Workshop: Section 6 - More Complex Structures: Roles, Matching

Philipp Chapkovski
University of Bonn
chapkovski@uni-bonn.de

September 11th - 12th, 2023

Assigning Different Roles

- Explanation of the concept of roles in economic games.
- Common roles: Investor, Allocator, Buyer, Seller, etc.

Ultimatum game:

- Player making offer: `first_mover`
- Player accepting or rejecting: `second_mover`

Steps to Assign Roles

Dictator game:

- Player deciding over the distribution: `dictator`
- Passive receiver: ‘recipient“

1. Define roles in `models.py` within a group class.

```
class C(BaseConstants):

    DICTATOR_ROLE = 'Dictator'
    RECIPIENT_ROLE = 'Recipient'
```

Demonstrate Role Assignment

- Conditional role assignment based on round or other factors.

```
class Player(BasePlayer):
  def role(self):
    if self.round_number % 2 == 0:
        return 'dictator'
    else:
        return 'recipient'
```

Conditionally Show Pages

- Show or skip pages based on roles.

```
class MyPage(Page):
    @staticmethod
    def is_displayed(player):
        return player.role == 'Investor'
```

Multi-Round Games

Introduction to Multi-Round Games

- Concept of repeated games in economic experiments.
- Importance of multi-round structure for various research questions.

Configuring oTree for Multiple Rounds

- Use of `C.NUM_ROUNDS` to specify the number of rounds.
- `player.in_rounds(1, 3)` and `group.in_rounds(1, 3)` for data across rounds.

```
class C(BaseConstants):
    NUM_ROUNDS = 5
```

Implementing Multi-Round Logic

Retrieving Data from Other Rounds

- Use `player.in_round(x)` to get data from a specific round.

```
previous_round_player = self.player.in_round(self.round_number - 1)
```

- Use `player.in_all_rounds()` to get data from all rounds.

```
total_contributions = sum([p.contribution for p in self.player.in_all_rounds()
```

Methods for Passing Information: Within Group

Useful functions for sharing information within a group:

- `get_others_in_group`:

```
other_players = player.get_others_in_group()
```

  Returns a list of other players in the same group.

- `get_players`:

```
all_players = group.get_players()
```

  Returns a list of all players in the group.

- `get_player_by_id`:

```
specific_player = group.get_player_by_id(1)
```

  Returns a player object with the specified ID within the group.

- `get_player_by_role`:

```
specific_player = group.get_player_by_role('RoleName')
```

  Returns a player object with the specified role within the group.

Methods for Passing Information: Across Apps

- `player.participant.vars` and `session.vars` are dictionaries:
  - Storing data that persists across different apps for a single participant or all participants.
  - Can be dumped into a model field using `json.dumps()` if needed.
- Accessing variables:
  - `player.participant.vars['some_variable'] = value`
  - `self.session.vars['some_variable'] = value`
- Safe variable checking:

```
player.participant.vars.get('some_variable')
```

- Using `session.config` and `session.vars` directly in templates:
  - Useful for managing between-session treatments.