oTree: An open-source platform for lab, web, and field experiments

Daniel L. Chen, Martin Schonger, Chris Wickens

www.oTree.org
What is oTree?

A platform for lab, web and field experiments

- Open source - easy to extend & customize
- Online (runs in browser) - no installation, any device
- Object oriented programming - easy & versatile
Background

- oTree started in 2013 (.. 2008 MTurk experiments)
- Created it for our own research (modular)
- We were not satisfied with any of the existing tools
  - Ease of use
  - Flexibility
- Decided to release oTree publicly in hopes others would find it useful
4 usage scenarios

1.) Lab
   - Can run in any lab
   - Multiple labs simultaneously

2.) Online
   - Hybrid approaches possible (lab + online)
   - Amazon Mechanical Turk integration

3.) Field
   - Internet access not necessary
   - e.g. remote village – bring 30 tablets and server laptop

4.) Classroom
   - Post a URL and have students play on their own devices
   - Play live in class
   - Also could be used to grade homework, administer surveys etc.
   - Use instead of clickers to flip the classroom
Who’s using oTree?

• Academic
  • Armin Falk (1000 participants)
  • John List
  • IBSEN: largest ever public goods game

• Government
  • IRS
  • Consumer Financial Protection Bureau
  • Korea Labor & Income Panel (900 participants)
  • Singapore Life Panel (3000 participants)

• Industry
  • GfK (top 5 market research company)

• 3rd party tools
  • oTree Virtual Machine Manager
  • Continuous time matrix games
  • Continuous time auctions
  • Network games

• 600 people on discussion list, 1000 discussion topics

• Hundreds of universities
Demo

  - from oTree users
Sample games

22 simple games (with source code) available at: demo.otree.org

- Demo Game
- Public Goods Game
- Prisoners’ Dilemma
- Trust Game
- Dictator Game
- Cournot Competition
- Bertrand Competition
- Stackelberg Competition
- Common Value Auction
- Private Value Auction
- Volunteer's Dilemma
- Principal Agent Game
- Stag Hunt
- Battle of the Sexes
- Coordination Game
- Matching Pennies
- Traveler's Dilemma
- Survey
- Divide a Pie
- Guessing Game
- 2 x 2 Matrix Game (Symmetric)
- 2 x 2 Matrix Game (Asymmetric)
Demo

- Experimenter mode
  - public goods game
  - participants contribute
  - see status updates
    - export data in CSV
Amazon Mechanical Turk

- Publish to MTurk through oTree’s admin interface
- Automatically pay workers the bonus they earned in your game
- Filter workers based on location, skill level, etc.
Use charts

**Stacked bar chart**

- Apples
- Oranges
- Pears
- Grapes
- Bananas

**Monthly Average Temperature**

Source: WorldClimate.com

- Tokyo
- New York
- Berlin
- London
Waiting rooms

- See how many participants are ready, then create a session of appropriate size
- **Classroom**: each student gets a unique permanent URL
- **Lab/field**: each PC has a unique permanent URL
- **Spontaneous live**: give the audience an easy-to-type link like http://mysite.com/rooms/demo
Configurable sessions
Customizable experimenter dashboard

App name: lemon_market (3 rounds)
Round number: 3

Lemon market results

Results

Points

Round

Transaction Price
Earnings for Buyer
Earnings for Seller 1
Earnings for Seller 2
Live chat

- Live chat between participants
- Configurable
HTML5 user interfaces

- Radio buttons, dropdowns, money input
- Pictures, tables, formatting, video/audio input/output
- Localization: supports any language or currency
  - a tool to translate for co-authors
  - multi-national capability
Dynamic elements

Your choice

The other player was selected to give money first, and chose to give €0.15 of the initial sum. The experimenter tripled this amount; you have therefore received €0.45 in addition to your initial €1.00.

Please select how much of the now-tripled amount you wish to give back to the other player.

How much would you like to give back?

- €0.00
- €0.05
- €0.10
- €0.15
- €0.20
- €0.25
- €0.30
- €0.35
- €0.40
- €0.45

Results for Rounds 1–3

This is a summary of the rounds played.

<table>
<thead>
<tr>
<th>Round</th>
<th>Player and outcome</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>You were Player 1 and won</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>You were Player 1 and lost</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>You were Player 2 and won</td>
<td>100</td>
</tr>
</tbody>
</table>

You earned 200 points. In addition to that, you get a participation fee of 50 points.

In total, you receive 250 points.

To complete the study, please answer the questionnaire that will now follow.

Instructions

In this study you will play three rounds of “Matching Pennies.”
Automated testing with bots

- Bots can simulate hundreds of participants (deterministic or Monte Carlo)
- Detect programming errors as well as design errors (e.g. negative payoffs)
- Saves time by reducing manual testing
- Bots run from command line or web browser
- Load testing the number of participants

```python
class PlayerBot(Bot):
    def play_round(self):
        yield (views.Contribute, {'contribution': c(1)})
        yield (views.Results)
```
Group matching

groups = [[P, P],
          [P, P],
          [P, P],
          [P, P],
          [P, P]]

for group in groups:
    group.reverse()

2-player game, 5 simultaneous groups

Groups are represented as a matrix, which you can permute any way you want

Example: P1 and P2 swap roles
Group matching

```python
>>> self.get_group_matrix()

[[<Player  1>, <Player  2>, <Player  3>],
 [ <Player  4>, <Player  5>, <Player  6>],
 [ <Player  7>, <Player  8>, <Player  9>],
 [ <Player 10>, <Player 11>, <Player 12>]]

>>> self.group_randomly(fixed_id_in_group=True)

[[<Player  1>, <Player  8>, <Player 12>],
 [ <Player 10>, <Player  5>, <Player  3>],
 [ <Player  4>, <Player  2>, <Player  6>],
 [ <Player  7>, <Player 11>, <Player  9>]]
```
Export data in CSV format

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>visited</td>
<td>index</td>
<td>bonus</td>
<td>contributed</td>
<td>deduction_noncontrib</td>
<td>deduction_contrib</td>
<td>deduction_1_contrib</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>420</td>
<td>1</td>
<td>-3</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>300</td>
<td>1</td>
<td>-8</td>
<td>-9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>260</td>
<td>1</td>
<td>-11</td>
<td>-4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>420</td>
<td>1</td>
<td>-5</td>
<td>-6</td>
</tr>
</tbody>
</table>

Documentation extracted from comments in source code

```plaintext
index_among_participants_in_match
type positive integer
doc Index starting from 1. In multiplayer games, indicates whether this is participant 1, participant 2, etc.

bonus
type positive integer
doc The bonus the participant made in this subsession, in cents

deduction_to_each_noncontributor
type integer
doc How much to deduct from each, if neither contributes. For strategy method, all 4 variables are populated. For direct response, we don’t populate all 4. Only the one(s) that actually correspond to how the game was played. For example, if P1 contributed but not P2 or P3, then from P1’s perspective, neither of the other participants contributed.
```

Data analysis separate from programmer
Modern programming

- Python is easy to learn, popular, and versatile
  - creates the dynamic elements
  - most popular language for intro to computer science/programming
  - many libraries (e.g. statistical tests or network analyses)

- Good skill investment
  - web-scraping, text processing, NLP, data analysis, machine learning

- Can recruit developers with Python/Django expertise
  - undergrads or oDesk/upWork
3 Code examples

• 3 examples of how oTree solves a problem
• Benchmark to z-Tree
• z-Tree solutions taken from answers provided on mailing list
Example 1: Are preferences monotonic? check if list is sorted smallest→largest

z-Tree

\[
\text{iterator}(i, 10).\text{sum}(\ \text{iterator}(j, 10).\text{count}( :i<j \ \& \ \& \text{risk}[ :i ] > \text{risk}[ j ] )) ==0
\]

oTree

\[
\text{values}==\text{sorted(values)}
\]
Example 2: randomly shuffle a list
A randomized set of choices to avoid bias

**z-Tree**

```
iterator(i, size_array - 1).do {
    address = roundup(random() * (:size_array + 1 - i), 1);
    if (address != :size_array + 1 - i) {
        temp = :random_sequence[ :size_array + 1 - i ];
        :random_sequence[ :size_array + 1 - i ] = :random_sequence[ address ];
        :random_sequence[ address ] = temp;
    }
}
```

**oTree**

```
random.shuffle(random_sequence)
```
Example 3: Select 3 random periods for payment
(random lottery incentive)

```python
if (Period == 1) {
    array totalRounds[numRounds];
    i = 1;
    while (i <= numRounds) {
        totalRounds[i] = i;
        i = i + 1;
    }
    i = 1;
    array_end = numRounds;
    while (i <= numPayRounds) {
        rand = round(random()*(array_end-1), 1) + 1;
        RandomRound[i] = totalRounds[rand];
        k = if (rand != array_end, rand+1, rand);
        while (k <= array_end) {
            t = if (rand != array_end, k-1, k);
            totalRounds[t] = totalRounds[k];
            k = k + 1;
        }
    }
}
```

if (self.subsession.round_number == Constants.num_rounds):
    random_players = random.sample(self.in_all_rounds(), 3)
    self.payoff = sum([p.theoretical_payoff for p in random_players])
Demo mode

- Easy to put your game online (play in browser, no installation or download)
- Send link to co-authors, referees, students
- Example at demo.otree.org
# oTree Payments

Oct. 7, 2014

## Session

<table>
<thead>
<tr>
<th>Name</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>redalco</td>
</tr>
<tr>
<td>Base pay</td>
<td>$10.00</td>
</tr>
</tbody>
</table>

## Participants

<table>
<thead>
<tr>
<th>Name</th>
<th>Bonus</th>
<th>Total pay</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$5.05</td>
<td>$15.05</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$7.30</td>
<td>$17.30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$3.80</td>
<td>$13.80</td>
<td></td>
</tr>
</tbody>
</table>

| Total | $16.15 | $46.15 |

## Notes/Signature

-
Documentation


**oTree**

**Live demo**

http://demo.otree.org/

**Homepage**

http://www.otree.org/

**About**

oTree is a framework based on Python that lets you build:

- Multiplayer strategy games, like the prisoner's dilemma, public goods game, and auctions
- Controlled behavioral experiments in economics, psychology, and related fields
- Surveys and quizzes

**Support**

For help, post to our Google Groups forum.
Discussion Forum

https://groups.google.com/forum/#!forum/otree
Integration

- Rest API (automates/integrates oTree with other web services)
- Plugin framework (so people can write add-ons, like a chat widget)
Why should you pick oTree?

• **One platform**
  - Lab, online, field, classroom
  - Desktops, tablets, smartphones
  - Games, surveys

• **Flexible**
  - The **easiest** general-purpose programming experience

• **Modern user interface**

• **Stable & proven**

• **Bots** for automated testing

• Excellent **long-term viability** (open source, large community, solid technical design)
Future work

- Network games (network topology)
- Measuring reaction time, mouse movements, keystrokes
- Integration of physiological data (pupil dilation, eyetracking, fMRI)
- Continuous-time games (markets)

Goal
- Building a game only requires expressing the logic of the game
- Abstract away the details a web developer usually has to think about
- Best way to learn is to just jump in
- “Bicycle for the mind”
Future work

• Open Science
  • Code and data repository (like the demo page)
  • Searchable, time stamped, allow Bayesian/meta analysis
  • Develop standards and defaults that reduce the cost of creating accessible data
Future work

• Education
  • RCT impact on integration, recognition and respect, digital literacy
    • Teaching social preferences
    • Instill scientific curiosity in the behavior and preferences of others
    • Statistical concepts
    • Programming literacy (e.g., code.org)
      • 15 minutes to program a public goods game “app”
  • Entrepreneurship
  • Design thinking
    • Empathy
  • Integration
    • Economic, civic, reasoning with others
  • Tools for students to construct their own knowledge
Legitimacy and Perceived Indifference

Justice: equal treatment before the law \( (y = f(X) + \varepsilon, a \rightarrow X) \)

equality based on recognition of difference
\( (y \perp W, \text{var}(\varepsilon) \perp W, a \rightarrow W) \)

Sympathy and Empathy

“settings where people are closer to indifference among options are more likely to lead to detectable effects [of behavioral biases] outside of it.” (Sohn 2011)
A Theory of Surveys

Measurement
- Talk is cheap
  - Trump, Brexit, Colombia peace vote—all mispredicted
  - Sophisticated adjustments of polls still failed

Model
- Make costly the expression of moral and ideological beliefs in surveys
- Revealed preference heuristic
  - Marginal benefit of an additional “vote” scales linearly, so should the marginal cost
  - Implies quadratic costs $\sum_{i=1}^{N}(v_i^j)^2 = B$

Applications
- Polls, attitudinal surveys, World Value Survey, GSS
- Preference curvature, ideal point estimation
- Decision-making in social & political settings
Paper

- Published paper at: http://www.otree.org/oTree.pdf
- Please cite us if you use oTree for your research

---

oTree
An Open-Source Platform for Laboratory, Online and Field Experiments

Daniel Chen · Martin Schonger · Chris Wickens

September 2014

Abstract oTree is an open-source, online, and object-oriented software platform for implementing social science experiments be they laboratory, online or field experiments or combinations thereof. oTree is cloud-based and does not require installation of software on subjects’ devices. Subjects can be using desktops, tablets or smartphones running different operating systems. This facilitates usage in online and field settings, and allows bring-your-own-device approaches. Not requiring specific hardware also enables replication at low cost by others. Deployment can be internet-based without a shared local network, or conversely local network based without internet access. oTree uses industry standard open-source technologies like HTML5 and Python. With HTML5 the usual internet-range of graphical elements, form inputs, sound, and video can be employed. Python is the most popular programming language for beginners and taught at most universities, which allows researchers to tap into a large pool of programming talent. Creating experiments can be learned quickly, especially by those already familiar with Python. oTree.org offers a library of standard games, which can be used for teaching or as templates for experiments. A player can be simulated with an oTree bot. Bots can simulate thousands of game plays to check game logic and programming. Using bots, even multi-player games can be put online as supplementary, interactive material.

Keywords experimental economics · software for laboratory experiments · software for field experiments · software for online experiments

JEL Codes: C70, C88, C90

---

Center for Law and Economics, DI-GRESS, ETH Zurich, CH-8092 Zurich
www.oTree.org
E-mail: wickens@post.harvard.edu
Preliminary working paper, we apologize for oversights and still lacking citations. To play demo games please visit oTree.org.
Motivational quotes (from users)

• "In the last few weeks I started to learn how to build apps within this great environment and I just love it!"

• “Switching to oTree is like a breath of fresh air thanks to more explicit structure of the configuration files and documentation. I am currently working on transferring the existing experiments made in z-Tree to oTree.”

• "I must say that I've loved oTree so far and that it's a huge improvement over zTree."

• "Things have been going well in the lab. Participants leave a lot of good comments about the interface and game. It also runs very smoothly on my end."

• “Takes one-tenth of the time to program an experiment compared to programming from scratch in php and mysql.”
Thank you

chris@oTree.org
dlchen@oTree.org
Installation

Install Python
   /Applications/Python\ 3.6/Install\ Certificates.command
Install oTree
   pip3 install –U otree
Create project folder
   otree startproject oTree
   cd oTree
   otree devserver
Browse to http://localhost:8000/
Install PyCharm
Pages & Models

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

Python
  Models.py for calculations, logic of the game
  Pages.py for logic of the web pages
Tutorial

0. Survey

1. Public Goods game – can program in $\leq 15$ minutes

2. Trust game
   two roles: players see different pages

3. Matching pennies
   swap roles, randomize round for payment
Bonus slides
Optional slides
class Constants(BaseConstants):  
    name_in_url = 'public_goods_simple'  
    players_per_group = 3  
    num_rounds = 1  
    endowment = c(100)  
    efficiency_factor = 1.8

class Subsession(BaseSubsession):  
    pass

class Group(BaseGroup):  
    total_contribution = models.CurrencyField()  
    individual_share = models.CurrencyField()

    def set_payoffs(self):  
        self.total_contribution = sum([p.contribution for p in self.get_players()])  
        self.individual_share = self.total_contribution * Constants.efficiency_factor / Constants.players_per_group  
        for p in self.get_players():  
            p.payoff = Constants.endowment - p.contribution + self.individual_share

class Player(BasePlayer):  
    contribution = models.CurrencyField(min=0, max=Constants.endowment)
This is a public goods game with
{"Constants.players_per_group"} players per group,
an endowment of{"Constants.endowment"},
and an efficiency factor of{"Constants.efficiency_factor"}.

{% formfield player.contribution with label="How much will you contribute?" %}

{% next_button %}

You started with an endowment of{"Constants.endowment"},
of which you contributed{"player.contribution"}.
Your group contributed{"group.total_contribution"},
resulting in an individual share of{"group.individual_share"}.
Your profit is therefore{"player.payoff"}.

{% next_button %}