Documentation

This application facilitates conducting the multiple price list task between pairs of lotteries with fixed lottery outcomes and varying probabilities, as proposed by Holt/Laury (2002), as an *oTree* application (Chen et al., 2016) in numerous different variants 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 “mpl” 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': 'mpl',
      'display_name': "Multiple Price List (Holt/Laury)",
      'num_demo_participants': 1,
      'app_sequence': ['mpl'],
   },
   ...
]
```

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:

`lottery_•_•` (decimal/currency fields):

`lottery_a_lo`, `lottery_a_hi`, `lottery_b_lo`, and `lottery_b_hi` determine the "low" and "high" outcomes for "Lottery A" and "Lottery B", respectively. Outcomes are identical for all choices and only probabilities of high and low outcomes change. (Please note that the currency of payoffs displayed to subjects is determined by the global settings of *oTree* in `settings.py`.)

`num_choices` (integer field):

Number of choices between "Lottery A" and "Lottery B". As in Holt/Laury (2002), `num_choices` determines the probabilities of high and low outcomes of both lottery "A" and "B": for `num_choices` being specified as *x*, the probability of outcome "high" is *1/x* in the first choice, *2/x* in the second, etc.

`certain_choice` (boolean field):

Defines whether a certain choice is included in the choice list or not. That is, if `certain_choice = True`, the choice between lottery "A" and "B" with a probability of 1 for the high outcome is included; if `certain_choice = False`, the list only contains (`num_choices` – 1) binary decision pairs. Note, however, that the probability of outcome "high" is set by `num_choices`, not by (`num_choices` – 1).

`one_choice_per_page` (boolean field):

If `one_choice_per_page = True`, each single binary choice between "Lottery A" and "Lottery B" will be rendered on a separate page. Technically, each decision is separated into rounds, i.e. the first choice is made in round 1, the second choice in round 2, etc. Accordingly, the dataset for download contains `num_choices` rows, one for each round, for each subject with a single observation corresponding to the respective choice. If `one_choice_per_page = False`, all `num_choices` pairs are displayed in a single table on one page.

`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.

`enforce_consistency` (boolean field):

If `enforce_consistency = True`, subjects are enforced to answer the choice list without preference reversals. That is, all “A” lotteries above a selected option “A” and all “B” lotteries below a selected option “B” are automatically checked (i.e., subject have to click two radio buttons at most). Therefore, `enforce_consistency = True` implies a single switching point and thereby imposes strict monotonicity of revealed preferences and enforces transitivity. Note that `enforce_consistency = True` is only implementable for single-page choice lists in ascending order, i.e. if `one_choice_per_page = False` and `random_order = False`.

`percentage` (boolean field):

In the classical task proposed by Holt/Laury (2002), probabilities are displayed as fractions with the number of choices determining the denominator, which can be achieved by specifying `percentage = False`. To render the probabilities of outcome “high” as a percentage number, set `percentage = True`.

`small_pies` (boolean field):

To emphasize differences in probabilities, `small_pies = True` renders smallish pie charts on the left of “Lottery A” and on the right of “Lottery B”, respectively, in each row of the table listing the binary choices (see Figure 1 for an example of the Holt/Laury (2002) parameterization with `small_pies = True`). If `small_pies = False`, no graphical illustrations will be displayed.

`large_pies` (boolean field):

If `one_choice_per_page = True`, lotteries “A” and “B” can be rendered in terms of large pie charts instead of verbal descriptions in tabular format (as for instance applied by Hey/Orme, 1994). If `large_pies = True`, lotteries are depicted as pie charts representing the probabilities with labels for the payoffs (see Figure 2). Moreover, input is facilitated by replacing the radio buttons and the “next” button by choice buttons for option “A” and “B” respectively.

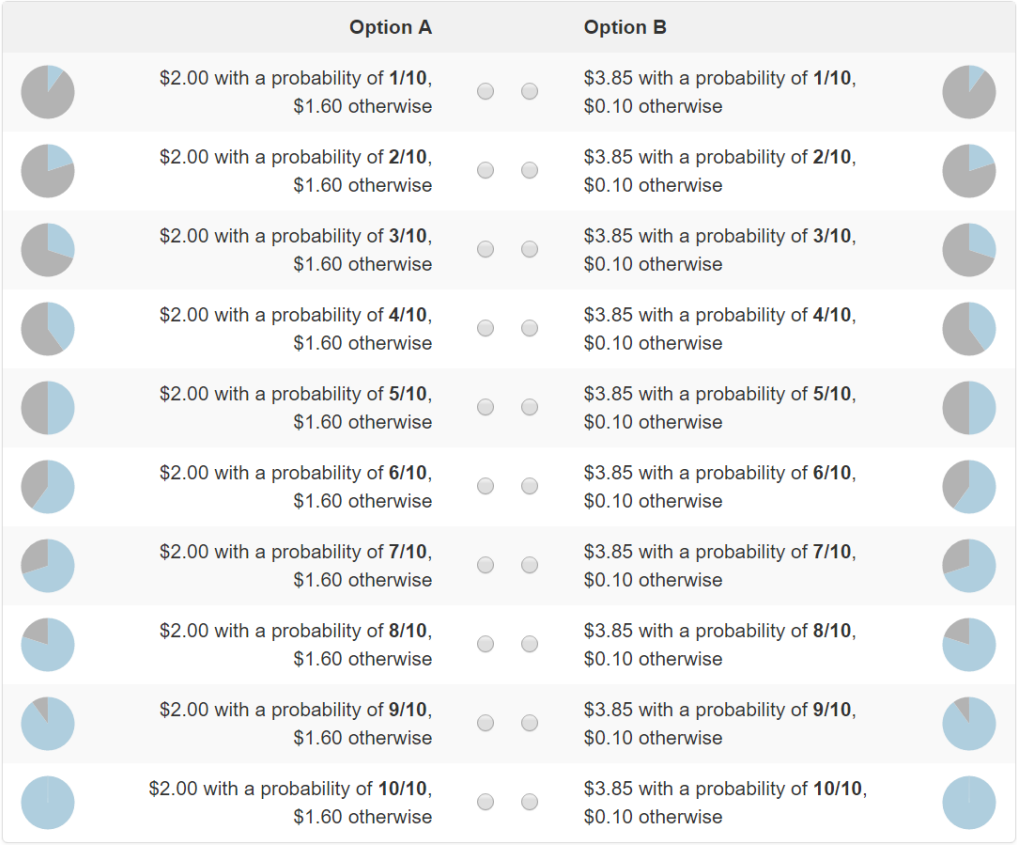

Figure 1. Choice list with parameterization of Holt/Laury (2002) and options `certain_choice = True`, `one_choice_per_page = False`, `random_order = False`, and `small_pies = True`.

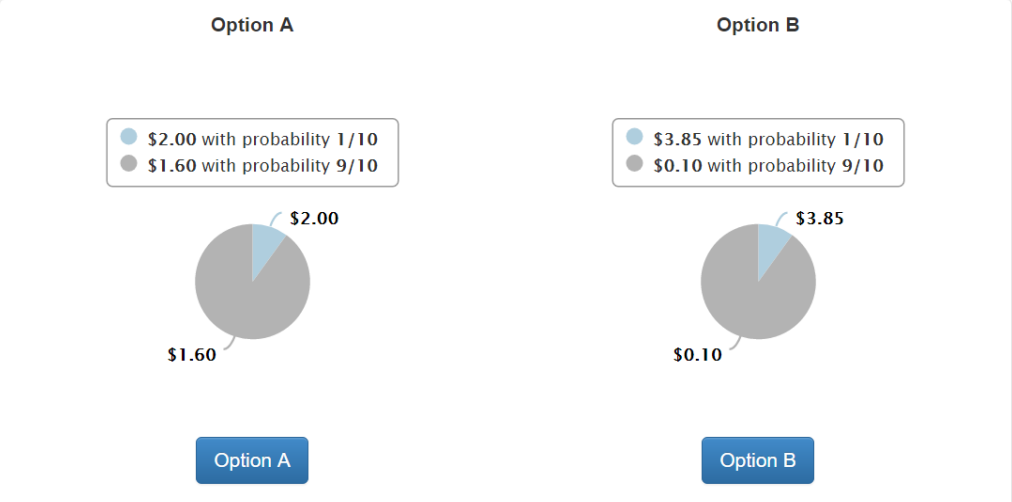

Figure 2. Fourth choice from a choice list with parameterization of Holt/Laury (2002) and options `certain_choice = True`, `one_choice_per_page = True`, `random_order = False`, and `large_pies = True`.

`progress_bar` (boolean field):

A progress bar, optionally, allows for graphical highlighting of the subject's advance within the task, in terms of how many decision have already been completed. If `progress_bar = True` and `one_choice_per_page = True`, a progress bar is rendered; if `progress_bar = False`, no information with respect to the advance within the task is displayed. Furthermore, information in terms of "page *x* out of `num_choices`" (with *x* denoting the current decision) is provided. Note that this variable does not affect the display of choices if all choices are shown at once, i.e. if `one_choice_per_page = False`.

`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 `config.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 randomly picked lottery to be payed, the randomly determined outcome to pay, 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.

Hey, J. D., Orme, C., 1994. “Investigating generalizations of expected utility theory using experimental data”. *Econometrica* 62 (6), 1291–1326.

Holt, C. A., Laury, S. K., 2002. “Risk aversion and incentive effects”. *American Economic Review* 92 (5), 1644–1655.