Randomizing apps in oTree

Philipp Chapkovski

2/15/2021

Abstract

This document is a short description of how oTree builds a participant-to-page correspondence and the instructions for the experimenter how to use a code provided by the author to randomize the apps.

Randomizing apps in oTree

Due to numerous reasons, experimenters cannot randomize the order of apps in an oTree session. It is hard to say in advance whether this would be ever possible, but most likely in the foreseeble future this feature won't appear.

That is connected to the way oTree in general builds the correspondence between sequence of pages that participant needs to visit in a course of the study, and a specific participant. The technicalities of how this correspondence (or in terms of `otree`, 'lookup') is built changes from one version to another. The essence though is the same:

During the creation of the session, oTree loops through `app_sequence`, then for each app in this sequence it obtains the n number of periods (`num_rounds`) from `models.Constants` of this app, and `pages.page_sequence` and it creates $n\times p$ where p is a number of elements in the `page_sequence`. It repeats this procedure for each app, `gluing together` the apps. After that the `creating_session` is called. This fact is important because since `creating_session` is called afterwards, that doesn't let us intervene and modify `app_sequence` before participant-to-page correspondence is created.

This all leaves us little room for change of app_sequence on the individual basis. Before introduction of version `3.0.8` that was done with capturing `ParticipantToPlayerLookup` and modifying it during the session creation. That is the version of app randomization that is used in the code presented here: http://github.com/chapkovski/randomizing-apps-otree.

Specifically, it was done in `creating_session` of the `StartApp` that was used as 'entry point' for a sequence of apps (App1, App2, App3 in the demo provided in the same GitHub repo above):

```
class Subsession(BaseSubsession):
    def creating_session(self):
        for p in self.get_players():
            if not p.sequence_of_apps:
                ParticipantToPlayerLookup.objects.filter(
                  participant=p.participant).delete()
                p.sequence_of_apps = json.dumps(
                  get_new_sequence_of_apps(
                  self.session.config['app_sequence']))
                build_participant_to_player_lookups(p.participant,
                json.loads(p.sequence_of_apps),
                self.session)
```

In addition to bundling new lookups (using function `build_participant_to_player_lookups` shown below, the sequence was stored in a database for the further analysis in a variable `player.p.sequence_of_apps`.

Unfortunately with an introduction of new version (3.0.8+) of `oTree` that approach is no loner attainable. A new approach is not yet implemented, but those who are interested should subscribe for the changes in the corresponding GitHub repo.