Social Experiments with oTree and Python

Moinul Islam

9/15/2022

Table of contents

Preface

oTree is an open-source and online software for implementing interactive experiments in the laboratory, online, the field or combinations thereof. The basic experimental setup in oTree consists of (i) an experiment written within oTree, (ii) a server computer, which can be a cloud server or a local laptop and (iii) subjects' devices with a web browser. oTree creates an experimental session on the server, as well as links for all the participants and the experimenter.

oTree is a framework based on Python that lets you build multiplayer strategy games, like the prisoner's dilemma, public goods game and many other behavioral experiments in economics, psychology, and related field. This tutorial will develop experiments by using oTree software. For Python code generation, we will use Pycharm IDE.

1 Installation and environments

1.1 Install Python

- Before installing oTree, it is required to have Python installed in your environment to run oTree.
- Open the python website and install **Python3** for your OS from this Python website.
- Choose the most recent version of Python

1.2 Install pip

If your Python environment does not have pip installed, you can install it by following this pip instructions.

1.3 Install Pycharm

- PyCharm is a famous Python IDE (Integrated Development Environment). We recommend you to use PyCharm if you want to further customize your oTree app by programming. Please go to this Pycharm website to download the installation package.
- There are different version of Pycharm. I recommend to install the Pycharm professional because it has better syntax highlighting. To have the professional version for academic use, you need to have an academic email address.
- Alternatively you can install the community version of Pycharm which is free of charge.

1.3.1 Run Pycharm

- Create a new Pycharm project. Choose pure python. Give a name of the project. Pycharm will create the project for you.

1.4 Install GitHub

Make sure you have some kind of version control, I use GitHub. If you crash your code, you can always go back to a previous version of the code.

1.5 Install oTree

- oTree can be installed through pip. Open the terminal (Mac/Linux) or command prompt (Windows PowerShell) and type:

```
pip3 install otree
```

- You can install oTree in your computer by using the terminal in Pycharm as well.

```
pip3 install otree
```

1.5.1 Upgrading/reinstalling oTree

```
pip3 install -U otree
```

- Recommend upgrading every couple of weeks.

1.5.2 Regarding the installation of oTree in Linux

- You might need to do the proxy server settings
- In the Pycharm, you can open the terminal and install oTree as below:

```
set https_proxy address:password
pip install otree
```

1.6 oTree setup

- From your command prompt, create your **IGG** project

```
otree startproject IGG
```

- Move into the folder you just created

```
cd IGG
```

- Run the server

  Start your own terminal as a server.

```
otree devserver
```

- Open your browser to http://localhost:8000/. You should see the oTree demo site.
- To stop the server, press **Control+C** at your command line.
- To create a new app, run

```
otree startapp app_name
```

- Session configs are defined in **settings.py**

2 Structure of an oTree project

2.1 App

- One app can be an experiment or a part of experiment. A project is collection of all different types of App.
- If your experiment consist of a prisoner's dilemma game and a survey, then you need to create two apps, one app for prisoners dilemma and another app is for survey.
- Later you can combine these two apps and run one session.
- App is like one experiment and project is like a container of different experiments.
- How you structure the projects is up to you. You can choose the sequence.
- Now if you see the structure of the examples, you can see that there are various different types of folder.
- The following two folders are known as global settings as they are same level of the project

  _Static

- Images, sounds, videos

  _Template

- Design how web page look like. If you want a specific style of your webpage, you can include a template

  settings.py

- It is also global option. From this setting file, you can change various global options, for instance Your currency, language, interface
- In this **setting.py** file you will find the session_configs where you can setup the sequence of apps for your project.

2.2 How to create a new app.

- Navigate your working folder by using **cd**
- For instance you are using the project **IGG**. Then use

```
cd IGG
```

To navigate in this folder

- Now you can create a new app from the terminal.

If you want to create a app, name “SVO”

```
otree startapp SVO
```

- You will notice that a new app named “SVO” is listed on your Pycharm
- If you don’t see a models.py in each folder, that means you are using the new no-self format.
- If you open the app, you can see that it contains three different types of files:
  - _init.py
  - MyPage.html
  - Results.html

If you open the **init.py** you can find several different class.

You can edit it based on your game. Two most important classes are:

- class (Constants)
- class (Player)

Everything we want to analyze and store in data should be stored in Player.

3 The building blocks of oTree

Once you have installed oTree and created an app you are ready to build your own experiment. If you examine an experiment, you will see that from a programming-perspective an experiment consists of some things. These can be seen as the four building blocks of oTree.

1. **Templates**

Every page in an experiment is a template with a certain lay-out and text. Templates are .html files in which you can use HTML, CSS, Javascript, and Django.

2. **Models.py**

In **Models.py** you define the variables that will be stored in your data-output. For instance, if you want to ask the age of a participant you create a variable:

```
age = models.IntegerField(label="What is your age?")
```

Here, a model is a database table and a field is one column within that table. In this case, an *IntegerField* is used, indicating that only integer numbers are valid. The label determines the text that is shown to participants next to the variable. There are six *StringField* that you can use:

- *IntegerField* for integer numbers
- *FloatField* for numbers that may include decimals
- *StringField* for strings of text
- *LongStringField* for long strings of text
- *BooleanField* for True/False values
- *CurrencyField* for currency amounts

3. **Pages.py**

In *Pages.py* you define every page that is shown to participants and specify which variables are allocated to that page. For instance:

```
class Instructions(Page):
    form_model = 'player'
    form_fields = ['Comprehension1', 'Comprehension2']
```

Here, Instructions is the name of the page, and form_model refers to the level at which the variables should be stored. form_fields includes a list of variables that can be elicited on the page. In this case, we include two comprehension checks that are further defined in Models.py. Next to defining and specifying variables for every page, Pages.py is also used to determine the page sequence.

```
page_sequence = [Instructions, Main_Page, Results]
```

Moreover, there are a battery of functions that can be used to specify conditions for each page. For example, the is_displayed() function determines conditions to which participants the page is displayed.

4. **Settings.py**

The last building block of oTree is *Settings.py*. Here you define some overarching settings of your oTree project, such as the language used, the currency used, your admin username and password, how experimental currency translates to real-world currency, and the apps that are available. For the latter, if you have an app called attribute_substitution. You can make it available in your oTree server by typing:

```
SESSION_CONFIGS = [
    dict(
       name='attribute_substitution',
       display_name="Attribute Substitution Game",
       num_demo_participants=3,
       app_sequence=['attribute_substitution']
    ),
]
```

Here, the name and display name are whatever you prefer (for name you cannot use spaces). The number of demo participants is the number of slots available to play a demo version of the experiment. The most important thing here is the app_sequence. In the app_sequence you define the sequence of apps that is displayed to participants. Suppose that you want to combine the attribute_substitution app with a public goods game or a survey, you can add, public_goods or survey, respectively.

4 Pycharm to customize oTree studio project

oTree Studio is a point-and-click interface for building oTree apps. It is free to use for small/medium sized projects.

4.1 Create project by oTree Hub

- If you are a new, it is recommended to use oTree Studio to create and edit the project instead of hard coding. Please click this link to register an oTree Hub account then login.
- Click the "+ Project" button to create a new project.
- Then you're able to visit the project configuration page where you can config the meta-data for your project.
- Go to the "download" and click the "Download .otreezip" button to download the project.

4.2 PyCharm to customize the project

- After creating and customizing your project on oTree Studio, you can also further customize your project using PyCharm or other IDE. First, you need to unpack your downloaded **.otreezip** file. To do this, run:

```
otree unzip xx.otreezip
```

- The command will produce a folder with an identical name. Use PyCharm to open that folder.
- Then, you can add python code to further customize the project. After adding your code, run the following command to re-pack the project into the .otreezip file:

```
otree zip
```

4.3 Host the App locally

- Open a terminal/Command Prompt on your computer and change the working directory to where you store your previously downloaded project file. Execute the following command to start an oTree test server:

```
otree zipserver
```

- Then, you can open http://localhost:8000/ in your browser to test your project out.

5 Deploy on Heroku from oTree

In order to make the oTree instance accessible globally, a server is needed. We use Heroku which is a cloud hosting server that provides a platform for apps where you can easily host your oTree application and open access to the public.

If you run an experiment with a substantial number of participants, you must upgrade to a paid server. As you can immediately scale the server capacity it is not expensive to use. For instance, if you run your experiment in the timeframe of three days, you only pay 3/30th of the monthly fees as you scale it up before you use it and scale it down after you use it. After you have created an account on Heroku, download and install the Heroku CLI. After installing the Heroku Command Line Interface (CLI), there are three ways to deploy experiments to Heroku:

1. oTree Hub
2. GitHub
3. Terminal

5.1 Connecting oTree to Heroku

5.1.1 oTree Hub

oTree Hub is the easiest way and requires least coding. However, convenience comes at a cost as you can have limited free project space and everything you deploy using oTree Hub becomes publicly available immediately, unless you pay.

5.1.2 GitHub

GitHub also does not require programming and you can even auto-deploy your experiment every time you commit changes to GitHub. If you have a GitHub account, you can integrate this with Heroku. In Heroku, create an app and click on "Deploy". Here, you can authorize Heroku to access your GitHub repositories. After authorizing Heroku, you can search for your GitHub repository and connect it to Heroku. You can either manually deploy your GitHub repository to Heroku or automatically deploy all pushes you make to GitHub. If you add new

variables in your experiment, you should reset the database in Heroku by using the following code in the Command Prompt.

```
heroku login
heroku run "otree resetdb" --app your_appname
```

5.1.3 Terminal

You can use the command line (or Terminal for iOS users) to deploy your experiment to Heroku. In this case, you do not need any third-party software except Heroku. To use the command prompt you need to follow the following steps:

- Locate the project root folder (particular oTree instance).

```
cd '/Users/Name/Folder/'
```

- If you have created a Heroku account, use the following command. If you do not have a Heroku account yet, you can create one here.

```
heroku login
```

- If the above command does not work, you probably have not installed Heroku CLI properly.
- Then, you can create a new Heroku-app, if you don't have one yet.

```
heroku create your_appname
```

- Since you are already in the project root folder you can create the .git here.

```
git init
heroku git:remote -a your_appname
```

- You can push your code locally to Heroku. Finally push the local repository to the Heroku server.

```
git push heroku master
```

- In order to view the Heroku app, enter the following command or enter the URL in your browser.

```
heroku open
```

- If you add new variables in the models.py, you should reset the database in Heroku.

```
heroku login
heroku run "otree reset db" --APP your_appname
```

- Or do it manually in the Heroku app, simlarly to the way described earlier.

6 Models.py

Models.py for calculations, logic of the game. Responsible for processing related to data access. Perform interactions between web apps and databases. In oTree, `models.py` mainly plays that role. As an image, it deals with "definition of necessary data" and "calculation (logic)"

determine the variables in the experiment

- Repeat for how many periods?
- How many players?
- What variables affect all players?
- What variables can individual players decide?
- What kind of calculation do you want it to do?
- Create a function.

6.1 Constants

```
class Constants(BaseConstants):
   name_in_url = 'public_goods_simple'
   players_per_group = 3
   num_rounds = 1

   endowmwnt = c(100)
   efficiency_factor = 1.8
```

7 Pages.py

Pages.py for logic of the web pages

Responsible for screen display. It defines the order of screen display and the data (input items) to be displayed. In oTree, `pages.py` (`views.py`) mainly plays that role. As an image, it shows "what data is displayed (What)"

Decide what to do when each page is displayed.

- where do you run the function?
- On which page do you enter certain variables?
- In what order do page transitions occur?

8 Template.html

Templates for web pages that the participant sees, e.g. Instructions, WaitPage, Results participant inputs.

Responsible for screen display. The part for creating the web page displayed on the screen. In oTree, each html file in the `template` folder mainly plays that role. As an image, it shows "How to display data".

create a screen in html format

- Where to output characters.
- Where to create the input field.
- Where to output the calculation results.

9 Settings.py

To implement experiments with oTree, you need to register your app in `SESSION_CONFIGS` in `settings.py`.

You can also register multiple apps at once here.

10 Questionnaire survey with oTree

10.1 Outline of the experimental program to be created:

- Questionnaire (single player)
  - Question item
    - Gender (radio button)
    - Age (optional)
    - Prefecture of residence (optional)
    - Device used for answering (select)

10.2 Create an app

Create a base application

```
otree startapp questionnaire
```

10.2.1 Defining the Constants class: basic design

- Open models.py in questionnaire folder
- Set the number of people, the number of repetitions, the initial holding amount, and the coefficient in the Constants class.

```
class Constants(BaseConstants):
    name_in_url = 'questionnaire'
    players_per_group = None
    num_rounds = 1
```

Notes:

- For `players_per_group`, enter “None” for 1 player.
- For `num_rounds`, enter “1” to ask only once.

- `names_in_url`, `players_per_group`, `num_rounds` are defined on oTree, so it is not desirable to use them as arbitrary variable names.

10.2.2 Subsession class definition

- Since there is no interaction between players this time, it is not defined.

```
class Subsession(BaseSubsession):
    pass
```

10.2.3 Group class definition

- Since there is no interaction between players this time, it is not defined.

```
class Group(BaseGroup):
    pass
```

10.2.4 Player class definition

- Define variables for each player in the Player class

```
class Player(BasePlayer):
    q_gender = models.CharField(initial=None,
                                choices=['male', 'female', 'no answer'],
                                verbose_name='What is your gender? ',
                                widget=widgets.RadioSelect())

    q_age = models.PositiveIntegerField(verbose_name='What is your age?',
                                        choices=range(0, 125),
                                        initial=None)

    q_prefecture = models.CharField(initial=None,
                                choices=['Hokkaido', 'Aomori', 'Iwate', 'Miyagi', 'Akita',
                                verbose_name='What area do you live in? ',
                                widget=forms.Select())

    q_device = models.CharField(initial=None,
                                choices=['computer',
```

```
                                        'Tablet',
                                        'smartphone',
                                        'other than that'
                                        ],
                                verbose_name='By which electronic device are you answering
                                widget=forms.Select())
```

10.3 Defining templates:

- In templates, we will decide on the page to display specific items.
- Create an html file called `Page1.html` in `questionnaire/templates/questionnaire`
  - You just need to rewrite the existing `MyPage.html`

10.3.1 pages

```
{% extends "global/Page.html" %}
{% load otree static %}

{% block title %}
    Questions:
{%endblock%}

{% block content %}

<div>
    <p>
        Please select the most appropriate (closest) option for the following questions. <
    </p>
</div>

<div style="background-color:#e6e6e6;">
    {% formfield player.q_device %}
    {% formfield player.q_gender %}
    {% formfield player.q_age %}
    {% formfield player.q_prefecture %}
</div>
```

```
    {% next_button %}

{%endblock%}
```

10.3.2 easy explanation

```
{% block title %}
    Questions:
{%endblock%}
```

- Set the “Title” in the web page.

```
{% block content %}
    ...
{%endblock%}
```

- Write the actual question items in this.

```
  {% formfield player.q_device %}
  {% formfield player.q_gender %}
  {% formfield player.q_age %}
  {% formfield player.q_prefecture %}
```

- Write `{% formfield [defined in models] %}` for an item that requires some kind of input.
  - In this case, the input defined in the player class is required, so it is written as `player.`.

```
  {% next_button %}
```

- This is the “Next” button.
  - Click this button to go to the next screen.

10.4 definition of pages:

- In `pages.py`, set the `page display order'',`input items‘‘, ** “function calculation order”**, etc.
  - Actually, the “order of function calculation’ ’ is important.
- However, there is no function calculation this time, so don’t worry about it.
- Behavior set in `pages.py`
  - Define the contents of `Page1.html`.
    - Display the items to be entered.

10.4.1 About Page1

- On `Page1`, there is an input of the answer to the question item.
  - Let’s make an input screen.

```
class Page1(Page):
    form_model = 'player'
    form_fields = [
    'q_gender',
    'q_age',
    'q_prefecture',
    'q_device'
    ]
```

10.4.2 Define the display order

- Defines the order in which the screen is displayed at the end.

```
page_sequence = [Page1]
```

10.5 Defining session configs in setting:

- To implement experiments with oTree, you need to register your app in `SESSION_CONFIGS` in `settings.py`.

```
SESSION_CONFIGS = [
    dict(
        name='questionnaire',
        display_name="Initial Questionnaire",
        num_demo_participants=1, # Here we need to define how many people will participate
        app_sequence=['questionnaire']
    ),
]
```

10.6 start as server

- Start your own terminal as a server

```
otree devserver
```

- Now you can run the experiment on your own terminal.
- Access http://localhost:8000/.

11 Conclusion

11.1 oTree-organizing idea:

It's easier to understand if you think about the program while thinking about this

- how many players?
- What are the variables that affect everyone?
- What variables are determined by individual players?
- How many pages to create?
- What kind of page transition is performed?

References

1. 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.

2. Fischbacher, U. (2007). z-Tree: Zurich toolbox for ready-made economic experiments. *Experimental Economics*, 10, 171-178.