ChoiceDesign Documentation

Release latest

Jose Ignacio Hernandez

Nov 06, 2022

Contents:

choicedesign.design	Modules to construct efficient designs

1 choicedesign.design

Modules to construct efficient designs

Classes

RUMDesign(atts_list, n_alts, ncs[, optout, asc])	RUM-consistent discrete choice experiment design class

class choicedesign.design.**RUMDesign**(*atts_list: list*, *n_alts: int*, *ncs: int*, *optout: bool = False*, *asc: Optional[dict] = None*)

RUM-consistent discrete choice experiment design class

This class allows to create an efficient design for a discrete choice experiment based in a Random Utility Maximisation (RUM) model [1].

Parameters

- **atts_list** (*list[dict]*) – List of attributes of the design. Each element is a dictionary that contains the following keys:
  - *name*: the attribute name
  - *levels*: a list of attribute levels
  - *coding*: the coding of the attribute levels. Supporte types are 'numeric' and 'dummy'.
  - *pars*: prior parameters of the attribute. If *coding = 'numeric' then only one parameter is required. If `coding = 'dummy', then `levels-1* parameters must be used. The first attribute level is taken as a baseline.
- **n_alts** (*int*) – Number of alternatives, apart from the opt-out (if defined).

- **self.ncs** (*int*) – Number of choice situations.
- **optout** (*bool, optional*) – Should an opt-out be included? If so, it is included as the last alternative, by default False.
- **asc** (*dict, optional*) – Dictionary that defines alternative-specific constants (ASC) and their respective prior parameters. Each key is the alternative in which an ASC is desired, and each value is the corresponding prior parameter.

optimise(*cond: Optional[list[str]] = None*, *n_blocks: Optional[int] = None*, *iter_lim: Optional[int] = None*, *noimprov_lim: Optional[int] = None*, *time_lim: Optional[int] = None*, *seed: Optional[int] = None*, *verbose: bool = False*)

Create D-efficient RUM design

Starts the optimisation of the design using a random swapping algorithm and the attributes and prior parameters specified in the *RUMDesign* object. Allows for conditions, blocking and user-defined stopping criteria.

Parameters

- **cond** (*list[str], optional*) – List of conditions that the final design must hold. Each element is a string that contains a single condition. Conditions can be of the form of binary relations (e.g., $X > Y$ where X and Y are attributes of a specific alternative) or conditional relations (e.g., *if* $X > a$ *then* $Y < b$ where a and b are values). Users can specify multiple conditions when the operator *if* is defined, separated by the operator *&*, by default None
- **n_blocks** (*int, optional*) – Number of blocks of the final design. Must be a multiple of the number of choice situations, by default None
- **iter_lim** (*int, optional*) – Number of iterations before the algorithm stops, by default None
- **noimprov_lim** (*int, optional*) – Number of iterations without improvement before the algorithm stops, by default None
- **time_lim** (*int, optional*) – Time (in minutes) before the algorithm stops, by default None
- **seed** (*int, optional*) – Random seed, by default None
- **verbose** (*bool, optional*) – Whether status messages and progress are shown, by default False

Returns

- **optimal_design** (*pandas.DataFrame*) – The final (optimal) design
- **init_perf** (*float*) – D-error of the initial design
- **final_perf** (*float*) – D-error of the final design
- **final_iter** (*int*) – Total number of iterations
- **ubalance_ratio** (*float*) – Utility balance ratio

Index

C

M

O

R