Advanced Treatments

A Chat Box Treatment Within a Session

In our public goods game we assigned the treatment at the session level. You may want to run multiple treatments simultaneously though. In our competition app we could set up a treatment that lets sellers use a chat box to coordinate on a price (cheap talk). First, we have to define a variable that tracks the treatment that subjects are in. Head to init.py and add a Boolean variable chat in the Player class.

```
class Player(BasePlayer):
    offer = models.CurrencyField(
        label="How much do you want to offer?",
        min = 0,
        max = 10
    )
    buy_from = models.IntegerField(
        label="Who do you want to buy from?",
        widget=widgets.RadioSelect()
    )
    game_role = models.StringField()
    chat = models.BooleanField()
```

Let's assume that if a player is in a treatment we want them to remain in that treatment. If we randomly assign the treatment on the player (or group) variable, participants may change from one treatment to the next between rounds. The following logic prevents this:

- For each group in the session in the first round of the game, draw the treatment variable once for all members of the group.
- Store the treatment variable for the participant (not the current player).
- Store the treatment variable for all player incarnations of the participant.

To store a variable for a participant between their incarnations as players we have to declare a participant variable. We can do this in settings.py. Locate PARTICPANT_FIELDS and add a variable called in_chat_treatment. Participant variables are not saved in the database. If you want to save a participant variable, save it in a player field (as we already plan to do).

```
PARTICIPANT_FIELDS = ['in_chat_treatment']
```

Return to init.py and head to the top of the page. To randomly assign treatments we have to use the random library. It is a default Python library so you can just type import random at the top.

```
import random
```

We already have some code in place when a new session is created. Head to the creating_subsession function at the bottom of the page. First, we want to randomly draw a treatment for each group, but only in round 1. In round 2 we want to use round 1's draw. Leave the existing code in creating_session as it is and start a new loop through all the groups below. First, write an if condition that checks if we are in round 1. Then, for every group, start with drawing a random number between 0 and 1. If the random number is greater than 0.5, assign the value True, else False.

```
if subsession.round_number == 1:
    for g in groups:
        if random.random() > 0.5:
            treatment = True
        else:
            treatment = False
```

Next, assign this treatment variable to every participant's treatment variable.

```
    players = g.get_players()
    for p in players:
        p.participant.in_chat_treatment = treatment
```

Using this code we randomly assign a treatment once for the participant. Now we need to assign the player incarnations the value of the participant variable. This has to happen outside of the if condition.

```
for g in groups:
    players = group.get_players()
    for p in players:
        p.chat = p.participant.in_chat_treatment
```

Run otree devserver to check if the assignment to treatments works.

Chat Setup

oTree has a pretty straight-forward implementation of chat. In the html template we can simply add {{ chat }}, and by default all players in the same group can chat with one another. Go to Offer.html and add a chat that is only viewable if the player is in the chat treatment.

```
{{ block title }}
    Offer
{{ endblock }}
{{ block content }}

    {{ if player.chat }}
        {{ chat }}
    {{ endif }}
    {{ formfields }}
```

```
    {{ next_button }}

{{ endblock }}
```

Test again with otree devserver. You can download the chat logs on the data tab.