Documentation

This application facilitates conducting the single-choice list task to elicit individual level risk preferences as proposed by Eckel/Grossman (2002, 2008), as an *oTree* application (Chen et al., 2016) in different variants and parameterizations by simply altering the documented variables in `config.py`.

> Please note that, starting with oTree version 1.2.x, the template `Base.html` (in _templates/global) has been renamed to `Page.html`. If you wish to stick to an older version, please replace `{% extends "global/Page.html" %}` by `{% extends "global/Base.html" %}` in all HTML-templates in the app folder.

Installation

To install the app to your local *oTree* directory, copy the folder "mcl" to your *oTree* Django project and extent the session configurations in your `settings.py` at the root of the *oTree* directory by something like

```
SESSION_CONFIG = [
   ...
   {
      'name': 'scl',
      'display_name': "Single Choice (Eckel/Grossman)",
      'num_demo_participants': 1,
      'app_sequence': ['scl'],
   },
   ...
]
```

Please note that global settings as `REAL_WORLD_CURRENCY_CODE`, `USE_POINTS`, as well as `SESSION_CONFIG_DEFAULTS` (including `participation_fee` and `real_world_currency_per_point`) are – as for all *oTree* apps – specified in *oTree*'s `settings.py` rather than the application itself but do affect the display of currency figures as well as the calculations of payoffs and amounts to pay.

Setup

To set up the task, the only thing to be done is to alter pre-defined variables in the file `config.py` at the root of the app's directory. Any combination of the variables described below is operable. By that means, several different variations of the task are easily implemented. The following variables can be specified:

`num_lotteries` (integer field):

Number (`N`) of lotteries to be generated, with `i = 1, 2, ..., N`.

`sure_payoff` (decimal/currency fields):

`sure_payoff` determines the "low" and "high" outcomes for the first of the `num_lotteries` lotteries and the "starting point" for all lotteries in the task. That is, the first lottery yields either a payoff equal to `sure_payoff` with a probability of 1 while the subsequent lotteries are defined relative to this outcomes. (Please note that the currency of payoffs displayed to subjects is determined by the global settings of *oTree* in `settings.py`.)

`delta_lo` and **`delta_hi`** (decimal/currency fields):

`delta_lo` and `delta_hi` define the increments of the low and high outcomes over the number of lotteries. While the first lottery pays `sure_payoff` for sure, the second lottery either yields (`sure_payoff - delta_lo`) or (`sure_payoff + delta_hi`). Generally, lottery `i` yields a "low" outcome of (`sure_payoff - (i-1) · delta_lo`) or a "high" outcome of (`sure_payoff + (i-1) · delta_hi`). Note that `delta_lo` should be strictly smaller than `delta_hi` (such that both the expected value and the standard deviation increase with `i`) in order to discriminate between different levels of risk aversion.

`risk_loving` (boolean field):

Includes an additional lottery to separate risk-loving from risk-neutral preferences. If `risk_loving = True`, a lottery with the same expected value but a higher standard deviation as lottery `N` is generated in addition. Note that `risk_loving = True` implies that the overall number of lotteries rendered will be (`num_lotteries + 1`).

`probability` (integer/float field):

Defines the probability of the “high” lottery outcomes denoted in percent. Thus, `probability = x` implies an `x`%-chance that outcome “high” and a `(100-x)`%-chance that outcome “low” is realized. Note that the probability in this task is constant across all lotteries and only the lottery outcomes change.

`random_order` (boolean field):

If `random_order = False`, all `num_choices` binary decisions are listed in ascending order of the probability of the high outcome; if `random_order = True`, the ordering of binary decisions is randomized for display.

`instructions` (boolean field):

If `instructions = True`, a separate template `Instructions.html` is rendered prior to the task in round one. If `instructions = False`, the task starts immediately (e.g. in case of printed instructions). Please note that the instructions included serve only exemplary purposes and need to be adjusted to your settings in `setup.py`.

`results` (boolean field):

Determines whether a results page summarizing game outcome is rendered or not. If `results = True`, a separate view `Results.html` containing all relevant information (i.e. the chosen lottery, the randomly picked outcome to be payed, and the payoff) is rendered. If `results = False`, no separate page is displayed.

References

Chen, D. L., Schonger, M., Wickens, C., 2016. “oTree – an open-source platform for laboratory, online and field experiments”. *Journal of Behavioral and Experimental Finance* 9, 88–97.

Eckel, C. C., Grossman, P. J., 2002. “Sex differences and statistical stereotyping in attitudes toward financial risk“. *Evolution and Human Behavior* 23, 281–295.

Eckel, C. C., Grossman, P. J., 2008. “Forecasting risk attitudes: An experimental study using actual and forecast gamble choices“. *Journal of Economic Behavior & Organization* 68, 1–17.