Preliminary working paper.

oTree

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

Daniel L. Chen · Martin Schonger · Chris Wickens

November 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 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. Deployment can be internet-based without a shared local network, or alternatively local network based even without internet access. With Python, oTree uses an industry standard, open-source programming language. Creating experiments can be learned quickly, especially by those already familiar with Python. www.oTree.org provides a library of standard games, which can be used for teaching or as templates for experiments. For stress testing game logic, programming and the hardware setup, oTree bots can simulate hundreds of players and game plays. A demo mode allows researchers to put their games online as supplementary, interactive material. We use it at the www.oTree.org website, where you can play sample games from your browser. Also at that website you will find the oTree source code and further documentation in form of a wiki.

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

JEL Codes: A20, C70, C88 ,C90

Preliminary working paper. We appreciate your comments.

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 library games in demo mode visit www.oTree.org.
oTree38.LYX

1 Introduction

Fig. 1: oTree on different devices and operating systems

Experimental economics has become an established field in economics. The traditional paradigm of experimental economics is to conduct incentivized experiments in dedicated university computer labs employing student subjects. In these settings Fischbacher (Fischbacher 1998, 2007) 's z-Tree is the leading software platform. z-Tree allows the investigation of strategic interaction using desktop computers running the Windows operating system. In homage to the public good Fischbacher created in z-Tree, we named our platform oTree[1]. The o stands for open-source, online and object-oriented. Open-source means that the oTree source code is available for inspection (which follows the scientific spirit of replicability and accessibility of all tools and methods), online means that it runs on any device that can run a web browser, and object-oriented means that it uses Python, an object-oriented programming language.

Scientific advances are often driven by the availability of new and cheaper scientific instruments, think of the microscope and its different incarnations. oTree aims to be such an instrument. Three main advances of oTree are: (i) oTree works across experimental settings: be it on desktop computers in the lab, on tablets in the field, on students' smartphones in the classroom or on combinations of devices at various locations. (ii) User interfaces have seen a sea change in the last decade, and oTree has the feel of contemporary web applications. (iii) oTree is open source meaning that the source code can be inspected.

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

[1] This follows the tradition in the open source community to name open source software projects after existing software. For example OpenOffice is named after Microsoft Office, Linux is named after Unix, MySQL is named after mSQL, and C++ and C# are named after C.

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. It is sufficient to either have an internet or a local area network, both are not necessary. The architecture is also robust to failure; for example if there is a hardware failure on anything but the server, the link of that participant or experimenter can be loaded immediately on another device.

Section 3 lists the most useful oTree features. There are many ways in which oTree improves 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 of our sample games. We have a library of dozens of sample games. Not only is the code for these sample games available, but we put the sample games 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 is at http://demo.otree.org/demo/. As an example of a sample game take our implementation of Basu (1994)'s traveler's dilemma. Via the sample game website you reach the static URL http://demo.otree.org/demo/traveler_dilemma/ which hosts the program in demo mode. On that start page you get two start links, one for each of the two players in that game. You can either play both players yourself using different tabs in your browser or open the links on different devices if you want to play against someone.

Section 4 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. It is familiar to every web developer and designer. The programming language oTree is Python, an easy-to-learn modern object-oriented computer language widely used in science for analysis and modeling. Hence, there are many researchers who already dispose of the necessary programming skills in order to use oTree, requiring little time to learn to handle oTree efficiently and to implement experiments. As a proof of concept of oTree's flexibility and applicability to a broad range of scenarios, a library of several dozen standard economics games is provided at www.oTree.org. oTree is easy to use, but at the same time, complex experiments are reasonably easy to program by taking advantage of the capabilities of Python.

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 possibilities for experimental eco-

nomics, the main purpose of oTree is to cover the entire spectrum of modern computing. 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.

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 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.

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 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 such as Windows, Mac OS X, 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.

oTree has been used in the laboratory setting with over a thousand participants in actual experiments at laboratories in Zurich, Magdeburg and Hamburg. Two papers using data gathered in the lab with oTree are **?** and Chen and Schonger (2014). In addition to the lab, oTree has been used 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.

2 Usage: Lab, online, field, and classroom

? 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). Field settings range from sports cards shows (**?**), to primary schools (Angerer et al., 2013), to online labor markets (**?**), to financial trade fairs (Cohn et al., 2014). 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.

[2] See oTree.org/license

This has required either ad hoc programming solutions or limited the range of field experiments.

2.1 Dedicated or ad hoc laboratories

oTree can be used in a research laboratory and run on the existing computer workstations. No installation of software is necessary on subjects' computers, a modern, i.e. HTML5-compliant, web browser is sufficient. To prevent subjects from browsing the web, it is advisable to set the browser into a kiosk mode. Kiosk modes are available in many browsers, the oTree wiki gives details for several browsers.

Researchers using oTree do not require a dedicated experimental lab, they can use existing computer laboratories or rely on tablets. Touch input works with oTree, so tablets can be used even without external keyboards if desired. Dozen of tablets can easily 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, 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.

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

2.2 Online

oTree experiments can be conducted online. The experimenter provides each participant a unique, individual start URL. These URLs contain a random alphanumeric code so even if participants communicated between 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 to test the setup with bots (see subsection 3.4). Since the programming of experiments (though presumably not all the instructions and payouts) can be unchanged between online and lab settings, running a particular experiment can be a complement to 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 cost another advantage of online experiments is that they are more scalable in terms of participant numbers. It is possible to run even short experiments with many participants, which in a lab would often only attract few people due to high fixed opportunity cost of getting to the lab. Amazon Mechanical Turk (AMT) is a popular platform[3] to conduct online experiments, and thus oTree integrates with the AMT payment system.

2.3 Field

oTree can be used in field settings. Subjects can use their own smartphones or tablets. Another approach is to have a rolling suitcase full of tablets, as many consultants and salespeople do. With this one can set up temporary lab to conduct artefactual field experiments. Similarly recruitment of 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 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.

Field experiments are of crucial importance for certain questions and can serve as small scale test for policy makers (**?**; **?**). Both laboratory and field experiments have their advantages and shortcomings (**?**), but field experiments can serve to bridge between laboratory data and naturally-occurring data (Levitt and List, 2008).

The participant pool in university-labs is usually "WEIRD" - Western, Educated, Industrialized, Rich, and Democratic. as pointed out by Henrich et al. (2010). The authors state that the study of human psychology and behavior draws heavily on WEIRD subjects, often times even only on undergraduate students. They found that, if other populations are considered, experimental results of behavioral sciences vary, often times drastically, across such research areas as: visual perception, fairness, cooperation, moral reasoning, reasoning styles, self-concepts and related motivations. Often there is little or no evidence as to whether results gained with WEIRD subjects are generalizable. The authors remark that by relying on the WEIRD population, research topics have been limited and the development of evolutionary theories of human behavior have been hindered. Thus, it is warranted to investigate whether, for a given research question, differences between populations are relevant. Certainly, it is strongly advisable to diversify the research population.

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

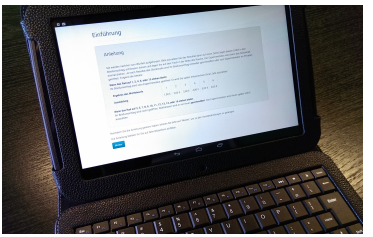


Fig. 2: oTree on a tablet with bluetooth keyboard

2.4 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, but to eliminate unobserved post-game interaction the counterpart is in a lab in a different city. For oTree this is no more complicated than having the two players in the same room.

2.5 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 can become more playful while also being more easily monitored. Dynamic graphics (see subsection 3.10) can be displayed on the classroom overhead to discuss the game played. Students can play the game can play the game on whatever device they have at hand, be it their laptop, tablet or smartphone. The connection to the server occurs seamless 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 such a software-based and free clicker, and it is easy to add functionality that goes way beyond the ones of traditional clickers.

To use oTree in the classroom it is easiest to use the demo mode discussed in subsection 3.5. Regardless of whether the demo mode or the regular session mode is used each student gets a unique URL at the beginning of the lecture or the course. The teacher can use the session interface to project results on the overhead, and open the progress monitor on a different machine to privately check how students are doing.

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

3 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 4 on programming. Since experiments are web pages all that is needed on participants' and the experimenter's devices is a web browser. On each device one opens a URL that is unique to the respective participant and session. In particular no admin rights for the devices are necessary. In laboratory and many field settings one will want to prevent participants from using the provided devices for purposes other than experiment. Thus it is useful if one can lock down the devices to a web browser kiosk mode which means 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 mode. A natural question to ask is what happens if a user presses the browser's reload or back button. Even in kiosk mode that could be a problem as users might know the keyboard shortcuts. oTree simply leaves these user actions without consequence as pressing the back or refresh buttons will just redisplay the current page in the experiment.

3.1 Progress monitor

The *progress monitor* 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.

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)

Fig. 3: Progress Monitor

As 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. As columns it includes the participant ID (which in the lab would be the computer number), 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.

3.2 Session Interface

While the progress monitor exists in all oTree experiments, the *session interface* is an optional feature convenient to have in some experiments. In many experimental settings, in addition to an experimenter monitoring progress for which the progress monitor is 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 out aloud. 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 may 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 her to draw a ball from the urn. It will then also ask which color was drawn, so that oTree can calculate participants' payoffs.

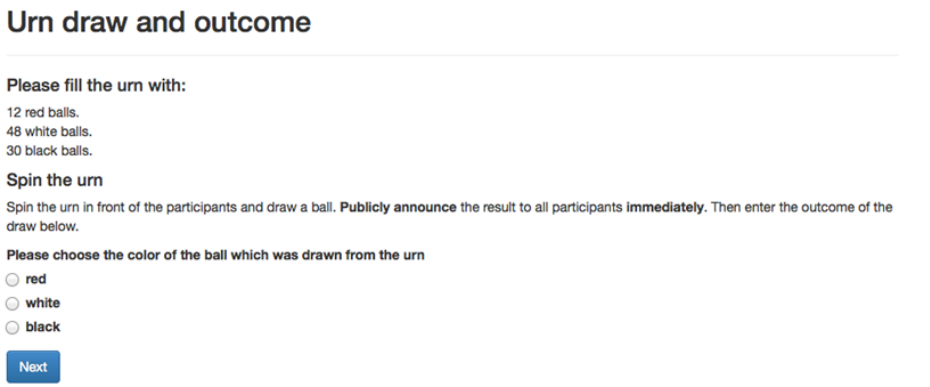
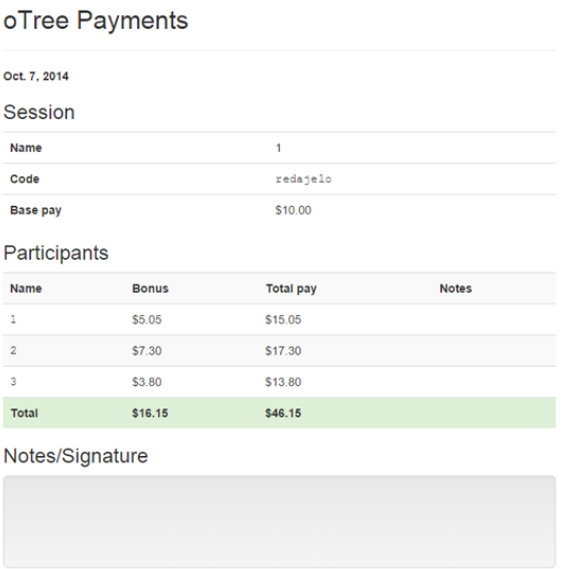


Fig. 4: Session interface

3.3 Payments PDF

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



oTree Payments

Oct. 7, 2014

Session

Name	1
Code	redajelo
Base pay	$10.00

Participants

Name	Bonus	Total pay	Notes
1	$5.05	$15.05	
2	$7.30	$17.30	
3	$3.80	$13.80	
Total	$16.15	$46.15	

Notes/Signature

Fig. 5: Payments PDF

3.4 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 an experiment, devices or the server, bots can be programmed to make random but valid entries. Tests with hundreds of bots complete with in seconds and are thus a tool to test if everything was programmed correctly. Sometimes programmers might also discover errors in the logic or the economics of their game (for example 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.

3.5 Demo Mode

Experiments can be put online using the demo mode. This creates a web page with a static URL. Any visitor to that web page can play the game. For example for a 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. We use it for example for our collection of sample games. Not only is the code of these games available, but we you can also play them at any time from your browser: demo.otree.org. Using bots, even multi-player games can be put online as supplementary, interactive material.

3.6 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, however, need to have an employer account with AMT, which currently requires a U.S. address and bank account.

3.7 Paper-like treatment-specific instructions

Many experimenters hand out instructions on paper, even in computer-based lab or artefactual field experiments. Instructions on paper have the advantage that subjects can consult them at any time during a session, and that they know this. A minor downside of paper-based instructions is that it is difficult to ensure that all instructions are returned at the end of a session[5]. A bigger challenge is that it makes instructions that differ by treatment group hard to implement. Indeed it is usually desirable that subjects are unaware of what the research question of the experiment is to avoid experimenter demand (non-deceptive obfuscation in the words of Zizzo (2010)), and that subjects are not in a position to infer what the differences in treatments are.[6]. To provide maximal anonymity subjects should also not be able to tell who the other subjects in their treatment group are. To address these practical challenges oTree provides instructions that have a paper-like look (figure 6). Typically they will be displayed by themselves on a first 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.

3.8 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

[5] Subjects in later sessions might otherwise see them prior to the experiment.

[6] See Bardsley et al. 2010, fn.39

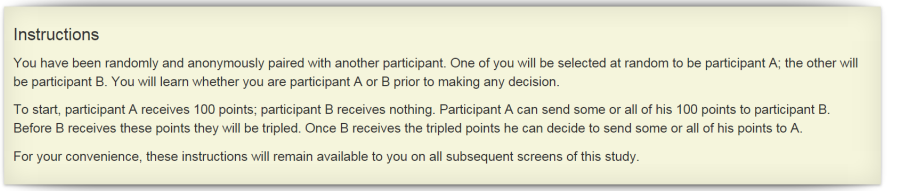
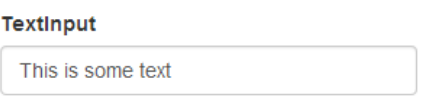


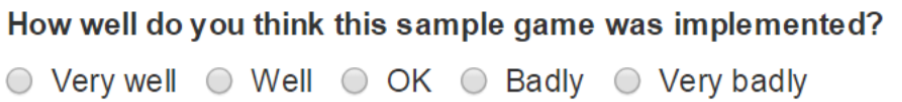
Fig. 6: Paper-like instructions

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 have, and can create their own input forms. Figure 7 gives examples of input forms available.

(a) Dropdown menu



(b) Free text input



(c) Likert scale

Fig. 7: Examples of user input forms

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 in addition specify that only numbers that are a negative odd integer between -51 and -1 should be valid. Among valid answers some answers might be better or more rational than others but the user can proceed with that 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.

3.9 Customizable randomization and matching

Participants in a session are usually partitioned into groups. First, there are treatment groups that allows participants playing concurrently to be exposed

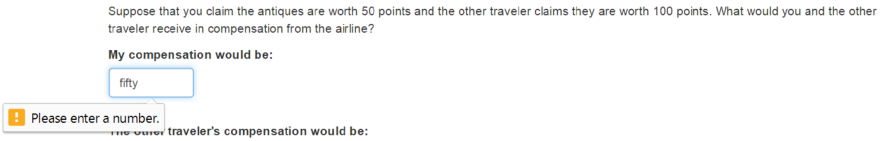


Fig. 8: Input validation

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 using matrices and standard matrix operations.

3.10 Dynamic graphics

In many experiments it is useful to have visual content created and updated during the experiment. For example in a market game participants could be shown a graphic with the evolution of prices. For this purpose oTree seamlessly works with HighCharts[7]. Figure 9 is a screenshot taken from round six of a 50-round game. .

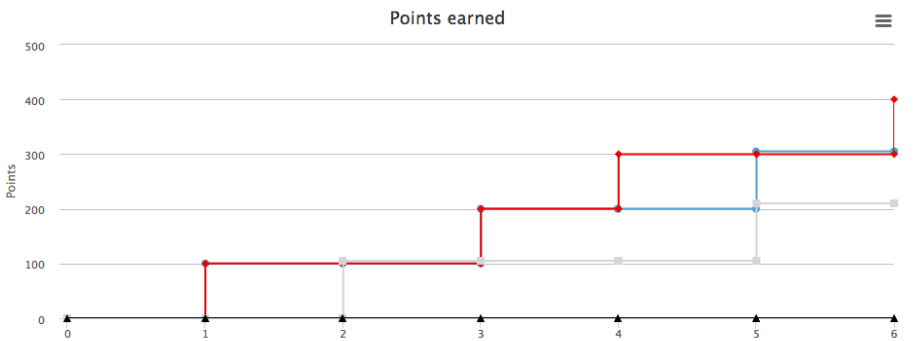


Fig. 9: Dynamic graphics

[7] www.highcharts.com

3.11 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/. You can play these games without any installation just from your web browser. 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 new games from scratch are also encouraged to review the sample games to learn about the recommended coding style.

3.12 Localization

Of course oTree applications can be translated to multiple languages. oTree has multiple features to ease translation and localization. A single switch controls 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 screen shots in a paper reporting the experiment. oTree is able to display monetary amounts in various currency formats (e.g. "$5.00" vs "€5,00").

3.13 Debug info

Any app can be run in a way such that debug info is displayed on the bottom of all screens. The debug info consists of the ID in group, the group, the player, the participant label and the session code. The session code and participant labels are two randomly generated alphanumeric code uniquely identifying a session respectively participant. The ID in group identifies the role of the player (for example in a principal-agent game principals might have the ID in group 1 while agents have the 2).

3.14 Data recording, export and auto-documentation

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. Apart from 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 IP address, and the current page of each participant or whether he is finished.

oTree auto-generates a human readable documentation from the app'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 values the documentation

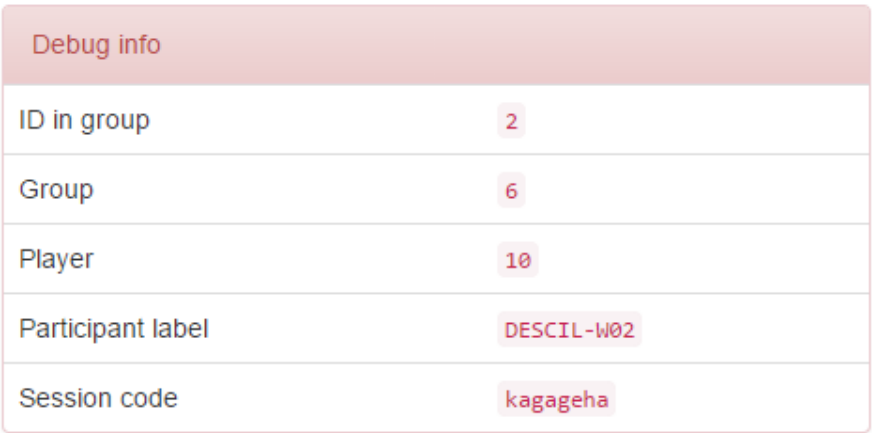


Fig. 10: Debug info

will print that list showing both the internal name and the displayed name the participant sees. As most programmers comment their code anyways, oTree intelligently extracts these comments and adds them to the documentation. Figure 3.14 shows a snippet of auto-documentation for a Likert scale asking participants about their experience in the experiment. Here user satisfaction is measured as a positive integer as the type field tells us, doc is the comment the programmer added in the code, and choices gives the list of options the user sees (here in German) as well as the numeric representation.

```
feedback
   type
      positive integer
   doc
      Likert scale
   choices
      5: Sehr gut
      4: Gut
      3: OK
      2: Schlecht
      1: Sehr schlecht
```

3.15 Open source and replicability

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/licen
Lerner and Tirole (2005) argue that open source software will grow strongly, and that academic economists could both learn and benefit from the model. To develop oTree we use GitHub, a web service used by many open source projects which offers source code management and distributed revision con-

trol. 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 of the code by running two commands. 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. To contribute to the source code itself, the best way to do so, is to open a GitHub account and to get in touch with us.

4 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 which comes with dozens of sample games and consult the wiki at oTree.org/wiki. oTree's programming model follows 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 app 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 type, 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.

4.1 Skills required to program in oTree

oTree is based on Python, a wide-spread programming language which is itself available as open source software from the Python Software Foundation. Python is often the first language taught in introductory computer sciences courses[8]. 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 right away, and program her first experiment within hours.

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

4.2 Hardware requirements

On participants' computers all that is required 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 limitations operating systems impose, for example Flash does not run on iOS devices. A good recommendation for oTree experiments as for any website would be to use JavaScript if necessary, but not Java or Flash. For the server computer requirements depend on the number of participants and the experiment. Often a standard laptop will do. If reliable internet is available the easiest solution is often to rent a server in the cloud. This makes particular sense if the experiment is run online or has high demands in terms of graphics. Prior to an experimental session it is a good idea to test the server using bots, where one should use more bots than the maximum number of subjects planned for the session to be on the safe side.

4.3 Front-end

Since working directly with HTML5 for the user interface can be cumbersome, oTree uses a free, standard front-ed framework for websites called Bootstrap[9]. Bootstrap originates from the 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 be 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).

4.4 Simple code

oTree's code is remarkably simple as it builds on Python. Compared to z-Tree it 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 by now become easy using modern programming languages. To give an idea of the simplicity of oTree consider the standard experimental requirement of paying a random subset of rounds in a game with many rounds. In oTree this requires three lines of code:

```
if self.subsession.round_number == Constants.number_of_rounds:
```

[9] www.getbootstrap.com

```
random_players = random.sample(self.me_in_all_rounds(), 3)
self.payoff = sum([p.theoretical_payoff for p in random_players])
```

To solve this problem in z-Tree Bausch (2012) uses a list traversal algorithm and ends up with about two dozen lines of code.

4.5 Programming assistance

oTree works with standard code editors such as PyCharm[10]. 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, thus for example suggestions of allowed variable names are suggested as code is typed (figure 11).

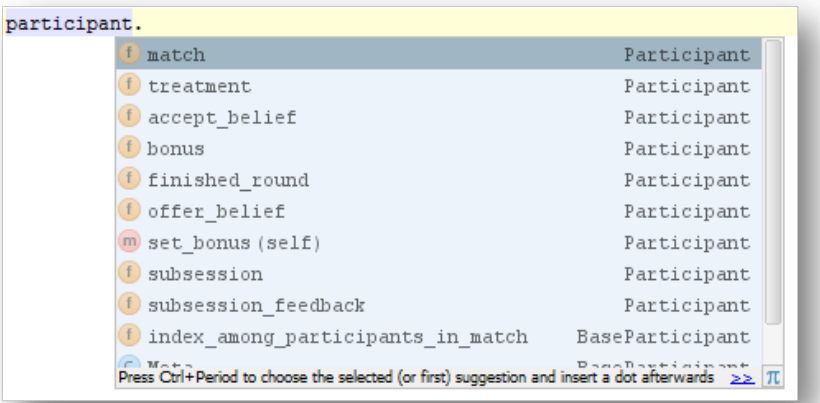


Fig. 11: Autocompletion while writing code

If oTree encounters an error during execution, an error page pinpointing the source of the error is shown. For example in figure 12 a division by zero error in line 17 is pointed out by oTree.

4.6 Collaborative development

oTree is integrated with Git, a source code versioning system. This encourages good backup and versioning practices, and allows developers to synchro-

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

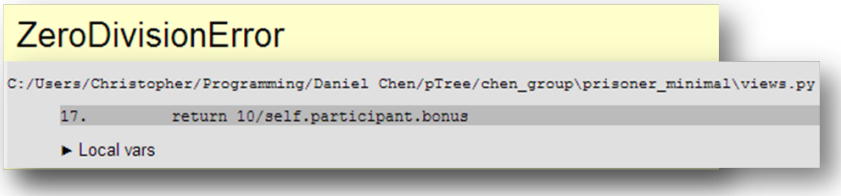


Fig. 12: Informative feedback to programmer

nize files across computers, develop collaboratively, manage separate branches, and merge synchronization conflicts. We invite other researchers to join us on GitHub, but this is no requirement to using oTree.

5 Conclusion

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 operating system independent and deployable on any device that has a web browser, including tablets and smartphones that has a web browser. To connect players with the server and thus each other either an internet connection or alternatively a local area network is necessary. 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 from either the server computer or any other compute 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. Young 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 thus indirectly contribute to foster 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's 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.

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 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 client devices.

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

Angerer, S., D. G.-R. P. Lergetporer, and M. Sutter (2013): "Childrens Cooperation and Discrimination in a Bilingual Province," Working Paper.

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, pp. 391–395.

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

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

Cohn, A., J. Engelmann, E. Fehr, and M. Maréchal (2014): "Evidence for Countercyclical Risk Aversion: An Experiment with Financial Professionals," *American Economic Review*, forthcoming.

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

——— (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, pp.460 ff.

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.

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–.

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

Levitt, S. D. and J. A. List (2008): "Field Experiments in Economics: The Past, the Present, and the Future," Working paper 14356, NBER.

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.

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.

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, pp.2443–2466.

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