## *Documentation*

This app allows conducting the certainty equivalent method to elicit individual level risk preferences, as proposed by Cohen et al. (1987) and Abdellaoui et al. (2011), as well as variations thereof, as suggested by Bruner (2009) and Gächter et al. (2010), 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 “cem” 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': 'cem',
      'display_name': "Certainty Equivalence Test",
      'num_demo_participants': 1,
      'app_sequence': ['cem'],
   },
   ...
]
```

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:

**`variation`** (string field):

`variation` defines the which of the task's parameter should be varied across the choices (see `step_size`), including `'sure_payoff'`, `'probability'`, `'lottery_hi'`, and `'lottery_lo'`, while the remaining three parameters are held constant over all choices.

If `variation = 'sure_payoff'`, the task resembles the choice list procedure proposed by Cohen et al. (1987) and Abdellaoui et al. (2011), where a fixed lottery constitutes "Option A" while "Option B" is an increasing sure payoff. The properties of choice lists with varying probabilities (`variation = 'probability'`) and high payoffs (`variation = 'lottery_hi'`) are discussed by Bruner (2009). If `variation = 'lottery_lo'`, the task resembles the properties of the choice list introduced by Gächter et al. (2010) which is frequently used to elicit individual-level loss aversion (given that low lottery payoffs turn negative).

**`num_choices`** (integer field):

Number (`n`) of choices with `i = 1, 2, …, n`.

**`lottery_hi`** and **`lottery_lo`** (decimal/currency fields):

`lottery_hi` and `lottery_lo` determine the "high" and the "low" payoff of the lottery ("Option A"). The lottery payoffs remain constant if `variation = 'sure_payoff'` or `variation = 'probability'`. If variation = `'lottery_hi'` (`variation = 'lottery_lo'`), `lottery_hi` (`lottery_lo`) constitutes the high (low) lottery payoff in the first choice in the list. For subsequent choices, the high (low) outcome is determined by `step_size`. (Please note that the currency of payoffs displayed to subjects is determined by the global settings of *oTree* in `settings.py`.)

**`probabilty`** (integer/decimal field):

`probability` determines the likelihood of outcome "high" as a percentage number for the lottery in the choice list, i.e. `probability = x` implies an `x`% chance that the prospect pays `lottery_hi` and a `(1-x)`% chance of yielding `lottery_lo`. The probability of lottery payoffs is held constant for `variation = 'sure_payoff'` and `variation = 'lottery_*'`. If `variation = 'probability'`, `probability` determines the likelihood of the high payoff in the first choice of the list while the probabilities of subsequent choices are defined by `step_size`.

**`sure_payoff`** (decimal/currency field):

`sure_payoff` constitutes the certain amount offered as the alternative choice to the lottery ("Option B"). The sure payment remains constant if `variation = 'probability'` or `variation = 'lottery_*'`. If `variation = 'sure_payoff'`, `sure_payoff` constitutes the certain amount offered in the first choice while sure payoffs in subsequent choices increase by `step_size`. (Please note that the currency of payoffs displayed to subjects is determined by the global settings of *oTree* in `settings.py`.)

**`step_size`** (integer/decimal/currency field):

`step_size` is defined in units of the parameter specified in variation, i.e. it refers to currency units if `variation` is set to `'sure_payoff'` or `'lottery_*'` but to percentage if variation is set to `'probability'`. `step_size` determines the increment of the varying parameter (while the remaining three are held constant).

For choice `i = 1, 2, …, n`, the varying parameter, $var_i$, is defined by $var_i$ `= var + (i-1) *` `step_size` (with *var* referring to `lottery_hi`, `lottery_lo`, `probability`, or `sure_payoff`, depending on the setting of `variation`). Put differently, for each choice, the varying parameter increases by `step_size`.

**`endowment`** (decimal/currency field):

`endowment` defines an additional endowment for the task, which is added to a subject's payoff, in order to capture potential losses if `variation = 'lottery_lo'`. If no endowment shall be implemented, set `endowment = 0`.

**`accept_reject`** (boolean field):

If `accept_reject = False`, subjects face a table with a lottery on the left and a certain payment on the right. Subjects are required to reveal their preference for "Option A" (the lottery) or "Option B" (the sure payoff) for each of the `num_choices` choices. If `accept_reject = True`, only the lottery will be displayed in the table but not the sure payoff, as in the choice list procedure suggested by Gächter et al. (2010). Subjects are asked to indicate whether they want to accept or to reject each of the lotteries instead. Note that `accept_reject = True` is only implementable if `variation` is set to `'probability'` or `'lottery_*'`. If `accept_reject = True`, `choice = "A"` refers to acceptance while `choice = "B"` refers to rejection

**`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`.

**`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*


Abdellaoui, M., Driouchi, A., L'Haridon, O., 2011. "Risk aversion elicitation: Reconciling tractability and bias minimization". *Theory and Decision* 71, 63–80.

Bruner, D. M., 2009. "Changing the probability versus changing the reward". *Experimental Economics* 12 (4), 367–385.

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.

Cohen, M., Jaffray, J.-Y., Said, A., 1987. "Experimental comparison of individual behavior under risk and under uncertainty for gains and for losses". *Organizational Behavior and Human Decision Processes* 39, 1–22.

Gächter, S., Johnson, E. J., Herrmann, A., 2010. "Individual level loss aversion in riskless and risky choices". *Centre for Decision Research and Experimental Economics*, Discussion Paper No. 2010-20.