PortChoice Documentation

Release lastest

Jose Ignacio Hernandez

Oct 24, 2022

Contents:

Modules

portchoice.design	Functions to create experimental designs for portfolio choice models.
portchoice.generate	Portfolio choice data generation functions.
portchoice.models	Portfolio choice model functions.

1 portchoice.design

Functions to create experimental designs for portfolio choice models.

Classes

PortDesign(ATTLIST, NCS)	Experimental design class

class portchoice.design.**PortDesign**(*ATTLIST: list*, *NCS: int*)

Experimental design class

This is the class to create experimental designs for portfolio choice models.

Parameters

- **ATTLIST** (*list*) – List that contains the specification of each alternative of the experiment. Each element is a dictionary, in which each element is an attribute. The keys of each element are the attribute names and each value is a list that contains the attribute levels.
- **NCS** (*int*) – Number of choice situations.

generate_design(*ITERLIM: Optional[int] = None*, *NOIMPROVLIM: Optional[int] = None*, *TIMELIM: Optional[float] = None*, *CRIT: str = 'deff'*, *COND: Optional[list] = None*, *SEED: Optional[int] = None*, *VERBOSE: bool = True*)

Generate experimental design

It generates an experimental design based on the specification of the parent class and the parameters provided by the user.

Parameters

- **ITERLIM** (*int, optional*) – Iteration limit, by default None
- **NOIMPROVLIM** (*int, optional*) – Limit of iterations without improvement, by default None
- **TIMELIM** (*float, optional*) – Time limit in minutes, by default None
- **CRIT** (*str, optional*) – Efficiency criterion. If *maxcorr*, then the algorithm minimises the maximum value of the correlation between attributes. If *deff*, the algorithm maximises the D-efficiency of a linear model, by default 'deff'
- **COND** (*list, optional*) – Conditions between attributes of the design. See the examples for an overview , by default None
- **SEED** (*int, optional*) – Random seed, by default None
- **VERBOSE** (*bool, optional*) – Whether the algorithm prints output while optimising, by default True

Returns

- **bestdes** (*np.ndarray*) – Optimal design
- **perf** (*float*) – Value of the efficiency criterion at the **first** iteration
- **bestperf** (*float*) – Value of the efficiency criterion at the **last** iteration
- **elapsed_time** (*float*) – Optimisation time
- **best_t** (*int*) – Number of iterations

2 portchoice.generate

Portfolio choice data generation functions.

Classes

PortGen(V[, C, delta_0, B])	Portfolio choice data generator class.

class portchoice.generate.**PortGen**(*V: ndarray*, *C: Optional[ndarray] = None*, *delta_0: Optional[float] = None*, *B: Optional[float] = None*)

Portfolio choice data generator class.

This class generates synthetic choices of a portfolio choice model. It takes the utility of each individual alternative and generates synthetic choices and the 'true' log-likelihood function. *PortGen* allows for unconstrained and constrained choice situations (i.e., with resource constraints).

Parameters

- **V** (*np.ndarray*) – Array of deterministic utilities of each individual alternative.
- **C** (*np.ndarray, optional*) – Array of costs of each individual alternative, by default None
- **delta_0** (*float, optional*) – Parameter of the marginal utility of non-spent resources. Must be different from None if *C* is defined, by default None
- **B** (*float, optional*) – Available resources. Must be different from None if *C* is defined, by default None

get_choices()

Generate portfolio synthetic choices and log-likelihood.

It takes the configurations parameters of *PortGen* and generates a *numpy* array that contains the pseudo-synthetic choices for each observation. Additionally, it returns the 'true' log-likelihood.

Returns

- **y** (*numpy.ndarray*) – A *numpy* array with the synthetic choices for each observation.
- **ll** (*float*) – The 'true' log-likelihood.

3 portchoice.models

Portfolio choice model functions.

Classes

PortLogit(Y[, X, Z, C, B])	Portfolio logit model class.

class portchoice.models.**PortLogit**(*Y: DataFrame*, *X: Optional[DataFrame] = None*, *Z: Optional[DataFrame] = None*, *C: Optional[DataFrame] = None*, *B: Optional[float] = None*)

Portfolio logit model class.

It contains the routines to prepare the data and estimate a portfolio logit model, as well as for the computation of the optimal portfolio.

Parameters

- **Y** (*pd.DataFrame*) – A data frame with choices of each alternative for each respondent.
- **X** (*pd.DataFrame, optional*) – A data frame with the alternative-specific variables (e.g., attributes), by default None
- **Z** (*pd.DataFrame, optional*) – A data frame with the individual-specific variables, by default None
- **C** (*pd.DataFrame, optional*) – A data frame with the costs of each individual alternative for each respondent, by default None
- **B** (*float, optional*) – Resource constraint, by default None

estimate(*startv: ndarray*, *asc: ndarray*, *beta_j: Optional[ndarray] = None*, *delta_0: Optional[float] = None*, *hess: bool = True*, *tol: float = 1e-06*, *verbose: bool = True*)

Estimate portfolio logit model

It starts the optimisation routine of the portfolio logit model. The user can specify the presence of alternative-specific constants (*asc*), separate parameters for the alternative-specific variables (*beta_j*) and the presence of a parameter that captures the marginal utility of non-spent resources (*delta_0*).

Parameters

- **startv** (*np.ndarray*) – Starting values for the maximum-likelihood estimation routine.
- **asc** (*np.ndarray*) – An array of length *n_alternatives*, in which each element can be either equal to one if the ASC of the corresponding alternative is estimated, and zero otherwise.
- **beta_j** (*np.ndarray, optional*) – An array of dimension *n_alternatives*n_attributes*, in which each element can be either equal to one if the corresponding alternative-specific parameter is estimated, and zero otherwise. If *beta_j = None* and *X* exists then single attribute-specific parameters (i.e., equal across alternatives) are estimated , by default None
- **delta_0** (*float, optional*) – If None and *C* exists, then the parameter of the marginal utility of non-spent resources is estimated. If *delta_0* is a float, then the parameter is fixed to the value of *delta_0*, by default None
- **hess** (*bool, optional*) – Whether the finite-difference hessian is estimated at the end of the estimation routine, by default True
- **tol** (*float, optional*) – Tolerance of the gradient in the estimation routine, by default 1e-6
- **verbose** (*bool, optional*) – Whether the estimation routine returns information at each iteration. See the documentation of *scipy.optimize.minimize* with method *l-bfgs-b* for more information, by default True

Returns

- **ll** (*float*) – Log-likelihood function at the optimum
- **coef** (*numpy.ndarray*) – Estimated parameters at the optimum
- **se** (*numpy.ndarray*) – Standard errors of *coef*. If *hess = False* then *se = 0.*
- **hessian** (*numpy.ndarray*) – Finite-difference Hessian. If *hess = False* then *hessian = 0.*
- **diff_time** (*float*) – Estimation time in seconds.

optimal_portfolio(*X: Optional[Series] = None*, *Z: Optional[DataFrame] = None*, *C: Optional[Series] = None*, *B: Optional[float] = None*, *sims: int = 1000*)

Compute the optimal portfolio

Computes the optimal portfolio based on the estimation results (i.e., obtained from *estimate()*) and user-defined variables. The optimal portfolio is computed by computing the expected utility of all possible combinations of alternatives. The expected utility is computed by simulation using *sims* error draws from an Extreme Value (Gumbel) distribution.

Parameters

- **X** (*pd.Series, optional*) – Series of alternative-specific variables, by default None
- **Z** (*pd.DataFrame, optional*) – Data frame with individual-specific variables, by default None
- **C** (*pd.Series, optional*) – Series with individual costs per alternative, by default None
- **B** (*float, optional*) – Resource constraint, by default None
- **sims** (*int, optional*) – Number of Extreme Value random draws, by default 1000

Returns

portfolio – Data frame with the optimal portfolio (ranked combinations), its expected utility and its total cost (if *C* is not None).

Return type
pd.DataFrame

4 Indices and tables

- genindex
- modindex
- search

Index

E

G

M

O

P