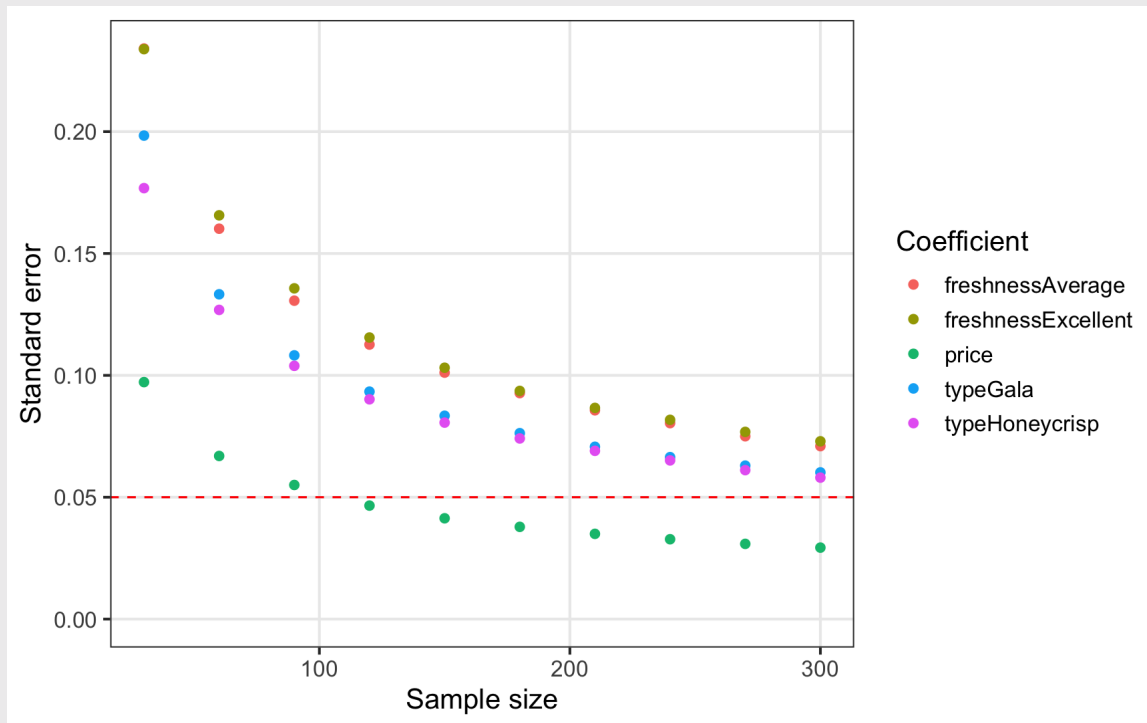
Conduct a power analysis

```
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)
```
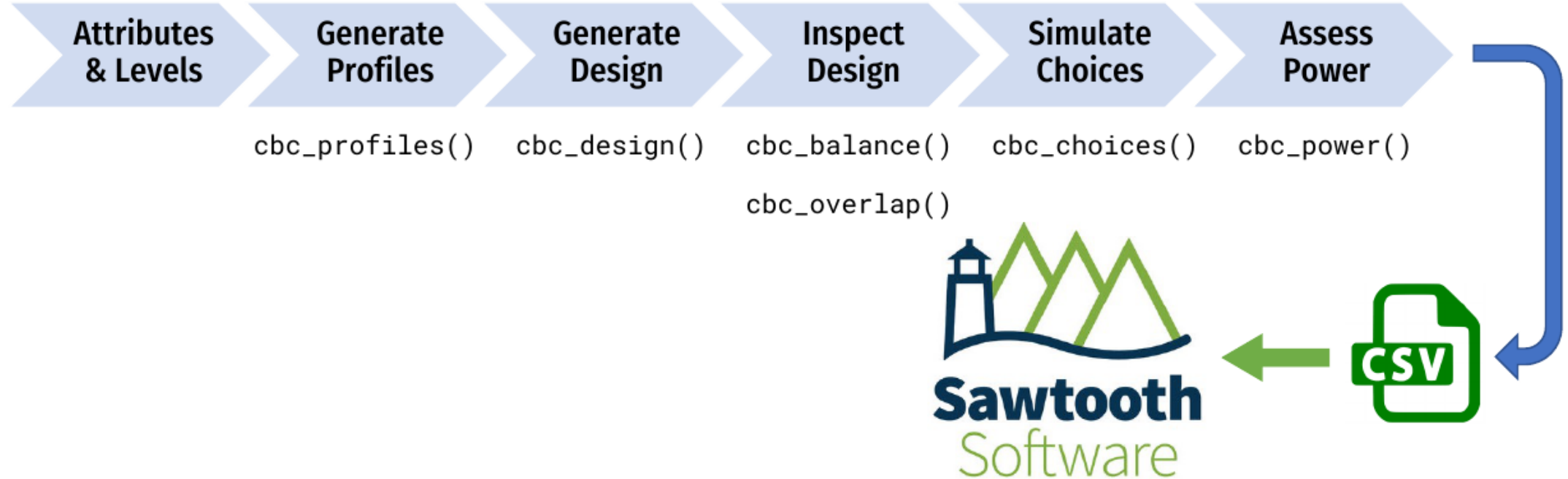
Back to workshop website:

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

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