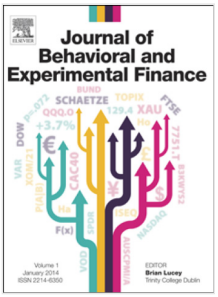
Accepted Manuscript



oTree—An open-source platform for laboratory, online, and field experiments

Daniel L. Chen, Martin Schonger, Chris Wickens

PII: S2214-6350(16)00010-1
DOI: http://dx.doi.org/10.1016/j.jbef.2015.12.001
Reference: JBEF 63

To appear in: *Journal of Behavioral and Experimental Finance*

Received date: 5 August 2015
Revised date: 21 December 2015
Accepted date: 21 December 2015

Please cite this article as: Chen, D.L., Schonger, M., Wickens, C., oTree—An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance* (2016), http://dx.doi.org/10.1016/j.jbef.2015.12.001

OTREE - AN OPEN-SOURCE PLATFORM FOR LABORATORY, ONLINE, AND FIELD EXPERIMENTS

DANIEL L. CHEN, MARTIN SCHONGER, AND CHRIS WICKENS*

oTree is an open-source and online software for implementing interactive experiments in the laboratory, online, the field or combinations thereof. oTree does not require installation of software on subjects' devices; it can run on any device that has a web browser, be that a desktop computer, a tablet or a smartphone. Deployment can be internet-based without a shared local network, or local-network-based even without internet access. For coding, Python is used, a popular, open-source programming language. www.oTree.org provides the source code, a library of standard game templates and demo games which can be played by anyone.

JEL Codes: A20, C88, C90

Keywords: experimental economics, software, laboratory experiments, field experiments, online experiments, classroom experiments

FIGURE 1.— oTree on different devices and operating systems

Experimental economics has become an established field. The common procedure is to conduct incentivized experiments in dedicated university

*Toulouse Institute for Advanced Studies; Center for Law and Economics, D-GESS, ETH Zurich. For comments on the oTree software or the paper, we thank the people at the experimental econ laboratories in Magdeburg, Hamburg and ETH Zurich, Juan Cabral, Donja Darai, Som Datye, Gregor Muellegger, Kelly Reeve, Alexander Sandukovskiy, and Stefan Wehrli. Stefan Bucher provided superb research assistance.

computer labs using students as subjects. In these settings Fischbacher's z-Tree was and is the dominant software platform. z-Tree allows the investigation of strategic interaction using desktop computers running the Windows operating system. According to Google Scholar, Fischbacher's z-Tree (1999, 2007) has been used at least 4801 times, which is a proxy for how important a public good Fischbacher created.

In homage to z-Tree our platform is named oTree[1], where the "o" stands for open-source, online and object-oriented. Why is a new software platform even necessary? There are many reasons, including these main three: First, experimental economics has begun to complement laboratory experiments with field experiments. Field settings range from sportscards shows (List and Lucking-Reiley, 2000) to primary schools (Angerer et al., 2014) to online labor markets (Chen, 2012) to financial trade fairs (Cohn et al., 2015). The field provides external validity and taps a subject pool other than students. It is difficult to take a Windows-desktop networked environment to field settings. This has required either ad hoc programming solutions and thereby hindered experiments. The situation for field experiments is akin to what the situation for laboratory experiments was prior to the release of z-Tree. Second, computing devices have seen a sea change in the last years. Usage has shifted from desktop computers to phones and tablet computers. Graphics and user interfaces have been revolutionized. This calls for software that is platform-independent and deployable on these devices. Third, the programming paradigm has shifted. It is now clear that open-source and industry-standard compatible solutions are a better fit for many scientific needs. Experimental software should use existing industry-standards wherever possible, be extensible and compatible with plug-ins and allow users to inspect the source code. oTree addresses these desiderata.

oTree has been used in the laboratory setting with over a thousand participants in actual experiments at laboratories in Zurich, Magdeburg, and Hamburg. Papers using data gathered by oTree in the laboratory now include Chen and Schonger (2014a), Chen and Schonger (2014b), Chen and Schonger (2015a) and Chen and Schonger (2015b). In addition to

[1]This follows the tradition in the open source community of naming open source software projects after existing software. For example, OpenOffice is named after Microsoft Office, Linux is named after Unix, and MySQL is named after mSQL.

the laboratory, oTree has also been used for experiments on Amazon's Mechanical Turk. Thus, several new features have already been successfully demonstrated: The flexible and easy implementation of graphical and experimental features, such as different background colors and randomization of graphical elements in the presentation, has been shown to work. Second, the mobility of the setup was shown by running the same or similar experiments across various laboratories. Third, the software and hardware requirements of oTree were shown to be minimal. It is sufficient to have an internet connection (or a local area network, both are not necessary). The architecture enables oTree to be used in almost any experimental scenario.

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

1. USAGE: LAB, ONLINE, FIELD, AND CLASSROOM

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. With these links, participants are sent to individualized web pages displaying the experiment and recording their answers. The experimenter accesses the *Progress Monitor,* where real-time information about the progress and entries of all participants is displayed. This architecture enables oTree to be used in almost any experimental scenario. The software and hardware requirements of the laboratory are minimal: only a connection to the server and a browser are needed, no other software needs to be installed.There is a 1-click installer and graphical launcher. The architecture is also robust to failure; if, for example, there is a hardware failure on anything but the server, the link of that participant or experimenter can be loaded immediately on another device.

[2]The MIT license is a standard license used in the open source community; oTree licensing information is at oTree.org/license.

Section 2 lists the most useful oTree features. There are many ways in which oTree is an improvement over existing instruments, many of which might only be appreciated in actual usage. An easy way to get a first feel for oTree is to play some games from the library of dozens of sample games. Not only is the code for these sample games available, but these samples are online using oTree's demo mode. This means that anyone can play them at any time using a web browser without any installation whatsoever. The library of sample games can be found at http://demo.otree.org/demo/. One example of a sample game is our implementation of Basu (1994)'s traveler's dilemma. This game can be played online in demo mode by visiting the demo page of the website and clicking on Traveler's Dilemma. On that start page there are two start links, one for each of the two players in that game. One can play both players using different tabs in a browser, or open the links on different devices.

Section 3 elaborates on programming in oTree. Almost any graphical or design feature can be implemented (i.e., visual styles, animations, dynamic adaptation to screen sizes, and multimedia). oTree's user interface is based on HTML5, which is supported in all modern browsers and is familiar to every web developer and designer. The programming language used in oTree is Python, an easy-to-learn modern object-oriented computer language widely used in science for analysis and modeling. Hence, many researchers already possess the necessary programming skills to use oTree, and will require little time to learn to use oTree efficiently and to implement experiments. oTree is easy to use, and even complex experiments are reasonably easy to program by taking advantage of the capabilities of Python.

oTree is based on the Django web application framework; oTree applications are web applications. Once oTree is installed on a web server, applications can be played instantly from any web browser by opening the game's URL, and experimenters can log in to the admin dashboard 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, including Windows, Mac OS X, iOS, Android, and Linux. The user interface automatically adjusts to the appropriate screen resolution, eliminating the need for every application

to have separate designs for desktop and mobile platforms.

oTree is built to last. The source code is highly organized and follows best practices, allowing 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 tool sets require a particular software program to run on the client machine, such as Windows, Java, or Flash. These tool sets risk becoming obsolete when the market share of that platform declines. In contrast, oTree only requires a web browser on the clients' devices.

1.1. *Dedicated or ad hoc laboratories*

Researchers using oTree do not require dedicated experimental laboratories; they can use existing computer laboratories or rely on tablets. Since touch input works with oTree, tablets can be used without external keyboards, if desired. oTree can be run on low-cost devices such as commodity tablets and mobile devices, thereby reducing hardware costs. Several dozen tablets can be easily transported to any space able to accommodate the participants (such as a classroom or meeting room) and be set up as a temporary laboratory. 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, anti-virus, 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. oTree and its graphical launcher can be installed in one click.

1.2. *Online*

oTree experiments can be conducted online. The experimenter provides each participant a unique start URL. These URLs contain a random alphanumeric code so even if participants communicated with one another, the link would not allow participants to deduce the identity of players in a particular match. There is no set limit on the number of simultaneous participants, thus large-scale market experiments are feasible. As is the

case for websites in general, running an experiment with many users requires a server with sufficient resources (processor, RAM, and database). To test whether hardware resources are sufficient and fast enough, we recommend testing the setup with bots (see subsection 2). Since the code of experiments (though presumably not all the instructions and payouts) needs no change between online and laboratory settings, running an experiment online can complement running it in the lab. It can serve as an initial setting for a low-cost pilot and enable authors to secure funding for a more expensive lab experiment. It could also be a setting for a pre-publication replication, a practice suggested by Gelman and Loken (2014). Besides low costs, another advantage of online experiments is that they are more scalable in terms of participant numbers. It is even possible to run short experiments with many participants. This is likely an advantage over using a lab where recruitment of large numbers of participants can be challenging due to the high fixed opportunity cost of reaching the laboratory site. Amazon Mechanical Turk (AMT) is a popular platform[3] for conducting online experiments, and thus oTree integrates with the AMT payment system.

1.3. *Field*

Field experiments are of crucial importance to answer certain questions and can serve as small-scale tests for policy makers (Carpenter, 2005; Gneezy and Rustichini, 2000). Both laboratory and field experiments have their advantages and shortcomings (Levitt and List, 2007), but field experiments can serve as bridges between laboratory data and naturally-occurring data (Levitt and List, 2009).

The field provides external validity and taps a subject pool other than students. It is difficult to take a Windows-desktop networked environment to field settings. This has required either ad hoc programming solutions or has limited the range of field experiments. If a researcher uses existing computers in the field, no administration rights are necessary on them. In laboratory and many field settings, one will want to prevent participants from using the provided devices for purposes other than the experiment. Thus, it is useful to lock down the devices to a web browser kiosk mode,

[3] See Horton et al. (2011) and Mason and Suri (2012).

meaning that a user cannot open other web pages or leave the browser without the kiosk password. Internet Explorer, Chrome, and Firefox all have built-in kiosk modes. Even in kiosk mode, it typically could be a problem if a user presses the browser's reload or back button or the keyboard shortcuts. In oTree, these user actions simply result in a redisplay of the current page in the experiment.

oTree can be used in field settings and subjects can use their own smartphones or tablets. Another approach is to have a rolling suitcase full of tablets. In this way, one can set up temporary laboratories to conduct artefactual field experiments. Henrich et al. (2010) cautioned that the participant pool in university laboratories is usually "WEIRD"—Western, Educated, Industrialized, Rich, and Democratic. With oTree it becomes easy to set up temporary laboratories in field settings where non-student subjects are easier to recruit.

For some research questions, it is difficult to gather enough data in a laboratory setting. Behavioral experiments that go beyond existence of a treatment effect can require hundreds to thousands of subjects. Obtaining large numbers of participants is especially difficult and costly if the experiment is short. With oTree, a team of research assistants can take a suitcase with 40 tablets to a high school, a corporation, a trade fair, a train station, or an airport and recruit convenience samples of hundreds of subjects in a single day.

1.4. *Hybrid settings*

Experiments can be run in multiple labs in different parts of the world simultaneously and participants need not be at the same location to interact with each other. In some experiments—such as those investigating the effect of beauty on peer effects and team work—it is necessary to show a player the face of the counterpart. To eliminate unobserved post-game interaction, the counterpart is in a lab in a different city. For oTree, this is no more complicated than it would be if the two players were in the same room.

1.5. *Classroom*

Many social scientists let students play games to teach them game theory and mechanism design.[4] oTree can make lectures more vivid by having students play games during class and homework more playful but also more easily monitored. Dynamic graphics (see subsection 2) can be projected in the classroom enabling discussion of the game played. The connection to the server occurs seamlessly via the internet, so students can use the classroom wifi, mobile cell coverage, or their home ISP. Mazur (2009) suggests to "flip the classroom", which requires hardware- or software-based clickers. oTree can be a software-based, free clicker, and it is easy to add functionality that goes far beyond that of traditional clickers.

To use oTree in the classroom, it is easiest to use the demo mode discussed in subsection 2. Regardless of whether the demo mode or the regular session mode is used, each student is given a unique URL at the beginning of the lecture or the course. The teacher can use the session interface to project results in the classroom, and open the progress monitor on a different machine to privately check how students are doing.

2. FEATURES

oTree experiments are web pages using the HTML5 standard. To make it easy to program visually appealing experiments, oTree uses the open source web front-end framework, Bootstrap. For details, see section 3 on programming. Since experiments are web pages, all that is needed on participants' and the experimenters' devices is a web browser. On each device, one opens a URL that is unique to the respective participant and session.

Paper-like treatment-specific instructions: Many experimenters hand out instructions on paper, even in computer-based lab or artefactual field experiments. oTree provides instructions that have a paper-like look (Figure 2). Typically, they will be displayed by themselves on an introductory screen, and then redisplayed at the bottom of all subsequent screens, a fact which should be pointed out in the last sentence of the instructions. To minimize experimenter demand it is desirable to hide from subjects

[4] See for example Rubinstein's (1999) recommendations.

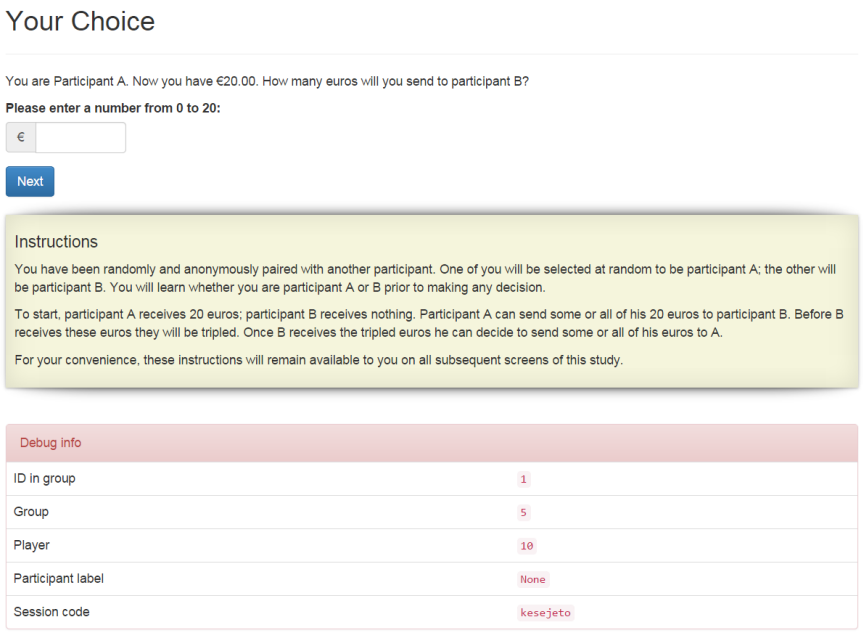


FIGURE 2.— Participant screen

which treatment groups and dimensions exist, which is made easier by on-screen instructions.[5]

AMT integration: oTree provides integration with Amazon Mechanical Turk (AMT). oTree authenticates users visiting from the AMT service, and then sends payments to the correct AMT account. Researchers can now publish experiments and send payments directly to AMT. Researchers, however, must have an employer account with AMT, which currently requires a U.S. address and bank account.

Debug info: Any application can be run so that that debug information is displayed on the bottom of all screens (see Figure 2). The debug information consists of the ID in group, the group, the player, the participant label, and the session code. The session code and participant label are two randomly generated alphanumeric codes uniquely identifying the

[5] On experimenter demand and non-deceptive obfuscation see Zizzo (2010) and Bardsley et al. (2010), fn.39.

session and participant. The ID in group identifies the role of the player (e.g., in a principal-agent game, principals might have the ID in group 1, while agents have 2).

Testing with bots: A bot is a an artificial player executing a pre-determined strategy. The strategy can be pure or mixed. For example, to stress test experiments, devices, or servers, bots can be programmed to make random but valid entries. Tests with hundreds of bots are completed within seconds and are thus able to test if everything was programmed correctly. Sometimes programmers might also discover errors in the logic or the economics of their game (e.g., negative payoffs that might violate lab rules or participation constraints). Axelrod and Hamilton (1981) invited submissions of strategies to play a computer tournament of a repeated Prisoners' Dilemma. In oTree, such an exercise becomes easy: each submitted strategy corresponds to a particularly programmed bot. Then bots are instantiated in the next round according to their payoffs in the previous round. A bot can also be used for communication and teaching purposes. Rather than having a student or reader play all the players in an n-player game, n-1 bots might be used, and the n-th player would be the human.

Progress Monitor: The *progress monitor* (Figure 3) allows the researcher to monitor the progress of an experiment. It features a display that can be filtered and sorted, for example by computer name or group. The experimenter can see the progress of all participants, the current action a participant is taking, and decisions made by participants. Updates are shown as they happen in real time and cells that change are highlighted in yellow. Because the progress monitor is web-based, multiple collaborators can simultaneously open it on several devices on premises or at remote locations. The admin dashboard is automatically generated for each experiment. Columns included are: the participant ID (the computer number in a laboratory setting), the participant code (a short alphabetic code uniquely identifying the participant), and the current earnings of the participant. Figure 3 displays a principal-agent game.

Session interface: While the progress monitor exists in all oTree experiments, the *session interface* is an optional feature that is convenient to have in some experiments. In many experimental settings, in addition to an experimenter monitoring progress for which the progress monitor is

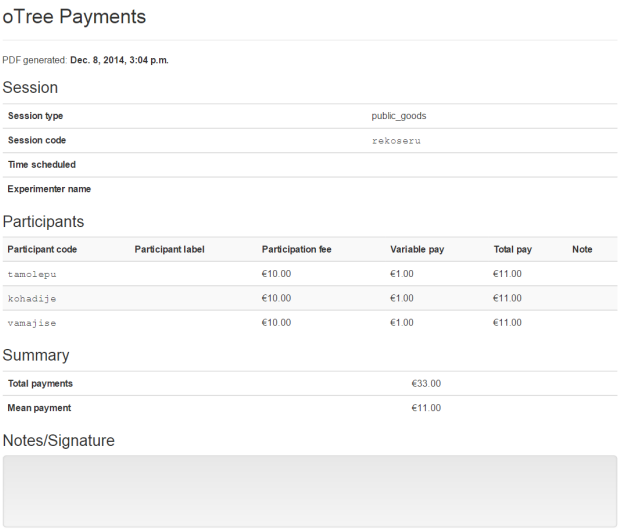
Total return	Agent fixed pay	Agent return share	Agent work effort	Agent work cost	Contract accepted
(None)	(None)	(None)	(None)	(None)	(None)
(None)	$2.00	0.6	(None)	(None)	Yes
(None)	$5.00	0.2	(None)	(None)	(None)
(None)	(None)	(None)	(None)	(None)	(None)
(None)	$5.50	0.1	(None)	(None)	(None)
(None)	$6.50	0.7	(None)	(None)	Yes
(None)	$5.00	0.6	(None)	(None)	(None)

FIGURE 3.— Progress Monitor

intended, an experimenter needs to receive instructions or provide input for the experiment. The session interface can instruct an experimenter on what to do next and show text to be read aloud. An earlier version of oTree supported additional forms of experimenter input. The session interface can also request input from the experimenter at a specific point in the session. For example, in an Ellsberg experiment, the experimenter might fill an opaque urn prior to the session; the session interface will remind the experimenter to show the urn to the participants, and tell the experimenter when all participants have selected their bets and instruct the experimenter to draw a ball from the urn. It will then ask the color that was drawn, so that oTree can calculate participants' payoffs.

Payments PDF: In a lab session, dozens of participants must be paid in a short amount of time after the end of a session. To minimize waiting times for subjects, oTree automatically generates a PDF with the earnings of each participant as soon as the earnings-relevant part of a session is over (usually before an exit questionnaire).

Demo Mode: Experiments can be put online using the demo mode which creates a web page with a static URL. Any visitor to that web page can play the game. For example, for an Ultimatum Game, that web page would have two participant links, one for proposer and one for responder. Whenever someone new visits the static demo URL of the game, a new session is instantiated meaning that new participant URLs are created. If the game has a study-interface a new URL for that is also created. The demo mode makes it possible to easily share a game with colleagues, referees, and other readers. For example, we use it for our collection of sample games. Not only is the code of these games available, but one



oTree Payments

PDF generated: **Dec. 8, 2014, 3:04 p.m.**

Session

Session type	public_goods
Session code	rekoseru
Time scheduled	
Experimenter name	

Participants

Participant code	Participant label	Participation fee	Variable pay	Total pay	Note
tamolepu		€10.00	€1.00	€11.00	
kohadije		€10.00	€1.00	€11.00	
vamajise		€10.00	€1.00	€11.00	

Summary

Total payments	€33.00
Mean payment	€11.00

Notes/Signature

FIGURE 4.— Payments PDF

can also play them at any time online: demo.otree.org. Using bots, even multi-player games can be put online as supplementary, interactive material.

User input and validation: oTree supports all the standard input forms, such as radio buttons, check boxes, drop-down menus, numerical entry, free text entry, date/currency input, email, file upload, and image fields. To simplify usage of input forms, oTree relies on Django, a widely used Python web application framework. Some custom form widgets, such as Likert scales, go beyond what Django offers by default, but are included in oTree. oTree users are not limited to what Django or oTree include, and can create their own input forms. Figure 5 gives examples of input forms available.

If a participant does not fill out a required form or submits an invalid value, the form is automatically re-displayed, highlighting and explaining the user's error. The experimenter can specify what answers are valid. In the case of a numeric input field, the experimenter could, for example, specify that only numbers that are a negative odd integers between -51 and -1 should be valid. Among valid answers, there may be some that are

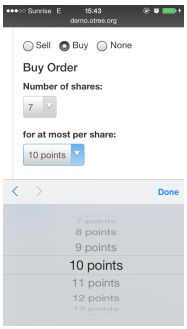


FIGURE 5.— oTree on smartphone

better or more rational than others, but the user can proceed with a chosen answer and is not held back on that page. Validation is only intended for cases where the experimenter wants to prevent the participant from proceeding without making an answer satisfying some specified criteria.

Auto-advancing participants: Sometimes a single participant holds up an entire experiment. The experimenter can force that participant to the next page by pressing the auto advance button. This functionality is also useful for debugging or demonstration purposes.

Customizable randomization and matching: Participants in a session are usually partitioned into groups. First, there are treatment groups that allow 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. Within a match participants may have different roles such as principal and agent. oTree handles this by a matrix model, where each cell corresponds to a participant, and the rows are groups, while the columns are roles. Players can be grouped randomly or by arrival order.

Dynamic graphics: In many experiments it is useful to have visual content created and updated during the experiment. In a market game, for example, participants could be shown a graphic depicting the evolution of prices. For this purpose, oTree integrates HighCharts, a JavaScript-based service. Figure 6 is a screenshot of players' points after round

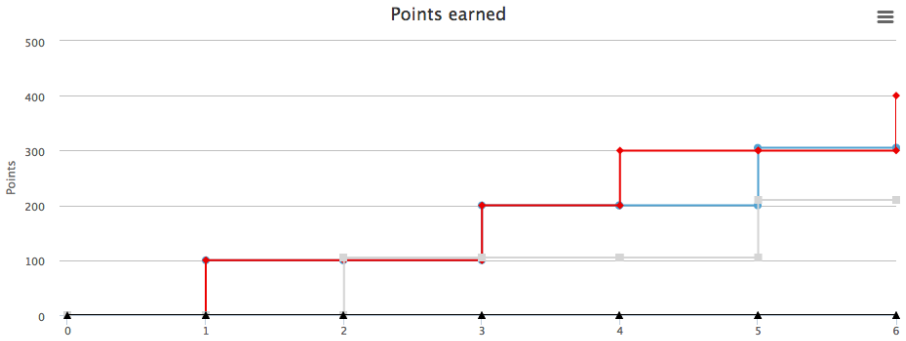


FIGURE 6.— Dynamic graphics

six of a 50-round game. Other graphical capabilities are illustrated at www.highcharts.com.

Sample games: oTree includes a library of dozens of standard economics games. To see these games and play them, go to http://demo.otree.org/demo/. Incidentally, these games are put online using oTree's demo mode.

Researchers can use the code of these sample games as templates for their own games. Those who are developing completely new games are also encouraged to review the sample games to learn about recommended coding styles. Authors are invited to send their codes, which will be added to our library.

Localization: Of course, oTree applications can be translated to multiple languages and oTree has multiple features to ease translation and localization. A single switch controls the language in which the application is currently displayed. This is useful in many scenarios. For example, an experimenter can deploy the same experiment in the U.S., 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 screen shots in a paper reporting the experiment. oTree is able to display monetary amounts in various currency formats (compare Figure 2) In addition to various currencies, the games may also be played for points.

Data export: All data in an experiment is saved to the server. Data from subsequent sessions is appended. At any point in time the data can

be downloaded in standard CSV format using a web interface. In addition to participants' choices and answers, the data recorded includes the date and time a participant first accesses the experiment, time spent on each page, the device's IP address, and the current page of each participant or whether he or she is finished.

Data auto-documentation: oTree auto-generates human-readable documentation from the application's code. It gives the name and data type (e.g., positive integer, string) of all variables. If the variable can take a list of specific value, the documentation will print that list showing both the internal name and the displayed name that the participant sees. As most programmers comment in their code, oTree intelligently extracts these comments and adds them to the documentation.

Open source: Sonnenburg et al. (2007) have argued that in scientific applications, the source code should be available for inspection and available free of charge to enable replication and extension. Janssen et al. (2014) point out that that binary software cannot be checked by people other than the software authors. Lerner and Tirole (2005) argue that open-source software will grow strongly, and that academic economists could both learn and benefit from the model. Thus oTree is open source, meaning that the source code is freely available and anyone can add to it subject to our open-source license terms (http://www.otree.org/license). To develop oTree, we use GitHub, a web service familiar to most programmers which hosts many open-source projects and offers source code management and distributed revision control. Researchers can also use the oTree space on GitHub to deposit their applications in addition to doing so at journal archives. This enables future researchers to find numerous programs and build on them. Each time an experiment is run, the server records what version of the code was used to run the experiment. Every version of oTree is archived, and a programmer can revert to an old version using the Git version control system.

3. TECHNOLOGY AND PROGRAMMING

This section gives an overview of the oTree architecture and programming. For details, and to get started, consult the website, oTree.org, download the source code that comes with dozens of sample games, and consult the wiki at www.docs.oTree.org. oTree's programming model fol-

lows the Model-View-Controller approach as discussed in Leff and Rayfield (2001) for the specific case of web applications. The controller resides in the core of oTree. Each oTree application consists of a models.py file, a views.py file and several html-templates—one for each screen. The models.py file defines the players, variables, and their data types; defines the validation logic; and specifies the arithmetic operations to be performed. The view is given by the views.py file and the html-templates. The views.py file defines the sequence of pages a participant sees, references the templates to be used, and which variables are inserted into the templates.

3.1. *Skills required to program in oTree*

oTree is based on Python, a widely used programming language that is available as open-source software from the Python Software Foundation. Python is often the first language taught in introductory computer sciences courses.[6] Learning Python might be a good investment for empirical and experimental researchers as it is useful not only for oTree, but for many tasks such as automatic data extraction and web-scraping. oTree is designed to be simple and to be accessible to people with basic programming experience.

Researchers who prefer to hire a programmer for oTree, rather than code themselves, should hire someone who knows Python. For best results, the programmer should also be familiar with Django, which is the most popular Python-based web development framework. A programmer familiar with Python and Django can be productive in oTree almost immediately and program his/her first experiment within hours.

3.2. *Hardware requirements*

The only requirement on participants' computers is a web browser capable of HTML5. Anything that works in HTML5 can be included in oTree, for example hyperlinks, dialog boxes, tables, images, audio, and video. Video recording is still experimental in HTML5. JavaScript, Java, and Flash are supported by oTree, but programmers should be aware of

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

the limitations operating systems impose (e.g., Flash does not run on iOS devices). A good recommendation for oTree experiments—and for any website—is to use JavaScript if necessary, but not Java or Flash. Computer requirements for the server depend on the number of participants and the experiment; often a standard laptop will suffice. If reliable internet is available, the easiest solution is often to rent a server in the cloud. Such servers can be rented flexibly for a short time. This makes particularly good sense if the experiment is run online or has high demands in terms of graphics. Prior to an experimental session, it is advisable to test the server using bots; use more bots than the maximum number of subjects planned for the session to be on the safe side.

3.3. *Front-end*

Since working directly with HTML5 for the user interface can be cumbersome, oTree uses a free, standard front-end framework for websites called Bootstrap.[7] Bootstrap originates from Twitter, a private corporation, but has since been released as open source and become ubiquitous. Thus, oTree users benefit from the continuous development of Bootstrap independently of any updates to oTree itself. All standard HTML5 elements can be beautifully and easily implemented in oTree thanks to Bootstrap. But, perhaps most importantly, Bootstrap intelligently scales the layout of a website including buttons, tables, alerts, error messages, and menus to different screen and device sizes. By default, oTree has a neutral layout but experimenters can change the style by using cascading style sheets (CSS).

3.4. *Simple code*

oTree's code is remarkably simple as it builds on Python. Compared to z-Tree, oTree is less specifically tailored to some experiments but arguably more general. In z-Tree, users spend considerable time and effort implementing solutions to problems that have now become easy using modern programming languages. To give an idea of the simplicity of oTree, consider the standard experimental requirement of playing a ran-

[7] www.getbootstrap.com.

dom subset of rounds in a game with many rounds, the code for this is shown in figure 7.[8]

```
if self.subsession.round_number == Constants.number_of_rounds:
    random_players = random.sample(self.me_in_all_rounds(), 3)
    self.payoff = sum([p.theoretical_payoff for p in random_players])
```

FIGURE 7.— oTree code

3.5. *Programming assistance*

oTree works with standard code editors such as PyCharm.[9] A code editor makes programming easier by providing 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 the auto-complete functionality of PyCharm. Suggestions of variable names are presented as code is typed. If oTree encounters an error during execution, an error page pinpointing the source of the error is shown.

3.6. *Collaborative development*

oTree is developed using 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. We invite other researchers to join us on GitHub, but this is not a requirement for using oTree.

4. CONCLUSION

oTree is an open-source, online, and object-oriented software platform for implementing social science experiments in the laboratory, online, in

[8] To solve this problem in z-Tree one needs a list of traversal algorithms resulting in about 25 lines of code, compare Bausch (2012).

[9] PyCharm has a free license for classroom use.

the field, or combinations thereof. oTree is operating-system independent and deployable on any device that has a web browser, including tablets and smartphones. The server can be an ordinary laptop. Once oTree is installed on that one machine, no further installation is necessary on participants' computers and experiments can be started instantly from any web browser by opening the game's URL. Experimenters can log in to the web-based progress monitor to monitor the experiment. This flexibility allows for substantial cost savings in laboratory hardware and maintenance, and also makes it easier for researchers to recruit large numbers or particular kinds of subjects in ad hoc laboratory or field settings. Using a single software for both online and lab experiments makes life easier for researchers. Researchers with limited funds can first run their experiment online (where participation fees are measured in cents or single digits) and then—armed with the data and results—either raise more funds or decide to abandon the project as unpromising. Running the same experiment in both online and lab settings also tests for the robustness of results and may support the external validity of results. oTree enables a quick and easy replication of research results at low cost. We hope that oTree will indirectly contribute to fostering transparency and replication in the experimental social sciences. The demo mode allows any researcher, referee, or student to understand any experiment easily by simply playing the game in a browser. Furthermore, the demo mode can be used to quickly run and demonstrate games in the classroom. oTree is based on a sequence of pages, each containing an optional form for the participant to fill out and submit. This interactive 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, ConG (Pettit et al., 2014) would be more suitable here. Scientific advances are often driven by the availability of new and cheaper scientific instruments—oTree aims to be such an instrument.

REFERENCES

ANGERER, S., D. GLÄTZLE-RÜTZLER, P. LERGETPORER, AND M. SUTTER (2014): "Third-party punishment increases cooperation in children through (misaligned) expectations and conditional cooperation," *Proceedings of the National Academy of Sciences*, 111, 6916–6921.

AXELROD, R. AND W. HAMILTON (1981): "The evolution of cooperation," *Science*, 211, 1390–1396.

BARDSLEY, N., R. CUBITT, G. LOOMES, P. MOFFATT, C. STARMER, AND R. SUGDEN (2010): *Experimental economics: Rethinking the rules*, Princeton University Press.

BASU, K. (1994): "The Traveler's Dilemma: Paradoxes of Rationality in Game Theory," *The American Economic Review*, 84, 391–395.

BAUSCH, A. W. (2012): "Introduction to z-Tree: Day 3," https://files.nyu.edu/awb257/public/slides/Bausch-ztreeslides3.pdf.

CARPENTER, J. P. (2005): "Endogenous Social Preferences," *Review of Radical Political Economics*, 37, 63–84.

CHEN, D. AND M. SCHONGER (2014a): "Social Preferences or Sacred Values? Theory and Evidence of Deontological Motivations," Working paper, ETH Zurich.

——— (2015a): "Testing Axiomatizations of Ambiguity Aversion," Working paper, ETH Zurich.

CHEN, D. L. (2012): "Markets and Morality: How Does Competition Affect Moral Judgment," Working paper, Duke University School of Law.

CHEN, D. L. AND M. SCHONGER (2014b): "A Theory of Experiments: Invariance of Equilibrium to the Strategy Method and Implications for Social Preferences," Working paper, ETH Zurich, Mimeo.

——— (2015b): "Ambiguity Aversion without Asymmetric Information," Working paper, ETH Zurich.

COHN, A., J. ENGELMANN, E. FEHR, AND M. MARÉCHAL (2015): "Evidence for Countercyclical Risk Aversion: An Experiment with Financial Professionals," *American Economic Review*, 105, 860–885.

FISCHBACHER, U. (1999): *Z-Tree-Zurich Toolbox for Readymade Economic Experiments: Experimenter's Manual*, Citeseer.

——— (2007): "z-Tree: Zurich toolbox for ready-made economic experiments," *Experimental Economics*, 10, 171–178.

GELMAN, A. AND E. LOKEN (2014): "The Statistical Crisis in Science," *American Scientist*, 102, 460.

GNEEZY, U. AND A. RUSTICHINI (2000): "A Fine Is a Price," *The Journal of Legal Studies*, 29, 1–17.

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," *Ecology and Society*, 19.

LEFF, A. AND J. T. RAYFIELD (2001): "Web-Application Development Using the Model/View/Controller Design Pattern," in *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, Washington, DC,

USA: IEEE Computer Society, EDOC '01, 118–127.

LERNER, J. AND J. TIROLE (2005): "The Economics of Technology Sharing: Open Source and Beyond," *Journal of Economic Perspectives*, 19, 99–120.

LEVITT, S. AND J. LIST (2007): "What Do Laboratory Experiments Tell Us About the Real World?" *The Journal of Economic Perspectives*, 21, 153–174.

LEVITT, S. D. AND J. A. LIST (2009): "Field Experiments in Economics: The Past, the Present, and the Future," *European Economic Review*, 53, 1–18.

LIST, J. A. AND D. LUCKING-REILEY (2000): "Demand Reduction in Multi-Unit Auctions: Evidence from a Sportscard Field Experiment," *The American Economic Review*, 90, 961–972.

MASON, W. AND S. SURI (2012): "Conducting Behavioral Research on Amazon's Mechanical Turk," *Behavior Research Methods*, 44, 1–23.

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

PETTIT, J., D. FRIEDMAN, C. KEPHART, AND R. OPREA (2014): "Software for continuous game experiments," *Experimental Economics*, 17, 631–648.

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

SONNENBURG, S., M. L. BRAUN, C. S. ONG, S. BENGIO, L. BOTTOU, G. HOLMES, Y. LECUNN, K.-R. MULLER, F. PEREIRA, C. E. RASMUSSEN, G. RATSCH, B. SCHOLKOPF, A. SMOLA, P. VINCENT, J. WESTON, AND R. C. WILLIAMSON (2007): "The need for open source software in machine learning," *Journal of Machine Learning Research*, 8, 2443–2466.

ZIZZO, D. (2010): "Experimenter demand effects in economic experiments," *Experimental Economics*, 13, 75–98.