Preliminary working paper.

oTree

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

Daniel Chen · Martin Schonger · Chris Wickens

September 2014

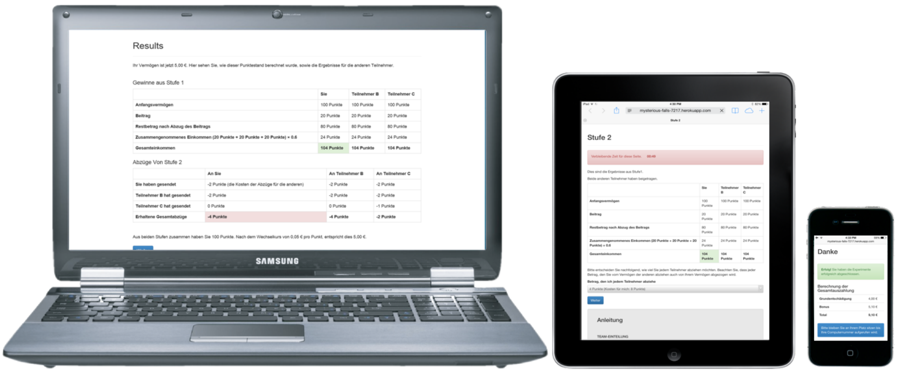
Abstract oTree is an open-source, online, and object-oriented software platform for implementing social science experiments be they laboratory, online or field experiments or combinations thereof. oTree is cloud-based and does not require installation of software on subjects' devices. Subjects can be 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, D-GESS, 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.



Results

Ihr Vermögen ist jetzt 5,00 €. Hier sehen Sie, wie dieser Punktestand berechnet wurde, sowie die Ergebnisse für die anderen Teilnehmer.

Gewinne aus Stufe 1

	Sie	Teilnehmer B	Teilnehmer C
Anfangsvermögen	100 Punkte	100 Punkte	100 Punkte
Beitrag	20 Punkte	20 Punkte	20 Punkte
Restbetrag nach Abzug des Beitrags	80 Punkte	80 Punkte	80 Punkte
Zusammengenommenes Einkommen (20 Punkte + 20 Punkte + 20 Punkte) × 0.6	24 Punkte	24 Punkte	24 Punkte
Gesamteinkommen	104 Punkte	104 Punkte	104 Punkte

Abzüge Von Stufe 2

	An Sie	An Teilnehmer B	An Teilnehmer C
Sie haben gesendet	-2 Punkte (die Kosten der Abzüge für die anderen)	-2 Punkte	-2 Punkte
Teilnehmer B hat gesendet	-2 Punkte	-2 Punkte	-2 Punkte
Teilnehmer C hat gesendet	0 Punkte	0 Punkte	-1 Punkte
Erhaltene Gesamtabzüge	-4 Punkte	-4 Punkte	-2 Punkte

Aus beiden Stufen zusammen haben Sie 100 Punkte. Nach dem Wechselkurs von 0,05 € pro Punkt, entspricht dies 5,00 €.

mysterious-falls-7217.herokuapp.com

Stufe 2

Verbleibende Zeit für diese Seite. 00:49

Dies sind die Ergebnisse aus Stufe1.

Beide anderen Teilnehmer haben beigetragen.

	Sie	Teilnehmer B	Teilnehmer C
Anfangsvermögen	100 Punkte	100 Punkte	100 Punkte
Beitrag	20 Punkte	20 Punkte	20 Punkte
Restbetrag nach Abzug des Beitrags	80 Punkte	80 Punkte	80 Punkte
Zusammengenommenes Einkommen (20 Punkte + 20 Punkte + 20 Punkte) × 0.6	24 Punkte	24 Punkte	24 Punkte
Gesamteinkommen	104 Punkte	104 Punkte	104 Punkte

Bitte entscheiden Sie nachfolgend, wie viel Sie jedem Teilnehmer abziehen möchten. Beachten Sie, dass jeder Betrag, den Sie vom Vermögen der anderen abziehen auch von Ihrem Vermögen abgezogen wird.

Betrag, den ich jedem Teilnehmer abziehe

4 Punkte (Kosten für mich: 8 Punkte)

Weiter

Anleitung

TEAM-EINTEILUNG

mysterious-falls-7217.herokuapp.com

Danke

Erfolg! Sie haben die Experimente erfolgreich abgeschlossen.

Berechnung der Gesamtauszahlung

Grundentschädigung	4,00 €
Bonus	5,10 €
Total	9,10 €

Bitte bleiben Sie an Ihrem Platz sitzen bis Ihre Computernummer aufgerufen wird.

Fig. 1 oTree on different devices and operating systems

1 Introduction

Experimental economics started out with thought experiments and informal experiments, going at least back to the St. Petersburg Paradox proposed by Nicholaus Bernoulli which led Daniel Bernoulli to propose expected utility. In the latter half of the 20th century classroom or laboratory paper-based experiments became common, which with the advent of the PC in the 1980s often became computer-based. In 1998 Fischbacher released z-Tree Version 1 Fischbacher (1998), which has become an ubiquitous tool in experimental economics and served as a role model for many software tools (see Janssen et al. (2014)). Thus oTree is named in homage to this original groundbreaking economics experiment platform.[1] In recent years, there has been a drive to complement or substitute laboratory experiments with field experiments. At the same time, computing has become more mobile and ubiquitous and moved from the desktop into the field. This opens up a whole new array of possiblities for experimental economics, the main purpose of oTree is to cover the entire spectrum of modern computing.

Real-time behavioral measurements in experiments are increasingly seen to complement and explain field data across the social sciences. Extremely large data collections and cross-cultural experiments hold the additional promise to address research questions that were previously difficult to answer. Current experimental tools are limited by physical infrastructure. For example, in developed countries, researchers typically set up a networked computer lab and, in remote areas of developing countries, rely on paper and pencil. We introduce oTree, a cloud-based open-source software platform for implementing experiments in the social sciences, be they laboratory, online, or field (or any combination thereof). Its cloud-based nature means it can be used on any hardware (e.g., desktops, tablets, smartphones) and operating system. oTree

[1] This naming follows in the tradition of naming open-source software projects after existing ones (e.g. OpenOffice (after Microsoft Office),Linux (after Unix), C++ (after C), and MySQL (after mSQL)).

allows a wide range of graphical elements, form inputs, sound, and video. It comes with a large, openly accessible library of standard games, such as public goods games, matrix games, markets, and auctions – that can be used as a starting point for development of one's own experiments. Checking experiments is facilitated with automated testing by oTree bots that simulate thousands of sessions and replicating experiments is facilitated via online posting of supplementary material for broad distribution of experimental protocol.

Harrison and List (2004) propose a taxonomy to partition the continuum from classic laboratory experiments to pure field experiments: conventional laboratory experiments (in a laboratory, standard subject pool), artefactual field experiments (which have a non-standard subject pool), framed field experiments (experiments that are conducted in the field), and natural field experiment (subjects naturally do the experiment without knowing they are participating in one). In addition, there are Web-based experiments such as those run on Amazon Mechanical Turk (see Horton et al. 2011; Dreber et al. 2009). Due to the changed scientific demands (moving away from a desktop-laboratory environment to more natural ones) and possibilities (cheap and ubiquituous mobile computing devices), we felt the need for a system for experimental economics that is platform independent. A system that can be used in the lab, on the web, in the field and in the classroom: oTree. oTree is built on open standards (requiring only knowledge of Python). The new platform will allow lab, online and artefactual field experiments on computers, tablets and smartphones.

oTree is open-source, licensed under an adaptation of the MIT license[2]. We ask that people cite this paper when using it for academic or other publications. It can be downloaded for free at www.oTree.org, and gives the Python source code. Contributions and improvements to the source code are welcome and should be submitted via GitHub.

oTree is based on the Django web application framework; oTree apps are web applications. Once oTree is installed on a web server, apps can be played instantly from any web browser by opening the game's URL, and experimenters can log in to the web-based experimenter console from anywhere to monitor the experiment.

oTree works on any device with a modern web browser, whether it is a desktop, tablet, or smartphone. It works on all major operating systems such as Windows, Mac OSX, iOS, Android, and Linux. The user interface automatically adjusts to the appropriate screen resolution, eliminating the need for every app to have separate designs for desktop and mobile platforms.

[2] See http://opensource.org/licenses/MIT

Fig. 2 Cloud-based architecture

2 Usage scenarios

2.1 Laboratory with desktops

oTree can be used in a research lab and run on the lab's existing computer workstations. No installation of software is necessary, all that is required are web browsers. Though of course it is helpful to set those in a locked kiosk mode.

2.2 Ad-hoc laboratory

A group using oTree does not even require a dedicated experimental lab, since the experimenter can easily transport several dozen tablets to any space that can accommodate the participants (such as a classroom or meeting room) and set up a temporary lab. oTree can be run on low cost devices such as commodity tablets and mobile devices, reducing hardware costs.

Another advantage of not requiring a desktop operating system is less time and money spent on maintenance and administration. Desktop operating systems offer great flexibility, but they also require investments in IT tasks such as software installation (e.g. applications, drivers, antivirus, and updates thereof) and configuration (e.g. security permissions and networking). In contrast, tablets and mobile operating systems tend to require less administration, and can often run "out of the box" with minimal configuration.

2.3 Online

oTree experiments can be conducted online. Participants just need to visit the experiment's URL to play. oTree has special integration with Amazon Mechanical Turk's payment system.

oTree does not have any set limit on the number of participants who can play simultaneously, and can be used for experiments with hundreds of participants. As is the case for websites in general, running an experiment with many users requires a server with sufficient resources (processor, RAM, and database).

2.4 Lab/Online hybrid

Experiments can be run in multiple labs around the world simultaneously. Each participant may be paired with a participant at a different lab, to ensure anonymity.

2.5 Field

oTree can be used in any off-campus field setting. With a rolling suitcase containing smartphones or tablets, one can set up a temporary lab in locations where subjects may be easier or cheaper to recruit (like malls or airports), or where one can recruit participants are interesting from a research perspective, outside the traditional WEIRD demographic (Henrich et al., 2010) . For example, an experimenter could take the mobile lab to a remote village in a developing country.

2.6 Classroom

oTree has can be used in the classroom as an aid to illustrate concepts in economics, philosophy, psychology, or other social sciences. Lecture can be made more vivid by having students play the game, and with the plug-in highcharts it is possible to live display the results on the overhead. With oTree, students play the game on whatever device they have at hand, whether it is a smartphone, tablet or laptop. oTree can be used to teach game theory and experimental economics as Rubinstein (1999) suggests, or to flip the classroom as Mazur (2009)does.

By the same token, oTree can be used for educational purposes in non-academic settings, such as institutions that need to train their employees about economics, psychology, or business concepts. Other software programs for economics experiments are currently being used in seminars and training sessions, but it is often costly and cumbersome to set up the workstations.

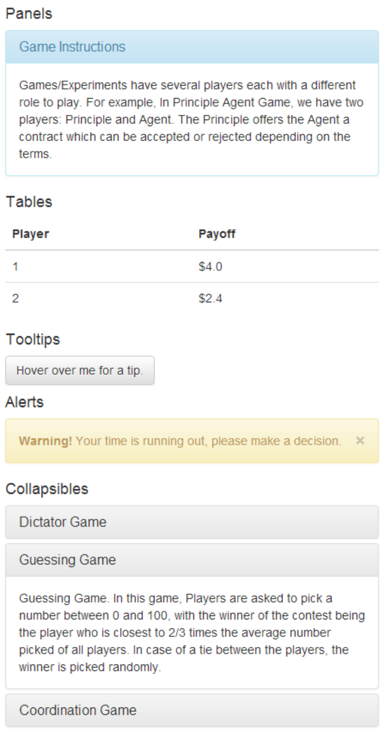
3 User interface

3.1 Participant interface

For the look and design of the user interface we rely on twitter-bootstrap[3], a free library for creating user interfaces for websites. Any HTML5 element can

[3] http://getbootstrap.com/

be included, such as tables, images, headings, hyperlinks, and dialog boxes. The figures show a few examples.


Panels

Game Instructions

Games/Experiments have several players each with a different role to play. For example, In Principle Agent Game, we have two players: Principle and Agent. The Principle offers the Agent a contract which can be accepted or rejected depending on the terms.

Tables

Player	Payoff
1	$4.0
2	$2.4

Tooltips

Hover over me for a tip.

Alerts

Warning! Your time is running out, please make a decision. ×

Collapsibles

Dictator Game

Guessing Game

Guessing Game. In this game, Players are asked to pick a number between 0 and 100, with the winner of the contest being the player who is closest to 2/3 times the average number picked of all players. In case of a tie between the players, the winner is picked randomly.

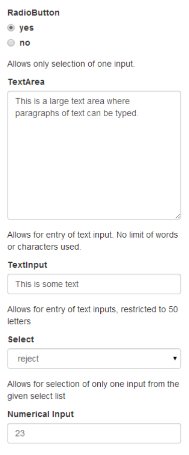
Coordination Game

Fig. 3 HTML5 static content

Dynamic content: video, audio, visualizations, applets oTree supports embedding of various dynamic elements via standard HTML5 embedding. This includes video, audio, JavaScript, Java, and Flash.

3.1.1 Input

Support for standard data types Since oTree's user interface is coded with HTML5, it supports all the standard form controls, such as dropdown menus, radio buttons, checkboxes, numerical entry, free text entry, date/currency input, email field, file upload and image field. For this we rely on the Python web application framework Django. Custom form widgets, for example horizontal sliders or Likert scales, are being implemented but can be created by oTree users as well. Figure 4 is an example.



RadioButton

● yes
○ no

Allows only selection of one input.

TextArea

This is a large text area where paragraphs of text can be typed.

Allows for entry of text input. No limit of words or characters used.

TextInput

This is some text

Allows for entry of text inputs, restricted to 50 letters

Select

reject

Allows for selection of only one input from the given select list

Numerical Input

23

Fig. 4 Input Forms

Tolerance to user error If a participant submits an invalid value in a form, the form is automatically re-displayed, highlighting and explaining the user's error. If a participant accidentally clicks the back button, or tries to skip to the next page without completing the current one, they are redirected back to the current page.

3.1.2 Mobile devices

oTree can be used on any device with a modern web browser. The user interface scales automatically to screens of different sizes, and interaction can be done with mouse and keyboard, or with a touch screen.

3.1.3 Visual style

oTree comes out of the box with a neutral-looking user interface, but this can be styled with CSS.

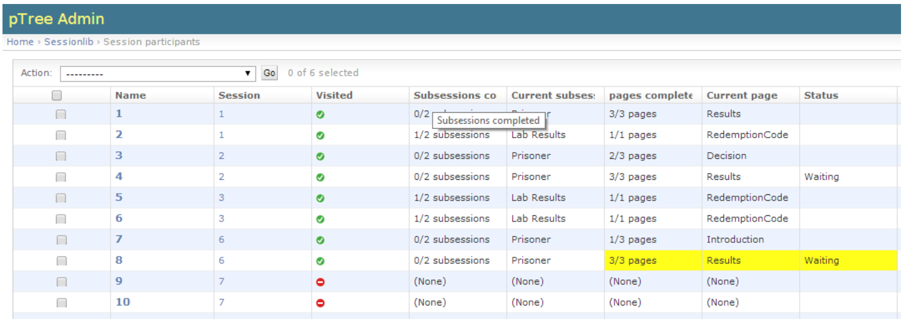
3.2 Experimenter interface

3.2.1 Deploying experiments via URLs

oTree does not require admin rights on the participants' devices. One simply needs to open the participant's device to a persistent URL, which when opened, will route the participant to the currently active experiment.

3.2.2 Live updating experimenter console

oTree has an experimenter console that allows the experimenter to monitor the progress of the experiment. It features a display that can be filtered and sorted.



pTree Admin

Home › Sessionlib › Session participants

Action: --------- Go 0 of 6 selected

Subsessions completed

	Name	Session	Visited	Subsessions co	Current subses:	pages complete	Current page	Status
☐	1	1	✓	0/2	r	3/3 pages	Results	
☐	2	1	✓	1/2 subsessions	Lab Results	1/1 pages	RedemptionCode	
☐	3	2	✓	0/2 subsessions	Prisoner	2/3 pages	Decision	
☐	4	2	✓	0/2 subsessions	Prisoner	3/3 pages	Results	Waiting
☐	5	3	✓	1/2 subsessions	Lab Results	1/1 pages	RedemptionCode	
☐	6	3	✓	1/2 subsessions	Lab Results	1/1 pages	RedemptionCode	
☐	7	6	✓	0/2 subsessions	Prisoner	1/3 pages	Introduction	
☐	8	6	✓	0/2 subsessions	Prisoner	3/3 pages	Results	Waiting
☐	9	7	⊖	(None)	(None)	(None)	(None)	
☐	10	7	⊖	(None)	(None)	(None)	(None)	

Fig. 5 Experimenter console

The experimenter can see the progress of all participants, the current action a participant is taking, and decisions made by participants during gameplay. Updates are shown as they happen in real time by highlighting and fading out cells that change. Data is displayed in a compact table.

Since the console is web-based, it can be monitored in real time by multiple collaborators at in-house or remote locations.

3.2.3 Experimenter input

oTree allows experimenters to interrupt or insert inputs at any time during the study. For example, if researchers want to use a physical urn in an Ellsberg experiment, rather than a virtual one, they can do so and input the color of the ball drawn using the experimenter input feature.

3.2.4 Payments page

After a session is complete, the experimenter can print a page that shows how much to pay each participant.

4 Implementing an experiment

4.1 Library of games

oTree comes with a library of dozens of standard economics games. To see these games and play them go to demo.oTree.org, there is no installation required, just web browser. These games can either be used as-is, or modified as desired. Those who are developing new games from scratch are encouraged to review existing games in the library for examples of recommended coding style.

4.2 Simple programming with Python and HTML5

4.2.1 Game logic

Game logic is programmed in Python, the most popular language used in introductory computer science courses.[4]

oTree is designed to be simple and to be accessible to people with basic programming experience. The quickest way to get an idea of the difficulty level of oTree programming is to browse the source code of various games at github.com/oTree-org/otree. Programming oTree experiments requires a basic understanding of general programming concepts. Prior experience with Python is the most relevant; however, those who have experience with basic object-oriented programming in a language like Java, JavaScript, C#, or C++ should quickly be able to learn oTree. Prior web development experience is not necessary, but the programmer should have a conceptual understanding of how websites work.

4.2.2 Modern programming experience

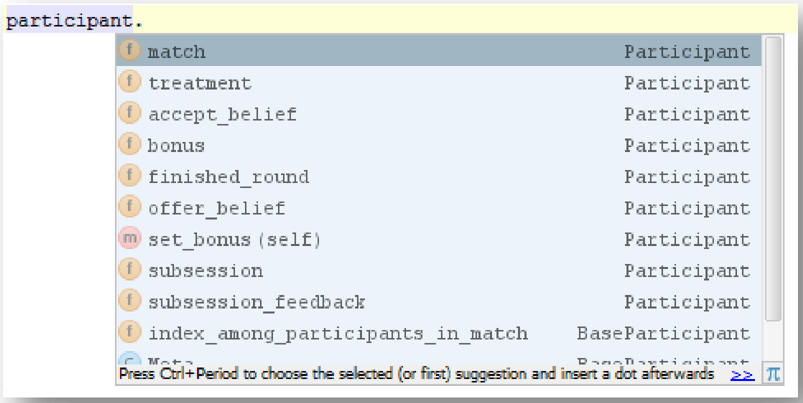
Editor assistance It is recommended to use a modern code editor (such as PyCharm, which has a free license for use in classrooms). These editors have features that make programming easier, such as live error checking, syntax highlighting, interactive debugging (to test the effect of each line of code), and code reorganization functionalities (such as the ability to easily rename a variable). oTree integrates with editors' autocomplete functionality to provide suggestions of allowed variable names as the programmer types code as shown in figure 6.

Informative error pages If an error is encountered during execution, an error page is shown pinpointing the source of the error. In figure 7 an error in line 17 is highlighted to the programmer. In this case that a division by zero is occurring.

4.3 Automated test bots

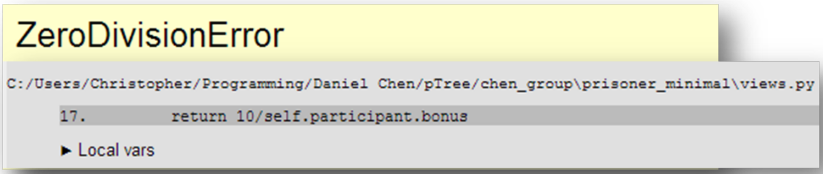
oTree also includes an automated test system, which is an essential component of any application development toolset. App developers can easily program "bots" that simulate participants simultaneously playing in a live experiment. Bots can be programmed to simulate playing the game according to an ordinary strategy, to test "boundary conditions" (e.g. by entering invalid input to see if the application correctly rejects it), or to enter random input on each

[4] See http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext



```
participant.
  f match                                      Participant
  f treatment                                  Participant
  f accept_belief                              Participant
  f bonus                                      Participant
  f finished_round                             Participant
  f offer_belief                               Participant
  m set_bonus(self)                            Participant
  f subsession                                 Participant
  f subsession_feedback                        Participant
  f index_among_participants_in_match      BaseParticipant
Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >> π
```

Fig. 6 Autocompletion while writing code



```
ZeroDivisionError
C:/Users/Christopher/Programming/Daniel Chen/pTree/chen_group\prisoner_minimal\views.py
    17.        return 10/self.participant.bonus
    ▸ Local vars
```

Fig. 7 Informative feedback to programmer

page. Tests with dozens of bots complete with in seconds, and afterward automated tests can be run to verify correctness of the app (e.g. to ensure that payoffs are being calculated correctly).

This automated test system is a major advantage for oTree. It saves the programmer the time and frustration of having to re-test the application every time something is changed.

4.4 Flexibility

4.4.1 One programming model for all types of experiments

oTree has a simple and generic conceptual model that is well suited for experiments in economics, psychology, or any other behavioral science. It tries to avoid features, concepts, and terminology that are specific to a particular discipline or research paradigm.

An experimenter desiring to implement an experiment consisting of a multiplayer game followed by a questionnaire would create two oTree apps. Each app consists of a sequence of discrete web pages. After participants fill out input fields on a page, they click a submit button and are taken to the next page in the sequence. Each page can contain a rule determining whether it should be shown or skipped, and waiting pages can be used when some participants need to wait for others to complete an action before proceeding. At the end of the sequence of pages, the participant can be awarded a dynamically calculated monetary bonus.

4.4.2 Customizable randomization and matching

Participants in a lab session can separated into groups. First, there are treatment groups that allows participants playing concurrently to be exposed to different experimental parameters. Second, there are match groups, which are for experiments that involve interaction between two or more participants (like a prisoner's dilemma or public goods game). If a game is to be played for multiple rounds, participants can either play with the same partners, or with new partners. oTree provides several standard matching algorithms. Third parties can define custom matching algorithms in a straightforward way.

4.5 Collaborative development

oTree comes integrated with Git, a source code versioning system. This encourages good backup and versioning practices, and allows developers to synchronize files across computers, develop collaboratively, manage separate branches, and merge synchronization conflicts.

4.6 Hiring a programmer

Some researchers may prefer paying a programmer to create the experiment, rather than programming it themselves. In this situation, prior experience with oTree is not necessary. For best results, one should find a developer who has experience with the Django web development framework. Django is the most popular Python-based web development framework, and since oTree is based on Django, a Django developer can be productive in oTree within hours.

5 Documentation and Reproducability

5.1 Data recording and export

After an experiment is run, data can be downloaded from the web interface in standard CSV format. oTree auto-generates documentation from the app's source code, to explain to the data analyst the data type and meaning of each field in the CSV file. It doesn't lose any data, even if a computer is restarted or crashes.

5.2 Demo Mode

oTree has a “demo mode” where any experiment can be put online at a static URL so that colleagues, referees, and other readers can try the experiment in their browsers. demo.otree.org itself is an example of demo mode being used.

5.3 Reproducibility

Each time an experiment is run, the server records what version of the code was used to run the experiment (using an auto-generated hash code). Every version of oTree is archived forever, and a programmer can revert to an old version of the code by running 2 commands.

5.4 Localization

Of course oTree applications can be translated to multiple languages. Multiple features support translation and localization. A setting can be switched to control which language the app is currently displayed in. This is useful in many scenarios. For example, an experimenter can deploy the same experiment in the US, Germany, and China with all participants reading in their native tongue. Or one can deploy an experiment in German but send English-speaking colleagues a demo link in English and publish English screenshots in a paper reporting the experiment. oTree is able to display monetary amounts in any currency format (e.g. “$5.00” vs “5,00 CHF”). An application developer simply needs to switch to a global code setting and their apps update everywhere.

6 Limitations and sustainability

6.1 Limitations

oTree’s interaction model is based on a sequence of pages, each containing an optional form for the participant to fill out and submit. This interaction model does not suit every use case. It is not real-time, in the sense that the user interface does not respond within milliseconds to actions from other participants. Although it has support for continuous-time gameplay, it is more oriented toward a discrete-time model. Although its HTML user interface supports embedding of any kind of dynamic content like Flash or Java games, oTree is not a video game platform.

6.2 Sustainability

oTree is built to last over the next decade and beyond. The source code is highly organized and follows best practices, making for easy further development and extension. Features are added sparingly and with thought to long-term needs. It is also robust to changes in the market share of various technologies. Some software toolsets require a particular software program to run on the client machine, such as Windows, Java, or Flash. These toolsets risk becoming obsolete when the market share of that platform declines. In contrast, oTree only requires a web browser on the client's device. Furthermore, alternative front-ends can be built alongside the existing front-end. For example, an alternative iPhone or Android interface can be built to deliver a richer experience on mobile devices.

7 Conclusion

oTree is platform-independent and can be deployed on any device, including tablets and smartphones, with a modern web browser and internet connection (or alternatively, a local network). This flexibility not only allows for substantial cost savings in laboratory hardware and maintenance, but also enables researchers to recruit large numbers of subjects, or particular subjects, and to conduct artefactual field experiments. Furthermore, it allows any researcher, referee or student to examine any study easily by simply playing the game in the browser. Pages and interactive elements scale automatically to screen size.

oTree is open-source and based on Python, a standard, wide-spread programming language familiar to many economists and students. oTree is now in beta; a reference manual and a tutorial are available at oTree.org.

Acknowledgements For comments on the oTree software we thank the people at the experimental econ laboratories in Magdeburg, Hamburg and ETH Zurich, in particular Stefan Wehrli. Superb research assistance by Stefan Bucher is acknowledged.

References

Dreber, A., J. Horton, and D. Rand (2009): "The 21st Century Mechanical Turk: A True Automaton for Running Experimental Games," http://nber.org/ dlchen/papers.htm.

Fischbacher, U. (1998): *Z-Tree-Zurich Toolbox for Readymade Economic Experiments: Instruktionen für Experimentatoren*, http://www.iew.uzh.ch/ztree/ztreemand.pdf.

Harrison, G. W. and J. A. List (2004): "Field Experiments," *Journal of Economic Literature*, 42, pp. 1009–1055.

Henrich, J., S. J. Heine, and A. Norenzayan (2010): "The weirdest people in the world?" *Behavioral and Brain Sciences*, 33, 61–83.

Horton, J. J., D. G. Rand, and R. J. Zeckhauser (2011): "The online laboratory: conducting experiments in a real labor market," *Experimental Economics*, 14, 399–425.

Janssen, M. A., A. Lee, T. Waring, and D. Galafassi (2014): "Experimental Platforms for Behavioral Experiments on Social-Ecological Systems," CSID Working Paper Series CSID-2014-001, Arizona State University, http://hdl.handle.net/10535/9285.

Mazur, E. (2009): "Farewell, Lecture?" *Science*, 323, 50–51.

Rubinstein, A. (1999): "Experience from a Course in Game Theory: Pre- and Postclass Problem Sets as a Didactic Device," *Games and Economic Behavior*, 28, 155 – 170.